AZURE2

Generated by Doxygen 1.9.7

Chapter 1

AZURE2

Thank you for choosing AZURE2 to for all your R-Matrix needs.

This file contains a brief description of how to compile the AZURE2 package.

1.1 Dependencies

CMake

AZURE2 is compiled using the CMake package. Version 2.8 or higher of CMake is (probably) required to build AZURE2. CMake can be downloaded from `http://www.cmake.org`.

GNU Scientific Library

Additionally, much of the mathematics in AZURE2 uses the GSL routines. This library can be obtained from `http://www.gnu.org/software/gsl/`.

ROOT, MINUIT2, and OpenMP

The minimization routines utilized by AZURE are from the Minuit2 package, distributed as part of the ROOT distribution. If ROOT is compiled from source, the –enable-minuit2 flag must be set when running the configure script. The libraries are available as a stand-alone package from `http://seal.web.cern.ch/seal/snapshot/work-packages/mathlibs/minuit/release/download.html`. It is important to note that the ROOT library DOES NOT build Minuit2 with OpenMP support by default, while the stand-alone version does. On a multi-core machine the fit process will be much slower without OpenMP. To build ROOT with OpenMP support for Minuit2, set the enviromnet variables USE_PARALLEL_MINUIT2 and USE_OPENMP prior to building.

Readline Development Libraries

If not already available on your system, you can obtain the readline development libraries via your package manager. For instance, on ubuntu you can run:

```
sudo apt-get install libreadline-dev
```

Qt4.X

The final compile time dependency is Qt. Qt is a cross-platform interface API, and required to compile the graphical setup program. While AZURE2 can be compiled without the graphical setup program, this is not recommended. Qt is available from `http://qt.nokia.com`. Qt 4.4 or greater is required to build AZURE2.

Qt4 can also be obtained via package managers of some Linux distributions. For instance, on ubuntu you can simply run:

```
 bash
sudo apt-get install qt4-dev-tools
```

1.2 Build

1.2.1 Basic Building

The following steps should be performed to build the AZURE2 package:

1. Create a subdirectory of the AZURE2 root named build [mkdir build], and change to that directory [cd build].
2. Run CMake to generate Makefiles from AZURE2 root [cmake ..]. The reference to the parent directory tells CMake where the root of the source tree is located.
3. Build the package [make && make install]. The resulting binary will be created in the current directory.

Alternatively, run the *build.sh* script.

1.2.2 Options

Compiling options (i.e. switching compilers, etc.) are available with flags to CMake. See the CMake documentation for more details.

A few options are available to the user when building AZURE2. The can be passed to cmake using the -D[OPTION]=[VALUE] syntax. CMake also provides a utility to switch these options ON/OFF. After an initial configuration of the build directory (step 2 above), the command [ccmake ..] can be run to view and edit the configured options.

These are:

BUILD_GUI [ON/OFF] - Toggles whether the graphical setup utility is built and linked to AZURE2. This is ON by default, and it is not recommended to turn it OFF.

USE_QWT [ON/OFF] - Toggles whether the built-in plotting tab is added to AZURE2. This is OFF by default. The plotting features are recommended but require the additional QWT libraries to be installed on the build system.

USE_STAT [ON/OFF] - Toggles whether the stat() function should be used to test for properly set directories at start. This only applied when running AZURE2 in console (–no-gui) mode, and is ON by default. This function has been seen to not work properly for Windows, and it is recommended to select OFF if building on/for a Windows system.

MINUIT_PATH [dir] - If Minuit2 is in a non-standard path, this will add the directory to the search path.

GSL_PATH [dir] - If GSL is in a non-standard path, this will add the directory to the search path.

After changing options, execute [make clean] and then step 3 above to build/rebuild AZURE2.

Qt5

Install libqwt-qt5-dev libqt5svg5-dev

Probably only some of the below are required...

Install qt5-default qtscript5-dev

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 ROOT Namespace Reference

Namespaces

- namespace Minuit2

6.2 ROOT::Minuit2 Namespace Reference

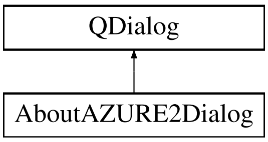
Chapter 7

Class Documentation

7.1 AboutAZURE2Dialog Class Reference

```
#include <AboutAZURE2Dialog.h>
```

Inheritance diagram for AboutAZURE2Dialog:



Public Member Functions

- AboutAZURE2Dialog (QWidget ∗parent=0)

7.1.1 Detailed Description

Definition at line 9 of file AboutAZURE2Dialog.h.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 AboutAZURE2Dialog()

```
AboutAZURE2Dialog::AboutAZURE2Dialog (
            QWidget * parent = 0 )
```

Definition at line 7 of file AboutAZURE2Dialog.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AboutAZURE2Dialog.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AboutAZURE2Dialog.cpp

7.2 AChannel Class Reference

An AZURE channel object.

```
#include <AChannel.h>
```

Public Member Functions

- AChannel (NucLine, int)
- AChannel (int, double, int, char)
- int GetPairNum () const
- int GetL () const
- double GetS () const
- double GetBoundaryCondition () const
- char GetRadType () const
- void SetBoundaryCondition (double)

7.2.1 Detailed Description

An AZURE channel object.

An R-Matrix channel for a given J^{π} group represents a specfic combination of α, s, l couplings.

Definition at line 12 of file AChannel.h.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 AChannel() [1/2]

```
AChannel::AChannel (
            NucLine nucLine,
            int pairNum )
```

This constructor can be used if a channel is to be created from a specific line of the nuclear input file.

Definition at line 11 of file AChannel.cpp.

7.2.2.2 AChannel() [2/2]

```
AChannel::AChannel (
            int lValue,
            double sValue,
            int pairNum,
            char radType )
```

This constructor can be used if a channel is to be created directly using specified channel couplings.

Definition at line 30 of file AChannel.cpp.

7.2.3 Member Function Documentation

7.2.3.1 GetBoundaryCondition()

```
double AChannel::GetBoundaryCondition ( ) const
```

Returns the boundary condition for the channel. The energy of the boundary condition is fixed at the first level given in the nuclear input file for a given J^π group.

Definition at line 65 of file AChannel.cpp.

7.2.3.2 GetL()

```
int AChannel::GetL ( ) const
```

Returns the orbital angular momentum for the channel.

Definition at line 49 of file AChannel.cpp.

7.2.3.3 GetPairNum()

```
int AChannel::GetPairNum ( ) const
```

Returns the pair number, or position of the corresponding particle pair in the PPair vector, for the channel.

Definition at line 41 of file AChannel.cpp.

7.2.3.4 GetRadType()

```
char AChannel::GetRadType ( ) const
```

Returns the radiation type for the channel. Radiation types are:

- P: Particle Radiation
- E: EL Electromagnetic Radiation
- M: ML Electromagnetic Radiation

Definition at line 76 of file AChannel.cpp.

7.2.3.5 GetS()

```
double AChannel::GetS ( ) const
```

Returns the coupled channel spin for the channel.

Definition at line 57 of file AChannel.cpp.

7.2.3.6 SetBoundaryCondition()

```
void AChannel::SetBoundaryCondition (
            double boundaryCondition )
```

This function is used to set the boundary condition for the channel.

Definition at line 83 of file AChannel.cpp.

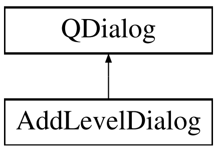
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/AChannel.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/AChannel.cpp

7.3 AddLevelDialog Class Reference

```
#include <AddLevelDialog.h>
```

Inheritance diagram for AddLevelDialog:



Public Member Functions

- AddLevelDialog (QWidget ∗parent=0)

Public Attributes

- QLineEdit ∗ jValueText
- QComboBox ∗ piValueCombo
- QLineEdit ∗ energyText

7.3.1 Detailed Description

Definition at line 14 of file AddLevelDialog.h.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 AddLevelDialog()

```
AddLevelDialog::AddLevelDialog (
            QWidget * parent = 0 )
```

Definition at line 3 of file AddLevelDialog.cpp.

7.3.3 Member Data Documentation

7.3.3.1 energyText

`QLineEdit* AddLevelDialog::energyText`

Definition at line 21 of file AddLevelDialog.h.

7.3.3.2 jValueText

`QLineEdit* AddLevelDialog::jValueText`

Definition at line 19 of file AddLevelDialog.h.

7.3.3.3 piValueCombo

`QComboBox* AddLevelDialog::piValueCombo`

Definition at line 20 of file AddLevelDialog.h.

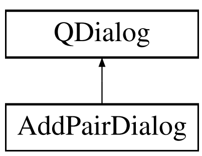
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddLevelDialog.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddLevelDialog.cpp

7.4 AddPairDialog Class Reference

```
#include <AddPairDialog.h>
```

Inheritance diagram for AddPairDialog:



Public Slots

- void updateLightParticle (int index)

Public Member Functions

- AddPairDialog (QWidget *parent=0)

Public Attributes

- QLineEdit ∗ lightJText
- QComboBox ∗ lightPiCombo
- QLineEdit ∗ lightZText
- QLineEdit ∗ lightMText
- QLineEdit ∗ heavyJText
- QComboBox ∗ heavyPiCombo
- QLineEdit ∗ heavyZText
- QLineEdit ∗ heavyMText
- QLineEdit ∗ excitationEnergyText
- QLineEdit ∗ seperationEnergyText
- QLineEdit ∗ channelRadiusText
- QComboBox ∗ pairTypeCombo
- QCheckBox ∗ e1Check
- QCheckBox ∗ e2Check
- QGroupBox ∗ multBox

7.4.1 Detailed Description

Definition at line 17 of file AddPairDialog.h.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 AddPairDialog()

```
AddPairDialog::AddPairDialog (
            QWidget * parent = 0 )
```

Definition at line 13 of file AddPairDialog.cpp.

7.4.3 Member Function Documentation

7.4.3.1 updateLightParticle

```
void AddPairDialog::updateLightParticle (
            int index )  [slot]
```

Definition at line 151 of file AddPairDialog.cpp.

7.4.4 Member Data Documentation

7.4.4.1 channelRadiusText

```
QLineEdit* AddPairDialog::channelRadiusText
```

Definition at line 34 of file AddPairDialog.h.

7.4.4.2 e1Check

`QCheckBox* AddPairDialog::e1Check`

Definition at line 36 of file AddPairDialog.h.

7.4.4.3 e2Check

`QCheckBox* AddPairDialog::e2Check`

Definition at line 38 of file AddPairDialog.h.

7.4.4.4 excitationEnergyText

`QLineEdit* AddPairDialog::excitationEnergyText`

Definition at line 32 of file AddPairDialog.h.

7.4.4.5 heavyJText

`QLineEdit* AddPairDialog::heavyJText`

Definition at line 27 of file AddPairDialog.h.

7.4.4.6 heavyMText

`QLineEdit* AddPairDialog::heavyMText`

Definition at line 30 of file AddPairDialog.h.

7.4.4.7 heavyPiCombo

`QComboBox* AddPairDialog::heavyPiCombo`

Definition at line 28 of file AddPairDialog.h.

7.4.4.8 heavyZText

`QLineEdit* AddPairDialog::heavyZText`

Definition at line 29 of file AddPairDialog.h.

7.4.4.9 lightJText

`QLineEdit* AddPairDialog::lightJText`

Definition at line 22 of file AddPairDialog.h.

7.4.4.10 lightMText

```
QLineEdit* AddPairDialog::lightMText
```

Definition at line 25 of file AddPairDialog.h.

7.4.4.11 lightPiCombo

```
QComboBox* AddPairDialog::lightPiCombo
```

Definition at line 23 of file AddPairDialog.h.

7.4.4.12 lightZText

```
QLineEdit* AddPairDialog::lightZText
```

Definition at line 24 of file AddPairDialog.h.

7.4.4.13 multBox

```
QGroupBox* AddPairDialog::multBox
```

Definition at line 39 of file AddPairDialog.h.

7.4.4.14 pairTypeCombo

```
QComboBox* AddPairDialog::pairTypeCombo
```

Definition at line 35 of file AddPairDialog.h.

7.4.4.15 seperationEnergyText

```
QLineEdit* AddPairDialog::seperationEnergyText
```

Definition at line 33 of file AddPairDialog.h.

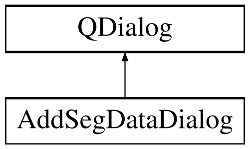
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddPairDialog.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddPairDialog.cpp

7.5 AddSegDataDialog Class Reference

```
#include <AddSegDataDialog.h>
```

Inheritance diagram for AddSegDataDialog:



Public Slots

- void setChooseFile ()
- void dataTypeChanged (int)
- void varyNormChanged (int)

Public Member Functions

- AddSegDataDialog (QWidget *parent=0)

Public Attributes

- QSpinBox * entrancePairIndexSpin
- QSpinBox * exitPairIndexSpin
- QLineEdit * lowEnergyText
- QLineEdit * highEnergyText
- QLineEdit * lowAngleText
- QLineEdit * highAngleText
- QComboBox * dataTypeCombo
- QLineEdit * dataFileText
- QLineEdit * dataNormText
- QLineEdit * dataNormErrorText
- QLabel * dataNormErrorLabel
- QCheckBox * varyNormCheck
- QLineEdit * phaseJValueText
- QLineEdit * phaseLValueText
- QLabel * phaseLValueLabel
- QLabel * phaseJValueLabel
- QLabel * totalCaptureLabel

7.5.1 Detailed Description

Definition at line 17 of file AddSegDataDialog.h.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 AddSegDataDialog()

```
AddSegDataDialog::AddSegDataDialog (
            QWidget * parent = 0 )
```

Definition at line 8 of file AddSegDataDialog.cpp.

7.5.3 Member Function Documentation

7.5.3.1 dataTypeChanged

```
void AddSegDataDialog::dataTypeChanged (
            int index )  [slot]
```

Definition at line 154 of file AddSegDataDialog.cpp.

7.5.3.2 setChooseFile

```
void AddSegDataDialog::setChooseFile ( )  [slot]
```

Definition at line 147 of file AddSegDataDialog.cpp.

7.5.3.3 varyNormChanged

```
void AddSegDataDialog::varyNormChanged (
            int state )  [slot]
```

Definition at line 184 of file AddSegDataDialog.cpp.

7.5.4 Member Data Documentation

7.5.4.1 dataFileText

```
QLineEdit* AddSegDataDialog::dataFileText
```

Definition at line 29 of file AddSegDataDialog.h.

7.5.4.2 dataNormErrorLabel

```
QLabel* AddSegDataDialog::dataNormErrorLabel
```

Definition at line 32 of file AddSegDataDialog.h.

7.5.4.3 dataNormErrorText

`QLineEdit* AddSegDataDialog::dataNormErrorText`

Definition at line 31 of file AddSegDataDialog.h.

7.5.4.4 dataNormText

`QLineEdit* AddSegDataDialog::dataNormText`

Definition at line 30 of file AddSegDataDialog.h.

7.5.4.5 dataTypeCombo

`QComboBox* AddSegDataDialog::dataTypeCombo`

Definition at line 28 of file AddSegDataDialog.h.

7.5.4.6 entrancePairIndexSpin

`QSpinBox* AddSegDataDialog::entrancePairIndexSpin`

Definition at line 22 of file AddSegDataDialog.h.

7.5.4.7 exitPairIndexSpin

`QSpinBox* AddSegDataDialog::exitPairIndexSpin`

Definition at line 23 of file AddSegDataDialog.h.

7.5.4.8 highAngleText

`QLineEdit* AddSegDataDialog::highAngleText`

Definition at line 27 of file AddSegDataDialog.h.

7.5.4.9 highEnergyText

`QLineEdit* AddSegDataDialog::highEnergyText`

Definition at line 25 of file AddSegDataDialog.h.

7.5.4.10 lowAngleText

`QLineEdit* AddSegDataDialog::lowAngleText`

Definition at line 26 of file AddSegDataDialog.h.

7.5.4.11 lowEnergyText

`QLineEdit* AddSegDataDialog::lowEnergyText`

Definition at line 24 of file AddSegDataDialog.h.

7.5.4.12 phaseJValueLabel

`QLabel* AddSegDataDialog::phaseJValueLabel`

Definition at line 37 of file AddSegDataDialog.h.

7.5.4.13 phaseJValueText

`QLineEdit* AddSegDataDialog::phaseJValueText`

Definition at line 34 of file AddSegDataDialog.h.

7.5.4.14 phaseLValueLabel

`QLabel* AddSegDataDialog::phaseLValueLabel`

Definition at line 36 of file AddSegDataDialog.h.

7.5.4.15 phaseLValueText

`QLineEdit* AddSegDataDialog::phaseLValueText`

Definition at line 35 of file AddSegDataDialog.h.

7.5.4.16 totalCaptureLabel

`QLabel* AddSegDataDialog::totalCaptureLabel`

Definition at line 38 of file AddSegDataDialog.h.

7.5.4.17 varyNormCheck

`QCheckBox* AddSegDataDialog::varyNormCheck`

Definition at line 33 of file AddSegDataDialog.h.

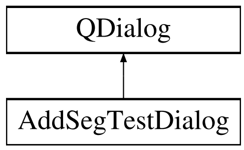
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddSegDataDialog.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddSegDataDialog.cpp

7.6 AddSegTestDialog Class Reference

```
#include <AddSegTestDialog.h>
```

Inheritance diagram for AddSegTestDialog:



Public Slots

- void dataTypeChanged (int)

Public Member Functions

- AddSegTestDialog (QWidget *parent=0)

Public Attributes

- QSpinBox * entrancePairIndexSpin
- QSpinBox * exitPairIndexSpin
- QLineEdit * lowEnergyText
- QLineEdit * highEnergyText
- QLineEdit * energyStepText
- QLineEdit * lowAngleText
- QLineEdit * highAngleText
- QLineEdit * angleStepText
- QComboBox * dataTypeCombo
- QLineEdit * phaseJValueText
- QLineEdit * phaseLValueText
- QLabel * phaseJValueLabel
- QLabel * phaseLValueLabel
- QLabel * angDistLabel
- QLabel * totalCaptureLabel
- QSpinBox * angDistSpin

7.6.1 Detailed Description

Definition at line 16 of file AddSegTestDialog.h.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 AddSegTestDialog()

```
AddSegTestDialog::AddSegTestDialog (
            QWidget * parent = 0 )
```

Definition at line 9 of file AddSegTestDialog.cpp.

7.6.3 Member Function Documentation

7.6.3.1 dataTypeChanged

```
void AddSegTestDialog::dataTypeChanged (
             int index )  [slot]
```

Definition at line 135 of file AddSegTestDialog.cpp.

7.6.4 Member Data Documentation

7.6.4.1 angDistLabel

```
QLabel* AddSegTestDialog::angDistLabel
```

Definition at line 34 of file AddSegTestDialog.h.

7.6.4.2 angDistSpin

```
QSpinBox* AddSegTestDialog::angDistSpin
```

Definition at line 36 of file AddSegTestDialog.h.

7.6.4.3 angleStepText

```
QLineEdit* AddSegTestDialog::angleStepText
```

Definition at line 28 of file AddSegTestDialog.h.

7.6.4.4 dataTypeCombo

```
QComboBox* AddSegTestDialog::dataTypeCombo
```

Definition at line 29 of file AddSegTestDialog.h.

7.6.4.5 energyStepText

```
QLineEdit* AddSegTestDialog::energyStepText
```

Definition at line 25 of file AddSegTestDialog.h.

7.6.4.6 entrancePairIndexSpin

```
QSpinBox* AddSegTestDialog::entrancePairIndexSpin
```

Definition at line 21 of file AddSegTestDialog.h.

7.6.4.7 exitPairIndexSpin

`QSpinBox* AddSegTestDialog::exitPairIndexSpin`

Definition at line 22 of file AddSegTestDialog.h.

7.6.4.8 highAngleText

`QLineEdit* AddSegTestDialog::highAngleText`

Definition at line 27 of file AddSegTestDialog.h.

7.6.4.9 highEnergyText

`QLineEdit* AddSegTestDialog::highEnergyText`

Definition at line 24 of file AddSegTestDialog.h.

7.6.4.10 lowAngleText

`QLineEdit* AddSegTestDialog::lowAngleText`

Definition at line 26 of file AddSegTestDialog.h.

7.6.4.11 lowEnergyText

`QLineEdit* AddSegTestDialog::lowEnergyText`

Definition at line 23 of file AddSegTestDialog.h.

7.6.4.12 phaseJValueLabel

`QLabel* AddSegTestDialog::phaseJValueLabel`

Definition at line 32 of file AddSegTestDialog.h.

7.6.4.13 phaseJValueText

`QLineEdit* AddSegTestDialog::phaseJValueText`

Definition at line 30 of file AddSegTestDialog.h.

7.6.4.14 phaseLValueLabel

`QLabel* AddSegTestDialog::phaseLValueLabel`

Definition at line 33 of file AddSegTestDialog.h.

7.6.4.15 phaseLValueText

`QLineEdit* AddSegTestDialog::phaseLValueText`

Definition at line 31 of file AddSegTestDialog.h.

7.6.4.16 totalCaptureLabel

`QLabel* AddSegTestDialog::totalCaptureLabel`

Definition at line 35 of file AddSegTestDialog.h.

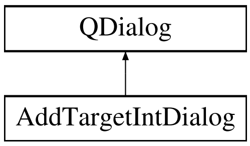
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddSegTestDialog.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddSegTestDialog.cpp

7.7 AddTargetIntDialog Class Reference

`#include <AddTargetIntDialog.h>`

Inheritance diagram for AddTargetIntDialog:



Public Slots

- void convolutionCheckChanged (bool checked)
- void targetIntCheckChanged (bool checked)
- void parameterSpinChanged (int newNumber)
- void parameterChanged (int row, int column)
- void qCoefficientCheckChanged (bool checked)
- void qCoefficientSpinChanged (int newNumber)
- void qCoefficientChanged (int row, int column)

Public Member Functions

- AddTargetIntDialog (QWidget *parent=0)
- void createParameterItem (int row, double value=0.0)
- void createQCoefficientItem (int row, double value=1.0)

Public Attributes

- QCheckBox ∗ isConvolutionCheck
- QCheckBox ∗ isTargetIntegrationCheck
- QCheckBox ∗ isQCoefficientCheck
- QLineEdit ∗ sigmaText
- QLineEdit ∗ segmentsListText
- QSpinBox ∗ numPointsSpin
- QSpinBox ∗ numParametersSpin
- QSpinBox ∗ numQCoefficientSpin
- QLineEdit ∗ densityText
- QLineEdit ∗ stoppingPowerEqText
- QTableWidget ∗ parametersTable
- QTableWidget ∗ qCoefficientTable
- QList< double > tempParameters
- QList< double > tempQCoefficients

7.7.1 Detailed Description

Definition at line 17 of file AddTargetIntDialog.h.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 AddTargetIntDialog()

```
AddTargetIntDialog::AddTargetIntDialog (
            QWidget * parent = 0 )
```

Definition at line 14 of file AddTargetIntDialog.cpp.

7.7.3 Member Function Documentation

7.7.3.1 convolutionCheckChanged

```
void AddTargetIntDialog::convolutionCheckChanged (
            bool checked )  [slot]
```

Definition at line 175 of file AddTargetIntDialog.cpp.

7.7.3.2 createParameterItem()

```
void AddTargetIntDialog::createParameterItem (
            int row,
            double value = 0.0 )
```

Definition at line 153 of file AddTargetIntDialog.cpp.

7.7.3.3 createQCoefficientItem()

```
void AddTargetIntDialog::createQCoefficientItem (
            int row,
            double value = 1.0 )
```

Definition at line 164 of file AddTargetIntDialog.cpp.

7.7.3.4 parameterChanged

```
void AddTargetIntDialog::parameterChanged (
            int row,
            int column ) [slot]
```

Definition at line 209 of file AddTargetIntDialog.cpp.

7.7.3.5 parameterSpinChanged

```
void AddTargetIntDialog::parameterSpinChanged (
            int newNumber ) [slot]
```

Definition at line 198 of file AddTargetIntDialog.cpp.

7.7.3.6 qCoefficientChanged

```
void AddTargetIntDialog::qCoefficientChanged (
            int row,
            int column ) [slot]
```

Definition at line 241 of file AddTargetIntDialog.cpp.

7.7.3.7 qCoefficientCheckChanged

```
void AddTargetIntDialog::qCoefficientCheckChanged (
            bool checked ) [slot]
```

Definition at line 219 of file AddTargetIntDialog.cpp.

7.7.3.8 qCoefficientSpinChanged

```
void AddTargetIntDialog::qCoefficientSpinChanged (
            int newNumber ) [slot]
```

Definition at line 230 of file AddTargetIntDialog.cpp.

7.7.3.9 targetIntCheckChanged

```
void AddTargetIntDialog::targetIntCheckChanged (
            bool checked )  [slot]
```

Definition at line 185 of file AddTargetIntDialog.cpp.

7.7.4 Member Data Documentation

7.7.4.1 densityText

```
QLineEdit* AddTargetIntDialog::densityText
```

Definition at line 30 of file AddTargetIntDialog.h.

7.7.4.2 isConvolutionCheck

```
QCheckBox* AddTargetIntDialog::isConvolutionCheck
```

Definition at line 22 of file AddTargetIntDialog.h.

7.7.4.3 isQCoefficientCheck

```
QCheckBox* AddTargetIntDialog::isQCoefficientCheck
```

Definition at line 24 of file AddTargetIntDialog.h.

7.7.4.4 isTargetIntegrationCheck

```
QCheckBox* AddTargetIntDialog::isTargetIntegrationCheck
```

Definition at line 23 of file AddTargetIntDialog.h.

7.7.4.5 numParametersSpin

```
QSpinBox* AddTargetIntDialog::numParametersSpin
```

Definition at line 28 of file AddTargetIntDialog.h.

7.7.4.6 numPointsSpin

```
QSpinBox* AddTargetIntDialog::numPointsSpin
```

Definition at line 27 of file AddTargetIntDialog.h.

7.7.4.7 numQCoefficientSpin

`QSpinBox* AddTargetIntDialog::numQCoefficientSpin`

Definition at line 29 of file AddTargetIntDialog.h.

7.7.4.8 parametersTable

`QTableWidget* AddTargetIntDialog::parametersTable`

Definition at line 32 of file AddTargetIntDialog.h.

7.7.4.9 qCoefficientTable

`QTableWidget* AddTargetIntDialog::qCoefficientTable`

Definition at line 33 of file AddTargetIntDialog.h.

7.7.4.10 segmentsListText

`QLineEdit* AddTargetIntDialog::segmentsListText`

Definition at line 26 of file AddTargetIntDialog.h.

7.7.4.11 sigmaText

`QLineEdit* AddTargetIntDialog::sigmaText`

Definition at line 25 of file AddTargetIntDialog.h.

7.7.4.12 stoppingPowerEqText

`QLineEdit* AddTargetIntDialog::stoppingPowerEqText`

Definition at line 31 of file AddTargetIntDialog.h.

7.7.4.13 tempParameters

`QList<double> AddTargetIntDialog::tempParameters`

Definition at line 34 of file AddTargetIntDialog.h.

7.7.4.14 tempQCoefficients

```
QList<double> AddTargetIntDialog::tempQCoefficients
```

Definition at line 35 of file AddTargetIntDialog.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddTargetIntDialog.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddTargetIntDialog.cpp

7.8 ALevel Class Reference

An AZURE level object.

```
#include <ALevel.h>
```

Public Member Functions

- ALevel (NucLine)
- ALevel (double)
- bool IsInRMatrix () const
- bool EnergyFixed () const
- bool ChannelFixed (int) const
- bool IsECLevel () const
- int NumNFIntegrals () const
- int GetTransformIterations () const
- int GetECPairNum () const
- unsigned char GetECMultMask () const
- double GetE () const
- double GetGamma (int) const
- double GetFitGamma (int) const
- double GetFitE () const
- double GetNFIntegral (int) const
- double GetSqrtNFFactor () const
- double GetECConversionFactor (int) const
- double GetTransformGamma (int) const
- double GetTransformE () const
- double GetBigGamma (int) const
- double GetShiftFunction (int) const
- complex GetExternalGamma (int) const
- void AddGamma (NucLine)
- void AddGamma (double)
- void SetGamma (int, double)
- void SetE (double)
- void SetFitGamma (int, double)
- void SetFitE (double)
- void AddNFIntegral (double)
- void SetSqrtNFFactor (double)
- void AddECConversionFactor (double)
- void SetTransformGamma (int, double)
- void SetTransformE (double)
- void SetBigGamma (int, double)
- void SetTransformIterations (int)
- void SetExternalGamma (int, complex)
- void SetShiftFunction (int, double)
- void SetECParams (int, unsigned char)

7.8.1 Detailed Description

An AZURE level object.

An R-matrix level represents a specific eigenstate of the compound nucleus.

Definition at line 14 of file ALevel.h.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 ALevel() [1/2]

```
ALevel::ALevel (
            NucLine nucLine )
```

This constructor is used when a level object is created from an entry in the nuclear file.

Definition at line 8 of file ALevel.cpp.

7.8.2.2 ALevel() [2/2]

```
ALevel::ALevel (
            double energy )
```

This constructor is used when a level object is created using a specific energy.

Definition at line 19 of file ALevel.cpp.

7.8.3 Member Function Documentation

7.8.3.1 AddECConversionFactor()

```
void ALevel::AddECConversionFactor (
            double conversionFactor )
```

This function adds a conversion factor from reduced width amplitude to ANC.

Definition at line 271 of file ALevel.cpp.

7.8.3.2 AddGamma() [1/2]

```
void ALevel::AddGamma (
            double reducedWidth )
```

This function adds a position in the width vectors corresponding to a new channel.
The initial reduced width amplitude is set directly.

Definition at line 208 of file ALevel.cpp.

7.8.3.3 AddGamma() [2/2]

```
void ALevel::AddGamma (
            NucLine nucLine )
```

This function adds a position in the width vectors corresponding to a new channel.
The initial reduced width amplitude is set from an entry in the nuclear input file.

Definition at line 191 of file ALevel.cpp.

7.8.3.4 AddNFIntegral()

```
void ALevel::AddNFIntegral (
            double integral )
```

This function creates and fills a position for the channel integral in the denominator of the $N_f^{1/2}$ term. The integral is of the form $\int_a^\infty \left[\frac{W_c(kr)}{W_c k a_c}\right]^2$.

Definition at line 255 of file ALevel.cpp.

7.8.3.5 ChannelFixed()

```
bool ALevel::ChannelFixed (
            int channelNum ) const
```

Returns true if the reduced width amplitude for corresponding channel number is to be fixed in the fitting process, otherwise returns false.

Definition at line 45 of file ALevel.cpp.

7.8.3.6 EnergyFixed()

```
bool ALevel::EnergyFixed ( ) const
```

Returns true if the level energy is to be fixed in the fitting process, otherwise returns false.

Definition at line 27 of file ALevel.cpp.

7.8.3.7 GetBigGamma()

```
double ALevel::GetBigGamma (
            int channelNum ) const
```

Returns the Breit-Wigner partial width for a given channel number.

Definition at line 166 of file ALevel.cpp.

7.8.3.8 GetE()

```
double ALevel::GetE ( ) const
```

Returns the energy of the level.

Definition at line 93 of file ALevel.cpp.

7.8.3.9 GetECConversionFactor()

```
double ALevel::GetECConversionFactor (
             int channelNum ) const
```

Returns the conversion factor from reduced width amplitude to ANC for a given channel number.

Definition at line 142 of file ALevel.cpp.

7.8.3.10 GetECMultMask()

```
unsigned char ALevel::GetECMultMask ( ) const
```

Returns the multipolarity mask of external capture gammas to the level.

Definition at line 85 of file ALevel.cpp.

7.8.3.11 GetECPairNum()

```
int ALevel::GetECPairNum ( ) const
```

Returns the position in the pairs vector corresponding the the external capture level.

Definition at line 77 of file ALevel.cpp.

7.8.3.12 GetExternalGamma()

```
complex ALevel::GetExternalGamma (
             int channelNum ) const
```

Returns the external portion of the reduced width amplitude for a given channel number.

Definition at line 182 of file ALevel.cpp.

7.8.3.13 GetFitE()

```
double ALevel::GetFitE ( ) const
```

Returns the fitted energy of the level.

Definition at line 117 of file ALevel.cpp.

7.8.3.14 GetFitGamma()

```
double ALevel::GetFitGamma (
            int channelNum ) const
```

Returns the fitted internal reduced width amplitude for a given channel number.

Definition at line 109 of file ALevel.cpp.

7.8.3.15 GetGamma()

```
double ALevel::GetGamma (
            int channelNum ) const
```

Returns the internal reduced width amplitude for a given channel number.

Definition at line 101 of file ALevel.cpp.

7.8.3.16 GetNFIntegral()

```
double ALevel::GetNFIntegral (
            int channelNum ) const
```

Returns the calculated channel integral in the denominator of the $N_f^{1/2}$ term for a given channel number.

Definition at line 125 of file ALevel.cpp.

7.8.3.17 GetShiftFunction()

```
double ALevel::GetShiftFunction (
            int channelNum ) const
```

Returns the Shift function for the specified channel number calculated at the resonance energy.

Definition at line 174 of file ALevel.cpp.

7.8.3.18 GetSqrtNFFactor()

```
double ALevel::GetSqrtNFFactor ( ) const
```

Returns the $N_f^{1/2}$ term for the level.

Definition at line 133 of file ALevel.cpp.

7.8.3.19 GetTransformE()

```
double ALevel::GetTransformE ( ) const
```

Returns the physical level energy.

Definition at line 158 of file ALevel.cpp.

7.8.3.20 GetTransformGamma()

```
double ALevel::GetTransformGamma (
            int channelNum ) const
```

Returns the physical internal reduced width amplitude for a given channel number.

Definition at line 150 of file ALevel.cpp.

7.8.3.21 GetTransformIterations()

```
int ALevel::GetTransformIterations ( ) const
```

Returns the number of iterations required to transform the level from formal to physical parameters.

Definition at line 69 of file ALevel.cpp.

7.8.3.22 IsECLevel()

```
bool ALevel::IsECLevel ( ) const
```

Returns true if the level is a final state for external capture, otherwise returns false.

Definition at line 53 of file ALevel.cpp.

7.8.3.23 IsInRMatrix()

```
bool ALevel::IsInRMatrix ( ) const
```

Returns true if the level is to be included in the A-/R-Matrix calculation, otherwise returns false. A level may specify a bound state for external capture, but may not be an R-Matrix state (i.e. subthreshold state).

Definition at line 36 of file ALevel.cpp.

7.8.3.24 NumNFIntegrals()

```
int ALevel::NumNFIntegrals ( ) const
```

Returns non-zero only if the level is a final state for external capture.

Definition at line 61 of file ALevel.cpp.

7.8.3.25 SetBigGamma()

```
void ALevel::SetBigGamma (
            int channelNum,
            double partialWidth )
```

This function sets the Breit-Wigner partial width for a given channel number.

Definition at line 295 of file ALevel.cpp.

7.8.3.26 SetE()

```
void ALevel::SetE (
             double energy )
```

This function sets the level energy.

Definition at line 229 of file ALevel.cpp.

7.8.3.27 SetECParams()

```
void ALevel::SetECParams (
             int pairNum,
             unsigned char multMask )
```

Sets the external capture parameters for the level.

Definition at line 327 of file ALevel.cpp.

7.8.3.28 SetExternalGamma()

```
void ALevel::SetExternalGamma (
             int channelNum,
             complex reducedWidth )
```

This function sets the external reduced width amplitude for a given channel number.

Definition at line 311 of file ALevel.cpp.

7.8.3.29 SetFitE()

```
void ALevel::SetFitE (
             double energy )
```

This function sets the fitted level energy.

Definition at line 245 of file ALevel.cpp.

7.8.3.30 SetFitGamma()

```
void ALevel::SetFitGamma (
             int channelNum,
             double reducedWidth )
```

This function sets the fitted internal reduced width amplitude for a given channel number.

Definition at line 237 of file ALevel.cpp.

7.8.3.31 SetGamma()

```
void ALevel::SetGamma (
            int channelNum,
            double reducedWidth )
```

This function sets the internal reduced width amplitude for a given channel number.

Definition at line 221 of file ALevel.cpp.

7.8.3.32 SetShiftFunction()

```
void ALevel::SetShiftFunction (
            int channelNum,
            double shiftFunction )
```

Sets the value of the shift function calculated at the resonance energy.

Definition at line 319 of file ALevel.cpp.

7.8.3.33 SetSqrtNFFactor()

```
void ALevel::SetSqrtNFFactor (
            double term )
```

This function sets the $N_f^{1/2}$ term for the level.

Definition at line 263 of file ALevel.cpp.

7.8.3.34 SetTransformE()

```
void ALevel::SetTransformE (
            double energy )
```

This function sets the physical level energy.

Definition at line 287 of file ALevel.cpp.

7.8.3.35 SetTransformGamma()

```
void ALevel::SetTransformGamma (
            int channelNum,
            double reducedWidth )
```

This function sets the physical reduced width amplitude for a given channel number.

Definition at line 279 of file ALevel.cpp.

7.8.3.36 SetTransformIterations()

```
void ALevel::SetTransformIterations (
            int iterations )
```

This function sets the number of iterations that were required for the transformation from formal to physical parameters.

Definition at line 303 of file ALevel.cpp.

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

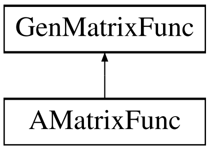
- /Users/kuba/Desktop/R-Matrix/AZURE2/include/ALevel.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/ALevel.cpp

7.9 AMatrixFunc Class Reference

A function class to calculate the T-Matrix using the A-Matrix.

```
#include <AMatrixFunc.h>
```

Inheritance diagram for AMatrixFunc:



Public Member Functions

- AMatrixFunc (CNuc ∗, const Config &configure)
- CNuc ∗ compound () const
- const Config & configure () const
- void ClearMatrices ()
- void FillMatrices (EPoint ∗)
- void InvertMatrices ()
- void CalculateTMatrix (EPoint ∗)
- void CalculateCrossSection ()
- complex GetAMatrixElement (int, int, int) const
- matrix_c ∗ GetJSpecAInvMatrix (int)
- void AddAInvMatrixElement (int, int, int, complex)
- void AddAMatrix (matrix_c)

Public Member Functions inherited from GenMatrixFunc

- GenMatrixFunc ()
- virtual ~GenMatrixFunc ()
- virtual void ClearMatrices ()=0
- virtual void FillMatrices (EPoint *)=0
- virtual void InvertMatrices ()=0
- virtual void CalculateTMatrix (EPoint *)=0
- void CalculateCrossSection (EPoint *)
- void NewTempTMatrix (TempTMatrix)
- void AddToTempTMatrix (int, complex)
- void ClearTempTMatrices ()
- void AddTMatrixElement (int, int, complex, int decayNum=1)
- void AddECTMatrixElement (int, int, complex)
- int IsTempTMatrix (double, int, int)
- int NumTempTMatrices () const
- TempTMatrix * GetTempTMatrix (int)
- complex GetTMatrixElement (int, int, int decayNum=1) const
- complex GetECTMatrixElement (int, int) const
- virtual CNuc * compound () const =0
- virtual const Config & configure () const =0

Additional Inherited Members

Protected Attributes inherited from GenMatrixFunc

- std::vector< matrix_c > tmatrix_

    Vector of internal T-matrix elements accessable to child class.
- matrix_c ec_tmatrix_

    Vector of external T-matrix elements accessable to child class.

7.9.1 Detailed Description

A function class to calculate the T-Matrix using the A-Matrix.

The AMatrixFunc function class calculates the T-Matrix for a given energy point using the compound nucleus object. The AMatrixFunc class is a child class of GenMatrixFunc, where the cross section is calculated from the T-Matrix.

Definition at line 14 of file AMatrixFunc.h.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 AMatrixFunc()

```
AMatrixFunc::AMatrixFunc (
            CNuc * compound,
            const Config & configure )
```

The AMatrixFunc object is created with reference to a CNuc object.

Definition at line 12 of file AMatrixFunc.cpp.

7.9.3 Member Function Documentation

7.9.3.1 AddAInvMatrixElement()

```
void AMatrixFunc::AddAInvMatrixElement (
            int jGroupNum,
            int lambdaNum,
            int muNum,
            complex aMatrixElement )
```

This function adds an inverse A-Matrix element specified by positions in the JGroup and ALevel vectors.

Definition at line 225 of file AMatrixFunc.cpp.

7.9.3.2 AddAMatrix()

```
void AMatrixFunc::AddAMatrix (
            matrix_c aMatrix )
```

This function adds an entire A-Matrix to a vector.

Definition at line 238 of file AMatrixFunc.cpp.

7.9.3.3 CalculateCrossSection()

```
void AMatrixFunc::CalculateCrossSection ( )
```

Instantiated in the parent class.

7.9.3.4 CalculateTMatrix()

```
void AMatrixFunc::CalculateTMatrix (
            EPoint * point ) [virtual]
```

This function calculates the T-Matrix for each reaction pathway based on the A-Matrix.

Implements GenMatrixFunc.

Definition at line 119 of file AMatrixFunc.cpp.

7.9.3.5 ClearMatrices()

```
void AMatrixFunc::ClearMatrices ( ) [virtual]
```

Clears all matrices associated with the AMatrixFunc object.

Implements GenMatrixFunc.

Definition at line 36 of file AMatrixFunc.cpp.

7.9.3.6 compound()

```
CNuc * AMatrixFunc::compound ( ) const [inline], [virtual]
```

Returns a pointer to the compound nucleus object.

Implements GenMatrixFunc.

Definition at line 20 of file AMatrixFunc.h.

7.9.3.7 configure()

```
const Config & AMatrixFunc::configure ( ) const [inline], [virtual]
```

Returns a reference to the Config structure.

Implements GenMatrixFunc.

Definition at line 24 of file AMatrixFunc.h.

7.9.3.8 FillMatrices()

```
void AMatrixFunc::FillMatrices (
            EPoint * point ) [virtual]
```

This function creates the inverted A-Matrix from the parameters in the CNuc object.

Implements GenMatrixFunc.

Definition at line 47 of file AMatrixFunc.cpp.

7.9.3.9 GetAMatrixElement()

```
complex AMatrixFunc::GetAMatrixElement (
            int jGroupNum,
            int lambdaNum,
            int muNum ) const
```

Returns an A-Matrix element specified by positions in the JGroup and ALevel vectors.

Definition at line 19 of file AMatrixFunc.cpp.

7.9.3.10 GetJSpecAInvMatrix()

```
matrix_c * AMatrixFunc::GetJSpecAInvMatrix (
            int jGroupNum )
```

Returns a pointer to an entire A-Matrix specified by a position in the JGroup vector.

Definition at line 27 of file AMatrixFunc.cpp.

7.9.3.11 InvertMatrices()

```
void AMatrixFunc::InvertMatrices ( )  [virtual]
```

This function inverts the inverse A-Matrix to yeild the A-Matrix.

Implements GenMatrixFunc.

Definition at line 105 of file AMatrixFunc.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/AMatrixFunc.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/AMatrixFunc.cpp

7.10 AngCoeff Class Reference

A container class for angular coupling coefficient functions.

```
#include <AngCoeff.h>
```

Static Public Member Functions

- static double ClebGord (double, double, double, double, double, double)
- static double Racah (double, double, double, double, double, double)

7.10.1 Detailed Description

A container class for angular coupling coefficient functions.

The AngCoeff class serves as a container class for the angular momentum coupling coefficients.

Definition at line 11 of file AngCoeff.h.

7.10.2 Member Function Documentation

7.10.2.1 ClebGord()

```
double AngCoeff::ClebGord (
            double j1,
            double j2,
            double j3,
            double m1,
            double m2,
            double m3 )  [static]
```

Returns the Clebsh-Gordan coefficient for the given angular momentum quantum numbers.

Definition at line 5 of file AngCoeff.cpp.

7.10.2.2 Racah()

```
double AngCoeff::Racah (
            double j1,
            double j2,
            double l2,
            double l1,
            double j3,
            double l3 )  [static]
```

Returns the Racah coefficient for the given angular momentum quantum numbers.

Definition at line 19 of file AngCoeff.cpp.

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

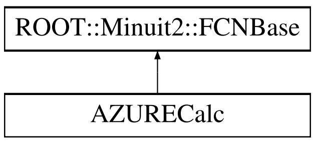
- /Users/kuba/Desktop/R-Matrix/AZURE2/include/AngCoeff.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/AngCoeff.cpp

7.11 AZURECalc Class Reference

A function class to perform the calculation of the chi-squared value.

`#include <AZURECalc.h>`

Inheritance diagram for AZURECalc:



Public Member Functions

- AZURECalc (EData ∗data, CNuc ∗compound, const Config &configure)
- ∼AZURECalc ()
- virtual double Up () const
- virtual double operator() (const vector_r &) const
- const Config & configure () const
- EData ∗ data () const
- CNuc ∗ compound () const
- void SetErrorDef (double def)

7.11.1 Detailed Description

A function class to perform the calculation of the chi-squared value.

The AZURECalc function class calculates the cross section based on a parameter set for all available data, and returns a chi-squared value. This function class is what Minuit calls repeatedly during the fitting process to perform the minimization.

Definition at line 21 of file AZURECalc.h.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 AZURECalc()

```
AZURECalc::AZURECalc (
            EData * data,
            CNuc * compound,
            const Config & configure )  [inline]
```

The AZURECalc object is created with reference to an EData and CNuc object. . The runtime configurations are also passed through a Config structure.

Definition at line 27 of file AZURECalc.h.

7.11.2.2 ∼AZURECalc()

```
AZURECalc::∼AZURECalc ( )  [inline]
```

Definition at line 32 of file AZURECalc.h.

7.11.3 Member Function Documentation

7.11.3.1 compound()

```
CNuc * AZURECalc::compound ( ) const  [inline]
```

Returns a pointer to the CNuc object.

Definition at line 55 of file AZURECalc.h.

7.11.3.2 configure()

```
const Config & AZURECalc::configure ( ) const  [inline]
```

Returns a reference to the Config structure.

Definition at line 47 of file AZURECalc.h.

7.11.3.3 data()

```
EData * AZURECalc::data ( ) const  [inline]
```

Returns a pointer to the EData object.

Definition at line 51 of file AZURECalc.h.

7.11.3.4 operator()()

```
double AZURECalc::operator() (
            const vector_r & p ) const  [virtual]
```

Overloaded operator to make the class instance callable as a function. A Minuit parameter array is passed as the dependent variable. The function returns the total chi-squared value.

Definition at line 8 of file AZURECalc.cpp.

7.11.3.5 SetErrorDef()

```
void AZURECalc::SetErrorDef (
            double def )  [inline]
```

See Minuit2 documentation for an explanation of this function.

Definition at line 60 of file AZURECalc.h.

7.11.3.6 Up()

```
virtual double AZURECalc::Up ( ) const  [inline], [virtual]
```

See Minuit2 documentation for an explanation of this function.

Definition at line 36 of file AZURECalc.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZURECalc.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZURECalc.cpp

7.12 AZUREFBuffer Class Reference

A container class for a pointer to a file buffer.

```
#include <AZUREFBuffer.h>
```

Public Member Functions

- AZUREFBuffer (int entranceKey, int exitKey, std::string outputdir, bool isExtrap, bool isAngDist)
- ∼AZUREFBuffer ()
- bool IsAngDist () const
- int GetEntranceKey () const
- int GetExitKey () const
- std::filebuf ∗ GetFBuffer ()

7.12.1 Detailed Description

A container class for a pointer to a file buffer.

The AZUREFBuffer class contains a pointer to an acutal file buffer, as well as the entrance and exit pair keys to which the file buffer corresponds.

Definition at line 15 of file AZUREFBuffer.h.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 AZUREFBuffer()

```
AZUREFBuffer::AZUREFBuffer (
            int entranceKey,
            int exitKey,
            std::string outputdir,
            bool isExtrap,
            bool isAngDist )  [inline]
```

The AZUREFBuffer object is created with an entrance and exit pair key, as well as an output directory. The filename is determined, and a file buffer is created with that filename.

Definition at line 21 of file AZUREFBuffer.h.

7.12.2.2 ∼AZUREFBuffer()

```
AZUREFBuffer::∼AZUREFBuffer ( )  [inline]
```

The file buffer is closed and destroyed with the instance of AZUREFBuffer.

Definition at line 43 of file AZUREFBuffer.h.

7.12.3 Member Function Documentation

7.12.3.1 GetEntranceKey()

```
int AZUREFBuffer::GetEntranceKey ( ) const  [inline]
```

Returns the entrance pair key of the object.

Definition at line 54 of file AZUREFBuffer.h.

7.12.3.2 GetExitKey()

```
int AZUREFBuffer::GetExitKey ( ) const  [inline]
```

Returns the exit pair key of the object.

Definition at line 58 of file AZUREFBuffer.h.

7.12.3.3 GetFBuffer()

```
std::filebuf * AZUREFBuffer::GetFBuffer ( ) [inline]
```

Returns a pointer to the corresponding file buffer.

Definition at line 62 of file AZUREFBuffer.h.

7.12.3.4 IsAngDist()

```
bool AZUREFBuffer::IsAngDist ( ) const [inline]
```

Returns true if the buffer is for angular distribution, otherwise returns false.

Definition at line 50 of file AZUREFBuffer.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREFBuffer.h

7.13 AZUREMain Class Reference

The top-level AZURE function class.

```
#include <AZUREMain.h>
```

Public Member Functions

- AZUREMain (const Config &configure)
- ∼AZUREMain ()
- int operator() ()
- const Config & configure () const
- CNuc ∗ compound () const
- EData ∗ data () const

7.13.1 Detailed Description

The top-level AZURE function class.

The AZUREMain function class is the top level function class in the AZURE package. It is called directly from them main() using the configuration parameters read from the runtime file as well as from the command shell prompt.

Definition at line 14 of file AZUREMain.h.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 AZUREMain()

```
AZUREMain::AZUREMain (
            const Config & configure )  [inline]
```

The AZUREMain function class is created using a Config structure. New CNuc and EData objects are created at initialization of an AZUREMain object.

Definition at line 20 of file AZUREMain.h.

7.13.2.2 ∼AZUREMain()

```
AZUREMain::∼AZUREMain ( )  [inline]
```

The CNuc and EData objects are destroyed with the AZUREMain instance.

Definition at line 27 of file AZUREMain.h.

7.13.3 Member Function Documentation

7.13.3.1 compound()

```
CNuc * AZUREMain::compound ( ) const  [inline]
```

Returns a pointer to the CNuc object.

Definition at line 43 of file AZUREMain.h.

7.13.3.2 configure()

```
const Config & AZUREMain::configure ( ) const  [inline]
```

Returns a reference to the Config structure.

Definition at line 39 of file AZUREMain.h.

7.13.3.3 data()

```
EData * AZUREMain::data ( ) const  [inline]
```

Returns a pointer to the EData object.

Definition at line 47 of file AZUREMain.h.

7.13.3.4 operator()()

```
int AZUREMain::operator() ( )
```

The parenthesis operator is defined so the instance of AZUREMain can be called as a function. This executes AZURE against the configuration parameters.

Definition at line 12 of file AZUREMain.cpp.

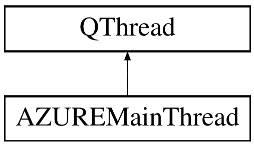
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREMain.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREMain.cpp

7.14 AZUREMainThread Class Reference

```
#include <AZUREMainThread.h>
```

Inheritance diagram for AZUREMainThread:



Public Slots

- void stopAZURE ()

Signals

- void readyToRun ()

Public Member Functions

- AZUREMainThread (RunTab ∗tab, const Config &configure)
- const Config & configure () const

Protected Member Functions

- void run ()

7.14.1 Detailed Description

Definition at line 32 of file AZUREMainThread.h.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 AZUREMainThread()

```
AZUREMainThread::AZUREMainThread (
            RunTab * tab,
            const Config & configure ) [inline]
```

Definition at line 35 of file AZUREMainThread.h.

7.14.3 Member Function Documentation

7.14.3.1 configure()

```
const Config & AZUREMainThread::configure ( ) const [inline]
```

Definition at line 53 of file AZUREMainThread.h.

7.14.3.2 readyToRun

```
void AZUREMainThread::readyToRun ( ) [signal]
```

7.14.3.3 run()

```
void AZUREMainThread::run ( ) [inline], [protected]
```

Definition at line 61 of file AZUREMainThread.h.

7.14.3.4 stopAZURE

```
void AZUREMainThread::stopAZURE ( ) [inline], [slot]
```

Definition at line 57 of file AZUREMainThread.h.

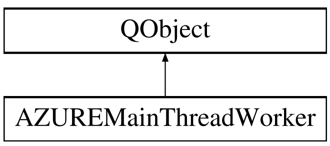
The documentation for this class was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREMainThread.h

7.15 AZUREMainThreadWorker Class Reference

```
#include <AZUREMainThread.h>
```

Inheritance diagram for AZUREMainThreadWorker:



Public Slots

- void run ()

Signals

- void done ()

Public Member Functions

- AZUREMainThreadWorker (const Config &configure)

7.15.1 Detailed Description

Definition at line 15 of file AZUREMainThread.h.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 AZUREMainThreadWorker()

```
AZUREMainThreadWorker::AZUREMainThreadWorker (
            const Config & configure )  [inline]
```

Definition at line 19 of file AZUREMainThread.h.

7.15.3 Member Function Documentation

7.15.3.1 done

```
void AZUREMainThreadWorker::done ( )  [signal]
```

7.15.3.2 run

```
void AZUREMainThreadWorker::run ( )  [inline], [slot]
```

Definition at line 24 of file AZUREMainThread.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREMainThread.h

7.16 AZUREOutput Class Reference

A class to assist in writing AZURE output files.

`#include <AZUREOutput.h>`

Public Member Functions

- AZUREOutput (std::string)
- ∼AZUREOutput ()
- bool IsExtrap () const
- std::filebuf ∗ operator() (int entranceKey, int exitKey, bool isAngDist=false)
- int NumAZUREFBuffers () const
- int IsAZUREFBuffer (int, int, bool)
- std::string GetOutputDir () const
- void AddAZUREFBuffer (AZUREFBuffer ∗)
- void SetExtrap ()
- AZUREFBuffer ∗ GetAZUREFBuffer (int)

7.16.1 Detailed Description

A class to assist in writing AZURE output files.

The EData::WriteOutputFiles function simply loops over all ESegment and EPoint objects when writing the output of an AZURE calculation. To ensure that all output for a given entrance and exit pair combination is written to a single file, the AZUREOutput class is used. The AZUREOutput object is a container for a vector of AZUREFBuffer objects.

Definition at line 16 of file AZUREOutput.h.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 AZUREOutput()

```
AZUREOutput::AZUREOutput (
            std::string outputdir )
```

The AZUREOutput object is created with reference to an output directory.

Definition at line 7 of file AZUREOutput.cpp.

7.16.2.2 ∼AZUREOutput()

```
AZUREOutput::∼AZUREOutput ( )
```

On destruction of the AZUREOutput instance, each AZUREFBuffer object is also destroyed.

Definition at line 16 of file AZUREOutput.cpp.

7.16.3 Member Function Documentation

7.16.3.1 AddAZUREFBuffer()

```
void AZUREOutput::AddAZUREFBuffer (
            AZUREFBuffer * azureFBuffer )
```

Adds a pointer to an AZUREFBuffer object to the vector.

Definition at line 86 of file AZUREOutput.cpp.

7.16.3.2 GetAZUREFBuffer()

```
AZUREFBuffer * AZUREOutput::GetAZUREFBuffer (
              int fBufferNum )
```

Returns a pointer to the AZUREFBuffer object specified by a position in the vector.

Definition at line 102 of file AZUREOutput.cpp.

7.16.3.3 GetOutputDir()

```
std::string AZUREOutput::GetOutputDir ( ) const
```

Returns the output directory for the AZUREOutput object.

Definition at line 78 of file AZUREOutput.cpp.

7.16.3.4 IsAZUREFBuffer()

```
int AZUREOutput::IsAZUREFBuffer (
              int entranceKey,
              int exitKey,
              bool isAngDist )
```

Tests if a pointer to an AZUREFBuffer object corresponding to the specified entrance and exit keys exists in the vector. If such a pointer exists, the position in the vector is returned. Otherwise, the function returns 0.

Definition at line 61 of file AZUREOutput.cpp.

7.16.3.5 IsExtrap()

```
bool AZUREOutput::IsExtrap ( ) const
```

Returns true if the output is an extrapolation.

Definition at line 24 of file AZUREOutput.cpp.

7.16.3.6 NumAZUREFBuffers()

```
int AZUREOutput::NumAZUREFBuffers ( ) const
```

Returns the number of pointers to AZUREFBuffer objects stored in the vector.

Definition at line 51 of file AZUREOutput.cpp.

7.16.3.7 operator()()

```
std::filebuf * AZUREOutput::operator() (
            int entranceKey,
            int exitKey,
            bool isAngDist = false )
```

The parenthesis operator is defined so that the instance of AZUREOutput can be called as a function.
The instance is called using a reference to an entrance and exit pair key combination. The function tests if there is a pointer to an AZUREFBuffer object in a vector. If such a pointer exists, the pointer to the actual file buffer contained in the corresponding AZUREFBuffer object is returned. Otherwise, a new AZUREFBuffer object is created with the entrance and exit key, a pointer to that object is stored in a vector, and the pointer to the actual new file buffer is returned.

Definition at line 37 of file AZUREOutput.cpp.

7.16.3.8 SetExtrap()

```
void AZUREOutput::SetExtrap ( )
```

Sets the extrapolation flag to true.

Definition at line 94 of file AZUREOutput.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREOutput.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREOutput.cpp

7.17 AZUREParams Class Reference

A container class to hold Minuit parameters in AZURE.

```
#include <AZUREParams.h>
```

Public Member Functions

- ROOT::Minuit2::MnUserParameters & GetMinuitParams ()
- void ReadUserParameters (const Config &)
- void WriteUserParameters (const Config &, bool)
- void WriteParameterErrors (const std::vector< std::pair< double, double > > &, const Config &)

7.17.1 Detailed Description

A container class to hold Minuit parameters in AZURE.

The AZUREParams class holds the Minuit parameters determined in the fit. The class also has member functions corresponding to reading and writing of the parameters and their errors.

Definition at line 20 of file AZUREParams.h.

7.17.2 Member Function Documentation

7.17.2.1 GetMinuitParams()

```
ROOT::Minuit2::MnUserParameters & AZUREParams::GetMinuitParams ( )
```

This function returns the MnUserParameters object used by Minuit to store the fit parameters.

Definition at line 9 of file AZUREParams.cpp.

7.17.2.2 ReadUserParameters()

```
void AZUREParams::ReadUserParameters (
            const Config & configure )
```

This function reads the user specified parameters from a given file.
These parameters are formal R-matrix parameters, and overwrite any initial parameters determined from the nuclear input file.

Definition at line 19 of file AZUREParams.cpp.

7.17.2.3 WriteParameterErrors()

```
void AZUREParams::WriteParameterErrors (
            const std::vector< std::pair< double, double > > & errors,
            const Config & configure )
```

This function writes the parameter errors to a file if Minos has been invoked.

Definition at line 88 of file AZUREParams.cpp.

7.17.2.4 WriteUserParameters()

```
void AZUREParams::WriteUserParameters (
            const Config & configure,
            bool fitParameters )
```

This function writes the formal R-matrix parameters to a file.

Definition at line 66 of file AZUREParams.cpp.

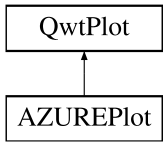
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREParams.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREParams.cpp

7.18 AZUREPlot Class Reference

```
#include <AZUREPlot.h>
```

Inheritance diagram for AZUREPlot:



Public Slots

- void draw (QList< PlotEntry * > newEntries)
- void update ()
- void exportPlot ()
- void print ()

Public Member Functions

- AZUREPlot (PlotTab *plotTab, QWidget *parent=0)
- void setXAxisLog (bool set)
- void setYAxisLog (bool set)
- void setXAxisType (unsigned int type)
- void setYAxisType (unsigned int type)
- void clearEntries ()

7.18.1 Detailed Description

Definition at line 59 of file AZUREPlot.h.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 AZUREPlot()

```
AZUREPlot::AZUREPlot (
                PlotTab * plotTab,
                QWidget * parent = 0 )
```

Definition at line 229 of file AZUREPlot.cpp.

7.18.3 Member Function Documentation

7.18.3.1 clearEntries()

```
void AZUREPlot::clearEntries ( )
```

Definition at line 383 of file AZUREPlot.cpp.

7.18.3.2 draw

```
void AZUREPlot::draw (
            QList< PlotEntry * > newEntries )  [slot]
```

Definition at line 285 of file AZUREPlot.cpp.

7.18.3.3 exportPlot

```
void AZUREPlot::exportPlot ( )  [slot]
```

Definition at line 326 of file AZUREPlot.cpp.

7.18.3.4 print

```
void AZUREPlot::print ( )  [slot]
```

Definition at line 367 of file AZUREPlot.cpp.

7.18.3.5 setXAxisLog()

```
void AZUREPlot::setXAxisLog (
            bool set )
```

Definition at line 248 of file AZUREPlot.cpp.

7.18.3.6 setXAxisType()

```
void AZUREPlot::setXAxisType (
            unsigned int type )
```

Definition at line 267 of file AZUREPlot.cpp.

7.18.3.7 setYAxisLog()

```
void AZUREPlot::setYAxisLog (
            bool set )
```

Definition at line 256 of file AZUREPlot.cpp.

7.18.3.8 setYAxisType()

```
void AZUREPlot::setYAxisType (
            unsigned int type )
```

Definition at line 278 of file AZUREPlot.cpp.

7.18.3.9 update

```
void AZUREPlot::update ( )  [slot]
```

Definition at line 313 of file AZUREPlot.cpp.

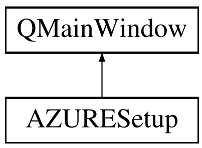
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREPlot.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZUREPlot.cpp

7.19 AZURESetup Class Reference

```
#include <AZURESetup.h>
```

Inheritance diagram for AZURESetup:



Public Slots

- void SaveAndRun ()
- void DeleteThread ()

Public Member Functions

- AZURESetup ()
- Config & GetConfig ()
- void open (QString filename)

7.19.1 Detailed Description

Definition at line 35 of file AZURESetup.h.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 AZURESetup()

```
AZURESetup::AZURESetup ( )
```

Definition at line 35 of file AZURESetup.cpp.

7.19.3 Member Function Documentation

7.19.3.1 DeleteThread

```
void AZURESetup::DeleteThread ( )  [slot]
```

Definition at line 962 of file AZURESetup.cpp.

7.19.3.2 GetConfig()

```
Config & AZURESetup::GetConfig ( )
```

Definition at line 80 of file AZURESetup.cpp.

7.19.3.3 open()

```
void AZURESetup::open (
            QString filename )
```

Definition at line 205 of file AZURESetup.cpp.

7.19.3.4 SaveAndRun

```
void AZURESetup::SaveAndRun ( )  [slot]
```

Definition at line 783 of file AZURESetup.cpp.

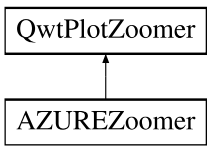
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZURESetup.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZURESetup.cpp

7.20 AZUREZoomer Class Reference

```
#include <AZUREPlot.h>
```

Inheritance diagram for AZUREZoomer:



Public Member Functions

- AZUREZoomer (QWidget ∗canvas)

Protected Member Functions

- QwtText trackerTextF (const QPointF &pos) const

7.20.1 Detailed Description

Definition at line 24 of file AZUREPlot.h.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 AZUREZoomer()

```
AZUREZoomer::AZUREZoomer (
             QWidget * canvas )  [inline]
```

Definition at line 26 of file AZUREPlot.h.

7.20.3 Member Function Documentation

7.20.3.1 trackerTextF()

```
QwtText AZUREZoomer::trackerTextF (
             const QPointF & pos ) const  [protected]
```

Definition at line 23 of file AZUREPlot.cpp.

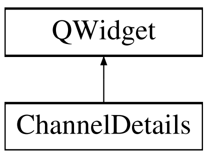
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREPlot.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZUREPlot.cpp

7.21 ChannelDetails Class Reference

```
#include <ChannelDetails.h>
```

Inheritance diagram for ChannelDetails:



Public Member Functions

- ChannelDetails (QWidget *parent=0)
- void setNormParam (int which)

Public Attributes

- QLineEdit * reducedWidthText
- QLabel * details

7.21.1 Detailed Description

Definition at line 13 of file ChannelDetails.h.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 ChannelDetails()

```
ChannelDetails::ChannelDetails (
            QWidget * parent = 0 )
```

Definition at line 8 of file ChannelDetails.cpp.

7.21.3 Member Function Documentation

7.21.3.1 setNormParam()

```
void ChannelDetails::setNormParam (
            int which )
```

Definition at line 31 of file ChannelDetails.cpp.

7.21.4 Member Data Documentation

7.21.4.1 details

```
QLabel* ChannelDetails::details
```

Definition at line 20 of file ChannelDetails.h.

7.21.4.2 reducedWidthText

```
QLineEdit* ChannelDetails::reducedWidthText
```

Definition at line 19 of file ChannelDetails.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelDetails.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChannelDetails.cpp

7.22 ChannelsData Struct Reference

```
#include <ChannelsModel.h>
```

Public Attributes

- int isFixed
- int levelIndex
- int pairIndex
- double sValue
- int lValue
- QChar radType
- double reducedWidth

Static Public Attributes

- static const int SIZE = 7

7.22.1 Detailed Description

Definition at line 7 of file ChannelsModel.h.

7.22.2 Member Data Documentation

7.22.2.1 isFixed

```
int ChannelsData::isFixed
```

Definition at line 9 of file ChannelsModel.h.

7.22.2.2 levelIndex

```
int ChannelsData::levelIndex
```

Definition at line 10 of file ChannelsModel.h.

7.22.2.3 lValue

```
int ChannelsData::lValue
```

Definition at line 13 of file ChannelsModel.h.

7.22.2.4 pairIndex

```
int ChannelsData::pairIndex
```

Definition at line 11 of file ChannelsModel.h.

7.22.2.5 radType

```
QChar ChannelsData::radType
```

Definition at line 14 of file ChannelsModel.h.

7.22.2.6 reducedWidth

```
double ChannelsData::reducedWidth
```

Definition at line 15 of file ChannelsModel.h.

7.22.2.7 SIZE

```
const int ChannelsData::SIZE = 7  [static]
```

Definition at line 8 of file ChannelsModel.h.

7.22.2.8 sValue

```
double ChannelsData::sValue
```

Definition at line 12 of file ChannelsModel.h.

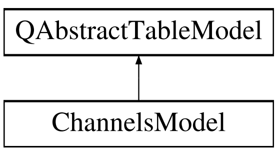
The documentation for this struct was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelsModel.h

7.23 ChannelsModel Class Reference

```
#include <ChannelsModel.h>
```

Inheritance diagram for ChannelsModel:



Public Member Functions

- ChannelsModel (QObject ∗parent=0)
- int rowCount (const QModelIndex &parent) const
- int columnCount (const QModelIndex &parent) const
- QVariant data (const QModelIndex &index, int role) const
- QVariant headerData (int section, Qt::Orientation orientation, int role) const
- QList< ChannelsData > getChannels () const
- bool setData (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- bool insertRows (int position, int rows, const QModelIndex &index=QModelIndex())
- bool removeRows (int position, int rows, const QModelIndex &index=QModelIndex())
- Qt::ItemFlags flags (const QModelIndex &index) const
- bool isChannel (const ChannelsData &channel) const
- QString getSpinLabel (const ChannelsData &channel) const
- void setPairsModel (PairsModel ∗model)

7.23.1 Detailed Description

Definition at line 20 of file ChannelsModel.h.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 ChannelsModel()

```
ChannelsModel::ChannelsModel (
            QObject * parent = 0 )
```

Definition at line 4 of file ChannelsModel.cpp.

7.23.3 Member Function Documentation

7.23.3.1 columnCount()

```
int ChannelsModel::columnCount (
            const QModelIndex & parent ) const
```

Definition at line 12 of file ChannelsModel.cpp.

7.23.3.2 data()

```
QVariant ChannelsModel::data (
            const QModelIndex & index,
            int role ) const
```

Definition at line 17 of file ChannelsModel.cpp.

7.23.3.3 flags()

```
Qt::ItemFlags ChannelsModel::flags (
            const QModelIndex & index ) const
```

Definition at line 156 of file ChannelsModel.cpp.

7.23.3.4 getChannels()

```
QList< ChannelsData > ChannelsModel::getChannels ( ) const  [inline]
```

Definition at line 30 of file ChannelsModel.h.

7.23.3.5 getSpinLabel()

```
QString ChannelsModel::getSpinLabel (
            const ChannelsData & channel ) const
```

Definition at line 179 of file ChannelsModel.cpp.

7.23.3.6 headerData()

```
QVariant ChannelsModel::headerData (
            int section,
            Qt::Orientation orientation,
            int role ) const
```

Definition at line 71 of file ChannelsModel.cpp.

7.23.3.7 insertRows()

```
bool ChannelsModel::insertRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 131 of file ChannelsModel.cpp.

7.23.3.8 isChannel()

```
bool ChannelsModel::isChannel (
            const ChannelsData & channel ) const
```

Definition at line 162 of file ChannelsModel.cpp.

7.23.3.9 removeRows()

```
bool ChannelsModel::removeRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 144 of file ChannelsModel.cpp.

7.23.3.10 rowCount()

```
int ChannelsModel::rowCount (
            const QModelIndex & parent ) const
```

Definition at line 7 of file ChannelsModel.cpp.

7.23.3.11 setData()

```
bool ChannelsModel::setData (
            const QModelIndex & index,
            const QVariant & value,
            int role = Qt::EditRole )
```

Definition at line 98 of file ChannelsModel.cpp.

7.23.3.12 setPairsModel()

```
void ChannelsModel::setPairsModel (
            PairsModel * model )
```

Definition at line 184 of file ChannelsModel.cpp.

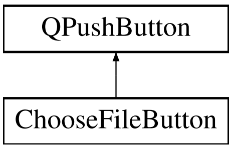
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelsModel.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChannelsModel.cpp

7.24 ChooseFileButton Class Reference

```
#include <ChooseFileButton.h>
```

Inheritance diagram for ChooseFileButton:



Public Slots

- void click ()

Signals

- void clicked (QLineEdit ∗lineEdit)

Public Member Functions

- ChooseFileButton (const QString &text, QWidget ∗parent=0)
- void setLineEdit (QLineEdit ∗lineEdit)

7.24.1 Detailed Description

Definition at line 9 of file ChooseFileButton.h.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 ChooseFileButton()

```
ChooseFileButton::ChooseFileButton (
             const QString & text,
             QWidget * parent = 0 )
```

Definition at line 3 of file ChooseFileButton.cpp.

7.24.3 Member Function Documentation

7.24.3.1 click

```
void ChooseFileButton::click ( )  [slot]
```

Definition at line 12 of file ChooseFileButton.cpp.

7.24.3.2 clicked

```
void ChooseFileButton::clicked (
             QLineEdit * lineEdit )  [signal]
```

7.24.3.3 setLineEdit()

```
void ChooseFileButton::setLineEdit (
            QLineEdit * lineEdit )
```

Definition at line 8 of file ChooseFileButton.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChooseFileButton.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChooseFileButton.cpp

7.25 CNuc Class Reference

An AZURE compound nucleus.

```
#include <CNuc.h>
```

Public Member Functions

- bool IsPairKey (int)
- int NumPairs () const
- int NumJGroups () const
- int IsPair (PPair)
- int IsJGroup (JGroup)
- int GetPairNumFromKey (int)
- int Fill (const Config &)
- void ParseExternalCapture (const Config &, std::map< int, int > &)
- int GetMaxLValue () const
- void Initialize (const Config &)
- void AddPair (PPair)
- void AddJGroup (JGroup)
- void PrintNuc (const Config &)
- void TransformIn (const Config &)
- void SortPathways (const Config &)
- void PrintPathways (const Config &)
- void CalcBoundaryConditions (const Config &)
- void PrintBoundaryConditions (const Config &)
- void CalcAngularDists (int)
- void PrintAngularDists (const Config &)
- void FillMnParams (ROOT::Minuit2::MnUserParameters &)
- void FillCompoundFromParams (const vector_r &)
- void TransformOut (const Config &)
- void PrintTransformParams (const Config &)
- void SetMaxLValue (int)
- void CalcShiftFunctions (const Config &)
- complex CalcExternalWidth (JGroup *, ALevel *, AChannel *, bool, const Config &)
- PPair * GetPair (int)
- JGroup * GetJGroup (int)
- CNuc * Clone () const

7.25.1 Detailed Description

An AZURE compound nucleus.

The compound nucleus is the fundamental concept of R-Matrix theory. As such, the CNuc object in AZURE is the top level container object for all structure and reaction objects. Specifically, the CNuc object is the container object for vectors of PPair and JGroup objects, within which all other nuclear data objects are contained.

Definition at line 25 of file CNuc.h.

7.25.2 Member Function Documentation

7.25.2.1 AddJGroup()

```
void CNuc::AddJGroup (
             JGroup jGroup )
```

Adds a J^π group to the JGroup vector.

Definition at line 314 of file CNuc.cpp.

7.25.2.2 AddPair()

```
void CNuc::AddPair (
             PPair pPair )
```

Adds a particle pair to the PPair vector.

Definition at line 306 of file CNuc.cpp.

7.25.2.3 CalcAngularDists()

```
void CNuc::CalcAngularDists (
             int maxL )
```

Creates and sorts the KLGroup and Interference objects and calculates the appropriate coefficients.

Definition at line 995 of file CNuc.cpp.

7.25.2.4 CalcBoundaryConditions()

```
void CNuc::CalcBoundaryConditions (
             const Config & configure )
```

Calculates the boundary conditions. Boundary conditions for each channel are evaluated at the energy of the first level read from the nuclear input file in the J^π group.

Definition at line 916 of file CNuc.cpp.

7.25.2.5 CalcExternalWidth()

```
complex CNuc::CalcExternalWidth (
            JGroup * theJGroup,
            ALevel * theLevel,
            AChannel * theChannel,
            bool isInitial,
            const Config & configure )
```

Calculates the external reduced width amplitudes for a given channel.

Definition at line 1536 of file CNuc.cpp.

7.25.2.6 CalcShiftFunctions()

```
void CNuc::CalcShiftFunctions (
            const Config & configure )
```

This function is called for each iteration to calculate the shift functions at new level energies when the Brune parametrization is used.

Definition at line 1497 of file CNuc.cpp.

7.25.2.7 Clone()

```
CNuc * CNuc::Clone ( ) const
```

Creates a new copy of the CNuc object in memory and returns a pointer to the new object. Used in AZURECalc function class for thread safety.

Definition at line 1635 of file CNuc.cpp.

7.25.2.8 Fill()

```
int CNuc::Fill (
            const Config & configure )
```

Fills the compound nucleus object, and all nested objects, from data specified in the nuclear and external capture input files. Returns -1 if the files could not be read, and 0 if the files were read successfully.

Definition at line 103 of file CNuc.cpp.

7.25.2.9 FillCompoundFromParams()

```
void CNuc::FillCompoundFromParams (
            const vector_r & p )
```

Fills the CNuc object from the Minuit parameter array.

Definition at line 1176 of file CNuc.cpp.

7.25.2.10 FillMnParams()

```
void CNuc::FillMnParams (
            ROOT::Minuit2::MnUserParameters & p )
```

Fills the Minuit parameter array from initial values in the CNuc object.

Definition at line 1147 of file CNuc.cpp.

7.25.2.11 GetJGroup()

```
JGroup * CNuc::GetJGroup (
            int jGroupNum )
```

Returns a pointer to the J^{π} group specified by a position in the JGroup vector.

Definition at line 1625 of file CNuc.cpp.

7.25.2.12 GetMaxLValue()

```
int CNuc::GetMaxLValue ( ) const
```

Returns the maximum value of orbital angular momentum read from channels in the nuclear file.

Definition at line 265 of file CNuc.cpp.

7.25.2.13 GetPair()

```
PPair * CNuc::GetPair (
            int pairNum )
```

Returns a pointer to the particle pair specified by a position in the PPair vector.

Definition at line 1616 of file CNuc.cpp.

7.25.2.14 GetPairNumFromKey()

```
int CNuc::GetPairNumFromKey (
            int key )
```

Returns the position of a particle pair in the PPair vector based on the pair key. Pair keys are how particle pairs are specified in the setup files, but may not correspond to the position of the particle pair in the PPair vector. If the pair exists, the position in the vector is returned. Otherwise, the function returns 0.

Definition at line 87 of file CNuc.cpp.

7.25.2.15 Initialize()

```
void CNuc::Initialize (
            const Config & configure )
```

Initializes the compound nucleus object. This includes calculating the boundary conditions, transforming from physical to formal parameters, creating and sorting all reaction pathways, and calculating angular interference contributions and coefficients. A CNuc object can only be initialized for use AFTER it is filled.

Definition at line 275 of file CNuc.cpp.

7.25.2.16 IsJGroup()

```
int CNuc::IsJGroup (
            JGroup jGroup )
```

Tests if a J^{π} group exists in the JGroup vector. If the group exists, the position in the vector is returned. Otherwise, the function returns 0.

Definition at line 67 of file CNuc.cpp.

7.25.2.17 IsPair()

```
int CNuc::IsPair (
            PPair pair )
```

Tests if a particle pair exists in the PPair vector. If pair exists, the position in the vector is returned. Otherwise, the function returns 0.

Definition at line 50 of file CNuc.cpp.

7.25.2.18 IsPairKey()

```
bool CNuc::IsPairKey (
            int key )
```

Returns true if a specified pair key exists in the PPair vector, otherwise returns false.

Definition at line 19 of file CNuc.cpp.

7.25.2.19 NumJGroups()

```
int CNuc::NumJGroups ( ) const
```

Returns the number of J^{π} groups in the JGroup vector.

Definition at line 41 of file CNuc.cpp.

7.25.2.20 NumPairs()

```
int CNuc::NumPairs ( ) const
```

Returns the number of particle pairs in the PPair vector.

Definition at line 33 of file CNuc.cpp.

7.25.2.21 ParseExternalCapture()

```
void CNuc::ParseExternalCapture (
            const Config & configure,
            std::map< int, int > & ecPairs )
```

Fills the ECLevel vector from information in the external capture file. Also tests if the final state for external capture exists from the nuclear file. If not, the state is created.

Definition at line 182 of file CNuc.cpp.

7.25.2.22 PrintAngularDists()

```
void CNuc::PrintAngularDists (
            const Config & configure )
```

Prints the KLGroup and Interference object structure as well as the calculated coefficients.

Definition at line 1095 of file CNuc.cpp.

7.25.2.23 PrintBoundaryConditions()

```
void CNuc::PrintBoundaryConditions (
            const Config & configure )
```

Prints the boundary conditions.

Definition at line 955 of file CNuc.cpp.

7.25.2.24 PrintNuc()

```
void CNuc::PrintNuc (
            const Config & configure )
```

Prints the initial structure of the compound nucleus object after filling but before initialization. This includes all particle pairs, J^{π} groups, levels and channels.

Definition at line 323 of file CNuc.cpp.

7.25.2.25 PrintPathways()

```
void CNuc::PrintPathways (
            const Config & configure )
```

Prints the internal and external reaction pathways.

Definition at line 809 of file CNuc.cpp.

7.25.2.26 PrintTransformParams()

```
void CNuc::PrintTransformParams (
            const Config & configure )
```

Writes the final transformed parameters to ¨parameters.out¨ file.

Definition at line 1407 of file CNuc.cpp.

7.25.2.27 SetMaxLValue()

```
void CNuc::SetMaxLValue (
            int maxL )
```

Sets the maximum orbital angular momentum value read from the nuclear input file.

Definition at line 1488 of file CNuc.cpp.

7.25.2.28 SortPathways()

```
void CNuc::SortPathways (
            const Config & configure )
```

Calculates internal and external reaction pathways.

Definition at line 619 of file CNuc.cpp.

7.25.2.29 TransformIn()

```
void CNuc::TransformIn (
            const Config & configure )
```

Performs the initial parameter transformations from physical to formal parameters.

Definition at line 403 of file CNuc.cpp.

7.25.2.30 TransformOut()

```
void CNuc::TransformOut (
            const Config & configure )
```

Performs the final parameter transformations from formal to physical parameters.

Definition at line 1200 of file CNuc.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/CNuc.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/CNuc.cpp

7.26 Config Class Reference

A configuration structure for AZURE.

```
#include <Config.h>
```

Public Types

- enum ParameterFlags {
  USE_AMATRIX =(1<<0) , PERFORM_ERROR_ANALYSIS =(1<<1) , PERFORM_FIT =(1<<2) , CALCULATE_WITH_DATA =(1<<3) ,
  USE_PREVIOUS_PARAMETERS =(1<<4) , USE_EXTERNAL_CAPTURE =(1<<5) , USE_PREVIOUS_INTEGRALS =(1<<6) , CALCULATE_REACTION_RATE =(1<<7) ,
  TRANSFORM_PARAMETERS =(1<<8) , USE_BRUNE_FORMALISM =(1<<9) , IGNORE_ZERO_WIDTHS =(1<<10) , USE_RMC_FORMALISM =(1<<11) ,
  USE_GSL_COULOMB_FUNC =(1<<12) , USE_LONGWAVELENGTH_APPROX =(1<<13) }
- enum CheckFileFlags {
  CHECK_COMPOUND_NUCLEUS =(1<<0) , CHECK_PATHWAYS =(1<<1) , CHECK_DATA =(1<<2) , CHECK_ENERGY_DEP =(1<<3) ,
  CHECK_LEGENDRE =(1<<4) , CHECK_BOUNDARY_CONDITIONS =(1<<5) , CHECK_ANGULAR_DISTS =(1<<6) , CHECK_COUL_AMPLITUDES =(1<<7) }

Public Member Functions

- Config (std::ostream &stream)
- void Reset ()
- int ReadConfigFile ()
- int CheckForInputFiles ()

Public Attributes

- std::ostream & outStream

    Output stream.
- std::string configfile

    The runtime configuration file name.
- bool stopFlag

    A control variable to stop AZURE calculation.
- unsigned int paramMask

    A bitmask for the encoding of configuration flags.
- unsigned int screenCheckMask

    A bitmask storing which checks are printed to screen.
- unsigned int fileCheckMask

    A bitmask storing which checks are printed to file.
- double chiVariance

    If performError is true, sets the value of Up (the acceptable variance from the minimum chi-squared.
- std::string outputdir

    The path of the output directory.
- std::string checkdir

    The path of the check files directory.
- std::string paramfile

    The name of the parameters file from which to read.
- std::string integralsfile

    The name of the external capture amplitudes file from which to read.
- RateParams rateParams

    Parameters for calculating reaction rate.

Static Public Attributes

- static const int maxLOrder =20

    A constant indicating the maximum order of the Legendre polynomials to calculate.

7.26.1 Detailed Description

A configuration structure for AZURE.

The configuration structure is created from the runtime file passed to the AZURE executable, as well as the options specified in the command shell prompt.

Definition at line 37 of file Config.h.

7.26.2 Member Enumeration Documentation

7.26.2.1 CheckFileFlags

```
enum Config::CheckFileFlags
```

Bit flags for check file control in AZURE2.

Enumerator

CHECK_COMPOUND_NUCLEUS	
CHECK_PATHWAYS	
CHECK_DATA	
CHECK_ENERGY_DEP	
CHECK_LEGENDRE	
CHECK_BOUNDARY_CONDITIONS	
CHECK_ANGULAR_DISTS	
CHECK_COUL_AMPLITUDES	

Definition at line 63 of file Config.h.

7.26.2.2 ParameterFlags

```
enum Config::ParameterFlags
```

Bit flags for various options in AZURE2.

Enumerator

USE_AMATRIX	
PERFORM_ERROR_ANALYSIS	
PERFORM_FIT	
CALCULATE_WITH_DATA	
USE_PREVIOUS_PARAMETERS	
USE_EXTERNAL_CAPTURE	
USE_PREVIOUS_INTEGRALS	
CALCULATE_REACTION_RATE	
TRANSFORM_PARAMETERS	
USE_BRUNE_FORMALISM	
IGNORE_ZERO_WIDTHS	
USE_RMC_FORMALISM	
USE_GSL_COULOMB_FUNC	
USE_LONGWAVELENGTH_APPROX	

Definition at line 44 of file Config.h.

7.26.3 Constructor & Destructor Documentation

7.26.3.1 Config()

```
Config::Config (
            std::ostream & stream )
```

The constructor of the Config class sets defaults and the stream reference for output.

Definition at line 12 of file Config.cpp.

7.26.4 Member Function Documentation

7.26.4.1 CheckForInputFiles()

```
int Config::CheckForInputFiles ( )
```

If stat() is enabled, this function checks for the output and checks directories at runtime.

Definition at line 102 of file Config.cpp.

7.26.4.2 ReadConfigFile()

```
int Config::ReadConfigFile ( )
```

This funciton reads the configuration file and parses various options.

Definition at line 35 of file Config.cpp.

7.26.4.3 Reset()

```
void Config::Reset ( )
```

This function resets Config structure.

Definition at line 20 of file Config.cpp.

7.26.5 Member Data Documentation

7.26.5.1 checkdir

```
std::string Config::checkdir
```

The path of the check files directory.

Definition at line 90 of file Config.h.

7.26.5.2 chiVariance

```
double Config::chiVariance
```

If performError is true, sets the value of Up (the acceptable variance from the minimum chi-squared.

Definition at line 86 of file Config.h.

7.26.5.3 configfile

`std::string Config::configfile`

The runtime configuration file name.

Definition at line 76 of file Config.h.

7.26.5.4 fileCheckMask

`unsigned int Config::fileCheckMask`

A bitmask storing which checks are printed to file.

Definition at line 84 of file Config.h.

7.26.5.5 integralsfile

`std::string Config::integralsfile`

The name of the external capture amplitudes file from which to read.

Definition at line 94 of file Config.h.

7.26.5.6 maxLOrder

`const int Config::maxLOrder =20  [static]`

A constant indicating the maximum order of the Legendre polynomials to calculate.

Definition at line 98 of file Config.h.

7.26.5.7 outputdir

`std::string Config::outputdir`

The path of the output directory.

Definition at line 88 of file Config.h.

7.26.5.8 outStream

`std::ostream& Config::outStream`

Output stream.

Definition at line 74 of file Config.h.

7.26.5.9 paramfile

`std::string Config::paramfile`

The name of the parameters file from which to read.

Definition at line 92 of file Config.h.

7.26.5.10 paramMask

`unsigned int Config::paramMask`

A bitmask for the encoding of configuration flags.

Definition at line 80 of file Config.h.

7.26.5.11 rateParams

`RateParams Config::rateParams`

Parameters for calculating reaction rate.

Definition at line 96 of file Config.h.

7.26.5.12 screenCheckMask

`unsigned int Config::screenCheckMask`

A bitmask storing which checks are printed to screen.

Definition at line 82 of file Config.h.

7.26.5.13 stopFlag

`bool Config::stopFlag`

A control variable to stop AZURE calculation.

Definition at line 78 of file Config.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/Config.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/Config.cpp

7.27 CoulFunc Class Reference

A function class to calculate Coulomb functions for positive energy channels.

`#include <CoulFunc.h>`

Public Member Functions

- CoulFunc (PPair *pPair, bool useGSLFunctions)
- int z1 () const
- int z2 () const
- double redmass () const
- int lLast () const
- double radiusLast () const
- double energyLast () const
- struct CoulWaves coulLast () const
- void setLast (int, double, double, CoulWaves)
- CoulWaves operator() (int, double, double)
- double Penetrability (int, double, double)
- double PEShift (int, double, double)
- double PEShift_dE (int, double, double)

7.27.1 Detailed Description

A function class to calculate Coulomb functions for positive energy channels.

The CoulFunc function class calculates the solutions to the Coulomb equation, as well as other useful quantities such as shift functions and their energy derivative and penetrabilities.

Definition at line 32 of file CoulFunc.h.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 CoulFunc()

```
CoulFunc::CoulFunc (
            PPair * pPair,
            bool useGSLFunctions )
```

The CoulFunc object is created with reference to a PPair object.

Definition at line 12 of file CoulFunc.cpp.

7.27.3 Member Function Documentation

7.27.3.1 coulLast()

```
struct CoulWaves CoulFunc::coulLast ( ) const
```

Returns the last Coulomb functions which were calculated.

Definition at line 82 of file CoulFunc.cpp.

7.27.3.2 energyLast()

```
double CoulFunc::energyLast ( ) const
```

Returns the last energy value at which the Coulomb functions were calculated.

Definition at line 74 of file CoulFunc.cpp.

7.27.3.3 lLast()

```
int CoulFunc::lLast ( ) const
```

Returns the last orbital angular momentum value at which the Coulomb functions were calculated.

Definition at line 56 of file CoulFunc.cpp.

7.27.3.4 operator()()

```
CoulWaves CoulFunc::operator() (
            int l,
            double radius,
            double energy )
```

The parenthesis operator is defined to make the class instance callable as a function. The orbital angular momentum, radius, and energy in the center of mass system are the dependent variables. The function returns the Coulomb waves.

Definition at line 106 of file CoulFunc.cpp.

7.27.3.5 Penetrability()

```
double CoulFunc::Penetrability (
            int l,
            double radius,
            double energy )
```

Returns the penetrability as a function of orbital angular momentum, radius, and energy in the center of mass system.

Definition at line 151 of file CoulFunc.cpp.

7.27.3.6 PEShift()

```
double CoulFunc::PEShift (
            int l,
            double radius,
            double energy )
```

Returns the positive energy shift function a function of orbital angular momentum, radius, and energy in the center of mass system.

Definition at line 162 of file CoulFunc.cpp.

7.27.3.7 PEShift_dE()

```
double CoulFunc::PEShift_dE (
            int l,
            double radius,
            double energy )
```

Returns the energy derivative of the shift function a function of orbital angular momentum, radius, and energy in the center of mass system.

Definition at line 184 of file CoulFunc.cpp.

7.27.3.8 radiusLast()

```
double CoulFunc::radiusLast ( ) const
```

Returns the last radius value at which the Coulomb functions were calculated.

Definition at line 65 of file CoulFunc.cpp.

7.27.3.9 redmass()

```
double CoulFunc::redmass ( ) const
```

Returns the reduced mass of the particle pair.

Definition at line 47 of file CoulFunc.cpp.

7.27.3.10 setLast()

```
void CoulFunc::setLast (
            int lLast,
            double rLast,
            double eLast,
            CoulWaves coulLast )
```

Sets the last calculated Coulomb waves and the values for which they were calculated.

Definition at line 90 of file CoulFunc.cpp.

7.27.3.11 z1()

```
int CoulFunc::z1 ( ) const
```

Returns the atomic number of the first particle in the pair.

Definition at line 31 of file CoulFunc.cpp.

7.27.3.12 z2()

```
int CoulFunc::z2 ( ) const
```

Returns the atomic number of the second particle in the pair.

Definition at line 39 of file CoulFunc.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/CoulFunc.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/CoulFunc.cpp

7.28 Coulomb_wave_functions Class Reference

```
#include <cwfcomp.H>
```

Public Member Functions

- Coulomb_wave_functions (const bool is_it_normalized_c, const std::complex< double > &l_c, const std::complex< double > &eta_c)
- ~Coulomb_wave_functions (void)
- void F_dF_init (const std::complex< double > &z, const std::complex< double > &F, const std::complex< double > &dF)
- void F_dF (const std::complex< double > &z, std::complex< double > &F, std::complex< double > &dF)
- void G_dG (const std::complex< double > &z, std::complex< double > &G, std::complex< double > &dG)
- void H_dH (const int omega, const std::complex< double > &z, std::complex< double > &H, std::complex< double > &dH)
- void H_dH_scaled (const int omega, const std::complex< double > &z, std::complex< double > &H, std::complex< double > &dH)

Public Attributes

- const std::complex< double > l
- const std::complex< double > eta
- const bool is_it_normalized

7.28.1 Detailed Description

Definition at line 9 of file cwfcomp.H.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 Coulomb_wave_functions()

```
Coulomb_wave_functions::Coulomb_wave_functions (
            const bool is_it_normalized_c,
            const std::complex< double > & l_c,
            const std::complex< double > & eta_c ) [inline]
```

Definition at line 25 of file cwfcomp.H.

7.28.2.2 ∼Coulomb_wave_functions()

```
Coulomb_wave_functions::~Coulomb_wave_functions (
             void ) [inline]
```

Definition at line 68 of file cwfcomp.H.

7.28.3 Member Function Documentation

7.28.3.1 F_dF()

```
void Coulomb_wave_functions::F_dF (
             const std::complex< double > & z,
             std::complex< double > & F,
             std::complex< double > & dF )
```

Definition at line 1162 of file cwfcomp.cpp.

7.28.3.2 F_dF_init()

```
void Coulomb_wave_functions::F_dF_init (
             const std::complex< double > & z,
             const std::complex< double > & F,
             const std::complex< double > & dF )
```

Definition at line 1433 of file cwfcomp.cpp.

7.28.3.3 G_dG()

```
void Coulomb_wave_functions::G_dG (
             const std::complex< double > & z,
             std::complex< double > & G,
             std::complex< double > & dG )
```

Definition at line 1233 of file cwfcomp.cpp.

7.28.3.4 H_dH()

```
void Coulomb_wave_functions::H_dH (
             const int omega,
             const std::complex< double > & z,
             std::complex< double > & H,
             std::complex< double > & dH )
```

Definition at line 1304 of file cwfcomp.cpp.

7.28.3.5 H_dH_scaled()

```
void Coulomb_wave_functions::H_dH_scaled (
            const int omega,
            const std::complex< double > & z,
            std::complex< double > & H,
            std::complex< double > & dH )
```

Definition at line 1384 of file cwfcomp.cpp.

7.28.4 Member Data Documentation

7.28.4.1 eta

```
const std::complex<double> Coulomb_wave_functions::eta
```

Definition at line 91 of file cwfcomp.H.

7.28.4.2 is_it_normalized

```
const bool Coulomb_wave_functions::is_it_normalized
```

Definition at line 93 of file cwfcomp.H.

7.28.4.3 l

```
const std::complex<double> Coulomb_wave_functions::l
```

Definition at line 91 of file cwfcomp.H.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/cwfcomp.H
- /Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/cwfcomp.cpp

7.29 CoulWaves Struct Reference

The return structure of the CoulFunc function class.

```
#include <CoulFunc.h>
```

Public Attributes

- double F

  Regular solution.
- double dF

  Derivative of regular solution with respect to ρ.
- double G

  Irregular solution.
- double dG

  Derivative of irregular solution with respect to ρ.

7.29.1 Detailed Description

The return structure of the CoulFunc function class.

The CoulWaves structure contains both the irregular and regular solutions to the Coulomb equation, as well as their derivatives with respect to ρ.

Definition at line 13 of file CoulFunc.h.

7.29.2 Member Data Documentation

7.29.2.1 dF

```
double CoulWaves::dF
```

Derivative of regular solution with respect to ρ.

Definition at line 17 of file CoulFunc.h.

7.29.2.2 dG

```
double CoulWaves::dG
```

Derivative of irregular solution with respect to ρ.

Definition at line 21 of file CoulFunc.h.

7.29.2.3 F

```
double CoulWaves::F
```

Regular solution.

Definition at line 15 of file CoulFunc.h.

7.29.2.4 G

```
double CoulWaves::G
```

Irregular solution.

Definition at line 19 of file CoulFunc.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/CoulFunc.h

7.30 DataLine Class Reference

A class to read and store a line from a data file.

```
#include <DataLine.h>
```

Public Member Functions

- DataLine (std::ifstream &stream)
- double angle () const
- double energy () const
- double crossSection () const
- double error () const

7.30.1 Detailed Description

A class to read and store a line from a data file.

The DataLine class reads and stores a formatted line from a data file.

Definition at line 12 of file DataLine.h.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 DataLine()

```
DataLine::DataLine (
            std::ifstream & stream )  [inline]
```

Constructor fills the DataLine object from an input stream.

Definition at line 17 of file DataLine.h.

7.30.3 Member Function Documentation

7.30.3.1 angle()

```
double DataLine::angle ( ) const  [inline]
```

Returns the angle for the read in data point.

Definition at line 23 of file DataLine.h.

7.30.3.2 crossSection()

```
double DataLine::crossSection ( ) const  [inline]
```

Returns the cross section for the read in data point.

Definition at line 31 of file DataLine.h.

7.30.3.3 energy()

```
double DataLine::energy ( ) const  [inline]
```

Returns the energy for the read in data point.

Definition at line 27 of file DataLine.h.

7.30.3.4 error()

```
double DataLine::error ( ) const  [inline]
```

Returns the cross section error for the read in data point.

Definition at line 35 of file DataLine.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/DataLine.h

7.31 Decay Class Reference

An AZURE decay pair.

```
#include <Decay.h>
```

Public Member Functions

- Decay (int)
- int GetPairNum () const
- int NumKGroups () const
- int NumKLGroups () const
- int IsKGroup (KGroup)
- int IsKLGroup (KLGroup)
- void AddKGroup (KGroup)
- void AddKLGroup (KLGroup)
- KGroup ∗ GetKGroup (int)
- KLGroup ∗ GetKLGroup (int)

7.31.1 Detailed Description

An AZURE decay pair.

In AZURE, a Decay object represents a decay pair of the compound nucleus. The Decay object is keyed to a particle pair in the PPair vector, and serves as a container class for the KGroup and the KLGroup vectors and their subsequent reaction pathways.

Definition at line 14 of file Decay.h.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 Decay()

```
Decay::Decay (
            int pairNum )
```

The Decay object is created with a reference to a particle pair number, which represents a position in the PPair vector.

Definition at line 8 of file Decay.cpp.

7.31.3 Member Function Documentation

7.31.3.1 AddKGroup()

```
void Decay::AddKGroup (
            KGroup kGroup )
```

Adds a s, s' combination to the KGroup vector.

Definition at line 75 of file Decay.cpp.

7.31.3.2 AddKLGroup()

```
void Decay::AddKLGroup (
            KLGroup klGroup )
```

Adds a k, L combination to the KLGroup vector.

Definition at line 83 of file Decay.cpp.

7.31.3.3 GetKGroup()

```
KGroup * Decay::GetKGroup (
            int kGroupNum )
```

Returns a pointer to a s, s' combination specified by a position in the KGroup vector.

Definition at line 91 of file Decay.cpp.

7.31.3.4 GetKLGroup()

```
KLGroup * Decay::GetKLGroup (
            int klGroupNum )
```

Returns a pointer to a k, L combination specified by a position in the KLGroup vector.

Definition at line 100 of file Decay.cpp.

7.31.3.5 GetPairNum()

```
int Decay::GetPairNum ( ) const
```

Returns the pair number of the decay.

Definition at line 15 of file Decay.cpp.

7.31.3.6 IsKGroup()

```
int Decay::IsKGroup (
            KGroup a )
```

Tests a specific s, s' combination to determine if it exists in the KGroup vector. If the combination exists, the position of the combination in the vector is returned. Otherwise, the function returns 0.

Definition at line 40 of file Decay.cpp.

7.31.3.7 IsKLGroup()

```
int Decay::IsKLGroup (
            KLGroup a )
```

Tests a specific k, L combination to determine if it exists in the KLGroup vector. If the combination exists, the position of the combination in the vector is returned. Otherwise, the function returns 0.

Definition at line 58 of file Decay.cpp.

7.31.3.8 NumKGroups()

```
int Decay::NumKGroups ( ) const
```

Returns the number of s, s' combinations in the KGroup vector.

Definition at line 23 of file Decay.cpp.

7.31.3.9 NumKLGroups()

```
int Decay::NumKLGroups ( ) const
```

Returns the number of k, L combinations in the KLGroup vector.

Definition at line 31 of file Decay.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/Decay.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/Decay.cpp

7.32 Directories Class Reference

`#include <AZURESetup.h>`

Public Member Functions

- Directories ()

Public Attributes

- QString outputDir
- QString checksDir

7.32.1 Detailed Description

Definition at line 28 of file AZURESetup.h.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 Directories()

```
Directories::Directories ( )  [inline]
```

Definition at line 30 of file AZURESetup.h.

7.32.3 Member Data Documentation

7.32.3.1 checksDir

```
QString Directories::checksDir
```

Definition at line 32 of file AZURESetup.h.

7.32.3.2 outputDir

```
QString Directories::outputDir
```

Definition at line 31 of file AZURESetup.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZURESetup.h

7.33 ECIntegral Class Reference

A function class to calculate external capture integrals.

```
#include <ECIntegral.h>
```

Public Member Functions

- ECIntegral (PPair ∗pPair, const Config &configure)
- ∼ECIntegral ()
- complex operator() (int, int, double, double, double, double, int, char, double, double, bool)

7.33.1 Detailed Description

A function class to calculate external capture integrals.

The ECIntegral function class calculates external capture integrals for both positive and negative energy channels. The results are returned as an ECIntResult structure.

Definition at line 18 of file ECIntegral.h.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 ECIntegral()

```
ECIntegral::ECIntegral (
            PPair * pPair,
            const Config & configure )  [inline]
```

The ECIntegral object is created with reference to a PPair object. The PPair object is used to create new instances of the CoulFunc and WhitFunc objects.

Definition at line 25 of file ECIntegral.h.

7.33.2.2 ∼ECIntegral()

```
ECIntegral::∼ECIntegral ( )  [inline]
```

The CoulFunc and WhitFunc objects are destroyed with the object.

Definition at line 34 of file ECIntegral.h.

7.33.3 Member Function Documentation

7.33.3.1 operator()()

```
complex ECIntegral::operator() (
            int theInitialLValue,
            int theFinalLValue,
            double theInitialSValue,
            double theFinalSValue,
            double theInitialJValue,
            double theFinalJValue,
            int theLMult,
            char radType,
            double inEnergy,
            double levelEnergy,
            bool isChannelCapture )
```

Overloaded operator to make the class instance callable as a function. The intial and final orbital angular momentum, the gamma multipolarity, and the incoming energy and final state energy in the compound system are the dependent variables. The function returns an ECIntResult structure.

Definition at line 102 of file ECIntegral.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/ECIntegral.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/ECIntegral.cpp

7.34 ECMGroup Class Reference

An AZURE external reaction pathway.

```
#include <ECMGroup.h>
```

Public Member Functions

- ECMGroup (char, int, int, double, int, int, int)
- ECMGroup (char, int, int, double, int, int, int, int, int, int, int)
- bool IsChannelCapture () const
- char GetRadType () const
- int GetMult () const
- int GetL () const
- int GetFinalChannel () const
- int GetJGroupNum () const
- int GetLevelNum () const
- int GetChanCapDecay () const
- int GetChanCapKGroup () const
- int GetChanCapMGroup () const
- int GetIntChannelNum () const
- double GetJ () const
- double GetStatSpinFactor () const
- void SetStatSpinFactor (double)

7.34.1 Detailed Description

An AZURE external reaction pathway.

An external reaction pathways in AZURE is one of two types: a hard sphere pathway or a resonant pathway. The hard sphere pathways refers to the portion of the initial incoming plus outgoing scattering wavefunction that is hard-shpere scattered and captured directly to a final state. The resonant pathways refer to the portion of the outgoing scattering wavefunction that is scattered/transformed by the R-Matrix. This type of pathway is linked to the internal resonant pathways, and can be thought of as first passing through a the resonant T-matrix before being captured directly to a final state.

Definition at line 17 of file ECMGroup.h.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 ECMGroup() [1/2]

```
ECMGroup::ECMGroup (
            char radType,
            int multipolarity,
            int lInitial,
            double jInitial,
            int finalChannelNum,
            int ecJGroupNum,
            int ecLevelNum )
```

This constructor is used to create hard sphere external reaction pathways. The type of radiation is specified, as well as the position of the final state in the ECLevel vector. The final channel number is also specified.

Definition at line 8 of file ECMGroup.cpp.

7.34.2.2 ECMGroup() [2/2]

```
ECMGroup::ECMGroup (
            char radType,
            int multipolarity,
            int lInitial,
            double jInitial,
            int finalChannelNum,
            int ecJGroupNum,
            int ecLevelNum,
            int decayNum,
            int kGroupNum,
            int mGroupNum,
            int internalChannel )
```

This constructor is used to create resonant external reaction pathways. The constructor is passed identical information as in the previous case, with the addition of references to the resonant reaction pathways through which the external pathways will pass.

Definition at line 19 of file ECMGroup.cpp.

7.34.3 Member Function Documentation

7.34.3.1 GetChanCapDecay()

```
int ECMGroup::GetChanCapDecay ( ) const
```

Returns the position of the resonant exit pair in the Decay vector. Used only for resonant external pathways.

Definition at line 84 of file ECMGroup.cpp.

7.34.3.2 GetChanCapKGroup()

```
int ECMGroup::GetChanCapKGroup ( ) const
```

Returns the resonant KGroup number. Used only for resonant external pathways.

Definition at line 92 of file ECMGroup.cpp.

7.34.3.3 GetChanCapMGroup()

```
int ECMGroup::GetChanCapMGroup ( ) const
```

Returns the resonant MGroup number. Used only for resonant external pathways.

Definition at line 100 of file ECMGroup.cpp.

7.34.3.4 GetFinalChannel()

```
int ECMGroup::GetFinalChannel ( ) const
```

Returns the final channel number for the pathway.

Definition at line 60 of file ECMGroup.cpp.

7.34.3.5 GetIntChannelNum()

```
int ECMGroup::GetIntChannelNum ( ) const
```

Returns the corresponding internal channel for an external channel pathway.

Definition at line 108 of file ECMGroup.cpp.

7.34.3.6 GetJ()

```
double ECMGroup::GetJ ( ) const
```

Returns the initial spin value for the pathway.

Definition at line 116 of file ECMGroup.cpp.

7.34.3.7 GetJGroupNum()

```
int ECMGroup::GetJGroupNum ( ) const
```

Returns the position of the final state jGroup in the JGroup vector.

Definition at line 68 of file ECMGroup.cpp.

7.34.3.8 GetL()

```
int ECMGroup::GetL ( ) const
```

Returns the initial orbital momentum value for the reaction pathway.

Definition at line 52 of file ECMGroup.cpp.

7.34.3.9 GetLevelNum()

```
int ECMGroup::GetLevelNum ( ) const
```

Returns the position of the final state in the ALevel vector.

Definition at line 76 of file ECMGroup.cpp.

7.34.3.10 GetMult()

```
int ECMGroup::GetMult ( ) const
```

Returns the multipolarity of the capture gamma.

Definition at line 44 of file ECMGroup.cpp.

7.34.3.11 GetRadType()

```
char ECMGroup::GetRadType ( ) const
```

Returns the radiation type for the capture gamma.

Definition at line 36 of file ECMGroup.cpp.

7.34.3.12 GetStatSpinFactor()

```
double ECMGroup::GetStatSpinFactor ( ) const
```

Returns the statistical spin factor, g_J, for the pathway.

Definition at line 124 of file ECMGroup.cpp.

7.34.3.13 IsChannelCapture()

```
bool ECMGroup::IsChannelCapture ( ) const
```

Returns true if the pathways is a resonant external pathway, otherwise returns false.

Definition at line 28 of file ECMGroup.cpp.

7.34.3.14 SetStatSpinFactor()

```
void ECMGroup::SetStatSpinFactor (
            double a )
```

Sets the statistical spin factor, g_J, for the pathway.

Definition at line 132 of file ECMGroup.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/ECMGroup.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/ECMGroup.cpp

7.35 EData Class Reference

An AZURE data object.

`#include <EData.h>`

Public Member Functions

- EData ()
- int NumSegments () const
- int Fill (const Config &, CNuc ∗)
- int MakePoints (const Config &, CNuc ∗)
- int Iterations () const
- int NumTargetEffects () const
- int GetNormParamOffset () const
- int ReadTargetEffectsFile (const Config &, CNuc ∗)
- bool IsFit () const
- bool IsErrorAnalysis () const
- bool IsSegmentKey (int)
- void SetFit (bool)
- void SetErrorAnalysis (bool)
- void Iterate ()
- void ResetIterations ()
- int Initialize (CNuc ∗, const Config &)
- void AddSegment (ESegment)
- void PrintData (const Config &)
- void CalcLegendreP (int)
- void PrintLegendreP (const Config &)

- int CalcEDependentValues (CNuc ∗, const Config &)
- void PrintEDependentValues (const Config &, CNuc ∗)
- void CalcCoulombAmplitude (CNuc ∗)
- void PrintCoulombAmplitude (const Config &, CNuc ∗)
- void WriteOutputFiles (const Config &, bool=false)
- int CalculateECAmplitudes (CNuc ∗, const Config &)
- void MapData ()
- void AddTargetEffect (TargetEffect)
- void SetNormParamOffset (int)
- void FillMnParams (ROOT::Minuit2::MnUserParameters &)
- void FillNormsFromParams (const vector_r &)
- void DeleteLastSegment ()
- ESegment ∗ GetSegment (int)
- ESegment ∗ GetSegmentFromKey (int)
- EData ∗ Clone () const
- TargetEffect ∗ GetTargetEffect (int)
- EDataIterator begin ()
- EDataIterator end ()
- std::vector< ESegment > & GetSegments ()

7.35.1 Detailed Description

An AZURE data object.

The EData object is the top level data object in AZURE. It is the container object for a vector of ESegment objects.

Definition at line 21 of file EData.h.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 EData()

```
EData::EData ( )
```

The EData object has a private attribute containing the number of iterations needed to find the best fit parameters. At creation, this attribute is set to 0.

Definition at line 19 of file EData.cpp.

7.35.3 Member Function Documentation

7.35.3.1 AddSegment()

```
void EData::AddSegment (
            ESegment segment )
```

Adds a segment to the ESegment vector.

Definition at line 479 of file EData.cpp.

7.35.3.2 AddTargetEffect()

```
void EData::AddTargetEffect (
             TargetEffect targetEffect )
```

Adds a TargetEffect object to the vector contained within the present object.

Definition at line 1036 of file EData.cpp.

7.35.3.3 begin()

```
EDataIterator EData::begin ( )
```

Returns an EDataIterator referring to the first data point in the set.

Definition at line 1149 of file EData.cpp.

7.35.3.4 CalcCoulombAmplitude()

```
void EData::CalcCoulombAmplitude (
             CNuc * theCNuc )
```

Calls EPoint::CalcCoulombAmplitude for each point in the entire EData object.

Definition at line 714 of file EData.cpp.

7.35.3.5 CalcEDependentValues()

```
int EData::CalcEDependentValues (
             CNuc * theCNuc,
             const Config & configure )
```

Calls EPoint::CalcEDependentValues for each point in the entire EData object.

Definition at line 631 of file EData.cpp.

7.35.3.6 CalcLegendreP()

```
void EData::CalcLegendreP (
             int maxL )
```

Calls EPoint::CalcLegendreP for each point in the entire EData object.

Definition at line 574 of file EData.cpp.

7.35.3.7 CalculateECAmplitudes()

```
int EData::CalculateECAmplitudes (
            CNuc * theCNuc,
            const Config & configure )
```

If external capture amplitudes are to be calculated, EPoint::CalculateECAmplitudes is called for each point with a corresponding external capture component in the EData object. Otherwise, the amplitudes are read from the specified file.

Definition at line 870 of file EData.cpp.

7.35.3.8 Clone()

```
EData * EData::Clone ( ) const
```

Creates a new copy of the EData object in memory and returns a pointer to the new object. Used in AZURECalc function class for thread safety.

Definition at line 1117 of file EData.cpp.

7.35.3.9 DeleteLastSegment()

```
void EData::DeleteLastSegment ( )
```

Deletes the last segment from the segment vector.

Definition at line 1069 of file EData.cpp.

7.35.3.10 end()

```
EDataIterator EData::end ( )
```

Returns an EDataIterator referring to one object past the last data point in the set.

Definition at line 1157 of file EData.cpp.

7.35.3.11 Fill()

```
int EData::Fill (
            const Config & configure,
            CNuc * theCNuc )
```

This function fills the data object with segments from the segment data files. After a segment is created, the ESegment::Fill method is called for that segment. Returns -1 if the input files could not be read, otherwise returns 0.

Definition at line 39 of file EData.cpp.

7.35.3.12 FillMnParams()

```
void EData::FillMnParams (
            ROOT::Minuit2::MnUserParameters & p )
```

Fills the Minuit parameter array from initial values in the EData object.

Definition at line 1052 of file EData.cpp.

7.35.3.13 FillNormsFromParams()

```
void EData::FillNormsFromParams (
            const vector_r & p )
```

Fills the Normalizations from the Minuit parameter array.

Definition at line 1077 of file EData.cpp.

7.35.3.14 GetNormParamOffset()

```
int EData::GetNormParamOffset ( ) const
```

Returns the offset of the normalization paramters in the Minuit parameter vector.

Definition at line 278 of file EData.cpp.

7.35.3.15 GetSegment()

```
ESegment * EData::GetSegment (
            int segmentNum )
```

Returns a pointer to a segment specified by a position in the ESegment vector.

Definition at line 1092 of file EData.cpp.

7.35.3.16 GetSegmentFromKey()

```
ESegment * EData::GetSegmentFromKey (
            int segmentKey )
```

Returns a pointer to a segment based on the segment key, as opposed to a position in the ESegment vector.

Definition at line 1101 of file EData.cpp.

7.35.3.17 GetSegments()

```
std::vector< ESegment > & EData::GetSegments ( )
```

Returns a reference to the vector of ESegment objects.

Definition at line 1166 of file EData.cpp.

7.35.3.18 GetTargetEffect()

```
TargetEffect * EData::GetTargetEffect (
            int effectNumber )
```

Returns a pointer to the specified TargetEffect object.

Definition at line 1137 of file EData.cpp.

7.35.3.19 Initialize()

```
int EData::Initialize (
            CNuc * compound,
            const Config & configure )
```

This function is identical in role to the EPoint::Initialize function, except that it initializes and entire EData object instead of a single EPoint object.

Definition at line 447 of file EData.cpp.

7.35.3.20 IsErrorAnalysis()

```
bool EData::IsErrorAnalysis ( ) const
```

Returns true if the call to function is for error analysis via Minos, otherwise returns false. Used in the AZURECalc function class to suppress transformation and file output during error analysis.

Definition at line 384 of file EData.cpp.

7.35.3.21 IsFit()

```
bool EData::IsFit ( ) const
```

Returns true if the data is to be fit, otherwise returns false. Used in the AZURECalc function class to determine if a clone of the CNuc and EData objects should be made for thread safety.

Definition at line 375 of file EData.cpp.

7.35.3.22 IsSegmentKey()

```
bool EData::IsSegmentKey (
            int segmentKey )
```

Sets the boolean indicating if the data is to be fit by AZURECalc function class. Used in AZUREMain function class before calls to Minuit and AZURECalc.

Returns true if the specified segment key exists corresponds to a segment in the ESegment vector, otherwise returns false.

Definition at line 398 of file EData.cpp.

7.35.3.23 Iterate()

```
void EData::Iterate ( )
```

This function updates the number of fit iterations per iteration during the fitting process.

Definition at line 430 of file EData.cpp.

7.35.3.24 Iterations()

```
int EData::Iterations ( ) const
```

Returns the number of fit iterations needed to minimize the parameters to the data.

Definition at line 263 of file EData.cpp.

7.35.3.25 MakePoints()

```
int EData::MakePoints (
            const Config & configure,
            CNuc * theCNuc )
```

If the AZURE calculation is not data driven, this function is called in place of the EData::Fill function to create points at specified energies and angles. Returns -1 if the input files could not be read, otherwise returns 0.

Definition at line 140 of file EData.cpp.

7.35.3.26 MapData()

```
void EData::MapData ( )
```

This function determined what points should be mapped to another to reduce redundant calculations at like energies.

Definition at line 1003 of file EData.cpp.

7.35.3.27 NumSegments()

```
int EData::NumSegments ( ) const
```

Returns the number of segment objects in the ESegment vector.

Definition at line 29 of file EData.cpp.

7.35.3.28 NumTargetEffects()

```
int EData::NumTargetEffects ( ) const
```

Returns the number of TargetEffect objects contained in the present object.

Definition at line 271 of file EData.cpp.

7.35.3.29 PrintCoulombAmplitude()

```
void EData::PrintCoulombAmplitude (
            const Config & configure,
            CNuc * theCNuc )
```

Prints the values calculated by EPoint::CalcCoulombAmplitude for each point in the entire EData object.

Definition at line 728 of file EData.cpp.

7.35.3.30 PrintData()

```
void EData::PrintData (
            const Config & configure )
```

Prints the data point after the object is filled or points are created.

Definition at line 487 of file EData.cpp.

7.35.3.31 PrintEDependentValues()

```
void EData::PrintEDependentValues (
            const Config & configure,
            CNuc * theCNuc )
```

Prints the values calculated by EPoint::CalcEDependentValues for each point in the entire EData object.

Definition at line 659 of file EData.cpp.

7.35.3.32 PrintLegendreP()

```
void EData::PrintLegendreP (
            const Config & configure )
```

Prints the Legendre polynomials for each point in the EData object.

Definition at line 591 of file EData.cpp.

7.35.3.33 ReadTargetEffectsFile()

```
int EData::ReadTargetEffectsFile (
            const Config & configure,
            CNuc * compound )
```

Reads the target effects input file and creates the TargetEffect objects to be applied to the data.

Definition at line 287 of file EData.cpp.

7.35.3.34 ResetIterations()

```
void EData::ResetIterations ( )
```

This function sets the number of iterations to zero.

Definition at line 438 of file EData.cpp.

7.35.3.35 SetErrorAnalysis()

```
void EData::SetErrorAnalysis (
             bool errorAnalysis )
```

Sets the boolean indicating if the call to the function is for error analysis via Minos.

Definition at line 422 of file EData.cpp.

7.35.3.36 SetFit()

```
void EData::SetFit (
             bool fit )
```

Sets an internal variable specifying if the data is to be fit by Minuit. Needed to determine cloning behavior in AZURECalc for thread safety.

Definition at line 414 of file EData.cpp.

7.35.3.37 SetNormParamOffset()

```
void EData::SetNormParamOffset (
             int offset )
```

Sets the normalization parameter offset in the parameter vector.

Definition at line 1044 of file EData.cpp.

7.35.3.38 WriteOutputFiles()

```
void EData::WriteOutputFiles (
             const Config & configure,
             bool isFit = false )
```

Writes the output files for the calculation. The output files are all in center of mass frame, and contain columns for energy, angle, calculated cross section, calculated s-factor, experimental cross section and error and experimental s-factor and error.

Definition at line 774 of file EData.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/EData.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/EData.cpp

7.36 EDataIterator Class Reference

An iterator class for an EData object.

`#include <EDataIterator.h>`

Public Member Functions

- EDataIterator (std::vector< ESegment > ∗)
- EDataIterator (const EDataIterator &it)
- EDataIterator & operator++ ()
- EDataIterator operator++ (int)
- bool operator== (const EDataIterator &)
- bool operator!= (const EDataIterator &)
- EDataIterator & SetEnd ()
- ESegmentIterator & segment ()
- EPointIterator & point ()

7.36.1 Detailed Description

An iterator class for an EData object.

The EDataIterator class is used to loop through all data segments and points contained in an EData object.

Definition at line 20 of file EDataIterator.h.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 EDataIterator() [1/2]

```
EDataIterator::EDataIterator (
            std::vector< ESegment > * segments )
```

The EDataIterator object is created with reference to a vector of ESegment objects.

Definition at line 9 of file EDataIterator.cpp.

7.36.2.2 EDataIterator() [2/2]

```
EDataIterator::EDataIterator (
            const EDataIterator & it )
```

The copy constructor creates a new EDataIterator object with reference to an already existing object.

Definition at line 20 of file EDataIterator.cpp.

7.36.3 Member Function Documentation

7.36.3.1 operator"!=()

```
bool EDataIterator::operator!= (
            const EDataIterator & rhs )
```

This function defines the boolean not-equal operator.

Definition at line 62 of file EDataIterator.cpp.

7.36.3.2 operator++() [1/2]

```
EDataIterator & EDataIterator::operator++ ( )
```

This function defines the prefix version of the iterating operator.

Definition at line 28 of file EDataIterator.cpp.

7.36.3.3 operator++() [2/2]

```
EDataIterator EDataIterator::operator++ (
            int )
```

This function defines the postfix version of the iterating operator.

Definition at line 44 of file EDataIterator.cpp.

7.36.3.4 operator==()

```
bool EDataIterator::operator== (
            const EDataIterator & rhs )
```

This function defines the boolean equal operator.

Definition at line 54 of file EDataIterator.cpp.

7.36.3.5 point()

```
EPointIterator & EDataIterator::point ( )
```

Returns a reference to the contained std::vector<EPoint>::iterator object.

Definition at line 89 of file EDataIterator.cpp.

7.36.3.6 segment()

`ESegmentIterator & EDataIterator::segment ( )`

Returns a reference to the contained std::vector<ESegment>::iterator object.

Definition at line 81 of file EDataIterator.cpp.

7.36.3.7 SetEnd()

`EDataIterator & EDataIterator::SetEnd ( )`

Sets the contained std::vector<ESegment>::iterator object to the refer to the last segment, and the contained std::vector<EData>::iterator object to refer to the default end() position (one-past).

Definition at line 71 of file EDataIterator.cpp.

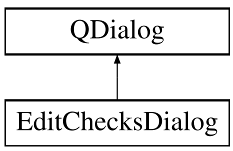
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/EDataIterator.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/EDataIterator.cpp

7.37 EditChecksDialog Class Reference

```
#include <EditChecksDialog.h>
```

Inheritance diagram for EditChecksDialog:



Public Member Functions

- EditChecksDialog (QWidget ∗parent=0)

Public Attributes

- QComboBox ∗ compoundCheckCombo
- QComboBox ∗ boundaryCheckCombo
- QComboBox ∗ dataCheckCombo
- QComboBox ∗ lMatrixCheckCombo
- QComboBox ∗ legendreCheckCombo
- QComboBox ∗ coulAmpCheckCombo
- QComboBox ∗ pathwaysCheckCombo
- QComboBox ∗ angDistsCheckCombo

7.37.1 Detailed Description

Definition at line 14 of file EditChecksDialog.h.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 EditChecksDialog()

```
EditChecksDialog::EditChecksDialog (
            QWidget * parent = 0 )
```

Definition at line 8 of file EditChecksDialog.cpp.

7.37.3 Member Data Documentation

7.37.3.1 angDistsCheckCombo

```
QComboBox* EditChecksDialog::angDistsCheckCombo
```

Definition at line 27 of file EditChecksDialog.h.

7.37.3.2 boundaryCheckCombo

```
QComboBox* EditChecksDialog::boundaryCheckCombo
```

Definition at line 21 of file EditChecksDialog.h.

7.37.3.3 compoundCheckCombo

```
QComboBox* EditChecksDialog::compoundCheckCombo
```

Definition at line 20 of file EditChecksDialog.h.

7.37.3.4 coulAmpCheckCombo

```
QComboBox* EditChecksDialog::coulAmpCheckCombo
```

Definition at line 25 of file EditChecksDialog.h.

7.37.3.5 dataCheckCombo

```
QComboBox* EditChecksDialog::dataCheckCombo
```

Definition at line 22 of file EditChecksDialog.h.

7.37.3.6 legendreCheckCombo

```
QComboBox* EditChecksDialog::legendreCheckCombo
```

Definition at line 24 of file EditChecksDialog.h.

7.37.3.7 lMatrixCheckCombo

```
QComboBox* EditChecksDialog::lMatrixCheckCombo
```

Definition at line 23 of file EditChecksDialog.h.

7.37.3.8 pathwaysCheckCombo

```
QComboBox* EditChecksDialog::pathwaysCheckCombo
```

Definition at line 26 of file EditChecksDialog.h.

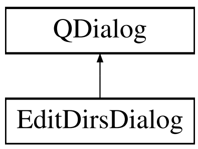
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditChecksDialog.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditChecksDialog.cpp

7.38 EditDirsDialog Class Reference

```
#include <EditDirsDialog.h>
```

Inheritance diagram for EditDirsDialog:



Public Member Functions

- EditDirsDialog (QWidget *parent=0)

Public Attributes

- QLineEdit * outputDirectoryText
- QLineEdit * checksDirectoryText

7.38.1 Detailed Description

Definition at line 15 of file EditDirsDialog.h.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 EditDirsDialog()

```
EditDirsDialog::EditDirsDialog (
            QWidget * parent = 0 )
```

Definition at line 10 of file EditDirsDialog.cpp.

7.38.3 Member Data Documentation

7.38.3.1 checksDirectoryText

```
QLineEdit* EditDirsDialog::checksDirectoryText
```

Definition at line 22 of file EditDirsDialog.h.

7.38.3.2 outputDirectoryText

```
QLineEdit* EditDirsDialog::outputDirectoryText
```

Definition at line 21 of file EditDirsDialog.h.

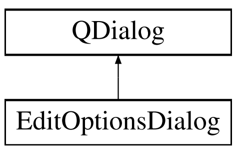
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditDirsDialog.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditDirsDialog.cpp

7.39 EditOptionsDialog Class Reference

```
#include <EditOptionsDialog.h>
```

Inheritance diagram for EditOptionsDialog:



Public Member Functions

- EditOptionsDialog (QWidget ∗parent=0)

Public Attributes

- QCheckBox * useBruneCheck
- QCheckBox * useGSLCoulCheck
- QCheckBox * ignoreExternalsCheck
- QCheckBox * useRMCCheck
- QCheckBox * noTransformCheck

7.39.1 Detailed Description

Definition at line 17 of file EditOptionsDialog.h.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 EditOptionsDialog()

```
EditOptionsDialog::EditOptionsDialog (
            QWidget * parent = 0 )
```

Definition at line 8 of file EditOptionsDialog.cpp.

7.39.3 Member Data Documentation

7.39.3.1 ignoreExternalsCheck

```
QCheckBox* EditOptionsDialog::ignoreExternalsCheck
```

Definition at line 25 of file EditOptionsDialog.h.

7.39.3.2 noTransformCheck

```
QCheckBox* EditOptionsDialog::noTransformCheck
```

Definition at line 27 of file EditOptionsDialog.h.

7.39.3.3 useBruneCheck

```
QCheckBox* EditOptionsDialog::useBruneCheck
```

Definition at line 23 of file EditOptionsDialog.h.

7.39.3.4 useGSLCoulCheck

```
QCheckBox* EditOptionsDialog::useGSLCoulCheck
```

Definition at line 24 of file EditOptionsDialog.h.

7.39.3.5 useRMCCheck

```
QCheckBox* EditOptionsDialog::useRMCCheck
```

Definition at line 26 of file EditOptionsDialog.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditOptionsDialog.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditOptionsDialog.cpp

7.40 EffectiveCharge Class Reference

A function class for calculating effective charge without long-wavelength approximation.

```
#include <EffectiveCharge.h>
```

Public Member Functions

- EffectiveCharge (PPair *, double, int)
- double operator() (double)

7.40.1 Detailed Description

A function class for calculating effective charge without long-wavelength approximation.

The EffectiveCharge class calculates the effective charge needed for external capture without using the long wavelength approximation. The formalism is based on J.L. Friar and S. Fallieros, Phys. Rev. C 29, 1645 (1984).

Definition at line 14 of file EffectiveCharge.h.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 EffectiveCharge()

```
EffectiveCharge::EffectiveCharge (
            PPair * pair,
            double energy,
            int L )
```

The constructor takes a PPair object, gamma energy, and multipolarity as arguments.

Definition at line 13 of file EffectiveCharge.cpp.

7.40.3 Member Function Documentation

7.40.3.1 operator()()

```
double EffectiveCharge::operator() (
            double r )
```

The operator() function takes the radius as an argument, carries out integration, and returns the rho-dependent effective charge.

Definition at line 28 of file EffectiveCharge.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/EffectiveCharge.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/EffectiveCharge.cpp

7.41 EigenFunc Class Reference

A function class to solve a eigenvalue problems.

```
#include <EigenFunc.h>
```

Public Member Functions

- EigenFunc (const matrix_r &)
- EigenFunc (const matrix_r &, const std::vector< vector_r > &)
- const vector_r & eigenvalues () const
- const matrix_r & eigenvectors () const

7.41.1 Detailed Description

A function class to solve a eigenvalue problems.

The EigenFunc class is a function class to solve eigenfunction problems.
The class is used by the parameter trasfomation subroutines in AZURE2.

Definition at line 13 of file EigenFunc.h.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 EigenFunc() [1/2]

```
EigenFunc::EigenFunc (
            const matrix_r & A )
```

This constructor calculates the eigenvectors and eigenvalues for a real symmetric matrix.

Definition at line 9 of file EigenFunc.cpp.

7.41.2.2 EigenFunc() [2/2]

```
EigenFunc::EigenFunc (
            const matrix_r & A,
            const std::vector< vector_r > & B )
```

This constructor calulates the eigenvectors and eigenvalues for a real symmetric matrix pair.

Definition at line 47 of file EigenFunc.cpp.

7.41.3 Member Function Documentation

7.41.3.1 eigenvalues()

```
const vector_r & EigenFunc::eigenvalues ( ) const [inline]
```

Returns a vector with the calculated eigenvalues.

Definition at line 20 of file EigenFunc.h.

7.41.3.2 eigenvectors()

```
const matrix_r & EigenFunc::eigenvectors ( ) const [inline]
```

Returns a matrix with the calculated eigenvectors.

Definition at line 24 of file EigenFunc.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/EigenFunc.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/EigenFunc.cpp

7.42 EnergyMap Struct Reference

A container structure for a reference to a data point.

```
#include <EPoint.h>
```

Public Attributes

- int segment

    The segment index for the map.
- int point

    The point index for the map.

7.42.1 Detailed Description

A container structure for a reference to a data point.

If a point is mapped back to another in the calculation, this structure hold the relevant indices of the map point.

Definition at line 13 of file EPoint.h.

7.42.2 Member Data Documentation

7.42.2.1 point

```
int EnergyMap::point
```

The point index for the map.

Definition at line 17 of file EPoint.h.

7.42.2.2 segment

```
int EnergyMap::segment
```

The segment index for the map.

Definition at line 15 of file EPoint.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/EPoint.h

7.43 EPoint Class Reference

An AZURE data point.

```
#include <EPoint.h>
```

Public Member Functions

- EPoint (DataLine, ESegment ∗)
- EPoint (double, double, ESegment ∗)
- EPoint (double, double, int, int, bool, bool, bool, double, int, int)
- bool IsDifferential () const
- bool IsPhase () const
- bool IsMapped () const
- bool IsTargetEffect () const
- bool IsAngularDist () const
- int GetEntranceKey () const
- int GetExitKey () const
- int GetMaxLOrder () const
- int GetL () const
- int NumLocalMappedPoints () const
- int NumSubPoints () const
- int GetTargetEffectNum () const
- int GetMaxAngDistOrder () const
- int GetNumAngularDists () const
- double GetLabAngle () const
- double GetCMAngle () const
- double GetLabEnergy () const
- double GetCMEnergy () const
- double GetExcitationEnergy () const
- double GetLegendreP (int) const
- double GetLabCrossSection () const
- double GetCMCrossSection () const
- double GetLabCrossSectionError () const
- double GetCMCrossSectionError () const
- double GetGeometricalFactor () const
- double GetFitCrossSection () const
- double GetSFactorConversion () const
- double GetSqrtPenetrability (int, int) const
- double GetJ () const
- double GetStoppingPower () const
- double GetTargetThickness () const
- double GetAngularDist (int) const
- complex GetLoElement (int, int) const
- complex GetExpCoulombPhase (int, int) const
- complex GetExpHardSpherePhase (int, int) const
- complex GetCoulombAmplitude () const
- complex GetECAmplitude (int, int) const
- EnergyMap GetMap () const
- void Initialize (CNuc ∗, const Config &)
- void ConvertLabEnergy (PPair ∗)
- void ConvertDecayEnergy (PPair ∗)
- void ConvertLabAngle (PPair ∗)
- void ConvertLabAngle (PPair ∗, PPair ∗, const Config &)
- void ConvertCrossSection (PPair ∗, PPair ∗)
- void AddLegendreP (double)
- void SetGeometricalFactor (double)
- void SetFitCrossSection (double)
- void SetSFactorConversion (double)
- void SetExitKey (int)
- void CalcLegendreP (int, TargetEffect ∗)

- void CalcEDependentValues (CNuc ∗, const Config &)
- void AddLoElement (int, int, complex)
- void AddSqrtPenetrability (int, int, double)
- void AddExpCoulombPhase (int, int, complex)
- void AddExpHardSpherePhase (int, int, complex)
- void CalcCoulombAmplitude (CNuc ∗)
- void SetCoulombAmplitude (complex)
- void CalculateECAmplitudes (CNuc ∗, const Config &)
- void AddECAmplitude (int, int, complex)
- void Calculate (CNuc ∗, const Config &configure, EPoint ∗parent=NULL, int subPointNum=0)
- void SetMap (int, int)
- void AddLocalMappedPoint (EPoint ∗)
- void ClearLocalMappedPoints ()
- void SetTargetEffectNum (int)
- void AddSubPoint (EPoint)
- void IntegrateTargetEffect ()
- void SetParentData (EData ∗)
- void SetStoppingPower (double)
- void SetTargetThickness (double)
- void SetAngularDists (vector_r)
- EData ∗ GetParentData () const
- EPoint ∗ GetLocalMappedPoint (int) const
- EPoint ∗ GetSubPoint (int)
- std::vector< EPoint > & GetSubPoints ()
- std::vector< EPoint ∗ > & GetMappedPoints ()

7.43.1 Detailed Description

An AZURE data point.

A data point object in AZURE consists of a defined entrance and exit pair, an energy, an angle, measured cross section and uncertainty, s-factor conversions, and several flags that determine the type of data (angle integrated or differential) to be analysed.

Definition at line 36 of file EPoint.h.

7.43.2 Constructor & Destructor Documentation

7.43.2.1 EPoint() [1/3]

```
EPoint::EPoint (
            DataLine dataLine,
            ESegment * parent )
```

This constructor is used if the data point is to be created from a line in a data file. A pointer to the parent segment is passed to the constructor for the intialization of the EPoint object.

Definition at line 23 of file EPoint.cpp.

7.43.2.2 EPoint() [2/3]

```
EPoint::EPoint (
            double angle,
            double energy,
            ESegment * parent )
```

This constructor is used if the data point is to be created with no experimental data. A pointer to the parent segment is passed to the constructor for the intialization of the EPoint object.

Definition at line 56 of file EPoint.cpp.

7.43.2.3 EPoint() [3/3]

```
EPoint::EPoint (
            double angle,
            double energy,
            int entranceKey,
            int exitKey,
            bool isDifferential,
            bool isPhase,
            bool isAngularDist,
            double jValue,
            int lValue,
            int maxAngDistOrder )
```

This constructor is used if the data point is to be created with no experimental data and no parent segment. Such a point is used if AZURE is to be called as an energy- dependent function for reaction rate and target effect calculations with dynamic integration. All options must be set manually.

Definition at line 90 of file EPoint.cpp.

7.43.3 Member Function Documentation

7.43.3.1 AddECAmplitude()

```
void EPoint::AddECAmplitude (
            int kGroupNum,
            int ecMGroupNum,
            complex ecAmplitude )
```

Adds an external capture amplitude with reference to a specified reaction pathway.

Definition at line 911 of file EPoint.cpp.

7.43.3.2 AddExpCoulombPhase()

```
void EPoint::AddExpCoulombPhase (
            int jGroupNum,
            int channelNum,
            complex expShift )
```

Adds an exponential of the Coulomb phase shift with reference to positions in the JGroup and subsequent AChannel vectors.

Definition at line 777 of file EPoint.cpp.

7.43.3.3 AddExpHardSpherePhase()

```
void EPoint::AddExpHardSpherePhase (
            int jGroupNum,
            int channelNum,
            complex expShift )
```

Adds an exponential of the hard sphere phase shift with reference to positions in the JGroup and subsequent AChannel vectors.

Definition at line 789 of file EPoint.cpp.

7.43.3.4 AddLegendreP()

```
void EPoint::AddLegendreP (
            double polynomial )
```

Adds a Legendre polynomial to the vector. Polynomials are added to the vector in the order as L=0,1,2,...

Definition at line 566 of file EPoint.cpp.

7.43.3.5 AddLocalMappedPoint()

```
void EPoint::AddLocalMappedPoint (
            EPoint * point )
```

If a point is mapped to the current point, a pointer to the mapped point is added to a vector.

Definition at line 975 of file EPoint.cpp.

7.43.3.6 AddLoElement()

```
void EPoint::AddLoElement (
            int jGroupNum,
            int channelNum,
            complex loElement )
```

Adds an L_o matrix element with reference to positions in the JGroup and subsequent AChannel vectors.

Definition at line 753 of file EPoint.cpp.

7.43.3.7 AddSqrtPenetrability()

```
void EPoint::AddSqrtPenetrability (
            int jGroupNum,
            int channelNum,
            double sqrtPene )
```

Adds a square root of penetrability with reference to positions in the JGroup and subsequent AChannel vectors.

Definition at line 765 of file EPoint.cpp.

7.43.3.8 AddSubPoint()

```
void EPoint::AddSubPoint (
            EPoint subPoint )
```

Adds a sub-point to the current point object for target effect integration.

Definition at line 999 of file EPoint.cpp.

7.43.3.9 CalcCoulombAmplitude()

```
void EPoint::CalcCoulombAmplitude (
            CNuc * theCNuc )
```

Calculates the Coulomb amplitude C_α for the data point.

Definition at line 800 of file EPoint.cpp.

7.43.3.10 CalcEDependentValues()

```
void EPoint::CalcEDependentValues (
            CNuc * theCNuc,
            const Config & configure )
```

This function calculates several energy dependent quantities simultaniously. This includes the geometrical cross section, the s-factor conversion, the L_o matrix elements, the square root of the penetrability, and the exponentials of the Coulomb phase shifts and hard sphere phase shfits.

Definition at line 643 of file EPoint.cpp.

7.43.3.11 CalcLegendreP()

```
void EPoint::CalcLegendreP (
            int maxL,
            TargetEffect * targetEffect )
```

Calculates Legendre polynomials up to a maximum order. The polynomials are added, in order, to a vector.

Definition at line 606 of file EPoint.cpp.

7.43.3.12 Calculate()

```
void EPoint::Calculate (
            CNuc * theCNuc,
            const Config & configure,
            EPoint * parent = NULL,
            int subPointNum = 0 )
```

Calculates the cross section for a data point based on the fit parameters in the compound nucleus.

Definition at line 922 of file EPoint.cpp.

7.43.3.13 CalculateECAmplitudes()

```
void EPoint::CalculateECAmplitudes (
            CNuc * theCNuc,
            const Config & configure )
```

Calculates the external capture amplitudes for the data point. The amplitudes are calculated for all reaction pathways with corresponding entrance and exit pairs.

Definition at line 834 of file EPoint.cpp.

7.43.3.14 ClearLocalMappedPoints()

```
void EPoint::ClearLocalMappedPoints ( )
```

Clears vector containing pointers to points mapped to the current point.

Definition at line 983 of file EPoint.cpp.

7.43.3.15 ConvertCrossSection()

```
void EPoint::ConvertCrossSection (
            PPair * entrancePair,
            PPair * exitPair )
```

Calculates center of mass cross sections. When a data point is initialized, the same cross section is copied into the attributes for center of mass and lab cross section. If this function is called, the center of mass cross section attribute is overwritten with the value calculated from the lab cross section attribute.

Definition at line 538 of file EPoint.cpp.

7.43.3.16 ConvertDecayEnergy()

```
void EPoint::ConvertDecayEnergy (
            PPair * pPair )
```

Calculates the total decay energy from the light particle decay energy, assuming the parent nucleus was at rest when it decayed. When a data point is initialized, the same energy is copied into the attributes for center of mass and lab energy. If this function is called, the center of mass energy attribute is overwritten with the value calculated from the lab energy attribute.

Definition at line 482 of file EPoint.cpp.

7.43.3.17 ConvertLabAngle() [1/2]

```
void EPoint::ConvertLabAngle (
            PPair * pPair )
```

Calculates center of mass angles. When a data point is initialized, the same angle is copied into the attributes for center of mass and lab angles. If this function is called, the center of mass angle attribute is overwritten with the value calculated from the lab angle attribute. This version of the overloaded function is for scattering channels.

Definition at line 496 of file EPoint.cpp.

7.43.3.18 ConvertLabAngle() [2/2]

```
void EPoint::ConvertLabAngle (
            PPair * entrancePair,
            PPair * exitPair,
            const Config & configure )
```

Calculates center of mass angles. When a data point is initialized, the same angle is copied into the attributes for center of mass and lab angles. If this function is called, the center of mass angle attribute is overwritten with the value calculated from the lab angle attribute. This version of the overloaded function is for non-elastic particle channels.

Definition at line 507 of file EPoint.cpp.

7.43.3.19 ConvertLabEnergy()

```
void EPoint::ConvertLabEnergy (
            PPair * pPair )
```

Calculates center of mass energy. When a data point is initialized, the same energy is copied into the attributes for center of mass and lab energy. If this function is called, the center of mass energy attribute is overwritten with the value calculated from the lab energy attribute.

Definition at line 468 of file EPoint.cpp.

7.43.3.20 GetAngularDist()

```
double EPoint::GetAngularDist (
            int order ) const
```

Returns the angular distribution coefficient corresponding to the given order;

Definition at line 382 of file EPoint.cpp.

7.43.3.21 GetCMAngle()

```
double EPoint::GetCMAngle ( ) const
```

Returns the angle of the data point in the center of mass frame.

Definition at line 251 of file EPoint.cpp.

7.43.3.22 GetCMCrossSection()

```
double EPoint::GetCMCrossSection ( ) const
```

Returms the experimental cross section in the center of mass frame.

Definition at line 300 of file EPoint.cpp.

7.43.3.23 GetCMCrossSectionError()

```
double EPoint::GetCMCrossSectionError ( ) const
```

Returns the experimental uncertainty in the center of mass frame.

Definition at line 316 of file EPoint.cpp.

7.43.3.24 GetCMEnergy()

```
double EPoint::GetCMEnergy ( ) const
```

Returns the energy of the point in the center of mass frame.

Definition at line 267 of file EPoint.cpp.

7.43.3.25 GetCoulombAmplitude()

```
complex EPoint::GetCoulombAmplitude ( ) const
```

Returns the Coulomb amplitude C_α for the data point.

Definition at line 419 of file EPoint.cpp.

7.43.3.26 GetECAmplitude()

```
complex EPoint::GetECAmplitude (
            int kGroupNum,
            int ecMGroupNum ) const
```

Returns the external capture amplitude for a given external reaction pathway specified by positions in the KGroup and subsequent ECMGroup vectors.

Definition at line 428 of file EPoint.cpp.

7.43.3.27 GetEntranceKey()

```
int EPoint::GetEntranceKey ( ) const
```

Returns the entrance particle pair key of the data point. The key need not be the same as the position of the pair in the PPair vector. Pair keys are used in the setup files of AZURE.

Definition at line 168 of file EPoint.cpp.

7.43.3.28 GetExcitationEnergy()

```
double EPoint::GetExcitationEnergy ( ) const
```

Returns the energy of the point in compound excitation energy.

Definition at line 275 of file EPoint.cpp.

7.43.3.29 GetExitKey()

```
int EPoint::GetExitKey ( ) const
```

Returns the exit particle pair key of the data point. The key need not be the same as the position of the pair in the PPair vector. Pair keys are used in the setup files of AZURE.

Definition at line 178 of file EPoint.cpp.

7.43.3.30 GetExpCoulombPhase()

```
complex EPoint::GetExpCoulombPhase (
            int jGroupNum,
            int channelNum ) const
```

Returns the factor $\exp(\omega_c)$ where ω_c is the Coulomb phase shift. The channel us specified by positions in the JGroup and subsequent AChannel vectors.

Definition at line 401 of file EPoint.cpp.

7.43.3.31 GetExpHardSpherePhase()

```
complex EPoint::GetExpHardSpherePhase (
            int jGroupNum,
            int channelNum ) const
```

Returns the factor $\exp(\delta_c)$ where δ_c is the hard sphere phase shift. The channel us specified by positions in the JGroup and subsequent AChannel vectors.

Definition at line 411 of file EPoint.cpp.

7.43.3.32 GetFitCrossSection()

```
double EPoint::GetFitCrossSection ( ) const
```

Returns the calculated AZURE cross section.

Definition at line 332 of file EPoint.cpp.

7.43.3.33 GetGeometricalFactor()

```
double EPoint::GetGeometricalFactor ( ) const
```

Returns the geometrical cross section factor $\frac{\pi}{k^2}$.

Definition at line 324 of file EPoint.cpp.

7.43.3.34 GetJ()

```
double EPoint::GetJ ( ) const
```

Returns the total spin value for the data point. Only applies if the point is phase shift.

Definition at line 358 of file EPoint.cpp.

7.43.3.35 GetL()

```
int EPoint::GetL ( ) const
```

Returns the orbital angular momentum value for the point. Only applies if the point is phase shift.

Definition at line 195 of file EPoint.cpp.

7.43.3.36 GetLabAngle()

```
double EPoint::GetLabAngle ( ) const
```

Returns the angle of the data point in the laboratory frame.

Definition at line 243 of file EPoint.cpp.

7.43.3.37 GetLabCrossSection()

```
double EPoint::GetLabCrossSection ( ) const
```

Returns the experimental cross section in the laboratory frame.

Definition at line 292 of file EPoint.cpp.

7.43.3.38 GetLabCrossSectionError()

```
double EPoint::GetLabCrossSectionError ( ) const
```

Returns the experimental uncertainty in the laboratory frame.

Definition at line 308 of file EPoint.cpp.

7.43.3.39 GetLabEnergy()

```
double EPoint::GetLabEnergy ( ) const
```

Returns the energy of the point in the laboratory frame.

Definition at line 259 of file EPoint.cpp.

7.43.3.40 GetLegendreP()

```
double EPoint::GetLegendreP (
            int lOrder ) const
```

Returns the Legendre polynomial specified by an order.

Definition at line 284 of file EPoint.cpp.

7.43.3.41 GetLocalMappedPoint()

```
EPoint * EPoint::GetLocalMappedPoint (
            int mappedPointNum ) const
```

Returns a pointer to a point mapped to the current point specified by a position in the mapped point vector.

Definition at line 1152 of file EPoint.cpp.

7.43.3.42 GetLoElement()

```
complex EPoint::GetLoElement (
            int jGroupNum,
            int channelNum ) const
```

Returns the L_o diagonal matrix element for a channel specified by positions in the JGroup and subsequent AChannel vectors.

Definition at line 391 of file EPoint.cpp.

7.43.3.43 GetMap()

```
EnergyMap EPoint::GetMap ( ) const
```

If a point is mapped, returns an EnergyMap structure containing the point to which it is mapped.

Definition at line 436 of file EPoint.cpp.

7.43.3.44 GetMappedPoints()

```
std::vector< EPoint * > & EPoint::GetMappedPoints ( )
```

Returns a reference to the vector of pointers to mapped EPoint objects.

Definition at line 1180 of file EPoint.cpp.

7.43.3.45 GetMaxAngDistOrder()

```
int EPoint::GetMaxAngDistOrder ( ) const
```

Returns the maximum polynomial order of the point is angular distribution.

Definition at line 227 of file EPoint.cpp.

7.43.3.46 GetMaxLOrder()

```
int EPoint::GetMaxLOrder ( ) const
```

The maximum order of the Legendre polynomials stored in the point object.

Definition at line 186 of file EPoint.cpp.

7.43.3.47 GetNumAngularDists()

```
int EPoint::GetNumAngularDists ( ) const
```

Return the number of angular distribution coefficients in the vector.

Definition at line 235 of file EPoint.cpp.

7.43.3.48 GetParentData()

```
EData * EPoint::GetParentData ( ) const
```

Returns a pointer to the parent EData object.

Definition at line 1144 of file EPoint.cpp.

7.43.3.49 GetSFactorConversion()

```
double EPoint::GetSFactorConversion ( ) const
```

Returns the multiplicative conversion factor from cross section to s-factor.

Definition at line 340 of file EPoint.cpp.

7.43.3.50 GetSqrtPenetrability()

```
double EPoint::GetSqrtPenetrability (
            int jGroupNum,
            int channelNum ) const
```

Returns the square root of the penetrability for a channel specified by positions in the JGroup and subsequent AChannel vectors.

Definition at line 349 of file EPoint.cpp.

7.43.3.51 GetStoppingPower()

```
double EPoint::GetStoppingPower ( ) const
```

For target integration to fit yield curves, the stopping power (or, rather, stopping cross section) is calculated and stored for each sub-point. This function returns the precalculated value.

Definition at line 366 of file EPoint.cpp.

7.43.3.52 GetSubPoint()

```
EPoint * EPoint::GetSubPoint (
             int subPoint )
```

Returns a pointer to the specified sub-point in the current EPoint object.

Definition at line 1160 of file EPoint.cpp.

7.43.3.53 GetSubPoints()

```
std::vector< EPoint > & EPoint::GetSubPoints ( )
```

Returns a reference to the vector of EPoint objects containing the subpoints used in target effect integration.

Definition at line 1172 of file EPoint.cpp.

7.43.3.54 GetTargetEffectNum()

```
int EPoint::GetTargetEffectNum ( ) const
```

Returns the position of the corresponding TargetEffect object in the parent EData object.

Definition at line 219 of file EPoint.cpp.

7.43.3.55 GetTargetThickness()

```
double EPoint::GetTargetThickness ( ) const
```

Returns the energy loss of the beam in the target for the current EPoint object.

Definition at line 374 of file EPoint.cpp.

7.43.3.56 Initialize()

```
void EPoint::Initialize (
             CNuc * compound,
             const Config & configure )
```

Initializes a data point to be used in a calculation. Initilization is done before the fitting process to calculate all energy dependent quantities that do no rely on the R-Matrix fit parameters.

Definition at line 454 of file EPoint.cpp.

7.43.3.57 IntegrateTargetEffect()

```
void EPoint::IntegrateTargetEffect ( )
```

This function is called to integrate the vector of sub-points to determine the yield considering a given target effect. The function uses Simpson's rule to perform the integration.

Definition at line 1009 of file EPoint.cpp.

7.43.3.58 IsAngularDist()

```
bool EPoint::IsAngularDist ( ) const
```

Returns true if the point is angular distribution, otherwise returns false.

Definition at line 138 of file EPoint.cpp.

7.43.3.59 IsDifferential()

```
bool EPoint::IsDifferential ( ) const
```

Returns true if the point is differential cross section, otherwise returns false.

Definition at line 122 of file EPoint.cpp.

7.43.3.60 IsMapped()

```
bool EPoint::IsMapped ( ) const
```

Returns true if the point is a mapped point, otherwise returns false. Mapping in AZURE is performed so calculations are not redundantly performed for like energies. Energy dependent quantities are calculated only once for a given energy, and then copied to mapped points.

Definition at line 149 of file EPoint.cpp.

7.43.3.61 IsPhase()

```
bool EPoint::IsPhase ( ) const
```

Returns true if the point is phase shift, otherwise returns false.

Definition at line 130 of file EPoint.cpp.

7.43.3.62 IsTargetEffect()

```
bool EPoint::IsTargetEffect ( ) const
```

This function retruns true if the point has a corresponding TargetEffect object, otherwise it returns false.

Definition at line 157 of file EPoint.cpp.

7.43.3.63 NumLocalMappedPoints()

```
int EPoint::NumLocalMappedPoints ( ) const
```

Returns the number of points mapped to the current point.

Definition at line 203 of file EPoint.cpp.

7.43.3.64 NumSubPoints()

```
int EPoint::NumSubPoints ( ) const
```

Returns the total number of sub-points contained within the present objet. The sub-points are used to calculate the target effect integrals.

Definition at line 211 of file EPoint.cpp.

7.43.3.65 SetAngularDists()

```
void EPoint::SetAngularDists (
            vector_r dists )
```

Sets the angular distribution coefficients.

Definition at line 1135 of file EPoint.cpp.

7.43.3.66 SetCoulombAmplitude()

```
void EPoint::SetCoulombAmplitude (
            complex amplitude )
```

Sets the Coulomb amplitude C_α for the data point.

Definition at line 824 of file EPoint.cpp.

7.43.3.67 SetExitKey()

```
void EPoint::SetExitKey (
            int key )
```

Sets the exit key to the given value;

Definition at line 598 of file EPoint.cpp.

7.43.3.68 SetFitCrossSection()

```
void EPoint::SetFitCrossSection (
            double crossSection )
```

Sets the calculated AZURE cross section.

Definition at line 582 of file EPoint.cpp.

7.43.3.69 SetGeometricalFactor()

```
void EPoint::SetGeometricalFactor (
            double geoFactor )
```

Sets the geometrical cross section factor $\frac{\pi}{k^2}$.

Definition at line 574 of file EPoint.cpp.

7.43.3.70 SetMap()

```
void EPoint::SetMap (
            int segmentNum,
            int pointNum )
```

If a point is mapped, sets the internal attribute indicating which point it is mapped to.

Definition at line 965 of file EPoint.cpp.

7.43.3.71 SetParentData()

```
void EPoint::SetParentData (
            EData * parentData )
```

This function sets an internal pointer to the parent EData object.

Definition at line 1110 of file EPoint.cpp.

7.43.3.72 SetSFactorConversion()

```
void EPoint::SetSFactorConversion (
            double conversion )
```

Sets the multiplicative conversion from cross section to s-factor.

Definition at line 590 of file EPoint.cpp.

7.43.3.73 SetStoppingPower()

```
void EPoint::SetStoppingPower (
            double stoppingPower )
```

This function sets the stopping cross section (effective) for the current EPoint object. Used only for sub-point involoved in target effect integration.

Definition at line 1119 of file EPoint.cpp.

7.43.3.74 SetTargetEffectNum()

```
void EPoint::SetTargetEffectNum (
            int targetEffectNum )
```

Sets the position of the corresponding TargetEffect object in the parent EData object.

Definition at line 991 of file EPoint.cpp.

7.43.3.75 SetTargetThickness()

```
void EPoint::SetTargetThickness (
            double targetThickness )
```

This functions sets the energy loss of the beam in the target for the current EPoint object.

Definition at line 1127 of file EPoint.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/EPoint.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/EPoint.cpp

7.44 Equation Class Reference

A class for parsing algebraic expressions.

```
#include <Equation.h>
```

Public Member Functions

- Equation ()
- Equation (std::string equation, int numParams, const Config &)
- Equation (std::string equation, std::vector< double > parameters, const Config &)
- Equation (std::string equation, double parameters[ ], size_t arraySize, const Config &)
- void Initialize (std::string equation, int numParams, const Config &)
- void SetParameter (unsigned int index, double value, const Config &)
- std::vector< double > GetParameters () const
- double Evaluate (const Config &, double x=0.0) const

7.44.1 Detailed Description

A class for parsing algebraic expressions.

The Equation class is used in AZURE to parametrize stopping cross sections as a function of energy. An arbitrary equation can be given, and parameters specified, which is then parsed and evaluated on the fly whenever stopping cross section is needed.

Definition at line 49 of file Equation.h.

7.44.2 Constructor & Destructor Documentation

7.44.2.1 Equation() [1/4]

```
Equation::Equation ( )
```

Empty constructor.

Definition at line 11 of file Equation.cpp.

7.44.2.2 Equation() [2/4]

```
Equation::Equation (
            std::string equation,
            int numParams,
            const Config & configure )
```

This constructor is used to create an Equation object with a equation string, and a specified number of parameters. The parameters are only initialized, and must be set with the Equation::SetParameter() function.

Definition at line 20 of file Equation.cpp.

7.44.2.3 Equation() [3/4]

```
Equation::Equation (
            std::string equation,
            std::vector< double > parameters,
            const Config & configure )
```

This constructor is used to create and Equation object with an equation string and an already created parameter vector. The size of the parameter vector must correspond to the number of parameters in the equation string.

Definition at line 40 of file Equation.cpp.

7.44.2.4 Equation() [4/4]

```
Equation::Equation (
            std::string equation,
            double parameters[],
            size_t arraySize,
            const Config & configure )
```

This constructor is used to create an Equation object with an equation string and an array of parameters. The array must be of the size corresponding to the number of parameters in the equation string.

Definition at line 56 of file Equation.cpp.

7.44.3 Member Function Documentation

7.44.3.1 Evaluate()

```
double Equation::Evaluate (
            const Config & configure,
            double x = 0.0 ) const
```

Evaluates the Equation object for a specified dependent variable.

Definition at line 491 of file Equation.cpp.

7.44.3.2 GetParameters()

```
std::vector< double > Equation::GetParameters ( ) const
```

Returns the vector containing all the parameters in the Equation object.

Definition at line 200 of file Equation.cpp.

7.44.3.3 Initialize()

```
void Equation::Initialize (
            std::string equation,
            int numParams,
            const Config & configure )
```

If the empty constructor was used to create the Equation object, this function can be used to set the equation string and initialize the parameter array. The parameters must be set manually.

Definition at line 73 of file Equation.cpp.

7.44.3.4 SetParameter()

```
void Equation::SetParameter (
            unsigned int index,
            double value,
            const Config & configure )
```

Sets the specified parameter.

Definition at line 208 of file Equation.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/Equation.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/Equation.cpp

7.45 ESegment Class Reference

An AZURE data segment.

```
#include <ESegment.h>
```

Public Member Functions

- ESegment (SegLine)
- ESegment (ExtrapLine)
- bool IsInSegment (EPoint)
- bool IsDifferential () const
- bool IsPhase () const
- bool IsTargetEffect () const
- bool IsVaryNorm () const
- bool IsAngularDist () const
- int IsTotalCapture () const
- int NumPoints () const
- int GetEntranceKey () const
- int GetExitKey () const
- int Fill (CNuc *, EData *, const Config &)
- int GetL () const
- int GetTargetEffectNum () const
- int GetSegmentKey () const
- int GetMaxAngDistOrder () const
- double GetMinEnergy () const
- double GetMaxEnergy () const
- double GetMinAngle () const
- double GetMaxAngle () const
- double GetSegmentChiSquared () const
- double GetEStep () const
- double GetAStep () const
- double GetJ () const
- double GetNorm () const
- double GetNominalNorm () const
- double GetNormError () const
- std::string GetDataFile () const
- void AddPoint (EPoint)
- void SetSegmentChiSquared (double)
- void SetTargetEffectNum (int)
- void SetSegmentKey (int)
- void SetNorm (double)
- void SetExitKey (int)
- void SetIsTotalCapture (int)
- void SetVaryNorm (bool)
- EPoint * GetPoint (int)
- std::vector< EPoint > & GetPoints ()

7.45.1 Detailed Description

An AZURE data segment.

An AZURE data segment is specified by an entrance and exit particle pair key, as well as a range of energy and angle value and a data file name. The segment also contains flags specifing they type of data point it contains. The ESegment object is the container object for a vector of EData objects.

Definition at line 17 of file ESegment.h.

7.45.2 Constructor & Destructor Documentation

7.45.2.1 ESegment() [1/2]

```
ESegment::ESegment (
            SegLine segLine )
```

This constructor is used if the segment contains actual experimental data. The segment is created with reference to an entry in the segments data file.

Definition at line 13 of file ESegment.cpp.

7.45.2.2 ESegment() [2/2]

```
ESegment::ESegment (
            ExtrapLine extrapLine )
```

This constructor is used if the segment contains no actual data, and the points must be created (such as in an extrapolation). The segment is created with reference to an entry in the segments extrapolation file.

Definition at line 52 of file ESegment.cpp.

7.45.3 Member Function Documentation

7.45.3.1 AddPoint()

```
void ESegment::AddPoint (
            EPoint point )
```

Adds a data point to the segment.

Definition at line 352 of file ESegment.cpp.

7.45.3.2 Fill()

```
int ESegment::Fill (
            CNuc * theCNuc,
            EData * theData,
            const Config & configure )
```

Fills the segment with points from the specified data file according to the maximum and minimum energy and angle ranges. The data is assumed to be entirely in the lab frame, and conversions are performed to the center of mass frame.

Definition at line 188 of file ESegment.cpp.

7.45.3.3 GetAStep()

`double ESegment::GetAStep ( ) const`

Returns the angle step to take when creating points in a segment. Only applies for extrapolation segments.

Definition at line 304 of file ESegment.cpp.

7.45.3.4 GetDataFile()

`std::string ESegment::GetDataFile ( ) const`

Returns the name of the data file from which to read.

Definition at line 344 of file ESegment.cpp.

7.45.3.5 GetEntranceKey()

`int ESegment::GetEntranceKey ( ) const`

Returns the entrance particle pair key of the segment.

Definition at line 170 of file ESegment.cpp.

7.45.3.6 GetEStep()

`double ESegment::GetEStep ( ) const`

Returns the energy step to take when creating points in a segment. Only applies for extrapolation segments.

Definition at line 296 of file ESegment.cpp.

7.45.3.7 GetExitKey()

`int ESegment::GetExitKey ( ) const`

Returns the exit particle pair key of the segment.

Definition at line 178 of file ESegment.cpp.

7.45.3.8 GetJ()

`double ESegment::GetJ ( ) const`

Returns the total spin of the segment. Only applies if the segment is phase shift.

Definition at line 312 of file ESegment.cpp.

7.45.3.9 GetL()

`int ESegment::GetL ( ) const`

Returns the orbital angular momentum value for the segment. Applies only if the segment is phase shift.

Definition at line 222 of file ESegment.cpp.

7.45.3.10 GetMaxAngDistOrder()

`int ESegment::GetMaxAngDistOrder ( ) const`

Returns the maximum polynomial order if segment is angular distribution.

Definition at line 248 of file ESegment.cpp.

7.45.3.11 GetMaxAngle()

`double ESegment::GetMaxAngle ( ) const`

Returns the maximum angle of the segment (lab frame).

Definition at line 280 of file ESegment.cpp.

7.45.3.12 GetMaxEnergy()

`double ESegment::GetMaxEnergy ( ) const`

Returns the maximum energy of the segment (lab frame).

Definition at line 264 of file ESegment.cpp.

7.45.3.13 GetMinAngle()

`double ESegment::GetMinAngle ( ) const`

Returns the minimum angle of the segment (lab frame).

Definition at line 272 of file ESegment.cpp.

7.45.3.14 GetMinEnergy()

`double ESegment::GetMinEnergy ( ) const`

Returns the minimum energy of the segment (lab frame).

Definition at line 256 of file ESegment.cpp.

7.45.3.15 GetNominalNorm()

```
double ESegment::GetNominalNorm ( ) const
```

Returns the nominal normalization parameter for the data segment.

Definition at line 328 of file ESegment.cpp.

7.45.3.16 GetNorm()

```
double ESegment::GetNorm ( ) const
```

Returns the normalization parameter for the data segment.

Definition at line 320 of file ESegment.cpp.

7.45.3.17 GetNormError()

```
double ESegment::GetNormError ( ) const
```

Returns the normalization error for the data segment.

Definition at line 336 of file ESegment.cpp.

7.45.3.18 GetPoint()

```
EPoint * ESegment::GetPoint (
            int pointNum )
```

Returns a pointer to the data point object specified by a position in the EPoint vector.

Definition at line 422 of file ESegment.cpp.

7.45.3.19 GetPoints()

```
std::vector< EPoint > & ESegment::GetPoints ( )
```

Returns a reference to the vector of EPoint objects.

Definition at line 431 of file ESegment.cpp.

7.45.3.20 GetSegmentChiSquared()

```
double ESegment::GetSegmentChiSquared ( ) const
```

Returns the chi-squared value of the segment after the fitting process.

Definition at line 288 of file ESegment.cpp.

7.45.3.21 GetSegmentKey()

```
int ESegment::GetSegmentKey ( ) const
```

Returns the segment key for the current ESegment object. The segment key is the order of the segment specified in the input file, INCLUDING non-active segments.

Definition at line 240 of file ESegment.cpp.

7.45.3.22 GetTargetEffectNum()

```
int ESegment::GetTargetEffectNum ( ) const
```

Returns the position of the corresponding TargetEffect object in the vector of the parent EData object.

Definition at line 231 of file ESegment.cpp.

7.45.3.23 IsAngularDist()

```
bool ESegment::IsAngularDist ( ) const
```

Returns true if the segment is angular distribution.

Definition at line 136 of file ESegment.cpp.

7.45.3.24 IsDifferential()

```
bool ESegment::IsDifferential ( ) const
```

Returns true if the segment is differential cross section, otherwise returns false.

Definition at line 110 of file ESegment.cpp.

7.45.3.25 IsInSegment()

```
bool ESegment::IsInSegment (
            EPoint point )
```

Returns true if a point is with the specified angle and energy ranges of a segment, otherwise returns false.

Definition at line 94 of file ESegment.cpp.

7.45.3.26 IsPhase()

```
bool ESegment::IsPhase ( ) const
```

Returns true if the segment is phase shift, otherwise returns false.

Definition at line 118 of file ESegment.cpp.

7.45.3.27 IsTargetEffect()

```
bool ESegment::IsTargetEffect ( ) const
```

Returns true if the segment has a corresponding TargetEffect object, otherwise returns false.

Definition at line 145 of file ESegment.cpp.

7.45.3.28 IsTotalCapture()

```
int ESegment::IsTotalCapture ( ) const
```

Returns the number of total capture segments to be summed. Should be zero if the segment is not total capture, otherwise the parameter should be the number of segments in the sum (inclusive of the current segment).

Definition at line 128 of file ESegment.cpp.

7.45.3.29 IsVaryNorm()

```
bool ESegment::IsVaryNorm ( ) const
```

Returns true if the normalization parameter for the segment is to be fit, otherwise returns false.

Definition at line 154 of file ESegment.cpp.

7.45.3.30 NumPoints()

```
int ESegment::NumPoints ( ) const
```

Returns the number of data point objects in the segment.

Definition at line 162 of file ESegment.cpp.

7.45.3.31 SetExitKey()

```
void ESegment::SetExitKey (
            int key )
```

Sets the exit pair key to the given value.

Definition at line 394 of file ESegment.cpp.

7.45.3.32 SetIsTotalCapture()

```
void ESegment::SetIsTotalCapture (
            int num )
```

Sets the number of total capture segments to be summed. Should be zero if the segment is not total capture, otherwise the parameter should be the number of segments in the sum (inclusive of the current segment).

Definition at line 406 of file ESegment.cpp.

7.45.3.33 SetNorm()

```
void ESegment::SetNorm (
            double norm )
```

Sets the normalization parameter for the segment.

Definition at line 386 of file ESegment.cpp.

7.45.3.34 SetSegmentChiSquared()

```
void ESegment::SetSegmentChiSquared (
            double chiSquared )
```

Sets the chi squared value for the segment during the fitting process.

Definition at line 360 of file ESegment.cpp.

7.45.3.35 SetSegmentKey()

```
void ESegment::SetSegmentKey (
            int segmentKey )
```

Sets the segment key for the current ESegment object.

Definition at line 378 of file ESegment.cpp.

7.45.3.36 SetTargetEffectNum()

```
void ESegment::SetTargetEffectNum (
            int targetEffectNum )
```

Sets the position of the corresponding TargetEffect object in the vector of the parent EData object.

Definition at line 369 of file ESegment.cpp.

7.45.3.37 SetVaryNorm()

```
void ESegment::SetVaryNorm (
            bool varyNorm )
```

Set the flag determining if the normalization is varied.

Definition at line 414 of file ESegment.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/ESegment.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/ESegment.cpp

7.46 ExtrapLine Class Reference

A class to read and store a line from the extrapolation input file.

```
#include <ExtrapLine.h>
```

Public Member Functions

- ExtrapLine (std::istream &stream)
- int isActive () const
- int entranceKey () const
- int exitKey () const
- double minE () const
- double maxE () const
- double minA () const
- double maxA () const
- double eStep () const
- double aStep () const
- int isDiff () const
- double phaseJ () const
- int phaseL () const
- int maxAngDistOrder () const

7.46.1 Detailed Description

A class to read and store a line from the extrapolation input file.

The ExtrapLine class reads and stores a line from the extrapolation input file.

Definition at line 13 of file ExtrapLine.h.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 ExtrapLine()

```
ExtrapLine::ExtrapLine (
            std::istream & stream )  [inline]
```

Constructor fill the ExtrapLine object from an input stream.

Definition at line 18 of file ExtrapLine.h.

7.46.3 Member Function Documentation

7.46.3.1 aStep()

```
double ExtrapLine::aStep ( ) const  [inline]
```

Returns the size angle step between generated points.

Definition at line 62 of file ExtrapLine.h.

7.46.3.2 entranceKey()

```
int ExtrapLine::entranceKey ( ) const  [inline]
```

Returns the particle pair key corresponding to the entrance channel for the data segment.

Definition at line 33 of file ExtrapLine.h.

7.46.3.3 eStep()

```
double ExtrapLine::eStep ( ) const  [inline]
```

Returns the size energy step between generated points.

Definition at line 58 of file ExtrapLine.h.

7.46.3.4 exitKey()

```
int ExtrapLine::exitKey ( ) const  [inline]
```

Returns the particle pair key corresponding to the exit channel for the data segment.

Definition at line 38 of file ExtrapLine.h.

7.46.3.5 isActive()

```
int ExtrapLine::isActive ( ) const  [inline]
```

Returns non-zero if the line is to be included in the calculation.

Definition at line 28 of file ExtrapLine.h.

7.46.3.6 isDiff()

```
int ExtrapLine::isDiff ( ) const  [inline]
```

Return 0 if the segment is angle-integrated cross section, 1 for differential cross section, and 2 for phase shift.

Definition at line 67 of file ExtrapLine.h.

7.46.3.7 maxA()

```
double ExtrapLine::maxA ( ) const  [inline]
```

Returns the maximum angle to be generated.

Definition at line 54 of file ExtrapLine.h.

7.46.3.8 maxAngDistOrder()

```
int ExtrapLine::maxAngDistOrder ( ) const  [inline]
```

Returns the maximum polynomial order if segment is angular distribution.

Definition at line 82 of file ExtrapLine.h.

7.46.3.9 maxE()

```
double ExtrapLine::maxE ( ) const  [inline]
```

Returns the maximum energy to be generated.

Definition at line 46 of file ExtrapLine.h.

7.46.3.10 minA()

```
double ExtrapLine::minA ( ) const  [inline]
```

Returns the minimum angle to be generated.

Definition at line 50 of file ExtrapLine.h.

7.46.3.11 minE()

```
double ExtrapLine::minE ( ) const  [inline]
```

Returns the minimum energy to be generated.

Definition at line 42 of file ExtrapLine.h.

7.46.3.12 phaseJ()

```
double ExtrapLine::phaseJ ( ) const  [inline]
```

Returns the spin value for the segment if the segment is to contain phase shift.

Definition at line 72 of file ExtrapLine.h.

7.46.3.13 phaseL()

```
int ExtrapLine::phaseL ( ) const  [inline]
```

Returns the orbital angular momentum value for the segment if the segment is to contain phase shift.

Definition at line 77 of file ExtrapLine.h.

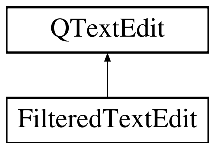
The documentation for this class was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/ExtrapLine.h

7.47 FilteredTextEdit Class Reference

```
#include <FilteredTextEdit.h>
```

Inheritance diagram for FilteredTextEdit:



Public Slots

- void write (QString string)

Public Member Functions

- FilteredTextEdit (QWidget ∗parent=0)
- void SetMouseFiltered (bool filtered)
- bool IsMouseFiltered () const
- void mousePressEvent (QMouseEvent ∗event)
- void mouseDoubleClickEvent (QMouseEvent ∗event)

7.47.1 Detailed Description

Definition at line 8 of file FilteredTextEdit.h.

7.47.2 Constructor & Destructor Documentation

7.47.2.1 FilteredTextEdit()

```
FilteredTextEdit::FilteredTextEdit (
            QWidget * parent = 0 )  [inline]
```

Definition at line 12 of file FilteredTextEdit.h.

7.47.3 Member Function Documentation

7.47.3.1 IsMouseFiltered()

```
bool FilteredTextEdit::IsMouseFiltered ( ) const  [inline]
```

Definition at line 19 of file FilteredTextEdit.h.

7.47.3.2 mouseDoubleClickEvent()

```
void FilteredTextEdit::mouseDoubleClickEvent (
            QMouseEvent * event ) [inline]
```

Definition at line 33 of file FilteredTextEdit.h.

7.47.3.3 mousePressEvent()

```
void FilteredTextEdit::mousePressEvent (
            QMouseEvent * event ) [inline]
```

Definition at line 30 of file FilteredTextEdit.h.

7.47.3.4 SetMouseFiltered()

```
void FilteredTextEdit::SetMouseFiltered (
            bool filtered ) [inline]
```

Definition at line 18 of file FilteredTextEdit.h.

7.47.3.5 write

```
void FilteredTextEdit::write (
            QString string ) [inline], [slot]
```

Definition at line 21 of file FilteredTextEdit.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/FilteredTextEdit.h

7.48 GenericFunction Class Reference

A wrapper class for function pointers used by Equation class.

```
#include <Equation.h>
```

Public Member Functions

- GenericFunction ()
- GenericFunction (double(∗function)(double))
- double Evaluate (double value) const

7.48.1 Detailed Description

A wrapper class for function pointers used by Equation class.

The GenericFunction class is just a wrapper for a function pointer to a function of the form double function(double).

Definition at line 19 of file Equation.h.

7.48.2 Constructor & Destructor Documentation

7.48.2.1 GenericFunction() [1/2]

```
GenericFunction::GenericFunction ( ) [inline]
```

The default constructor for the class.

Definition at line 24 of file Equation.h.

7.48.2.2 GenericFunction() [2/2]

```
GenericFunction::GenericFunction (
            double(*)(double) function ) [inline]
```

Constructs the object from a function pointer.

Definition at line 28 of file Equation.h.

7.48.3 Member Function Documentation

7.48.3.1 Evaluate()

```
double GenericFunction::Evaluate (
            double value ) const [inline]
```

Evaluates the function object with a given double.

Definition at line 33 of file Equation.h.

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

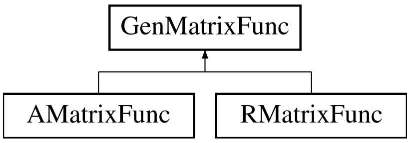
- /Users/kuba/Desktop/R-Matrix/AZURE2/include/Equation.h

7.49 GenMatrixFunc Class Reference

A generalized function class to calculate cross sections.

```
#include <GenMatrixFunc.h>
```

Inheritance diagram for GenMatrixFunc:



Public Member Functions

- GenMatrixFunc ()
- virtual ∼GenMatrixFunc ()
- virtual void ClearMatrices ()=0
- virtual void FillMatrices (EPoint ∗)=0
- virtual void InvertMatrices ()=0
- virtual void CalculateTMatrix (EPoint ∗)=0
- void CalculateCrossSection (EPoint ∗)
- void NewTempTMatrix (TempTMatrix)
- void AddToTempTMatrix (int, complex)
- void ClearTempTMatrices ()
- void AddTMatrixElement (int, int, complex, int decayNum=1)
- void AddECTMatrixElement (int, int, complex)
- int IsTempTMatrix (double, int, int)
- int NumTempTMatrices () const
- TempTMatrix ∗ GetTempTMatrix (int)
- complex GetTMatrixElement (int, int, int decayNum=1) const
- complex GetECTMatrixElement (int, int) const
- virtual CNuc ∗ compound () const =0
- virtual const Config & configure () const =0

Protected Attributes

- std::vector< matrix_c > tmatrix_

    Vector of internal T-matrix elements accessable to child class.
- matrix_c ec_tmatrix_

    Vector of external T-matrix elements accessable to child class.

7.49.1 Detailed Description

A generalized function class to calculate cross sections.

The GenMatrixFunc function class is the general form of the function used to calculate cross section from R-Matrix parameters. It is the parent class of AMatrixFunc and RMatrixFunc.

Definition at line 38 of file GenMatrixFunc.h.

7.49.2 Constructor & Destructor Documentation

7.49.2.1 GenMatrixFunc()

```
GenMatrixFunc::GenMatrixFunc ( )  [inline]
```

Definition at line 40 of file GenMatrixFunc.h.

7.49.2.2 ∼GenMatrixFunc()

```
virtual GenMatrixFunc::∼GenMatrixFunc ( )  [inline], [virtual]
```

Definition at line 41 of file GenMatrixFunc.h.

7.49.3 Member Function Documentation

7.49.3.1 AddECTMatrixElement()

```
void GenMatrixFunc::AddECTMatrixElement (
            int kGroupNum,
            int mGroupNum,
            complex tMatrixElement )
```

Adds an external T-Matrix element to the vector of external T-matrix elements corresponding to a specified external reaction pathway.

Definition at line 249 of file GenMatrixFunc.cpp.

7.49.3.2 AddTMatrixElement()

```
void GenMatrixFunc::AddTMatrixElement (
            int kGroupNum,
            int mGroupNum,
            complex tMatrixElement,
            int decayNum = 1 )
```

Adds an internal T-Matrix element to the vector of internal T-matrix elements corresponding to a specified internal reaction pathway.

Definition at line 234 of file GenMatrixFunc.cpp.

7.49.3.3 AddToTempTMatrix()

```
void GenMatrixFunc::AddToTempTMatrix (
            int tempTMatrixNum,
            complex tempValue )
```

Adds a value to the temporary T-Matrix element specified by its position in the TempTMatrix vector.

Definition at line 216 of file GenMatrixFunc.cpp.

7.49.3.4 CalculateCrossSection()

```
void GenMatrixFunc::CalculateCrossSection (
            EPoint * point )
```

The child classes AMatrixFunc or RMatrixFunc contain functions to calculate the T-Matrix from the fitted R-Matrix parameters. This function then calculates the cross section from the T-Matrix elements.

Definition at line 12 of file GenMatrixFunc.cpp.

7.49.3.5 CalculateTMatrix()

```
virtual void GenMatrixFunc::CalculateTMatrix (
            EPoint * ) [pure virtual]
```

This virtual function in implemented in the child class.

Implemented in AMatrixFunc, and RMatrixFunc.

7.49.3.6 ClearMatrices()

```
virtual void GenMatrixFunc::ClearMatrices ( ) [pure virtual]
```

This virtual function in implemented in the child class.

Implemented in AMatrixFunc, and RMatrixFunc.

7.49.3.7 ClearTempTMatrices()

```
void GenMatrixFunc::ClearTempTMatrices ( )
```

Clears the temporary T-Matrices.

Definition at line 225 of file GenMatrixFunc.cpp.

7.49.3.8 compound()

```
virtual CNuc * GenMatrixFunc::compound ( ) const [pure virtual]
```

This virtual function in implemented in the child class.

Implemented in AMatrixFunc, and RMatrixFunc.

7.49.3.9 configure()

```
virtual const Config & GenMatrixFunc::configure ( ) const [pure virtual]
```

This virtual function in implemented in the child class.

Implemented in AMatrixFunc, and RMatrixFunc.

7.49.3.10 FillMatrices()

```
virtual void GenMatrixFunc::FillMatrices (
             EPoint * ) [pure virtual]
```

This virtual function in implemented in the child class.

Implemented in AMatrixFunc, and RMatrixFunc.

7.49.3.11 GetECTMatrixElement()

```
complex GenMatrixFunc::GetECTMatrixElement (
             int kGroupNum,
             int ecMGroupNum ) const
```

Returns the value of the external T-Matrix element specified by an external reaction pathway.

Definition at line 303 of file GenMatrixFunc.cpp.

7.49.3.12 GetTempTMatrix()

```
TempTMatrix * GenMatrixFunc::GetTempTMatrix (
             int tempTMatrixNum )
```

Returns a pointer to the temporary T-Matrix element specified by a position in the TempTMatrix vector.

Definition at line 286 of file GenMatrixFunc.cpp.

7.49.3.13 GetTMatrixElement()

```
complex GenMatrixFunc::GetTMatrixElement (
             int kGroupNum,
             int mGroupNum,
             int decayNum = 1 ) const
```

Returns the value of the internal T-Matrix element specified by an internal reaction pathway.

Definition at line 295 of file GenMatrixFunc.cpp.

7.49.3.14 InvertMatrices()

```
virtual void GenMatrixFunc::InvertMatrices ( ) [pure virtual]
```

This virtual function in implemented in the child class.

Implemented in AMatrixFunc, and RMatrixFunc.

7.49.3.15 IsTempTMatrix()

```
int GenMatrixFunc::IsTempTMatrix (
            double jValue,
            int lValue,
            int lPrimeValue )
```

Tests if a temporary T-Matrix element already exists for a given J, l, l' combination. If the element exists, returns the position in the TempTMatrix vector, otherwise returns 0.

Definition at line 261 of file GenMatrixFunc.cpp.

7.49.3.16 NewTempTMatrix()

```
void GenMatrixFunc::NewTempTMatrix (
            TempTMatrix tempTMatrix )
```

Creates a new temporary T-Matrix element.

Definition at line 208 of file GenMatrixFunc.cpp.

7.49.3.17 NumTempTMatrices()

```
int GenMatrixFunc::NumTempTMatrices ( ) const
```

Returns the number of temporary T-Matrix elements in the TempTMatrix vector.

Definition at line 278 of file GenMatrixFunc.cpp.

7.49.4 Member Data Documentation

7.49.4.1 ec_tmatrix_

```
matrix_c GenMatrixFunc::ec_tmatrix_ [protected]
```

Vector of external T-matrix elements accessable to child class.

Definition at line 82 of file GenMatrixFunc.h.

7.49.4.2 tmatrix_

```
std::vector<matrix_c> GenMatrixFunc::tmatrix_ [protected]
```

Vector of internal T-matrix elements accessable to child class.

Definition at line 80 of file GenMatrixFunc.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/GenMatrixFunc.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/GenMatrixFunc.cpp

7.50 gsl_reactionrate_params Struct Reference

Public Member Functions

- gsl_reactionrate_params (const Config &config)

Public Attributes

- const Config & configure
- double temperature
- CNuc * compound
- int entranceKey
- int exitKey

7.50.1 Detailed Description

Definition at line 13 of file ReactionRate.cpp.

7.50.2 Constructor & Destructor Documentation

7.50.2.1 gsl_reactionrate_params()

```
gsl_reactionrate_params::gsl_reactionrate_params (
             const Config & config )  [inline]
```

Definition at line 14 of file ReactionRate.cpp.

7.50.3 Member Data Documentation

7.50.3.1 compound

```
CNuc* gsl_reactionrate_params::compound
```

Definition at line 17 of file ReactionRate.cpp.

7.50.3.2 configure

```
const Config& gsl_reactionrate_params::configure
```

Definition at line 15 of file ReactionRate.cpp.

7.50.3.3 entranceKey

```
int gsl_reactionrate_params::entranceKey
```

Definition at line 18 of file ReactionRate.cpp.

7.50.3.4 exitKey

```
int gsl_reactionrate_params::exitKey
```

Definition at line 19 of file ReactionRate.cpp.

7.50.3.5 temperature

```
double gsl_reactionrate_params::temperature
```

Definition at line 16 of file ReactionRate.cpp.

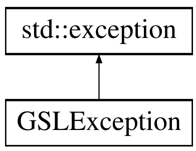
The documentation for this struct was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/src/ReactionRate.cpp

7.51 GSLException Class Reference

```
#include <GSLException.h>
```

Inheritance diagram for GSLException:



Public Member Functions

- GSLException (std::string message, std::string line="", std::string file="")
- ∼GSLException () throw ()
- virtual const char ∗ what () const throw ()

Static Public Member Functions

- static void GSLErrorHandler (const char ∗, const char ∗, int, int)

7.51.1 Detailed Description

The GSLException class is an exception class thrown by the CoulFunc class. It should not be used directly.

Definition at line 13 of file GSLException.h.

7.51.2 Constructor & Destructor Documentation

7.51.2.1 GSLException()

```
GSLException::GSLException (
            std::string message,
            std::string line = "",
            std::string file = "" )  [inline]
```

Definition at line 15 of file GSLException.h.

7.51.2.2 ~GSLException()

```
GSLException::~GSLException ( ) throw ( )  [inline]
```

Definition at line 27 of file GSLException.h.

7.51.3 Member Function Documentation

7.51.3.1 GSLErrorHandler()

```
void GSLException::GSLErrorHandler (
            const char * reason,
            const char * file,
            int line,
            int errorCode )  [static]
```

Definition at line 4 of file GSLException.cpp.

7.51.3.2 what()

```
virtual const char * GSLException::what ( ) const throw ( )  [inline], [virtual]
```

Definition at line 29 of file GSLException.h.

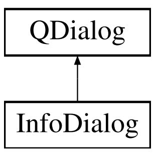
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/GSLException.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/GSLException.cpp

7.52 InfoDialog Class Reference

```
#include <InfoDialog.h>
```

Inheritance diagram for InfoDialog:



Public Member Functions

- InfoDialog (const QString &, QWidget ∗parent=0, QString title="")

7.52.1 Detailed Description

Definition at line 6 of file InfoDialog.h.

7.52.2 Constructor & Destructor Documentation

7.52.2.1 InfoDialog()

```
InfoDialog::InfoDialog (
            const QString & string,
            QWidget * parent = 0,
            QString title = "" )
```

Definition at line 7 of file InfoDialog.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/InfoDialog.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InfoDialog.cpp

7.53 IntegratedFermiFunc Class Reference

A function class to calculate the integrated Fermi function for beta decay.

```
#include <IntegratedFermiFunc.h>
```

Public Member Functions

- IntegratedFermiFunc (int, double V0=0.)
- double operator() (double, double, double)

7.53.1 Detailed Description

A function class to calculate the integrated Fermi function for beta decay.

This function class calculates the integrated Fermi function for beta decay channels. The integrand is taken directly from Konopinski and Rose. While screening potentials can be added, by default the screening potential is set to zero. This function class should be valid for either electron or positron emission.

Definition at line 13 of file IntegratedFermiFunc.h.

7.53.2 Constructor & Destructor Documentation

7.53.2.1 IntegratedFermiFunc()

```
IntegratedFermiFunc::IntegratedFermiFunc (
            int charge,
            double V0 = 0. )
```

The constructor for the function takes the charge of the Fermion as the first argument. This is expected to be either -1 or 1 for electrons and positrons, respectively. The second optional argument is the screening potential V_0, where the actual screening potential, V, will be calculated as $V = V_0\alpha^2 Z^{4/3}$.

Definition at line 19 of file IntegratedFermiFunc.cpp.

7.53.3 Member Function Documentation

7.53.3.1 operator()()

```
double IntegratedFermiFunc::operator() (
            double W0,
            double Z,
            double radius )
```

The calling operator for the function class takes the end point energy (in units of m_0c^2), the charge of the daughter nucleus, and the nuclear radius parameter as arguments. The return value is the integrated Fermi function.

Definition at line 29 of file IntegratedFermiFunc.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/IntegratedFermiFunc.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/IntegratedFermiFunc.cpp

7.54 Interference Class Reference

An AZURE $l_1, l_2, l_1', l_2', J_1, J_2$ combination.

```
#include <Interference.h>
```

Public Member Functions

- Interference (int, int, double, std::string)
- std::string GetInterferenceType () const
- int GetM1 () const
- int GetM2 () const
- double GetZ1Z2 () const

7.54.1 Detailed Description

An AZURE $l_1, l_2, l_1', l_2', J_1, J_2$ combination.

In the differential cross section formula of R-Matrix, nested inside the s, s', L sum is a sum over $l_1, l_2, l_1', l_2', J_1, J_2$. In the language of AZURE, these are equivalent to combinations of two reaction pathways. If the pathways are the same, the term represents the actual contribution from the pathway to the cross section. If they are different, the term represents the interference between the two.

Definition at line 16 of file Interference.h.

7.54.2 Constructor & Destructor Documentation

7.54.2.1 Interference()

```
Interference::Interference (
            int mGroupNum1,
            int mGroupNum2,
            double z1z2Coeff,
            std::string interferenceType )
```

The pathways combination is created specifically using references to two positions in the MGroup and ECMGroup vectors under the corresponding KGroup object. Additionally, the $Z_1 Z_2$ coefficients are passed along with the interference type. The interference type is either RR, ER, RE,or EE, indicating which vector, the MGroup or ECMGroup, the stored indices refer to.

Definition at line 11 of file Interference.cpp.

7.54.3 Member Function Documentation

7.54.3.1 GetInterferenceType()

```
std::string Interference::GetInterferenceType ( ) const
```

Returns the interference type.

Definition at line 18 of file Interference.cpp.

7.54.3.2 GetM1()

```
int Interference::GetM1 ( ) const
```

Returns the position in the MGroup or ECMGroup vector of the first pathway.

Definition at line 26 of file Interference.cpp.

7.54.3.3 GetM2()

```
int Interference::GetM2 ( ) const
```

Returns the position in the MGroup or ECMGroup vector of the second pathway.

Definition at line 34 of file Interference.cpp.

7.54.3.4 GetZ1Z2()

```
double Interference::GetZ1Z2 ( ) const
```

Returns the corresponding $Z_1 Z_2$ coefficient.

Definition at line 42 of file Interference.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/Interference.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/Interference.cpp

7.55 JGroup Class Reference

An AZURE J^{π} group.

```
#include <JGroup.h>
```

Public Member Functions

- JGroup (NucLine)
- JGroup (double, int)
- bool IsInRMatrix () const
- int IsLevel (ALevel)
- int GetPi () const
- int NumLevels () const
- int NumChannels ()
- int IsChannel (AChannel)
- double GetJ () const
- void AddLevel (ALevel)
- void AddChannel (AChannel)
- AChannel $*$ GetChannel (int)
- ALevel $*$ GetLevel (int)

7.55.1 Detailed Description

An AZURE J^{π} group.

In R-Matrix theory, levels are grouped according to their J^{π} values. There is one R-/A-Matrix, and thus one T-Matrix, for each J^{π} group. A JGroup object holds vectors of ALevel and AChannel objects.

Definition at line 17 of file JGroup.h.

7.55.2 Constructor & Destructor Documentation

7.55.2.1 JGroup() [1/2]

```
JGroup::JGroup (
            NucLine nucLine )
```

This constructor is used when a J^{π} group is created from an entry in the nuclear input file.

Definition at line 8 of file JGroup.cpp.

7.55.2.2 JGroup() [2/2]

```
JGroup::JGroup (
            double j,
            int pi )
```

This constructor is used when a J^{π} group is created from specified values of spin and parity.

Definition at line 15 of file JGroup.cpp.

7.55.3 Member Function Documentation

7.55.3.1 AddChannel()

```
void JGroup::AddChannel (
            AChannel channel )
```

Adds a new channel to the vector of AChannel objects.

Definition at line 108 of file JGroup.cpp.

7.55.3.2 AddLevel()

```
void JGroup::AddLevel (
            ALevel level )
```

Adds a new level to the vector of ALevel objects.

Definition at line 100 of file JGroup.cpp.

7.55.3.3 GetChannel()

```
AChannel * JGroup::GetChannel (
            int channelNum )
```

Returns a pointer to a specified channel in the AChannel vector.

Definition at line 117 of file JGroup.cpp.

7.55.3.4 GetJ()

```
double JGroup::GetJ ( ) const
```

Returns the spin value of the J^{π} group.

Definition at line 92 of file JGroup.cpp.

7.55.3.5 GetLevel()

```
ALevel * JGroup::GetLevel (
            int levelNum )
```

Returns a pointer to a specified level in the ALevel vector.

Definition at line 126 of file JGroup.cpp.

7.55.3.6 GetPi()

```
int JGroup::GetPi ( ) const
```

Returns the parity of the the J^{π} group as ± 1.

Definition at line 50 of file JGroup.cpp.

7.55.3.7 IsChannel()

```
int JGroup::IsChannel (
            AChannel channel )
```

This function tests if a given channel already exists in the vector of AChannel objects. If the channel exists the position of the channel in the vector is returned, otherwise the function returns 0.

Definition at line 75 of file JGroup.cpp.

7.55.3.8 IsInRMatrix()

```
bool JGroup::IsInRMatrix ( ) const
```

Returns true if the J^{π} group is to be included in the A-/R-Matrix calculation, otherwise returns false. A J^{π} group may specify only a bound state for external capture, but may not correspond to an R-Matrix state (i.e. subthreshold state).

Definition at line 23 of file JGroup.cpp.

7.55.3.9 IsLevel()

```
int JGroup::IsLevel (
            ALevel level )
```

This function tests if a given level already exists in the vector of ALevel objects. If the level exists the position of the level in the vector is returned, otherwise the function returns 0.

Definition at line 32 of file JGroup.cpp.

7.55.3.10 NumChannels()

```
int JGroup::NumChannels ( )
```

Returns the number of channels in the AChannel vector.

Definition at line 66 of file JGroup.cpp.

7.55.3.11 NumLevels()

```
int JGroup::NumLevels ( ) const
```

Returns the number of levels in the ALevel vector.

Definition at line 58 of file JGroup.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/JGroup.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/JGroup.cpp

7.56 KGroup Class Reference

An AZURE s, s' group.

```
#include <KGroup.h>
```

Public Member Functions

- KGroup (double, double)
- int NumMGroups () const
- int NumECMGroups () const
- int IsMGroup (MGroup)
- double GetS () const
- double GetSp () const
- void AddMGroup (MGroup)
- void AddECMGroup (ECMGroup)
- MGroup * GetMGroup (int)
- ECMGroup * GetECMGroup (int)

7.56.1 Detailed Description

An AZURE s, s' group.

In R-Matrix formalism, the equations required to calculate the cross section usually nested inside sums over entrance and exit channel spins. For this reason AZURE groups reaction pathways according to their entrance and exit channel spins. Each KGroup object is a container for vectors of MGroup and ECMGroup objects.

Definition at line 16 of file KGroup.h.

7.56.2 Constructor & Destructor Documentation

7.56.2.1 KGroup()

```
KGroup::KGroup (
            double s,
            double sPrime )
```

The KGroup is created from a specific combination of entrance and exit channel spin values.

Definition at line 7 of file KGroup.cpp.

7.56.3 Member Function Documentation

7.56.3.1 AddECMGroup()

```
void KGroup::AddECMGroup (
            ECMGroup ecMGroup )
```

Adds a new external reaction pathway to the ECMGroup vector.

Definition at line 74 of file KGroup.cpp.

7.56.3.2 AddMGroup()

```
void KGroup::AddMGroup (
            MGroup mGroup )
```

Adds a new internal reaction pathway to the MGroup vector.

Definition at line 66 of file KGroup.cpp.

7.56.3.3 GetECMGroup()

```
ECMGroup * KGroup::GetECMGroup (
            int ecMGroupNum )
```

Returns a pointer to the external reaction pathway specified by a position in the ECMGroup vector.

Definition at line 90 of file KGroup.cpp.

7.56.3.4 GetMGroup()

```
MGroup * KGroup::GetMGroup (
            int mGroupNum )
```

Returns a pointer to the internal reaction pathway specified by a position in the MGroup vector.

Definition at line 82 of file KGroup.cpp.

7.56.3.5 GetS()

```
double KGroup::GetS ( ) const
```

Returns the value of the entrance channel spin.

Definition at line 50 of file KGroup.cpp.

7.56.3.6 GetSp()

```
double KGroup::GetSp ( ) const
```

Returns the value of the exit channel spin.

Definition at line 58 of file KGroup.cpp.

7.56.3.7 IsMGroup()

```
int KGroup::IsMGroup (
            MGroup mGroup )
```

Tests a specific internal reaction pathway to see if it already exists in the MGroup vector. If the pathway exists, its position in the vector is returned. Otherwise, the function returns 0.

Definition at line 32 of file KGroup.cpp.

7.56.3.8 NumECMGroups()

```
int KGroup::NumECMGroups ( ) const
```

Returns the number of external reaction pathways in the ECMGroup vector;

Definition at line 22 of file KGroup.cpp.

7.56.3.9 NumMGroups()

```
int KGroup::NumMGroups ( ) const
```

Returns the number of internal reaction pathways in the MGroup vector.

Definition at line 14 of file KGroup.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/KGroup.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/KGroup.cpp

7.57 KLGroup Class Reference

An AZURE s, s', L group.

```
#include <KLGroup.h>
```

Public Member Functions

- KLGroup (int, int)
- int GetK () const
- int GetLOrder () const
- int NumInterferences () const
- int IsInterference (Interference)
- void AddInterference (Interference)
- Interference $*$ GetInterference (int)

7.57.1 Detailed Description

An AZURE s, s', L group.

Differential cross sections in R-Matrix theory contains terms nested inside a sum over entrance and exit spins as well as Legendre polynomial orders, L. In AZURE, an s, s' combination is given by a KGroup object. It is therefore convenient to group KGroup objects with a specified polynomial orders for the calculation of differenial cross sections. The KLGroup object serves as a container class for a vector of Interference objects.

Definition at line 16 of file KLGroup.h.

7.57.2 Constructor & Destructor Documentation

7.57.2.1 KLGroup()

```
KLGroup::KLGroup (
            int kGroupNum,
            int lOrder )
```

The object is created with reference to a specfic KGroup number as well as Legendre polynomial order.

Definition at line 7 of file KLGroup.cpp.

7.57.3 Member Function Documentation

7.57.3.1 AddInterference()

```
void KLGroup::AddInterference (
            Interference interference )
```

Adds an interference combination to the Interference vector.

Definition at line 57 of file KLGroup.cpp.

7.57.3.2 GetInterference()

```
Interference * KLGroup::GetInterference (
            int interferenceNum )
```

Returns a pointer to an interference combination specified by a position in the Interference vector.

Definition at line 65 of file KLGroup.cpp.

7.57.3.3 GetK()

```
int KLGroup::GetK ( ) const
```

Returns the position of the s, s' combination in the KGroup vector.

Definition at line 14 of file KLGroup.cpp.

7.57.3.4 GetLOrder()

```
int KLGroup::GetLOrder ( ) const
```

Returns the Legendre polynomial order.

Definition at line 22 of file KLGroup.cpp.

7.57.3.5 IsInterference()

```
int KLGroup::IsInterference (
            Interference interference )
```

Tests an interference combination to determine if it exists in the Interference vector. If the combination exists, its position in the vector is returned. Otherwise, the function returns 0.

Definition at line 39 of file KLGroup.cpp.

7.57.3.6 NumInterferences()

```
int KLGroup::NumInterferences ( ) const
```

Returns the number of interference combinations in the Interference vector.

Definition at line 30 of file KLGroup.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/KLGroup.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/KLGroup.cpp

7.58 LevelsData Struct Reference

```
#include <LevelsModel.h>
```

Public Attributes

- int isActive
- int isFixed
- double jValue
- int piValue
- double energy

Static Public Attributes

- static const int SIZE = 5

7.58.1 Detailed Description

Definition at line 7 of file LevelsModel.h.

7.58.2 Member Data Documentation

7.58.2.1 energy

```
double LevelsData::energy
```

Definition at line 13 of file LevelsModel.h.

7.58.2.2 isActive

```
int LevelsData::isActive
```

Definition at line 9 of file LevelsModel.h.

7.58.2.3 isFixed

```
int LevelsData::isFixed
```

Definition at line 10 of file LevelsModel.h.

7.58.2.4 jValue

```
double LevelsData::jValue
```

Definition at line 11 of file LevelsModel.h.

7.58.2.5 piValue

```
int LevelsData::piValue
```

Definition at line 12 of file LevelsModel.h.

7.58.2.6 SIZE

```
const int LevelsData::SIZE = 5  [static]
```

Definition at line 8 of file LevelsModel.h.

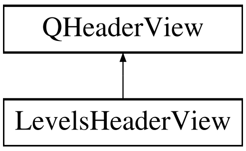
The documentation for this struct was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsModel.h

7.59 LevelsHeaderView Class Reference

```
#include <LevelsHeaderView.h>
```

Inheritance diagram for LevelsHeaderView:



Public Member Functions

- LevelsHeaderView (Qt::Orientation orientation, QWidget *parent)

Protected Member Functions

- virtual void mouseMoveEvent (QMouseEvent *e)
- virtual void mousePressEvent (QMouseEvent *e)
- virtual void mouseReleaseEvent (QMouseEvent *e)

7.59.1 Detailed Description

Definition at line 7 of file LevelsHeaderView.h.

7.59.2 Constructor & Destructor Documentation

7.59.2.1 LevelsHeaderView()

```
LevelsHeaderView::LevelsHeaderView (
            Qt::Orientation orientation,
            QWidget * parent )  [inline]
```

Definition at line 9 of file LevelsHeaderView.h.

7.59.3 Member Function Documentation

7.59.3.1 mouseMoveEvent()

```
virtual void LevelsHeaderView::mouseMoveEvent (
            QMouseEvent * e )  [inline], [protected], [virtual]
```

Definition at line 14 of file LevelsHeaderView.h.

7.59.3.2 mousePressEvent()

```
virtual void LevelsHeaderView::mousePressEvent (
            QMouseEvent * e )  [inline], [protected], [virtual]
```

Definition at line 19 of file LevelsHeaderView.h.

7.59.3.3 mouseReleaseEvent()

```
virtual void LevelsHeaderView::mouseReleaseEvent (
            QMouseEvent * e )  [inline], [protected], [virtual]
```

Definition at line 24 of file LevelsHeaderView.h.

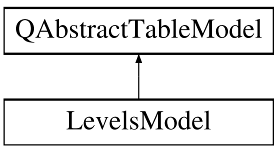
The documentation for this class was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsHeaderView.h

7.60 LevelsModel Class Reference

```
#include <LevelsModel.h>
```

Inheritance diagram for LevelsModel:



Public Member Functions

- LevelsModel (QObject *parent=0)
- int rowCount (const QModelIndex &parent) const
- int columnCount (const QModelIndex &parent) const
- QVariant data (const QModelIndex &index, int role) const
- QVariant headerData (int section, Qt::Orientation orientation, int role) const
- bool setData (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- bool insertRows (int position, int rows, const QModelIndex &index=QModelIndex())
- bool removeRows (int position, int rows, const QModelIndex &index=QModelIndex())
- Qt::ItemFlags flags (const QModelIndex &index) const
- int isLevel (const LevelsData &level) const
- QList< LevelsData > getLevels () const
- QString getSpinLabel (const LevelsData &level) const

7.60.1 Detailed Description

Definition at line 16 of file LevelsModel.h.

7.60.2 Constructor & Destructor Documentation

7.60.2.1 LevelsModel()

```
LevelsModel::LevelsModel (
            QObject * parent = 0 )
```

Definition at line 3 of file LevelsModel.cpp.

7.60.3 Member Function Documentation

7.60.3.1 columnCount()

```
int LevelsModel::columnCount (
            const QModelIndex & parent ) const
```

Definition at line 11 of file LevelsModel.cpp.

7.60.3.2 data()

```
QVariant LevelsModel::data (
            const QModelIndex & index,
            int role ) const
```

Definition at line 16 of file LevelsModel.cpp.

7.60.3.3 flags()

```
Qt::ItemFlags LevelsModel::flags (
             const QModelIndex & index ) const
```

Definition at line 127 of file LevelsModel.cpp.

7.60.3.4 getLevels()

```
QList< LevelsData > LevelsModel::getLevels ( ) const  [inline]
```

Definition at line 31 of file LevelsModel.h.

7.60.3.5 getSpinLabel()

```
QString LevelsModel::getSpinLabel (
             const LevelsData & level ) const
```

Definition at line 147 of file LevelsModel.cpp.

7.60.3.6 headerData()

```
QVariant LevelsModel::headerData (
             int section,
             Qt::Orientation orientation,
             int role ) const
```

Definition at line 45 of file LevelsModel.cpp.

7.60.3.7 insertRows()

```
bool LevelsModel::insertRows (
             int position,
             int rows,
             const QModelIndex & index = QModelIndex() )
```

Definition at line 102 of file LevelsModel.cpp.

7.60.3.8 isLevel()

```
int LevelsModel::isLevel (
             const LevelsData & level ) const
```

Definition at line 133 of file LevelsModel.cpp.

7.60.3.9 removeRows()

```
bool LevelsModel::removeRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 115 of file LevelsModel.cpp.

7.60.3.10 rowCount()

```
int LevelsModel::rowCount (
            const QModelIndex & parent ) const
```

Definition at line 6 of file LevelsModel.cpp.

7.60.3.11 setData()

```
bool LevelsModel::setData (
            const QModelIndex & index,
            const QVariant & value,
            int role = Qt::EditRole )
```

Definition at line 68 of file LevelsModel.cpp.

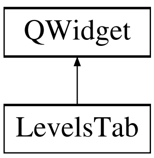
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsModel.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/LevelsModel.cpp

7.61 LevelsTab Class Reference

```
#include <LevelsTab.h>
```

Inheritance diagram for LevelsTab:



Public Slots

- void addLevel ()
- void addLevel (LevelsData level, bool fromFile)
- void removeLevel ()
- void editLevel ()
- void updateButtons (const QItemSelection &selection)
- void updateFilter (const QItemSelection &selection)
- void updateChannelsPairAddedEdited ()
- void updateChannelsPairRemoved (int pairIndex)
- void updateDetails (const QItemSelection &selection)
- void updateReducedWidth (const QString &string)
- void showInfo (int which=0, QString title="")

Signals

- void readNewPair (PairsData, int, bool)
- void readExistingPair (PairsData, int, bool)

Public Member Functions

- LevelsTab (QWidget ∗parent=0)
- void setPairsModel (PairsModel ∗)
- void updateChannelsLevelAdded (int levelIndex)
- void updateChannelsLevelDeleted (int levelIndex)
- void updateChannelsLevelEdited (int levelIndex)
- QList< ChannelsData > calculateChannels (int levelIndex)
- bool writeNuclearFile (QTextStream &outStream)
- bool readNuclearFile (QTextStream &inStream)
- void reset ()

7.61.1 Detailed Description

Definition at line 20 of file LevelsTab.h.

7.61.2 Constructor & Destructor Documentation

7.61.2.1 LevelsTab()

```
LevelsTab::LevelsTab (
            QWidget * parent = 0 )
```

Definition at line 10 of file LevelsTab.cpp.

7.61.3 Member Function Documentation

7.61.3.1 addLevel [1/2]

```
void LevelsTab::addLevel ( )  [slot]
```

Definition at line 185 of file LevelsTab.cpp.

7.61.3.2 addLevel [2/2]

```
void LevelsTab::addLevel (
            LevelsData level,
            bool fromFile )  [slot]
```

Definition at line 199 of file LevelsTab.cpp.

7.61.3.3 calculateChannels()

```
QList< ChannelsData > LevelsTab::calculateChannels (
            int levelIndex )
```

Definition at line 365 of file LevelsTab.cpp.

7.61.3.4 editLevel

```
void LevelsTab::editLevel ( )  [slot]
```

Definition at line 230 of file LevelsTab.cpp.

7.61.3.5 readExistingPair

```
void LevelsTab::readExistingPair (
            PairsData ,
            int ,
            bool  )  [signal]
```

7.61.3.6 readNewPair

```
void LevelsTab::readNewPair (
            PairsData ,
            int ,
            bool  )  [signal]
```

7.61.3.7 readNuclearFile()

```
bool LevelsTab::readNuclearFile (
            QTextStream & inStream )
```

Definition at line 680 of file LevelsTab.cpp.

7.61.3.8 removeLevel

```
void LevelsTab::removeLevel ( )  [slot]
```

Definition at line 220 of file LevelsTab.cpp.

7.61.3.9 reset()

```
void LevelsTab::reset ( )
```

Definition at line 818 of file LevelsTab.cpp.

7.61.3.10 setPairsModel()

```
void LevelsTab::setPairsModel (
            PairsModel * model )
```

Definition at line 180 of file LevelsTab.cpp.

7.61.3.11 showInfo

```
void LevelsTab::showInfo (
            int which = 0,
            QString title = "" )  [slot]
```

Definition at line 833 of file LevelsTab.cpp.

7.61.3.12 updateButtons

```
void LevelsTab::updateButtons (
            const QItemSelection & selection )  [slot]
```

Definition at line 274 of file LevelsTab.cpp.

7.61.3.13 updateChannelsLevelAdded()

```
void LevelsTab::updateChannelsLevelAdded (
            int levelIndex )
```

Definition at line 284 of file LevelsTab.cpp.

7.61.3.14 updateChannelsLevelDeleted()

```
void LevelsTab::updateChannelsLevelDeleted (
            int levelIndex )
```

Definition at line 306 of file LevelsTab.cpp.

7.61.3.15 updateChannelsLevelEdited()

```
void LevelsTab::updateChannelsLevelEdited (
            int levelIndex )
```

Definition at line 324 of file LevelsTab.cpp.

7.61.3.16 updateChannelsPairAddedEdited

```
void LevelsTab::updateChannelsPairAddedEdited ( )  [slot]
```

Definition at line 430 of file LevelsTab.cpp.

7.61.3.17 updateChannelsPairRemoved

```
void LevelsTab::updateChannelsPairRemoved (
            int pairIndex )  [slot]
```

Definition at line 468 of file LevelsTab.cpp.

7.61.3.18 updateDetails

```
void LevelsTab::updateDetails (
            const QItemSelection & selection )  [slot]
```

Definition at line 487 of file LevelsTab.cpp.

7.61.3.19 updateFilter

```
void LevelsTab::updateFilter (
            const QItemSelection & selection )  [slot]
```

Definition at line 418 of file LevelsTab.cpp.

7.61.3.20 updateReducedWidth

```
void LevelsTab::updateReducedWidth (
            const QString & string )  [slot]
```

Definition at line 573 of file LevelsTab.cpp.

7.61.3.21 writeNuclearFile()

```
bool LevelsTab::writeNuclearFile (
            QTextStream & outStream )
```

Definition at line 584 of file LevelsTab.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsTab.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InTabDocs.cpp
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/LevelsTab.cpp

7.62 MatrixInv Class Reference

A Function class to perform matrix inversion.

```
#include <MatrixInv.h>
```

Public Member Functions

- MatrixInv (const matrix_c &)
- const matrix_c & inverse () const

7.62.1 Detailed Description

A Function class to perform matrix inversion.

The MatrixInv class is a function class for matrix inversion using the GSL functions.

Definition at line 12 of file MatrixInv.h.

7.62.2 Constructor & Destructor Documentation

7.62.2.1 MatrixInv()

```
MatrixInv::MatrixInv (
            const matrix_c & A )
```

The MatrixInv constructor takes a complex matrix as an argument and stores the inverse in a private member variable accessable by the MatrixInv::inverse() function.

Definition at line 11 of file MatrixInv.cpp.

7.62.3 Member Function Documentation

7.62.3.1 inverse()

```
const matrix_c & MatrixInv::inverse ( ) const [inline]
```

The function returns the inverse as calculated by the constructor.

Definition at line 18 of file MatrixInv.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/MatrixInv.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/MatrixInv.cpp

7.63 MGroup Class Reference

An AZURE internal reaction pathway.

```
#include <MGroup.h>
```

Public Member Functions

- MGroup (int, int, int)
- int GetChNum () const
- int GetChpNum () const
- int GetJNum () const
- double GetStatSpinFactor () const
- void SetStatSpinFactor (double)

7.63.1 Detailed Description

An AZURE internal reaction pathway.

An MGroup in AZURE represents a given entrance and exit channel through a J^{π} group. These can be visualized as paths entering one row of the T-Matrix, and exiting through a column.

Definition at line 13 of file MGroup.h.

7.63.2 Constructor & Destructor Documentation

7.63.2.1 MGroup()

```
MGroup::MGroup (
              int jGroupNum,
              int channelNum,
              int channelPrimeNum )
```

This constructor is used to create an MGroup object with reference to positions in the JGroup and subsequent AChannel vectors.

Definition at line 7 of file MGroup.cpp.

7.63.3 Member Function Documentation

7.63.3.1 GetChNum()

```
int MGroup::GetChNum ( ) const
```

Returns the position of the entrance channel in the AChannel vector below the corresponding JGroup object.

Definition at line 15 of file MGroup.cpp.

7.63.3.2 GetChpNum()

```
int MGroup::GetChpNum ( ) const
```

Returns the position of the exit channel in the AChannel vector below the corresponding JGroup object.

Definition at line 23 of file MGroup.cpp.

7.63.3.3 GetJNum()

```
int MGroup::GetJNum ( ) const
```

Returns the position of the J^{π} group in the JGroup vector.

Definition at line 31 of file MGroup.cpp.

7.63.3.4 GetStatSpinFactor()

```
double MGroup::GetStatSpinFactor ( ) const
```

Returns the statistical spin factor, g_J, for the reaction pathway.

Definition at line 39 of file MGroup.cpp.

7.63.3.5 SetStatSpinFactor()

```
void MGroup::SetStatSpinFactor (
             double spinFactor )
```

Sets the statistical spin factor, g_J, for the reaction pathway.

Definition at line 47 of file MGroup.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/MGroup.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/MGroup.cpp

7.64 NFIntegral Class Reference

A function class to calculate the channel integrals in the denominator of the $N_f^{1/2}$ term.

```
#include <NFIntegral.h>
```

Public Member Functions

- NFIntegral (PPair ∗pPair)
- ∼NFIntegral ()
- double operator() (int lFinal, double levelEnergy)
- double chanRad () const
- double totalSepE () const

7.64.1 Detailed Description

A function class to calculate the channel integrals in the denominator of the $N_f^{1/2}$ term.

The NFIntegral class returns the channel integral given by $\int_a^\infty \left[\frac{W_c(kr)}{W_c k a_c}\right]^2$.

Definition at line 15 of file NFIntegral.h.

7.64.2 Constructor & Destructor Documentation

7.64.2.1 NFIntegral()

```
NFIntegral::NFIntegral (
            PPair * pPair )  [inline]
```

The NFIntegral object is created with reference to a PPair object. A WhitFunc object is also created.

Definition at line 20 of file NFIntegral.h.

7.64.2.2 ∼NFIntegral()

```
NFIntegral::∼NFIntegral ( )  [inline]
```

The WhitFunc object is destroyed with the NFIntegral object.

Definition at line 28 of file NFIntegral.h.

7.64.3 Member Function Documentation

7.64.3.1 chanRad()

```
double NFIntegral::chanRad ( ) const  [inline]
```

Returns the channel radius of the particle pair.

Definition at line 40 of file NFIntegral.h.

7.64.3.2 operator()()

```
double NFIntegral::operator() (
            int lFinal,
            double levelEnergy )
```

The parenthesis operator is defined so the instance can be callable as a function. The final channel orbital angular momentum and final state energy in the compound system are passed as dependent variables.

Definition at line 15 of file NFIntegral.cpp.

7.64.3.3 totalSepE()

```
double NFIntegral::totalSepE ( ) const  [inline]
```

Returns the total seperation energy of the particle pair.

Definition at line 44 of file NFIntegral.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/NFIntegral.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/NFIntegral.cpp

7.65 NucLine Class Reference

A class to read and store a line from a nuclear input file.

```
#include <NucLine.h>
```

Public Member Functions

- NucLine (std::istream &stream)
- double levelJ () const
- int levelPi () const
- double levelE () const
- int levelFix () const
- int aa () const
- int ir () const
- double s () const
- int l () const
- int levelID () const
- int isActive () const
- int channelFix () const
- double gamma () const
- double j1 () const
- int pi1 () const
- double j2 () const
- int pi2 () const
- double e2 () const
- double m1 () const
- double m2 () const
- int z1 () const
- int z2 () const
- double entranceSepE () const
- double sepE () const
- int j3 () const
- int pi3 () const
- double e3 () const
- int pType () const
- double chRad () const
- double g1 () const
- double g2 () const
- double ecMultMask () const

7.65.1 Detailed Description

A class to read and store a line from a nuclear input file.

The NucLine class reads and stores a line from a formatted nuclear input file.

Definition at line 13 of file NucLine.h.

7.65.2 Constructor & Destructor Documentation

7.65.2.1 NucLine()

```
NucLine::NucLine (
            std::istream & stream )  [inline]
```

Constructor fills the NucLine object from an input stream.

Definition at line 18 of file NucLine.h.

7.65.3 Member Function Documentation

7.65.3.1 aa()

```
int NucLine::aa ( ) const  [inline]
```

Returns the entrance key of the corresponding particle pair for the channel line.

Definition at line 49 of file NucLine.h.

7.65.3.2 channelFix()

```
int NucLine::channelFix ( ) const  [inline]
```

Returns non-zero if the reduced width amplitude for the channel is not to be varied.

Definition at line 76 of file NucLine.h.

7.65.3.3 chRad()

```
double NucLine::chRad ( ) const  [inline]
```

Returns the channel radius for the corresponding particle pair.

Definition at line 146 of file NucLine.h.

7.65.3.4 e2()

```
double NucLine::e2 ( ) const [inline]
```

Returns the excitation energy of the heavy particle in the corresponding pair.

Definition at line 101 of file NucLine.h.

7.65.3.5 e3()

```
double NucLine::e3 ( ) const [inline]
```

This function is depriciated and not used.

Definition at line 137 of file NucLine.h.

7.65.3.6 ecMultMask()

```
double NucLine::ecMultMask ( ) const [inline]
```

Returns the external capture multiplicity mask for the corresponding pair.

Definition at line 158 of file NucLine.h.

7.65.3.7 entranceSepE()

```
double NucLine::entranceSepE ( ) const [inline]
```

This function is depriciated and not used.

Definition at line 121 of file NucLine.h.

7.65.3.8 g1()

```
double NucLine::g1 ( ) const [inline]
```

Returns the g-factor for the light particle in the corresponding pair.

Definition at line 150 of file NucLine.h.

7.65.3.9 g2()

```
double NucLine::g2 ( ) const [inline]
```

Returns the g-factor for the heavy particle in the corresponding pair.

Definition at line 154 of file NucLine.h.

7.65.3.10 gamma()

```
double NucLine::gamma ( ) const [inline]
```

Returns the initial reduced width or physical amplitude for the channel.

Definition at line 80 of file NucLine.h.

7.65.3.11 ir()

```
int NucLine::ir ( ) const [inline]
```

Returns the exit key of the corresponding particle pair for the channel line.

Definition at line 54 of file NucLine.h.

7.65.3.12 isActive()

```
int NucLine::isActive ( ) const [inline]
```

Returns non-zero if the corresponding level for the channel line is to be used in the calculation.

Definition at line 71 of file NucLine.h.

7.65.3.13 j1()

```
double NucLine::j1 ( ) const [inline]
```

Returns the spin of the light particle in the corresponding pair.

Definition at line 84 of file NucLine.h.

7.65.3.14 j2()

```
double NucLine::j2 ( ) const [inline]
```

Returns the spin of the heavy particle in the corresponding pair.

Definition at line 92 of file NucLine.h.

7.65.3.15 j3()

```
int NucLine::j3 ( ) const [inline]
```

This function is depriciated and not used.

Definition at line 129 of file NucLine.h.

7.65.3.16 l()

```
int NucLine::l ( ) const [inline]
```

Returns the orbital angular momentum of the channel.

Definition at line 62 of file NucLine.h.

7.65.3.17 levelE()

```
double NucLine::levelE ( ) const [inline]
```

Returns the excitation energy of the corresponding level for the channel line.

Definition at line 39 of file NucLine.h.

7.65.3.18 levelFix()

```
int NucLine::levelFix ( ) const [inline]
```

Returns non-zero if the corresponding level for the channel line is not to be varied in the fit.

Definition at line 44 of file NucLine.h.

7.65.3.19 levelID()

```
int NucLine::levelID ( ) const [inline]
```

Returns an indexing variable used by the graphical setup program.

Definition at line 66 of file NucLine.h.

7.65.3.20 levelJ()

```
double NucLine::levelJ ( ) const [inline]
```

Returns the spin of the corresponding level for the channel line.

Definition at line 30 of file NucLine.h.

7.65.3.21 levelPi()

```
int NucLine::levelPi ( ) const [inline]
```

Returns the parity of the corresponding level for the channel line.

Definition at line 34 of file NucLine.h.

7.65.3.22 m1()

```
double NucLine::m1 ( ) const  [inline]
```

Returns the mass of the light particle in the corresponding pair.

Definition at line 105 of file NucLine.h.

7.65.3.23 m2()

```
double NucLine::m2 ( ) const  [inline]
```

Returns the mass of the heavy particle in the corresponding pair.

Definition at line 109 of file NucLine.h.

7.65.3.24 pi1()

```
int NucLine::pi1 ( ) const  [inline]
```

Returns the parity of the light particle in the corresponding pair.

Definition at line 88 of file NucLine.h.

7.65.3.25 pi2()

```
int NucLine::pi2 ( ) const  [inline]
```

Returns the parity of the heavy particle in the corresponding pair.

Definition at line 96 of file NucLine.h.

7.65.3.26 pi3()

```
int NucLine::pi3 ( ) const  [inline]
```

This function is depriciated and not used.

Definition at line 133 of file NucLine.h.

7.65.3.27 pType()

```
int NucLine::pType ( ) const  [inline]
```

Returns 0 for particle-particle and 10 for particle-gamma types in the corresponding pair.

Definition at line 142 of file NucLine.h.

7.65.3.28 s()

```
double NucLine::s ( ) const  [inline]
```

Returns the channel spin of the channel.

Definition at line 58 of file NucLine.h.

7.65.3.29 sepE()

```
double NucLine::sepE ( ) const  [inline]
```

Returns the separation energy for the corresponding particle pair.

Definition at line 125 of file NucLine.h.

7.65.3.30 z1()

```
int NucLine::z1 ( ) const  [inline]
```

Returns the charge of the light particle in the corresponding pair.

Definition at line 113 of file NucLine.h.

7.65.3.31 z2()

```
int NucLine::z2 ( ) const  [inline]
```

Returns the charge of the heavy particle in the corresponding pair.

Definition at line 117 of file NucLine.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/NucLine.h

7.66 ODE_integration Class Reference

```
#include <ode_int.H>
```

Public Member Functions

- ODE_integration (const std::complex< double > &l_1, const std::complex< double > &two_eta_1)
- void operator() (const std::complex< double > &r0, const std::complex< double > &u0, const std::complex< double > &du0, const std::complex< double > &r, std::complex< double > &u, std::complex< double > &du) const

7.66.1 Detailed Description

Definition at line 12 of file ode_int.H.

7.66.2 Constructor & Destructor Documentation

7.66.2.1 ODE_integration()

```
ODE_integration::ODE_integration (
            const std::complex< double > & l_1,
            const std::complex< double > & two_eta_1 )  [inline]
```

Definition at line 15 of file ode_int.H.

7.66.3 Member Function Documentation

7.66.3.1 operator()()

```
void ODE_integration::operator() (
            const std::complex< double > & r0,
            const std::complex< double > & u0,
            const std::complex< double > & du0,
            const std::complex< double > & r,
            std::complex< double > & u,
            std::complex< double > & du ) const
```

Definition at line 118 of file ode_int.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/ode_int.H
- /Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/ode_int.cpp

7.67 PairsData Struct Reference

```
#include <PairsModel.h>
```

Public Attributes

- double lightJ
- int lightPi
- int lightZ
- double lightM
- double lightG
- double heavyJ
- int heavyPi
- int heavyZ
- double heavyM
- double heavyG
- double excitationEnergy
- double seperationEnergy
- double channelRadius
- int pairType
- int ecMultMask

Static Public Attributes

- static const int SIZE = 15

7.67.1 Detailed Description

Definition at line 7 of file PairsModel.h.

7.67.2 Member Data Documentation

7.67.2.1 channelRadius

```
double PairsData::channelRadius
```

Definition at line 21 of file PairsModel.h.

7.67.2.2 ecMultMask

```
int PairsData::ecMultMask
```

Definition at line 23 of file PairsModel.h.

7.67.2.3 excitationEnergy

```
double PairsData::excitationEnergy
```

Definition at line 19 of file PairsModel.h.

7.67.2.4 heavyG

```
double PairsData::heavyG
```

Definition at line 18 of file PairsModel.h.

7.67.2.5 heavyJ

```
double PairsData::heavyJ
```

Definition at line 14 of file PairsModel.h.

7.67.2.6 heavyM

```
double PairsData::heavyM
```

Definition at line 17 of file PairsModel.h.

7.67.2.7 heavyPi

```
int PairsData::heavyPi
```

Definition at line 15 of file PairsModel.h.

7.67.2.8 heavyZ

```
int PairsData::heavyZ
```

Definition at line 16 of file PairsModel.h.

7.67.2.9 lightG

```
double PairsData::lightG
```

Definition at line 13 of file PairsModel.h.

7.67.2.10 lightJ

```
double PairsData::lightJ
```

Definition at line 9 of file PairsModel.h.

7.67.2.11 lightM

```
double PairsData::lightM
```

Definition at line 12 of file PairsModel.h.

7.67.2.12 lightPi

```
int PairsData::lightPi
```

Definition at line 10 of file PairsModel.h.

7.67.2.13 lightZ

```
int PairsData::lightZ
```

Definition at line 11 of file PairsModel.h.

7.67.2.14 pairType

```
int PairsData::pairType
```

Definition at line 22 of file PairsModel.h.

7.67.2.15 seperationEnergy

```
double PairsData::seperationEnergy
```

Definition at line 20 of file PairsModel.h.

7.67.2.16 SIZE

```
const int PairsData::SIZE = 15  [static]
```

Definition at line 8 of file PairsModel.h.

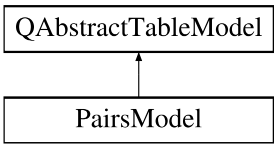
The documentation for this struct was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsModel.h

7.68 PairsModel Class Reference

```
#include <PairsModel.h>
```

Inheritance diagram for PairsModel:



Public Member Functions

- PairsModel (QObject ∗parent=0)
- int rowCount (const QModelIndex &parent) const
- int columnCount (const QModelIndex &parent) const
- QVariant data (const QModelIndex &index, int role) const
- QVariant headerData (int section, Qt::Orientation orientation, int role) const
- bool setData (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- bool insertRows (int position, int rows, const QModelIndex &index=QModelIndex())
- bool removeRows (int position, int rows, const QModelIndex &index=QModelIndex())
- int isPair (const PairsData &pair) const
- int numPairs () const
- QList< PairsData > getPairs () const
- QString getParticleLabel (const PairsData &pair, int which=-1) const
- QString getReactionLabel (const PairsData &firstPair, const PairsData &secondPair)
- QString getReactionLabelTotalCapture (const PairsData &firstPair)
- QString getSpinLabel (const PairsData &pair, int which) const

7.68.1 Detailed Description

Definition at line 26 of file PairsModel.h.

7.68.2 Constructor & Destructor Documentation

7.68.2.1 PairsModel()

```
PairsModel::PairsModel (
            QObject * parent = 0 )
```

Definition at line 5 of file PairsModel.cpp.

7.68.3 Member Function Documentation

7.68.3.1 columnCount()

```
int PairsModel::columnCount (
            const QModelIndex & parent ) const
```

Definition at line 13 of file PairsModel.cpp.

7.68.3.2 data()

```
QVariant PairsModel::data (
            const QModelIndex & index,
            int role ) const
```

Definition at line 18 of file PairsModel.cpp.

7.68.3.3 getPairs()

```
QList< PairsData > PairsModel::getPairs ( ) const [inline]
```

Definition at line 42 of file PairsModel.h.

7.68.3.4 getParticleLabel()

```
QString PairsModel::getParticleLabel (
            const PairsData & pair,
            int which = -1 ) const
```

Definition at line 187 of file PairsModel.cpp.

7.68.3.5 getReactionLabel()

```
QString PairsModel::getReactionLabel (
            const PairsData & firstPair,
            const PairsData & secondPair )
```

Definition at line 240 of file PairsModel.cpp.

7.68.3.6 getReactionLabelTotalCapture()

```
QString PairsModel::getReactionLabelTotalCapture (
            const PairsData & firstPair )
```

Definition at line 287 of file PairsModel.cpp.

7.68.3.7 getSpinLabel()

```
QString PairsModel::getSpinLabel (
            const PairsData & pair,
            int which ) const
```

Definition at line 330 of file PairsModel.cpp.

7.68.3.8 headerData()

```
QVariant PairsModel::headerData (
            int section,
            Qt::Orientation orientation,
            int role ) const
```

Definition at line 65 of file PairsModel.cpp.

7.68.3.9 insertRows()

```
bool PairsModel::insertRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 136 of file PairsModel.cpp.

7.68.3.10 isPair()

```
int PairsModel::isPair (
            const PairsData & pair ) const
```

Definition at line 161 of file PairsModel.cpp.

7.68.3.11 numPairs()

```
int PairsModel::numPairs ( ) const [inline]
```

Definition at line 41 of file PairsModel.h.

7.68.3.12 removeRows()

```
bool PairsModel::removeRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 149 of file PairsModel.cpp.

7.68.3.13 rowCount()

```
int PairsModel::rowCount (
            const QModelIndex & parent ) const
```

Definition at line 8 of file PairsModel.cpp.

7.68.3.14 setData()

```
bool PairsModel::setData (
            const QModelIndex & index,
            const QVariant & value,
            int role = Qt::EditRole )
```

Definition at line 108 of file PairsModel.cpp.

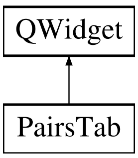
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsModel.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PairsModel.cpp

7.69 PairsTab Class Reference

```
#include <PairsTab.h>
```

Inheritance diagram for PairsTab:



Public Slots

- void addPair ()
- void addPair (PairsData pair, int pairIndex, bool fromFile)
- void editPair ()
- void editPair (PairsData pair, int pairIndex, bool fromFile)
- void removePair ()
- void updateButtons (const QItemSelection &selection)
- void showInfo (int which=0, QString title="")

Signals

- void pairAdded (int)
- void pairRemoved (int)
- void pairEdited (int)

Public Member Functions

- PairsTab (QWidget ∗parent=0)
- PairsModel ∗ getPairsModel ()
- bool parseOldECSection (QTextStream &)

7.69.1 Detailed Description

Definition at line 23 of file PairsTab.h.

7.69.2 Constructor & Destructor Documentation

7.69.2.1 PairsTab()

```
PairsTab::PairsTab (
            QWidget * parent = 0 )
```

Definition at line 14 of file PairsTab.cpp.

7.69.3 Member Function Documentation

7.69.3.1 addPair [1/2]

```
void PairsTab::addPair ( )  [slot]
```

Definition at line 86 of file PairsTab.cpp.

7.69.3.2 addPair [2/2]

```
void PairsTab::addPair (
            PairsData pair,
            int pairIndex,
            bool fromFile )  [slot]
```

Definition at line 122 of file PairsTab.cpp.

7.69.3.3 editPair [1/2]

```
void PairsTab::editPair ( )  [slot]
```

Definition at line 184 of file PairsTab.cpp.

7.69.3.4 editPair [2/2]

```
void PairsTab::editPair (
            PairsData pair,
            int pairIndex,
            bool fromFile ) [slot]
```

Definition at line 297 of file PairsTab.cpp.

7.69.3.5 getPairsModel()

```
PairsModel * PairsTab::getPairsModel ( )
```

Definition at line 82 of file PairsTab.cpp.

7.69.3.6 pairAdded

```
void PairsTab::pairAdded (
            int ) [signal]
```

7.69.3.7 pairEdited

```
void PairsTab::pairEdited (
            int ) [signal]
```

7.69.3.8 pairRemoved

```
void PairsTab::pairRemoved (
            int ) [signal]
```

7.69.3.9 parseOldECSection()

```
bool PairsTab::parseOldECSection (
            QTextStream & inStream )
```

Definition at line 357 of file PairsTab.cpp.

7.69.3.10 removePair

```
void PairsTab::removePair ( ) [slot]
```

Definition at line 165 of file PairsTab.cpp.

7.69.3.11 showInfo

```
void PairsTab::showInfo (
            int which = 0,
            QString title = "" )  [slot]
```

Definition at line 382 of file PairsTab.cpp.

7.69.3.12 updateButtons

```
void PairsTab::updateButtons (
            const QItemSelection & selection )  [slot]
```

Definition at line 347 of file PairsTab.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsTab.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InTabDocs.cpp
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PairsTab.cpp

7.70 PlotEntry Class Reference

```
#include <AZUREPlot.h>
```

Public Member Functions

- PlotEntry (int type, int entranceKey, int exitKey, int index, QString filename)
- ∼PlotEntry ()
- int type () const
- bool readData ()
- void attach (QwtPlot ∗, int, int, QwtSymbol::Style)
- void detach ()

Friends

- class AZUREPlot

7.70.1 Detailed Description

Definition at line 32 of file AZUREPlot.h.

7.70.2 Constructor & Destructor Documentation

7.70.2.1 PlotEntry()

```
PlotEntry::PlotEntry (
            int type,
            int entranceKey,
            int exitKey,
            int index,
            QString filename )
```

Definition at line 40 of file AZUREPlot.cpp.

7.70.2.2 ∼PlotEntry()

```
PlotEntry::∼PlotEntry ( )
```

Definition at line 45 of file AZUREPlot.cpp.

7.70.3 Member Function Documentation

7.70.3.1 attach()

```
void PlotEntry::attach (
            QwtPlot * plot,
            int xAxisType,
            int yAxisType,
            QwtSymbol::Style style )
```

Definition at line 111 of file AZUREPlot.cpp.

7.70.3.2 detach()

```
void PlotEntry::detach ( )
```

Definition at line 223 of file AZUREPlot.cpp.

7.70.3.3 readData()

```
bool PlotEntry::readData ( )
```

Definition at line 51 of file AZUREPlot.cpp.

7.70.3.4 type()

```
int PlotEntry::type ( ) const [inline]
```

Definition at line 37 of file AZUREPlot.h.

7.70.4 Friends And Related Symbol Documentation

7.70.4.1 AZUREPlot

```
friend class AZUREPlot  [friend]
```

Definition at line 44 of file AZUREPlot.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREPlot.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZUREPlot.cpp

7.71 PlotPoint Struct Reference

```
#include <AZUREPlot.h>
```

Public Attributes

- double energy
- double excitationEnergy
- double angle
- double fitCrossSection
- double fitSFactor
- double dataCrossSection
- double dataErrorCrossSection
- double dataSFactor
- double dataErrorSFactor

7.71.1 Detailed Description

Definition at line 12 of file AZUREPlot.h.

7.71.2 Member Data Documentation

7.71.2.1 angle

```
double PlotPoint::angle
```

Definition at line 15 of file AZUREPlot.h.

7.71.2.2 dataCrossSection

```
double PlotPoint::dataCrossSection
```

Definition at line 18 of file AZUREPlot.h.

7.71.2.3 dataErrorCrossSection

`double PlotPoint::dataErrorCrossSection`

Definition at line 19 of file AZUREPlot.h.

7.71.2.4 dataErrorSFactor

`double PlotPoint::dataErrorSFactor`

Definition at line 21 of file AZUREPlot.h.

7.71.2.5 dataSFactor

`double PlotPoint::dataSFactor`

Definition at line 20 of file AZUREPlot.h.

7.71.2.6 energy

`double PlotPoint::energy`

Definition at line 13 of file AZUREPlot.h.

7.71.2.7 excitationEnergy

`double PlotPoint::excitationEnergy`

Definition at line 14 of file AZUREPlot.h.

7.71.2.8 fitCrossSection

`double PlotPoint::fitCrossSection`

Definition at line 16 of file AZUREPlot.h.

7.71.2.9 fitSFactor

`double PlotPoint::fitSFactor`

Definition at line 17 of file AZUREPlot.h.

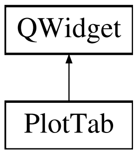
The documentation for this struct was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREPlot.h

7.72 PlotTab Class Reference

```
#include <PlotTab.h>
```

Inheritance diagram for PlotTab:



Public Slots

- void draw ()
- void xAxisTypeChanged ()
- void yAxisTypeChanged ()
- void xAxisLogScaleChanged (bool)
- void yAxisLogScaleChanged (bool)
- void showInfo (int which=0, QString title="")

Public Member Functions

- PlotTab (Config &config, SegmentsDataModel ∗dataModel, SegmentsTestModel ∗testModel, QWidget ∗parent=0)
- QList< PlotEntry ∗ > getDataSegments ()
- QList< PlotEntry ∗ > getTestSegments ()
- void reset ()

Friends

- class AZUREPlot

7.72.1 Detailed Description

Definition at line 39 of file PlotTab.h.

7.72.2 Constructor & Destructor Documentation

7.72.2.1 PlotTab()

```
PlotTab::PlotTab (
            Config & config,
            SegmentsDataModel * dataModel,
            SegmentsTestModel * testModel,
            QWidget * parent = 0 )
```

Definition at line 50 of file PlotTab.cpp.

7.72.3 Member Function Documentation

7.72.3.1 draw

```
void PlotTab::draw ( ) [slot]
```

Definition at line 223 of file PlotTab.cpp.

7.72.3.2 getDataSegments()

```
QList< PlotEntry * > PlotTab::getDataSegments ( )
```

Definition at line 165 of file PlotTab.cpp.

7.72.3.3 getTestSegments()

```
QList< PlotEntry * > PlotTab::getTestSegments ( )
```

Definition at line 194 of file PlotTab.cpp.

7.72.3.4 reset()

```
void PlotTab::reset ( )
```

Definition at line 248 of file PlotTab.cpp.

7.72.3.5 showInfo

```
void PlotTab::showInfo (
            int which = 0,
            QString title = "" ) [slot]
```

Definition at line 256 of file PlotTab.cpp.

7.72.3.6 xAxisLogScaleChanged

```
void PlotTab::xAxisLogScaleChanged (
            bool checked ) [slot]
```

Definition at line 240 of file PlotTab.cpp.

7.72.3.7 xAxisTypeChanged

```
void PlotTab::xAxisTypeChanged ( ) [slot]
```

Definition at line 229 of file PlotTab.cpp.

7.72.3.8 yAxisLogScaleChanged

```
void PlotTab::yAxisLogScaleChanged (
            bool checked )  [slot]
```

Definition at line 244 of file PlotTab.cpp.

7.72.3.9 yAxisTypeChanged

```
void PlotTab::yAxisTypeChanged ( )  [slot]
```

Definition at line 233 of file PlotTab.cpp.

7.72.4 Friends And Related Symbol Documentation

7.72.4.1 AZUREPlot

```
friend class AZUREPlot  [friend]
```

Definition at line 58 of file PlotTab.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PlotTab.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InTabDocs.cpp
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PlotTab.cpp

7.73 PPair Class Reference

An AZURE Particle Pair.

`#include <PPair.h>`

Public Member Functions

- PPair (NucLine)
- bool IsEntrance () const
- int GetZ (int) const
- int GetPi (int) const
- int GetPType () const
- int NumDecays () const
- int IsDecay (Decay)
- int IsDecay (int)
- int GetPairKey () const
- double GetM (int) const
- double GetG (int) const
- double GetJ (int) const
- double GetExE () const
- double GetSepE () const
- double GetChRad () const
- double GetRedMass () const
- double GetI1I2Factor () const
- void AddDecay (Decay)
- void SetEntrance ()
- Decay * GetDecay (int)

7.73.1 Detailed Description

An AZURE Particle Pair.

In R-Matrix theory, the configuration space in the external region is decomposed into combinations of particle pairs, traditionally given by the symbol α. In AZURE, these particle pair are represented by a PPair object. PPair objects are containers for vectors of Decay objects.

Definition at line 16 of file PPair.h.

7.73.2 Constructor & Destructor Documentation

7.73.2.1 PPair()

```
PPair::PPair (
            NucLine nucLine )
```

A particle pair object is created from and entry in the nuclear input file.

Definition at line 9 of file PPair.cpp.

7.73.3 Member Function Documentation

7.73.3.1 AddDecay()

```
void PPair::AddDecay (
            Decay decay )
```

Adds a decay particle pair to the Decay vector.

Definition at line 183 of file PPair.cpp.

7.73.3.2 GetChRad()

```
double PPair::GetChRad ( ) const
```

Returns the channel radius of the particle pair.

Definition at line 159 of file PPair.cpp.

7.73.3.3 GetDecay()

```
Decay * PPair::GetDecay (
            int decayNum )
```

Returns a pointer to the decay particle pair specified by a position in the Decay vector.

Definition at line 199 of file PPair.cpp.

7.73.3.4 GetExE()

```
double PPair::GetExE ( ) const
```

Returns the excitation energy of the particle pair.

Definition at line 143 of file PPair.cpp.

7.73.3.5 GetG()

```
double PPair::GetG (
            int particle ) const
```

Returns the g-factor of the specified particle (1 or 2).

Definition at line 127 of file PPair.cpp.

7.73.3.6 GetI1I2Factor()

```
double PPair::GetI1I2Factor ( ) const
```

Returns the factor $\frac{1}{(2I_1+1)(2I_2+1)}$ of the particle pair.

Definition at line 175 of file PPair.cpp.

7.73.3.7 GetJ()

```
double PPair::GetJ (
            int particle ) const
```

Returns the total spin of the specified particle (1 or 2).

Definition at line 135 of file PPair.cpp.

7.73.3.8 GetM()

```
double PPair::GetM (
            int particle ) const
```

Returns the mass number of the specified particle (1 or 2).

Definition at line 119 of file PPair.cpp.

7.73.3.9 GetPairKey()

```
int PPair::GetPairKey ( ) const
```

Returns the pair key for the particle pair.

Definition at line 111 of file PPair.cpp.

7.73.3.10 GetPi()

```
int PPair::GetPi (
            int particle ) const
```

Returns the parity of the specified particle (1 or 2).

Definition at line 52 of file PPair.cpp.

7.73.3.11 GetPType()

```
int PPair::GetPType ( ) const
```

Returns the integer particle pair type. Pair types currently used in AZURE are 0: particle,particle and 10: particle,gamma.

Definition at line 60 of file PPair.cpp.

7.73.3.12 GetRedMass()

```
double PPair::GetRedMass ( ) const
```

Returns the reduced mass of the particle pair.

Definition at line 167 of file PPair.cpp.

7.73.3.13 GetSepE()

```
double PPair::GetSepE ( ) const
```

Returns the seperation energy of the particle pair.

Definition at line 151 of file PPair.cpp.

7.73.3.14 GetZ()

```
int PPair::GetZ (
            int particle ) const
```

Returns the atomic number of the specified particle (1 or 2).

Definition at line 44 of file PPair.cpp.

7.73.3.15 IsDecay() [1/2]

```
int PPair::IsDecay (
            Decay decay )
```

Tests a given decay particle pair to determine if it is in the Decay vector. If the decay particle pair exists in the vector, the position in the vector is returned. Otherwise, the function returns 0.

Definition at line 78 of file PPair.cpp.

7.73.3.16 IsDecay() [2/2]

```
int PPair::IsDecay (
            int pairNum )
```

Tests a given particle pair number to determine if there exists a corresponding particle pair decay in the Decay vector. If the object exists, the position in the vector is returned. Otherwise, the function returns 0.

Definition at line 95 of file PPair.cpp.

7.73.3.17 IsEntrance()

```
bool PPair::IsEntrance ( ) const
```

Returns true if the particle pair is an internal entrance pair, otherwise returns false.

Definition at line 36 of file PPair.cpp.

7.73.3.18 NumDecays()

```
int PPair::NumDecays ( ) const
```

Returns the number of decay particle pairs for a given pair. Size of Decay vector will only be nonzero if PPair object is an entrance pair.

Definition at line 69 of file PPair.cpp.

7.73.3.19 SetEntrance()

```
void PPair::SetEntrance ( )
```

Sets the particle pair to be an internal entrance pair.

Definition at line 191 of file PPair.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/PPair.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/PPair.cpp

7.74 RateData Class Reference

A container structure for a reaction rate.

```
#include <ReactionRate.h>
```

Public Member Functions

- RateData (double t, double r)

  Constructor creates RateData object from a given temperature and rate value.
- bool operator< (const RateData &right) const

  This function defines the ¨less than¨ operator for use in sorting.

Public Attributes

- double temperature

  Temperature at which the rate was calculated.
- double rate

  Reaction rate at corresponding temperature.

7.74.1 Detailed Description

A container structure for a reaction rate.

The RateData container structure hold a temperature and the corresponding reaction rate.

Definition at line 16 of file ReactionRate.h.

7.74.2 Constructor & Destructor Documentation

7.74.2.1 RateData()

```
RateData::RateData (
            double t,
            double r )  [inline]
```

Constructor creates RateData object from a given temperature and rate value.

Definition at line 19 of file ReactionRate.h.

7.74.3 Member Function Documentation

7.74.3.1 operator<()

```
bool RateData::operator< (
            const RateData & right ) const  [inline]
```

This function defines the ¨less than¨ operator for use in sorting.

Definition at line 22 of file ReactionRate.h.

7.74.4 Member Data Documentation

7.74.4.1 rate

```
double RateData::rate
```

Reaction rate at corresponding temperature.

Definition at line 28 of file ReactionRate.h.

7.74.4.2 temperature

```
double RateData::temperature
```

Temperature at which the rate was calculated.

Definition at line 26 of file ReactionRate.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/ReactionRate.h

7.75 RateParams Struct Reference

A structure holding the reaction rate calculation configuration.

```
#include <Config.h>
```

Public Attributes

- bool useFile

  False for looped temperatures, true for temperatures from file.
- std::string temperatureFile

  String containing filename with temperatures to use.
- int entrancePair

  The entrance pair number for the rate calculation.
- int exitPair

  The exit pair number for the rate calculation.
- double minTemp

  The minimum temperature for the rate calculation.
- double maxTemp

  The maximum temperature for the rate calculation.
- double tempStep

  The temperature step for the rate calculation.

7.75.1 Detailed Description

A structure holding the reaction rate calculation configuration.

The RateParams structure holds the configuration information for a reaction rate calculation.

Definition at line 13 of file Config.h.

7.75.2 Member Data Documentation

7.75.2.1 entrancePair

```
int RateParams::entrancePair
```

The entrance pair number for the rate calculation.

Definition at line 19 of file Config.h.

7.75.2.2 exitPair

```
int RateParams::exitPair
```

The exit pair number for the rate calculation.

Definition at line 21 of file Config.h.

7.75.2.3 maxTemp

```
double RateParams::maxTemp
```

The maximum temperature for the rate calculation.

Definition at line 25 of file Config.h.

7.75.2.4 minTemp

```
double RateParams::minTemp
```

The minimum temperature for the rate calculation.

Definition at line 23 of file Config.h.

7.75.2.5 temperatureFile

```
std::string RateParams::temperatureFile
```

String containing filename with temperatures to use.

Definition at line 17 of file Config.h.

7.75.2.6 tempStep

```
double RateParams::tempStep
```

The temperature step for the rate calculation.

Definition at line 27 of file Config.h.

7.75.2.7 useFile

```
bool RateParams::useFile
```

False for looped temperatures, true for temperatures from file.

Definition at line 15 of file Config.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/Config.h

7.76 ReactionRate Class Reference

A function class to calculate the reaction rate.

```
#include <ReactionRate.h>
```

Public Member Functions

- ReactionRate (CNuc *, const vector_r &, const Config &, int, int)
- CNuc * compound () const
- const Config & configure () const
- int entranceKey () const
- int exitKey () const
- void CalculateRates ()
- void CalculateFileRates ()
- void WriteRates ()

7.76.1 Detailed Description

A function class to calculate the reaction rate.

The ReactionRate function class is used to calculate the reaction rate based on a set of R-Matrix parameters over a range of stellar temperatures.

Definition at line 38 of file ReactionRate.h.

7.76.2 Constructor & Destructor Documentation

7.76.2.1 ReactionRate()

```
ReactionRate::ReactionRate (
             CNuc * compound,
             const vector_r & params,
             const Config & configure,
             int entranceKey,
             int exitKey )
```

The ReactionRate object is created with reference to a CNuc object, a vector of Minuit parameters, a Config structure, and a set of entrance and exit pair keys.

Definition at line 71 of file ReactionRate.cpp.

7.76.3 Member Function Documentation

7.76.3.1 CalculateFileRates()

```
void ReactionRate::CalculateFileRates ( )
```

Calculates the astrophysical reaction rates at temperatures from file.

Definition at line 123 of file ReactionRate.cpp.

7.76.3.2 CalculateRates()

```
void ReactionRate::CalculateRates ( )
```

Calculates the astrophysical reaction rates over a range of stellar temperatures.

Definition at line 84 of file ReactionRate.cpp.

7.76.3.3 compound()

```
CNuc * ReactionRate::compound ( ) const  [inline]
```

Returns a pointer to the CNuc object.

Definition at line 48 of file ReactionRate.h.

7.76.3.4 configure()

```
const Config & ReactionRate::configure ( ) const  [inline]
```

Returns a reference to the Config structure.

Definition at line 52 of file ReactionRate.h.

7.76.3.5 entranceKey()

```
int ReactionRate::entranceKey ( ) const [inline]
```

Returns the entrance pair key.

Definition at line 56 of file ReactionRate.h.

7.76.3.6 exitKey()

```
int ReactionRate::exitKey ( ) const [inline]
```

Returns the exit pair key.

Definition at line 60 of file ReactionRate.h.

7.76.3.7 WriteRates()

```
void ReactionRate::WriteRates ( )
```

Writes the rates to an output file.

Definition at line 170 of file ReactionRate.cpp.

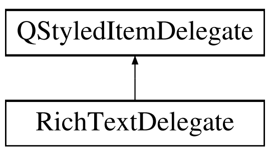
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/ReactionRate.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/ReactionRate.cpp

7.77 RichTextDelegate Class Reference

```
#include <RichTextDelegate.h>
```

Inheritance diagram for RichTextDelegate:



Protected Member Functions

- void paint (QPainter ∗painter, const QStyleOptionViewItem &option, const QModelIndex &index) const
- QSize sizeHint (const QStyleOptionViewItem &option, const QModelIndex &index) const

7.77.1 Detailed Description

Definition at line 8 of file RichTextDelegate.h.

7.77.2 Member Function Documentation

7.77.2.1 paint()

```
void RichTextDelegate::paint (
            QPainter * painter,
            const QStyleOptionViewItem & option,
            const QModelIndex & index ) const  [protected]
```

Definition at line 9 of file RichTextDelegate.cpp.

7.77.2.2 sizeHint()

```
QSize RichTextDelegate::sizeHint (
            const QStyleOptionViewItem & option,
            const QModelIndex & index ) const  [protected]
```

Definition at line 35 of file RichTextDelegate.cpp.

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

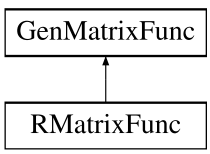
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RichTextDelegate.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RichTextDelegate.cpp

7.78 RMatrixFunc Class Reference

A function class to calculate the T-Matrix using the R-Matrix.

`#include <RMatrixFunc.h>`

Inheritance diagram for RMatrixFunc:



Public Member Functions

- RMatrixFunc (CNuc ∗, const Config &)
- CNuc ∗ compound () const
- const Config & configure () const
- void ClearMatrices ()
- void FillMatrices (EPoint ∗)
- void InvertMatrices ()
- void CalculateTMatrix (EPoint ∗)
- void CalculateCrossSection ()
- complex GetRMatrixElement (int, int, int) const
- complex GetRLMatrixElement (int, int, int) const
- complex GetRLInvMatrixElement (int, int, int) const
- complex GetRLInvRMatrixElement (int, int, int) const
- matrix_c ∗ GetJSpecRLMatrix (int)
- void AddRMatrixElement (int, int, int, complex)
- void AddRLMatrixElement (int, int, int, complex)
- void AddRLInvMatrix (matrix_c)
- void AddRLInvRMatrixElement (int, int, int, complex)

Public Member Functions inherited from GenMatrixFunc

- GenMatrixFunc ()
- virtual ∼GenMatrixFunc ()
- virtual void ClearMatrices ()=0
- virtual void FillMatrices (EPoint ∗)=0
- virtual void InvertMatrices ()=0
- virtual void CalculateTMatrix (EPoint ∗)=0
- void CalculateCrossSection (EPoint ∗)
- void NewTempTMatrix (TempTMatrix)
- void AddToTempTMatrix (int, complex)
- void ClearTempTMatrices ()
- void AddTMatrixElement (int, int, complex, int decayNum=1)
- void AddECTMatrixElement (int, int, complex)
- int IsTempTMatrix (double, int, int)
- int NumTempTMatrices () const
- TempTMatrix ∗ GetTempTMatrix (int)
- complex GetTMatrixElement (int, int, int decayNum=1) const
- complex GetECTMatrixElement (int, int) const
- virtual CNuc ∗ compound () const =0
- virtual const Config & configure () const =0

Additional Inherited Members

Protected Attributes inherited from GenMatrixFunc

- std::vector< matrix_c > tmatrix_

  Vector of internal T-matrix elements accessable to child class.
- matrix_c ec_tmatrix_

  Vector of external T-matrix elements accessable to child class.

7.78.1 Detailed Description

A function class to calculate the T-Matrix using the R-Matrix.

The RMatrixFunc function class calculates the T-Matrix for a given energy point using the compound nucleus object. The RMatrixFunc class is a child class of GenMatrixFunc, where the cross section is calculated from the T-Matrix.

Definition at line 14 of file RMatrixFunc.h.

7.78.2 Constructor & Destructor Documentation

7.78.2.1 RMatrixFunc()

```
RMatrixFunc::RMatrixFunc (
            CNuc * compound,
            const Config & configure )
```

The RMatrixFunc object is created with reference to a CNuc object.

Definition at line 11 of file RMatrixFunc.cpp.

7.78.3 Member Function Documentation

7.78.3.1 AddRLInvMatrix()

```
void RMatrixFunc::AddRLInvMatrix (
            matrix_c matrix )
```

This function adds an entire $[1-RL]^{-1}$ matrix to a vector.

Definition at line 315 of file RMatrixFunc.cpp.

7.78.3.2 AddRLInvRMatrixElement()

```
void RMatrixFunc::AddRLInvRMatrixElement (
            int jGroupNum,
            int channelNum,
            int channelPrimeNum,
            complex matrixElement )
```

This function adds a $[1-RL]^{-1}R$ matrix element specified by positions in the JGroup and AChannel vectors.

Definition at line 302 of file RMatrixFunc.cpp.

7.78.3.3 AddRLMatrixElement()

```
void RMatrixFunc::AddRLMatrixElement (
            int jGroupNum,
            int channelNum,
            int channelPrimeNum,
            complex matrixElement )
```

This function adds a $[1-RL]$ matrix element specified by positions in the JGroup and AChannel vectors.

Definition at line 289 of file RMatrixFunc.cpp.

7.78.3.4 AddRMatrixElement()

```
void RMatrixFunc::AddRMatrixElement (
            int jGroupNum,
            int channelNum,
            int channelPrimeNum,
            complex matrixElement )
```

This function adds an R-Matrix element specified by positions in the JGroup and AChannel vectors.

Definition at line 276 of file RMatrixFunc.cpp.

7.78.3.5 CalculateCrossSection()

```
void RMatrixFunc::CalculateCrossSection ( )
```

Instantiated in the parent class.

7.78.3.6 CalculateTMatrix()

```
void RMatrixFunc::CalculateTMatrix (
            EPoint * point ) [virtual]
```

This function calculates the T-Matrix for each reaction pathways based on the $[1-RL]^{-1}R$ matrix.

Implements GenMatrixFunc.

Definition at line 197 of file RMatrixFunc.cpp.

7.78.3.7 ClearMatrices()

```
void RMatrixFunc::ClearMatrices ( ) [virtual]
```

Clears all matrices associated with the RMatrixFunc object.

Implements GenMatrixFunc.

Definition at line 59 of file RMatrixFunc.cpp.

7.78.3.8 compound()

```
CNuc * RMatrixFunc::compound ( ) const [inline], [virtual]
```

Returns a pointer to the compound nucleus object.

Implements GenMatrixFunc.

Definition at line 20 of file RMatrixFunc.h.

7.78.3.9 configure()

```
const Config & RMatrixFunc::configure ( ) const [inline], [virtual]
```

This virtual function in implemented in the child class.

Implements GenMatrixFunc.

Definition at line 21 of file RMatrixFunc.h.

7.78.3.10 FillMatrices()

```
void RMatrixFunc::FillMatrices (
            EPoint * point ) [virtual]
```

This function creates the $[1-RL]$ and R Matrices from the CNuc object.

Implements GenMatrixFunc.

Definition at line 72 of file RMatrixFunc.cpp.

7.78.3.11 GetJSpecRLMatrix()

```
matrix_c * RMatrixFunc::GetJSpecRLMatrix (
            int jGroupNum )
```

Returns an entire $[1-RL]$ Matrix specified by a position in the JGroup vector.

Definition at line 50 of file RMatrixFunc.cpp.

7.78.3.12 GetRLInvMatrixElement()

```
complex RMatrixFunc::GetRLInvMatrixElement (
            int jGroupNum,
            int channelNum,
            int channelPrimeNum ) const
```

Returns a $[1-RL]^{-1}$ Matrix element specified by positions in the JGroup and AChannel vectors.

Definition at line 34 of file RMatrixFunc.cpp.

7.78.3.13 GetRLInvRMatrixElement()

```
complex RMatrixFunc::GetRLInvRMatrixElement (
            int jGroupNum,
            int channelNum,
            int channelPrimeNum ) const
```

Returns a $[1-RL]^{-1}R$ Matrix element specified by positions in the JGroup and AChannel vectors.

Definition at line 42 of file RMatrixFunc.cpp.

7.78.3.14 GetRLMatrixElement()

```
complex RMatrixFunc::GetRLMatrixElement (
            int jGroupNum,
            int channelNum,
            int channelPrimeNum ) const
```

Returns an $[1-RL]$ Matrix element specified by positions in the JGroup and AChannel vectors.

Definition at line 26 of file RMatrixFunc.cpp.

7.78.3.15 GetRMatrixElement()

```
complex RMatrixFunc::GetRMatrixElement (
            int jGroupNum,
            int channelNum,
            int channelPrimeNum ) const
```

Returns an R-Matrix element specified by positions in the JGroup and AChannel vectors.

Definition at line 18 of file RMatrixFunc.cpp.

7.78.3.16 InvertMatrices()

```
void RMatrixFunc::InvertMatrices ( )  [virtual]
```

This function inverts the $[1-RL]$ matrices and creates the $[1-RL]^{-1}R$ matrices.

Implements GenMatrixFunc.

Definition at line 173 of file RMatrixFunc.cpp.

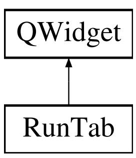
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/RMatrixFunc.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/RMatrixFunc.cpp

7.79 RunTab Class Reference

```
#include <RunTab.h>
```

Inheritance diagram for RunTab:



Public Slots

- void showInfo (int which=0, QString title="")

Public Member Functions

- RunTab (QWidget *parent=0)
- void reset ()

Friends

- class AZURESetup
- class AZUREMainThread

7.79.1 Detailed Description

Definition at line 23 of file RunTab.h.

7.79.2 Constructor & Destructor Documentation

7.79.2.1 RunTab()

```
RunTab::RunTab (
            QWidget * parent = 0 )
```

Definition at line 15 of file RunTab.cpp.

7.79.3 Member Function Documentation

7.79.3.1 reset()

```
void RunTab::reset ( )
```

Definition at line 225 of file RunTab.cpp.

7.79.3.2 showInfo

```
void RunTab::showInfo (
            int which = 0,
            QString title = "" )  [slot]
```

Definition at line 241 of file RunTab.cpp.

7.79.4 Friends And Related Symbol Documentation

7.79.4.1 AZUREMainThread

```
friend class AZUREMainThread [friend]
```

Definition at line 29 of file RunTab.h.

7.79.4.2 AZURESetup

```
friend class AZURESetup [friend]
```

Definition at line 28 of file RunTab.h.

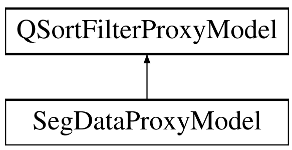
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RunTab.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InTabDocs.cpp
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RunTab.cpp

7.80 SegDataProxyModel Class Reference

```
#include <PlotTab.h>
```

Inheritance diagram for SegDataProxyModel:



Public Member Functions

- SegDataProxyModel (QWidget *parent=0)
- QVariant data (const QModelIndex &index, int role=Qt::DisplayRole) const

7.80.1 Detailed Description

Definition at line 33 of file PlotTab.h.

7.80.2 Constructor & Destructor Documentation

7.80.2.1 SegDataProxyModel()

```
SegDataProxyModel::SegDataProxyModel (
            QWidget * parent = 0 ) [inline]
```

Definition at line 35 of file PlotTab.h.

7.80.3 Member Function Documentation

7.80.3.1 data()

```
QVariant SegDataProxyModel::data (
            const QModelIndex & index,
            int role = Qt::DisplayRole ) const
```

Definition at line 40 of file PlotTab.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PlotTab.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PlotTab.cpp

7.81 SegLine Class Reference

A class to read and store a line from the data segments input file.

```
#include <SegLine.h>
```

Public Member Functions

- SegLine (std::istream &stream)
- int isActive () const
- int entranceKey () const
- int exitKey () const
- double minE () const
- double maxE () const
- double minA () const
- double maxA () const
- int isDiff () const
- std::string dataFile () const
- double dataNorm () const
- double dataNormError () const
- int varyNorm () const
- double phaseJ () const
- int phaseL () const

7.81.1 Detailed Description

A class to read and store a line from the data segments input file.

The SegLine class reads and stores a line from the data segments input file.

Definition at line 13 of file SegLine.h.

7.81.2 Constructor & Destructor Documentation

7.81.2.1 SegLine()

```
SegLine::SegLine (
            std::istream & stream )  [inline]
```

Constructor fill the SegLine object from an input stream.

Definition at line 18 of file SegLine.h.

7.81.3 Member Function Documentation

7.81.3.1 dataFile()

```
std::string SegLine::dataFile ( ) const  [inline]
```

Returns the path of the data file for the segment.

Definition at line 69 of file SegLine.h.

7.81.3.2 dataNorm()

```
double SegLine::dataNorm ( ) const  [inline]
```

Returns the data normalization for the segment.

Definition at line 73 of file SegLine.h.

7.81.3.3 dataNormError()

```
double SegLine::dataNormError ( ) const  [inline]
```

Returns the data normalization error for the segment.

Definition at line 77 of file SegLine.h.

7.81.3.4 entranceKey()

```
int SegLine::entranceKey ( ) const  [inline]
```

Returns the particle pair key corresponding to the entrance channel for the data segment.

Definition at line 39 of file SegLine.h.

7.81.3.5 exitKey()

```
int SegLine::exitKey ( ) const  [inline]
```

Returns the particle pair key corresponding to the exit channel for the data segment.

Definition at line 44 of file SegLine.h.

7.81.3.6 isActive()

```
int SegLine::isActive ( ) const  [inline]
```

Returns non-zero if the line is to be included in the calculation.

Definition at line 34 of file SegLine.h.

7.81.3.7 isDiff()

```
int SegLine::isDiff ( ) const  [inline]
```

Return 0 if the segment is angle-integrated cross section, 1 for differential cross section, and 2 for phase shift.

Definition at line 65 of file SegLine.h.

7.81.3.8 maxA()

```
double SegLine::maxA ( ) const  [inline]
```

Returns the maximum angle to be included in the segment from the data.

Definition at line 60 of file SegLine.h.

7.81.3.9 maxE()

```
double SegLine::maxE ( ) const  [inline]
```

Returns the maximum energy to be included in the segment from the data.

Definition at line 52 of file SegLine.h.

7.81.3.10 minA()

```
double SegLine::minA ( ) const  [inline]
```

Returns the minimum angle to be included in the segment from the data.

Definition at line 56 of file SegLine.h.

7.81.3.11 minE()

```
double SegLine::minE ( ) const  [inline]
```

Returns the minimum energy to be included in the segment from the data.

Definition at line 48 of file SegLine.h.

7.81.3.12 phaseJ()

```
double SegLine::phaseJ ( ) const  [inline]
```

Returns the spin value for the segment if the segment contains phase shift.

Definition at line 85 of file SegLine.h.

7.81.3.13 phaseL()

```
int SegLine::phaseL ( ) const  [inline]
```

Returns the orbital angular momentum value for the segment if the segment contains phase shift.

Definition at line 90 of file SegLine.h.

7.81.3.14 varyNorm()

```
int SegLine::varyNorm ( ) const  [inline]
```

Returns non-zero of the normalization is to be fit.

Definition at line 81 of file SegLine.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/SegLine.h

7.82 SegmentsDataData Struct Reference

```
#include <SegmentsDataModel.h>
```

Public Attributes

- int isActive
- int entrancePairIndex
- int exitPairIndex
- double lowEnergy
- double highEnergy
- double lowAngle
- double highAngle
- int dataType
- QString dataFile
- double dataNorm
- double dataNormError
- int varyNorm
- double phaseJ
- int phaseL

Static Public Attributes

- static const int SIZE = 14

7.82.1 Detailed Description

Definition at line 9 of file SegmentsDataModel.h.

7.82.2 Member Data Documentation

7.82.2.1 dataFile

```
QString SegmentsDataData::dataFile
```

Definition at line 19 of file SegmentsDataModel.h.

7.82.2.2 dataNorm

```
double SegmentsDataData::dataNorm
```

Definition at line 20 of file SegmentsDataModel.h.

7.82.2.3 dataNormError

```
double SegmentsDataData::dataNormError
```

Definition at line 21 of file SegmentsDataModel.h.

7.82.2.4 dataType

`int SegmentsDataData::dataType`

Definition at line 18 of file SegmentsDataModel.h.

7.82.2.5 entrancePairIndex

`int SegmentsDataData::entrancePairIndex`

Definition at line 12 of file SegmentsDataModel.h.

7.82.2.6 exitPairIndex

`int SegmentsDataData::exitPairIndex`

Definition at line 13 of file SegmentsDataModel.h.

7.82.2.7 highAngle

`double SegmentsDataData::highAngle`

Definition at line 17 of file SegmentsDataModel.h.

7.82.2.8 highEnergy

`double SegmentsDataData::highEnergy`

Definition at line 15 of file SegmentsDataModel.h.

7.82.2.9 isActive

`int SegmentsDataData::isActive`

Definition at line 11 of file SegmentsDataModel.h.

7.82.2.10 lowAngle

`double SegmentsDataData::lowAngle`

Definition at line 16 of file SegmentsDataModel.h.

7.82.2.11 lowEnergy

`double SegmentsDataData::lowEnergy`

Definition at line 14 of file SegmentsDataModel.h.

7.82.2.12 phaseJ

```
double SegmentsDataData::phaseJ
```

Definition at line 23 of file SegmentsDataModel.h.

7.82.2.13 phaseL

```
int SegmentsDataData::phaseL
```

Definition at line 24 of file SegmentsDataModel.h.

7.82.2.14 SIZE

```
const int SegmentsDataData::SIZE = 14 [static]
```

Definition at line 10 of file SegmentsDataModel.h.

7.82.2.15 varyNorm

```
int SegmentsDataData::varyNorm
```

Definition at line 22 of file SegmentsDataModel.h.

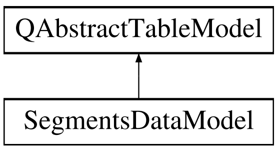
The documentation for this struct was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsDataModel.h

7.83 SegmentsDataModel Class Reference

```
#include <SegmentsDataModel.h>
```

Inheritance diagram for SegmentsDataModel:



Public Member Functions

- SegmentsDataModel (QObject *parent=0)
- int rowCount (const QModelIndex &parent) const
- int columnCount (const QModelIndex &parent) const
- QVariant data (const QModelIndex &index, int role) const
- QVariant headerData (int section, Qt::Orientation orientation, int role) const
- bool setData (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- bool insertRows (int position, int rows, const QModelIndex &index=QModelIndex())
- bool removeRows (int position, int rows, const QModelIndex &index=QModelIndex())
- Qt::ItemFlags flags (const QModelIndex &index) const
- int isSegDataLine (const SegmentsDataData &line) const
- QList< SegmentsDataData > getLines () const
- void setPairsModel (PairsModel *model)
- QString getReactionLabel (const QModelIndex &index)

7.83.1 Detailed Description

Definition at line 27 of file SegmentsDataModel.h.

7.83.2 Constructor & Destructor Documentation

7.83.2.1 SegmentsDataModel()

```
SegmentsDataModel::SegmentsDataModel (
            QObject * parent = 0 )
```

Definition at line 5 of file SegmentsDataModel.cpp.

7.83.3 Member Function Documentation

7.83.3.1 columnCount()

```
int SegmentsDataModel::columnCount (
            const QModelIndex & parent ) const
```

Definition at line 13 of file SegmentsDataModel.cpp.

7.83.3.2 data()

```
QVariant SegmentsDataModel::data (
            const QModelIndex & index,
            int role ) const
```

Definition at line 18 of file SegmentsDataModel.cpp.

7.83.3.3 flags()

```
Qt::ItemFlags SegmentsDataModel::flags (
            const QModelIndex & index ) const
```

Definition at line 226 of file SegmentsDataModel.cpp.

7.83.3.4 getLines()

```
QList< SegmentsDataData > SegmentsDataModel::getLines ( ) const  [inline]
```

Definition at line 42 of file SegmentsDataModel.h.

7.83.3.5 getReactionLabel()

```
QString SegmentsDataModel::getReactionLabel (
            const QModelIndex & index )
```

Definition at line 260 of file SegmentsDataModel.cpp.

7.83.3.6 headerData()

```
QVariant SegmentsDataModel::headerData (
            int section,
            Qt::Orientation orientation,
            int role ) const
```

Definition at line 123 of file SegmentsDataModel.cpp.

7.83.3.7 insertRows()

```
bool SegmentsDataModel::insertRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 201 of file SegmentsDataModel.cpp.

7.83.3.8 isSegDataLine()

```
int SegmentsDataModel::isSegDataLine (
            const SegmentsDataData & line ) const
```

Definition at line 232 of file SegmentsDataModel.cpp.

7.83.3.9 removeRows()

```
bool SegmentsDataModel::removeRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 214 of file SegmentsDataModel.cpp.

7.83.3.10 rowCount()

```
int SegmentsDataModel::rowCount (
            const QModelIndex & parent ) const
```

Definition at line 8 of file SegmentsDataModel.cpp.

7.83.3.11 setData()

```
bool SegmentsDataModel::setData (
            const QModelIndex & index,
            const QVariant & value,
            int role = Qt::EditRole )
```

Definition at line 164 of file SegmentsDataModel.cpp.

7.83.3.12 setPairsModel()

```
void SegmentsDataModel::setPairsModel (
            PairsModel * model )
```

Definition at line 256 of file SegmentsDataModel.cpp.

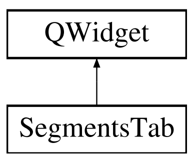
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsDataModel.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsDataModel.cpp

7.84 SegmentsTab Class Reference

```
#include <SegmentsTab.h>
```

Inheritance diagram for SegmentsTab:



Public Slots

- void addSegDataLine ()
- void addSegDataLine (SegmentsDataData line)
- void addSegTestLine ()
- void addSegTestLine (SegmentsTestData line)
- void editSegDataLine ()
- void editSegTestLine ()
- void deleteSegDataLine ()
- void deleteSegTestLine ()
- void moveSegDataLineUp ()
- void moveSegDataLineDown ()
- void moveSegTestLineUp ()
- void moveSegTestLineDown ()
- void updateSegDataButtons (const QItemSelection &selection)
- void updateSegTestButtons (const QItemSelection &selection)
- bool readSegDataFile (QTextStream &inStream)
- bool writeSegDataFile (QTextStream &outStream)
- bool readSegTestFile (QTextStream &inStream)
- bool writeSegTestFile (QTextStream &outStream)
- void setPairsModel (PairsModel *model)
- void showInfo (int which=0, QString title="")

Public Member Functions

- SegmentsTab (QWidget *parent=0)
- SegmentsTestModel * getSegmentsTestModel ()
- SegmentsDataModel * getSegmentsDataModel ()
- void reset ()

7.84.1 Detailed Description

Definition at line 18 of file SegmentsTab.h.

7.84.2 Constructor & Destructor Documentation

7.84.2.1 SegmentsTab()

```
SegmentsTab::SegmentsTab (
            QWidget * parent = 0 )
```

Definition at line 12 of file SegmentsTab.cpp.

7.84.3 Member Function Documentation

7.84.3.1 addSegDataLine [1/2]

```
void SegmentsTab::addSegDataLine ( )  [slot]
```

Definition at line 188 of file SegmentsTab.cpp.

7.84.3.2 addSegDataLine [2/2]

```
void SegmentsTab::addSegDataLine (
            SegmentsDataData line )  [slot]
```

Definition at line 213 of file SegmentsTab.cpp.

7.84.3.3 addSegTestLine [1/2]

```
void SegmentsTab::addSegTestLine ( )  [slot]
```

Definition at line 252 of file SegmentsTab.cpp.

7.84.3.4 addSegTestLine [2/2]

```
void SegmentsTab::addSegTestLine (
            SegmentsTestData line )  [slot]
```

Definition at line 275 of file SegmentsTab.cpp.

7.84.3.5 deleteSegDataLine

```
void SegmentsTab::deleteSegDataLine ( )  [slot]
```

Definition at line 170 of file SegmentsTab.cpp.

7.84.3.6 deleteSegTestLine

```
void SegmentsTab::deleteSegTestLine ( )  [slot]
```

Definition at line 179 of file SegmentsTab.cpp.

7.84.3.7 editSegDataLine

```
void SegmentsTab::editSegDataLine ( )  [slot]
```

Definition at line 312 of file SegmentsTab.cpp.

7.84.3.8 editSegTestLine

```
void SegmentsTab::editSegTestLine ( )  [slot]
```

Definition at line 446 of file SegmentsTab.cpp.

7.84.3.9 getSegmentsDataModel()

```
SegmentsDataModel * SegmentsTab::getSegmentsDataModel ( )
```

Definition at line 166 of file SegmentsTab.cpp.

7.84.3.10 getSegmentsTestModel()

```
SegmentsTestModel * SegmentsTab::getSegmentsTestModel ( )
```

Definition at line 162 of file SegmentsTab.cpp.

7.84.3.11 moveSegDataLineDown

```
void SegmentsTab::moveSegDataLineDown ( ) [slot]
```

Definition at line 573 of file SegmentsTab.cpp.

7.84.3.12 moveSegDataLineUp

```
void SegmentsTab::moveSegDataLineUp ( ) [slot]
```

Definition at line 569 of file SegmentsTab.cpp.

7.84.3.13 moveSegTestLineDown

```
void SegmentsTab::moveSegTestLineDown ( ) [slot]
```

Definition at line 627 of file SegmentsTab.cpp.

7.84.3.14 moveSegTestLineUp

```
void SegmentsTab::moveSegTestLineUp ( ) [slot]
```

Definition at line 623 of file SegmentsTab.cpp.

7.84.3.15 readSegDataFile

```
bool SegmentsTab::readSegDataFile (
             QTextStream & inStream ) [slot]
```

Definition at line 709 of file SegmentsTab.cpp.

7.84.3.16 readSegTestFile

```
bool SegmentsTab::readSegTestFile (
            QTextStream & inStream )   [slot]
```

Definition at line 781 of file SegmentsTab.cpp.

7.84.3.17 reset()

```
void SegmentsTab::reset ( )
```

Definition at line 856 of file SegmentsTab.cpp.

7.84.3.18 setPairsModel

```
void SegmentsTab::setPairsModel (
            PairsModel * model )   [inline], [slot]
```

Definition at line 60 of file SegmentsTab.h.

7.84.3.19 showInfo

```
void SegmentsTab::showInfo (
            int which = 0,
            QString title = "" )   [slot]
```

Definition at line 861 of file SegmentsTab.cpp.

7.84.3.20 updateSegDataButtons

```
void SegmentsTab::updateSegDataButtons (
            const QItemSelection & selection )   [slot]
```

Definition at line 676 of file SegmentsTab.cpp.

7.84.3.21 updateSegTestButtons

```
void SegmentsTab::updateSegTestButtons (
            const QItemSelection & selection )   [slot]
```

Definition at line 692 of file SegmentsTab.cpp.

7.84.3.22 writeSegDataFile

```
bool SegmentsTab::writeSegDataFile (
            QTextStream & outStream )   [slot]
```

Definition at line 757 of file SegmentsTab.cpp.

7.84.3.23 writeSegTestFile

```
bool SegmentsTab::writeSegTestFile (
            QTextStream & outStream )  [slot]
```

Definition at line 826 of file SegmentsTab.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTab.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InTabDocs.cpp
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTab.cpp

7.85 SegmentsTestData Struct Reference

```
#include <SegmentsTestModel.h>
```

Public Attributes

- int isActive
- int entrancePairIndex
- int exitPairIndex
- double lowEnergy
- double highEnergy
- double energyStep
- double lowAngle
- double highAngle
- double angleStep
- int dataType
- double phaseJ
- int phaseL
- int maxAngDistOrder

Static Public Attributes

- static const int SIZE = 13

7.85.1 Detailed Description

Definition at line 9 of file SegmentsTestModel.h.

7.85.2 Member Data Documentation

7.85.2.1 angleStep

```
double SegmentsTestData::angleStep
```

Definition at line 19 of file SegmentsTestModel.h.

7.85.2.2 dataType

```
int SegmentsTestData::dataType
```

Definition at line 20 of file SegmentsTestModel.h.

7.85.2.3 energyStep

```
double SegmentsTestData::energyStep
```

Definition at line 16 of file SegmentsTestModel.h.

7.85.2.4 entrancePairIndex

```
int SegmentsTestData::entrancePairIndex
```

Definition at line 12 of file SegmentsTestModel.h.

7.85.2.5 exitPairIndex

```
int SegmentsTestData::exitPairIndex
```

Definition at line 13 of file SegmentsTestModel.h.

7.85.2.6 highAngle

```
double SegmentsTestData::highAngle
```

Definition at line 18 of file SegmentsTestModel.h.

7.85.2.7 highEnergy

```
double SegmentsTestData::highEnergy
```

Definition at line 15 of file SegmentsTestModel.h.

7.85.2.8 isActive

```
int SegmentsTestData::isActive
```

Definition at line 11 of file SegmentsTestModel.h.

7.85.2.9 lowAngle

```
double SegmentsTestData::lowAngle
```

Definition at line 17 of file SegmentsTestModel.h.

7.85.2.10 lowEnergy

```
double SegmentsTestData::lowEnergy
```

Definition at line 14 of file SegmentsTestModel.h.

7.85.2.11 maxAngDistOrder

```
int SegmentsTestData::maxAngDistOrder
```

Definition at line 23 of file SegmentsTestModel.h.

7.85.2.12 phaseJ

```
double SegmentsTestData::phaseJ
```

Definition at line 21 of file SegmentsTestModel.h.

7.85.2.13 phaseL

```
int SegmentsTestData::phaseL
```

Definition at line 22 of file SegmentsTestModel.h.

7.85.2.14 SIZE

```
const int SegmentsTestData::SIZE = 13 [static]
```

Definition at line 10 of file SegmentsTestModel.h.

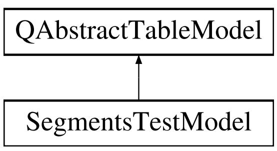
The documentation for this struct was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTestModel.h

7.86 SegmentsTestModel Class Reference

```
#include <SegmentsTestModel.h>
```

Inheritance diagram for SegmentsTestModel:



Public Member Functions

- SegmentsTestModel (QObject ∗parent=0)
- int rowCount (const QModelIndex &parent) const
- int columnCount (const QModelIndex &parent) const
- QVariant data (const QModelIndex &index, int role) const
- QVariant headerData (int section, Qt::Orientation orientation, int role) const
- bool setData (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- bool insertRows (int position, int rows, const QModelIndex &index=QModelIndex())
- bool removeRows (int position, int rows, const QModelIndex &index=QModelIndex())
- Qt::ItemFlags flags (const QModelIndex &index) const
- int isSegTestLine (const SegmentsTestData &line) const
- QList< SegmentsTestData > getLines () const
- void setPairsModel (PairsModel ∗model)
- QString getReactionLabel (const QModelIndex &index)

7.86.1 Detailed Description

Definition at line 26 of file SegmentsTestModel.h.

7.86.2 Constructor & Destructor Documentation

7.86.2.1 SegmentsTestModel()

```
SegmentsTestModel::SegmentsTestModel (
            QObject * parent = 0 )
```

Definition at line 4 of file SegmentsTestModel.cpp.

7.86.3 Member Function Documentation

7.86.3.1 columnCount()

```
int SegmentsTestModel::columnCount (
            const QModelIndex & parent ) const
```

Definition at line 12 of file SegmentsTestModel.cpp.

7.86.3.2 data()

```
QVariant SegmentsTestModel::data (
            const QModelIndex & index,
            int role ) const
```

Definition at line 17 of file SegmentsTestModel.cpp.

7.86.3.3 flags()

```
Qt::ItemFlags SegmentsTestModel::flags (
            const QModelIndex & index ) const
```

Definition at line 215 of file SegmentsTestModel.cpp.

7.86.3.4 getLines()

```
QList< SegmentsTestData > SegmentsTestModel::getLines ( ) const [inline]
```

Definition at line 41 of file SegmentsTestModel.h.

7.86.3.5 getReactionLabel()

```
QString SegmentsTestModel::getReactionLabel (
            const QModelIndex & index )
```

Definition at line 248 of file SegmentsTestModel.cpp.

7.86.3.6 headerData()

```
QVariant SegmentsTestModel::headerData (
            int section,
            Qt::Orientation orientation,
            int role ) const
```

Definition at line 115 of file SegmentsTestModel.cpp.

7.86.3.7 insertRows()

```
bool SegmentsTestModel::insertRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 190 of file SegmentsTestModel.cpp.

7.86.3.8 isSegTestLine()

```
int SegmentsTestModel::isSegTestLine (
            const SegmentsTestData & line ) const
```

Definition at line 221 of file SegmentsTestModel.cpp.

7.86.3.9 removeRows()

```
bool SegmentsTestModel::removeRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 203 of file SegmentsTestModel.cpp.

7.86.3.10 rowCount()

```
int SegmentsTestModel::rowCount (
            const QModelIndex & parent ) const
```

Definition at line 7 of file SegmentsTestModel.cpp.

7.86.3.11 setData()

```
bool SegmentsTestModel::setData (
            const QModelIndex & index,
            const QVariant & value,
            int role = Qt::EditRole )
```

Definition at line 154 of file SegmentsTestModel.cpp.

7.86.3.12 setPairsModel()

```
void SegmentsTestModel::setPairsModel (
            PairsModel * model )
```

Definition at line 244 of file SegmentsTestModel.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTestModel.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTestModel.cpp

7.87 SegPairs Struct Reference

Public Attributes

- int firstPair
- int secondPair

7.87.1 Detailed Description

Definition at line 28 of file AZURESetup.cpp.

7.87.2 Member Data Documentation

7.87.2.1 firstPair

```
int SegPairs::firstPair
```

Definition at line 28 of file AZURESetup.cpp.

7.87.2.2 secondPair

```
int SegPairs::secondPair
```

Definition at line 28 of file AZURESetup.cpp.

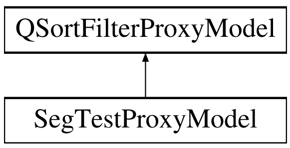
The documentation for this struct was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZURESetup.cpp
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZURE2.cpp

7.88 SegTestProxyModel Class Reference

```
#include <PlotTab.h>
```

Inheritance diagram for SegTestProxyModel:



Public Member Functions

- SegTestProxyModel (QWidget ∗parent=0)
- QVariant data (const QModelIndex &index, int role=Qt::DisplayRole) const
- bool filterAcceptsRow (int source_row, const QModelIndex &source_parent) const

7.88.1 Detailed Description

Definition at line 26 of file PlotTab.h.

7.88.2 Constructor & Destructor Documentation

7.88.2.1 SegTestProxyModel()

```
SegTestProxyModel::SegTestProxyModel (
            QWidget * parent = 0 )  [inline]
```

Definition at line 28 of file PlotTab.h.

7.88.3 Member Function Documentation

7.88.3.1 data()

```
QVariant SegTestProxyModel::data (
            const QModelIndex & index,
            int role = Qt::DisplayRole ) const
```

Definition at line 20 of file PlotTab.cpp.

7.88.3.2 filterAcceptsRow()

```
bool SegTestProxyModel::filterAcceptsRow (
            int source_row,
            const QModelIndex & source_parent ) const
```

Definition at line 29 of file PlotTab.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PlotTab.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PlotTab.cpp

7.89 ShftFunc Class Reference

A function class for negative energy shift functions.

```
#include <ShftFunc.h>
```

Public Member Functions

- ShftFunc (PPair ∗pPair)
- ∼ShftFunc ()
- double operator() (int l, double energy)
- double EnergyDerivative (int l, double energy)

7.89.1 Detailed Description

A function class for negative energy shift functions.

A shift function for negative energy channels is calculated as $S = \rho \frac{W_c{}'(k\rho)}{W_c(k\rho)}$, where the prime indicates the derivative with respect to ρ. The AZURE function class ShftFunc uses the GSL package to calculates the numerical derivative.

Definition at line 18 of file ShftFunc.h.

7.89.2 Constructor & Destructor Documentation

7.89.2.1 ShftFunc()

```
ShftFunc::ShftFunc (
            PPair * pPair )  [inline]
```

The ShftFunc object is created with reference to a particle pair.

Definition at line 23 of file ShftFunc.h.

7.89.2.2 ∼ShftFunc()

```
ShftFunc::∼ShftFunc ( )  [inline]
```

Definition at line 31 of file ShftFunc.h.

7.89.3 Member Function Documentation

7.89.3.1 EnergyDerivative()

```
double ShftFunc::EnergyDerivative (
            int l,
            double energy )
```

Returns the energy derivative of the shift function at the specified orbital angular momentum and energy in the compound system.

Definition at line 37 of file ShftFunc.cpp.

7.89.3.2 operator()()

```
double ShftFunc::operator() (
            int l,
            double energy )
```

The parenthesis operator is defined to make the class instance callable as a function. The orbital angular momentum and energy in the compound system are the dependent variables. The function returns the value of the shift function.

Definition at line 21 of file ShftFunc.cpp.

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

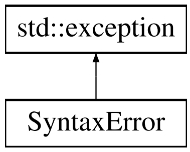
- /Users/kuba/Desktop/R-Matrix/AZURE2/include/ShftFunc.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/ShftFunc.cpp

7.90 SyntaxError Class Reference

An exception class thrown by the Equation class.

```
#include <Equation.h>
```

Inheritance diagram for SyntaxError:



Public Member Functions

- SyntaxError (std::string equation, int type, int position=-1)
- ~SyntaxError () throw ()
- virtual const char * what () const throw ()

7.90.1 Detailed Description

An exception class thrown by the Equation class.

The SyntaxError class is an exception class thrown by the Equation class. It should not be used directly.

Definition at line 89 of file Equation.h.

7.90.2 Constructor & Destructor Documentation

7.90.2.1 SyntaxError()

```
SyntaxError::SyntaxError (
            std::string equation,
            int type,
            int position = -1 )  [inline]
```

Constructs the SyntaxError with the message type, equation string, and position in the string.

Definition at line 95 of file Equation.h.

7.90.2.2 ~SyntaxError()

```
SyntaxError::~SyntaxError ( ) throw ( )  [inline]
```

Definition at line 105 of file Equation.h.

7.90.3 Member Function Documentation

7.90.3.1 what()

```
virtual const char * SyntaxError::what ( ) const throw ( )   [inline], [virtual]
```

Returns the message of the thrown SyntaxError.

Definition at line 111 of file Equation.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/Equation.h

7.91 TargetEffect Class Reference

An AZURE target effect entry.

```
#include <TargetEffect.h>
```

Public Member Functions

- TargetEffect (std::istream &, const Config &)
- bool IsActive () const
- bool IsConvolution () const
- bool IsTargetIntegration () const
- bool IsQCoefficients () const
- int NumSubPoints () const
- int NumQCoefficients () const
- double GetSigma () const
- double GetDensity () const
- double TargetThickness (double, const Config &)
- double GetConvolutionFactor (double, double) const
- double GetQCoefficient (int) const
- void SetSigma (double)
- void SetNumSubPoints (int)
- std::vector< int > GetSegmentsList () const
- Equation * GetStoppingPowerEq ()

Static Public Attributes

- static constexpr double convolutionRange =3.

  The multiple of sigma above and below centroid energy to use as integration range.

7.91.1 Detailed Description

An AZURE target effect entry.

Experimential effects including gaussian beam convolution, target integration, and a combination of the two are grouped under the TargetEffect class. An object is created corresponding to each corresponding entry in AZURESetup2.

Definition at line 19 of file TargetEffect.h.

7.91.2 Constructor & Destructor Documentation

7.91.2.1 TargetEffect()

```
TargetEffect::TargetEffect (
            std::istream & stream,
            const Config & configure )
```

Constructor reads directly from an std::ifstream pointing to the target effect input file. If a valid target effect is read, a TargetEffect object is created.

Definition at line 11 of file TargetEffect.cpp.

7.91.3 Member Function Documentation

7.91.3.1 GetConvolutionFactor()

```
double TargetEffect::GetConvolutionFactor (
            double energy,
            double centroid ) const
```

Returns the multiplicative convolution factor for evaluation of the integrand of a target effect.

Definition at line 225 of file TargetEffect.cpp.

7.91.3.2 GetDensity()

```
double TargetEffect::GetDensity ( ) const
```

Returns the density of the target in atoms/cm^2. Only needed for target integration, not Gaussian beam convolution.

Definition at line 138 of file TargetEffect.cpp.

7.91.3.3 GetQCoefficient()

```
double TargetEffect::GetQCoefficient (
            int order ) const
```

Returns the attenuation coefficients for the given order specified in by the target effect.

Definition at line 155 of file TargetEffect.cpp.

7.91.3.4 GetSegmentsList()

```
std::vector< int > TargetEffect::GetSegmentsList ( ) const
```

Parses and returns a vector of integers corresponding to the segment list specified as a string. The segments list contains the segments for which the target effect is applicable.

Definition at line 181 of file TargetEffect.cpp.

7.91.3.5 GetSigma()

```
double TargetEffect::GetSigma ( ) const
```

Returns the sigma of the Guassian for beam convolution.

Definition at line 129 of file TargetEffect.cpp.

7.91.3.6 GetStoppingPowerEq()

```
Equation * TargetEffect::GetStoppingPowerEq ( )
```

Returns the Equation object corresponding to the parametrized stopping cross section.

Definition at line 214 of file TargetEffect.cpp.

7.91.3.7 IsActive()

```
bool TargetEffect::IsActive ( ) const
```

Returns true if the target effect was marked as active in the target effects input file, otherwise returns false.

Definition at line 76 of file TargetEffect.cpp.

7.91.3.8 IsConvolution()

```
bool TargetEffect::IsConvolution ( ) const
```

Returns true if the target effect contains Gaussian beam convolution, otherwise returns false.

Definition at line 85 of file TargetEffect.cpp.

7.91.3.9 IsQCoefficients()

```
bool TargetEffect::IsQCoefficients ( ) const
```

Returns true if the target effect contains attenuation coefficients, otherwise returns false.

Definition at line 103 of file TargetEffect.cpp.

7.91.3.10 IsTargetIntegration()

```
bool TargetEffect::IsTargetIntegration ( ) const
```

Returns true if the target effect contains target integration, otherwise returns false.

Definition at line 94 of file TargetEffect.cpp.

7.91.3.11 NumQCoefficients()

```
int TargetEffect::NumQCoefficients ( ) const
```

Returns the number of attenuation coefficients for the target effect in the input file.

Definition at line 121 of file TargetEffect.cpp.

7.91.3.12 NumSubPoints()

```
int TargetEffect::NumSubPoints ( ) const
```

Returns the number of sub-points specified for the target effect in the input file.

Definition at line 112 of file TargetEffect.cpp.

7.91.3.13 SetNumSubPoints()

```
void TargetEffect::SetNumSubPoints (
             int numPoints )
```

Sets the number of sub-points for the TargetEffect object.

Definition at line 171 of file TargetEffect.cpp.

7.91.3.14 SetSigma()

```
void TargetEffect::SetSigma (
             double sigma )
```

Sets the convolution sigma to a new value.

Definition at line 163 of file TargetEffect.cpp.

7.91.3.15 TargetThickness()

```
double TargetEffect::TargetThickness (
             double energy,
             const Config & configure )
```

Calculates the Target thickness from the stopping cross section and the target density as a function of energy.

Definition at line 147 of file TargetEffect.cpp.

7.91.4 Member Data Documentation

7.91.4.1 convolutionRange

```
constexpr double TargetEffect::convolutionRange =3.  [static], [constexpr]
```

The multiple of sigma above and below centroid energy to use as integration range.

Definition at line 38 of file TargetEffect.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/TargetEffect.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/TargetEffect.cpp

7.92 TargetIntData Struct Reference

```
#include <TargetIntModel.h>
```

Public Attributes

- int isActive
- QString segmentsList
- int numPoints
- bool isConvolution
- double sigma
- bool isTargetIntegration
- double density
- QString stoppingPowerEq
- int numParameters
- QList< double > parameters
- bool isQCoefficients
- QList< double > qCoefficients

Static Public Attributes

- static const int SIZE = 12

7.92.1 Detailed Description

Definition at line 9 of file TargetIntModel.h.

7.92.2 Member Data Documentation

7.92.2.1 density

```
double TargetIntData::density
```

Definition at line 17 of file TargetIntModel.h.

7.92.2.2 isActive

`int TargetIntData::isActive`

Definition at line 11 of file TargetIntModel.h.

7.92.2.3 isConvolution

`bool TargetIntData::isConvolution`

Definition at line 14 of file TargetIntModel.h.

7.92.2.4 isQCoefficients

`bool TargetIntData::isQCoefficients`

Definition at line 21 of file TargetIntModel.h.

7.92.2.5 isTargetIntegration

`bool TargetIntData::isTargetIntegration`

Definition at line 16 of file TargetIntModel.h.

7.92.2.6 numParameters

`int TargetIntData::numParameters`

Definition at line 19 of file TargetIntModel.h.

7.92.2.7 numPoints

`int TargetIntData::numPoints`

Definition at line 13 of file TargetIntModel.h.

7.92.2.8 parameters

`QList<double> TargetIntData::parameters`

Definition at line 20 of file TargetIntModel.h.

7.92.2.9 qCoefficients

`QList<double> TargetIntData::qCoefficients`

Definition at line 22 of file TargetIntModel.h.

7.92.2.10 segmentsList

```
QString TargetIntData::segmentsList
```

Definition at line 12 of file TargetIntModel.h.

7.92.2.11 sigma

```
double TargetIntData::sigma
```

Definition at line 15 of file TargetIntModel.h.

7.92.2.12 SIZE

```
const int TargetIntData::SIZE = 12 [static]
```

Definition at line 10 of file TargetIntModel.h.

7.92.2.13 stoppingPowerEq

```
QString TargetIntData::stoppingPowerEq
```

Definition at line 18 of file TargetIntModel.h.

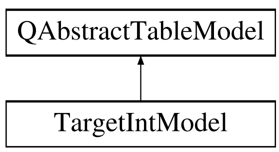
The documentation for this struct was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntModel.h

7.93 TargetIntModel Class Reference

```
#include <TargetIntModel.h>
```

Inheritance diagram for TargetIntModel:



Public Member Functions

- TargetIntModel (QObject ∗parent=0)
- int rowCount (const QModelIndex &parent) const
- int columnCount (const QModelIndex &parent) const
- QVariant data (const QModelIndex &index, int role) const
- QVariant headerData (int section, Qt::Orientation orientation, int role) const
- bool setData (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- bool insertRows (int position, int rows, const QModelIndex &index=QModelIndex())
- bool removeRows (int position, int rows, const QModelIndex &index=QModelIndex())
- Qt::ItemFlags flags (const QModelIndex &index) const
- QList< TargetIntData > getLines () const

7.93.1 Detailed Description

Definition at line 25 of file TargetIntModel.h.

7.93.2 Constructor & Destructor Documentation

7.93.2.1 TargetIntModel()

```
TargetIntModel::TargetIntModel (
            QObject * parent = 0 )
```

Definition at line 5 of file TargetIntModel.cpp.

7.93.3 Member Function Documentation

7.93.3.1 columnCount()

```
int TargetIntModel::columnCount (
            const QModelIndex & parent ) const
```

Definition at line 13 of file TargetIntModel.cpp.

7.93.3.2 data()

```
QVariant TargetIntModel::data (
            const QModelIndex & index,
            int role ) const
```

Definition at line 18 of file TargetIntModel.cpp.

7.93.3.3 flags()

```
Qt::ItemFlags TargetIntModel::flags (
            const QModelIndex & index ) const
```

Definition at line 158 of file TargetIntModel.cpp.

7.93.3.4 getLines()

```
QList< TargetIntData > TargetIntModel::getLines ( ) const [inline]
```

Definition at line 38 of file TargetIntModel.h.

7.93.3.5 headerData()

```
QVariant TargetIntModel::headerData (
            int section,
            Qt::Orientation orientation,
            int role ) const
```

Definition at line 64 of file TargetIntModel.cpp.

7.93.3.6 insertRows()

```
bool TargetIntModel::insertRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 133 of file TargetIntModel.cpp.

7.93.3.7 removeRows()

```
bool TargetIntModel::removeRows (
            int position,
            int rows,
            const QModelIndex & index = QModelIndex() )
```

Definition at line 146 of file TargetIntModel.cpp.

7.93.3.8 rowCount()

```
int TargetIntModel::rowCount (
            const QModelIndex & parent ) const
```

Definition at line 8 of file TargetIntModel.cpp.

7.93.3.9 setData()

```
bool TargetIntModel::setData (
            const QModelIndex & index,
            const QVariant & value,
            int role = Qt::EditRole )
```

Definition at line 99 of file TargetIntModel.cpp.

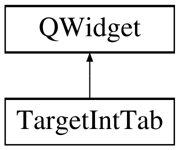
The documentation for this class was generated from the following files:

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntModel.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntModel.cpp

7.94 TargetIntTab Class Reference

```
#include <TargetIntTab.h>
```

Inheritance diagram for TargetIntTab:



Public Slots

- void addLine ()
- void addLine (TargetIntData line)
- void editLine ()
- void deleteLine ()
- void updateButtons (const QItemSelection &selection)
- void showInfo (int which=0, QString title="")

Public Member Functions

- TargetIntTab (QWidget ∗parent=0)
- TargetIntModel ∗ getTargetIntModel ()
- bool writeFile (QTextStream &outStream)
- bool readFile (QTextStream &inStream)
- void reset ()

7.94.1 Detailed Description

Definition at line 21 of file TargetIntTab.h.

7.94.2 Constructor & Destructor Documentation

7.94.2.1 TargetIntTab()

```
TargetIntTab::TargetIntTab (
            QWidget * parent = 0 )
```

Definition at line 14 of file TargetIntTab.cpp.

7.94.3 Member Function Documentation

7.94.3.1 addLine [1/2]

```
void TargetIntTab::addLine ( )  [slot]
```

Definition at line 74 of file TargetIntTab.cpp.

7.94.3.2 addLine [2/2]

```
void TargetIntTab::addLine (
            TargetIntData line )  [slot]
```

Definition at line 100 of file TargetIntTab.cpp.

7.94.3.3 deleteLine

```
void TargetIntTab::deleteLine ( )  [slot]
```

Definition at line 257 of file TargetIntTab.cpp.

7.94.3.4 editLine

```
void TargetIntTab::editLine ( )  [slot]
```

Definition at line 131 of file TargetIntTab.cpp.

7.94.3.5 getTargetIntModel()

```
TargetIntModel * TargetIntTab::getTargetIntModel ( )
```

Definition at line 70 of file TargetIntTab.cpp.

7.94.3.6 readFile()

```
bool TargetIntTab::readFile (
            QTextStream & inStream )
```

Definition at line 304 of file TargetIntTab.cpp.

7.94.3.7 reset()

```
void TargetIntTab::reset ( )
```

Definition at line 369 of file TargetIntTab.cpp.

7.94.3.8 showInfo

```
void TargetIntTab::showInfo (
            int which = 0,
            QString title = "" )  [slot]
```

Definition at line 373 of file TargetIntTab.cpp.

7.94.3.9 updateButtons

```
void TargetIntTab::updateButtons (
            const QItemSelection & selection )  [slot]
```

Definition at line 265 of file TargetIntTab.cpp.

7.94.3.10 writeFile()

```
bool TargetIntTab::writeFile (
            QTextStream & outStream )
```

Definition at line 275 of file TargetIntTab.cpp.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntTab.h
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InTabDocs.cpp
- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntTab.cpp

7.95 TempTMatrix Struct Reference

A temporaray T-Matrix structure.

```
#include <GenMatrixFunc.h>
```

Public Attributes

- double jValue

  Total spin value of temporary matrix element.
- int lValue

  Entrance orbital angular momentum for temporary matrix element.
- int lpValue

  Exit orbital angular momentum for temporary matrix element.
- complex TMatrix

  Value of temporary matrix element.

7.95.1 Detailed Description

A temporaray T-Matrix structure.

The TempTMatrix structure is used to coherently add T-matrix elements from pathways with like J, l, l' values for the calculation of angle integrated cross section. This is primarly used to facilitate the interference between internal and external pathways.

Definition at line 18 of file GenMatrixFunc.h.

7.95.2 Member Data Documentation

7.95.2.1 jValue

`double TempTMatrix::jValue`

Total spin value of temporary matrix element.

Definition at line 20 of file GenMatrixFunc.h.

7.95.2.2 lpValue

`int TempTMatrix::lpValue`

Exit orbital angular momentum for temporary matrix element.

Definition at line 24 of file GenMatrixFunc.h.

7.95.2.3 lValue

`int TempTMatrix::lValue`

Entrance orbital angular momentum for temporary matrix element.

Definition at line 22 of file GenMatrixFunc.h.

7.95.2.4 TMatrix

`complex TempTMatrix::TMatrix`

Value of temporary matrix element.

Definition at line 26 of file GenMatrixFunc.h.

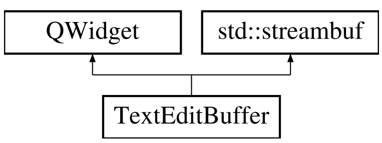
The documentation for this struct was generated from the following file:

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/GenMatrixFunc.h

7.96 TextEditBuffer Class Reference

`#include <TextEditBuffer.h>`

Inheritance diagram for TextEditBuffer:



Signals

- void updateLog (QString)

Public Member Functions

- TextEditBuffer (std::size_t buff_size=256, QWidget *parent=0)

Protected Member Functions

- virtual int_type overflow (int_type ch)
- virtual int sync ()

7.96.1 Detailed Description

Definition at line 13 of file TextEditBuffer.h.

7.96.2 Constructor & Destructor Documentation

7.96.2.1 TextEditBuffer()

```
TextEditBuffer::TextEditBuffer (
            std::size_t buff_size = 256,
            QWidget * parent = 0 )  [inline]
```

Definition at line 18 of file TextEditBuffer.h.

7.96.3 Member Function Documentation

7.96.3.1 overflow()

```
virtual int_type TextEditBuffer::overflow (
            int_type ch )  [inline], [protected], [virtual]
```

Definition at line 26 of file TextEditBuffer.h.

7.96.3.2 sync()

```
virtual int TextEditBuffer::sync ( )  [inline], [protected], [virtual]
```

Definition at line 35 of file TextEditBuffer.h.

7.96.3.3 updateLog

```
void TextEditBuffer::updateLog (
            QString  )  [signal]
```

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TextEditBuffer.h

7.97 WhitFunc Class Reference

A function class to calculate Whittaker functions for negative energy channels.

```
#include <WhitFunc.h>
```

Public Member Functions

- WhitFunc (PPair ∗pPair)
- int z1 () const
- int z2 () const
- double redmass () const
- double operator() (int l, double radius, double energy) const

7.97.1 Detailed Description

A function class to calculate Whittaker functions for negative energy channels.

The function class WhitFunc uses the GSL package to calculate Whittaker functions for negative energy channels from the GSL confluent hypergeometric functions.

Definition at line 16 of file WhitFunc.h.

7.97.2 Constructor & Destructor Documentation

7.97.2.1 WhitFunc()

```
WhitFunc::WhitFunc (
            PPair * pPair )  [inline]
```

The WhitFunc object is created with reference to a PPair object.

Definition at line 21 of file WhitFunc.h.

7.97.3 Member Function Documentation

7.97.3.1 operator()()

```
double WhitFunc::operator() (
            int l,
            double radius,
            double energy ) const  [inline]
```

The parenthesis operator is defined to make the class instance callable as a function. The orbital angular momentum, binding energy, and radius are the dependent variables. The function returns the value of the Whittaker function.

Definition at line 49 of file WhitFunc.h.

7.97.3.2 redmass()

```
double WhitFunc::redmass ( ) const  [inline]
```

Returns the reduced mass of the particle pair.

Definition at line 41 of file WhitFunc.h.

7.97.3.3 z1()

```
int WhitFunc::z1 ( ) const  [inline]
```

Returns the atomic number of the first particle in the pair.

Definition at line 29 of file WhitFunc.h.

7.97.3.4 z2()

```
int WhitFunc::z2 ( ) const  [inline]
```

Returns the atomic number of the second particle in the pair.

Definition at line 35 of file WhitFunc.h.

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

- /Users/kuba/Desktop/R-Matrix/AZURE2/include/WhitFunc.h

Chapter 8

File Documentation

8.1 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/complex_↩ functions.H File Reference

```
#include <complex>
#include <iostream>
#include <cstdlib>
```

Macros

- #define SIGN(a) (((a) < 0) ? (-1) : (1))

Functions

- double inf_norm (const std::complex< double > &z)
- bool isfinite (const std::complex< double > &z)
- std::complex< double > operator+ (const std::complex< double > &z, const int n)
- std::complex< double > operator- (const std::complex< double > &z, const int n)
- std::complex< double > operator* (const std::complex< double > &z, const int n)
- std::complex< double > operator/ (const std::complex< double > &z, const int n)
- std::complex< double > operator+ (const int n, const std::complex< double > &z)
- std::complex< double > operator- (const int n, const std::complex< double > &z)
- std::complex< double > operator* (const int n, const std::complex< double > &z)
- std::complex< double > operator/ (const int n, const std::complex< double > &z)
- std::complex< double > operator+ (const std::complex< double > &z, const unsigned int n)
- std::complex< double > operator- (const std::complex< double > &z, const unsigned int n)
- std::complex< double > operator* (const std::complex< double > &z, const unsigned int n)
- std::complex< double > operator/ (const std::complex< double > &z, const unsigned int n)
- std::complex< double > operator+ (const unsigned int n, const std::complex< double > &z)
- std::complex< double > operator- (const unsigned int n, const std::complex< double > &z)
- std::complex< double > operator* (const unsigned int n, const std::complex< double > &z)
- std::complex< double > operator/ (const unsigned int n, const std::complex< double > &z)
- bool operator== (const std::complex< double > &z, const int n)
- bool operator!= (const std::complex< double > &z, const int n)
- bool operator== (const int n, const std::complex< double > &z)

- bool operator!= (const int n, const std::complex< double > &z)
- bool operator== (const std::complex< double > &z, const unsigned int n)
- bool operator!= (const std::complex< double > &z, const unsigned int n)
- bool operator== (const unsigned int n, const std::complex< double > &z)
- bool operator!= (const unsigned int n, const std::complex< double > &z)
- std::complex< double > expm1 (const std::complex< double > &z)
- std::complex< double > log1p (const std::complex< double > &z)
- std::complex< double > log_Gamma (const std::complex< double > &z)
- std::complex< double > sigma_l_calc (const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > log_Cl_eta_calc (const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > log_cut_constant_AS_calc (const int omega, const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > log_cut_constant_CFa_calc (const bool is_it_normalized, const int omega, const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > log_cut_constant_CFb_calc (const bool is_it_normalized, const int omega, const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > sin_chi_calc (const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > exp_I_omega_chi_calc (const int omega, const std::complex< double > &l, const std::complex< double > &eta)

Variables

- const double precision = 1E-10
- const double sqrt_precision = 1E-5

8.1.1 Macro Definition Documentation

8.1.1.1 SIGN

```
#define SIGN(
              a ) (((a) < 0) ? (-1) : (1))
```

Definition at line 7 of file complex_functions.H.

8.1.2 Function Documentation

8.1.2.1 exp_I_omega_chi_calc()

```
std::complex< double > exp_I_omega_chi_calc (
            const int omega,
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 360 of file complex_functions.cpp.

8.1.2.2 expm1()

```
std::complex< double > expm1 (
            const std::complex< double > & z )
```

Definition at line 10 of file complex_functions.cpp.

8.1.2.3 inf_norm()

```
double inf_norm (
            const std::complex< double > & z ) [inline]
```

Definition at line 14 of file complex_functions.H.

8.1.2.4 isfinite()

```
bool isfinite (
            const std::complex< double > & z ) [inline]
```

Definition at line 24 of file complex_functions.H.

8.1.2.5 log1p()

```
std::complex< double > log1p (
            const std::complex< double > & z )
```

Definition at line 31 of file complex_functions.cpp.

8.1.2.6 log_Cl_eta_calc()

```
std::complex< double > log_Cl_eta_calc (
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 160 of file complex_functions.cpp.

8.1.2.7 log_cut_constant_AS_calc()

```
std::complex< double > log_cut_constant_AS_calc (
            const int omega,
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 200 of file complex_functions.cpp.

8.1.2.8 log_cut_constant_CFa_calc()

```
std::complex< double > log_cut_constant_CFa_calc (
            const bool is_it_normalized,
            const int omega,
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 254 of file complex_functions.cpp.

8.1.2.9 log_cut_constant_CFb_calc()

```
std::complex< double > log_cut_constant_CFb_calc (
            const bool is_it_normalized,
            const int omega,
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 307 of file complex_functions.cpp.

8.1.2.10 log_Gamma()

```
std::complex< double > log_Gamma (
            const std::complex< double > & z )
```

Definition at line 68 of file complex_functions.cpp.

8.1.2.11 operator"!=() [1/4]

```
bool operator!= (
            const int n,
            const std::complex< double > & z ) [inline]
```

Definition at line 131 of file complex_functions.H.

8.1.2.12 operator"!=() [2/4]

```
bool operator!= (
            const std::complex< double > & z,
            const int n ) [inline]
```

Definition at line 121 of file complex_functions.H.

8.1.2.13 operator"!=() [3/4]

```
bool operator!= (
            const std::complex< double > & z,
            const unsigned int n ) [inline]
```

Definition at line 141 of file complex_functions.H.

8.1.2.14 operator"!=() [4/4]

```
bool operator!= (
            const unsigned int n,
            const std::complex< double > & z )  [inline]
```

Definition at line 151 of file complex_functions.H.

8.1.2.15 operator*() [1/4]

```
std::complex< double > operator* (
            const int n,
            const std::complex< double > & z )  [inline]
```

Definition at line 66 of file complex_functions.H.

8.1.2.16 operator*() [2/4]

```
std::complex< double > operator* (
            const std::complex< double > & z,
            const int n )  [inline]
```

Definition at line 46 of file complex_functions.H.

8.1.2.17 operator*() [3/4]

```
std::complex< double > operator* (
            const std::complex< double > & z,
            const unsigned int n )  [inline]
```

Definition at line 86 of file complex_functions.H.

8.1.2.18 operator*() [4/4]

```
std::complex< double > operator* (
            const unsigned int n,
            const std::complex< double > & z )  [inline]
```

Definition at line 106 of file complex_functions.H.

8.1.2.19 operator+() [1/4]

```
std::complex< double > operator+ (
            const int n,
            const std::complex< double > & z )  [inline]
```

Definition at line 56 of file complex_functions.H.

8.1.2.20 operator+() [2/4]

```
std::complex< double > operator+ (
            const std::complex< double > & z,
            const int n ) [inline]
```

Definition at line 36 of file complex_functions.H.

8.1.2.21 operator+() [3/4]

```
std::complex< double > operator+ (
            const std::complex< double > & z,
            const unsigned int n ) [inline]
```

Definition at line 76 of file complex_functions.H.

8.1.2.22 operator+() [4/4]

```
std::complex< double > operator+ (
            const unsigned int n,
            const std::complex< double > & z ) [inline]
```

Definition at line 96 of file complex_functions.H.

8.1.2.23 operator-() [1/4]

```
std::complex< double > operator- (
            const int n,
            const std::complex< double > & z ) [inline]
```

Definition at line 61 of file complex_functions.H.

8.1.2.24 operator-() [2/4]

```
std::complex< double > operator- (
            const std::complex< double > & z,
            const int n ) [inline]
```

Definition at line 41 of file complex_functions.H.

8.1.2.25 operator-() [3/4]

```
std::complex< double > operator- (
            const std::complex< double > & z,
            const unsigned int n ) [inline]
```

Definition at line 81 of file complex_functions.H.

8.1.2.26 operator-() [4/4]

```
std::complex< double > operator- (
            const unsigned int n,
            const std::complex< double > & z )  [inline]
```

Definition at line 101 of file complex_functions.H.

8.1.2.27 operator/() [1/4]

```
std::complex< double > operator/ (
            const int n,
            const std::complex< double > & z )  [inline]
```

Definition at line 71 of file complex_functions.H.

8.1.2.28 operator/() [2/4]

```
std::complex< double > operator/ (
            const std::complex< double > & z,
            const int n )  [inline]
```

Definition at line 51 of file complex_functions.H.

8.1.2.29 operator/() [3/4]

```
std::complex< double > operator/ (
            const std::complex< double > & z,
            const unsigned int n )  [inline]
```

Definition at line 91 of file complex_functions.H.

8.1.2.30 operator/() [4/4]

```
std::complex< double > operator/ (
            const unsigned int n,
            const std::complex< double > & z )  [inline]
```

Definition at line 111 of file complex_functions.H.

8.1.2.31 operator==() [1/4]

```
bool operator== (
            const int n,
            const std::complex< double > & z )  [inline]
```

Definition at line 126 of file complex_functions.H.

8.1.2.32 operator==() [2/4]

```
bool operator== (
            const std::complex< double > & z,
            const int n )  [inline]
```

Definition at line 116 of file complex_functions.H.

8.1.2.33 operator==() [3/4]

```
bool operator== (
            const std::complex< double > & z,
            const unsigned int n )  [inline]
```

Definition at line 136 of file complex_functions.H.

8.1.2.34 operator==() [4/4]

```
bool operator== (
            const unsigned int n,
            const std::complex< double > & z )  [inline]
```

Definition at line 146 of file complex_functions.H.

8.1.2.35 sigma_l_calc()

```
std::complex< double > sigma_l_calc (
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 126 of file complex_functions.cpp.

8.1.2.36 sin_chi_calc()

```
std::complex< double > sin_chi_calc (
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 330 of file complex_functions.cpp.

8.1.3 Variable Documentation

8.1.3.1 precision

```
const double precision = 1E-10
```

Definition at line 8 of file complex_functions.H.

8.1.3.2 sqrt_precision

```
const double sqrt_precision = 1E-5
```

Definition at line 8 of file complex_functions.H.

8.2 complex_functions.H

Go to the documentation of this file.

```
00001 #ifndef COMPLEX_FUNCTIONS_H
00002 #define COMPLEX_FUNCTIONS_H
00003
00004 #include <complex>
00005 #include <iostream>
00006 #include <cstdlib>
00007 #define SIGN(a) (((a) < 0) ? (-1) : (1))
00008 const double precision = 1E-10,sqrt_precision = 1E-5;
00009
00010 // Infinite norm of a complex number.
00011 // ----------------------------------
00012 // It is max(|Re[z]|,|Im[z]|)
00013
00014 inline double inf_norm (const std::complex<double> &z)
00015 {
00016   return std::max (std::abs (real (z)),std::abs (imag (z)));
00017 }
00018
00019 // Test of finiteness of a complex number
00020 // --------------------------------------
00021 // If real or imaginary parts are finite, true is returned.
00022 // Otherwise, false is returned
00023
00024 inline bool isfinite (const std::complex<double> &z)
00025 {
00026   const double x = real (z), y = imag (z);
00027
00028   return (finite (x) && finite (y));
00029 }
00030
00031 // Usual operator overloads of complex numbers with integers
00032 // ---------------------------------------------------------
00033 // Recent complex libraries do not accept for example z+n or z==n with n integer, signed or unsigned.
00034 // The operator overload is done here, by simply putting a cast on double to the integer.
00035
00036 inline std::complex<double> operator + (const std::complex<double> &z,const int n)
00037 {
00038   return (z+static_cast<double> (n));
00039 }
00040
00041 inline std::complex<double> operator - (const std::complex<double> &z,const int n)
00042 {
00043   return (z-static_cast<double> (n));
00044 }
00045
00046 inline std::complex<double> operator * (const std::complex<double> &z,const int n)
00047 {
00048   return (z*static_cast<double> (n));
00049 }
00050
00051 inline std::complex<double> operator / (const std::complex<double> &z,const int n)
00052 {
00053   return (z/static_cast<double> (n));
00054 }
00055
00056 inline std::complex<double> operator + (const int n,const std::complex<double> &z)
00057 {
00058   return (static_cast<double> (n)+z);
00059 }
00060
00061 inline std::complex<double> operator - (const int n,const std::complex<double> &z)
00062 {
00063   return (static_cast<double> (n)-z);
00064 }
00065
00066 inline std::complex<double> operator * (const int n,const std::complex<double> &z)
00067 {
00068   return (static_cast<double> (n)*z);
00069 }
```

```
inline std::complex<double> operator / (const int n,const std::complex<double> &z)
{
  return (static_cast<double> (n)/z);
}

inline std::complex<double> operator + (const std::complex<double> &z,const unsigned int n)
{
  return (z+static_cast<double> (n));
}

inline std::complex<double> operator - (const std::complex<double> &z,const unsigned int n)
{
  return (z-static_cast<double> (n));
}

inline std::complex<double> operator * (const std::complex<double> &z,const unsigned int n)
{
  return (z*static_cast<double> (n));
}

inline std::complex<double> operator / (const std::complex<double> &z,const unsigned int n)
{
  return (z/static_cast<double> (n));
}

inline std::complex<double> operator + (const unsigned int n,const std::complex<double> &z)
{
  return (static_cast<double> (n)+z);
}

inline std::complex<double> operator - (const unsigned int n,const std::complex<double> &z)
{
  return (static_cast<double> (n)-z);
}

inline std::complex<double> operator * (const unsigned int n,const std::complex<double> &z)
{
  return (static_cast<double> (n)*z);
}

inline std::complex<double> operator / (const unsigned int n,const std::complex<double> &z)
{
  return (static_cast<double> (n)/z);
}

inline bool operator == (const std::complex<double> &z,const int n)
{
  return (z == static_cast<double> (n));
}

inline bool operator != (const std::complex<double> &z,const int n)
{
  return (z != static_cast<double> (n));
}

inline bool operator == (const int n,const std::complex<double> &z)
{
  return (static_cast<double> (n) == z);
}

inline bool operator != (const int n,const std::complex<double> &z)
{
  return (static_cast<double> (n) != z);
}

inline bool operator == (const std::complex<double> &z,const unsigned int n)
{
  return (z == static_cast<double> (n));
}

inline bool operator != (const std::complex<double> &z,const unsigned int n)
{
  return (z != static_cast<double> (n));
}

inline bool operator == (const unsigned int n,const std::complex<double> &z)
{
  return (static_cast<double> (n) == z);
}

inline bool operator != (const unsigned int n,const std::complex<double> &z)
{
  return (static_cast<double> (n) != z);
}

extern std::complex<double> expm1 (const std::complex<double> &z);
```

```
extern std::complex<double> log1p (const std::complex<double> &z);
extern std::complex<double> log_Gamma (const std::complex<double> &z);
extern std::complex<double> sigma_l_calc (const std::complex<double> &l,const std::complex<double> &eta);
extern std::complex<double> log_Cl_eta_calc (const std::complex<double> &l,const std::complex<double> &eta);
extern std::complex<double> log_cut_constant_AS_calc (const int omega,const std::complex<double> &l,const std::complex<double> &eta);
extern std::complex<double> log_cut_constant_CFa_calc (const bool is_it_normalized,const int omega,const std::complex<double> &l,const std::complex<double> &eta);
extern std::complex<double> log_cut_constant_CFb_calc (const bool is_it_normalized,const int omega,const std::complex<double> &l,const std::complex<double> &eta);
extern std::complex<double> sin_chi_calc (const std::complex<double> &l,const std::complex<double> &eta);
extern std::complex<double> exp_I_omega_chi_calc (const int omega,const std::complex<double> &l,const std::complex<double> &eta);
#endif
```

8.3 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/cwfcomp.H File Reference

```
#include ¨ode_int.H¨
```

Classes

- class Coulomb_wave_functions

8.4 cwfcomp.H

Go to the documentation of this file.

```
#ifndef CWFCOMP_H
#define CWFCOMP_H
#include "ode_int.H"
// Class to calculate the Coulomb wave functions
// ---------------------------------------------
class Coulomb_wave_functions
{
public:
  // Constructor.
  // ------------
  // Constants are defined in the constructor,
  // plus a pointer to class ODE_integration, ODE_ptr, to integrate numerically the regular Coulomb wave function.
  //
  // Variables:
  // ----------
  // is_it_normalized_c : true if one wants normalized functions, i.e. the standard normalization,
  //                      false if one wants F -> F/C(l,eta) and H+/H-/G -> H+/H-/G.C(l,eta), to avoid overflows for |eta| » 1 and |z| small.
  // l_c : orbital angular momentum.
  // eta_c : Sommerfeld parameter.
  Coulomb_wave_functions (const bool is_it_normalized_c,const std::complex<double> &l_c,const std::complex<double> &eta_c)
    : l (l_c),
```

```
      is_it_normalized (is_it_normalized_c),
      eta (eta_c),
      neg_int_omega_one ((rint (real (l_c + std::complex<double> (-imag (eta_c),real (eta_c)))) == l_c + std::complex<double> (-imag (eta_c),real (eta_c))) &&
             (rint (real (1 + l_c + std::complex<double> (-imag (eta_c),real (eta_c)))) <= 0.0)),
      neg_int_omega_minus_one ((rint (real (l_c - std::complex<double> (-imag (eta_c),real (eta_c)))) == l_c - std::complex<double> (-imag (eta_c),real (eta_c))) &&
                   (rint (real (1 + l_c - std::complex<double> (-imag (eta_c),real (eta_c)))) <= 0.0)),
      sigma_l (sigma_l_calc (l_c,eta_c)),
      log_Cl_eta (log_Cl_eta_calc (l_c,eta_c)),
      Cl_eta (exp (log_Cl_eta_calc (l_c,eta_c))),
      exp_I_chi (exp_I_omega_chi_calc (1,l_c,eta_c)),
      exp_minus_I_chi (exp_I_omega_chi_calc (-1,l_c,eta_c)),
      one_over_sin_chi (1.0/sin_chi_calc (l_c,eta_c)),
      log_cut_constant_CFa_plus (log_cut_constant_CFa_calc (is_it_normalized_c,1,l_c,eta_c)),
      log_cut_constant_CFa_minus (log_cut_constant_CFa_calc (is_it_normalized_c,-1,l_c,eta_c)),
      cut_constant_CFa_plus (exp (log_cut_constant_CFa_calc (is_it_normalized_c,1,l_c,eta_c))),
      cut_constant_CFa_minus (exp (log_cut_constant_CFa_calc (is_it_normalized_c,-1,l_c,eta_c))),
      log_cut_constant_CFb_plus (log_cut_constant_CFb_calc (is_it_normalized_c,1,l_c,eta_c)),
      log_cut_constant_CFb_minus (log_cut_constant_CFb_calc (is_it_normalized_c,-1,l_c,eta_c)),
      log_cut_constant_AS_plus (log_cut_constant_AS_calc (1,l_c,eta_c)),
      log_cut_constant_AS_minus (log_cut_constant_AS_calc (-1,l_c,eta_c)),
      cut_constant_CFb_plus (exp (log_cut_constant_CFb_calc (is_it_normalized_c,1,l_c,eta_c))),
      cut_constant_CFb_minus (exp (log_cut_constant_CFb_calc (is_it_normalized_c,-1,l_c,eta_c))),
      log_sym_constant_arg_neg ((is_it_normalized_c) ? (-M_PI*(eta_c+(l_c+1)*std::complex<double> (0.0,1.0))) : (-M_PI*(l_c+1)*std::complex<double> (0.0,1.0))),
      log_sym_constant_arg_pos ((is_it_normalized_c) ? (-M_PI*(eta_c-(l_c+1)*std::complex<double> (0.0,1.0))) : (M_PI*(l_c+1)*std::complex<double> (0.0,1.0))),
      sym_constant_arg_neg ((is_it_normalized_c) ? (exp (-M_PI*(eta_c+(l_c+1)*std::complex<double> (0.0,1.0)))) : (exp (-M_PI*(l_c+1)*std::complex<double> (0.0,1.0)))),
      sym_constant_arg_pos ((is_it_normalized_c) ? (exp (-M_PI*(eta_c-(l_c+1)*std::complex<double> (0.0,1.0)))) : (exp (M_PI*(l_c+1)*std::complex<double> (0.0,1.0)))),
      turning_point (std::max (1.0,abs (eta_c) + sqrt (abs (l_c*(l_c+1.0)) + abs (eta_c*eta_c)))),
      is_H_dir_int_naive (false),cwf_real_ptr (0),cwf_real_eta_plus_ptr (0),cwf_real_eta_minus_ptr (0),cwf_real_l_plus_ptr (0),cwf_real_l_minus_ptr (0),
      cwf_minus_eta_ptr (0),cwf_lp_ptr (0),prec_first_order_expansion (0.1*sqrt_precision)
  {
    ODE_ptr = new class ODE_integration (l,2.0*eta);

    debut = 0.0;

    if (real (l) >= 0.0)
    {
      F_debut = 0.0;
      dF_debut = (l == 0) ? ((is_it_normalized) ? (Cl_eta) : (1.0)) : (0.0);
    }
  }

  ~Coulomb_wave_functions (void)
  {
    delete cwf_real_ptr;

    delete cwf_real_l_plus_ptr;
    delete cwf_real_l_minus_ptr;

    delete cwf_real_eta_plus_ptr;
    delete cwf_real_eta_minus_ptr;

    delete cwf_minus_eta_ptr;
    delete cwf_lp_ptr;

    delete ODE_ptr;
  }

  void F_dF_init (const std::complex<double> &z,const std::complex<double> &F,const std::complex<double> &dF);

  void F_dF (const std::complex<double> &z,std::complex<double> &F,std::complex<double> &dF);
  void G_dG (const std::complex<double> &z,std::complex<double> &G,std::complex<double> &dG);
  void H_dH (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH);
  void H_dH_scaled (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH);

  const std::complex<double> l,eta; // Angular momentum and Sommerfeld parameter.

  const bool is_it_normalized;
  // true if F(z) ~ C(l,eta).z^{l+1} in 0, false if F(z) ~ z^{l+1} in 0.

private:

  void asymptotic_series (const int omega,const std::complex<double> &one_over_z,std::complex<double> sum[],std::complex<double> dsum[],bool &is_it_successful);

  std::complex<double> continued_fraction_f (const std::complex<double> &z,const int omega);
  std::complex<double> continued_fraction_h (const std::complex<double> &z,const int omega);
```

```
  void F_dF_power_series (const std::complex<double> &z,std::complex<double> &F,std::complex<double> &dF);

  void asymptotic_expansion_F_dF (const std::complex<double> &z,std::complex<double> &F,std::complex<double> &dF,bool &is_it_successful);
  void asymptotic_expansion_H_dH_scaled (const int omega,const std::complex<double> &one_over_z,
                        std::complex<double> &H_scaled,std::complex<double> &dH_scaled,bool &is_it_successful);

  void F_dF_direct_integration (const std::complex<double> &z,std::complex<double> &F,std::complex<double> &dF,bool &is_it_successful);
  void H_dH_direct_integration (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH);

  void partial_derivatives (const bool is_it_regular,const bool is_it_eta,const double x,double &d_chi_Bx,double &d_chi_dBx);
  void first_order_expansions (const bool is_it_regular,const std::complex<double> &z,std::complex<double> &B,std::complex<double> &dB);
  void H_dH_from_first_order_expansions (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH);

  void H_dH_with_F_dF_and_CF (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH);
  void H_dH_with_expansion (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH,bool &is_it_successful);

  void F_dF_with_symmetry_relations (const std::complex<double> &z,std::complex<double> &F,std::complex<double> &dF);

  const bool neg_int_omega_one,neg_int_omega_minus_one;
  // neg_int_omega_one : true if 1+l+i.eta is negative integer, false if not.
  // neg_int_omega_minus_one : true if 1+l-i.eta is negative integer, false if not.

  const std::complex<double> log_Cl_eta,Cl_eta,sigma_l; // log[C(l,eta)], C(l,eta), sigma(l,eta)

  const std::complex<double> cut_constant_CFa_plus,cut_constant_CFa_minus,cut_constant_CFb_plus,cut_constant_CFb_minus;
  const std::complex<double> log_cut_constant_CFa_plus,log_cut_constant_CFa_minus,log_cut_constant_CFb_plus,log_cut_constant_CFb_minus;
  const std::complex<double> log_cut_constant_AS_plus,log_cut_constant_AS_minus;
  // cut constants and their logs for continued fractions (CFa and CFb) and asymptotic series (AS).
  // plus,minus is for omega = 1 or -1.
  // See functions log_cut_constant_AS_calc, log_cut_constant_CFa_calc and log_cut_constant_CFb_calc.

  const std::complex<double> exp_I_chi,exp_minus_I_chi,one_over_sin_chi;
  // exp (i.chi), exp (-i.chi), 1/sin (chi) with chi = sigma(l,eta) - sigma(-l-1,eta) - (l+1/2).Pi .
  // They are used to calculate H+/H- from F(l,eta,z) and F(-l-1,eta,z) if |Im[l]| >= 1 and |z| <= 1 .

  const std::complex<double> sym_constant_arg_neg,sym_constant_arg_pos,log_sym_constant_arg_neg,log_sym_constant_arg_pos;
  // Multiplicative constants and their logs used in the following reflection formulas :
  // F(l,eta,z) = -F(l,-eta,-z).exp[-Pi.(eta-i.l)] if arg (z) > 0 and is_it_normalized is true, so sym_constant_arg_pos = -exp[-Pi.(eta-i.l)],
  // F(l,eta,z) = -F(l,-eta,-z).exp[-Pi.(eta+i.l)] if arg (z) <= 0 and is_it_normalized is true, so sym_constant_arg_neg = -exp[-Pi.(eta-i.l)],
  // F(l,eta,z) = -F(l,-eta,-z).exp[i.Pi.l)] if arg (z) > 0 and is_it_normalized is false, so sym_constant_arg_pos = -exp[i.Pi.l)],
  // F(l,eta,z) = -F(l,-eta,-z).exp[-i.Pi.l)] if arg (z) <= 0 and is_it_normalized is false, so sym_constant_arg_neg = -exp[-i.Pi.l)].

  const double turning_point,prec_first_order_expansion; // turning_point : max (1,||eta| + sqrt[|l(l+1)| + |eta|^2]|).
                                                          // prec_first_order_expansion : 0.1*sqrt_precision. It is the precision used for first_order_expansions.

  bool is_H_dir_int_naive; // true if one integrates H+/H- forward without considering |H+/H-|, false if not. It is false except in continued_fraction_h.

  std::complex<double> debut,F_debut,dF_debut;
  // Coulomb wave functions and derivative at z = debut.
  // It is used to integrate the Coulomb wave function faster,
  // as debut is usually close to the argument of the Coulomb wave function so that the integration is quicker and more stable.

  class ODE_integration *ODE_ptr;  // pointer to class ODE_integration to integrate numerically the Coulomb equation.

  class Coulomb_wave_functions *cwf_real_ptr,*cwf_real_l_plus_ptr,*cwf_real_l_minus_ptr,*cwf_real_eta_plus_ptr,*cwf_real_eta_minus_ptr;
  // pointers to classes Coulomb_wave_functions of parameters (l_r,eta_r) (one has eta_r = Re[eta], eta_i = Im[eta], l_r = Re[l] and l_i = Im[l]),
  // (l_r +/- epsilon[l],eta_r) and (l_r,eta_r +/- epsilon[eta]).
  // They are first put to zero and allocated in the program when they are needed.
  // They are used for the first order expansion method when |l_i| « 1, |eta_i| « 1 and |Im[z]| « Re[z] with Re[z] > 0.

```

```
  class Coulomb_wave_functions *cwf_minus_eta_ptr,*cwf_lp_ptr;
  // pointers to classes Coulomb_wave_functions of parameters (l,-eta), (lp = -l-1,eta) and (l_r +/- precision,eta).
  // They are first put to zero and allocated in the program when they are needed.
  // They are used for symmetry relations : F(l,eta,z) \propto F(l,-eta,-z) and h[omega](l,eta,z) = -h[omega](l,-eta,-z)
  // and to calculate H+/H- from F(l,eta,z) and F(lp,eta,z) if |Im[l]| >= 1 and |z| <= 1.
};

#endif
```

8.5 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/ode_int.H File Reference

`#include "complex_functions.H"`

Classes

- class ODE_integration

8.6 ode_int.H

Go to the documentation of this file.

```
#ifndef ODE_INT_H
#define ODE_INT_H

#include "complex_functions.H"

// Direct integration of the Coulomb equation
// ------------------------------------------
// One uses the Burlisch-Stoer-Henrici method, where one integrates on different meshes
// with the Henrici method, and then uses the Richardson method to get the final result by extrapolation.
// Numerical Recipes, Chap. 16.4 .

class ODE_integration
{
public:
  ODE_integration (const std::complex<double> &l_1,
                   const std::complex<double> &two_eta_1)
    : l (l_1), ll_plus_one (l_1*(l_1+1.0)), two_eta (two_eta_1)
    {
      for (int n = 0 ; n < 8 ; n++)
    for (int i = 0 ; i < n ; i++)
    {
      interpolation_term_tab[n][i] = 1.0;

      for (int j = 0 ; j < n ; j++)
        if (i != j)
          interpolation_term_tab[n][i] *= (i+1.0)/(i-j);
    }

      for (unsigned int k = 0 ; k < 8 ; k++) m_tab[k] = 2*(k+1);
      for (unsigned int k = 0 ; k < 8 ; k++) one_over_m_tab[k] = 1.0/static_cast<double> (m_tab[k]);
    }

  void operator() (const std::complex<double> &r0,const std::complex<double> &u0,const std::complex<double> &du0,
                   const std::complex<double> &r,std::complex<double> &u,std::complex<double> &du) const;

  private:
  std::complex<double> extrapolation_in_zero (const unsigned int n,const std::complex<double> *f) const;
  std::complex<double> F_r_u (const std::complex<double> &r,const std::complex<double> &u) const;
  void integration_Henrici (const unsigned int m,const std::complex<double> &h,
                            const std::complex<double> &r0,const std::complex<double> &u0,const std::complex<double> &du0,
                            const std::complex<double> &r,std::complex<double> &u,std::complex<double> &du) const;

```

```
  const std::complex<double> l,ll_plus_one;  // angular momentum,l(l+1).
  const std::complex<double> two_eta;        // 2.eta, with eta the Sommerfeld parameter.

  unsigned int m_tab[8];                                // integers used in the extrapolation method.
  double one_over_m_tab[8],interpolation_term_tab[8][8]; // doubles used in the extrapolation method.
};


#endif
```

8.7 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/complex_functions.cpp File Reference

`#include "complex_functions.H"`

Functions

- std::complex< double > expm1 (const std::complex< double > &z)
- std::complex< double > log1p (const std::complex< double > &z)
- std::complex< double > log_Gamma (const std::complex< double > &z)
- std::complex< double > sigma_l_calc (const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > log_Cl_eta_calc (const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > log_cut_constant_AS_calc (const int omega, const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > log_cut_constant_CFa_calc (const bool is_it_normalized, const int omega, const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > log_cut_constant_CFb_calc (const bool is_it_normalized, const int omega, const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > sin_chi_calc (const std::complex< double > &l, const std::complex< double > &eta)
- std::complex< double > exp_I_omega_chi_calc (const int omega, const std::complex< double > &l, const std::complex< double > &eta)

8.7.1 Function Documentation

8.7.1.1 exp_I_omega_chi_calc()

```
std::complex< double > exp_I_omega_chi_calc (
            const int omega,
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 360 of file complex_functions.cpp.

8.7.1.2 expm1()

```
std::complex< double > expm1 (
            const std::complex< double > & z )
```

Definition at line 10 of file complex_functions.cpp.

8.7.1.3 log1p()

```
std::complex< double > log1p (
            const std::complex< double > & z )
```

Definition at line 31 of file complex_functions.cpp.

8.7.1.4 log_Cl_eta_calc()

```
std::complex< double > log_Cl_eta_calc (
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 160 of file complex_functions.cpp.

8.7.1.5 log_cut_constant_AS_calc()

```
std::complex< double > log_cut_constant_AS_calc (
            const int omega,
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 200 of file complex_functions.cpp.

8.7.1.6 log_cut_constant_CFa_calc()

```
std::complex< double > log_cut_constant_CFa_calc (
            const bool is_it_normalized,
            const int omega,
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 254 of file complex_functions.cpp.

8.7.1.7 log_cut_constant_CFb_calc()

```
std::complex< double > log_cut_constant_CFb_calc (
            const bool is_it_normalized,
            const int omega,
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 307 of file complex_functions.cpp.

8.7.1.8 log_Gamma()

```
std::complex< double > log_Gamma (
            const std::complex< double > & z )
```

Definition at line 68 of file complex_functions.cpp.

8.7.1.9 sigma_l_calc()

```
std::complex< double > sigma_l_calc (
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 126 of file complex_functions.cpp.

8.7.1.10 sin_chi_calc()

```
std::complex< double > sin_chi_calc (
            const std::complex< double > & l,
            const std::complex< double > & eta )
```

Definition at line 330 of file complex_functions.cpp.

8.8 complex_functions.cpp

Go to the documentation of this file.

```
#include "complex_functions.H"

// Precise evaluation of exp[z]-1 for z complex
// ---------------------------------------------
// When |Re[z]| >= 1 or |Im[z]| >= 1, one uses directly the standard exp function as it is precise.
// Otherwise, numerical cancellations can occur.
// So, one uses the always stable formula exp[z]-1 = expm1(x) - 2.exp(x).sin^2(y/2) + i.exp(x).sin(y)
// with x = Re[z] and y = Im[z]. expm1(x) gives a precise evaluation of exp(x)-1 for x double.

std::complex<double> expm1 (const std::complex<double> &z)
{
  const double x = real (z),y = imag (z);

  if ((std::abs (x) >= 1.0) || (std::abs (y) >= 1.0)) return (exp (z) - 1.0);

  const double expm1_x = expm1 (x),exp_x = 1.0 + expm1_x,sin_y_over_two = sin (0.5*y),sin_y = sin (y);

  return std::complex<double> (expm1_x - 2.0*exp_x*sin_y_over_two*sin_y_over_two,exp_x*sin_y);
}



// Precise evaluation of log[1+z] for z complex
// ---------------------------------------------
// When |Re[z]| >= 1 or |Im[z]| >= 1, one uses directly the standard log function as it is precise.
// Otherwise, numerical cancellations can occur.
// So, one uses the always stable formula log[1+z] = log1p(x) + log1p([y/(1+x)]^2)/2 + i.atan2(y,1+x)
// with x = Re[z] and y = Im[z]. log1p(x) gives a precise evaluation of log(1+x) for x double.
// atan2(x,y) gives the arc tangent of y/x so it is in ]-Pi:Pi].

std::complex<double> log1p (const std::complex<double> &z)
{
  const double x = real (z),y = imag (z);

  const double xp1 = 1.0 + x,abs_x = std::abs (x), abs_y = std::abs (y);

  if ((abs_x >= 1.0) || (abs_y >= 1.0)) return log (1.0 + z);

  const double y_over_xp1 = y/xp1;

  return std::complex<double> (log1p (x) + 0.5*log1p (y_over_xp1*y_over_xp1),atan2 (y,xp1));
}




// Logarithm of Gamma[z], z anywhere in the complex plane except in the Gamma function poles.
// ---------------------------------------------------------------------------------------------
// If z is not finite or is a negative integer, the program returns an error message and stops.
// The Lanczos method is used. Precision : < 2E-10 in theory, < 1E-12 in almost every case.
// The method works for Re[z] > 0.
// If Re[z] <= 0, one uses the formula Gamma[z].Gamma[1-z] = Pi/sin (Pi.z).
```

```
// log[sin(Pi.z)] is calculated with the Kolbig method (K.S. Kolbig, Comp. Phys. Comm., Vol. 4, p.221 (1972)) :
// If z = x+iy and y >= 0, log[sin(Pi.z)] = log[sin(Pi.eps)] - i.Pi.n, with z = n + eps so 0 <= Re[eps] < 1 and n integer.
// If y > 110, log[sin(Pi.z)] = -i.Pi.z + log[0.5] + i.Pi/2 numerically so that no overflow can occur.
// If z = x+iy and y < 0, log[Gamma(z)] = [log[Gamma(z*)]]*, so that one can use the previous formula with z*.
//
// Variables:
// ----------
// z,z_p_0p5,z_p_5p5 : argument of the Gamma function, z+0.5, z+5.5
// sqrt_2Pi,log_Pi : sqrt(2.Pi), log(Pi).
// sum : Rational function in the Lanczos method.
// log_Gamma_z : log[Gamma(z)] value.
// c : table containing the seven coefficients in the expansion used in the Lanczos method.
// eps : z = n + eps so 0 <= Re[eps] < 1 and n integer.
// log_const : log[0.5] + i.Pi/2

std::complex<double> log_Gamma (const std::complex<double> &z)
{
  if (!isfinite (z)) std::cout«"z is not finite in log_Gamma."«std::endl, abort ();

  const double x = real (z),y = imag (z);

  if ((z == rint (x)) && (x <= 0)) std::cout«"z is negative integer in log_Gamma."«std::endl, abort ();

  if (x > 0.0)
  {
    const std::complex<double> z_p_0p5 = z + 0.5, z_p_5p5 = z + 5.5;
    const double sqrt_2Pi = 2.5066282746310005;
    const double c[7] = {1.000000000190015,
             7.618009172947146E+1,
            -8.650532032941677E+1,
             2.401409824083091E+1,
            -1.231739572450155,
             0.1208650973866179E-2,
            -0.5395239384953000E-5};

    std::complex<double> sum = c[0];
    for (int i = 1 ; i < 7 ; i++) sum += c[i]/(z + i);
    sum *= sqrt_2Pi;

    const std::complex<double> log_Gamma_z = log (sum) - log (z) + z_p_0p5*log (z_p_5p5) - z_p_5p5;

    return log_Gamma_z;
  }
  else if (y >= 0.0)
  {
    const int n = (x < rint (x)) ? (static_cast<int> (rint (x)) - 1) : (static_cast<int> (rint (x)));
    const double log_Pi = 1.1447298858494002;
    const std::complex<double> log_const(-M_LN2,M_PI_2),i_Pi(0.0,M_PI);
    const std::complex<double> eps = z - n,log_sin_Pi_z = (y > 110) ? (-i_Pi*z + log_const) : (log (sin (M_PI*eps)) - i_Pi*n);
    const std::complex<double> log_Gamma_z = log_Pi - log_sin_Pi_z - log_Gamma (1.0 - z);

    return log_Gamma_z;
  }
  else
    return conj (log_Gamma (conj (z)));
}



// Coulomb phase shift.
// --------------------
// It is given by the formula [Gamma[1+l+I.eta] - Gamma[1+l-I.eta]]/[2i].
// 0 is returned if 1+l+/-I.eta is a negative integer.
//
// Variables:
// ----------
// l : orbital angular momentum l.
// eta : Sommerfeld parameter.
// Ieta,one_over_two_I : i.eta,1/[2.i] .
// arg_plus,arg_minus : 1+l+i.eta, 1+l-i.eta.
// log_Gamma_plus,log_Gamma_minus : logs of Gamma[1+l+I.eta], Gamma[1+l-I.eta].
// sigma_l : returned result.

std::complex<double> sigma_l_calc (const std::complex<double> &l,const std::complex<double> &eta)
{
  const std::complex<double> Ieta(-imag (eta),real (eta)),one_over_two_I(0,-0.5);
  const std::complex<double> arg_plus = 1.0 + l + Ieta, arg_minus = 1.0 + l - Ieta;

  if ((rint (real (arg_plus)) == arg_plus) && (rint (real (arg_plus)) <= 0.0)) return 0.0;
  if ((rint (real (arg_minus)) == arg_minus) && (rint (real (arg_minus)) <= 0.0)) return 0.0;

  const std::complex<double> log_Gamma_plus = log_Gamma (arg_plus),log_Gamma_minus = log_Gamma
```

```
  (arg_minus);
  const std::complex<double> sigma_l = (log_Gamma_plus - log_Gamma_minus)*one_over_two_I;

  return sigma_l;
}






// log of C(l,eta)
// ---------------
// It is given by the formula l*log[2] - eta.Pi/2 + (log[Gamma[1+l+I.eta]] +
  log[Gamma[1+l-I.eta]])/2.0 - log[Gamma[2l+2]].
// 0 is returned if 1+l+/-I.eta is a negative integer.
// 2l+2 should not be a negative integer : one has to use -l-1 instead of l in this case.
//
// Variables:
// ----------
// l : orbital angular momentum l.
// eta : Sommerfeld parameter.
// Ieta : i.eta .
// arg_plus,arg_minus : 1+l+i.eta, 1+l-i.eta.
// log_Gamma_plus,log_Gamma_minus,log_Gamma_2l_plus_2 : logs of Gamma[1+l+I.eta], Gamma[1+l-I.eta],
  Gamma[2l+2].
// log_Cl_eta : returned result.

std::complex<double> log_Cl_eta_calc (const std::complex<double> &l,const std::complex<double> &eta)
{
  const std::complex<double> Ieta(-imag (eta),real (eta));
  const std::complex<double> arg_plus = 1.0 + l + Ieta, arg_minus = 1.0 + l - Ieta;

  if ((rint (real (arg_plus)) == arg_plus) && (rint (real (arg_plus)) <= 0.0)) return 0.0;
  if ((rint (real (arg_minus)) == arg_minus) && (rint (real (arg_minus)) <= 0.0)) return 0.0;

  const std::complex<double> log_Gamma_plus = log_Gamma (arg_plus),log_Gamma_minus = log_Gamma
  (arg_minus),log_Gamma_2l_plus_2 = log_Gamma (2.0*l + 2.0);
  const std::complex<double> log_Cl_eta = l*M_LN2 - M_PI_2*eta + 0.5*(log_Gamma_plus +
  log_Gamma_minus) - log_Gamma_2l_plus_2;

  return log_Cl_eta;
}



// Cut constant log for the asymptotic series.
// -------------------------------------------
// The asymptotic series and H[omega] behave differently near the negative real axis.
// Then, if one is in the bad quadrant of H[omega], one has to take into account the cut directly.
// One is in the bad quadrant of H[omega] if Re[z] < 0.0 and sign(Im[z]) = -omega.
//
// H[omega] = [H[omega] from asymptotic function formula] + (1 - exp(2.i.Pi.(i.eta -
  l.omega))).[H[-omega] from asymptotic function formula]
//
//
// The cut constant log is then log [1 - exp(2.i.Pi.(i.eta - l.omega))].
// Its returned imaginary part is not necessarily in ]-Pi:Pi]
//
//
// Variables:
// ----------
// omega : 1 or -1.
// l : orbital angular momentum l.
// eta : Sommerfeld parameter.
// Ieta : i.eta
// l_int,Ieta_int : closest integers to Re[l],Re[i.eta]
// eps :  (Ieta - Ieta_int) - (l.omega - l_int.omega)
// two_I_Pi, two_I_Pi_eps : 2.i.Pi, 2.i.Pi.eps .
// log_cut_constant : returned result.

std::complex<double> log_cut_constant_AS_calc (const int omega,const std::complex<double> &l,const
  std::complex<double> &eta)
{
  const std::complex<double> Ieta(-imag (eta),real (eta));
  const double l_int = rint (real (l)), Ieta_int = rint (real (Ieta));
  const std::complex<double> eps = (Ieta - Ieta_int) - omega*(l - l_int);

  const std::complex<double> two_I_Pi(0,2.0*M_PI),two_I_Pi_eps = two_I_Pi*eps;

  if (real (two_I_Pi_eps) > -0.1)
  {
    const std::complex<double> log_cut_constant = log (expm1 (-two_I_Pi_eps)) + two_I_Pi_eps;

    return log_cut_constant;
  }
  else
```

```
  {
    const std::complex<double> log_cut_constant = log1p (-exp (two_I_Pi_eps));

    return log_cut_constant;
  }
}





// Cut constant log for continued fractions : H[omega] from H[omega, not corrected] case.
// ---------------------------------------------------------------------------------------
// The continued fraction has no cut on the negative real axis, whereas H[omega] has one.
// Then, if one is in the bad quadrant of H[omega], one has to take into account the cut directly.
// One is in the bad quadrant of H[omega] if Re[z] < 0.0 and sign(Im[z]) = -omega.
//
// H[omega] = H[omega, not corrected] - cut_constant.F .
//
// The cut constant is 2i.omega.norm.(exp (2.i.Pi.[l.omega - i.eta]) - 1), and one takes its log.
// The imaginary part of the log is not necessarily in ]-Pi:Pi].
// Norm is 1.0 for normalized wave functions, C(l,eta)^2 for unnormalized wave functions.
//
//
// Variables:
// ----------
// is_it_normalized : true if one wants normalized functions, i.e. the standard normalization,
//                    false if one wants F -> F/C(l,eta) and H+/H-/G -> H+/H-/G.C(l,eta), to avoid overflows for |eta| » 1 and |z| small.
// omega : 1 or -1.
// l : orbital angular momentum l.
// eta : Sommerfeld parameter.
// Ieta : i.eta .
// l_int,Ieta_int : closest integers to Re[l],Re[i.eta]
// eps : (l.omega - l_int.omega) - (Ieta - Ieta_int)
// log_norm : log[C(l,eta)^2] if is_it_normalized is false, 0.0 if it is true.
// two_I_Pi, two_I_Pi_eps : 2.i.Pi, 2.i.Pi.eps .
// log_two_I_omega : log[2.i.omega] = log[2] + i.omega.Pi/2 .
// log_cut_constant : returned result.

std::complex<double> log_cut_constant_CFa_calc (const bool is_it_normalized,const int omega,const std::complex<double> &l,const std::complex<double> &eta)
{
  const std::complex<double> Ieta(-imag (eta),real (eta));
  const double l_int = rint (real (l)), Ieta_int = rint (real (Ieta));
  const std::complex<double> eps = omega*(l - l_int) - (Ieta - Ieta_int);
  const std::complex<double> log_norm = (!is_it_normalized) ? (2.0*log_Cl_eta_calc (l,eta)) : (0.0);
  const std::complex<double> two_I_Pi(0,2.0*M_PI),two_I_Pi_eps = two_I_Pi*eps,log_two_I_omega(M_LN2,omega*M_PI_2);

  if (real (two_I_Pi_eps) < 0.1)
  {
    const std::complex<double> log_cut_constant = log_two_I_omega + log (expm1 (two_I_Pi_eps)) + log_norm;

    return log_cut_constant;
  }
  else
  {
    const std::complex<double> log_cut_constant = log_two_I_omega + log1p (-exp (-two_I_Pi_eps)) + two_I_Pi_eps + log_norm;

    return log_cut_constant;
  }
}




// Cut constant log for continued fractions : H[omega] from H[-omega, not corrected] case.
// ----------------------------------------------------------------------------------------
// The continued fraction has no cut on the negative real axis, whereas H[-omega] has one.
// Then, if one is in the bad quadrant of H[-omega], one has to take into account the cut directly.
// One is in the bad quadrant of H[-omega] if Re[z] < 0.0 and sign(Im[z]) = omega.
//
// H[omega] = H[-omega, not corrected] - cut_constant.F .
//
// The cut constant is 2i.omega.norm.exp (-2.i.Pi.[l.omega + i.eta]), and one takes its log.
// The returned imaginary part of the log is not necessarily in ]-Pi:Pi].
// Norm is 1.0 for normalized wave functions, C(l,eta)^2 for unnormalized wave functions.
//
//
// Variables:
// ----------
// is_it_normalized : true if one wants normalized functions, i.e. the standard normalization,
//                    false if one wants F -> F/C(l,eta) and H+/H-/G -> H+/H-/G.C(l,eta), to avoid overflows for |eta| » 1 and |z| small.
```

```
// omega : 1 or -1.
// l : orbital angular momentum l.
// eta : Sommerfeld parameter = Coulomb_constant.Z.(2.mu/hbar^2)/(2k).
// Ieta : i.eta .
// l_int,Ieta_int : closest integers to Re[l],Re[i.eta]
// eps : (l.omega - l_int.omega) + (Ieta - Ieta_int)
// two_I_Pi : 2.i.Pi .
// log_norm : log[C(l,eta)^2] if is_it_normalized is false, 0.0 if it is true.
// log_two_I_omega : log[2.i.omega] = log[2] + i.omega.Pi/2 .
// log_cut_constant : returned result.

std::complex<double> log_cut_constant_CFb_calc (const bool is_it_normalized,const int omega,const std::complex<double> &l,const std::complex<double> &eta)
{
  const std::complex<double> Ieta(-imag (eta),real (eta));
  const double l_int = rint (real (l)), Ieta_int = rint (real (Ieta));
  const std::complex<double> eps = omega*(l - l_int) + (Ieta - Ieta_int);
  const std::complex<double> log_norm = (!is_it_normalized) ? (2.0*log_Cl_eta_calc (l,eta)) : (0.0);
  const std::complex<double> two_I_Pi(0,2.0*M_PI),log_two_I_omega(M_LN2,omega*M_PI_2),log_cut_constant = log_two_I_omega - two_I_Pi*eps + log_norm;

  return log_cut_constant;
}

// Sin (chi) calculation
// ---------------------
// If 2l is integer, 0.0 is returned as chi is zero.
// If not, one calculates sin (chi) with chi = sigma(l,eta) - sigma(-l-1,eta) - (l+1/2).Pi .
// One uses the stable formula sin (chi) = -(2l+1).C(l,eta).C(-l-1,eta).
//
// Variables
// ---------
// l : orbital angular momentum l.
// eta : Sommerfeld parameter.
// sin_chi : sin (chi)

std::complex<double> sin_chi_calc (const std::complex<double> &l,const std::complex<double> &eta)
{
  if (rint (real (2.0*l)) == 2.0*l) return 0.0;

  const std::complex<double> sin_chi = -(2*l+1)*exp (log_Cl_eta_calc (l,eta) + log_Cl_eta_calc (-l-1,eta));

  return sin_chi;
}


// exp (i.omega.chi) calculation.
// ------------------------------
// One calculates exp (i.omega.chi), with chi = sigma(l,eta) - sigma(l,-eta) - (l+1/2).Pi .
// If 2l is integer, 1.0 is returned as chi is zero.
// If not, one first calculates sin (chi) with the previous routine.
// If |sin (chi)| > 0.5, chi obtained with the formula sigma(l,eta) - sigma(l,-eta) - (l+1/2).Pi is stable so exp[i.omega.chi] follows directly.
// If not, one uses exp[i.omega.chi] = cos (chi) + i.omega.sin (chi), with cos (chi) = sqrt [1 - sin (chi)*sin (chi)].sign[Re[cos (chi)]],
// with chi given by sigma(l,eta) - sigma(l,-eta) - (l+1/2).Pi .
//
// Variables
// ---------
// omega : 1 or -1
// l : orbital angular momentum l.
// eta : Sommerfeld parameter = Coulomb_constant.Z.(2.mu/hbar^2)/(2k).
// I_omega : i.omega
// sin_chi : sin (chi)
// chi : sigma(l,eta) - sigma(l,-eta) - (l+1/2).Pi .
// cos_chi : sign[Re[cos (chi)]].sqrt[1 - [sin(chi)]^2]
// exp_I_omega_chi : exp[i.omega.chi], returned result.

std::complex<double> exp_I_omega_chi_calc (const int omega,const std::complex<double> &l,const std::complex<double> &eta)
{
  if (rint (real (2.0*l)) == 2.0*l) return 1.0;

  const std::complex<double> I_omega(0,omega),sin_chi = sin_chi_calc (l,eta);
  const std::complex<double> chi = sigma_l_calc (l,eta) - sigma_l_calc (-l-1,eta) - (l+0.5)*M_PI;

  if (abs (sin_chi) > 0.5)
  {
    const std::complex<double> exp_I_omega_chi = exp (I_omega*chi);

    return exp_I_omega_chi;
  }
  else
  {
    const std::complex<double> cos_chi = SIGN (real (cos (chi)))*sqrt (1.0 - sin_chi*sin_chi),exp_I_omega_chi = cos_chi + I_omega*sin_chi;
```

```
    return exp_I_omega_chi;
  }
}
```

8.9 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/cwfcomp.cpp File Reference

```
#include "cwfcomp.H"
```

8.10 cwfcomp.cpp

Go to the documentation of this file.

```
#include "cwfcomp.H"

// Calculation of h(omega) = H(omega)'/H(omega) with a continued fraction.
// ------------------------------------------------------------------------
// One calculates the ratio h = H'/H with the continued fraction of the associated hypergeometric confluent function.
// One uses Lentz's method.
// One has : h = [b[0] + a[1]/b[1]+ a[2]/b[2]+ ... a[n]/b[n]+ ...].i.omega/z with :
// b[0] = z - eta, a[n] = (1 + l + i.omega + n-1).(i.omega.eta - l + n-1), b[n] = 2[z - eta] + i.omega.n .
//
// If the number of iterations reaches 100000 and |z| > 0.5, the convergence is too slow. One is probably very close to the imaginary axis.
// If l=0 and |z| <= 0.5, the direct integration is still done as it is stable.
// If 1+l+/-i.eta is negative integer, it has to be done otherwise it is too long even if |z| is not exceedingly small.
// One first considers Re[z] >= 0.
// One takes the starting point z0 = 2 + i.(Im[z] + 2.sign[Im[z]]) (0.6 + i.sign[Re[z]].0.6.sign[Im[z]] if |z| <= 0.5).
// Then, one calculates H[omega],H[omega]' at the starting point, and one integrates numerically H[omega] until z.
// h(omega)(z) is then H[omega](z)'/H[omega](z).
// If Re[z] < 0, one calculates H[-omega],H[-omega]' with l, eta -> -eta, and z -> -z.
// One uses the Coulomb wave functions class cwf_minus_eta_ptr defined with l and -eta.
// The ratio h(omega,l,eta,z) is then equal to -h(-omega,l,-eta,-z).
// To avoid infinite loops, continued_fraction_h must not be used in this integration. So, the starting point is chosen so |H[+/-]| is very likely
// to increase in modulus. Then, one always integrates forward. Forward integration is enforced putting
// is_H_dir_int_naive to true. It is put to false again at the end of the calculation.
//
// Variables:
// ----------
// z : variable of the Coulomb wave function.
// omega : 1 for the outgoing wave function ratio H+'/H+, -1 for the incoming wave function ratio H-'/H-.
// large,small : 1E50,1E-50. They are used in the case of vanishing denominators or numerators.
// I_omega,I_omega_eta,two_I_omega,z_minus_eta,two_z_minus_eta : i.omega,i.omega.eta, 2.i.omega, z-eta, 2(z-eta).
// a,c : 1 + l + i.omega.eta, i.omega.eta - l.
// b0,Cn,Dn,an,bn,Delta_n : variables used in the Lentz method.
// n,nm1 : index of a[n] and b[n], n-1
// bn_plus_an_Dn : bn + an.Dn. Dn = 1/[bn + an.Dn] or 1E50 if bn + an.Dn is zero.
// bn_plus_an_over_Cn : bn + an/Cn. Cn = bn + an/Dn or 1E-50 if bn + an/Dn is zero.
// hn : value of the continuous fraction during the iteration process.
// test : test of convergence of hn.
// cwf : reference to *this if Re[z] >= 0, reference to cwf_minus_eta_ptr if Re[z] < 0
//       cwf_minus_eta_ptr is allocated first if it is zero, in the case of Re[z] < 0 .
//       It is used to integrate the Coulomb wave functions with l,eta for Re[z] >= 0 or l,-eta for Re[z] < 0.
// z00 : 2 + i.sign[Re[z]].(Im[z] + 2.sign[Im[z]]).
// z01 : 0.6 + i.0.6.sign[Re[z]].sign[Im[z]].
// abs_z : |z|
// z_start,F_start,dF_start : starting point of the direct integration, F(l,+/-eta,z_start),F'(l,+/-eta,z_start)
// H,dH : Coulomb wave function and derivative in l,eta,omega,z if Re[z] > 0, in l,-eta,-omega,-z if Re[z] < 0.
// h : value of H(omega)'/H(omega).
// debut_cwf,F_debut_cwf,dF_debut_cwf : values stored in cwf put back in cwf at the end of direct integration as they change values.
```

```

std::complex<double> Coulomb_wave_functions::continued_fraction_h (const std::complex<double> &z,const int omega)
{
  const double small = 1E-50,large = 1E50,abs_z = abs (z);
  const std::complex<double> I_omega(0.0,omega),two_I_omega(0.0,2.0*omega),I_omega_eta = I_omega*eta;
  const std::complex<double> a = I_omega_eta + l + 1.0,c = I_omega_eta - l,z_minus_eta = z - eta,two_z_minus_eta = 2.0*z_minus_eta;

  std::complex<double> b0 = z_minus_eta,hn = (b0 != 0.0) ? (b0) : (1E-50), Cn = hn, Dn = 0.0;
  int n = 1;
  double test;
  do
  {
    const int nm1 = n-1;
    const std::complex<double> an = (a + nm1)*(c + nm1),bn = two_z_minus_eta + n*two_I_omega,bn_plus_an_Dn = bn + an*Dn,bn_plus_an_over_Cn = bn + an/Cn;

    Dn = (bn_plus_an_Dn != 0.0) ? (1.0/bn_plus_an_Dn) : (large);
    Cn = (bn_plus_an_over_Cn != 0.0) ? (bn_plus_an_over_Cn) : (small);

    const std::complex<double> Delta_n = Dn*Cn;
    hn *= Delta_n;
    test = inf_norm (1.0 - Delta_n);

    if ((n++ > 100000) && ((l == 0.0) || (abs_z > 0.5) || neg_int_omega_one || neg_int_omega_minus_one))
    {
      if ((real (z) < 0.0) && (cwf_minus_eta_ptr == 0)) cwf_minus_eta_ptr = new class Coulomb_wave_functions (is_it_normalized,l,-eta);
      class Coulomb_wave_functions &cwf = (real (z) < 0.0) ? (*cwf_minus_eta_ptr) : (*this);

      const std::complex<double> z00(2.0,SIGN (real (z))*(imag (z) + 2.0*SIGN (imag (z)))),z01(0.6,0.6*SIGN (real (z))*SIGN (imag (z)));
      const std::complex<double> z_start = (abs_z > 0.5) ? (z00) : (z01),debut_cwf = cwf.debut,F_debut_cwf = cwf.F_debut,dF_debut_cwf = cwf.dF_debut;
      std::complex<double> F_start,dF_start,H,dH;
      cwf.F_dF (z_start,F_start,dF_start);
      cwf.is_H_dir_int_naive = true, cwf.H_dH_direct_integration (SIGN (real (z))*omega,SIGN (real (z))*z,H,dH), cwf.is_H_dir_int_naive = false;
      cwf.debut = debut_cwf, cwf.F_debut = F_debut_cwf, cwf.dF_debut = dF_debut_cwf;
      return (SIGN (real (z))*dH/H);
    }
  }
  while (test > 1E-15);

  const std::complex<double> h = hn*I_omega/z;
  return h;
}










// Calculation of H(omega) and dH(omega)/dz (scaled) with asymptotic series
// ----------------------------------------------------------------------------
// H[omega](z) = exp[i.omega.[z - eta.log[2z] - l.Pi/2 + sigma(l,eta)]].S[omega](z) for Re[z] >= 0.
// H[omega]'(z) = exp[i.omega.[z - eta.log[2z] - l.Pi/2 + sigma(l,eta)]].[S[omega]'(z) + i.omega.(1 - eta/z).S[omega](z)] for Re[z] >= 0.
//
// S[omega](z) is the asymptotic series and S[omega]'(z) its derivative calculated in asymptotic_series.
// If they did not converge, one leaves the routine.
//
// The negative cut is taken into account if log [cut_constant_AS] is finite, i.e. cut_constant_AS is not exactly zero :
//
// If Re[z] < 0.0 and omega.Im[z] < 0.0 :
// ---------------------------------------
// H[omega] = H[omega][ASd] + cut_constant_AS_plus.H[-omega][ASd].
// H[omega][ASd] is given by directly by the asymptotic series.
// H[-omega][ASd] is given by the asymptotic series, which gives the good result H[-omega] in this case as one is not in its bad quadrant.
//
// The function is scaled, so one returns : H[omega](z).exp[-i.omega.[z - eta.log[2z]]] and H[omega]'(z).exp[-i.omega.[z - eta.log[2z]]] .
//
// In the case of overflows or underflows, one uses logs.
//
//
// Variables :
// -----------
```

```
// omega : 1 if one calculates H+(z) and H+'(z), -1 if one calculates H-(z) and H-'(z).
// one_over_z : 1/z.
// z : variable of the Coulomb wave function.
// H_scaled,dH_scaled : H[omega](z).exp(-i.omega.[z - eta.log[2z]]) and H[omega]'(z).exp(-iomega.[z - eta.log[2z]]).
// is_it_successful : true if the asymptotic expansions converged, false it not
// sum,dsum : {S[omega](z),S[-omega](z)} and {S[omega]'(z),S[-omega]'(z)}
// I_omega,two_I_omega : i.omega, 2i.omega.
// I_omega_one_minus_eta_over_z : i.omega.(1 - eta/z).
// phase_shift : i.omega.[-l.Pi/2 + sigma(l,eta)]
// exp_phase_shift : exp (i.omega.(-l.Pi/2 + sigma(l,eta))).
// log_cut_constant_AS : log_cut_constant_AS_plus if omega = 1, log_cut_constant_AS_minus if omega =-1.
//                       If log_cut_constant_AS is not finite (i.e. cut_constant_AS = 0 if it is defined), there is no branch cut to consider.
// factor : exp (log_cut_constant_AS - 2.i.omega.(z - eta.log(2z)) - phase_shift).

void Coulomb_wave_functions::asymptotic_expansion_H_dH_scaled (const int omega,const std::complex<double> &one_over_z,
                                                               std::complex<double> &H_scaled,std::complex<double> &dH_scaled,bool &is_it_successful)
{
  std::complex<double> sum[2],dsum[2];

  asymptotic_series (omega,one_over_z,sum,dsum,is_it_successful);
  if (!is_it_successful) return;

  const std::complex<double> I_omega(0,omega),two_I_omega(0,2*omega),I_omega_one_minus_eta_over_z = I_omega*(1.0 - eta*one_over_z);

  const std::complex<double> phase_shift = I_omega*(sigma_l - l*M_PI_2),exp_phase_shift = exp (phase_shift);

  H_scaled = sum[0]*exp_phase_shift;
  dH_scaled = (dsum[0] + sum[0]*I_omega_one_minus_eta_over_z)*exp_phase_shift;

  const std::complex<double> log_cut_constant_AS = (omega == 1) ? (log_cut_constant_AS_plus) : (log_cut_constant_AS_minus);

  if (one_over_z != 0.0)
  {
    const std::complex<double> z = 1.0/one_over_z;

    if (isfinite (log_cut_constant_AS) && (real (z) < 0.0) && (SIGN (imag (z)) == -omega))
    {
      const std::complex<double> factor = exp (-two_I_omega*(z - eta*(M_LN2 + log (z))) - phase_shift + log_cut_constant_AS);

      H_scaled += sum[1]*factor;
      dH_scaled += (dsum[1] - sum[1]*I_omega_one_minus_eta_over_z)*factor;
    }
  }

  if (!is_it_normalized)
  {
    if ((Cl_eta == 0.0) || (!isfinite (Cl_eta)))
      H_scaled = exp (log_Cl_eta + log (H_scaled)),dH_scaled = exp (log_Cl_eta + log (dH_scaled));
    else
      H_scaled *= Cl_eta,dH_scaled *= Cl_eta;
  }
}






// Calculation of H[omega](z) and H[omega]'(z) by direct integration.
// -------------------------------------------------------------------
// To calculate H[omega](z) and H'[omega](z), one integrates numerically H[omega]"(z) = [l(l+1)/z^2 + 2.eta/z - 1].H[omega](z)
// starting from debut,H_debut = H[omega](debut) and dH_debut = H'[omega](debut).
// There is no branch cut problem as Re[z] >= 0.
// The starting point comes for the regular function from the stored values debut, F_debut and dF_debut.
// If debut = 0, one puts debut = debut_omega = z/|2z| and calculates F(debut) and F'(debut) with power series.
// Then, the starting point {debut_omega,H[omega](debut),H'[omega](debut)} is calculated
// from {debut,F(debut),F'(debut)} and the continued fraction h[omega](debut).
// The first order expansions method is used if one is very close to the real axes of l,eta and z (Re[z] > 0).
// The step of the integration is (z - debut)/N_num, with N_num = [|z - debut|/min (0.1,10.turning_point)] + 1.
// The value of min (0.1,10/turning_point) gives a smaller step when turning_point increases,
// as calculations become there more difficult as |H[omega]| typically varies faster in this case.
// The intermediates points are called z_aft. They go from debut to z,
// and (debut_omega,H_debut,dH_debut) is put to {z_aft,H[omega](z_aft),H'[omega](z_aft)} at each step.
```

```
// If |H[omega]| increases along the path, the integration is stable.
// If is_H_dir_int_naive is true, one has to integrate forward, as this integration is used to calculate the continued fraction.
// If not, and if |H[omega]| decreases,
// one integrates backwards from z_aft to debut_omega with the knowledge of h[omega](z_aft) = H'[omega](z_aft)/H[omega](z_aft).
// H'[omega](z_aft)/H[omega](z_aft) is given by the continued fraction formula.
// One then obtains by direct integration C.H[omega](debut) and C.H'[omega](debut).
// Knowing H_debut, one deduces H[omega](z) = 1/C and H'[omega](z) = f(z_aft).H[omega](z).
// Increase or decrease is known using the Taylor expansion of H[omega] near debut_omega in z up to second order.
//
// If H(z_aft) is not finite, one stops the integration.
//
// Variables
// ---------
// omega : 1 for H+,H+', -1 for H-,H-'.
// z : variable of the Coulomb wave function.
// H,dH : wave function H[omega] and derivative H'[omega] to calculate.
// x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
// debut_omega, H_debut,dH_debut : starting point or the integration. debut_omega = debut at the beginning.
// ODE : reference to pointer ODE_ptr, which performs direct integration from one point to another.
// step_abs : length of the integration step from z to debut. It is min (0.1,10/turning_point)
// N_num : number of integrations to do from debut to z. It is |z - debut|/step_abs + 1.
// step_num : std::complex step from debut to z. It is (z - debut)/N_num .
// ll_plus_one,two_eta : l(l+1), 2.eta. They are used in the Taylor expansion test.
// z_aft : next point in which H[omega] is calculated by direct integration . It is z - i*step_num, with i from N_num-1 to 0.
// one_over_debut,log_H_debut_der,d2H_debut_over_H_debut: 1/debut, H'[omega](debut)/H[omega](debut), H[omega]''(debut)/H[omega](debut).
//                                                        They are used in the Taylor expansion test.
// h : continued fraction h[omega] = H[omega]'/H[omega] in z.
// H_debut_not_normed,dH_debut_not_normed : unnormed regular wave function and derivative in debut_omega,
//                                           after integration from the starting point {z_aft, 1.0, H'[omega](z_aft)/H[omega](z_aft)}.
//                                           Then, H[omega](z_aft) =  H_debut/H_debut_not_normed and H'[omega](z_aft) = h[omega].H[omega](z_aft) .

void Coulomb_wave_functions::H_dH_direct_integration (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH)
{
  const double x = real (z),y = imag (z),l_r = real (l),l_i = imag (l),eta_r = real (eta),eta_i = imag (eta);

  if (debut == 0.0)
  {
    debut = 0.5*z/abs (z);
    F_dF_power_series (debut,F_debut,dF_debut);
  }

  std::complex<double> debut_omega = debut,H_debut,dH_debut;

  if (((y != 0.0) || (eta_i != 0.0) || (l_i != 0.0))
      && (std::abs (y) < sqrt_precision*std::min (1.0,x)) && (std::abs (eta_i) < sqrt_precision) && (std::abs (l_i) < sqrt_precision)
      && (!neg_int_omega_one && !neg_int_omega_minus_one))
    first_order_expansions (omega,debut_omega,H_debut,dH_debut);
  else
    H_dH_with_F_dF_and_CF (omega,debut_omega,H_debut,dH_debut);

  const class ODE_integration &ODE = *ODE_ptr;
  const double step_abs = std::min(0.1,10.0/turning_point);
  const unsigned int N_num = static_cast<unsigned int> ((abs (z-debut_omega)/step_abs) + 1);
  const std::complex<double> step_num = (z - debut_omega)/static_cast<double> (N_num),ll_plus_one = l*(l+1.0),two_eta = 2.0*eta;

  for (unsigned int i = N_num-1 ; i <= N_num ; i--)
  {
    const std::complex<double> z_aft = z - i*step_num,one_over_debut = 1.0/debut,log_H_debut_der = dH_debut/H_debut;
    const std::complex<double> d2H_debut_over_H_debut = (ll_plus_one*one_over_debut + two_eta)*one_over_debut - 1.0;

    if (is_H_dir_int_naive)
      ODE (debut_omega,H_debut,dH_debut,z_aft,H,dH);
    else if (abs (1.0 + step_num*(log_H_debut_der + 0.5*step_num*d2H_debut_over_H_debut)) < 1.0)
    {
      const std::complex<double> h = continued_fraction_h (z_aft,omega);
      std::complex<double> H_debut_not_normed,dH_debut_not_normed;

      ODE (z_aft,1.0,h,debut_omega,H_debut_not_normed,dH_debut_not_normed);
      H = H_debut/H_debut_not_normed;
      dH = h*H;
    }
    else ODE (debut_omega,H_debut,dH_debut,z_aft,H,dH);
```

```
    if (!isfinite (H) || !isfinite (dH)) std::cout«"Numerical failure encountered in H_dH_direct_integration."«std::endl,exit (1);

    debut_omega = z_aft,H_debut = H,dH_debut = dH;
  }
}







// Calculation of H[omega](z), H'[omega](z) with the first order expansion method.
// ---------------------------------------------------------------------------------
// When imaginary parts of l,eta,z are much smaller than their real parts but not all zero, with Re[z] > 0,
// one has to separate the calculation of the real and imaginary parts of H[omega] and H'[omega], as they can differ by tens of orders of magnitude.
//
// For that, one expands F(z),G(z),F'(z),G'(z) in first order in y, eta_i and l_i in first_order_expansions.
//
// Then, H[omega](z) = G(z) + i.omega.norm.F(z) and H'[omega](z) = G'(z) + i.omega.norm.F'(z),
// with norm = 1 if the wave functions are normalized and C(l,eta)^2 if not.
// One uses logs if norm underflows or overflows.
//
// Variables:
// ----------
// omega : 1 if one calculates H+(z),H+'(z), -1 if one calculates H-(z),H-'(z).
// z : variable of the Coulomb wave function.
// H,dH : wave function H[omega] and derivative H'[omega] to calculate.
// F,dF,G,dG : regular and irregular wave functions and derivatives in z, calculated with first order expansions.
// I_omega : i.omega
// norm_functions,log_norm : 1 if the wave functions are normalized and C(l,eta)^2 if not, its log.


void Coulomb_wave_functions::H_dH_from_first_order_expansions (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH)
{
  std::complex<double> F,dF,G,dG;
  first_order_expansions (true,z,F,dF);
  first_order_expansions (false,z,G,dG);

  const std::complex<double> I_omega(0,omega),norm_functions = (!is_it_normalized) ? (Cl_eta*Cl_eta) : (1.0);

  if ((norm_functions == 0.0) || (!isfinite (norm_functions)))
  {
    const std::complex<double> log_norm = (!is_it_normalized) ? (2.0*log_Cl_eta) : (0.0);

    H = G + I_omega*exp (log (F) + log_norm);
    dH = dG + I_omega*exp (log (dF) + log_norm);
  }
  else
  {
    H = G + I_omega*norm_functions*F;
    dH = dG + I_omega*norm_functions*dF;
  }
}














// Calculation of H(omega) and H(omega)' with F(z), F'(z) and continued fractions.
// ---------------------------------------------------------------------------------
// If 1+l-i.omega.eta is negative integer, one uses H[omega] = 1/F/(f - h[omega]) as it is the only available solution
// linearly independent of F.
// If 1+l+i.omega.eta is negative integer, H[omega] is proportional to F so the values F and F' are arbitrarily chosen for H[omega] and H'[omega].
//
// If not, one calculates h[sign(Im[z])](z), and one has f = F'(z)/F(z).
// One chooses sign(Im[z]), as h converges fastest for z in this region.
// If |f - h[sign(Im[z])]|oo >= 1, F and H[sign] are numerically linearly independent so the continued fraction is meaningful.
```

```
// Then, h[-sign(Im[z])](z) is not needed and is put to f, the worst value it can have. If not, h[-sign(Im[z])](z) is needed and calculated.
// Then, h[omega] and h[-omega] take their values from h[sign(Im[z])]| and h[-sign(Im[z])]|.
//
// If |f - h[omega]|oo > |f - h[-omega]|oo, one uses the continued fraction h[omega].
// If Re[z] > 0 or sign(Im[z]) = omega (good quadrants), H[omega] = 1/F/(f - h[omega]) and H'[omega] = h[omega].H[omega].
// If not, one has to take the branch cut into account, as one is in the bad quadrant of H[omega] :
// H[omega] = 1/F/(f - h[omega]) - cut_constant.F and H'[omega] = h[omega].H[omega] - cut_constant.F'.
// cut_constant is cut_constant_CFa_plus if omega = 1, and cut_constant_CFa_minus if omega = -1.
// If log [cut_constant] is not finite, it means that cut_constant is exactly zero if it is defined, so that there is no branch cut to consider.
//
// If |f - h[omega]|oo < |f - h[-omega]|oo, one uses the continued fraction h[-omega],
// calculating H[omega] and H'[omega] using H[omega] = H[-omega] + constant.F .
// constant is 2.i.omega.norm if Re[z] > 0 or sign(Im[z]) = -omega (good quadrants), and cut_constant if not.
// cut_constant is cut_constant_CFb_plus if omega = 1, cut_constant_CFb_minus if omega = -1.
// norm is 1 if is_it_normalized is true, C(l,eta)^2 if not.
//
// If cut_constant underflows or overflows, one uses logs of F,F' and log_cut_constant for the calculation.
//
// Variables:
// ----------
// omega : 1 if one calculates H+(z) and H+'(z), -1 if one calculates H-(z) and H-'(z).
// z : variable of the Coulomb wave function.
// F,dF: regular Coulomb wave function and derivative.
// H,dH : H+(z) and H+'(z) if omega=1, H-(z) and H-'(z) if omega=-1.
// x,y : Re[z], Im[z].
// f,two_I_omega : ratio F'(z)/F(z), 2.i.omega .
// h_sign,h_minus_sign : continuous fractions h[-SIGN[y]] and h[SIGN[y]]. See before for their calculations
// h_omega,h_minus_omega : continuous fraction h[omega] and h[-omega].
// cut_constant_CFa,log_cut_constant_CFa : cut constant for H[omega], so in its bad quadrant H[omega](z) =  1/F/(f - h[omega]) - cut_constant.F and its log.
//                                         cut_constant is cut_constant_CFa_plus if omega = 1, cut_constant_CFa_minus if omega = -1.
//                                         If the log is not finite, cut_constant has to be strictly zero if it is defined so there is no branch cut to consider.
// H_minus_omega,dH_minus_omega : H-(z) and H-'(z) if omega=1, H+(z) and H+'(z) if omega=-1 in good quadrants.
//                               In bad quadrants, they are not equal to the previous functions, but combined with branch cut formulas, one calculates them.
// cut_constant_CFb,constant,norm_functions : cut_constant is cut_constant_CFb_plus if omega = 1, cut_constant_CFb_minus if omega = -1.
//                                            constant is 2.i.omega.norm_functions in good quadrants, cut_constant in bad quadrants.
//                                            norm_functions is 1 for normalized functions, C(l,eta)^2 for unnormalized wave functions.
//                                            One has : H[-omega](z) =  1/F/(f - h[-omega]) + constant.F in all quadrants.
// log_cut_constant_CFb,log_constant,log_norm : logs of previous values.
// log_two_I_omega : log[2.i.omega] = log[2] + i.omega.Pi/2 .

void Coulomb_wave_functions::H_dH_with_F_dF_and_CF (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH)
{
  std::complex<double> F,dF;
  F_dF (z,F,dF);

  const double x = real (z),y = imag (z);
  const std::complex<double> f = dF/F,two_I_omega(0.0,2.0*omega);

  if (((neg_int_omega_one && (omega == -1))) || ((neg_int_omega_minus_one && (omega == 1))))
  {
    const std::complex<double> h_omega = continued_fraction_h (z,omega);
    H = 1.0/(F*(f - h_omega)),dH =  H*h_omega;
  }
  else if (neg_int_omega_one || neg_int_omega_minus_one)
    H = F,dH = dF;
  else
  {
    const std::complex<double> h_sign = continued_fraction_h (z,SIGN (y)),h_minus_sign = (abs (f - h_sign) < 1.0) ? (continued_fraction_h (z,-SIGN (y))) : (f);
    const std::complex<double> h_omega = (omega == SIGN (y)) ? (h_sign) : (h_minus_sign),h_minus_omega = (omega == SIGN (y)) ? (h_minus_sign) : (h_sign);

    if (inf_norm (f - h_omega) > inf_norm (f - h_minus_omega))
    {
      H = 1.0/(F*(f - h_omega)),dH =  H*h_omega;

      const std::complex<double> log_cut_constant_CFa = (omega == 1) ? (log_cut_constant_CFa_plus) : (log_cut_constant_CFa_minus);

      if ((isfinite (log_cut_constant_CFa)) && (x < 0.0) && (SIGN (y) == -omega))
```

```
      {
    const std::complex<double> cut_constant_CFa = (omega == 1) ? (cut_constant_CFa_plus) : (cut_constant_CFa_minus);

    if ((cut_constant_CFa == 0.0) || (!isfinite (cut_constant_CFa)))
      H -= exp (log_cut_constant_CFa + log (F)),dH -= exp (log_cut_constant_CFa + log (dF));
    else
      H -= cut_constant_CFa*F,dH -= cut_constant_CFa*dF;
      }
    }
    else
    {
      const std::complex<double> H_minus_omega = 1.0/(F*(f - h_minus_omega)),dH_minus_omega = H_minus_omega*h_minus_omega;
      const std::complex<double> norm_functions = (!is_it_normalized) ? (Cl_eta*Cl_eta) : (1.0);
      const std::complex<double> cut_constant_CFb = (omega == 1) ? (cut_constant_CFb_plus) : (cut_constant_CFb_minus);
      const std::complex<double> constant = ((x < 0.0) && (SIGN (y) == omega)) ? (cut_constant_CFb) : (two_I_omega*norm_functions);

      if ((constant == 0.0) || (!isfinite (constant)))
      {
    const std::complex<double> log_norm = (!is_it_normalized) ? (2.0*log_Cl_eta) : (0.0),log_two_I_omega(M_LN2,omega*M_PI_2);
    const std::complex<double> log_cut_constant_CFb = (omega == 1) ? (log_cut_constant_CFb_plus) : (log_cut_constant_CFb_minus);
    const std::complex<double> log_constant = ((x < 0.0) && (SIGN (y) == omega)) ? (log_cut_constant_CFb) : (log_two_I_omega + log_norm);

    H = exp (log_constant + log (F)) + H_minus_omega,dH = exp (log_constant + log (dF)) + dH_minus_omega;
      }
      else H = constant*F + H_minus_omega,dH = constant*dF + dH_minus_omega;
    }
  }
}


// Calculation of H[omega],H'[omega] for std::complex l with the expansion formula.
// ---------------------------------------------------------------------------
// When 2l is non-integer, one can expand H[omega] with F[l,eta,z] and F[-l-1,eta,z].
// H[omega] = (exp[i.omega.chi].F - Fp)/sin (chi), H'[omega] = (exp[i.omega.chi].F' - Fp')/sin (chi) if wave functions are normalized,
// H[omega] = (exp[i.omega.chi].F.C(l,eta)^2/sin (chi) + Fp/(2l+1), H'[omega] = (exp[i.omega.chi].F'.C(l,eta)^2/sin (chi) + Fp'/(2l+1) if not.
// chi is sigma(l,eta) - sigma(-l-1,eta) - (l+0.5).Pi, and Fp is F(-l-1,eta).
// Fp is calculated using a class Coulomb_wave_functions with parameters -l-1 and eta.
// To avoid numerical imprecisions, sin (chi) is calculated with the stable formula -(2l+1).C(l,eta).C(-l-1,eta).
// The validity of this expansion is checked with the wronskian of F and Fp, which must be correct up to precision :
// F'.Fp - Fp'.F = sin (chi) if wave functions are normalized, Fp'.F - F'.Fp = 2l + 1 if not.
// If the wronskians are numerically correct, one does the calculation and is_it_successful is put to true.
// If not, one puts is_it_successful to false and quits the routine.
// If C(l,eta)^2 underflows or overflows, one uses logs of F,F' and C(l,eta)^2 for the calculation.
//
// Variables
// ---------
// omega : 1 if one calculates H+(z) and H+'(z), -1 if one calculates H-(z) and H-'(z).
// z : variable of the Coulomb wave function.
// F,dF: regular Coulomb wave function and derivative in l,eta and z.
// H,dH : H+(z) and H+'(z) if omega=1, H-(z) and H-'(z) if omega=-1.
// is_it_successful : false if the wronskian between F(l,eta,z) and F(-l-1,eta,z) is not equal to zero up to precision, true if not.
// Fp,dFp : F(-l-1,eta,z), F'(-l-1,eta,z).
// exp_I_omega_chi : exp[i.omega.chi]
// one_over_2lp1 : 1/(2l + 1)
// Cl_eta_2,exp_I_omega_chi_over_sin_chi : C(l,eta)^2, exp[i.omega.chi]/sin (chi)
// F_Cl_eta_2, dF_Cl_eta_2 : F(z).C(l,eta)^2, F'(z).C(l,eta)^2.

void Coulomb_wave_functions::H_dH_with_expansion (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH,bool &is_it_successful)
{
  if (cwf_lp_ptr == 0) cwf_lp_ptr = new class Coulomb_wave_functions (is_it_normalized,-l-1,eta);

  std::complex<double> F,dF,Fp,dFp;
```

```
  F_dF (z,F,dF);
  cwf_lp_ptr->F_dF (z,Fp,dFp);

  const std::complex<double> exp_I_omega_chi = (omega == 1) ? (exp_I_chi) : (exp_minus_I_chi);

  if (is_it_normalized)
  {
    if (inf_norm ((dFp*F - dF*Fp)*one_over_sin_chi - 1.0) < precision)
    {
      H = (exp_I_omega_chi*F - Fp)*one_over_sin_chi;
      dH = (exp_I_omega_chi*dF - dFp)*one_over_sin_chi;
    }
    else {is_it_successful = false; return;}
  }
  else
  {
    const std::complex<double> one_over_2lp1 = 1.0/(2*l+1);

    if (inf_norm ((dF*Fp - dFp*F)*one_over_2lp1 - 1.0) < precision)
    {
      const std::complex<double> Cl_eta_2 = Cl_eta*Cl_eta,exp_I_omega_chi_over_sin_chi = exp_I_omega_chi*one_over_sin_chi;
      const std::complex<double> F_Cl_eta_2 = ((Cl_eta_2 == 0.0) || (!isfinite (Cl_eta_2))) ? (exp (2.0*log_Cl_eta + log (F))) : (Cl_eta_2*F);
      const std::complex<double> dF_Cl_eta_2 = ((Cl_eta_2 == 0.0) || (!isfinite (Cl_eta_2))) ? (exp (2.0*log_Cl_eta + log (dF))) : (Cl_eta_2*dF);

      H = exp_I_omega_chi_over_sin_chi*F_Cl_eta_2 + Fp*one_over_2lp1;
      dH = exp_I_omega_chi_over_sin_chi*dF_Cl_eta_2 + dFp*one_over_2lp1;
    }
    else {is_it_successful = false; return;}
  }

  is_it_successful = true;
}



// Calculation of F and F' by power series.
// ----------------------------------------
// It is used only when |z| <= 0.5, to avoid numerical inaccuracies.
//
// F(z) = norm.z^(l+1).\sum a[n], n in [0:+oo[, where :
// a[0] = 1.0.
// a[1] = z.eta/(l+1).
// a[n] = (2.z.eta.a[n-1] - a[n-2].(z^2))/(n.(n+2l+1)),n >= 2.
//
// The z = 0 case is treated first. It is defined only for Re[l] > 0 or l = 0. The program aborts for other cases.
// Norm is C(l,eta) if one uses normalized wave functions, 1.0 if not.
// So, one multiplies by C(l,eta) at the end if one uses normalized functions.
// If there is overflow or underflow for C(l,eta) in this last case, one uses logs of F,F' and C(l,eta) for the calculation.
//
// Variables:
// ----------
// z : variable of the Coulomb wave function.
// F,dF : regular wave function and derivative
// z_square,z_two_eta,z_pow_l_plus_one : z^2, 2.z.eta, z^{l+1}.
// n : index of the power series term. It begins at two.
// an,an_minus_one,an_minus_two : a[n], a[n-1], a[n-2].
// The test of convergence is |(n+l-1).a[n-2]|oo + |(n+l).a[n-1]|oo, as one of the two can be zero even before convergence.

void Coulomb_wave_functions::F_dF_power_series (const std::complex<double> &z,std::complex<double> &F,std::complex<double> &dF)
{
  if (z == 0.0)
  {
    if (l == 0) F = 0.0,dF = (is_it_normalized) ? (Cl_eta) : (1.0);
    else if (real (l) > 0) F = dF = 0.0;
    else std::cout«"F(z=0) and/or F'(z=0) are undefined."«std::endl, abort ();
  }
  else
  {
    const std::complex<double> z_square = z*z,z_two_eta = 2.0*eta*z;

    int n = 2;
    std::complex<double> an_minus_two = 1.0,an_minus_one = z*eta/(l+1.0);

    F = an_minus_two + an_minus_one;
    dF = (l+1.0)*an_minus_two + (l+2.0)*an_minus_one;

```

```
    while (inf_norm (an_minus_two*(n+l-1.0)) + inf_norm (an_minus_one*(n+l)) > precision)
    {
      const std::complex<double> an = (z_two_eta*an_minus_one - an_minus_two*z_square)/(n*(n + l + l + 1.0));

      F += an;
      dF += an*(n + l + 1.0);

      n++;
      an_minus_two = an_minus_one;
      an_minus_one = an;
    }

    const std::complex<double> z_pow_l_plus_one = pow (z,l+1.0);
    F *= z_pow_l_plus_one;
    dF *= z_pow_l_plus_one/z;

    if (is_it_normalized)
    {
      if ((Cl_eta == 0.0) || (!isfinite (Cl_eta))) F = exp (log_Cl_eta + log (F)),dF = exp (log_Cl_eta + log (dF));
      else F *= Cl_eta,dF *= Cl_eta;
    }
  }
}







// Calculation of f = F'/F with a continued fraction.
// ---------------------------------------------------
// One calculates the ratio f = F'/F with the continued fraction of the associated hypergeometric confluent function.
// One uses Lentz's method.
// One has : f = [b[0] + a[1]/b[1]+ a[2]/b[2]+ ... a[n]/b[n]+ ...]/z with :
// b[0] = l + 1 + i.omega.z, a[n] = -2.i.omega.[1 + l + i.omega.eta] + (n-1).[-2.i.omega.z], b[n] = 2l + 2 + 2.i.omega.z + n-1.
// omega is 1 or -1, and theoretically the result is the same.
// If they are not equal numerically, omega = sign[-Im [z]] gives usually the best result.
// If 1+l+i.omega.eta is a negative integer, f[omega] is finite and must be used.
//
// Variables:
// ----------
// z : variable of the Coulomb wave function.
// omega : 1 or -1. Both values should be tried to test stability.
// large,small : 1E50,1E-50. They are used in the case of vanishing denominators or numerators.
// I_omega,a,b : i.omega, 1 + l + i.omega.eta, 2l + 2.
// minus_two_I_omega_z,minus_two_I_omega_a_z,b_plus_two_I_omega_z : -2.i.omega.z, -2.i.omega.a.z, b + 2.i.omega.z
// b0,Cn,Dn,an,bn,Delta_n : variables used in the Lentz method.
// n,nm1 : index of a[n] and b[n], n-1
// bn_plus_an_Dn : bn + an.Dn. Dn = 1/[bn + an.Dn] or large if bn + an.Dn is zero.
// bn_plus_an_over_Cn : bn + an/Cn. Cn = bn + an/Dn or small if bn + an/Dn is zero.
// fn : value of the continuous fraction during the iteration process.
// test : test of convergence of fn.
// f : value of F'(z)/F(z).

std::complex<double> Coulomb_wave_functions::continued_fraction_f (const std::complex<double> &z,const int omega)
{
  const double small = 1E-50,large = 1E50;

  const std::complex<double> I_omega(0.0,omega);
  const std::complex<double> a = I_omega*eta + l + 1.0,b = 2*l + 2;
  const std::complex<double> minus_two_I_omega_z = -2.0*I_omega*z,minus_two_I_omega_a_z = minus_two_I_omega_z*a,b_plus_two_I_omega_z = b - minus_two_I_omega_z;

  const std::complex<double> b0 = l + 1.0 + I_omega*z;
  std::complex<double> fn = (b0 != 0.0) ? (b0) : (small), Cn = fn, Dn = 0.0;

  int n = 1;
  double test;
  do
  {
    const int nm1 = n-1;
    const std::complex<double> an = minus_two_I_omega_a_z + nm1*minus_two_I_omega_z;
    const std::complex<double> bn = b_plus_two_I_omega_z + nm1;

    const std::complex<double> bn_plus_an_Dn = bn + an*Dn,bn_plus_an_over_Cn = bn + an/Cn;

    Dn = (bn_plus_an_Dn != 0.0) ? (1.0/bn_plus_an_Dn) : (large);
    Cn = (bn_plus_an_over_Cn != 0.0) ? (bn_plus_an_over_Cn) : (small);

    const std::complex<double> Delta_n = Dn*Cn;
```

```
    fn *= Delta_n;
    test = inf_norm (1.0 - Delta_n);
    n++;
  }
  while (test > 1E-15);

  const std::complex<double> f = fn/z;

  return f;
}








// Calculation of F(z) and F'(z) with asymptotic series
// ------------------------------------------------------
// F(z) = [H+(z) - H-(z)]/[2.i.norm].
// F'(z) = [H+'(z) - H-'(z)]/[2.i.norm].
// In this routine, Re[z] >= 0, so there is no branch cut problem.
//
// H+(z) = exp[i.[z - eta.log[2z] - l.Pi/2 + sigma(l,eta)]].S+(z) .
// H-(z) = exp[-i.[z - eta.log[2z] - l.Pi/2 + sigma(l,eta)]].S-(z) .
//
// H+'(z) = exp[i.[z - eta.log[2z] - l.Pi/2 + sigma(l,eta)]].[S+'(z) + S+(z).i.(1 - eta/z)] .
// H-'(z) = exp[-i.[z - eta.log[2z] - l.Pi/2 + sigma(l,eta)]].[S-'(z) - S-(z).i.(1 - eta/z)] .
//
//
// S+ and S- and derivatives are calculated in asymptotic_series. If is_it_successful is true, the
series are meaningful. If not, one leaves the routine.
// Norm is C(l,eta) if one uses normalized wave functions, 1.0 if not.
// If there is overflow or underflow for C(l,eta) in this last case, one uses logs of F,F' and
C(l,eta) for the calculation.
//
// Variables :
// -----------
// z : variable of the Coulomb wave function.
// one_over_z : 1/z
// F,dF : regular wave function and derivative to calculate.
// is_it_successful : true if the calculation converged, i.e. the series are good up to precision and
the wronskian of H+,H- up to precision, false if not.
// sum,dsum : asymptotic series. sum[0] = S+(z), sum[1] = S-(z), dsum[0] = S+'(z), dsum[1] = S-'(z).
// I,one_over_two_I : i,1/[2i].
// I_one_minus_eta_over_z : i*(1 - eta/z).
// exp_phase_shift_plus : exp (i*(z - eta.log(2z) - l.Pi/2 + sigma(l,eta)))).
// exp_phase_shift_minus : exp (-i*(z - eta.log(2z) - l.Pi/2 + sigma(l,eta)))).
// H_plus,dH_plus,H_minus,dH_minus : exp (+/-i*(z - eta.log(2z) - l.Pi/2 + sigma(l,eta))).S(+/-)(z)
and derivatives.

void Coulomb_wave_functions::asymptotic_expansion_F_dF (const std::complex<double>
&z,std::complex<double> &F,std::complex<double> &dF,bool &is_it_successful)
{
  std::complex<double> sum[2],dsum[2];

  const std::complex<double> one_over_z = 1.0/z;

  asymptotic_series (1,one_over_z,sum,dsum,is_it_successful);
  if (!is_it_successful) return;

  const std::complex<double> I(0,1),one_over_two_I(0.0,-0.5),I_one_minus_eta_over_z = I*(1.0 -
eta*one_over_z);

  const std::complex<double> exp_phase_shift_plus = exp (I*(z - eta*(M_LN2 + log (z)) - l*M_PI_2 +
sigma_l));
  const std::complex<double> exp_phase_shift_minus = 1.0/exp_phase_shift_plus;

  const std::complex<double> H_plus = sum[0]*exp_phase_shift_plus,dH_plus = (dsum[0] +
sum[0]*I_one_minus_eta_over_z)*exp_phase_shift_plus;
  const std::complex<double> H_minus = sum[1]*exp_phase_shift_minus,dH_minus = (dsum[1] -
sum[1]*I_one_minus_eta_over_z)*exp_phase_shift_minus;

  F = (H_plus - H_minus)*one_over_two_I;
  dF = (dH_plus - dH_minus)*one_over_two_I;

  if (!is_it_normalized)
  {
    if ((Cl_eta == 0.0) || (!isfinite (Cl_eta))) F = exp (log (F) - log_Cl_eta),dF = exp (log (dF) -
log_Cl_eta);
    else F /= Cl_eta,dF /= Cl_eta;
  }


}

```

```
// Calculation of F(z) and F'(z) by direct integration.
// ----------------------------------------------------
// To calculate F(z) and F'(z), one integrates numerically F"(z) = [l(l+1)/z^2 + 2.eta/z - 1].F(z)
// starting from debut, F_debut = F(debut) and dF_debut = F'(debut).
// One always has Re[z] >= 0.0, so there is no branch cut problem.
// If z = debut, the previous values are returned.
// The starting point come from the stored values debut, F_debut and dF_debut.
// If debut = 0, one puts debut = z/|2z| and calculates F(debut) and F'(debut) with power series.
// The step of the integration is (z - debut)/N_num, with N_num = [|z - debut|/min (0.1,10.turning_point)] + 1.
// The value of min (0.1,10/turning_point) gives a smaller step when turning_point increases,
// as calculations become there more difficult as |F| typically varies faster in this case.
// The intermediates points are called z_aft. They go from debut to z, and (debut,F_debut,dF_debut) is put to {z_aft,F(z_aft),F'(z_aft)} at each step.
// If |F| increases along the path, the integration is stable.
// If it decreases, and if z_aft decreases in modulus or one does not integrate with constant argument (i.e. theta constant in z = |z|.exp[i.theta])
// for Re[l] > -1, one reintegrates F(z) from debut = z/|2z|, as integration is usually stable at constant argument for Re[l] > -1.
// Increase or decrease is known using the Taylor expansion of F near debut in z up to second order.
// If one integrates with constant argument and |F| decreases,
// one integrates backwards from z_aft to debut with the knowledge of f(z_aft) = F'(z_aft)/F(z_aft).
// F'(z_aft)/F(z_aft) is given by the continued fraction formula.
// One then obtains by direct integration C.F(debut) and C.F'(debut).
// Knowing F_debut, one deduces F(z) = 1/C and F'(z) = f(z_aft).F(z).
// If 1+l+i.omega.eta is a negative integer, f[omega] is finite and is used.
// Otherwise, f(z_aft) is calculated with omega = 1 and -1. If they are equal up to precision, f(omega) is correct and used.
// omega is chosen so Re[-2.i.omega.z] < 0,
// for which the anomalous convergence phenomenon of Gautschi of f is the smallest (W. Gautschi, Math. Comp. Vol. 31 p.994).
// If not, but |norm.F| < 0.1, one still has to use f[omega], as it is probably correct as F is the minimal solution, and also one has no other way to calculate F.
// If |norm.F| > 0.1 in this case, one stops the procedure and F will be calculated from H+ and H-, given by direct integration and continued fraction formulae.
// In this case, is_it_successful is put to false, and otherwise the integration worked and it is put to true.
// Norm is 1.0 if one uses normalized wave functions, C(l,eta) if not.
//
// If F(z_aft) is not finite, one stops the integration and is_it_successful is put to false.
//
// Variables
// ---------
// z : variable of the Coulomb wave function.
// F,dF : regular wave function and derivative to calculate.
// is_it_successful : false is the calculation is unstable,
// i.e. |F| > 0.1 decreasing on the integration path and f(omega) is not equal to f(-omega) up to precision, true if not.
// ODE : reference to pointer ODE_ptr, which performs direct integration from one point to another.
// step_abs : length of the integration step from z to debut. It is min (0.1,10/turning_point)
// N_num : number of integrations to do from debut to z. It is |z - debut|/step_abs + 1.
// step_num : std::complex step from debut to z. It is (z - debut)/N_num .
// ll_plus_one,two_eta : l(l+1), 2.eta. They are used in the Taylor expansion test.
// z_aft : next point in which F is calculated by direct integration . It is z - i*step_num, with i from N_num-1 to 0.
// one_over_debut,log_F_debut_der,d2F_debut_over_F_debut: 1/debut, F'(debut)/F(debut), F"(debut)/F(debut). They are used in the Taylor expansion test.
// ratio : debut/z. It is used to know if one integrates with constant argument or if |z_aft| increases.
//          If not and |F| increases and Re[l] > -1, one reintegrates from z/|2z|.
// f_omega,f_minus_omega : continued fractions F'/F with omega = sign(-Im[z_aft]) and -omega.
// fp,fm : continued fractions F'/F with omega = 1 or -1 used in the case of 1+l +/- i.eta negative integer, as continued fractions are finite in this case.
// norm_functions : C(l,eta)^2 for unnormalized functions, 1 for normalized functions.
// F_debut_not_normed,dF_debut_not_normed : unnormed regular wave function and derivative in debut,
//                                           after integration from the starting point {z_aft, 1.0, F'(z_aft)/F(z_aft)}.
//                                           Then, F(z_aft) =  F_debut/F_debut_not_normed and F'(z_aft) = f[omega].F .

void Coulomb_wave_functions::F_dF_direct_integration (const std::complex<double> &z,std::complex<double> &F,std::complex<double> &dF,bool &is_it_successful)
{
  if (z == debut) {F = F_debut,dF = dF_debut,is_it_successful = true; return;}
  if (debut == 0.0) debut = 0.5*z/abs (z),F_dF_power_series (debut,F_debut,dF_debut);

  const class ODE_integration &ODE = *ODE_ptr;
  const double step_abs = std::min(0.1,10.0/turning_point);
  const unsigned int N_num = static_cast<unsigned int> (rint (abs (z-debut)/step_abs) + 1);
  const std::complex<double> step_num = (z - debut)/static_cast<double> (N_num),ll_plus_one = l*(l+1.0),two_eta = 2.0*eta;
  const std::complex<double> norm_functions = (!is_it_normalized) ? (Cl_eta*Cl_eta) : (1.0);
```

```
  for (unsigned int i = N_num-1 ; i <= N_num ; i--)
  {
    const std::complex<double> z_aft = z - i*step_num,one_over_debut = 1.0/debut,log_F_debut_der = dF_debut/F_debut;
    const std::complex<double> d2F_debut_over_F_debut = (ll_plus_one*one_over_debut + two_eta)*one_over_debut - 1.0;

    if (abs (1.0 + step_num*(log_F_debut_der + 0.5*step_num*d2F_debut_over_F_debut)) < 1.0)
    {
      const std::complex<double> ratio = debut/z;
      if ((real (l) > -1.0) && ((std::abs (imag (ratio)) > precision) || (real (ratio) > 1.0)))
    {debut = 0.0, F_dF_direct_integration (z,F,dF,is_it_successful); return;}

      std::complex<double> F_debut_not_normed,dF_debut_not_normed;

      if (neg_int_omega_one)
      {
    const std::complex<double> fp = continued_fraction_f (z_aft,1);
    ODE (z_aft,1.0,fp,debut,F_debut_not_normed,dF_debut_not_normed);
    F = F_debut/F_debut_not_normed;
    dF = fp*F;
      }
      else if (neg_int_omega_minus_one)
      {
    const std::complex<double> fm = continued_fraction_f (z_aft,-1);
    ODE (z_aft,1.0,fm,debut,F_debut_not_normed,dF_debut_not_normed);
    F = F_debut/F_debut_not_normed;
    dF = fm*F;
      }
      else
      {
    const std::complex<double> f_omega = continued_fraction_f (z_aft,SIGN(-imag(z_aft))),f_minus_omega = continued_fraction_f (z_aft,-SIGN(-imag(z_aft)));

    if ((abs (F*norm_functions) > 0.1) && (abs (f_minus_omega/f_omega - 1.0) > precision)) {is_it_successful = false; return;}

    ODE (z_aft,1.0,f_omega,debut,F_debut_not_normed,dF_debut_not_normed);
    F = F_debut/F_debut_not_normed;
    dF = f_omega*F;
      }
    }
    else ODE (debut,F_debut,dF_debut,z_aft,F,dF);

    debut = z_aft,F_debut = F,dF_debut = dF;

    if (!isfinite (F) || !isfinite (dF)) std::cout«"Numerical failure encountered in F_dF_direct_integration."«std::endl,exit (1);
  }
  is_it_successful = true;
}



// Regular wave function and derivative from symmetry relations.
// ------------------------------------------------------------
// If |z| > 0.5 and Re[z] < 0, one calculates F(l,eta,z),F'(l,eta,z) from F(l,-eta,-z),F'(l,-eta,-z) using the formulas :
// F(l,eta,z) = -F(l,-eta,-z).exp[-Pi.(eta-i.l)], F'(l,eta,z) = F'(l,-eta,-z).exp[-Pi.(eta-i.l)] if arg (z) > 0 and is_it_normalized is true,
// F(l,eta,z) = -F(l,-eta,-z).exp[-Pi.(eta+i.l)], F'(l,eta,z) = F'(l,-eta,-z).exp[-Pi.(eta+i.l)] if arg (z) <= 0 and is_it_normalized is true,
// F(l,eta,z) = -F(l,-eta,-z).exp[i.Pi.l)], F'(l,eta,z) = F'(l,-eta,-z).exp[i.Pi.l)] if arg (z) > 0 and is_it_normalized is false,
// F(l,eta,z) = -F(l,-eta,-z).exp[-i.Pi.l)], F'(l,eta,z) = F'(l,-eta,-z).exp[-i.Pi.l)] if arg (z) <= 0 and is_it_normalized is false.
//
// F(l,-eta,-z) is calculated using the class cwf_minus_eta_ptr defined with (l,-eta).
// The debut point of the class cwf_minus_eta_ptr is initialized with {debut,F_debut,dF_debut} and previous relations
// if cwf_minus_eta_ptr->debut and -debut are different and debut non zero.
//
// If the normalization constant underflows or overflows, one uses logs.
//
// Variables
// ---------
// z : variable of the Coulomb wave function.
// F,dF : regular wave function and derivative to calculate.
// arg_debut, arg_z : argument angles of debut and z.
// sym_constant_debut,log_sym_constant_debut : constant C so F(l,-eta,-debut) = C.F(l,eta,debut), its log.
// sym_constant,log_sym_constant : constant C so F(l,eta,z) = C.F(l,-eta,-z), its log.

void Coulomb_wave_functions::F_dF_with_symmetry_relations (const std::complex<double> &z,std::complex<double> &F,std::complex<double> &dF)
{
```

```
  if (cwf_minus_eta_ptr == 0) cwf_minus_eta_ptr = new class Coulomb_wave_functions (is_it_normalized,l,-eta);

  if ((debut != 0.0) && (cwf_minus_eta_ptr->debut != -debut))
  {
    const double arg_debut = arg (debut);
    const std::complex<double> sym_constant_debut = (arg_debut <= 0.0) ? (1.0/sym_constant_arg_neg) : (1.0/sym_constant_arg_pos);

    cwf_minus_eta_ptr->debut = -debut;

    if ((sym_constant_debut == 0.0) || (!isfinite (sym_constant_debut)))
    {
      const std::complex<double> log_sym_constant_debut = (arg (debut) <= 0.0) ? (-log_sym_constant_arg_neg) : (-log_sym_constant_arg_pos);

      cwf_minus_eta_ptr->F_debut = exp (log_sym_constant_debut + log (F_debut));
      cwf_minus_eta_ptr->dF_debut = -exp (log_sym_constant_debut + log (dF_debut));
    }
    else
    {
      cwf_minus_eta_ptr->F_debut = F_debut*sym_constant_debut;
      cwf_minus_eta_ptr->dF_debut = -dF_debut*sym_constant_debut;
    }
  }

  const double arg_z = arg (z);
  const std::complex<double> sym_constant = (arg_z <= 0.0) ? (sym_constant_arg_neg) : (sym_constant_arg_pos);

  cwf_minus_eta_ptr->F_dF (-z,F,dF);

  if ((sym_constant == 0.0) || (!isfinite (sym_constant)))
  {
    const std::complex<double> log_sym_constant = (arg_z <= 0.0) ? (log_sym_constant_arg_neg) : (log_sym_constant_arg_pos);

    F = exp (log_sym_constant + log (F));
    dF = -exp (log_sym_constant + log (dF));
  }
  else
  {
    F *= sym_constant;
    dF *= -sym_constant;
  }
}





// Calculation of the asymptotic series
// ------------------------------------
// Asymptotic expansion:
// S(+/-)(z) = 1.0+\sum_(n=1)^N a[n] with a[n+1]=a[n].[n.[n+1+/-2i.eta]+i.eta.(i.eta+/-1)-l(l+1)]/[+/-2i.(n+1)]/z, n >= 0 and a[0] = 1.
// This expansion diverges : it is only useful with the smallest term summation method.
// The test of convergence is max(|a[n]|oo,|n.a[n]/z|oo), so the largest norm of the term of series of function and derivative.
// Practically, one stops when test < precision (it worked) or when test is not finite (it failed).
// After that, one tests the series with the wronskian of H[omega] and H[-omega].
//
// Variables :
// -----------
// z is the variable of the Coulomb wave function.
// omega : 1 if one calculates H+(z) and H+'(z), -1 if one calculates H-(z) and H-'(z).
// one_over_z : 1/z.
// sum : sum[0] is the series in H(omega), sum[1] the one in H(-omega).
// dsum : dsum[0] is the series in H'(omega), dsum[1] the one in H'(-omega).
// is_it_successful : true if the series converged and |wronskian - 2.i.omega|oo is smaller than precision, false if not.
// test : test of convergence of the asymptotic series. It is max(|a[n]|oo,|n.a[n]/z|oo).
// sign : if i=0, it is omega, if i=1, it is -omega.
// Ieta,two_I_eta_sign : i.eta, 2.i.eta if sign = 1, -2.i.eta if sign = -1.
// Ieta_Ieta_plus_sign_minus_ll_plus_one : i.eta(i.eta+1) - l(l+1) if sign=1, i.eta(i.eta-1) - l(l+1) if sign=-1.
// n,an_sign : index of the series, a[n+1]
// n_plus_one,sign_one_over_two_I_n_plus_one : n+1, 1/(2.i.(n+1)) if sign=1, -1/(2.i.(n+1)) if sign=-1.
// sum_term, dsum_term : an_sign, (n+1).an_sign/z
// two_I_omega : 2.i.omega .

void Coulomb_wave_functions::asymptotic_series (const int omega,const std::complex<double> &one_over_z,
                                  std::complex<double> sum[],std::complex<double> dsum[],bool &is_it_successful)
{
  sum[0] = sum[1] = 1.0;
```

```
  dsum[0] = dsum[1] = 0.0;

  double test;

  for (unsigned int i = 0 ; i <= 1 ; i++)
  {
    const int sign = (i == 0) ? (omega) : (-omega);
    const std::complex<double> Ieta(-imag (eta),real (eta)),two_I_eta_sign = 2.0*sign*Ieta,Ieta_Ieta_plus_sign_minus_ll_plus_one = Ieta*(Ieta + sign) - l*(l+1.0);

    int n = 0;
    std::complex<double> an_sign = 1.0;

    do
    {
      const double n_plus_one = n + 1.0;
      const std::complex<double> sign_one_over_two_I_n_plus_one(0,-sign*0.5/n_plus_one);

      an_sign *= one_over_z*(n*(n_plus_one + two_I_eta_sign) + Ieta_Ieta_plus_sign_minus_ll_plus_one)*sign_one_over_two_I_n_plus_one;

      const std::complex<double> sum_term = an_sign,dsum_term = n_plus_one*an_sign*one_over_z;

      sum[i] += sum_term;
      dsum[i] -= dsum_term;

      test = std::max (inf_norm (sum_term),inf_norm (dsum_term));
      n++;
    }
    while ((test > precision) && (isfinite (test)));

    if (!isfinite (test)) {is_it_successful = false; return;}
  }

  const std::complex<double> two_I_omega(0.0,2.0*omega);
  is_it_successful = (inf_norm (sum[1]*dsum[0] - sum[0]*dsum[1] + two_I_omega*(1.0 - eta*one_over_z)*sum[0]*sum[1] - two_I_omega) < precision);
}









// Numerical partial derivatives according to l or eta.
// ----------------------------------------------------
// One calculates here the partial derivatives according to l or eta
// of the Coulomb wave functions F(x) or G(x) and of their derivatives with x F'(x) or G'(x),
// with l_r=Re[l] and eta_r=Re[eta].
// One considers here the parameters l_r and eta_r with the argument x.
// For this, one uses the standard formula : df/d_chi(x) = [f(x,chi+eps) - f(x,chi-eps)]/[2.eps],
// with chi = l_r or eta_r and eps = prec_first_order_expansion*chi. f is either F, F', G or G'.
// One then calculates F or G, F' or G' with chi+eps and chi-eps.
// With them, one obtains dF/d_chi(x),dF'/d_chi(x) or dG/d_chi(x),dG'/d_chi(x), with chi = l or eta.
// All functions are real, so double values are returned, taking the real part of std::complex variables.
//
// Variables
// ---------
// is_it_regular : true if one calculates derivatives of F(z),F'(z), false if one calculates derivatives of G(z),G'(z).
// is_it_eta : true if one calculates the partial derivatives according to eta,
//             false if one calculates the partial derivatives according to l.
// cwf_plus,cwf_minus : references on class Coulomb_wave_functions with parameters l+eps,eta and l-eps,eta or l,eta+eps and l,eta-eps.
//                      If one calculates the partial derivatives according to eta, they are *cwf_real_eta_plus_ptr and *cwf_real_eta_minus_ptr.
//                      If one calculates the partial derivatives according to l, they are *cwf_real_l_plus_ptr and *cwf_real_l_minus_ptr.
// x : Re[z], z the argument of the wave function.
// d_chi_Bx,d_chi_dBx : partial derivatives of F and F' in l_r,eta_r,x according to chi = l or eta if is_it_regular is true, or G and G' if not.
// l_r,eta_r,chi_r : Re[l],Re[eta], Re[eta] is is_it_eta is true, Re[l] if not.
// chi_r_plus,chi_r_minus : chi_r+eps, chi_r-eps.
// A_plus,dA_plus,A_minus,dA_minus : F(x,chi+eps),F'(x,chi+eps),F(x,chi-eps),F'(x,chi-eps) if is_it_regular is true,
//                                   H+(x,chi+eps),H+'(x,chi+eps),H+(x,chi-eps),H+'(x,chi-eps) if is_it_regular is false.
// B_plus,B_minus,dB_plus,dB_minus : F(x,chi+eps),F'(x,chi+eps),F(x,chi-eps),F'(x,chi-eps) if is_it_regular is true,
//                                   G(x,chi+eps),G'(x,chi+eps),G(x,chi-eps),G'(x,chi-eps) if is_it_regular is false.

void Coulomb_wave_functions::partial_derivatives (const bool is_it_regular,const bool is_it_eta,const
```

```
double x,double &d_chi_Bx,double &d_chi_dBx)
{
  const double l_r = real (l),eta_r = real (eta),chi_r = (is_it_eta) ? (eta_r) : (l_r);
  const double chi_r_plus = (chi_r != 0.0) ? (chi_r*(1.0 + prec_first_order_expansion)) : (prec_first_order_expansion);
  const double chi_r_minus = (chi_r != 0.0) ? (chi_r*(1.0 - prec_first_order_expansion)) : (-prec_first_order_expansion);

  if (is_it_eta)
  {
    if (cwf_real_eta_plus_ptr == 0) cwf_real_eta_plus_ptr = new class Coulomb_wave_functions (is_it_normalized,l_r,chi_r_plus);
    if (cwf_real_eta_minus_ptr == 0) cwf_real_eta_minus_ptr = new class Coulomb_wave_functions (is_it_normalized,l_r,chi_r_minus);
  }
  else
  {
    if (cwf_real_l_plus_ptr == 0) cwf_real_l_plus_ptr = new class Coulomb_wave_functions (is_it_normalized,chi_r_plus,eta_r);
    if (cwf_real_l_minus_ptr == 0) cwf_real_l_minus_ptr = new class Coulomb_wave_functions (is_it_normalized,chi_r_minus,eta_r);
  }

  class Coulomb_wave_functions &cwf_plus = (is_it_eta) ? (*cwf_real_eta_plus_ptr) : (*cwf_real_l_plus_ptr);
  class Coulomb_wave_functions &cwf_minus = (is_it_eta) ? (*cwf_real_eta_minus_ptr) : (*cwf_real_l_minus_ptr);

  std::complex<double> A_plus,dA_plus,A_minus,dA_minus;

  if (is_it_regular)
  {
    cwf_plus.F_dF (x,A_plus,dA_plus);
    cwf_minus.F_dF (x,A_minus,dA_minus);
  }
  else
  {
    cwf_plus.H_dH (1,x,A_plus,dA_plus);
    cwf_minus.H_dH (1,x,A_minus,dA_minus);
  }

  const double B_plus = real (A_plus),B_minus = real (A_minus),dB_plus = real (dA_plus),dB_minus = real (dA_minus);

  d_chi_Bx = (B_plus - B_minus)/(chi_r_plus - chi_r_minus);
  d_chi_dBx = (dB_plus - dB_minus)/(chi_r_plus - chi_r_minus);
}









// Calculation of F(z),F'(z) or G(z),G'(z) with the first order expansion method.
// ----------------------------------------------------------------------------
// When imaginary parts of l,eta,z are much smaller than their real parts but not all zero, with Re[z] > 0,
// one has to separate the calculation of the real and imaginary parts of (F,G)(z) and (F',G')(z), as they can differ by tens of orders of magnitude.
// Re[z] < 0 is not considered, as G(z) is std::complex with Im[z] = Im[l] = Im[eta] = 0.
// So, with z = x+i.y, eta = eta_r + i.eta_i and l = l_r+i.l_i, one calculates (F,G)[l_r,eta_r](x) and (F,G)'[l_r,eta_r](x).
//
// One considers here the parameters l_r and eta_r with the argument x, and the parameters l and eta with the argument z.
//
// One then has F(x),F'(x) from usual relations and one takes their real parts.
// One also has H+(x),H+'(x) from usual relations and G(x) = Re[H+(x)],G'(x) = Re[H+'(x)].
//
// After that, one expands (F,G)(z) and (F',G')(z) in first order in y, eta_i and l_i and one has, up to y^2,eta_i^2 and l_i^2:
//
// (F,G)(z) = (F,G)(x) + i.y.d(F,G)/dx[omega](x) + i.eta_i.d(F,G)/d[eta](x) + i.l_i.d(F,G)/dl(x).
// (F',G')[omega](z) = (F',G')[omega](x) + i.y.(F",G")[omega](x) + i.eta_i.d(F',G')/d[eta](x) + i.l_i.d(F',G')/dl(x).
//
// Variables:
// ----------
// is_it_regular : true if one calculates F(z),F'(z), false if one calculates G(z),G'(z).
// z : variable of the Coulomb wave function.
// B,dB : F(z),F'(z) if is_it_regular is true, G(z),G'(z) if it is false.
// x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
// A_x,dA_x : F(x),F'(x) if is_it_regular is true, H+(x),H+'(x) if it is false.
// Bx,dBx : F(x),F'(x) if is_it_regular is true, G(x),G'(x) if it is false.
// cwf_real : class Coulomb_wave_functions of parameters l_r and eta_r.
```

```
// one_over_x,two_eta_r : 1/x, 2.eta_r.
// d_l_Bx,d_l_dBx,d_eta_Bx,d_eta_dBx : partial derivatives according to l and eta of B(x) and B'(x)
//                                     They are initialized at zero, and not calculated if they are multiplied by zero after.
// Coulomb_eq_x,d2Bx : l_r(l_r+1)/[x^2] + 2.eta_r/x - 1, B"(x) = [l_r(l_r+1)/[x^2] + 2.eta_r/x - 1].B(x),


void Coulomb_wave_functions::first_order_expansions (const bool is_it_regular,const std::complex<double> &z,std::complex<double> &B,std::complex<double> &dB)
{
  const double x = real (z),y = imag (z),l_r = real (l),l_i = imag (l),eta_r = real (eta),eta_i = imag (eta);

  if (cwf_real_ptr == 0) cwf_real_ptr = new class Coulomb_wave_functions (is_it_normalized,l_r,eta_r);
  class Coulomb_wave_functions &cwf_real = *cwf_real_ptr;

  std::complex<double> A_x,dA_x;
  if (is_it_regular)
    cwf_real.F_dF (x,A_x,dA_x);
  else
    cwf_real.H_dH (1,x,A_x,dA_x);

  const double Bx = real (A_x),dBx = real (dA_x);

  double d_l_Bx = 0.0,d_l_dBx = 0.0,d_eta_Bx = 0.0,d_eta_dBx = 0.0;
  if (l_i != 0.0) partial_derivatives (is_it_regular,false,x,d_l_Bx,d_l_dBx);
  if (eta_i != 0.0) partial_derivatives (is_it_regular,true,x,d_eta_Bx,d_eta_dBx);

  const double one_over_x = 1.0/x,Coulomb_eq_x = (l_r*(l_r+1)*one_over_x + 2.0*eta_r)*one_over_x - 1.0;
  const double d2Bx = Coulomb_eq_x*Bx;

  B = std::complex<double> (Bx,y*dBx + l_i*d_l_Bx + eta_i*d_eta_Bx);
  dB = std::complex<double> (dBx,y*d2Bx + l_i*d_l_dBx + eta_i*d_eta_dBx);
}














// Regular wave function and derivative.
// -------------------------------------
// One calculates F(z) and F'(z), so F(z) ~ z^{l+1} for z -> 0 if is_it_normalized is false,
//                                   F(z) ~ C(l,eta).z^{l+1} for z -> 0 if is_it_normalized is true.
// If |z| <= 0.5, one uses the power series.
//
// If |z| > 0.5 and Re[z] < 0, one calculates F(z) from F[l,-eta,-z] with F_dF_with_symmetry_relations.
//
// If |z| > 0.5 and Re[z] >= 0, and 1+l+/-i.eta no negative integer, one first tries the asymptotic series formula.
// If it failed, one integrates directly the regular Coulomb wave function with F_dF_direct_integration.
// If 1+l+/-i.eta is a negative integer, this is the only available method besides power series so it is accepted.
// If 1+l+/-i.eta is no negative integer but it failed again,
// one calculates the Coulomb wave function H[omega] with H_dH_direct_integration and omega = sign(Im[z]).
// omega is chosen so one cannot encounter the branch cut of h[omega].
// H[-omega] is calculated from H[omega] and continued fractions h[omega] and h[-omega].
// One then has F(z) = (H[omega] - H[-omega])/(2.i.omega.norm), F'(z) = (H'[omega] - H'[-omega])/(2.i.omega.norm),
// with norm = 1 if the wave functions are normalized and C(l,eta)^2 if not.
// The formula is stable as one uses this case only when |F(z)| > 0.1 .
//
// One takes only real parts if l, eta and z are real.
// At the end of the function, one puts {debut,F_debut,dF_debut} equal to {z,F,dF}.
//
// Variables:
// ----------
// z : variable of the Coulomb wave function.
// F,dF : regular Coulomb wave function in z and derivative in z.
// x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
// is_it_successful : true if the asymptotic expansions converges, false it not (after asymptotic_expansion_F_dF).
//                    true if the direct integration worked, false it not (after F_dF_direct_integration).
// omega : sign[Im[z]]. H[omega](z) is calculated if the direct integration of F failed.
```

```
// two_I_omega, two_I_term : 2.i.omega, 2.i.omega if is_it_normalized is true, 2.i.omega.Cl_eta^2 if not.
// h_omega, h_minus_omega : log derivatives of H[omega](z) and H[-omega](z) calculated with continued fractions.
// H_omega,dH_omega,H_minus_omega,dH_minus_omega : H[omega](z), H[-omega](z) and derivatives

void Coulomb_wave_functions::F_dF (const std::complex<double> &z,std::complex<double> &F,std::complex<double> &dF)
{
  const double x = real (z),y = imag (z),l_r = real (l),l_i = imag (l),eta_r = real (eta),eta_i = imag (eta);

  if (((y != 0.0) || (eta_i != 0.0) || (l_i != 0.0))
      && (std::abs (y) < sqrt_precision*std::min (1.0,x)) && (std::abs (eta_i) < sqrt_precision) && (std::abs (l_i) < sqrt_precision)
      && (!neg_int_omega_one && !neg_int_omega_minus_one))
    first_order_expansions (true,z,F,dF);
  else if (abs (z) <= 0.5)
    F_dF_power_series (z,F,dF);
  else
  {
    if (real (z) < 0.0)
      F_dF_with_symmetry_relations (z,F,dF);
    else
    {
      bool is_it_successful = false;
      if (!neg_int_omega_one && !neg_int_omega_minus_one) asymptotic_expansion_F_dF (z,F,dF,is_it_successful);

      if (!is_it_successful)
      {
    F_dF_direct_integration (z,F,dF,is_it_successful);

    if (!neg_int_omega_one && !neg_int_omega_minus_one && !is_it_successful)
    {
      const int omega = SIGN(imag (z));
      const std::complex<double> two_I_omega(0.0,2.0*omega),two_I_term = (is_it_normalized) ? (two_I_omega) : (two_I_omega*Cl_eta*Cl_eta);
      const std::complex<double> h_omega = continued_fraction_h (z,omega),h_minus_omega = continued_fraction_h (z,-omega),one_over_two_I_term = 1.0/two_I_term;

      std::complex<double> H_omega,dH_omega;
      H_dH_direct_integration (omega,z,H_omega,dH_omega);

      const std::complex<double> H_minus_omega = two_I_term/((h_omega - h_minus_omega)*H_omega),dH_minus_omega = h_minus_omega*H_minus_omega;
      F = (H_omega - H_minus_omega)*one_over_two_I_term;
      dF = (dH_omega - dH_minus_omega)*one_over_two_I_term;
    }}}
  }

  if (!isfinite (F) || !isfinite (dF)) std::cout«"Numerical failure encountered in F_dF."«std::endl,exit (1);

  if ((y == 0.0) && (eta_i == 0.0) && (l_i == 0.0)) F = real (F),dF = real (dF);
  debut = z,F_debut = F,dF_debut = dF;
}










// Calculation of G(z) and G'(z).
// ------------------------------
// One calculates the irregular Coulomb wave function from H+ and F.
// If 1+l+i.omega.eta is a negative integer, G is by definition H[-omega].
// If not, one uses the formulas :
// G(z) = H+(z) - i.F(z), G'(z) = H+'(z) - i.F'(z) if is_it_normalized is true,
// G(z) = H+(z) - i.Cl_eta^2.F(z), G'(z) = H+'(z) - i.Cl_eta^2.F'(z) if not.
// There is no numerical inaccuracy as G is never a minimal solution.
// One takes only real parts if l, eta and z are real.
//
// Variables :
// -----------
// z : variable of the Coulomb wave function.
// G,dG : irregular Coulomb wave functions and derivatives.
// x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
// H_plus,dH_plus : H+(z) and H+'(z).
// F,dF : regular Coulomb wave functions and derivatives.
// I_Cl_eta_square : i.C(l,eta)^2.

void Coulomb_wave_functions::G_dG (const std::complex<double> &z,std::complex<double> &G,std::complex<double> &dG)
```

```
{
  const double x = real (z),y = imag (z),l_r = real (l),l_i = imag (l),eta_r = real (eta),eta_i = imag (eta);

  if (((y != 0.0) || (eta_i != 0.0) || (l_i != 0.0))
      && (std::abs (y) < sqrt_precision*std::min (1.0,x)) && (std::abs (eta_i) < sqrt_precision) && (std::abs (l_i) < sqrt_precision)
      && (!neg_int_omega_one && !neg_int_omega_minus_one))
    first_order_expansions (false,z,G,dG);
  else
  {
    if (neg_int_omega_one)
      H_dH (-1,z,G,dG);
    else if (neg_int_omega_minus_one)
      H_dH (1,z,G,dG);
    else
    {
      std::complex<double> F,dF;
      F_dF (z,F,dF);

      std::complex<double> H_plus,dH_plus;
      H_dH (1,z,H_plus,dH_plus);

      const std::complex<double> I(0.0,1.0);

      if (is_it_normalized)
    G = H_plus - I*F,dG = dH_plus - I*dF;
      else
      {
    const std::complex<double> I_Cl_eta_square = I*Cl_eta*Cl_eta;

    if ((I_Cl_eta_square == 0.0) || (!isfinite (I_Cl_eta_square)))
      G = H_plus - I*exp (2.0*log_Cl_eta + log (F)),dG = dH_plus - I*exp (2.0*log_Cl_eta + log (dF));
    else
      G = H_plus - I_Cl_eta_square*F,dG = dH_plus - I_Cl_eta_square*dF;
      }

      if ((y == 0.0) && (eta_i == 0.0) && (l_i == 0.0)) G = real (G),dG = real (dG);
    }
  }
}




// Calculation of H[omega](z) and H'[omega](z).
// ---------------------------------------------
// One first tries the asymptotic expansion formula if 1+l+/-i.eta is no negative integer.
// On uses logs if the unscaling factor underflows or overflows.
// If it failed, and imaginary parts of l,eta,z are much smaller than their real parts but not all zero, with Re[z] > 0,
// one calculates H[omega](z) and H'[omega](z) with the first order expansion method.
// If one is not in this case, one calculates F(z)and F'(z).
// If |Im[l]| > 1 and |z| <= 1, one tries the expansion formula with F(l,eta,z) and F(-l-1,eta,z).
// If not, or if it failed, one uses the continued fraction formula.
// If l,eta and z are real, one rewrites H[omega] as H[omega] = Re[H[omega]] + i.omega.norm.Re[F] to avoid numerical inaccuracies for Im[H[omega]].
// norm is 1 if is_it_normalized is true, C(l,eta)^2 if not.
//
//
// Variables :
// -----------
// omega : 1 if one calculates H+(z) and H+'(z), -1 if one calculates H-(z) and H-'(z).
// z : variable of the Coulomb wave function.
// H,dH : H+(z) and H+'(z) if omega=1, H-(z) and H-'(z) if omega=-1.
// is_it_successful : true if the asymptotic expansions converges, false it not (in asymptotic_expansion_H_dH_scaled).
//                    true if the expansion formula with F(l,eta,z) and F(-l-1,eta,z) worked, false it not (in H_dH_with_expansion).
// H_scaled,dH_scaled : H[omega](z).exp[-i.omega.[z - eta.log[2z]]) and H'[omega](z).exp[-i.omega.z.[z - eta.log[2z]]) given by the asymptotic expansion formula.
// I_omega_z,log_unscale,unscale : i.omega,i.omega.[z - eta.log[2z]],exp[i.omega.[z - eta.log[2z]]]
// x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
// F,dF : regular Coulomb wave function and derivative in z.
// omega_norm_functions : for l,eta,z real, omega.C(l,eta)^2 if is_it_normalized is false, omega if it is true.
//                        One indeed has H[omega] = G + I.omega.norm.F, with in this case G and omega.norm.F as its real and imaginary parts.

void Coulomb_wave_functions::H_dH (const int omega,const std::complex<double> &z,std::complex<double> &H,std::complex<double> &dH)
{
  bool is_it_successful = false;

  std::complex<double> H_scaled,dH_scaled;
  if (!neg_int_omega_one && !neg_int_omega_minus_one) asymptotic_expansion_H_dH_scaled (omega,1.0/z,H_scaled,dH_scaled,is_it_successful);
```

```

  if (is_it_successful)
  {
    const std::complex<double> I_omega(0,omega),log_unscale = I_omega*(z - eta*(M_LN2 + log (z))),unscale = exp (log_unscale);

    if ((unscale == 0.0) || (!isfinite (unscale)))
      H = exp (log (H_scaled) + log_unscale),dH = exp (log (dH_scaled) + log_unscale);
    else
      H = H_scaled*unscale,dH = dH_scaled*unscale;
  }
  else
  {
    const double x = real (z),y = imag (z),l_r = real (l),l_i = imag (l),eta_r = real (eta),eta_i = imag (eta);

    if (((y != 0.0) || (eta_i != 0.0) || (l_i != 0.0))
    && (std::abs (y) < sqrt_precision*std::min (1.0,x)) && (std::abs (eta_i) < sqrt_precision) && (std::abs (l_i) < sqrt_precision)
    && (!neg_int_omega_one && !neg_int_omega_minus_one))
      H_dH_from_first_order_expansions (omega,z,H,dH);
    else
    {
      if (!neg_int_omega_one && !neg_int_omega_minus_one && (std::abs (l_i) >= 1.0) && (std::abs (z) <= 1.0)) H_dH_with_expansion (omega,z,H,dH,is_it_successful);

      if (!is_it_successful) H_dH_with_F_dF_and_CF (omega,z,H,dH);

      if ((y == 0.0) && (eta_i == 0.0) && (l_i == 0.0))
      {
    std::complex<double> F,dF;
    F_dF (z,F,dF);

    const double omega_norm_functions = (!is_it_normalized) ? (omega*real (Cl_eta)*real (Cl_eta)) : (omega);
    H = std::complex<double> (real (H),omega_norm_functions*real (F));
    dH = std::complex<double> (real (dH),omega_norm_functions*real (dF));
      }
    }
  }

  if (!isfinite (H) || !isfinite (dH)) std::cout«"Numerical failure encountered in H_dH."«std::endl,exit (1);
}







// Calculation of the scaled H[omega](z) and H'[omega](z).
// -------------------------------------------------------
// They are H(omega)(z).exp[-i.omega.[z - eta.log[2z]]] and dH(omega)/dz(z).exp[-i.omega.[z - eta.log[2z]]].
// One first tries the asymptotic expansion formula if 1+l+/-i.eta is no negative integer.
// If it failed, and imaginary parts of l,eta,z are much smaller than their real parts but not all zero, with Re[z] > 0,
// one calculates H[omega](z) and H'[omega](z) with the first order expansion method.
// If one is not in this case, one calculates F(z) and F'(z).
// If |Im[l]| > 1 and |z| <= 1, one tries the expansion formula with F(l,eta,z) and F(-l-1,eta,z).
// If not, or if it failed, one uses the continued fraction formula.
// If l,eta and z are real, one rewrites H[omega] as H[omega] = Re[H[omega]] + I.omega.norm.Re[F], to avoid numerical inaccuracies for Im[H[omega]].
// norm is 1 if is_it_normalized is true, C(l,eta)^2 if not.
// One uses logs if the scaling factor underflows or overflows.
//
// Variables :
// -----------
// omega : 1 if one calculates H+(z) and H+'(z) scaled, -1 if one calculates H-(z) and H-'(z) scaled.
// z : variable of the Coulomb wave function.
// H_scaled,dH_scaled : H[omega](z).exp[-i.omega.[z - eta.log[2z]]) and H'[omega](z).exp[-i.omega.[z - eta.log[2z]]).
// is_it_successful : true if the asymptotic expansions converges, false it not (in asymptotic_expansion_H_dH_scaled).
//                    true if the expansion formula with F(l,eta,z) and F(-l-1,eta,z) worked, false it not (in H_dH_with_expansion).
// x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
// F,dF : regular Coulomb wave function and derivative in z.
// H,dH : H+(z) and H+'(z) if omega=1, H-(z) and H-'(z) if omega=-1.
// omega_norm_functions : for l,eta,z real, omega.C(l,eta)^2 if is_it_normalized is false, omega if it is true.
//                          One indeed has H[omega] = G + I.omega.norm.F, with in this case G and omega.norm.F as its real and imaginary parts.
// I_omega_z,log_scale,scale : i.omega,-i.omega.[z - eta.log[2z]],exp[-i.omega.[z - eta.log[2z]]]

void Coulomb_wave_functions::H_dH_scaled (const int omega,const std::complex<double>
```

```
&z,std::complex<double> &H_scaled,std::complex<double> &dH_scaled)
{
  bool is_it_successful = false;
  if (!neg_int_omega_one && !neg_int_omega_minus_one) asymptotic_expansion_H_dH_scaled (omega,1.0/z,H_scaled,dH_scaled,is_it_successful);

  if (!is_it_successful)
  {
    std::complex<double> H,dH;

    const double x = real (z),y = imag (z),l_r = real (l),l_i = imag (l),eta_r = real (eta),eta_i = imag (eta);

    if (((y != 0.0) || (eta_i != 0.0) || (l_i != 0.0))
    && (std::abs (y) < sqrt_precision*std::min (1.0,x)) && (std::abs (eta_i) < sqrt_precision) && (std::abs (l_i) < sqrt_precision)
    && (!neg_int_omega_one && !neg_int_omega_minus_one))
      H_dH_from_first_order_expansions (omega,z,H,dH);
    else
    {
      if (!neg_int_omega_one && !neg_int_omega_minus_one && (std::abs (l_i) >= 1.0) && (abs (z) <= 1.0)) H_dH_with_expansion (omega,z,H,dH,is_it_successful);

      if (!is_it_successful) H_dH_with_F_dF_and_CF (omega,z,H,dH);

      if ((y == 0.0) && (eta_i == 0.0) && (l_i == 0.0))
      {
    std::complex<double> F,dF;
    F_dF (z,F,dF);

    const double omega_norm_functions = (!is_it_normalized) ? (omega*real (Cl_eta)*real (Cl_eta)) : (omega);
    H = std::complex<double> (real (H),omega_norm_functions*real (F));
    dH = std::complex<double> (real (dH),omega_norm_functions*real (dF));
      }
    }

    const std::complex<double> I_omega(0,omega),log_scale = -I_omega*(z - eta*(M_LN2 + log (z))),scale = exp (log_scale);

    if ((scale == 0.0) || (!isfinite (scale)))
      H_scaled = exp (log (H) + log_scale),dH_scaled = exp (log (dH) + log_scale);
    else
      H_scaled = H*scale,dH_scaled = dH*scale;
  }

  if (!isfinite (H_scaled) || !isfinite (dH_scaled)) std::cout«"Numerical failure encountered in H_dH_scaled."«std::endl,exit (1);
}


// Storage of initial conditions debut,F(debut),F'(debut)
// ---------------------------------------------------------

void Coulomb_wave_functions::F_dF_init (const std::complex<double> &z,const std::complex<double> &F,const std::complex<double> &dF)
{
  debut = z, F_debut = F, dF_debut = dF;
}

```

8.11 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/ode_int.cpp File Reference

```
#include "ode_int.H"
```

8.12 ode_int.cpp

Go to the documentation of this file.

```
#include "ode_int.H"

// Extrapolation in h=0 of a table of function values h close to h=0
```

```
// ------------------------------------------------------------------
//
// Variables:
// ----------
// n : number of points of the function f near h=0.
// T : table containing the points f[h(0)]...f[h(n-1)] close to h=0.
// f_in_zero : extrapolated value of the points f[h(0)]...f[h(n-1)] in h=0.


std::complex<double> ODE_integration::extrapolation_in_zero (const unsigned int n,const
std::complex<double> T[]) const
{
  std::complex<double> f_in_zero = 0.0;

  for (unsigned int i = 0 ; i < n ; i++)
    f_in_zero += interpolation_term_tab[n][i]*T[i];

  return f_in_zero;
}



// Calculation of F(z,u(z)) in u''(z) = F(z,u(z))
// ----------------------------------------------
//
// F(z,u(z))=(l(l+1)/(z^2) + 2.eta/z - 1).u(z),
//
// Variables:
// ----------
// z : parameter of the wave function.
// u : discretized wave function in z.
// one_over_z : 1.0/z

std::complex<double> ODE_integration::F_r_u (const std::complex<double> &z,const std::complex<double>
&u) const
{
  if (l == 0) return (two_eta/z - 1.0)*u;

  const std::complex<double> one_over_z = 1.0/z;

  return ((ll_plus_one*one_over_z + two_eta)*one_over_z - 1.0)*u;
}



// Integration with discretization of u''(r)=F(r,u(r)) with the Henrici method.
// ---------------------------------------------------------------------------
//
// See Numerical Recipes for the method.
//
// Initials conditions : r0,u(r0),du/dr(r0).
// Obtained functions : r,u(r),du/dr(r).
//
// Variables:
// ----------
// m : number of intervals between r0 and r
// h : integration step (r-r0)/m .
// r0,u0,du0 : r0,u(r0),du/dr(r0).
// r,u,du : r,u(r),du/dr(r).
// h_square : h*h
// half_h = 0.5*h
// delta : value used in the Henrici method.

void ODE_integration::integration_Henrici (const unsigned int m,const std::complex<double> &h,
                       const std::complex<double> &r0,const std::complex<double> &u0,const
std::complex<double> &du0,
                       const std::complex<double> &r,std::complex<double> &u,std::complex<double> &du)
const
{
  const std::complex<double> h_square = h*h,half_h = 0.5*h;

  std::complex<double> delta = h*(du0 + half_h*F_r_u (r0,u0));
  u = u0 + delta;

  for (unsigned int i = 1 ; i < m ; i++)
  {
    delta += h_square*F_r_u (r0 + i*h,u);
    u += delta;
  }

  du = delta/h + half_h*F_r_u (r,u);
}
```

```
// Integration of u''(r) = F(r,u(r)) with the Bulirsch-Stoer-Henrici method.
// ------------------------------------------------------------------------
//
// Initials conditions : r0,u0=u(r0),du0=du/dr(r0)
// Obtained functions : r,u=u(r),du=du/dr(r)
//
// See Numerical Recipes for the method.
//
// Variables:
// ----------
// r0,u0,du0 : r0,u(r0),du/dr(r0).
// r,u,du : r,u(r),du/dr(r).
// H,r_debut,r_end,u_debut,du_debut : length of an integration interval, debut and end of the integration interval, u(r_debut), u'(r_debut).
//                                      H is equal to r-r0 at the beginning and is divided by 2 each time the extrapolation fails with 16 sub-intervals
//                                      between r_debut and r_end. If H = [r-r0]/N, with N=2,4,8,16,...,
//                                      the integration intervals are [r_debut = r0:r_end = r0+H], ..., [r_debut = r0+(N-1).H,r_end = r].
// u_end,du_end,k : tables of u(r_end) and u'(r_end) values calculated with the Henrici method with 2,4,6,...,2(k+1) sub-intervals between r_debut and r_end,
//                  with 0 <= k <= 7.
// H_over_m_tab : H/m for m=2,4,6,...,16.
// inf_norm_half_H : |H/2|oo. It is used to know if r = r_debut up to numerical accuracy, as one has |r-r_debut|oo <= |H/2|oo for this case only.
// u_extrapolated,u_extrapolated_next : values of u extrapolated from the points of the table u_end with k and k->k+1 points, k >= 2.
// test : test to know if the method worked, i.e., |u_extrapolated/u_extrapolated_next - 1|oo < precision.
// du_extrapolated_next : value of u'(r_end) extrapolated from k points of the table du_end, k >= 3.
// r_debut_plus_H : r_debut+H. r_debut+H at the end of integration is not necessarily r because of numerical cancellations.
//                             In this case, r_end must be put equal to r.

void ODE_integration::operator() (const std::complex<double> &r0,const std::complex<double> &u0,const std::complex<double> &du0,
                       const std::complex<double> &r,std::complex<double> &u,std::complex<double> &du) const
{
  if (r == r0) {u = u0; du = du0; return;}

  std::complex<double> r_debut = r0,u_debut = u0,du_debut = du0,H = r-r0,u_end[8],du_end[8],u_extrapolated_next,du_extrapolated_next;
  double test = 1.0;

  while (test > precision)
  {
    std::complex<double> H_over_m_tab[8];
    for (unsigned int k = 0 ; k < 8 ; k++) H_over_m_tab[k] = H*one_over_m_tab[k];
    const double inf_norm_half_H = inf_norm (H_over_m_tab[0]);

    while (inf_norm (r_debut - r) > inf_norm_half_H)
    {
      const std::complex<double> r_debut_plus_H = r_debut + H, r_end = (inf_norm (r - r_debut_plus_H) > inf_norm_half_H) ? (r_debut_plus_H) : (r);

      integration_Henrici (2,H_over_m_tab[0],r_debut,u_debut,du_debut,r_end,u_end[0],du_end[0]);
      integration_Henrici (4,H_over_m_tab[1],r_debut,u_debut,du_debut,r_end,u_end[1],du_end[1]);
      std::complex<double> u_extrapolated = extrapolation_in_zero (2,u_end);

      unsigned int k = 2;
      do
      {
    integration_Henrici (m_tab[k],H_over_m_tab[k],r_debut,u_debut,du_debut,r_end,u_end[k],du_end[k]);
    u_extrapolated_next = extrapolation_in_zero (++k,u_end);
    test = inf_norm (u_extrapolated/u_extrapolated_next - 1.0);
    u_extrapolated = u_extrapolated_next;
      }
      while ((test > precision) && (k < 7));

      r_debut += H;
      u_debut = u_extrapolated_next;
      du_debut = du_extrapolated_next = extrapolation_in_zero (k,du_end);
    }

    H *= 0.5;
    r_debut = r0;
    u_debut = u0;
    du_debut = du0;
  }

  u = u_extrapolated_next;
```

```
  du = du_extrapolated_next;
}
```

8.13 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AboutAZURE2Dialog.h File Reference

```
#include <QDialog>
```

Classes

- class AboutAZURE2Dialog

8.14 AboutAZURE2Dialog.h

Go to the documentation of this file.

```
#ifndef ABOUTAZURE2DIALOG_H
#define ABOUTAZURE2DIALOG_H

#include <QDialog>

QT_BEGIN_NAMESPACE
QT_END_NAMESPACE

class AboutAZURE2Dialog : public QDialog {
  Q_OBJECT

 public:
  AboutAZURE2Dialog(QWidget *parent=0);

};

#endif
```

8.15 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddLevelDialog.h File Reference

```
#include <QDialog>
#include <QtWidgets>
```

Classes

- class AddLevelDialog

8.16 AddLevelDialog.h

Go to the documentation of this file.

```
#ifndef ADDLEVELDIALOG_H
#define ADDLEVELDIALOG_H

#include <QDialog>
#include <QtWidgets>

QT_BEGIN_NAMESPACE

class QLineEdit;
class QLabel;

QT_END_NAMESPACE

class AddLevelDialog : public QDialog {
  Q_OBJECT

 public:
  AddLevelDialog(QWidget *parent=0);
  QLineEdit *jValueText;
  QComboBox *piValueCombo;
  QLineEdit *energyText;

 private:
  QLabel *jValueLabel;
  QLabel *energyLabel;

  QPushButton *okButton;
  QPushButton *cancelButton;
};

#endif
```

8.17 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddPairDialog.h File Reference

```
#include <QDialog>
```

Classes

- class AddPairDialog

8.18 AddPairDialog.h

Go to the documentation of this file.

```
#ifndef ADDPAIRDIALOG_H
#define ADDPAIRDIALOG_H

#include <QDialog>

QT_BEGIN_NAMESPACE

class QLineEdit;
class QLabel;
class QCheckBox;
class QGroupBox;
class QComboBox;
class QPushButton;

QT_END_NAMESPACE

class AddPairDialog : public QDialog {
  Q_OBJECT

```

```
public:
  AddPairDialog(QWidget *parent=0);
  QLineEdit *lightJText;
  QComboBox *lightPiCombo;
  QLineEdit *lightZText;
  QLineEdit *lightMText;
  //QLineEdit *lightGText;
  QLineEdit *heavyJText;
  QComboBox *heavyPiCombo;
  QLineEdit *heavyZText;
  QLineEdit *heavyMText;
  //QLineEdit *heavyGText;
  QLineEdit *excitationEnergyText;
  QLineEdit *seperationEnergyText;
  QLineEdit *channelRadiusText;
  QComboBox *pairTypeCombo;
  QCheckBox *e1Check;
  //QCheckBox *m1Check;
  QCheckBox *e2Check;
  QGroupBox *multBox;

public slots:
  void updateLightParticle(int index);

private:
  QLabel *lightJLabel;
  QLabel *lightPiLabel;
  QLabel *lightZLabel;
  QLabel *lightMLabel;
  //QLabel *lightGLabel;
  QLabel *heavyJLabel;
  QLabel *heavyPiLabel;
  QLabel *heavyZLabel;
  QLabel *heavyMLabel;
  //QLabel *heavyGLabel;
  QLabel *excitationEnergyLabel;
  QLabel *seperationEnergyLabel;
  QLabel *channelRadiusLabel;
  QLabel *pairTypeLabel;

  QPushButton *okButton;
  QPushButton *cancelButton;
};

#endif
```

8.19 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddSegDataDialog.h File Reference

```
#include <QDialog>
#include <QLineEdit>
#include <QSpinBox>
#include <QComboBox>
#include <QLabel>
#include <QCheckBox>
```

Classes

- class AddSegDataDialog

8.20 AddSegDataDialog.h

Go to the documentation of this file.

```
#ifndef ADDSEGDATADIALOG_H
#define ADDSEGDATADIALOG_H

```

```
#include <QDialog>
#include <QLineEdit>
#include <QSpinBox>
#include <QComboBox>
#include <QLabel>
#include <QCheckBox>

QT_BEGIN_NAMESPACE

class QLabel;

QT_END_NAMESPACE

class AddSegDataDialog : public QDialog {
  Q_OBJECT

 public:
  AddSegDataDialog(QWidget *parent=0);
  QSpinBox *entrancePairIndexSpin;
  QSpinBox *exitPairIndexSpin;
  QLineEdit *lowEnergyText;
  QLineEdit *highEnergyText;
  QLineEdit *lowAngleText;
  QLineEdit *highAngleText;
  QComboBox *dataTypeCombo;
  QLineEdit *dataFileText;
  QLineEdit *dataNormText;
  QLineEdit *dataNormErrorText;
  QLabel *dataNormErrorLabel;
  QCheckBox *varyNormCheck;
  QLineEdit *phaseJValueText;
  QLineEdit *phaseLValueText;
  QLabel* phaseLValueLabel;
  QLabel* phaseJValueLabel;
  QLabel* totalCaptureLabel;

 public slots:
  void setChooseFile();
  void dataTypeChanged(int);
  void varyNormChanged(int);

 private:
  QPushButton *okButton;
  QPushButton *cancelButton;
};

#endif
```

8.21 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddSegTestDialog.h File Reference

```
#include <QDialog>
#include <QLineEdit>
#include <QSpinBox>
#include <QComboBox>
#include <QLabel>
```

Classes

- class AddSegTestDialog

8.22 AddSegTestDialog.h

Go to the documentation of this file.

```
#ifndef ADDSEGTESTDIALOG_H
#define ADDSEGTESTDIALOG_H

```

```
#include <QDialog>
#include <QLineEdit>
#include <QSpinBox>
#include <QComboBox>
#include <QLabel>

QT_BEGIN_NAMESPACE

class QLabel;

QT_END_NAMESPACE

class AddSegTestDialog : public QDialog {
  Q_OBJECT

 public:
  AddSegTestDialog(QWidget *parent=0);
  QSpinBox *entrancePairIndexSpin;
  QSpinBox *exitPairIndexSpin;
  QLineEdit *lowEnergyText;
  QLineEdit *highEnergyText;
  QLineEdit *energyStepText;
  QLineEdit *lowAngleText;
  QLineEdit *highAngleText;
  QLineEdit *angleStepText;
  QComboBox *dataTypeCombo;
  QLineEdit *phaseJValueText;
  QLineEdit *phaseLValueText;
  QLabel *phaseJValueLabel;
  QLabel *phaseLValueLabel;
  QLabel *angDistLabel;
  QLabel *totalCaptureLabel;
  QSpinBox *angDistSpin;

 public slots:
  void dataTypeChanged(int);

 private:
  QPushButton *okButton;
  QPushButton *cancelButton;
};

#endif
```

8.23 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddTargetIntDialog.h File Reference

`#include <QDialog>`

Classes

- class AddTargetIntDialog

8.24 AddTargetIntDialog.h

Go to the documentation of this file.

```
#ifndef ADDTARGETINTDIALOG_H
#define ADDTARGETINTDIALOG_H

#include <QDialog>

QT_BEGIN_NAMESPACE

class QLineEdit;
class QSpinBox;
class QCheckBox;
class QTableWidget;
class QGroupBox;
class QSize;
```

```

QT_END_NAMESPACE

class AddTargetIntDialog : public QDialog {
  Q_OBJECT

 public:
  AddTargetIntDialog(QWidget *parent=0);
  QCheckBox *isConvolutionCheck;
  QCheckBox *isTargetIntegrationCheck;
  QCheckBox *isQCoefficientCheck;
  QLineEdit *sigmaText;
  QLineEdit *segmentsListText;
  QSpinBox *numPointsSpin;
  QSpinBox *numParametersSpin;
  QSpinBox *numQCoefficientSpin;
  QLineEdit *densityText;
  QLineEdit *stoppingPowerEqText;
  QTableWidget *parametersTable;
  QTableWidget *qCoefficientTable;
  QList<double> tempParameters;
  QList<double> tempQCoefficients;
  void createParameterItem(int row, double value = 0.0);
  void createQCoefficientItem(int row, double value = 1.0);

 public slots:
  void convolutionCheckChanged(bool checked);
  void targetIntCheckChanged(bool checked);
  void parameterSpinChanged(int newNumber);
  void parameterChanged(int row, int column);
  void qCoefficientCheckChanged(bool checked);
  void qCoefficientSpinChanged(int newNumber);
  void qCoefficientChanged(int row, int column);

 private:
  QPushButton *okButton;
  QPushButton *cancelButton;
  QGroupBox *stoppingPowerBox;
  QGroupBox *qCoefficientBox;
};

#endif
```

8.25 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREMainThread.h File Reference

```
#include <QThread>
#include <QEventLoop>
#include <QPushButton>
#include "TextEditBuffer.h"
#include "FilteredTextEdit.h"
#include "RunTab.h"
#include "AZUREMain.h"
#include "Config.h"
#include <iostream>
```

Classes

- class AZUREMainThreadWorker
- class AZUREMainThread

8.26 AZUREMainThread.h

Go to the documentation of this file.

```
#ifndef AZUREMAINTHREAD_H
#define AZUREMAINTHREAD_H

#include <QThread>
#include <QEventLoop>
#include <QPushButton>

#include "TextEditBuffer.h"
#include "FilteredTextEdit.h"
#include "RunTab.h"
#include "AZUREMain.h"
#include "Config.h"
#include <iostream>

class AZUREMainThreadWorker : public QObject {
Q_OBJECT

 public:
  AZUREMainThreadWorker(const Config& configure) :
  azureMain_(configure) {};
 signals:
  void done();
 public slots:
  void run() {
    azureMain_();
    emit done();
  };
 private:
  AZUREMain azureMain_;
};

class AZUREMainThread : public QThread {
  Q_OBJECT
 public:
  AZUREMainThread(RunTab *tab, const Config& configure) :
  stream_(&buffer_), configure_(stream_), worker_(configure_) {
    configure_.configfile = configure.configfile;
    configure_.paramMask = configure.paramMask;
    configure_.screenCheckMask = configure.screenCheckMask;
    configure_.fileCheckMask = configure.fileCheckMask;
    configure_.chiVariance = configure.chiVariance;
    configure_.outputdir = configure.outputdir;
    configure_.checkdir = configure.checkdir;
    configure_.paramfile = configure.paramfile;
    configure_.integralsfile = configure.integralsfile;
    configure_.rateParams = configure.rateParams;
    connect(&buffer_,SIGNAL(updateLog(QString)),tab->runtimeText,SLOT(write(QString)));
    connect(tab->stopAZUREButton,SIGNAL(clicked()),this,SLOT(stopAZURE()));
    connect(this,SIGNAL(readyToRun()),&worker_,SLOT(run()));
    connect(&worker_,SIGNAL(done()),this,SLOT(quit()));
    worker_.moveToThread(this);
  };
  const Config& configure() const {return configure_;};
 signals:
  void readyToRun();
 public slots:
  void stopAZURE() {
    configure_.stopFlag=true;
  };
 protected:
  void run() {
    emit readyToRun();
    exec();
  };
 private:
  TextEditBuffer buffer_;
  std::ostream stream_;
  Config configure_;
  AZUREMainThreadWorker worker_;
};

#endif
```

8.27 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREPlot.h File Reference

```
#include <qwt_plot.h>
#include <qwt_symbol.h>
#include <qwt_plot_zoomer.h>
```

Classes

- struct PlotPoint
- class AZUREZoomer
- class PlotEntry
- class AZUREPlot

8.28 AZUREPlot.h

Go to the documentation of this file.

```
#ifndef AZUREPLOT_H
#define AZUREPLOT_H

#include <qwt_plot.h>
#include <qwt_symbol.h>
#include <qwt_plot_zoomer.h>

class QwtPlotCurve;
class QwtPlotIntervalCurve;
class PlotTab;

struct PlotPoint {
  double energy;
  double excitationEnergy;
  double angle;
  double fitCrossSection;
  double fitSFactor;
  double dataCrossSection;
  double dataErrorCrossSection;
  double dataSFactor;
  double dataErrorSFactor;
};

class AZUREZoomer : public QwtPlotZoomer {
 public:
  AZUREZoomer(QWidget *canvas) : QwtPlotZoomer(canvas) {};
 protected:
  QwtText trackerTextF( const QPointF &pos ) const;

};

class PlotEntry {
 public:
  PlotEntry(int type, int entranceKey, int exitKey, int index, QString filename);
  ~PlotEntry();

  int type() const {return type_;};

  bool readData();
  void attach(QwtPlot*,int,int,QwtSymbol::Style);
  void detach();

 public:
  friend class AZUREPlot;

 private:
  bool hasNegative_;
  int type_;
  int entranceKey_;
  int exitKey_;
  int index_;
  QString filename_;
  QwtPlotCurve* dataCurve_;
  QwtPlotIntervalCurve* dataErrorCurve_;
  QwtPlotCurve* fitCurve_;
  QVector<PlotPoint> points_;
};

class AZUREPlot : public QwtPlot {

  Q_OBJECT

 public:
  AZUREPlot(PlotTab* plotTab, QWidget* parent = 0);
  void setXAxisLog(bool set);
  void setYAxisLog(bool set);
  void setXAxisType(unsigned int type);
  void setYAxisType(unsigned int type);
```

```
 void clearEntries();

public slots:
 void draw(QList<PlotEntry*> newEntries);
 void update();
 void exportPlot();
 void print();

private:
 unsigned int xAxisType;
 unsigned int yAxisType;
 QList<PlotEntry*> entries;
 AZUREZoomer* zoomer;
 PlotTab* containingTab;
};

#endif
```

8.29 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZURESetup.h File Reference

```
#include <QMainWindow>
#include "PairsTab.h"
#include "LevelsTab.h"
#include "SegmentsTab.h"
#include "TargetIntTab.h"
#include "Config.h"
```

Classes

- class Directories
- class AZURESetup

8.30 AZURESetup.h

Go to the documentation of this file.

```
#ifndef AZURESETUP_H
#define AZURESETUP_H

#include <QMainWindow>

#include "PairsTab.h"
#include "LevelsTab.h"
#include "SegmentsTab.h"
#include "TargetIntTab.h"
#include "Config.h"

class RunTab;
#ifdef USE_QWT
class PlotTab;
#endif
class AZUREMainThread;

QT_BEGIN_NAMESPACE

class QTabWidget;
class QMenu;
class QAction;
class QActionGroup;
class QTextEdit;

QT_END_NAMESPACE

class Directories {
 public:
```

```
 Directories() : outputDir(QString("")), checksDir(QString("")) {};
  QString outputDir;
  QString checksDir;
};

class AZURESetup : public QMainWindow {

 Q_OBJECT

 public:
  AZURESetup();
  Config& GetConfig();
  void open(QString filename);

 public slots:
  void SaveAndRun();
  void DeleteThread();

 private slots:
  void reset();
  void open();
  void openRecent();
  void clearRecent();
  void save();
  void saveAs();
  void matrixChanged(QAction* action);
  void editChecks();
  void editDirs();
  void editOptions();
  void showAbout();
  void showTabInfo();
  void openWebsite();

 private:
  bool readFile(QString filename);
  bool readConfig(QTextStream& inStream);
  bool writeFile(QString filename);
  bool writeConfig(QTextStream& outStream, QString directory);
  bool readLastRun(QTextStream& inStream);
  bool writeLastRun(QTextStream& outStream);
  void createActions();
  void createMenus();
  void updateRecent();

  Config config;

  QAction* aboutAction;
  QAction* resetAction;
  QAction* quitAction;
  QAction* openAction;
  QAction* saveAction;
  QAction* saveAsAction;
  QAction* editChecksAction;
  QAction* editDirsAction;
  QAction* copyAction;
  QAction* aMatrixAction;
  QAction* rMatrixAction;
  QAction* editOptionsAction;
  QAction* recentSeparator;
  QAction* clearRecentAction;
  enum { numRecent = 5 };
  QAction* recentFileActions[numRecent];
  QAction* showTabInfoAction;
  QAction* openAZURESiteAction;

  QActionGroup* matrixActionGroup;

  QMenu *fileMenu;
  QMenu *editMenu;
  QMenu *configMenu;
  QMenu *formalismMenu;
  QMenu *recentFileMenu;
  QMenu *helpMenu;
  QTabWidget *tabWidget;
  PairsTab *pairsTab;
  LevelsTab *levelsTab;
  SegmentsTab *segmentsTab;
  TargetIntTab *targetIntTab;
  RunTab *runTab;
  AZUREMainThread *azureMain;
#ifdef USE_QWT
  PlotTab* plotTab;
#endif
};

#endif
```

8.31 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelDetails.h File Reference

```
#include <QWidget>
```

Classes

- class ChannelDetails

8.32 ChannelDetails.h

Go to the documentation of this file.

```
#ifndef CHANNELDETAILS_H
#define CHANNELDETAILS_H

#include <QWidget>

QT_BEGIN_NAMESPACE

class QLineEdit;
class QLabel;

QT_END_NAMESPACE

class ChannelDetails : public QWidget {
  Q_OBJECT

 public:
  ChannelDetails(QWidget *parent = 0);
  void setNormParam(int which);
  QLineEdit *reducedWidthText;
  QLabel *details;
 private:
  QLabel *normParam;
  QLabel *normUnits;
};

#endif
```

8.33 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelsModel.h File Reference

```
#include <QAbstractTableModel>
#include <QList>
```

Classes

- struct ChannelsData
- class ChannelsModel

8.34 ChannelsModel.h

Go to the documentation of this file.

```
#ifndef CHANNELSMODEL_H
#define CHANNELSMODEL_H

#include <QAbstractTableModel>
#include <QList>

struct ChannelsData {
  static const int SIZE = 7;
  int isFixed;
  int levelIndex;
  int pairIndex;
  double sValue;
  int lValue;
  QChar radType;
  double reducedWidth;
};

class PairsModel;

class ChannelsModel : public QAbstractTableModel {
  Q_OBJECT

 public:
  ChannelsModel(QObject *parent = 0);

  int rowCount(const QModelIndex &parent) const;
  int columnCount(const QModelIndex &parent) const;
  QVariant data(const QModelIndex &index, int role) const;
  QVariant headerData(int section, Qt::Orientation orientation, int role) const;
  QList<ChannelsData> getChannels() const {return channelsList;};
  bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
  bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
  bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
  Qt::ItemFlags flags(const QModelIndex &index) const;
  bool isChannel(const ChannelsData &channel) const;
  QString getSpinLabel(const ChannelsData &channel) const;
  void setPairsModel(PairsModel *model);

 private:
  QList<ChannelsData> channelsList;
  PairsModel *pairsModel;
};

#endif
```

8.35 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChooseFile↩ Button.h File Reference

```
#include <QPushButton>
#include <QWidget>
#include <QLineEdit>
#include <QString>
```

Classes

- class ChooseFileButton

8.36 ChooseFileButton.h

Go to the documentation of this file.

```
#ifndef CHOOSEFILEBUTTON_H
#define CHOOSEFILEBUTTON_H
```

```

#include <QPushButton>
#include <QWidget>
#include <QLineEdit>
#include <QString>

class ChooseFileButton : public QPushButton {

  Q_OBJECT;

 public:
  ChooseFileButton(const QString& text, QWidget *parent = 0);
  void setLineEdit(QLineEdit* lineEdit);

 public slots:
  void click();

 signals:
  void clicked(QLineEdit *lineEdit);

 private:
  QLineEdit *thisLineEdit;

};

#endif
```

8.37 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditChecksDialog.h File Reference

`#include <QDialog>`

Classes

- class EditChecksDialog

8.38 EditChecksDialog.h

Go to the documentation of this file.

```
#ifndef EDITCHECKSDIALOG_H
#define EDITCHECKSDIALOG_H

#include <QDialog>

QT_BEGIN_NAMESPACE

class QPushButton;
class QGroupBox;
class QComboBox;

QT_END_NAMESPACE

class EditChecksDialog : public QDialog {

Q_OBJECT

 public:
  EditChecksDialog(QWidget *parent = 0);
  QComboBox *compoundCheckCombo;
  QComboBox *boundaryCheckCombo;
  QComboBox *dataCheckCombo;
  QComboBox *lMatrixCheckCombo;
  QComboBox *legendreCheckCombo;
  QComboBox *coulAmpCheckCombo;
  QComboBox *pathwaysCheckCombo;
  QComboBox *angDistsCheckCombo;

 private:
  QPushButton *okButton;
```

```
  QPushButton *cancelButton;
  QGroupBox *stoppingPowerBox;

};

#endif
```

8.39 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditDirsDialog.h File Reference

```
#include <QDialog>
```

Classes

- class EditDirsDialog

8.40 EditDirsDialog.h

Go to the documentation of this file.

```
#ifndef EDITDIRSDIALOG_H
#define EDITDIRSDIALOG_H

#include <QDialog>

QT_BEGIN_NAMESPACE

class QPushButton;
class QGroupBox;
class QComboBox;
class QLineEdit;

QT_END_NAMESPACE

class EditDirsDialog : public QDialog {

Q_OBJECT

 public:
  EditDirsDialog(QWidget *parent = 0);
  QLineEdit *outputDirectoryText;
  QLineEdit *checksDirectoryText;

 private slots:
  void setChooseDirectory(QLineEdit*);

 private:
  QPushButton *okButton;
  QPushButton *cancelButton;
};

#endif
```

8.41 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditOptions←Dialog.h File Reference

```
#include <QDialog>
```

Classes

- class EditOptionsDialog

8.42 EditOptionsDialog.h

Go to the documentation of this file.

```
#ifndef EDITOPTIONSDIALOG_H
#define EDITOPTIONSDIALOG_H


#include <QDialog>

QT_BEGIN_NAMESPACE

class QPushButton;
class QGroupBox;
class QCheckBox;

QT_END_NAMESPACE

QT_END_NAMESPACE

class EditOptionsDialog : public QDialog {

Q_OBJECT

 public:
  EditOptionsDialog(QWidget *parent =0);
  QCheckBox* useBruneCheck;
  QCheckBox* useGSLCoulCheck;
  QCheckBox* ignoreExternalsCheck;
  QCheckBox* useRMCCheck;
  QCheckBox* noTransformCheck;
  // QCheckBox* noLongWavelengthCheck;

 private slots:
  void useBruneCheckChanged(int);
  void useRMCCheckChanged(int);

 private:
  QPushButton *okButton;
  QPushButton *cancelButton;

};

#endif
```

8.43 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ElementMap.h File Reference

```
#include <QString>
#include <map>
```

8.44 ElementMap.h

Go to the documentation of this file.

```
#ifndef ELEMENTMAP_H
#define ELEMENTMAP_H

#include <QString>
#include <map>

static const std::pair<int, QString> elements[] = {
  std::pair<int, QString>(0,"n"),
```

```
std::pair<int, QString>(1,"H"),
std::pair<int, QString>(2,"He"),
std::pair<int, QString>(3,"Li"),
std::pair<int, QString>(4,"Be"),
std::pair<int, QString>(5,"B"),
std::pair<int, QString>(6,"C"),
std::pair<int, QString>(7,"N"),
std::pair<int, QString>(8,"O"),
std::pair<int, QString>(9,"F"),
std::pair<int, QString>(10,"Ne"),
std::pair<int, QString>(11,"Na"),
std::pair<int, QString>(12,"Mg"),
std::pair<int, QString>(13,"Al"),
std::pair<int, QString>(14,"Si"),
std::pair<int, QString>(15,"P"),
std::pair<int, QString>(16,"S"),
std::pair<int, QString>(17,"Cl"),
std::pair<int, QString>(18,"Ar"),
std::pair<int, QString>(19,"K"),
std::pair<int, QString>(20,"Ca"),
std::pair<int, QString>(21,"Sc"),
std::pair<int, QString>(22,"Ti"),
std::pair<int, QString>(23,"V"),
std::pair<int, QString>(24,"Cr"),
std::pair<int, QString>(25,"Mn"),
std::pair<int, QString>(26,"Fe"),
std::pair<int, QString>(27,"Co"),
std::pair<int, QString>(28,"Ni"),
std::pair<int, QString>(29,"Cu"),
std::pair<int, QString>(30,"Zn"),
std::pair<int, QString>(31,"Ga"),
std::pair<int, QString>(32,"Ge"),
std::pair<int, QString>(33,"As"),
std::pair<int, QString>(34,"Se"),
std::pair<int, QString>(35,"Br"),
std::pair<int, QString>(36,"Kr"),
std::pair<int, QString>(37,"Rb"),
std::pair<int, QString>(38,"Sr"),
std::pair<int, QString>(39,"Y"),
std::pair<int, QString>(40,"Zr"),
std::pair<int, QString>(41,"Nb"),
std::pair<int, QString>(42,"Mo"),
std::pair<int, QString>(43,"Tc"),
std::pair<int, QString>(44,"Ru"),
std::pair<int, QString>(45,"Rh"),
std::pair<int, QString>(46,"Pd"),
std::pair<int, QString>(47,"Ag"),
std::pair<int, QString>(48,"Cd"),
std::pair<int, QString>(49,"In"),
std::pair<int, QString>(50,"Sn"),
std::pair<int, QString>(51,"Sb"),
std::pair<int, QString>(52,"Te"),
std::pair<int, QString>(53,"I"),
std::pair<int, QString>(54,"Xe"),
std::pair<int, QString>(55,"Cs"),
std::pair<int, QString>(56,"Ba"),
std::pair<int, QString>(57,"La"),
std::pair<int, QString>(58,"Ce"),
std::pair<int, QString>(59,"Pr"),
std::pair<int, QString>(60,"Nd"),
std::pair<int, QString>(61,"Pm"),
std::pair<int, QString>(62,"Sm"),
std::pair<int, QString>(63,"Eu"),
std::pair<int, QString>(64,"Gd"),
std::pair<int, QString>(65,"Tb"),
std::pair<int, QString>(66,"Dy"),
std::pair<int, QString>(67,"Ho"),
std::pair<int, QString>(68,"Er"),
std::pair<int, QString>(69,"Tm"),
std::pair<int, QString>(70,"Yb"),
std::pair<int, QString>(71,"Lu"),
std::pair<int, QString>(72,"Hf"),
std::pair<int, QString>(73,"Ta"),
std::pair<int, QString>(74,"W"),
std::pair<int, QString>(75,"Re"),
std::pair<int, QString>(76,"Os"),
std::pair<int, QString>(77,"Ir"),
std::pair<int, QString>(78,"Pt"),
std::pair<int, QString>(79,"Au"),
std::pair<int, QString>(80,"Hg"),
std::pair<int, QString>(81,"Tl"),
std::pair<int, QString>(82,"Pb"),
std::pair<int, QString>(83,"Bi"),
std::pair<int, QString>(84,"Po"),
std::pair<int, QString>(85,"At"),
std::pair<int, QString>(86,"Rn"),
std::pair<int, QString>(87,"Fr"),
```

```
  std::pair<int, QString>(88,"Ra"),
  std::pair<int, QString>(89,"Ac"),
  std::pair<int, QString>(90,"Th"),
  std::pair<int, QString>(91,"Pa"),
  std::pair<int, QString>(92,"U"),
  std::pair<int, QString>(93,"Np"),
  std::pair<int, QString>(94,"Pu"),
  std::pair<int, QString>(95,"Am"),
  std::pair<int, QString>(96,"Cm"),
  std::pair<int, QString>(97,"Bk"),
  std::pair<int, QString>(98,"Cf"),
  std::pair<int, QString>(99,"Es"),
  std::pair<int, QString>(100,"Fm"),
  std::pair<int, QString>(101,"Md"),
  std::pair<int, QString>(102,"No"),
  std::pair<int, QString>(103,"Lr"),
  std::pair<int, QString>(104,"Rf"),
  std::pair<int, QString>(105,"Db"),
  std::pair<int, QString>(106,"Sg"),
  std::pair<int, QString>(107,"Bh"),
  std::pair<int, QString>(108,"Hs"),
  std::pair<int, QString>(109,"Mt"),
  std::pair<int, QString>(110,"Ds"),
  std::pair<int, QString>(111,"Rg"),
  std::pair<int, QString>(112,"Cn")
};

static const std::map<int, QString>
elementMap(elements,elements+sizeof(elements)/sizeof(std::pair<int,QString>));

#endif
```

8.45 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/FilteredTextEdit.h File Reference

```
#include <QTextEdit>
#include <iostream>
```

Classes

- class FilteredTextEdit

8.46 FilteredTextEdit.h

Go to the documentation of this file.

```
#ifndef FILTEREDTEXTEDIT_H
#define FILTEREDTEXTEDIT_H

#include <QTextEdit>

#include <iostream>

class FilteredTextEdit : public QTextEdit {
  Q_OBJECT

 public:
  FilteredTextEdit(QWidget *parent = 0) :
  QTextEdit(parent), filtered_(false) {
    QFont font("Courier");
    font.setStyleHint(QFont::TypeWriter);
    setCurrentFont(font);
  };
  void SetMouseFiltered(bool filtered) { filtered_=filtered;};
  bool IsMouseFiltered() const {return filtered_;};
 public slots:
  void write(QString string) {
    if(string[0]=='\r') {
      QTextCursor cursor = textCursor();
```

```
      cursor.select(QTextCursor::LineUnderCursor);
      setTextCursor(cursor);
      string.remove(0,1);
    }
    insertPlainText(string);
  };
  void mousePressEvent(QMouseEvent *event) {
    if(!filtered_) QTextEdit::mousePressEvent(event);
  };
  void mouseDoubleClickEvent(QMouseEvent *event) {
    if(!filtered_) QTextEdit::mouseDoubleClickEvent(event);
  }
 private:
  bool filtered_;
};

#endif
```

8.47 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/InfoDialog.h File Reference

```
#include <QDialog>
```

Classes

- class InfoDialog

8.48 InfoDialog.h

Go to the documentation of this file.

```
#ifndef INFODIALOG_H
#define INFODIALOG_H

#include <QDialog>

class InfoDialog : public QDialog {
Q_OBJECT

 public:
  InfoDialog(const QString&, QWidget* parent=0, QString title="");

};

#endif
```

8.49 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsHeader↵View.h File Reference

```
#include <QHeaderView>
#include <QMouseEvent>
```

Classes

- class LevelsHeaderView

8.50 LevelsHeaderView.h

Go to the documentation of this file.

```
#ifndef LEVELSHEADERVIEW_H
#define LEVELSHEADERVIEW_H

#include <QHeaderView>
#include <QMouseEvent>

class LevelsHeaderView : public QHeaderView {
 public:
  LevelsHeaderView(Qt::Orientation orientation, QWidget *parent) :
    QHeaderView(orientation,parent) {
      setSectionsClickable(true);
    };
 protected:
   virtual void mouseMoveEvent ( QMouseEvent * e ) {
     int pos = orientation() == Qt::Horizontal ? e->x() : e->y();
     int section = logicalIndexAt(pos);
     if(section>1) QHeaderView::mouseMoveEvent(e);
    };
   virtual void mousePressEvent ( QMouseEvent * e ) {
     int pos = orientation() == Qt::Horizontal ? e->x() : e->y();
     int section = logicalIndexAt(pos);
     if(section>1) QHeaderView::mousePressEvent(e);
    };
   virtual void mouseReleaseEvent ( QMouseEvent * e ) {
     int pos = orientation() == Qt::Horizontal ? e->x() : e->y();
     int section = logicalIndexAt(pos);
     if(section>1) QHeaderView::mouseReleaseEvent(e);
    };
};

#endif
```

8.51 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsModel.h File Reference

```
#include <QAbstractTableModel>
#include <QList>
```

Classes

- struct LevelsData
- class LevelsModel

8.52 LevelsModel.h

Go to the documentation of this file.

```
#ifndef LEVELSMODEL_H
#define LEVELSMODEL_H

#include <QAbstractTableModel>
#include <QList>

struct LevelsData {
  static const int SIZE = 5;
  int isActive;
  int isFixed;
  double jValue;
  int piValue;
  double energy;
};

class LevelsModel : public QAbstractTableModel {
```

```
 Q_OBJECT

public:
 LevelsModel(QObject *parent = 0);

 int rowCount(const QModelIndex &parent) const;
 int columnCount(const QModelIndex &parent) const;
 QVariant data(const QModelIndex &index, int role) const;
 QVariant headerData(int section, Qt::Orientation orientation, int role) const;
 bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
 bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
 bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
 Qt::ItemFlags flags(const QModelIndex &index) const;
 int isLevel(const LevelsData &level) const;
 QList<LevelsData> getLevels() const {return levelsList;};
 QString getSpinLabel(const LevelsData &level) const;

private:
 QList<LevelsData> levelsList;
};

#endif
```

8.53 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsTab.h File Reference

```
#include <QWidget>
#include <QItemSelection>
#include <QLineEdit>
#include <QSpinBox>
#include <QPushButton>
#include <QTableView>
#include <QSortFilterProxyModel>
#include <QSignalMapper>
#include <QPointer>
#include ¨PairsModel.h¨
#include ¨LevelsModel.h¨
#include ¨ChannelsModel.h¨
#include ¨ChannelDetails.h¨
```

Classes

- class LevelsTab

8.54 LevelsTab.h

Go to the documentation of this file.

```
#ifndef LEVELSTAB_H
#define LEVELSTAB_H

#include <QWidget>
#include <QItemSelection>
#include <QLineEdit>
#include <QSpinBox>
#include <QPushButton>
#include <QTableView>
#include <QSortFilterProxyModel>
#include <QSignalMapper>
#include <QPointer>
#include "PairsModel.h"
#include "LevelsModel.h"
#include "ChannelsModel.h"
#include "ChannelDetails.h"
```

```

class InfoDialog;

class LevelsTab : public QWidget {
  Q_OBJECT

 public:
  LevelsTab(QWidget *parent = 0);
  void setPairsModel(PairsModel*);
  void updateChannelsLevelAdded(int levelIndex);
  void updateChannelsLevelDeleted(int levelIndex);
  void updateChannelsLevelEdited(int levelIndex);
  QList<ChannelsData> calculateChannels(int levelIndex);
  bool writeNuclearFile(QTextStream& outStream);
  bool readNuclearFile(QTextStream& inStream);
  void reset();

 public slots:
  void addLevel();
  void addLevel(LevelsData level, bool fromFile);
  void removeLevel();
  void editLevel();
  void updateButtons(const QItemSelection &selection);
  void updateFilter(const QItemSelection &selection);
  void updateChannelsPairAddedEdited();
  void updateChannelsPairRemoved(int pairIndex);
  void updateDetails(const QItemSelection &selection);
  void updateReducedWidth(const QString &string);
  void showInfo(int which=0,QString title="");

 signals:
  void readNewPair(PairsData,int,bool);
  void readExistingPair(PairsData,int,bool);

 private:
  QSpinBox *maxLSpin;
  QSpinBox *maxMultSpin;
  QSpinBox *maxNumMultSpin;
  QPushButton *addLevelButton;
  QPushButton *removeLevelButton;
  PairsModel *pairsModel;
  LevelsModel *levelsModel;
  ChannelsModel *channelsModel;
  QTableView *levelsView;
  QTableView *channelsView;
  QSortFilterProxyModel *levelsModelProxy;
  QSortFilterProxyModel *proxyModel;
  ChannelDetails *channelDetails;
  QSignalMapper* mapper;
  QPushButton *infoButton[5];
  static const std::vector<QString> infoText;
  QPointer<InfoDialog> infoDialog[5];
};


#endif
```

8.55 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsModel.h File Reference

```
#include <QAbstractTableModel>
#include <QList>
```

Classes

- struct PairsData
- class PairsModel

8.56 PairsModel.h

Go to the documentation of this file.

```
#ifndef PAIRSMODEL_H
#define PAIRSMODEL_H

#include <QAbstractTableModel>
#include <QList>

struct PairsData {
  static const int SIZE = 15;
  double lightJ;
  int lightPi;
  int lightZ;
  double lightM;
  double lightG;
  double heavyJ;
  int heavyPi;
  int heavyZ;
  double heavyM;
  double heavyG;
  double excitationEnergy;
  double seperationEnergy;
  double channelRadius;
  int pairType;
  int ecMultMask;
};

class PairsModel : public QAbstractTableModel {
  Q_OBJECT

 public:
  PairsModel(QObject *parent = 0);

  int rowCount(const QModelIndex &parent) const;
  int columnCount(const QModelIndex &parent) const;
  QVariant data(const QModelIndex &index, int role) const;
  QVariant headerData(int section, Qt::Orientation orientation, int role) const;
  bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
  bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
  bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());

  int isPair(const PairsData &pair) const;
  int numPairs() const {return pairsList.size();};
  QList<PairsData> getPairs() const {return pairsList;};
  QString getParticleLabel(const PairsData &pair, int which=-1) const;
  QString getReactionLabel(const PairsData &firstPair, const PairsData &secondPair);
  QString getReactionLabelTotalCapture(const PairsData &firstPair);
  QString getSpinLabel(const PairsData &pair, int which) const;

 private:
  QList<PairsData> pairsList;
};

#endif
```

8.57 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsTab.h File Reference

```
#include <QWidget>
#include <QItemSelection>
#include <QSignalMapper>
#include <QTableView>
#include <QPushButton>
#include <QPointer>
#include "PairsModel.h"
#include "AddPairDialog.h"
```

Classes

- class PairsTab

8.58 PairsTab.h

Go to the documentation of this file.

```
#ifndef PAIRSTAB_H
#define PAIRSTAB_H

#include <QWidget>
#include <QItemSelection>
#include <QSignalMapper>
#include <QTableView>
#include <QPushButton>
#include <QSignalMapper>
#include <QPointer>

#include "PairsModel.h"
#include "AddPairDialog.h"

QT_BEGIN_NAMESPACE

class QPushButton;

QT_END_NAMESPACE

class InfoDialog;

class PairsTab : public QWidget {
  Q_OBJECT

 public:
  PairsTab(QWidget *parent = 0);
  PairsModel *getPairsModel();
  bool parseOldECSection(QTextStream&);

 public slots:
  void addPair();
  void addPair(PairsData pair,int pairIndex,bool fromFile);
  void editPair();
  void editPair(PairsData pair,int pairIndex,bool fromFile);
  void removePair();
  void updateButtons(const QItemSelection &selection);
  void showInfo(int which=0,QString title="");

 signals:
  void pairAdded(int);
  void pairRemoved(int);
  void pairEdited(int);

 private:
  PairsModel *pairsModel;
  QTableView *pairsView;
  QPushButton *addButton;
  QPushButton *deleteButton;
  QSignalMapper* mapper;
  QPushButton *infoButton[5];
  static const std::vector<QString> infoText;
  QPointer<InfoDialog> infoDialog[5];
};

#endif
```

8.59 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PlotTab.h File Reference

```
#include <QWidget>
#include <QSortFilterProxyModel>
#include <QSignalMapper>
#include <QPointer>
```

Classes

- class SegTestProxyModel
- class SegDataProxyModel
- class PlotTab

8.60 PlotTab.h

Go to the documentation of this file.

```
#ifndef PLOTTAB_H
#define PLOTTAB_H

#include <QWidget>
#include <QSortFilterProxyModel>
#include <QSignalMapper>
#include <QPointer>

class Config;
class AZUREPlot;
class PlotEntry;
class SegmentsDataModel;
class SegmentsTestModel;
class InfoDialog;

QT_BEGIN_NAMESPACE

class QRadioButton;
class QListView;
class QCheckBox;
class QPushButton;
class QComboBox;

QT_END_NAMESPACE

class SegTestProxyModel : public QSortFilterProxyModel {
 public:
  SegTestProxyModel(QWidget* parent = 0) : QSortFilterProxyModel(parent) {};
  QVariant data(const QModelIndex& index, int role = Qt::DisplayRole) const;
  bool filterAcceptsRow(int source_row, const QModelIndex &source_parent) const;
};

class SegDataProxyModel : public QSortFilterProxyModel {
 public:
  SegDataProxyModel(QWidget* parent = 0) : QSortFilterProxyModel(parent) {};
  QVariant data(const QModelIndex& index, int role = Qt::DisplayRole) const;
};

class PlotTab : public QWidget {

  Q_OBJECT

 public:
  PlotTab(Config& config, SegmentsDataModel* dataModel, SegmentsTestModel* testModel, QWidget* parent = 0);
  QList<PlotEntry*> getDataSegments();
  QList<PlotEntry*> getTestSegments();
  void reset();

 public slots:
  void draw();
  void xAxisTypeChanged();
  void yAxisTypeChanged();
  void xAxisLogScaleChanged(bool);
  void yAxisLogScaleChanged(bool);
  void showInfo(int which=0,QString title="");

 public:
  friend class AZUREPlot;

 private:
  Config& configure;
  AZUREPlot* azurePlot;
  QListView* dataSegmentSelectorList;
  QListView* testSegmentSelectorList;
  QRadioButton* yAxisXSButton;
  QRadioButton* yAxisSFButton;
  QComboBox* xAxisTypeCombo;
  QCheckBox* xAxisIsLogCheck;
  QCheckBox* yAxisIsLogCheck;
  SegTestProxyModel* segTestProxyModel;
  SegDataProxyModel* segDataProxyModel;
  QPushButton* refreshButton;
  QPushButton* exportButton;
  QPushButton* printButton;
  QSignalMapper* mapper;
  QPushButton *infoButton[5];
  static const std::vector<QString> infoText;
  QPointer<InfoDialog> infoDialog[5];
};

#endif
```

8.61 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RichText↩ Delegate.h File Reference

```
#include <QStyledItemDelegate>
```

Classes

- class RichTextDelegate

8.62 RichTextDelegate.h

Go to the documentation of this file.

```
#ifndef RICHTEXTDELEGATE_H
#define RICHTEXTDELEGATE_H

#include <QStyledItemDelegate>

class QPainter;

class RichTextDelegate : public QStyledItemDelegate {

 protected:
  void paint(QPainter *painter, const QStyleOptionViewItem &option, const QModelIndex &index) const;
  QSize sizeHint ( const QStyleOptionViewItem & option, const QModelIndex & index ) const;

};

#endif
```

8.63 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RunTab.h File Reference

```
#include <QWidget>
#include <QSignalMapper>
#include <QPointer>
```

Classes

- class RunTab

8.64 RunTab.h

Go to the documentation of this file.

```
#ifndef RUNTAB_H
#define RUNTAB_H

#include <QWidget>
#include <QSignalMapper>
#include <QPointer>

class ChooseFileButton;
class FilteredTextEdit;
class InfoDialog;
```

```

QT_BEGIN_NAMESPACE

class QComboBox;
class QPushButton;
class QRadioButton;
class QLineEdit;
class QTextEdit;
class QGroupBox;

QT_END_NAMESPACE

class RunTab : public QWidget {
  Q_OBJECT

 public:
  RunTab(QWidget* parent=0);
  friend class AZURESetup;
  friend class AZUREMainThread;
  void reset();

 public slots:
  void showInfo(int which=0, QString title="");

 private slots:
  void calculationTypeChanged(int index);
  void paramFileButtonChanged(bool checked);
  void integralsFileButtonChanged(bool checked);
  void fileTempButtonChanged(bool checked);
  void setChooseFile(QLineEdit* lineEdit);

 private:
  QComboBox* calcType;
  QPushButton* calcButton;
  QPushButton* stopAZUREButton;
  QLineEdit* paramFileText;
  QLineEdit* integralsFileText;
  QRadioButton* newParamFileButton;
  QRadioButton* oldParamFileButton;
  QGroupBox* integralsFileGroup;
  QRadioButton* newIntegralsFileButton;
  QRadioButton* oldIntegralsFileButton;
  FilteredTextEdit* runtimeText;
  QLineEdit* chiVarianceText;
  QGroupBox* rateParamsGroup;
  QRadioButton* gridTempButton;
  QRadioButton* fileTempButton;
  QLineEdit* rateEntranceKey;
  QLineEdit* rateExitKey;
  QLineEdit* minTempText;
  QLineEdit* maxTempText;
  QLineEdit* tempStepText;
  QLineEdit* fileTempText;
  ChooseFileButton* rateParamsChoose;
  ChooseFileButton* paramFileChoose;
  ChooseFileButton* integralsFileChoose;
  QSignalMapper* mapper;
  QPushButton *infoButton[5];
  static const std::vector<QString> infoText;
  QPointer<InfoDialog> infoDialog[5];
};



#endif
```

8.65 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsDataModel.h File Reference

```
#include <QAbstractTableModel>
#include <QList>
```

Classes

- struct SegmentsDataData
- class SegmentsDataModel

8.66 SegmentsDataModel.h

Go to the documentation of this file.

```
#ifndef SEGMENTSDATAMODEL_H
#define SEGMENTSDATAMODEL_H

#include <QAbstractTableModel>
#include <QList>

class PairsModel;

struct SegmentsDataData {
  static const int SIZE = 14;
  int isActive;
  int entrancePairIndex;
  int exitPairIndex;
  double lowEnergy;
  double highEnergy;
  double lowAngle;
  double highAngle;
  int dataType;
  QString dataFile;
  double dataNorm;
  double dataNormError;
  int varyNorm;
  double phaseJ;
  int phaseL;
};

class SegmentsDataModel : public QAbstractTableModel {
  Q_OBJECT

 public:
  SegmentsDataModel(QObject *parent = 0);

  int rowCount(const QModelIndex &parent) const;
  int columnCount(const QModelIndex &parent) const;
  QVariant data(const QModelIndex &index, int role) const;
  QVariant headerData(int section, Qt::Orientation orientation, int role) const;
  bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
  bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
  bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
  Qt::ItemFlags flags(const QModelIndex &index) const;
  int isSegDataLine(const SegmentsDataData &line) const;
  QList<SegmentsDataData> getLines() const {return segDataLineList;};
  void setPairsModel(PairsModel* model);
  QString getReactionLabel(const QModelIndex &index);

 private:
  QList<SegmentsDataData> segDataLineList;
  PairsModel* pairsModel;
};


#endif
```

8.67 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTab.h File Reference

```
#include <QWidget>
#include <QLineEdit>
#include <QItemSelection>
#include <QTableView>
#include <QPushButton>
#include <QSignalMapper>
#include <QPointer>
#include "SegmentsDataModel.h"
#include "SegmentsTestModel.h"
#include "AddSegDataDialog.h"
#include "AddSegTestDialog.h"
```

Classes

- class SegmentsTab

8.68 SegmentsTab.h

Go to the documentation of this file.

```
#ifndef SEGMENTSTAB_H
#define SEGMENTSTAB_H

#include <QWidget>
#include <QLineEdit>
#include <QItemSelection>
#include <QTableView>
#include <QPushButton>
#include <QSignalMapper>
#include <QPointer>
#include "SegmentsDataModel.h"
#include "SegmentsTestModel.h"
#include "AddSegDataDialog.h"
#include "AddSegTestDialog.h"

class InfoDialog;

class SegmentsTab : public QWidget {
  Q_OBJECT

 public:
  SegmentsTab(QWidget *parent = 0);
  SegmentsTestModel* getSegmentsTestModel();
  SegmentsDataModel* getSegmentsDataModel();
  void reset();
  /*QLineEdit *getSegDataFileText() const {return segDataFileText;};
    QLineEdit *getSegTestFileText() const {return segTestFileText;};*/

 public slots:
  void addSegDataLine();
  void addSegDataLine(SegmentsDataData line);
  void addSegTestLine();
  void addSegTestLine(SegmentsTestData line);
  void editSegDataLine();
  void editSegTestLine();
  void deleteSegDataLine();
  void deleteSegTestLine();
  void moveSegDataLineUp();
  void moveSegDataLineDown();
  void moveSegTestLineUp();
  void moveSegTestLineDown();
  void updateSegDataButtons(const QItemSelection &selection);
  void updateSegTestButtons(const QItemSelection &selection);
  /*void openSegDataFile();
  void openSegDataFile(QString filename);
  void saveSegDataFile();
  void saveAsSegDataFile();*/
  /*bool readSegDataFile(QString filename);*/
  bool readSegDataFile(QTextStream& inStream);
  /*bool writeSegDataFile(QString filename);*/
  bool writeSegDataFile(QTextStream& outStream);
  /*void openSegTestFile();
  void openSegTestFile(QString filename);
  void saveSegTestFile();
  void saveAsSegTestFile();*/
  /*bool readSegTestFile(QString filename);*/
  bool readSegTestFile(QTextStream& inStream);
  /*bool writeSegTestFile(QString filename);*/
  bool writeSegTestFile(QTextStream& outStream);
  void setPairsModel(PairsModel* model) {
    segmentsDataModel->setPairsModel(model);
    segmentsTestModel->setPairsModel(model);
  }
  void showInfo(int which=0,QString title="");

 private:
  void moveSegDataLine(unsigned int upDown);
  void moveSegTestLine(unsigned int upDown);

  /*QLineEdit *segDataFileText;*/
  SegmentsDataModel *segmentsDataModel;
  QTableView *segmentsDataView;
  QPushButton *segDataAddButton;
```

```
  //QPushButton *segDataEditButton;
  QPushButton *segDataDeleteButton;
  QPushButton *segDataUpButton;
  QPushButton *segDataDownButton;
  /*QLineEdit *segTestFileText;*/
  SegmentsTestModel *segmentsTestModel;
  QTableView *segmentsTestView;
  QPushButton *segTestAddButton;
  //QPushButton *segTestEditButton;
  QPushButton *segTestDeleteButton;
  QPushButton *segTestUpButton;
  QPushButton *segTestDownButton;
  QSignalMapper* mapper;
  QPushButton *infoButton[5];
  static const std::vector<QString> infoText;
  QPointer<InfoDialog> infoDialog[5];
};

#endif
```

8.69 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTest↩ Model.h File Reference

```
#include <QAbstractTableModel>
#include <QList>
```

Classes

- struct SegmentsTestData
- class SegmentsTestModel

8.70 SegmentsTestModel.h

Go to the documentation of this file.

```
#ifndef SEGMENTSTESTMODEL_H
#define SEGMENTSTESTMODEL_H

#include <QAbstractTableModel>
#include <QList>

class PairsModel;

struct SegmentsTestData {
  static const int SIZE = 13;
  int isActive;
  int entrancePairIndex;
  int exitPairIndex;
  double lowEnergy;
  double highEnergy;
  double energyStep;
  double lowAngle;
  double highAngle;
  double angleStep;
  int dataType;
  double phaseJ;
  int phaseL;
  int maxAngDistOrder;
};

class SegmentsTestModel : public QAbstractTableModel {
  Q_OBJECT

 public:
  SegmentsTestModel(QObject *parent = 0);

  int rowCount(const QModelIndex &parent) const;
  int columnCount(const QModelIndex &parent) const;
```

```
  QVariant data(const QModelIndex &index, int role) const;
  QVariant headerData(int section, Qt::Orientation orientation, int role) const;
  bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
  bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
  bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
  Qt::ItemFlags flags(const QModelIndex &index) const;
  int isSegTestLine(const SegmentsTestData &line) const;
  QList<SegmentsTestData> getLines() const {return segTestLineList;};
  void setPairsModel(PairsModel* model);
  QString getReactionLabel(const QModelIndex &index);
 private:
  QList<SegmentsTestData> segTestLineList;
  PairsModel* pairsModel;
};


#endif
```

8.71 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntModel.h File Reference

```
#include <QAbstractTableModel>
#include <QList>
```

Classes

- struct TargetIntData
- class TargetIntModel

Functions

- Q_DECLARE_METATYPE (QList< double >)

8.71.1 Function Documentation

8.71.1.1 Q_DECLARE_METATYPE()

```
Q_DECLARE_METATYPE (
            QList< double >  )
```

8.72 TargetIntModel.h

Go to the documentation of this file.

```
#ifndef TARGETINTMODEL_H
#define TARGETINTMODEL_H

#include <QAbstractTableModel>
#include <QList>

Q_DECLARE_METATYPE(QList<double>);

struct TargetIntData {
  static const int SIZE = 12;
  int isActive;
  QString segmentsList;
  int numPoints;
  bool isConvolution;
```

```
 double sigma;
 bool isTargetIntegration;
 double density;
 QString stoppingPowerEq;
 int numParameters;
 QList<double> parameters;
 bool isQCoefficients;
 QList<double> qCoefficients;
};

class TargetIntModel : public QAbstractTableModel {
Q_OBJECT

public:
 TargetIntModel(QObject *parent = 0);
 int rowCount(const QModelIndex &parent) const;
 int columnCount(const QModelIndex &parent) const;
 QVariant data(const QModelIndex &index, int role) const;
 QVariant headerData(int section, Qt::Orientation orientation, int role) const;
 bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
 bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
 bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
 Qt::ItemFlags flags(const QModelIndex &index) const;
 QList<TargetIntData> getLines() const {return targetIntList;};
private:
 QList<TargetIntData> targetIntList;
};

#endif
```

8.73 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntTab.h File Reference

```
#include <QWidget>
#include <QItemSelection>
#include <QTableView>
#include <QSignalMapper>
#include <QPointer>
#include "TargetIntModel.h"
#include "AddTargetIntDialog.h"
```

Classes

- class TargetIntTab

8.74 TargetIntTab.h

Go to the documentation of this file.

```
#ifndef TARGETINTTAB_H
#define TARGETINTTAB_H

#include <QWidget>
#include <QItemSelection>
#include <QTableView>
#include <QSignalMapper>
#include <QPointer>
#include "TargetIntModel.h"
#include "AddTargetIntDialog.h"

QT_BEGIN_NAMESPACE

class QPushButton;
class QLineEdit;

QT_END_NAMESPACE

```

```
class InfoDialog;

class TargetIntTab : public QWidget {
  Q_OBJECT

 public:
  TargetIntTab(QWidget *parent = 0);
  TargetIntModel* getTargetIntModel();
  bool writeFile(QTextStream& outStream);
  bool readFile(QTextStream& inStream);
  void reset();

 public slots:
  void addLine();
  void addLine(TargetIntData line);
  void editLine();
  void deleteLine();
  void updateButtons(const QItemSelection &selection);
  void showInfo(int which=0,QString title="");

 private:
  TargetIntModel *targetIntModel;
  QTableView *targetIntView;
  QPushButton *addButton;
  QPushButton *deleteButton;
  QSignalMapper* mapper;
  QPushButton *infoButton[5];
  static const std::vector<QString> infoText;
  QPointer<InfoDialog> infoDialog[5];
};

#endif
```

8.75 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TextEditBuffer.h File Reference

```
#include <QWidget>
#include <QString>
#include <string>
#include <streambuf>
#include <assert.h>
#include <iostream>
```

Classes

- class TextEditBuffer

8.76 TextEditBuffer.h

Go to the documentation of this file.

```
#ifndef TEXTEDITBUFFER_H
#define TEXTEDITBUFFER_H

#include <QWidget>
#include <QString>
#include <string>
#include <streambuf>
#include <assert.h>
#include <iostream>

class QTextEdit;

class TextEditBuffer : public QWidget, public std::streambuf {

Q_OBJECT

```

```
public:
 TextEditBuffer(std::size_t buff_size = 256, QWidget* parent=0) : QWidget(parent), buffer_(buff_size+1) {
   char* base = &buffer_.front();
   setp(base, base + buffer_.size() -1);
 };
signals:
 void updateLog(QString);

protected:
 virtual int_type overflow(int_type ch) {
   if(ch !=traits_type::eof()){
     assert(std::less_equal<char*>()(pptr(),epptr()));
     *pptr()=ch;
     pbump(1);
     if(writeToTextEdit()) return traits_type::to_int_type(ch);
   }
   return traits_type::eof();
 };
 virtual int sync() {
   return writeToTextEdit() ? 0 : -1;
 };
private:
 bool writeToTextEdit() {
   std::string tempString(pbase(),pptr());
   pbump(pbase()-pptr());
   if(!tempString.empty()) emit updateLog(QString::fromStdString(tempString));
   return true;
 };
private:
 std::vector<char> buffer_;
};

#endif
```

8.77 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AboutAZURE2Dialog.cpp File Reference

```
#include <QLabel>
#include <QPushButton>
#include <QVBoxLayout>
#include "AboutAZURE2Dialog.h"
```

8.78 AboutAZURE2Dialog.cpp

Go to the documentation of this file.

```
#include <QLabel>
#include <QPushButton>
#include <QVBoxLayout>

#include "AboutAZURE2Dialog.h"

AboutAZURE2Dialog::AboutAZURE2Dialog(QWidget *parent) : QDialog(parent) {
  setWindowTitle(tr("About AZURE2"));

  QLabel *label = new QLabel("<center><img src=\":/azure-icon.png\" width=\"128\" height=\"128\" /><br/>"
                  "<b>AZURE2</b><br/>v1.0.0<br/><br/>"
                  "E. Uberseder, R.J. deBoer, R.E. Azuma<br/>"
                  "Joint Institute For Nuclear Astrophysics (JINA)</center>",
                  this);
  QPushButton*  okButton = new QPushButton(tr("OK"),this);
  okButton->setMaximumSize(80,30);

  QVBoxLayout* layout = new QVBoxLayout(this);

  layout->addWidget(label);
  layout->addWidget(okButton);
  layout->setAlignment(okButton,Qt::AlignHCenter);

  connect(okButton,SIGNAL(clicked()),this,SLOT(close()));

  setLayout(layout);
}
```

8.79 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddLevelDialog.cpp File Reference

#include ¨AddLevelDialog.h¨

8.80 AddLevelDialog.cpp

Go to the documentation of this file.

```
#include "AddLevelDialog.h"

AddLevelDialog::AddLevelDialog(QWidget *parent) : QDialog(parent) {

  this->setMaximumSize(250,150);
  //this->setMinimumSize(220,110);

  QRegExp rx("^\\d{0,2}(\\.[05]{0,1})?$");
  QValidator *validator = new QRegExpValidator(rx, this);
  jValueLabel = new QLabel(tr("Level Spin:"));
  jValueText = new QLineEdit;
  jValueText->setValidator(validator);
  piValueCombo = new QComboBox;
  piValueCombo->addItem("-");
  piValueCombo->addItem("+");

  energyLabel = new QLabel(tr("Excitation Energy [MeV]:"));
  energyText = new QLineEdit;

  cancelButton = new QPushButton(tr("Cancel"));
  okButton = new QPushButton(tr("Accept"));
  okButton->setDefault(true);

  QHBoxLayout *energyLayout = new QHBoxLayout;
  energyLayout->addWidget(energyLabel);
  energyLayout->addWidget(energyText);

  QGridLayout *spinLayout = new QGridLayout;
  spinLayout->addWidget(jValueLabel,0,0);
  spinLayout->addWidget(jValueText,0,1);
  spinLayout->addWidget(piValueCombo,0,2);
  spinLayout->setColumnStretch(1,1);

  QHBoxLayout *buttonBox = new QHBoxLayout;
  buttonBox->addWidget(cancelButton);
  buttonBox->addWidget(okButton);

  QVBoxLayout *mainLayout = new QVBoxLayout;
  mainLayout->addLayout(energyLayout);
  mainLayout->addLayout(spinLayout);
  mainLayout->addLayout(buttonBox);

  setLayout(mainLayout);

  connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
  connect(cancelButton,SIGNAL(clicked()),this,SLOT(reject()));

  setWindowTitle(tr("Add a Level"));
}
```

8.81 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddPairDialog.cpp File Reference

#include <QComboBox>
#include <QGridLayout>
#include <QSpacerItem>
#include <QLabel>
#include <QLineEdit>

```
#include <QPushButton>
#include <QGroupBox>
#include <QCheckBox>
#include ¨AddPairDialog.h¨
```

8.82 AddPairDialog.cpp

Go to the documentation of this file.

```
#include <QComboBox>
#include <QGridLayout>
#include <QSpacerItem>
#include <QLabel>
#include <QLineEdit>
#include <QPushButton>
#include <QGroupBox>
#include <QCheckBox>


#include "AddPairDialog.h"

AddPairDialog::AddPairDialog(QWidget *parent) : QDialog(parent) {

  //this->setMaximumSize(370,440);
  //this->setMinimumSize(370,440);
  this->setMaximumWidth(370);
  this->setMinimumWidth(370);

  excitationEnergyLabel = new QLabel(tr("Excitation Energy [MeV]:"));
  excitationEnergyText = new QLineEdit;
  seperationEnergyLabel = new QLabel(tr("Separation Energy [MeV]:"));
  seperationEnergyText = new QLineEdit;
  channelRadiusLabel = new QLabel(tr("Channel Radius [fm]:"));
  channelRadiusText = new QLineEdit;
  pairTypeLabel = new QLabel(tr("Particle Pair Type:"));
  pairTypeCombo = new QComboBox;
  pairTypeCombo->addItem(tr("Particle, Particle"));
  pairTypeCombo->addItem(tr("Particle, Gamma"));
  pairTypeCombo->addItem(tr("Beta Decay"));
  connect(pairTypeCombo,SIGNAL(currentIndexChanged(int)),this,SLOT(updateLightParticle(int)));

  QRegExp rx("^\\d{0,2}(\\.[05]{0,1})?$");
  QValidator *validator = new QRegExpValidator(rx, this);
  lightJLabel = new QLabel(tr("J:"));
  lightJText = new QLineEdit;
  lightJText->setValidator(validator);
  lightPiLabel = new QLabel(tr("Pi:"));
  lightPiCombo = new QComboBox;
  lightPiCombo->addItem(tr("-"));
  lightPiCombo->addItem(tr("+"));
  lightZLabel = new QLabel(tr("Z:"));
  lightZText = new QLineEdit;
  lightMLabel = new QLabel(tr("M:"));
  lightMText = new QLineEdit;
  //lightGLabel = new QLabel(tr("g:"));
  //lightGText = new QLineEdit;

  heavyJLabel = new QLabel(tr("J:"));
  heavyJText = new QLineEdit;
  heavyJText->setValidator(validator);
  heavyPiLabel = new QLabel(tr("Pi:"));
  heavyPiCombo = new QComboBox;
  heavyPiCombo->addItem(tr("-"));
  heavyPiCombo->addItem(tr("+"));
  heavyZLabel = new QLabel(tr("Z:"));
  heavyZText = new QLineEdit;
  heavyMLabel = new QLabel(tr("M:"));
  heavyMText = new QLineEdit;
  //heavyGLabel = new QLabel(tr("g:"));
  //heavyGText = new QLineEdit;

  e1Check = new QCheckBox(tr("E1"));
  e1Check->setChecked(false);
  //m1Check = new QCheckBox(tr("M1"));
  //m1Check->setChecked(false);
  e2Check = new QCheckBox(tr("E2"));
  e2Check->setChecked(false);

  cancelButton = new QPushButton(tr("Cancel"));
```

```
  okButton = new QPushButton(tr("Accept"));
  okButton->setDefault(true);


  QGridLayout *pairTypeLayout = new QGridLayout;
  pairTypeLayout->addWidget(pairTypeLabel,0,0);
  pairTypeLayout->addWidget(pairTypeCombo,0,1);
  pairTypeLayout->setColumnStretch(1,1);

  QGroupBox *channelGroup = new QGroupBox(tr("Channel Properties"));
  QGridLayout *channelLayout = new QGridLayout;
  channelLayout->addWidget(excitationEnergyLabel,0,0,Qt::AlignRight);
  channelLayout->addWidget(excitationEnergyText,0,1);
  channelLayout->addWidget(seperationEnergyLabel,1,0,Qt::AlignRight);
  channelLayout->addWidget(seperationEnergyText,1,1);
  channelLayout->addWidget(channelRadiusLabel,2,0,Qt::AlignRight);
  channelLayout->addWidget(channelRadiusText,2,1);
  channelGroup->setLayout(channelLayout);

  QGroupBox *lightGroup = new QGroupBox(tr("Light Particle"));
  QGridLayout *lightLayout = new QGridLayout;
  lightLayout->addWidget(lightJLabel,0,0,Qt::AlignRight);
  QHBoxLayout *lightSpinLayout = new QHBoxLayout;
  lightSpinLayout->addWidget(lightJText);
  lightSpinLayout->addWidget(lightPiCombo);
  lightLayout->addLayout(lightSpinLayout,0,1);
  lightLayout->addWidget(lightZLabel,2,0,Qt::AlignRight);
  lightLayout->addWidget(lightZText,2,1);
  lightLayout->addWidget(lightMLabel,3,0,Qt::AlignRight);
  lightLayout->addWidget(lightMText,3,1);
  //lightLayout->addWidget(lightGLabel,4,0,Qt::AlignRight);
  //lightLayout->addWidget(lightGText,4,1);
  lightGroup->setLayout(lightLayout);

  QGroupBox *heavyGroup = new QGroupBox(tr("Heavy Particle"));
  QGridLayout *heavyLayout = new QGridLayout;
  heavyLayout->addWidget(heavyJLabel,0,0,Qt::AlignRight);
  QHBoxLayout *heavySpinLayout = new QHBoxLayout;
  heavySpinLayout->addWidget(heavyJText);
  heavySpinLayout->addWidget(heavyPiCombo);
  heavyLayout->addLayout(heavySpinLayout,0,1);
  heavyLayout->addWidget(heavyZLabel,2,0,Qt::AlignRight);
  heavyLayout->addWidget(heavyZText,2,1);
  heavyLayout->addWidget(heavyMLabel,3,0,Qt::AlignRight);
  heavyLayout->addWidget(heavyMText,3,1);
  //heavyLayout->addWidget(heavyGLabel,4,0,Qt::AlignRight);
  //heavyLayout->addWidget(heavyGText,4,1);
  heavyGroup->setLayout(heavyLayout);

  QHBoxLayout *entryLayout = new QHBoxLayout;
  entryLayout->addWidget(lightGroup);
  entryLayout->addWidget(heavyGroup);

  QHBoxLayout *buttonBox = new QHBoxLayout;
  buttonBox->addWidget(cancelButton);
  buttonBox->addWidget(okButton);

  multBox= new QGroupBox(tr("External Capture Multipolarities"));
  multBox->hide();
  QHBoxLayout *multLayout = new QHBoxLayout;
  multLayout->addWidget(e1Check);
  //multLayout->addWidget(m1Check);
  multLayout->addWidget(e2Check);
  multBox->setLayout(multLayout);

  QVBoxLayout *mainLayout = new QVBoxLayout;
  mainLayout->addLayout(pairTypeLayout);
  mainLayout->addLayout(entryLayout);
  mainLayout->addWidget(channelGroup);
  mainLayout->addWidget(multBox);
  mainLayout->addLayout(buttonBox);

  setLayout(mainLayout);

  connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
  connect(cancelButton,SIGNAL(clicked()),this,SLOT(reject()));

  setWindowTitle(tr("Add a Particle Pair"));
}

void AddPairDialog::updateLightParticle(int index) {
  if(index==1) {
    lightJText->setText("1.0");
    lightJText->setEnabled(false);
    lightPiCombo->setCurrentIndex(1);
    lightPiCombo->setEnabled(false);
    lightZText->setText("0");
```

```
    lightZText->setEnabled(false);
    lightMText->setText("0.0");
    lightMText->setEnabled(false);
    //lightGText->setText("0.0");
    //lightGText->setEnabled(false);
    seperationEnergyText->setText("0.0");
    seperationEnergyText->setEnabled(false);
    excitationEnergyText->setEnabled(true);
    channelRadiusText->setText("0");
    channelRadiusText->setEnabled(false);
    multBox->show();
  } else if(index==2) {
    lightJText->setEnabled(false);
    lightJText->setText("0.5");
    lightPiCombo->setEnabled(false);
    lightPiCombo->setCurrentIndex(1);
    lightZText->setEnabled(true);
    lightMText->setEnabled(false);
    lightMText->setText("0.0005");
    //lightGText->setEnabled(false);
    //lightGText->setText("2.0023");
    seperationEnergyText->setEnabled(true);
    excitationEnergyText->setEnabled(false);
    excitationEnergyText->setText("0.000");
    channelRadiusText->setEnabled(true);
    multBox->hide();
    this->adjustSize();
  } else {
    lightJText->setEnabled(true);
    lightPiCombo->setEnabled(true);
    lightZText->setEnabled(true);
    lightMText->setEnabled(true);
    //lightGText->setEnabled(true);
    seperationEnergyText->setEnabled(true);
    excitationEnergyText->setEnabled(true);
    channelRadiusText->setEnabled(true);
    multBox->hide();
    this->adjustSize();
  }
}
```

8.83 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddSegDataDialog.cpp File Reference

```
#include "AddSegDataDialog.h"
#include <QPushButton>
#include <QGroupBox>
#include <QGridLayout>
#include <QFileDialog>
```

8.84 AddSegDataDialog.cpp

Go to the documentation of this file.

```
#include "AddSegDataDialog.h"

#include <QPushButton>
#include <QGroupBox>
#include <QGridLayout>
#include <QFileDialog>

AddSegDataDialog::AddSegDataDialog(QWidget *parent) : QDialog(parent) {

  //  this->setMaximumSize(370,420);
  //  this->setMinimumSize(370,420);

  entrancePairIndexSpin = new QSpinBox;
  entrancePairIndexSpin->setMinimum(1);
  entrancePairIndexSpin->setMaximum(100);
  entrancePairIndexSpin->setSingleStep(1);
  exitPairIndexSpin = new QSpinBox;
```

```
exitPairIndexSpin->setMinimum(1);
exitPairIndexSpin->setMaximum(100);
exitPairIndexSpin->setSingleStep(1);
lowEnergyText = new QLineEdit;
highEnergyText = new QLineEdit;
lowAngleText = new QLineEdit;
lowAngleText->setText("0");
lowAngleText->setEnabled(false);
highAngleText = new QLineEdit;
highAngleText->setText("180");
highAngleText->setEnabled(false);
dataTypeCombo = new QComboBox;
dataTypeCombo->addItem(tr("Angle Integrated"));
dataTypeCombo->addItem(tr("Differential"));
dataTypeCombo->addItem(tr("Phase Shift"));
dataTypeCombo->addItem(tr("Angle Integrated Total Capture"));
connect(dataTypeCombo,SIGNAL(currentIndexChanged(int)),this,SLOT(dataTypeChanged(int)));
QRegExp spinRX("^\\d{0,2}(\\.[05]{0,1})?$");
QValidator *spinValidator = new QRegExpValidator(spinRX, this);
phaseJValueText = new QLineEdit;
phaseJValueText->setValidator(spinValidator);
phaseJValueText->setVisible(false);
phaseJValueText->setMaximumWidth(50);
QRegExp intRX("^[0-6]$");
QValidator *intValidator = new QRegExpValidator(intRX, this);
phaseLValueText = new QLineEdit;
phaseLValueText->setValidator(intValidator);
phaseLValueText->setVisible(false);
phaseLValueText->setMaximumWidth(50);
dataFileText = new QLineEdit;
QPushButton *chooseFileButton = new QPushButton(tr("Choose..."));
connect(chooseFileButton,SIGNAL(clicked()),this,SLOT(setChooseFile()));
dataNormText = new QLineEdit;
dataNormText->setText("1.0");
dataNormErrorLabel = new QLabel(tr("Norm. Error [%]:"));
//  dataNormErrorLabel->setVisible(false);
dataNormErrorText = new QLineEdit(this);
//  dataNormErrorText->setVisible(false);
dataNormErrorText->setText("0.0");
dataNormErrorText->setMaximumWidth(50);
varyNormCheck = new QCheckBox(tr("Vary Norm?"));
//connect(varyNormCheck,SIGNAL(stateChanged(int)),this,SLOT(varyNormChanged(int)));

cancelButton = new QPushButton(tr("Cancel"));
okButton = new QPushButton(tr("Accept"));
okButton->setDefault(true);

QGroupBox *valueBox = new QGroupBox;
QGridLayout *valueLayout = new QGridLayout;
QGridLayout *pairLayout = new QGridLayout;
pairLayout->addWidget(new QLabel(tr("Entrance Pair Key:")),0,0,Qt::AlignRight);
pairLayout->addWidget(entrancePairIndexSpin,0,1);
pairLayout->addWidget(new QLabel(tr("Exit Pair Key:")),0,2,Qt::AlignRight);
pairLayout->addWidget(exitPairIndexSpin,0,3);
totalCaptureLabel = new QLabel(tr("Total Capture"));
totalCaptureLabel->setVisible(false);
pairLayout->addWidget(totalCaptureLabel,0,4);
valueLayout->addLayout(pairLayout,0,0,1,2);
QGroupBox* energyBox = new QGroupBox(tr("Lab Energy [MeV]"));
QGridLayout *energyLayout = new QGridLayout;
energyLayout->addWidget(new QLabel(tr("Low Energy:")),0,0,Qt::AlignRight);
energyLayout->addWidget(lowEnergyText,0,1);
energyLayout->addWidget(new QLabel(tr("High Energy:")),1,0,Qt::AlignRight);
energyLayout->addWidget(highEnergyText,1,1);
energyBox->setLayout(energyLayout);
valueLayout->addWidget(energyBox,1,0);
QGroupBox* angleBox = new QGroupBox(tr("Lab Angle [degrees]"));
QGridLayout *angleLayout = new QGridLayout;
angleLayout->addWidget(new QLabel(tr("Low Angle:")),0,0,Qt::AlignRight);
angleLayout->addWidget(lowAngleText,0,1);
angleLayout->addWidget(new QLabel(tr("High Angle:")),1,0,Qt::AlignRight);
angleLayout->addWidget(highAngleText,1,1);
angleBox->setLayout(angleLayout);
valueLayout->addWidget(angleBox,1,1);

QGridLayout* lowerLayout = new QGridLayout;
lowerLayout->addWidget(new QLabel(tr("Data Type:")),0,0,Qt::AlignRight);
lowerLayout->addWidget(dataTypeCombo,0,1);

QGridLayout* phaseLayout = new QGridLayout;
phaseLayout->addItem(new QSpacerItem(1,25),0,0);
phaseLayout->setColumnStretch(0,1);
phaseJValueLabel = new QLabel(tr("J:"));
phaseJValueLabel->setVisible(false);
phaseLayout->addWidget(phaseJValueLabel,0,1);
phaseLayout->addWidget(phaseJValueText,0,2);
phaseLValueLabel = new QLabel(tr("l:"));
```

```
  phaseLValueLabel->setVisible(false);
  phaseLayout->addWidget(phaseLValueLabel,0,3);
  phaseLayout->addWidget(phaseLValueText,0,4);
  lowerLayout->addLayout(phaseLayout,0,2);

  lowerLayout->addWidget(new QLabel(tr("Data Norm.:")),1,0,Qt::AlignRight);
  QHBoxLayout *normLayout = new QHBoxLayout;
  normLayout->addWidget(dataNormText);
  normLayout->addWidget(varyNormCheck);
  lowerLayout->addLayout(normLayout,1,1);
  QGridLayout *normErrorLayout = new QGridLayout;
  normErrorLayout->addItem(new QSpacerItem(1,25),0,0);
  normErrorLayout->setColumnStretch(0,1);
  normErrorLayout->addWidget(dataNormErrorLabel,0,1,Qt::AlignRight);
  normErrorLayout->addWidget(dataNormErrorText,0,2);
  lowerLayout->addLayout(normErrorLayout,1,2);

  lowerLayout->addWidget(new QLabel(tr("Data File:")),2,0,Qt::AlignRight);
  QGridLayout *fileLayout = new QGridLayout;
  fileLayout->addWidget(dataFileText,0,0);
  fileLayout->addWidget(chooseFileButton,0,1);
  fileLayout->setColumnStretch(0,1);
  lowerLayout->addLayout(fileLayout,2,1,1,2);
  valueLayout->addLayout(lowerLayout,2,0,1,2);
  valueBox->setLayout(valueLayout);

  QHBoxLayout *buttonBox = new QHBoxLayout;
  buttonBox->addWidget(cancelButton);
  buttonBox->addWidget(okButton);

  QVBoxLayout *mainLayout = new QVBoxLayout;
  mainLayout->addWidget(valueBox);
  mainLayout->addLayout(buttonBox);

  setLayout(mainLayout);

  connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
  connect(cancelButton,SIGNAL(clicked()),this,SLOT(reject()));

  setWindowTitle(tr("Add a Segment From Data"));
}

void AddSegDataDialog::setChooseFile() {
  QString filename = QFileDialog::getOpenFileName(this);
  if(!filename.isEmpty()) {
    dataFileText->setText(QDir::fromNativeSeparators(filename));
  }
}

void AddSegDataDialog::dataTypeChanged(int index) {
  if(index==2) {
    phaseJValueLabel->setVisible(true);
    phaseLValueLabel->setVisible(true);
    phaseJValueText->setVisible(true);
    phaseLValueText->setVisible(true);
  } else {
    phaseJValueLabel->setVisible(false);
    phaseLValueLabel->setVisible(false);
    phaseJValueText->setVisible(false);
    phaseLValueText->setVisible(false);
  }
  if(index==1) {
    lowAngleText->setEnabled(true);
    highAngleText->setEnabled(true);
  } else {
    lowAngleText->setEnabled(false);
    highAngleText->setEnabled(false);
    lowAngleText->setText("0");
    highAngleText->setText("180");
  }
  if(index==3) {
    exitPairIndexSpin->setVisible(false);
    totalCaptureLabel->setVisible(true);
  } else {
    totalCaptureLabel->setVisible(false);
    exitPairIndexSpin->setVisible(true);
  }
}

void AddSegDataDialog::varyNormChanged(int state) {
  if(state==Qt::Checked) {
    dataNormErrorLabel->setVisible(true);
    dataNormErrorText->setVisible(true);
  } else {
    dataNormErrorLabel->setVisible(false);
    dataNormErrorText->setVisible(false);
  }
```

```
}
```

8.85 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddSegTestDialog.cpp File Reference

```
#include ¨AddSegTestDialog.h¨
#include <QPushButton>
#include <QGroupBox>
#include <QGridLayout>
#include <QHBoxLayout>
```

8.86 AddSegTestDialog.cpp

Go to the documentation of this file.

```
#include "AddSegTestDialog.h"

#include <QPushButton>
#include <QGroupBox>
#include <QGridLayout>
#include <QHBoxLayout>


AddSegTestDialog::AddSegTestDialog(QWidget *parent) : QDialog(parent) {

  //  this->setMaximumSize(370,420);
  //  this->setMinimumSize(370,420);

  entrancePairIndexSpin = new QSpinBox;
  entrancePairIndexSpin->setMinimum(1);
  entrancePairIndexSpin->setMaximum(100);
  entrancePairIndexSpin->setSingleStep(1);
  exitPairIndexSpin = new QSpinBox;
  exitPairIndexSpin->setMinimum(1);
  exitPairIndexSpin->setMaximum(100);
  exitPairIndexSpin->setSingleStep(1);
  lowEnergyText = new QLineEdit;
  highEnergyText = new QLineEdit;
  energyStepText = new QLineEdit;
  lowAngleText = new QLineEdit;
  lowAngleText->setText("0");
  lowAngleText->setEnabled(false);
  highAngleText = new QLineEdit;
  highAngleText->setText("0");
  highAngleText->setEnabled(false);
  angleStepText = new QLineEdit;
  angleStepText->setText("0");
  angleStepText->setEnabled(false);
  dataTypeCombo = new QComboBox;
  dataTypeCombo->addItem(tr("Angle Integrated"));
  dataTypeCombo->addItem(tr("Differential"));
  dataTypeCombo->addItem(tr("Phase Shift"));
  dataTypeCombo->addItem(tr("Angular Distribution Coefficients"));
  dataTypeCombo->addItem(tr("Angle Integrated Total Capture"));
  connect(dataTypeCombo,SIGNAL(currentIndexChanged(int)),this,SLOT(dataTypeChanged(int)));
  QRegExp spinRX("^\\d{0,2}(\\.[05]{0,1})?$");
  QValidator *spinValidator = new QRegExpValidator(spinRX, this);
  phaseJValueText = new QLineEdit;
  phaseJValueText->setValidator(spinValidator);
  phaseJValueText->setVisible(false);
  phaseJValueText->setMaximumWidth(50);
  QRegExp intRX("^[0-6]$");
  QValidator *intValidator = new QRegExpValidator(intRX, this);
  phaseLValueText = new QLineEdit;
  phaseLValueText->setValidator(intValidator);
  phaseLValueText->setVisible(false);
  phaseLValueText->setMaximumWidth(50);

  cancelButton = new QPushButton(tr("Cancel"));
  okButton = new QPushButton(tr("Accept"));
  okButton->setDefault(true);
```

```

  QGroupBox *valueBox = new QGroupBox;
  QGridLayout *valueLayout = new QGridLayout;
  QGridLayout *pairLayout = new QGridLayout;
  pairLayout->addWidget(new QLabel(tr("Entrance Pair Key:")),0,0,Qt::AlignRight);
  pairLayout->addWidget(entrancePairIndexSpin,0,1);
  pairLayout->addWidget(new QLabel(tr("Exit Pair Key:")),0,2,Qt::AlignRight);
  pairLayout->addWidget(exitPairIndexSpin,0,3);
  totalCaptureLabel = new QLabel(tr("Total Capture"));
  totalCaptureLabel->setVisible(false);
  pairLayout->addWidget(totalCaptureLabel,0,4);
  valueLayout->addLayout(pairLayout,0,0,1,2);
  QGroupBox *energyBox = new QGroupBox(tr("Lab Energy [MeV]"));
  QGridLayout *energyLayout = new QGridLayout;
  energyLayout->addWidget(new QLabel(tr("Low Energy:")),0,0,Qt::AlignRight);
  energyLayout->addWidget(lowEnergyText,0,1);
  energyLayout->addWidget(new QLabel(tr("High Energy:")),1,0,Qt::AlignRight);
  energyLayout->addWidget(highEnergyText,1,1);
  energyLayout->addWidget(new QLabel(tr("Energy Step:")),2,0,Qt::AlignRight);
  energyLayout->addWidget(energyStepText,2,1);
  energyBox->setLayout(energyLayout);
  valueLayout->addWidget(energyBox,1,0);
  QGroupBox *angleBox = new QGroupBox(tr("Lab Angle [degrees]"));
  QGridLayout *angleLayout = new QGridLayout;
  angleLayout->addWidget(new QLabel(tr("Low Angle:")),0,0,Qt::AlignRight);
  angleLayout->addWidget(lowAngleText,0,1);
  angleLayout->addWidget(new QLabel(tr("High Angle:")),1,0,Qt::AlignRight);
  angleLayout->addWidget(highAngleText,1,1);
  angleLayout->addWidget(new QLabel(tr("Angle Step:")),2,0,Qt::AlignRight);
  angleLayout->addWidget(angleStepText,2,1);
  angleBox->setLayout(angleLayout);
  valueLayout->addWidget(angleBox,1,1);

  QGridLayout* lowerLayout = new QGridLayout;
  lowerLayout->addWidget(new QLabel(tr("Data Type:")),0,0,Qt::AlignRight);
  lowerLayout->addWidget(dataTypeCombo,0,1);
  lowerLayout->addItem(new QSpacerItem(1,25),0,2);
  lowerLayout->setColumnStretch(2,1);

  QHBoxLayout* phaseLayout = new QHBoxLayout;
  phaseJValueLabel = new QLabel(tr("J:"));
  phaseJValueLabel->setVisible(false);
  phaseLayout->addWidget(phaseJValueLabel);
  phaseLayout->addWidget(phaseJValueText);
  phaseLValueLabel = new QLabel(tr("l:"));
  phaseLValueLabel->setVisible(false);
  phaseLayout->addWidget(phaseLValueLabel);
  phaseLayout->addWidget(phaseLValueText);
  angDistLabel = new QLabel(tr("Maximum Order"));
  angDistSpin = new QSpinBox;
  angDistSpin->setMinimum(0);
  angDistSpin->setMaximum(10);
  angDistSpin->setSingleStep(1);
  angDistLabel->setVisible(false);
  angDistSpin->setVisible(false);
  phaseLayout->addWidget(angDistLabel);
  phaseLayout->addWidget(angDistSpin);
  lowerLayout->addLayout(phaseLayout,0,3);

  valueLayout->addLayout(lowerLayout,2,0,1,2);
  valueBox->setLayout(valueLayout);

  QHBoxLayout *buttonBox = new QHBoxLayout;
  buttonBox->addWidget(cancelButton);
  buttonBox->addWidget(okButton);

  QVBoxLayout *mainLayout = new QVBoxLayout;
  mainLayout->addWidget(valueBox);
  mainLayout->addLayout(buttonBox);

  setLayout(mainLayout);

  connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
  connect(cancelButton,SIGNAL(clicked()),this,SLOT(reject()));

  setWindowTitle(tr("Add a Segment Without Data"));
}

void AddSegTestDialog::dataTypeChanged(int index) {
  if(index==2) {
    angDistLabel->setVisible(false);
    angDistSpin->setVisible(false);
    phaseJValueLabel->setVisible(true);
    phaseLValueLabel->setVisible(true);
    phaseJValueText->setVisible(true);
    phaseLValueText->setVisible(true);
  } else if(index==3) {
```

```
    phaseJValueLabel->setVisible(false);
    phaseLValueLabel->setVisible(false);
    phaseJValueText->setVisible(false);
    phaseLValueText->setVisible(false);
    angDistLabel->setVisible(true);
    angDistSpin->setVisible(true);
  } else {
    phaseJValueLabel->setVisible(false);
    phaseLValueLabel->setVisible(false);
    phaseJValueText->setVisible(false);
    phaseLValueText->setVisible(false);
    angDistLabel->setVisible(false);
    angDistSpin->setVisible(false);
  }
  if(index==1) {
    lowAngleText->setEnabled(true);
    highAngleText->setEnabled(true);
    angleStepText->setEnabled(true);
  } else {
    lowAngleText->setEnabled(false);
    highAngleText->setEnabled(false);
    angleStepText->setEnabled(false);
    lowAngleText->setText("0");
    highAngleText->setText("0");
    angleStepText->setText("0");
  }
  if(index==4) {
    exitPairIndexSpin->setVisible(false);
    totalCaptureLabel->setVisible(true);
  } else {
    totalCaptureLabel->setVisible(false);
    exitPairIndexSpin->setVisible(true);
  }
}
```

8.87 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddTargetIntDialog.cpp File Reference

```
#include <QLineEdit>
#include <QSpinBox>
#include <QCheckBox>
#include <QTableWidget>
#include <QHBoxLayout>
#include <QLabel>
#include <QGroupBox>
#include <QHeaderView>
#include <QPushButton>
#include ¨AddTargetIntDialog.h¨
```

8.88 AddTargetIntDialog.cpp

Go to the documentation of this file.

```
#include <QLineEdit>
#include <QSpinBox>
#include <QCheckBox>
#include <QTableWidget>
#include <QHBoxLayout>
#include <QLabel>
#include <QGroupBox>
#include <QHeaderView>
#include <QPushButton>


#include "AddTargetIntDialog.h"

AddTargetIntDialog::AddTargetIntDialog(QWidget *parent) : QDialog(parent) {
  segmentsListText = new QLineEdit;
  numPointsSpin = new QSpinBox;
```

```
numPointsSpin->setEnabled(false);
numPointsSpin -> setMinimum(1);
numPointsSpin -> setMaximum(10000);
numPointsSpin -> setSingleStep(10);
numPointsSpin -> setValue(10);
isConvolutionCheck = new QCheckBox(tr("Include Gaussian Convolution"));
isConvolutionCheck->setChecked(false);
connect(isConvolutionCheck,SIGNAL(toggled(bool)),this,SLOT(convolutionCheckChanged(bool)));
sigmaText = new QLineEdit;
sigmaText->setEnabled(false);
isTargetIntegrationCheck = new QCheckBox(tr("Include Target Integration"));
isTargetIntegrationCheck->setChecked(false);
connect(isTargetIntegrationCheck,SIGNAL(toggled(bool)),this,SLOT(targetIntCheckChanged(bool)));
densityText = new QLineEdit;
densityText->setEnabled(false);
stoppingPowerEqText = new QLineEdit;
numParametersSpin = new QSpinBox;
numParametersSpin -> setMinimum(0);
numParametersSpin -> setMaximum(10);
numParametersSpin -> setSingleStep(1);
numParametersSpin -> setValue(0);
connect(numParametersSpin,SIGNAL(valueChanged(int)),this,SLOT(parameterSpinChanged(int)));
parametersTable = new QTableWidget(this);
parametersTable->setColumnCount(2);
parametersTable->setRowCount(0);
parametersTable->verticalHeader()->hide();
parametersTable->verticalHeader()->setHighlightSections(false);
parametersTable->horizontalHeader()->setHighlightSections(false);
parametersTable->horizontalHeader()->setSectionResizeMode(0,QHeaderView::Stretch);
parametersTable->horizontalHeader()->setSectionResizeMode(1,QHeaderView::Stretch);
parametersTable->setShowGrid(false);
connect(parametersTable,SIGNAL(cellChanged(int,int)),this,SLOT(parameterChanged(int,int)));
QStringList labelList;
labelList.append(QString(tr("Parameter")));
labelList.append(QString(tr("Value")));
parametersTable->setHorizontalHeaderLabels(labelList);
isQCoefficientCheck = new QCheckBox(tr("Include Attenuation Coefficients"));
isQCoefficientCheck->setChecked(false);
connect(isQCoefficientCheck,SIGNAL(toggled(bool)),this,SLOT(qCoefficientCheckChanged(bool)));
numQCoefficientSpin= new QSpinBox;
numQCoefficientSpin->setMinimum(0);
numQCoefficientSpin->setMaximum(6);
numQCoefficientSpin->setSingleStep(1);
numQCoefficientSpin->setValue(0);
numQCoefficientSpin->setEnabled(false);
connect(numQCoefficientSpin,SIGNAL(valueChanged(int)),this,SLOT(qCoefficientSpinChanged(int)));
qCoefficientTable = new QTableWidget(this);
qCoefficientTable->setColumnCount(2);
qCoefficientTable->setRowCount(0);
qCoefficientTable->verticalHeader()->hide();
qCoefficientTable->verticalHeader()->setHighlightSections(false);
qCoefficientTable->horizontalHeader()->setHighlightSections(false);
qCoefficientTable->horizontalHeader()->setSectionResizeMode(0,QHeaderView::Stretch);
qCoefficientTable->horizontalHeader()->setSectionResizeMode(1,QHeaderView::Stretch);
qCoefficientTable->setShowGrid(false);
connect(qCoefficientTable,SIGNAL(cellChanged(int,int)),this,SLOT(qCoefficientChanged(int,int)));
QStringList qlabelList;
qlabelList.append(QString(tr("Coefficent")));
qlabelList.append(QString(tr("Value")));
qCoefficientTable->setHorizontalHeaderLabels(qlabelList);


cancelButton = new QPushButton(tr("Cancel"));
okButton = new QPushButton(tr("Accept"));
okButton->setDefault(true);

QHBoxLayout *segListLayout = new QHBoxLayout;
segListLayout->addWidget(new QLabel(tr("Segments List:")));
segListLayout->addWidget(segmentsListText);

QHBoxLayout *numPointsLayout = new QHBoxLayout;
numPointsLayout->addWidget(new QLabel(tr("Number of Integration Points:")));
numPointsLayout->addWidget(numPointsSpin);

QHBoxLayout *topLayout = new QHBoxLayout;
topLayout->addLayout(segListLayout);
topLayout->addLayout(numPointsLayout);

QGridLayout *checkBoxLayout = new QGridLayout;
checkBoxLayout->addWidget(isConvolutionCheck,0,0);
checkBoxLayout->addWidget(new QLabel(tr("Sigma [MeV]:")),0,1,Qt::AlignRight);
checkBoxLayout->addWidget(sigmaText,0,2);

checkBoxLayout->addWidget(isTargetIntegrationCheck,1,0);
checkBoxLayout->addWidget(new QLabel(tr("Active Density [atoms/cm^2]:")),1,1,Qt::AlignRight);
checkBoxLayout->addWidget(densityText,1,2);

```

```
  stoppingPowerBox = new QGroupBox(tr("Effective Stopping Cross Section [MeV cm^2/atoms]"));
  QVBoxLayout *stoppingPowerLayout = new QVBoxLayout;
  QHBoxLayout *stoppingPowerTopLayout = new QHBoxLayout;
  QHBoxLayout *equationLayout = new QHBoxLayout;
  equationLayout->addWidget(new QLabel(tr("y=")));
  equationLayout->addWidget(stoppingPowerEqText);
  QHBoxLayout *numParamLayout = new QHBoxLayout;
  numParamLayout->addWidget(new QLabel(tr("Number of Parameters:")));
  numParamLayout->addWidget(numParametersSpin);
  stoppingPowerTopLayout->addLayout(equationLayout);
  stoppingPowerTopLayout->addLayout(numParamLayout);
  stoppingPowerLayout->addLayout(stoppingPowerTopLayout);
  stoppingPowerLayout->addWidget(parametersTable);
  stoppingPowerBox->setLayout(stoppingPowerLayout);
  stoppingPowerBox->hide();

  QGridLayout *qCoefficientCheckBoxLayout = new QGridLayout;
  qCoefficientCheckBoxLayout->addWidget(isQCoefficientCheck,0,0);
  qCoefficientCheckBoxLayout->addItem(new QSpacerItem(1,20),0,1);
  qCoefficientCheckBoxLayout->addWidget(new QLabel(tr("Number of Coefficients:")),0,2,Qt::AlignRight);
  qCoefficientCheckBoxLayout->addWidget(numQCoefficientSpin,0,3);
  qCoefficientCheckBoxLayout->setColumnStretch(1,1);

  qCoefficientBox = new QGroupBox(tr("Attenuation Coefficients"));
  QVBoxLayout *qCoefficientLayout = new QVBoxLayout;
  qCoefficientLayout->addWidget(qCoefficientTable);
  qCoefficientBox->setLayout(qCoefficientLayout);
  qCoefficientBox->hide();

  QHBoxLayout *buttonBox = new QHBoxLayout;
  buttonBox->addWidget(cancelButton);
  buttonBox->addWidget(okButton);

  QVBoxLayout *mainLayout = new QVBoxLayout;
  mainLayout->addLayout(topLayout);
  mainLayout->addLayout(checkBoxLayout);
  mainLayout->addWidget(stoppingPowerBox);
  mainLayout->addLayout(qCoefficientCheckBoxLayout);
  mainLayout->addWidget(qCoefficientBox);
  mainLayout->addLayout(buttonBox);

  setLayout(mainLayout);

  connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
  connect(cancelButton,SIGNAL(clicked()),this,SLOT(reject()));

  setWindowTitle(tr("Add an Experimental Effect Line"));
}

void AddTargetIntDialog::createParameterItem(int row, double value) {
  QTableWidgetItem *labelItem = new QTableWidgetItem(QString("a%1").arg(row));
  QTableWidgetItem *valueItem = new QTableWidgetItem(QString("%1").arg(value));
  labelItem->setTextAlignment(Qt::AlignCenter);
  labelItem->setFlags(Qt::ItemIsEditable);
  valueItem->setTextAlignment(Qt::AlignCenter);
  parametersTable->setItem(row,0,labelItem);
  parametersTable->setItem(row,1,valueItem);
  parametersTable->resizeRowsToContents();
}

void AddTargetIntDialog::createQCoefficientItem(int row, double value) {
  QTableWidgetItem *labelItem = new QTableWidgetItem(QString("q%1").arg(row));
  QTableWidgetItem *valueItem = new QTableWidgetItem(QString("%1").arg(value));
  labelItem->setTextAlignment(Qt::AlignCenter);
  labelItem->setFlags(Qt::ItemIsEditable);
  valueItem->setTextAlignment(Qt::AlignCenter);
  qCoefficientTable->setItem(row,0,labelItem);
  qCoefficientTable->setItem(row,1,valueItem);
  qCoefficientTable->resizeRowsToContents();
}

void AddTargetIntDialog::convolutionCheckChanged(bool checked) {
  if(checked) {
    sigmaText->setEnabled(true);
    numPointsSpin->setEnabled(true);
  } else {
    sigmaText->setEnabled(false);
    if(!isTargetIntegrationCheck->isChecked()) numPointsSpin->setEnabled(false);
  }
}

void AddTargetIntDialog::targetIntCheckChanged(bool checked) {
  if(checked) {
    stoppingPowerBox->show();
    densityText->setEnabled(true);
    numPointsSpin->setEnabled(true);
  } else {
```

```
    stoppingPowerBox->hide();
    densityText->setEnabled(false);
    if(!isConvolutionCheck->isChecked()) numPointsSpin->setEnabled(false);
    this->adjustSize();
  }
}

void AddTargetIntDialog::parameterSpinChanged(int newNumber) {
  parametersTable->clearContents();
  parametersTable->setRowCount(newNumber);
  for(int i=0;i<newNumber;i++) {
    if(i<tempParameters.size())
      createParameterItem(i,tempParameters.at(i));
    else
      createParameterItem(i);
  }
}

void AddTargetIntDialog::parameterChanged(int row, int column) {
  if(column==1) {
    while(row>=tempParameters.size()) {
      double tempDouble=0.0;
      tempParameters.append(tempDouble);
    }
    tempParameters[row]=parametersTable->item(row,column)->text().toDouble();
  }
}

void AddTargetIntDialog::qCoefficientCheckChanged(bool checked) {
  if(checked) {
    qCoefficientBox->show();
    numQCoefficientSpin->setEnabled(true);
  } else {
    qCoefficientBox->hide();
    numQCoefficientSpin->setEnabled(false);
    this->adjustSize();
  }
}

void AddTargetIntDialog::qCoefficientSpinChanged(int newNumber){
  qCoefficientTable->clearContents();
  qCoefficientTable->setRowCount(newNumber);
  for(int i=0;i<newNumber;i++) {
    if(i<tempQCoefficients.size())
      createQCoefficientItem(i,tempQCoefficients.at(i));
    else
      createQCoefficientItem(i);
  }
}

void AddTargetIntDialog::qCoefficientChanged(int row, int column) {
  if(column==1) {
    while(row>=tempQCoefficients.size()) {
      double tempDouble=0.0;
      tempQCoefficients.append(tempDouble);
    }
    tempQCoefficients[row]=qCoefficientTable->item(row,column)->text().toDouble();
  }
}
```

8.89 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZUREPlot.cpp File Reference

```
#include <QCheckBox>
#include <QDialog>
#include <QFileDialog>
#include <QListView>
#include <QPushButton>
#include <QMessageBox>
#include <QImageWriter>
#include <QtPrintSupport/QPrinter>
#include <QtPrintSupport/QPrintDialog>
#include "AZUREPlot.h"
#include "PlotTab.h"
#include "qwt_plot_curve.h"
```

```
#include "qwt_plot_intervalcurve.h"
#include "qwt_interval_symbol.h"
#include "qwt_scale_engine.h"
#include "qwt_plot_panner.h"
#include "qwt_plot_renderer.h"
#include <iostream>
```

8.90 AZUREPlot.cpp

Go to the documentation of this file.

```
#include <QCheckBox>
#include <QDialog>
#include <QFileDialog>
#include <QCheckBox>
#include <QListView>
#include <QPushButton>
#include <QMessageBox>
#include <QCheckBox>
#include <QImageWriter>
#include <QtPrintSupport/QPrinter>
#include <QtPrintSupport/QPrintDialog>

#include "AZUREPlot.h"
#include "PlotTab.h"
#include "qwt_plot_curve.h"
#include "qwt_plot_intervalcurve.h"
#include "qwt_interval_symbol.h"
#include "qwt_scale_engine.h"
#include "qwt_plot_panner.h"
#include "qwt_plot_renderer.h"
#include <iostream>

QwtText AZUREZoomer::trackerTextF( const QPointF &pos ) const
{
    QString text;
    switch (rubberBand()) {
        case HLineRubberBand:
            text.sprintf( "%.4g", pos.y() );
            break;
        case VLineRubberBand:
            text.sprintf( "%.4g", pos.x() );
            break;
        default:
            text.sprintf( "%.4g, %.4g", pos.x(), pos.y() );
    }
    return QwtText( text );
}


PlotEntry::PlotEntry(int type, int entranceKey, int exitKey, int index, QString filename) :
  type_(type), entranceKey_(entranceKey), exitKey_(exitKey), index_(index), filename_(filename),
  dataCurve_(NULL), dataErrorCurve_(NULL), fitCurve_(NULL) {
}

PlotEntry::~PlotEntry() {
    if(dataCurve_) delete dataCurve_;
    if(dataErrorCurve_) delete dataErrorCurve_;
    if(fitCurve_) delete fitCurve_;
}

bool PlotEntry::readData() {
  QFile file(filename_);
  if(!file.open(QIODevice::ReadOnly)) {
    return false;
  }
  QTextStream inStream(&file);
  QString line("");
  bool previousLineBreak=false;
  bool foundBlock=false;
  int blockNumber=0;
  while(!inStream.atEnd()&&!foundBlock) {
    line=inStream.readLine();
    if(line.trimmed().isEmpty()) {
      if(!previousLineBreak) {
    previousLineBreak=true;
    continue;
      }
```

```
      if(blockNumber==index_) {
    foundBlock=true;
    break;
      } else {
    blockNumber++;
    previousLineBreak=false;
    points_.clear();
    continue;
      }
    }
    if(!inStream.atEnd()&&!foundBlock) {
      QTextStream in(&line);
      PlotPoint newPoint = {0.,0.,0.,0.,0.,0.,0.,0.,0.};
      if(type_==0) {
    in » newPoint.energy » newPoint.excitationEnergy » newPoint.angle » newPoint.fitCrossSection » newPoint.fitSFactor
       » newPoint.dataCrossSection » newPoint.dataErrorCrossSection » newPoint.dataSFactor
       » newPoint.dataErrorSFactor;
      } else {
    in » newPoint.energy » newPoint.excitationEnergy » newPoint.angle » newPoint.fitCrossSection » newPoint.fitSFactor;
      }
      points_.push_back(newPoint);
    }
  }
  inStream.flush();
  file.close();
  if(!foundBlock) {
    return false;
  }
  hasNegative_=false;
  for(QVector<PlotPoint>::const_iterator it=points_.begin();
      it<points_.end();it++) {
    if(it->fitCrossSection<=0.||
       (type_==0&&
    (it->dataCrossSection<=0.||
     (fabs(it->dataCrossSection)-
      fabs(it->dataErrorCrossSection))<=0.))) {
      hasNegative_=true;
      break;
    }
  }
  return true;
}

void PlotEntry::attach(QwtPlot* plot, int xAxisType, int yAxisType, QwtSymbol::Style style) {
  QVector<QPointF> fit(points_.size());
  if(type_==0) {
    QVector<QPointF> data(points_.size());
    QVector<QwtIntervalSample> error(points_.size());

    if(xAxisType==0&&yAxisType==0) {
      for(int i=0;i<points_.size();i++) {
    data[i]=QPointF(points_[i].energy,points_[i].dataCrossSection);
    fit[i]=QPointF(points_[i].energy,points_[i].fitCrossSection);
    error[i]=QwtIntervalSample(points_[i].energy,
                   QwtInterval(points_[i].dataCrossSection-points_[i].dataErrorCrossSection,
                           points_[i].dataCrossSection+points_[i].dataErrorCrossSection));
      }
    } else if(xAxisType==0&&yAxisType==1) {
      for(int i=0;i<points_.size();i++) {
    data[i]=QPointF(points_[i].energy,points_[i].dataSFactor);
    fit[i]=QPointF(points_[i].energy,points_[i].fitSFactor);
    error[i]=QwtIntervalSample(points_[i].energy,
                   QwtInterval(points_[i].dataSFactor-points_[i].dataErrorSFactor,
                           points_[i].dataSFactor+points_[i].dataErrorSFactor));
      }
    } else if(xAxisType==1&&yAxisType==0) {
      for(int i=0;i<points_.size();i++) {
    data[i]=QPointF(points_[i].excitationEnergy,points_[i].dataCrossSection);
    fit[i]=QPointF(points_[i].excitationEnergy,points_[i].fitCrossSection);
    error[i]=QwtIntervalSample(points_[i].excitationEnergy,
                   QwtInterval(points_[i].dataCrossSection-points_[i].dataErrorCrossSection,
                           points_[i].dataCrossSection+points_[i].dataErrorCrossSection));
      }
    } else if(xAxisType==1&&yAxisType==1) {
      for(int i=0;i<points_.size();i++) {
    data[i]=QPointF(points_[i].excitationEnergy,points_[i].dataSFactor);
    fit[i]=QPointF(points_[i].excitationEnergy,points_[i].fitSFactor);
    error[i]=QwtIntervalSample(points_[i].excitationEnergy,
                   QwtInterval(points_[i].dataSFactor-points_[i].dataErrorSFactor,
                           points_[i].dataSFactor+points_[i].dataErrorSFactor));
      }
    } else if(xAxisType==2&&yAxisType==0) {
      for(int i=0;i<points_.size();i++) {
    data[i]=QPointF(points_[i].angle,points_[i].dataCrossSection);
    fit[i]=QPointF(points_[i].angle,points_[i].fitCrossSection);
```

```
    error[i]=QwtIntervalSample(points_[i].angle,
        QwtInterval(points_[i].dataCrossSection-points_[i].dataErrorCrossSection,
        points_[i].dataCrossSection+points_[i].dataErrorCrossSection));
      }
    } else if(xAxisType==2&&yAxisType==1) {
      for(int i=0;i<points_.size();i++) {
    data[i]=QPointF(points_[i].angle,points_[i].dataSFactor);
    fit[i]=QPointF(points_[i].angle,points_[i].fitSFactor);
    error[i]=QwtIntervalSample(points_[i].angle,
        QwtInterval(points_[i].dataSFactor-points_[i].dataErrorSFactor,
        points_[i].dataSFactor+points_[i].dataErrorSFactor));
      }
    }

    dataCurve_ = new QwtPlotCurve;
    dataCurve_->setRenderHint( QwtPlotItem::RenderAntialiased );
    dataCurve_->setStyle( QwtPlotCurve::NoCurve );
    QwtSymbol *symbol = new QwtSymbol(style);
    symbol->setSize( 6 );
    symbol->setPen( QPen( Qt::black ) );
    symbol->setColor( QColor( Qt::black ) );

    dataErrorCurve_ = new QwtPlotIntervalCurve;
    dataErrorCurve_->setRenderHint( QwtPlotItem::RenderAntialiased );
    dataErrorCurve_->setStyle( QwtPlotIntervalCurve::NoCurve );
    QwtIntervalSymbol *errorBar =
      new QwtIntervalSymbol( QwtIntervalSymbol::Bar );
    errorBar->setWidth( 8 );
    errorBar->setPen( QPen( Qt::black ) );

    dataCurve_->setSymbol( symbol );
    dataCurve_->setSamples(data);

    dataErrorCurve_->setSymbol(errorBar);
    dataErrorCurve_->setSamples(error);
  } else {
    if(xAxisType==0&&yAxisType==0) {
      for(int i=0;i<points_.size();i++)
    fit[i]=QPointF(points_[i].energy,points_[i].fitCrossSection);
    } else if(xAxisType==0&&yAxisType==1) {
      for(int i=0;i<points_.size();i++)
    fit[i]=QPointF(points_[i].energy,points_[i].fitSFactor);
    } else if(xAxisType==1&&yAxisType==0) {
      for(int i=0;i<points_.size();i++)
    fit[i]=QPointF(points_[i].excitationEnergy,points_[i].fitCrossSection);
    } else if(xAxisType==1&&yAxisType==1) {
      for(int i=0;i<points_.size();i++)
    fit[i]=QPointF(points_[i].excitationEnergy,points_[i].fitSFactor);
    } else if(xAxisType==2&&yAxisType==0) {
      for(int i=0;i<points_.size();i++)
    fit[i]=QPointF(points_[i].angle,points_[i].fitCrossSection);
    } else if(xAxisType==2&&yAxisType==1) {
      for(int i=0;i<points_.size();i++)
    fit[i]=QPointF(points_[i].angle,points_[i].fitSFactor);
    }
  }

  fitCurve_ = new QwtPlotCurve;
  fitCurve_->setRenderHint( QwtPlotItem::RenderAntialiased );
  fitCurve_->setStyle( QwtPlotCurve::Lines );
  fitCurve_->setPen( QPen( Qt::red , 2 ) );

  fitCurve_->setSymbol(new QwtSymbol(QwtSymbol::NoSymbol));
  fitCurve_->setSamples(fit);

  if(dataCurve_) dataCurve_->attach(plot);
  if(dataErrorCurve_) dataErrorCurve_->attach(plot);
  if(fitCurve_) fitCurve_->attach(plot);
}

void PlotEntry::detach() {
  if(dataCurve_) dataCurve_->detach();
  if(dataErrorCurve_) dataErrorCurve_->detach();
  if(fitCurve_) fitCurve_->detach();
}

AZUREPlot::AZUREPlot(PlotTab* plotTab,QWidget* parent) :
  containingTab(plotTab), QwtPlot(parent) {
  setCanvasBackground(QColor(Qt::white));
  setAutoReplot(true);

  zoomer = new AZUREZoomer( canvas() );
  zoomer->setRubberBandPen( QColor( Qt::black ) );
  zoomer->setTrackerPen( QColor( Qt::black ) );
  zoomer->setMousePattern( QwtEventPattern::MouseSelect2,
               Qt::RightButton, Qt::ControlModifier );
  zoomer->setMousePattern( QwtEventPattern::MouseSelect3,
```

```
                Qt::RightButton );

  QwtPlotPanner *panner = new QwtPlotPanner( canvas() );
  panner->setMouseButton( Qt::MidButton );

}


void AZUREPlot::setXAxisLog(bool set) {
  setAxisAutoScale(QwtPlot::xBottom,true);
  setAxisAutoScale(QwtPlot::yLeft,true);
  if(set) setAxisScaleEngine(QwtPlot::xBottom,new QwtLogScaleEngine);
  else setAxisScaleEngine(QwtPlot::xBottom,new QwtLinearScaleEngine);
  zoomer->setZoomBase(false);
};

void AZUREPlot::setYAxisLog(bool set) {
  setAxisAutoScale(QwtPlot::xBottom,true);
  setAxisAutoScale(QwtPlot::yLeft,true);
  if(set) {
    QwtLogScaleEngine* scaleEngine = new QwtLogScaleEngine;
    scaleEngine->setMargins(0.5,0.5);
    setAxisScaleEngine(QwtPlot::yLeft,scaleEngine);
  } else setAxisScaleEngine(QwtPlot::yLeft,new QwtLinearScaleEngine);
  zoomer->setZoomBase(false);
};

void AZUREPlot::setXAxisType(unsigned int type) {
  QwtText text;
  if(type==0) text=QwtText(QString("Center of Mass Energy [MeV]"));
  else if(type==1) text=QwtText(QString("Excitation Energy [MeV]"));
  else text=QwtText(QString("Center of Mass Angle [degrees]"));
  setAxisTitle(QwtPlot::xBottom,text);

  xAxisType=type;
  update();
}

void AZUREPlot::setYAxisType(unsigned int type) {
  QwtText text = (type==0) ? QwtText(QString("Cross Section [b]")) : QwtText(QString("S-Factor [MeV
b]"));
  setAxisTitle(QwtPlot::yLeft,text);
  yAxisType=type;
  update();
}

void AZUREPlot::draw(QList<PlotEntry*> newEntries) {
  clearEntries();

  int numDataEntries=0;
  bool hasNegative=false;
  for(int i = 0; i<newEntries.size(); i++) {
    if(newEntries[i]->readData()) {
      QwtSymbol::Style style = (newEntries[i]->type()==0) ? (QwtSymbol::Style)
    numDataEntries++ : QwtSymbol::NoSymbol;
      newEntries[i]->attach(this,xAxisType,yAxisType,style);
      entries.push_back(newEntries[i]);
      if(newEntries[i]->hasNegative_) hasNegative=true;
    } else delete newEntries[i];
  }
  setAxisAutoScale(QwtPlot::xBottom,true);
  setAxisAutoScale(QwtPlot::yLeft,true);
  replot();
  zoomer->setZoomBase(false);
  if(hasNegative) {
    containingTab->yAxisIsLogCheck->setChecked(false);
    containingTab->yAxisIsLogCheck->setEnabled(false);
    QMessageBox::information(this,
                 tr("Negative or Zero Values"),
                 tr("Negative or zero values were detected in a dataset. "
                "Log plotting is not available."));
  }
}

void AZUREPlot::update() {
  setAxisAutoScale(QwtPlot::xBottom,true);
  setAxisAutoScale(QwtPlot::yLeft,true);
  int numDataEntries=0;
  for(int i = 0; i<entries.size(); i++) {
    entries[i]->detach();
    QwtSymbol::Style style = (entries[i]->type()==0) ? (QwtSymbol::Style) numDataEntries++ :
QwtSymbol::NoSymbol;
    entries[i]->attach(this,xAxisType,yAxisType,style);
  }
  replot();
  zoomer->setZoomBase(false);
}
```

```

void AZUREPlot::exportPlot()
{
#ifndef QT_NO_PRINTER
  QString fileName = "AZUREPlot.pdf";
#else
  QString fileName = "AZUREPlot.png";
#endif

#ifndef QT_NO_FILEDIALOG
    const QList<QByteArray> imageFormats =
      QImageWriter::supportedImageFormats();

    QStringList filter;
    filter += "PDF Documents (*.pdf)";
#ifndef QWT_NO_SVG
    filter += "SVG Documents (*.svg)";
#endif
    filter += "Postscript Documents (*.ps)";

    if (imageFormats.size()>0) {
      QString imageFilter("Images (");
      for (int i=0;i<imageFormats.size();i++) {
    if (i>0) imageFilter += " ";
      imageFilter += "*.";
      imageFilter += imageFormats[i];
        }
      imageFilter += ")";

      filter += imageFilter;
    }
    fileName = QFileDialog::getSaveFileName(
                          this, "Export File Name", fileName,
                          filter.join(";;"), NULL, QFileDialog::DontConfirmOverwrite);
#endif
    if(!fileName.isEmpty()) {
      QwtPlotRenderer renderer;
      renderer.setDiscardFlag(QwtPlotRenderer::DiscardBackground, true);
      renderer.renderDocument(this, fileName, QSizeF(300, 200), 85);
    }
}

void AZUREPlot::print()
{
    QPrinter printer(QPrinter::HighResolution);
    QString docName("AZUREPlot");
    printer.setDocName (docName);

    printer.setCreator("AZURE2");
    printer.setOrientation(QPrinter::Landscape);

    QPrintDialog dialog(&printer);
    if (dialog.exec()) {
        QwtPlotRenderer renderer;
        renderer.renderTo(this, printer);
    }
}

void AZUREPlot::clearEntries() {
  for(int i = 0; i<entries.size(); i++) {
    entries[i]->detach();
    delete entries[i];
  }
  if(entries.size()>0) entries.clear();
  containingTab->yAxisIsLogCheck->setEnabled(true);
  update();
}
```

8.91 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZURESetup.cpp File Reference

#include <QTextEdit>
#include <QMessageBox>
#include <QScrollBar>
#include <QFileDialog>
#include <QMenu>
#include <QMenuBar>

```
#include <QRadioButton>
#include <QAction>
#include <QActionGroup>
#include <QSettings>
#include <QTextStream>
#include <QDesktopServices>
#include "AZURESetup.h"
#include "EditChecksDialog.h"
#include "EditDirsDialog.h"
#include "RunTab.h"
#include "Config.h"
#include "EditOptionsDialog.h"
#include "AZUREMainThread.h"
#include "AboutAZURE2Dialog.h"
#include <iostream>
```

Classes

- struct SegPairs

Functions

- bool readSegmentFile (const Config &configure, std::vector< SegPairs > &segPairs)
- bool checkExternalCapture (Config &configure, const std::vector< SegPairs > &segPairs)
- void startMessage (const Config &configure)
- void exitMessage (const Config &configure)

8.91.1 Function Documentation

8.91.1.1 checkExternalCapture()

```
bool checkExternalCapture (
            Config & configure,
            const std::vector< SegPairs > & segPairs )
```

This function checks the external capture file against a vector of segment key pairs. Only if the calculation includes external capture segments is the user prompted for an integrals file. The appropriate configure flag is set here.

Definition at line 438 of file AZURE2.cpp.

8.91.1.2 exitMessage()

```
void exitMessage (
            const Config & configure )
```

This function prints a message upon successful termination of the program.

Definition at line 56 of file AZURE2.cpp.

8.91.1.3 readSegmentFile()

```
bool readSegmentFile (
            const Config & configure,
            std::vector< SegPairs > & segPairs )
```

This function reads the segment file, and stores the active entrance and exit pair keys for cross reference with the External capture file. Only if an active external capture segment is required is the user prompted for an external integrals file.

Definition at line 247 of file AZURE2.cpp.

8.91.1.4 startMessage()

```
void startMessage (
            const Config & configure )
```

This function prints a brief start message describing the type of calculation that will be performed.

Definition at line 539 of file AZURE2.cpp.

8.92 AZURESetup.cpp

Go to the documentation of this file.

```
#include <QTextEdit>
#include <QMessageBox>
#include <QScrollBar>
#include <QFileDialog>
#include <QMenu>
#include <QMenuBar>
#include <QRadioButton>
#include <QAction>
#include <QActionGroup>
#include <QSettings>
#include <QTextStream>
#include <QDesktopServices>

#include "AZURESetup.h"
#include "EditChecksDialog.h"
#include "EditDirsDialog.h"
#include "RunTab.h"
#include "Config.h"
#include "EditOptionsDialog.h"
#include "AZUREMainThread.h"
#include "AboutAZURE2Dialog.h"

#ifdef USE_QWT
#include "PlotTab.h"
#endif
#include <iostream>

struct SegPairs {int firstPair; int secondPair;};
extern bool readSegmentFile(const Config& configure,std::vector<SegPairs>& segPairs);
extern bool checkExternalCapture(Config& configure, const std::vector<SegPairs>& segPairs);
extern void startMessage(const Config& configure);
extern void exitMessage(const Config& configure);


AZURESetup::AZURESetup() : config(std::cout) {
  setMinimumSize(1000,640);

  tabWidget=new QTabWidget();

  pairsTab=new PairsTab;

  levelsTab = new LevelsTab;
  levelsTab->setPairsModel(pairsTab->getPairsModel());
  connect(pairsTab,SIGNAL(pairAdded(int)),levelsTab,SLOT(updateChannelsPairAddedEdited()));
```

```
  connect(pairsTab,SIGNAL(pairEdited(int)),levelsTab,SLOT(updateChannelsPairAddedEdited()));
  connect(pairsTab,SIGNAL(pairRemoved(int)),levelsTab,SLOT(updateChannelsPairRemoved(int)));
connect(levelsTab,SIGNAL(readNewPair(PairsData,int,bool)),pairsTab,SLOT(addPair(PairsData,int,bool)));
connect(levelsTab,SIGNAL(readExistingPair(PairsData,int,bool)),pairsTab,SLOT(editPair(PairsData,int,bool)));

  segmentsTab = new SegmentsTab;
  segmentsTab->setPairsModel(pairsTab->getPairsModel());

  targetIntTab=new TargetIntTab;

  runTab = new RunTab();
  connect(runTab->calcButton,SIGNAL(clicked()),this,SLOT(SaveAndRun()));

#ifdef USE_QWT
  plotTab = new
PlotTab(config,segmentsTab->getSegmentsDataModel(),segmentsTab->getSegmentsTestModel());
#endif

  tabWidget->addTab(pairsTab,tr("&Particle Pairs"));
  tabWidget->addTab(levelsTab,tr("&Levels and Channels"));
  tabWidget->addTab(segmentsTab,tr("&Segments"));
  tabWidget->addTab(targetIntTab,tr("&Experimental Effects"));
  tabWidget->addTab(runTab,tr("&Calculate"));
#ifdef USE_QWT
  tabWidget->addTab(plotTab,tr("Pl&ot"));
#endif

  setCentralWidget(tabWidget);

  createActions();
  createMenus();


  setWindowTitle(tr("AZURE2 -- untitled"));
}

Config& AZURESetup::GetConfig() {
  return config;
}

void AZURESetup::createActions() {
  aboutAction = new QAction(tr("&About AZURE2..."),this);
  connect(aboutAction,SIGNAL(triggered()),this,SLOT(showAbout()));

  resetAction = new QAction(tr("&New Project"),this);
  resetAction->setShortcuts(QKeySequence::New);
  connect(resetAction,SIGNAL(triggered()),this,SLOT(reset()));

  quitAction = new QAction(tr("&Quit"),this);
  quitAction->setShortcuts(QKeySequence::Quit);
  connect(quitAction,SIGNAL(triggered()),this,SLOT(close()));

  openAction = new QAction(tr("&Open..."),this);
  openAction->setShortcuts(QKeySequence::Open);
  connect(openAction,SIGNAL(triggered()),this,SLOT(open()));

  saveAction = new QAction(tr("&Save"),this);
  saveAction->setShortcuts(QKeySequence::Save);
  connect(saveAction,SIGNAL(triggered()),this,SLOT(save()));

  saveAsAction = new QAction(tr("Save &As..."),this);
  saveAsAction->setShortcuts(QKeySequence::SaveAs);
  connect(saveAsAction,SIGNAL(triggered()),this,SLOT(saveAs()));

  for(int i=0;i<numRecent;i++) {
    recentFileActions[i] = new QAction(this);
    recentFileActions[i] -> setVisible(false);
    connect(recentFileActions[i],SIGNAL(triggered()),this,SLOT(openRecent()));
  }
  clearRecentAction = new QAction(tr("&Clear"),this);
  clearRecentAction->setVisible(false);
  connect(clearRecentAction,SIGNAL(triggered()),this,SLOT(clearRecent()));

  copyAction = new QAction(tr("&Copy"),this);
  copyAction->setShortcuts(QKeySequence::Copy);

  matrixActionGroup = new QActionGroup(this);
  aMatrixAction = new QAction(tr("&A-Matrix"),this);
  aMatrixAction->setCheckable(true);
  matrixActionGroup->addAction(aMatrixAction);
  rMatrixAction = new QAction(tr("&R-Matrix"),this);
  rMatrixAction->setCheckable(true);
  matrixActionGroup->addAction(rMatrixAction);
  aMatrixAction->setChecked(true);
  connect(matrixActionGroup,SIGNAL(triggered(QAction*)),this,SLOT(matrixChanged(QAction*)));
```

```

  editChecksAction = new QAction(tr("&Checks..."),this);
  connect(editChecksAction,SIGNAL(triggered()),this,SLOT(editChecks()));

  editDirsAction = new QAction(tr("&Directories..."),this);
  connect(editDirsAction,SIGNAL(triggered()),this,SLOT(editDirs()));

  editOptionsAction = new QAction(tr("&Runtime Options..."),this);
  connect(editOptionsAction,SIGNAL(triggered()),this,SLOT(editOptions()));

  showTabInfoAction = new QAction(tr("Show Documentation For Current Tab"),this);
  showTabInfoAction->setShortcut(QKeySequence(Qt::CTRL + Qt::Key_D));
  connect(showTabInfoAction,SIGNAL(triggered()),this,SLOT(showTabInfo()));

  openAZURESiteAction = new QAction(tr("Open AZURE Website..."),this);
  connect(openAZURESiteAction,SIGNAL(triggered()),this,SLOT(openWebsite()));
}

void AZURESetup::createMenus() {
  fileMenu = menuBar()->addMenu(tr("&File"));
  fileMenu->addAction(aboutAction);
  fileMenu->addSeparator();
  fileMenu->addAction(resetAction);
  fileMenu->addAction(openAction);
  recentFileMenu = fileMenu->addMenu(tr("Open &Recent..."));
  for(int i = 0; i < numRecent; i++ ) recentFileMenu->addAction(recentFileActions[i]);
  recentSeparator = recentFileMenu->addSeparator();
  recentFileMenu->addAction(clearRecentAction);
  updateRecent();
  fileMenu->addAction(saveAction);
  fileMenu->addAction(saveAsAction);
  fileMenu->addSeparator();
  fileMenu->addAction(quitAction);

  configMenu = menuBar()->addMenu(tr("Co&nfigure"));
  formalismMenu = configMenu->addMenu(tr("&Formalism"));
  formalismMenu->addAction(aMatrixAction);
  formalismMenu->addAction(rMatrixAction);
  configMenu->addAction(editChecksAction);
  configMenu->addAction(editDirsAction);
  configMenu->addAction(editOptionsAction);

  helpMenu = menuBar()->addMenu(tr("&Documentation"));
  helpMenu->addAction(showTabInfoAction);
  helpMenu->addAction(openAZURESiteAction);
}

void AZURESetup::updateRecent() {
  QSettings settings;
  QStringList files = settings.value("recentFileList").toStringList();

  int numFiles = qMin(files.size(),(int)numRecent);

  for(int i = 0; i<numFiles; i++) {
    recentFileActions[i]->setText(tr("&%1 %2").arg(i+1).arg(QFileInfo(files[i]).fileName()));
    recentFileActions[i]->setData(files[i]);
    recentFileActions[i]->setVisible(true);
  }

  for(int i = numFiles; i<numRecent; i++) recentFileActions[i]->setVisible(false);
  recentSeparator->setVisible(numFiles>0);
  clearRecentAction->setVisible(numFiles>0);
}

void AZURESetup::open() {
  QString filename = QFileDialog::getOpenFileName(this);
  if(!filename.isEmpty()) {
    if(!this->readFile(filename)) {
      reset();
      QMessageBox::information(this,
                      tr("Can't Access File"),
                      tr("An error was encountered while reading the file."));
    }
  }
}

void AZURESetup::open(QString filename) {
  if(!filename.isEmpty()) {
    if(!this->readFile(filename)) {
      reset();
      QMessageBox::information(this,
                      tr("Can't Access File"),
                      tr("An error was encountered while reading the file."));
    }
  }
}

```

```
void AZURESetup::openRecent() {
  QString filename = qobject_cast<QAction*>(sender())->data().toString();
  open(filename);
}

void AZURESetup::clearRecent() {
  QSettings settings;
  QStringList files = settings.value("recentFileList").toStringList();
  files.clear();
  settings.setValue("recentFileList",files);
  updateRecent();
}

bool AZURESetup::readFile(QString filename) {
  QFile file(filename);
  if(!file.open(QIODevice::ReadOnly)) return false;
  QFileInfo info(file);
  QString directory=info.absolutePath();

  reset();

  QTextStream in(&file);
  QString line("");

  while(line.trimmed()!=QString("<config>")&&!in.atEnd()) line = in.readLine();
  if(in.atEnd()) return false;
  if(!this->readConfig(in)) return false;

  line=QString("");
  while(line.trimmed()!=QString("<levels>")&&!in.atEnd()) line = in.readLine();
  if(in.atEnd()) return false;
  if(!levelsTab->readNuclearFile(in)) return false;

  line=QString("");
  while(line.trimmed()!=QString("<segmentsData>")&&!in.atEnd()) line = in.readLine();
  if(in.atEnd()) return false;
  if(!segmentsTab->readSegDataFile(in)) return false;

  line=QString("");
  while(line.trimmed()!=QString("<segmentsTest>")&&!in.atEnd()) line = in.readLine();
  if(in.atEnd()) return false;
  if(!segmentsTab->readSegTestFile(in)) return false;

  line=QString("");
  while(line.trimmed()!=QString("<targetInt>")&&!in.atEnd()) line = in.readLine();
  if(in.atEnd()) return false;
  if(!targetIntTab->readFile(in)) return false;

  line=QString("");
  while(line.trimmed()!=QString("<lastRun>")&&!in.atEnd()) line = in.readLine();
  if(!in.atEnd())
    if(!this->readLastRun(in)) return false;

  file.close();

  QFile file2(filename);
  if(!file2.open(QIODevice::ReadOnly)) return false;
  QTextStream in2(&file2);
  line=QString("");
  while(line.trimmed()!=QString("<externalCapture>")&&!in2.atEnd()) line = in2.readLine();
  if(!in2.atEnd()) {
    if(!pairsTab->parseOldECSection(in2)) return false;
  }
  file2.close();

  GetConfig().configfile=QDir::fromNativeSeparators(info.absoluteFilePath()).toStdString();
  setWindowTitle(QString("AZURE2 -- %1").arg(QString::fromStdString(GetConfig().configfile)));
  QDir::setCurrent(directory);

  QSettings settings;
  QStringList files = settings.value("recentFileList").toStringList();
  QString fullFileName = QDir::fromNativeSeparators(info.absoluteFilePath());
  files.removeAll(fullFileName);
  files.prepend(fullFileName);
  while(files.size()>numRecent) files.removeLast();

  settings.setValue("recentFileList",files);
  updateRecent();

  return true;
}

bool AZURESetup::readLastRun(QTextStream& inStream) {
  unsigned int paramMask;
  unsigned int useTempFile;
  unsigned int rateEntrancePair;
  unsigned int rateExitPair;
```

```
  QString paramFile;
  QString integralsFile;
  QString temperatureFile;
  QString dummyString;
  double minTemp;
  double maxTemp;
  double tempStep;

  inStream » paramMask;dummyString=inStream.readLine();
  dummyString=inStream.readLine();paramFile=dummyString.trimmed();
  dummyString=inStream.readLine();integralsFile=dummyString.trimmed();
  inStream » rateEntrancePair » rateExitPair;dummyString=inStream.readLine();
  inStream » useTempFile;dummyString=inStream.readLine();temperatureFile=dummyString.trimmed();
  inStream » minTemp » maxTemp » tempStep;

  QString line("");
  while(line.trimmed()!=QString("</lastRun>")&&!inStream.atEnd())
    line=inStream.readLine();
  if(line.trimmed()!=QString("</lastRun>")) return false;

  if(paramMask &  Config::CALCULATE_WITH_DATA) {
    if(paramMask & Config::PERFORM_FIT) runTab->calcType->setCurrentIndex(1);
    else if(paramMask & Config::PERFORM_ERROR_ANALYSIS) runTab->calcType->setCurrentIndex(3);
    else runTab->calcType->setCurrentIndex(0);
  } else {
    if(paramMask &  Config::CALCULATE_REACTION_RATE) runTab->calcType->setCurrentIndex(4);
    else  runTab->calcType->setCurrentIndex(2);
  }

  if(paramMask & Config::USE_GSL_COULOMB_FUNC) GetConfig().paramMask |= Config::USE_GSL_COULOMB_FUNC;
  else GetConfig().paramMask &= ~Config::USE_GSL_COULOMB_FUNC;

  if(paramMask & Config::USE_BRUNE_FORMALISM) GetConfig().paramMask |= Config::USE_BRUNE_FORMALISM;
  else GetConfig().paramMask &= ~Config::USE_BRUNE_FORMALISM;

  if(paramMask & Config::IGNORE_ZERO_WIDTHS) GetConfig().paramMask |= Config::IGNORE_ZERO_WIDTHS;
  else GetConfig().paramMask &= ~Config::IGNORE_ZERO_WIDTHS;

  if(paramMask & Config::USE_RMC_FORMALISM) GetConfig().paramMask |= Config::USE_RMC_FORMALISM;
  else GetConfig().paramMask &= ~Config::USE_RMC_FORMALISM;

  if(paramMask & Config::TRANSFORM_PARAMETERS) GetConfig().paramMask |= Config::TRANSFORM_PARAMETERS;
  else GetConfig().paramMask &= ~Config::TRANSFORM_PARAMETERS;

  if(paramMask & Config::USE_LONGWAVELENGTH_APPROX) GetConfig().paramMask |= Config::USE_LONGWAVELENGTH_APPROX;
  else GetConfig().paramMask &= ~Config::USE_LONGWAVELENGTH_APPROX;

  if(rateEntrancePair!=0) runTab->rateEntranceKey->setText(QString("%1").arg(rateEntrancePair));
  if(rateExitPair!=0) runTab->rateExitKey->setText(QString("%1").arg(rateExitPair));

  if(minTemp!=-1.) runTab->minTempText->setText(QString("%1").arg(minTemp));
  if(maxTemp!=-1.) runTab->maxTempText->setText(QString("%1").arg(maxTemp));
  if(tempStep!=-1.) runTab->tempStepText->setText(QString("%1").arg(tempStep));

  if(useTempFile==1) runTab->fileTempButton->setChecked(true);
  else runTab->gridTempButton->setChecked(true);
  if(temperatureFile[0]==QChar('"')) temperatureFile.remove(0,1);
  if(temperatureFile[temperatureFile.length()-1]==QChar('"')) temperatureFile.remove(temperatureFile.length()-1,1);
  if(!temperatureFile.trimmed().isEmpty()) runTab->fileTempText->setText(temperatureFile.trimmed());

  if(paramMask & Config::USE_PREVIOUS_PARAMETERS) runTab->oldParamFileButton->setChecked(true);
  else runTab->newParamFileButton->setChecked(true);
  if(paramFile[0]==QChar('"')) paramFile.remove(0,1);
  if(paramFile[paramFile.length()-1]==QChar('"')) paramFile.remove(paramFile.length()-1,1);
  if(!paramFile.trimmed().isEmpty()) runTab->paramFileText->setText(paramFile.trimmed());

  if(paramMask & Config::USE_PREVIOUS_INTEGRALS) runTab->oldIntegralsFileButton->setChecked(true);
  else runTab->newIntegralsFileButton->setChecked(true);
  if(integralsFile[0]==QChar('"')) integralsFile.remove(0,1);
  if(integralsFile[integralsFile.length()-1]==QChar('"')) integralsFile.remove(integralsFile.length()-1,1);
  if(!integralsFile.trimmed().isEmpty()) runTab->integralsFileText->setText(integralsFile.trimmed());

  return true;
}

bool AZURESetup::readConfig(QTextStream& inStream) {

  QString isAMatrix;
  QString outputDirectory;
  QString checksDirectory;
  QString compoundCheck;
  QString boundaryCheck;
  QString dataCheck;
  QString lMatrixCheck;
```

```
  QString legendreCheck;
  QString coulAmpCheck;
  QString pathwaysCheck;
  QString angDistsCheck;
  QString dummyString;

  inStream » isAMatrix;dummyString=inStream.readLine();
  dummyString=inStream.readLine();
  int poundSignPos = dummyString.lastIndexOf('#');
  if(poundSignPos==-1) outputDirectory=dummyString.trimmed();
  else outputDirectory=dummyString.left(poundSignPos).trimmed();
  dummyString=inStream.readLine();
  poundSignPos = dummyString.lastIndexOf('#');
  if(poundSignPos==-1) checksDirectory=dummyString.trimmed();
  else checksDirectory=dummyString.left(poundSignPos).trimmed();
  inStream » compoundCheck;dummyString=inStream.readLine();
  inStream » boundaryCheck;dummyString=inStream.readLine();
  inStream » dataCheck;dummyString=inStream.readLine();
  inStream » lMatrixCheck;dummyString=inStream.readLine();
  inStream » legendreCheck;dummyString=inStream.readLine();
  inStream » coulAmpCheck;dummyString=inStream.readLine();
  inStream » pathwaysCheck;dummyString=inStream.readLine();
  inStream » angDistsCheck;dummyString=inStream.readLine();

  QString line("");
  while(line.trimmed()!=QString("</config>")&&!inStream.atEnd())
    line=inStream.readLine();
  if(line.trimmed()!=QString("</config>")) return false;

  if(isAMatrix=="false") rMatrixAction->activate(QAction::Trigger);
  else aMatrixAction->activate(QAction::Trigger);
  GetConfig().outputdir=outputDirectory.toStdString();
  GetConfig().checkdir=checksDirectory.toStdString();
  if(compoundCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_COMPOUND_NUCLEUS;
  else if(compoundCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_COMPOUND_NUCLEUS;
  if(boundaryCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_BOUNDARY_CONDITIONS;
  else if(boundaryCheck=="screen")  GetConfig().screenCheckMask |= Config::CHECK_BOUNDARY_CONDITIONS;
  if(dataCheck=="file")  GetConfig().fileCheckMask |= Config::CHECK_DATA;
  else if(dataCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_DATA;
  if(lMatrixCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_ENERGY_DEP;
  else if(lMatrixCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_ENERGY_DEP;
  if(legendreCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_LEGENDRE;
  else if(legendreCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_LEGENDRE;
  if(coulAmpCheck=="file")  GetConfig().fileCheckMask |= Config::CHECK_COUL_AMPLITUDES;
  else if(coulAmpCheck=="screen")  GetConfig().screenCheckMask |= Config::CHECK_COUL_AMPLITUDES;
  if(pathwaysCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_PATHWAYS;
  else if(pathwaysCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_PATHWAYS;
  if(angDistsCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_ANGULAR_DISTS;
  else if(angDistsCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_ANGULAR_DISTS;

  return true;
}

void AZURESetup::save() {
  if(!GetConfig().configfile.empty()) {
    if(!this->writeFile(QString::fromStdString(GetConfig().configfile)))
      QMessageBox::information(this,
                      tr("Can't Access File"),
                      tr("An error occured while writing the file."));
  } else saveAs();
}

void AZURESetup::saveAs() {
  QString filename = QFileDialog::getSaveFileName(this);
  if(!filename.isEmpty()) {
    if(!this->writeFile(filename))
      QMessageBox::information(this,
                      tr("Can't Access File"),
                      tr("An error occured while writing the file."));
    else {
      QSettings settings;
      QStringList files = settings.value("recentFileList").toStringList();
      QFile file(filename);
      QFileInfo info(file);
      QString fullFileName = QDir::fromNativeSeparators(info.absoluteFilePath());
      files.removeAll(fullFileName);
      files.prepend(fullFileName);
      while(files.size()>numRecent) files.removeLast();
      settings.setValue("recentFileList",files);
      updateRecent();
    }
  }
}

bool AZURESetup::writeFile(QString filename) {
  QFile file(filename);
  if(!file.open(QIODevice::WriteOnly)) return false;
```

```
  QFileInfo info(file);
  QString directory=info.absolutePath();

  QTextStream out(&file);
  out « "<config>" « endl;
  if(!this->writeConfig(out,directory)) return false;
  out « "</config>" « endl;

  out « "<levels>" « endl;
  if(!levelsTab->writeNuclearFile(out)) return false;
  out « "</levels>" « endl;

  out « "<segmentsData>" « endl;
  if(!segmentsTab->writeSegDataFile(out)) return false;
  out « "</segmentsData>" « endl;

  out « "<segmentsTest>" « endl;
  if(!segmentsTab->writeSegTestFile(out)) return false;
  out « "</segmentsTest>" « endl;

  out « "<targetInt>" « endl;
  if(!targetIntTab->writeFile(out)) return false;
  out « "</targetInt>" « endl;

  out « "<lastRun>" « endl;
  if(!writeLastRun(out)) return false;
  out « "</lastRun>" « endl;

  GetConfig().configfile=QDir::fromNativeSeparators(info.absoluteFilePath()).toStdString();
  setWindowTitle(QString("AZURE2 -- %1").arg(QString::fromStdString(GetConfig().configfile)));
  QDir::setCurrent(directory);

  out.flush();
  file.close();
  return true;
}

bool AZURESetup::writeConfig(QTextStream& outStream, QString directory) {
  QString isAMatrix;
  QString outputDirectory;
  QString checksDirectory;
  QString compoundCheck;
  QString boundaryCheck;
  QString dataCheck;
  QString lMatrixCheck;
  QString legendreCheck;
  QString coulAmpCheck;
  QString pathwaysCheck;
  QString angDistsCheck;

  if(GetConfig().paramMask & Config::USE_AMATRIX) isAMatrix="true";
  else isAMatrix="false";
  bool emptyCheckDir=false;
  bool emptyOutputDir=false;
  if(!GetConfig().outputdir.empty())
    outputDirectory=QString::fromStdString(GetConfig().outputdir);
  else {
    outputDirectory=QDir::fromNativeSeparators(directory)+'/';
    GetConfig().outputdir=outputDirectory.toStdString();
    emptyOutputDir=true;
  }
  if(!GetConfig().checkdir.empty())
    checksDirectory=QString::fromStdString(GetConfig().checkdir);
  else {
    checksDirectory=QDir::fromNativeSeparators(directory)+'/';
    GetConfig().checkdir=checksDirectory.toStdString();
    emptyCheckDir=true;
  }
  if(emptyCheckDir&&emptyOutputDir) {
    QMessageBox::information(this,tr("Unspecified Directories"),
                 QString("The output and checks directories are unspecified. "
                     "They will be set to %1.").arg(outputDirectory.trimmed()));
  } else if(emptyCheckDir) {
    QMessageBox::information(this,tr("Unspecified Directory"),
                 QString("The checks directory is unspecified. "
                     "It will be set to %1.").arg(checksDirectory.trimmed()));
  } else if(emptyOutputDir) {
    QMessageBox::information(this,tr("Unspecified Directory"),
                 QString("The output directory is unspecified. "
                     "It will be set to %1.").arg(outputDirectory.trimmed()));
  }
  if(GetConfig().fileCheckMask & Config::CHECK_COMPOUND_NUCLEUS) compoundCheck="file";
  else if(GetConfig().screenCheckMask & Config::CHECK_COMPOUND_NUCLEUS) compoundCheck="screen";
  else compoundCheck="none";
  if(GetConfig().fileCheckMask & Config::CHECK_BOUNDARY_CONDITIONS) boundaryCheck="file";
  else if(GetConfig().screenCheckMask & Config::CHECK_BOUNDARY_CONDITIONS) boundaryCheck="screen";
  else boundaryCheck="none";
```

```
  if(GetConfig().fileCheckMask & Config::CHECK_DATA) dataCheck="file";
  else if(GetConfig().screenCheckMask & Config::CHECK_DATA) dataCheck="screen";
  else dataCheck="none";
  if(GetConfig().fileCheckMask & Config::CHECK_ENERGY_DEP) lMatrixCheck="file";
  else if(GetConfig().screenCheckMask & Config::CHECK_ENERGY_DEP) lMatrixCheck="screen";
  else lMatrixCheck="none";
  if(GetConfig().fileCheckMask & Config::CHECK_LEGENDRE) legendreCheck="file";
  else if(GetConfig().screenCheckMask & Config::CHECK_LEGENDRE) legendreCheck="screen";
  else legendreCheck="none";
  if(GetConfig().fileCheckMask & Config::CHECK_COUL_AMPLITUDES) coulAmpCheck="file";
  else if(GetConfig().screenCheckMask & Config::CHECK_COUL_AMPLITUDES) coulAmpCheck="screen";
  else coulAmpCheck="none";
  if(GetConfig().fileCheckMask & Config::CHECK_PATHWAYS) pathwaysCheck="file";
  else if(GetConfig().screenCheckMask & Config::CHECK_PATHWAYS) pathwaysCheck="screen";
  else pathwaysCheck="none";
  if(GetConfig().fileCheckMask & Config::CHECK_ANGULAR_DISTS) angDistsCheck="file";
  else if(GetConfig().screenCheckMask & Config::CHECK_ANGULAR_DISTS) angDistsCheck="screen";
  else angDistsCheck="none";

  outStream.setFieldAlignment(QTextStream::AlignLeft);
  outStream « qSetFieldWidth(100) « isAMatrix « qSetFieldWidth(0) « "#Perform A-Matrix Calculation" « endl;
  outStream « qSetFieldWidth(100) « outputDirectory « qSetFieldWidth(0) « "#Full Path to Output Directory" « endl;
  outStream « qSetFieldWidth(100) « checksDirectory « qSetFieldWidth(0) « "#Full Path to Checks Directory" « endl;
  outStream « qSetFieldWidth(100) « compoundCheck « qSetFieldWidth(0) « "#Compond Nucleus Check" « endl;
  outStream « qSetFieldWidth(100) « boundaryCheck « qSetFieldWidth(0) « "#Boundary Condition Check" « endl;
  outStream « qSetFieldWidth(100) « dataCheck « qSetFieldWidth(0) « "#Data Check" « endl;
  outStream « qSetFieldWidth(100) « lMatrixCheck « qSetFieldWidth(0) « "#Lo-Matrix and Penetrability Check" « endl;
  outStream « qSetFieldWidth(100) « legendreCheck « qSetFieldWidth(0) « "#Legendre Polynomial Check" « endl;
  outStream « qSetFieldWidth(100) « coulAmpCheck « qSetFieldWidth(0) « "#Coulomb Amplitudes Check" « endl;
  outStream « qSetFieldWidth(100) « pathwaysCheck « qSetFieldWidth(0) « "#Reaction Pathway Check" « endl;
  outStream « qSetFieldWidth(100) « angDistsCheck « qSetFieldWidth(0) « "#Angular Distributions Check" « endl;

  return true;
}

bool AZURESetup::writeLastRun(QTextStream& outStream) {
  unsigned int paramMask = GetConfig().paramMask;

  if(runTab->calcType->currentIndex()==1 ||
     runTab->calcType->currentIndex()==3) paramMask |= Config::PERFORM_FIT;
  else paramMask &= ~Config::PERFORM_FIT;
  if(runTab->calcType->currentIndex()==2||
     runTab->calcType->currentIndex()==4) paramMask &= ~Config::CALCULATE_WITH_DATA;
  else paramMask |= Config::CALCULATE_WITH_DATA;
  if(runTab->calcType->currentIndex()==3) paramMask |= Config::PERFORM_ERROR_ANALYSIS;
  else paramMask &= ~Config::PERFORM_ERROR_ANALYSIS;
  if(runTab->calcType->currentIndex()==4) paramMask |= Config::CALCULATE_REACTION_RATE;
  else paramMask &= ~Config::CALCULATE_REACTION_RATE;

  if(runTab->oldParamFileButton->isChecked())
    paramMask |= Config::USE_PREVIOUS_PARAMETERS;
  else paramMask &= ~Config::USE_PREVIOUS_PARAMETERS;
  if(runTab->oldIntegralsFileButton->isChecked())
    paramMask |= Config::USE_PREVIOUS_INTEGRALS;
  else paramMask &= ~Config::USE_PREVIOUS_INTEGRALS;

  outStream « paramMask « endl;
  outStream « '"' « runTab->paramFileText->text() « '"' « endl;
  outStream « '"' « runTab->integralsFileText->text() « '"' « endl;
  if(!runTab->rateEntranceKey->text().isEmpty()) outStream « runTab->rateEntranceKey->text() « ' ';
  else outStream « "0 ";
  if(!runTab->rateExitKey->text().isEmpty()) outStream « runTab->rateExitKey->text();
  else outStream « 0;
  outStream « endl;
  if(runTab->fileTempButton->isChecked()) outStream « "1 ";
  else outStream « "0 ";
  outStream « '"' « runTab->fileTempText->text() « '"' « endl;
  if(!runTab->minTempText->text().isEmpty()) outStream « runTab->minTempText->text() « ' ';
  else outStream « "-1. ";
  if(!runTab->maxTempText->text().isEmpty()) outStream « runTab->maxTempText->text() « ' ';
  else outStream « "-1. ";
  if(!runTab->tempStepText->text().isEmpty()) outStream « runTab->tempStepText->text();
  else outStream « "-1.";
  outStream « endl;

  return true;
}
```

```

void AZURESetup::matrixChanged(QAction *action) {
  if(action==aMatrixAction) GetConfig().paramMask |= Config::USE_AMATRIX;
  else {
    if(GetConfig().paramMask & Config::IGNORE_ZERO_WIDTHS) {
      QMessageBox::information(this,tr("Incompatible Option"),
                   tr("The option to ignore external widths is not possible for R-Matrix formalism. Remove option to use R-Matrix formalism."));
      aMatrixAction->activate(QAction::Trigger);
    } else GetConfig().paramMask &= ~Config::USE_AMATRIX;
  }
}

void AZURESetup::editChecks() {
  EditChecksDialog aDialog;
  if(GetConfig().fileCheckMask & Config::CHECK_COMPOUND_NUCLEUS)
    aDialog.compoundCheckCombo->setCurrentIndex(2);
  else if(GetConfig().screenCheckMask & Config::CHECK_COMPOUND_NUCLEUS)
    aDialog.compoundCheckCombo->setCurrentIndex(1);
  else aDialog.compoundCheckCombo->setCurrentIndex(0);

  if(GetConfig().fileCheckMask & Config::CHECK_BOUNDARY_CONDITIONS)
    aDialog.boundaryCheckCombo->setCurrentIndex(2);
  else if(GetConfig().screenCheckMask & Config::CHECK_BOUNDARY_CONDITIONS)
    aDialog.boundaryCheckCombo->setCurrentIndex(1);
  else aDialog.boundaryCheckCombo->setCurrentIndex(0);

  if(GetConfig().fileCheckMask & Config::CHECK_DATA)
    aDialog.dataCheckCombo->setCurrentIndex(2);
  else if(GetConfig().screenCheckMask & Config::CHECK_DATA)
    aDialog.dataCheckCombo->setCurrentIndex(1);
  else aDialog.dataCheckCombo->setCurrentIndex(0);

  if(GetConfig().fileCheckMask & Config::CHECK_ENERGY_DEP)
    aDialog.lMatrixCheckCombo->setCurrentIndex(2);
  else if(GetConfig().screenCheckMask & Config::CHECK_ENERGY_DEP)
    aDialog.lMatrixCheckCombo->setCurrentIndex(1);
  else aDialog.lMatrixCheckCombo->setCurrentIndex(0);

  if(GetConfig().fileCheckMask & Config::CHECK_LEGENDRE)
    aDialog.legendreCheckCombo->setCurrentIndex(2);
  else if(GetConfig().screenCheckMask & Config::CHECK_LEGENDRE)
    aDialog.legendreCheckCombo->setCurrentIndex(1);
  else aDialog.legendreCheckCombo->setCurrentIndex(0);

  if(GetConfig().fileCheckMask & Config::CHECK_COUL_AMPLITUDES)
    aDialog.coulAmpCheckCombo->setCurrentIndex(2);
  else if(GetConfig().screenCheckMask & Config::CHECK_COUL_AMPLITUDES)
    aDialog.coulAmpCheckCombo->setCurrentIndex(1);
  else aDialog.coulAmpCheckCombo->setCurrentIndex(0);

  if(GetConfig().fileCheckMask & Config::CHECK_PATHWAYS)
    aDialog.pathwaysCheckCombo->setCurrentIndex(2);
  else if(GetConfig().screenCheckMask & Config::CHECK_PATHWAYS)
    aDialog.pathwaysCheckCombo->setCurrentIndex(1);
  else aDialog.pathwaysCheckCombo->setCurrentIndex(0);

  if(GetConfig().fileCheckMask & Config::CHECK_ANGULAR_DISTS)
    aDialog.angDistsCheckCombo->setCurrentIndex(2);
  else if(GetConfig().screenCheckMask & Config::CHECK_ANGULAR_DISTS)
    aDialog.angDistsCheckCombo->setCurrentIndex(1);
  else aDialog.angDistsCheckCombo->setCurrentIndex(0);

  if(aDialog.exec()) {
    GetConfig().fileCheckMask=0;
    GetConfig().screenCheckMask=0;
    if(aDialog.compoundCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |= Config::CHECK_COMPOUND_NUCLEUS;
    else if(aDialog.compoundCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |= Config::CHECK_COMPOUND_NUCLEUS;
    if(aDialog.boundaryCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |= Config::CHECK_BOUNDARY_CONDITIONS;
    else if(aDialog.boundaryCheckCombo->currentIndex()==2)  GetConfig().fileCheckMask |= Config::CHECK_BOUNDARY_CONDITIONS;
    if(aDialog.dataCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |= Config::CHECK_DATA;
    else if(aDialog.dataCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |= Config::CHECK_DATA;
    if(aDialog.lMatrixCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |= Config::CHECK_ENERGY_DEP;
    else if(aDialog.lMatrixCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |= Config::CHECK_ENERGY_DEP;
    if(aDialog.legendreCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |= Config::CHECK_LEGENDRE;
    else if(aDialog.legendreCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |= Config::CHECK_LEGENDRE;
    if(aDialog.coulAmpCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |= Config::CHECK_COUL_AMPLITUDES;
```

```
    else if(aDialog.coulAmpCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |= Config::CHECK_COUL_AMPLITUDES;
    if(aDialog.pathwaysCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |= Config::CHECK_PATHWAYS;
    else if(aDialog.pathwaysCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |= Config::CHECK_PATHWAYS;
    if(aDialog.angDistsCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |= Config::CHECK_ANGULAR_DISTS;
    else if(aDialog.angDistsCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |= Config::CHECK_ANGULAR_DISTS;
  }
}

void AZURESetup::editDirs() {
  EditDirsDialog aDialog;

  aDialog.outputDirectoryText->setText(QString::fromStdString(GetConfig().outputdir));
  aDialog.checksDirectoryText->setText(QString::fromStdString(GetConfig().checkdir));

  if(aDialog.exec()) {
    GetConfig().outputdir=aDialog.outputDirectoryText->text().toStdString();
    GetConfig().checkdir=aDialog.checksDirectoryText->text().toStdString();
  }
}

void AZURESetup::editOptions() {
  EditOptionsDialog aDialog;

  if(GetConfig().paramMask & Config::USE_GSL_COULOMB_FUNC) aDialog.useGSLCoulCheck->setChecked(true);
  else aDialog.useGSLCoulCheck->setChecked(false);

  if(GetConfig().paramMask & Config::USE_BRUNE_FORMALISM) aDialog.useBruneCheck->setChecked(true);
  else aDialog.useBruneCheck->setChecked(false);

  if(GetConfig().paramMask & Config::IGNORE_ZERO_WIDTHS) aDialog.ignoreExternalsCheck->setChecked(true);
  else aDialog.ignoreExternalsCheck->setChecked(false);

  if(GetConfig().paramMask & Config::USE_RMC_FORMALISM) aDialog.useRMCCheck->setChecked(true);
  else aDialog.useRMCCheck->setChecked(false);

  if(!(GetConfig().paramMask & Config::TRANSFORM_PARAMETERS)) aDialog.noTransformCheck->setChecked(true);
  else aDialog.noTransformCheck->setChecked(false);

  //if(!(GetConfig().paramMask & Config::USE_LONGWAVELENGTH_APPROX)) aDialog.noLongWavelengthCheck->setChecked(true);
  //else aDialog.noLongWavelengthCheck->setChecked(false);

  if(aDialog.exec()) {
    if(aDialog.useGSLCoulCheck->isChecked()) GetConfig().paramMask |= Config::USE_GSL_COULOMB_FUNC;
    else GetConfig().paramMask &= ~Config::USE_GSL_COULOMB_FUNC;

    if(aDialog.useBruneCheck->isChecked())
      GetConfig().paramMask |= Config::USE_BRUNE_FORMALISM;
    else GetConfig().paramMask &= ~Config::USE_BRUNE_FORMALISM;

    if(aDialog.ignoreExternalsCheck->isChecked()) {
      GetConfig().paramMask |= Config::IGNORE_ZERO_WIDTHS;
      if(!(GetConfig().paramMask & Config::USE_AMATRIX)) {
    QMessageBox::information(this,tr("Incompatible Option"),
                 tr("The option to ignore external widths is not possible for R-Matrix formalism.  The formalism will be changed to A-Matrix. "));
    aMatrixAction->activate(QAction::Trigger);
      }
    } else  GetConfig().paramMask &= ~Config::IGNORE_ZERO_WIDTHS;

    if(aDialog.useRMCCheck->isChecked()) GetConfig().paramMask |= Config::USE_RMC_FORMALISM;
    else GetConfig().paramMask &= ~Config::USE_RMC_FORMALISM;

    if(aDialog.noTransformCheck->isChecked()) GetConfig().paramMask &= ~Config::TRANSFORM_PARAMETERS;
    else GetConfig().paramMask |= Config::TRANSFORM_PARAMETERS;

    //if(aDialog.noLongWavelengthCheck->isChecked()) GetConfig().paramMask &= ~Config::USE_LONGWAVELENGTH_APPROX;
    //else GetConfig().paramMask |= Config::USE_LONGWAVELENGTH_APPROX;
  }
}

void AZURESetup::SaveAndRun() {
  save();
  if(GetConfig().configfile.empty()) return;
  runTab->runtimeText->clear();
  QFile file(QString::fromStdString(GetConfig().configfile));
  QFileInfo info(file);
  QString directory=info.absolutePath();
  if(GetConfig().outputdir.empty())
```

```
GetConfig().outputdir=QDir::fromNativeSeparators(directory).toStdString()+'/';
  if(GetConfig().checkdir.empty())
GetConfig().checkdir=QDir::fromNativeSeparators(directory).toStdString()+'/';

  GetConfig().chiVariance=runTab->chiVarianceText->text().toDouble();

  if(runTab->calcType->currentIndex()==1 ||
     runTab->calcType->currentIndex()==3) GetConfig().paramMask |= Config::PERFORM_FIT;
  else GetConfig().paramMask &= ~Config::PERFORM_FIT;
  if(runTab->calcType->currentIndex()==2||
     runTab->calcType->currentIndex()==4) GetConfig().paramMask &= ~Config::CALCULATE_WITH_DATA;
  else GetConfig().paramMask |= Config::CALCULATE_WITH_DATA;
  if(runTab->calcType->currentIndex()==3) GetConfig().paramMask |= Config::PERFORM_ERROR_ANALYSIS;
  else GetConfig().paramMask &= ~Config::PERFORM_ERROR_ANALYSIS;
  if(runTab->calcType->currentIndex()==4) GetConfig().paramMask |= Config::CALCULATE_REACTION_RATE;
  else GetConfig().paramMask &= ~Config::CALCULATE_REACTION_RATE;

  if(runTab->oldParamFileButton->isChecked()) {
    GetConfig().paramMask |= Config::USE_PREVIOUS_PARAMETERS;
    GetConfig().paramfile=runTab->paramFileText->text().toStdString();
  } else GetConfig().paramMask &= ~Config::USE_PREVIOUS_PARAMETERS;

  std::vector<SegPairs> segPairs;
  if(!(GetConfig().paramMask & Config::CALCULATE_REACTION_RATE)) {
    if(!readSegmentFile(GetConfig(),segPairs)) return;
  } else {
    GetConfig().rateParams.entrancePair=runTab->rateEntranceKey->text().toInt();
    GetConfig().rateParams.exitPair=runTab->rateExitKey->text().toInt();
    if(GetConfig().rateParams.entrancePair==GetConfig().rateParams.exitPair) {
      QMessageBox::information(this,tr("No Scattering Rates"),
                   tr("Reaction rates cannot be calculated for elastic scattering."));
      return;
    }
    if(!runTab->fileTempButton->isChecked()) {
      GetConfig().rateParams.useFile=false;
      GetConfig().rateParams.minTemp = runTab->minTempText->text().toDouble();
      GetConfig().rateParams.maxTemp = runTab->maxTempText->text().toDouble();
      GetConfig().rateParams.tempStep = runTab->tempStepText->text().toDouble();
    } else {
      GetConfig().rateParams.useFile=true;
      GetConfig().rateParams.temperatureFile = runTab->fileTempText->text().toStdString();
    }
    SegPairs tempPair = {runTab->rateEntranceKey->text().toInt(),
             runTab->rateExitKey->text().toInt()};
    segPairs.push_back(tempPair);
  }
  if(segPairs.size()==0) {
    QMessageBox::information(this,tr("Empty Segments"),tr("No active segments have been found."));
    return;
  }
  int maxPairs=pairsTab->getPairsModel()->getPairs().size();
  for(std::vector<SegPairs>::const_iterator it = segPairs.begin();it<segPairs.end();it++) {
    if(it->secondPair==-1) {
      QList<PairsData> pairsList = pairsTab->getPairsModel()->getPairs();
      int i;
      for(i = 0; i<pairsList.size();i++)
    if(pairsList[i].pairType==10) break;
      if(i==pairsList.size()) {
    QMessageBox::information(this,tr("No Capture Pairs"),
                 tr("Total capture is specified, but no capture pairs exist."));
    return;
      }
    } else if(it->firstPair>maxPairs||it->secondPair>maxPairs||it->firstPair<1||it->secondPair<1) {
      QMessageBox::information(this,tr("Undefined Key"),tr("An undefined pair key is specified."));
      return;
    }
  }

  GetConfig().paramMask &= ~Config::USE_EXTERNAL_CAPTURE;
  if(!checkExternalCapture(GetConfig(),segPairs)) return;
  if(GetConfig().paramMask &Config::USE_EXTERNAL_CAPTURE) {
    if(runTab->oldIntegralsFileButton->isChecked() &&
       !(GetConfig().paramMask & Config::CALCULATE_REACTION_RATE)) {
      GetConfig().paramMask |= Config::USE_PREVIOUS_INTEGRALS;
      GetConfig().integralsfile=runTab->integralsFileText->text().toStdString();
    } else GetConfig().paramMask &= ~Config::USE_PREVIOUS_INTEGRALS;
  }

  if(!QDir(QString::fromStdString(GetConfig().outputdir)).exists()) {
    QMessageBox::information(this,tr("Directory Doesn't Exist"),
                 tr("The specified output directory doesn't exist."));
    return;
  }
  if(!QDir(QString::fromStdString(GetConfig().checkdir)).exists()) {
    QMessageBox::information(this,tr("Directory Doesn't Exist"),
                 tr("The specified checks directory doesn't exist."));
    return;
```

```
    }
    if((GetConfig().paramMask & Config::USE_PREVIOUS_PARAMETERS) &&
       !QFile(QString::fromStdString(GetConfig().paramfile)).exists()) {
      QMessageBox::information(this,tr("File Doesn't Exist"),
                               tr("The specified parameter file doesn't exist."));
      return;
    }
    if(((GetConfig().paramMask & Config::USE_PREVIOUS_INTEGRALS) &&
        (GetConfig().paramMask & Config::USE_EXTERNAL_CAPTURE)) &&
       !QFile(QString::fromStdString(GetConfig().integralsfile)).exists()) {
      QMessageBox::information(this,tr("File Doesn't Exist"),
                               tr("The specified integrals file doesn't exist."));
      return;
    }
    if((GetConfig().paramMask & Config::CALCULATE_REACTION_RATE &&
        GetConfig().rateParams.useFile) &&
       !QFile(QString::fromStdString(GetConfig().rateParams.temperatureFile)).exists()) {
      QMessageBox::information(this,tr("File Doesn't Exist"),
                               tr("The specified rate temperature file doesn't exist."));
      return;
    }

    if(!(GetConfig().paramMask & Config::CALCULATE_WITH_DATA) &&
       !(GetConfig().paramMask & Config::CALCULATE_REACTION_RATE)) {
      QList<TargetIntData> targetIntData = targetIntTab->getTargetIntModel()->getLines();
      QList<SegmentsTestData> segmentsTestData=segmentsTab->getSegmentsTestModel()->getLines();
      for(unsigned int i=0;i<targetIntData.size();i++) {
        if(targetIntData.at(i).isActive==1&&
       (targetIntData.at(i).isTargetIntegration||targetIntData.at(i).isConvolution)) {
      unsigned int j=0;
      unsigned int lastSegNum=0;
      bool inclusive=false;
      QList<unsigned int> tempList;
      QString segmentsList = targetIntData.at(i).segmentsList;
      while(j<segmentsList.length()) {
        if(segmentsList[j]>='0'&&segmentsList[j]<='9') {
          QString tempString;
          while(segmentsList[j]!=','&&segmentsList[j]!='-'&&
            j<segmentsList.length()) {
            tempString+=segmentsList[j];
            j++;
          }
          QTextStream stm(&tempString);
          unsigned int tempSegNum;stm»tempSegNum;
          if(inclusive==true) for(int k=lastSegNum+1;k<=tempSegNum;k++)
                    tempList.push_back(k);
          else tempList.push_back(tempSegNum);
          lastSegNum=tempSegNum;
        }
        if(segmentsList[j]=='-') inclusive=true;
        else inclusive =false;
        j++;
      }
      bool isAngularDistribution=false;
      for(j=0;j<tempList.size();j++) {
        if(tempList.at(j)<=segmentsTestData.size()) {
          for(int k = 0; k<segmentsTestData.size(); k++) {
            if(segmentsTestData.at(k).isActive==1&&
           tempList.at(j)-1==k&&
           segmentsTestData.at(k).dataType==3) {
          isAngularDistribution=true;
          break;
            }
          }
        }
        if(isAngularDistribution) break;
      }
      if(isAngularDistribution) {
        QMessageBox::information(this,tr("Incompatable Options"),
                                 tr("Angular distribution coefficients cannot be used with convolution or target
integration."));
        return;
      }
        }
      }
    }

    azureMain = new AZUREMainThread(runTab,GetConfig());
    connect(azureMain,SIGNAL(finished()),this,SLOT(DeleteThread()));
    setWindowTitle(QString("AZURE2 -- %1 --
Running").arg(QString::fromStdString(GetConfig().configfile)));
    runTab->calcButton->setEnabled(false);
    runTab->stopAZUREButton->setEnabled(true);
    runTab->runtimeText->SetMouseFiltered(true);
    startMessage(azureMain->configure());
    azureMain->start();
}
```

```

void AZURESetup::DeleteThread() {
  exitMessage(azureMain->configure());
  QScrollBar *sb = runTab->runtimeText->verticalScrollBar();
  sb->setValue(sb->maximum());

  setWindowTitle(QString("AZURE2 -- %1").arg(QString::fromStdString(GetConfig().configfile)));
  runTab->calcButton->setEnabled(true);
  runTab->stopAZUREButton->setEnabled(false);
  runTab->runtimeText->SetMouseFiltered(false);
  delete azureMain;
}

void AZURESetup::showAbout() {
  AboutAZURE2Dialog aboutDialog;
  aboutDialog.exec();
}

void AZURESetup::reset() {
  GetConfig().Reset();
  aMatrixAction->activate(QAction::Trigger);
  levelsTab->reset();
  segmentsTab->reset();
  targetIntTab->reset();
  runTab->reset();
#ifdef USE_QWT
  plotTab->reset();
#endif
  setWindowTitle(tr("AZURE2 -- untitled"));
  GetConfig().configfile="";
}

void AZURESetup::showTabInfo() {
  QString tabTitle = tabWidget->tabText(tabWidget->currentIndex()).remove(QChar('&'));
  if(tabWidget->currentIndex()==0) pairsTab->showInfo(0,tabTitle);
  if(tabWidget->currentIndex()==1) levelsTab->showInfo(0,tabTitle);
  if(tabWidget->currentIndex()==2) segmentsTab->showInfo(0,tabTitle);
  if(tabWidget->currentIndex()==3) targetIntTab->showInfo(0,tabTitle);
  if(tabWidget->currentIndex()==4) runTab->showInfo(0,tabTitle);
#ifdef USE_QWT
  if(tabWidget->currentIndex()==5) plotTab->showInfo(0,tabTitle);
#endif
}

void AZURESetup::openWebsite() {
  if(!QDesktopServices::openUrl(QUrl("https://azure.nd.edu")))
    QMessageBox::information(this,
                 tr("Can't Open Browser"),
                 tr("AZURE2 could not access your web browser.  "
                "Please navitgate to https://azure.nd.edu/ "
                "to visit the website."));
}
```

8.93 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChannelDetails.cpp File Reference

```
#include <QLabel>
#include <QLineEdit>
#include <QGridLayout>
#include <QVBoxLayout>
#include "ChannelDetails.h"
```

8.94 ChannelDetails.cpp

Go to the documentation of this file.

```
#include <QLabel>
#include <QLineEdit>
#include <QGridLayout>
#include <QVBoxLayout>

```

```
#include "ChannelDetails.h"

ChannelDetails::ChannelDetails(QWidget *parent) : QWidget(parent) {
  details=new QLabel;
  QFont font("Monospace");
  font.setStyleHint(QFont::TypeWriter);
  details->setFont(font);
  reducedWidthText = new QLineEdit;
  reducedWidthText->setMaximumWidth(100);
  normParam=new QLabel;
  normUnits=new QLabel;

  QGridLayout *reducedWidthLayout=new QGridLayout;
  reducedWidthLayout->addWidget(normParam,0,0);
  reducedWidthLayout->addWidget(reducedWidthText,0,1);
  reducedWidthLayout->addWidget(normUnits,0,2);
  reducedWidthLayout->addItem(new QSpacerItem(20,20),0,3);
  reducedWidthLayout->setColumnStretch(3,1);

  QVBoxLayout *mainLayout = new QVBoxLayout;
  mainLayout->addWidget(details);
  mainLayout->addLayout(reducedWidthLayout);
  setLayout(mainLayout);
}

void ChannelDetails::setNormParam(int which) {
  if(which==1) {
    normParam->setText("ANC:");
    normUnits->setText("fm^(-1/2)");
  } else if(which==2) {
    normParam->setText("Mu:");
    normUnits->setText("nm");
  } else if(which==3) {
    normParam->setText("Q:");
    normUnits->setText("b");
  } else if(which==4) {
    normParam->setText("B:");
    normUnits->setText("");
  } else {
    normParam->setText("Partial Width:");
    normUnits->setText("eV");
  }
}
```

8.95 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChannelsModel.cpp File Reference

```
#include "ChannelsModel.h"
#include "PairsModel.h"
```

8.96 ChannelsModel.cpp

Go to the documentation of this file.

```
#include "ChannelsModel.h"
#include "PairsModel.h"

ChannelsModel::ChannelsModel(QObject *parent) : QAbstractTableModel(parent) {
}

int ChannelsModel::rowCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return channelsList.size();
}

int ChannelsModel::columnCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return ChannelsData::SIZE;
}

QVariant ChannelsModel::data(const QModelIndex &index, int role) const {
  if(!index.isValid()) return QVariant();

```

```
  if(index.row() >= channelsList.size() || index.row() < 0)  return QVariant();

  if (role == Qt::DisplayRole) {
    ChannelsData channel = channelsList.at(index.row());
    if(index.column() == 1) return channel.levelIndex;
    else if(index.column() == 2) {
      PairsData pair=pairsModel->getPairs().at(channel.pairIndex);
      if(channel.reducedWidth!=0.)
    return QString("<center><font
style='font-weight:bold;'>%1</font></center>").arg(pairsModel->getParticleLabel(pair));
      else return QString("<center>%1</center>").arg(pairsModel->getParticleLabel(pair));
    } else if(index.column() == 3) {
       if(channel.reducedWidth!=0.)
     return QString("<center><font
style='font-weight:bold;'>%1</font></center>").arg(getSpinLabel(channel));
       else return QString("<center>%1</center>").arg(getSpinLabel(channel));
    } else if(index.column() == 4) {
      if(channel.radType=='P') {
    if(channel.reducedWidth!=0.)
      return QString("<center><font
style='font-weight:bold;'>%1</font></center>").arg(channel.lValue);
    else return QString("<center>%1</center>").arg(channel.lValue);
      } else if(channel.radType=='F') {
    if(channel.reducedWidth!=0.)
      return QString("<center><font style='font-weight:bold;'>F</font></center>");
    else return QString("<center>F</center>");
      } else if(channel.radType=='G') {
    if(channel.reducedWidth!=0.)
      return QString("<center><font style='font-weight:bold;'>GT</font></center>");
    else return QString("<center>GT</center>");
      } else {
    if(channel.reducedWidth!=0.)
      return QString("<center><font
style='font-weight:bold;'>%1%2</font></center>").arg(channel.radType).arg(channel.lValue);
    else return QString("<center>%1%2</center>").arg(channel.radType).arg(channel.lValue);
      }
    } else if(index.column() == 5) return channel.radType;
    else if(index.column() == 6) return channel.reducedWidth;
  } else if(role==Qt::EditRole) {
    ChannelsData channel = channelsList.at(index.row());
    if(index.column() == 1) return channel.levelIndex;
    else if(index.column() == 2) return channel.pairIndex;
    else if(index.column() == 3) return channel.sValue;
    else if(index.column() == 4) return channel.lValue;
    else if(index.column() == 5) return channel.radType;
    else if(index.column() == 6) return channel.reducedWidth;
  } else if(role==Qt::TextAlignmentRole) return Qt::AlignCenter;
  else if (role==Qt::CheckStateRole && index.column()==0) {
    ChannelsData channel = channelsList.at(index.row());
    if(channel.isFixed==1) return Qt::Checked;
    else return Qt::Unchecked;
  }
  return QVariant();
}

QVariant ChannelsModel::headerData(int section, Qt::Orientation orientation, int role) const {
  if(role!= Qt::DisplayRole) return QVariant();
  if(orientation == Qt::Horizontal) {
    switch(section) {
    case 0:
      return tr("Fix?");
    case 1:
      return tr("level");
    case 2:
      return tr("Channel\nPair");
    case 3:
      return tr("s");
    case 4:
      return tr("l");
    case 5:
      return tr("radiation type");
    case 6:
      return tr("reduced width");
    default:
      return QVariant();
    }
  } else if(orientation == Qt::Vertical) {
    return section+1;
  }
  return QVariant();
}

bool ChannelsModel::setData(const QModelIndex &index, const QVariant &value, int role) {
  if (index.isValid()){
    if(role == Qt::EditRole ) {
      int row = index.row();
      ChannelsData tempData = channelsList.value(row);
```

```
      if (index.column() == 0) tempData.isFixed=value.toInt();
      else if(index.column() == 1) tempData.levelIndex=value.toInt();
      else if(index.column() == 2) tempData.pairIndex=value.toInt();
      else if(index.column() == 3) tempData.sValue=value.toDouble();
      else if(index.column() == 4) tempData.lValue=value.toInt();
      else if(index.column() == 5) tempData.radType=value.toChar();
      else if(index.column() == 6) tempData.reducedWidth=value.toDouble();
      else return false;

      channelsList.replace(row,tempData);
      if(index.column()!=6) emit(dataChanged(index,index));
      return true;
    } else if(role== Qt::CheckStateRole) {
      int row = index.row();
      ChannelsData tempData = channelsList.value(row);
      if(index.column()==0) {
    if(value==Qt::Checked) tempData.isFixed=1;
    else tempData.isFixed=0;
      } else return false;

      channelsList.replace(row,tempData);
      emit(dataChanged(index,index));
      return true;
    }
  }
  return false;
}

bool ChannelsModel::insertRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginInsertRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows; row++) {
      ChannelsData tempData={0,-1,-1,0.0,0,'P',0.0};
      channelsList.insert(position,tempData);
    }
    endInsertRows();
  }
  return true;
}

bool ChannelsModel::removeRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginRemoveRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows;++row) {
      channelsList.removeAt(position);
    }
    endRemoveRows();
  }
  return true;
}

Qt::ItemFlags ChannelsModel::flags(const QModelIndex &index) const {
  if (!index.isValid()) return Qt::ItemIsEnabled;
  if(index.column()==0) return QAbstractTableModel::flags(index) | Qt::ItemIsUserCheckable;
  return QAbstractTableModel::flags(index);
}

bool ChannelsModel::isChannel(const ChannelsData &channel)  const {
  bool foundChannel=false;
  for(int i=0;i<channelsList.size();i++) {
    ChannelsData tempChannel=channelsList.value(i);
    if(tempChannel.levelIndex==channel.levelIndex&&
       tempChannel.pairIndex==channel.pairIndex&&
       tempChannel.sValue==channel.sValue&&
       tempChannel.lValue==channel.lValue&&
       tempChannel.radType==channel.radType) {
      foundChannel=true;
      break;
    }
  }
  return foundChannel;

}

QString ChannelsModel::getSpinLabel(const ChannelsData &channel) const {
  if(((int)(channel.sValue*2))%2!=0&&channel.sValue!=0.) return QString("%1/2").arg((int)(channel.sValue*2));
  else return QString("%1").arg(channel.sValue);
}

void ChannelsModel::setPairsModel(PairsModel* model) {
  pairsModel=model;
}
```

8.97 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChooseFile←Button.cpp File Reference

```
#include ¨ChooseFileButton.h¨
```

8.98 ChooseFileButton.cpp

Go to the documentation of this file.

```
#include "ChooseFileButton.h"

ChooseFileButton::ChooseFileButton(const QString& text, QWidget *parent) :
  QPushButton(text, parent) {
    connect(this,SIGNAL(clicked()),this,SLOT(click()));
};

void ChooseFileButton::setLineEdit(QLineEdit* lineEdit) {
  thisLineEdit=lineEdit;
};

void ChooseFileButton::click() {
  emit(clicked(thisLineEdit));
};

```

8.99 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditChecks←Dialog.cpp File Reference

```
#include <QComboBox>
#include <QGridLayout>
#include <QLabel>
#include <QPushButton>
#include <QGroupBox>
#include ¨EditChecksDialog.h¨
```

8.100 EditChecksDialog.cpp

Go to the documentation of this file.

```
#include <QComboBox>
#include <QGridLayout>
#include <QLabel>
#include <QPushButton>
#include <QGroupBox>
#include "EditChecksDialog.h"

EditChecksDialog::EditChecksDialog(QWidget *parent) : QDialog(parent) {
    compoundCheckCombo=new QComboBox;
  compoundCheckCombo->addItem(tr("None"));
  compoundCheckCombo->addItem(tr("Screen"));
  compoundCheckCombo->addItem(tr("File"));
  boundaryCheckCombo=new QComboBox;
  boundaryCheckCombo->addItem(tr("None"));
  boundaryCheckCombo->addItem(tr("Screen"));
  boundaryCheckCombo->addItem(tr("File"));
  dataCheckCombo=new QComboBox;
  dataCheckCombo->addItem(tr("None"));
  dataCheckCombo->addItem(tr("Screen"));
  dataCheckCombo->addItem(tr("File"));
  lMatrixCheckCombo=new QComboBox;
  lMatrixCheckCombo->addItem(tr("None"));
```

```
lMatrixCheckCombo->addItem(tr("Screen"));
lMatrixCheckCombo->addItem(tr("File"));
legendreCheckCombo=new QComboBox;
legendreCheckCombo->addItem(tr("None"));
legendreCheckCombo->addItem(tr("Screen"));
legendreCheckCombo->addItem(tr("File"));
coulAmpCheckCombo=new QComboBox;
coulAmpCheckCombo->addItem(tr("None"));
coulAmpCheckCombo->addItem(tr("Screen"));
coulAmpCheckCombo->addItem(tr("File"));
pathwaysCheckCombo=new QComboBox;
pathwaysCheckCombo->addItem(tr("None"));
pathwaysCheckCombo->addItem(tr("Screen"));
pathwaysCheckCombo->addItem(tr("File"));
angDistsCheckCombo=new QComboBox;
angDistsCheckCombo->addItem(tr("None"));
angDistsCheckCombo->addItem(tr("Screen"));
angDistsCheckCombo->addItem(tr("File"));

QGroupBox *checkFilesBox=new QGroupBox(tr("Check Configuration"));
QGridLayout *checkFilesLayout = new QGridLayout;
checkFilesLayout->addWidget(new QLabel(tr("Coumpound Nucleus:")),0,0,Qt::AlignRight);
checkFilesLayout->addWidget(compoundCheckCombo,0,1);
checkFilesLayout->addWidget(new QLabel(tr("Boundary Conditions:")),1,0,Qt::AlignRight);
checkFilesLayout->addWidget(boundaryCheckCombo,1,1);
checkFilesLayout->addWidget(new QLabel(tr("Data:")),2,0,Qt::AlignRight);
checkFilesLayout->addWidget(dataCheckCombo,2,1);
checkFilesLayout->addWidget(new QLabel(tr("L-Matrix,Phases,Penetrabilities:")),3,0,Qt::AlignRight);
checkFilesLayout->addWidget(lMatrixCheckCombo,3,1);
checkFilesLayout->addWidget(new QLabel(tr("Legendre Polynomials:")),4,0,Qt::AlignRight);
checkFilesLayout->addWidget(legendreCheckCombo,4,1);
checkFilesLayout->addWidget(new QLabel(tr("Coulomb Amplitudes:")),5,0,Qt::AlignRight);
checkFilesLayout->addWidget(coulAmpCheckCombo,5,1);
checkFilesLayout->addWidget(new QLabel(tr("Reaction Pathways:")),6,0,Qt::AlignRight);
checkFilesLayout->addWidget(pathwaysCheckCombo,6,1);
checkFilesLayout->addWidget(new QLabel(tr("Angular Distributions:")),7,0,Qt::AlignRight);
checkFilesLayout->addWidget(angDistsCheckCombo,7,1);
checkFilesLayout->setColumnStretch(0,0);
checkFilesLayout->setColumnStretch(1,1);
checkFilesBox->setLayout(checkFilesLayout);

cancelButton = new QPushButton(tr("Cancel"));
okButton = new QPushButton(tr("Accept"));
okButton->setDefault(true);
connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
connect(cancelButton,SIGNAL(clicked()),this,SLOT(reject()));

QHBoxLayout *buttonBox = new QHBoxLayout;
buttonBox->addWidget(cancelButton);
buttonBox->addWidget(okButton);

QVBoxLayout *mainLayout = new QVBoxLayout;
mainLayout->addWidget(checkFilesBox);
mainLayout->addLayout(buttonBox);

setWindowTitle(tr("Edit Check Configuration"));

setLayout(mainLayout);
}
```

8.101 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditDirsDialog.cpp File Reference

```
#include <QGroupBox>
#include <QGridLayout>
#include <QLabel>
#include <QHBoxLayout>
#include <QFileDialog>
#include "EditDirsDialog.h"
#include "ChooseFileButton.h"
```

8.102 EditDirsDialog.cpp

Go to the documentation of this file.

```
#include <QGroupBox>
#include <QGridLayout>
#include <QLabel>
#include <QHBoxLayout>
#include <QFileDialog>

#include "EditDirsDialog.h"
#include "ChooseFileButton.h"

EditDirsDialog::EditDirsDialog(QWidget *parent) : QDialog(parent) {
  this->setMinimumWidth(500);
  QGroupBox *directoryBox=new QGroupBox(tr("Directory Configuration"));
  QGridLayout *directoryLayout = new QGridLayout;
  directoryLayout->addWidget(new QLabel(tr("Output Directory:")),0,0,Qt::AlignRight);
  outputDirectoryText = new QLineEdit;
  directoryLayout->addWidget(outputDirectoryText,0,1);
  ChooseFileButton *chooseButton=new ChooseFileButton(tr("Choose..."));
  chooseButton->setLineEdit(outputDirectoryText);
  connect(chooseButton,SIGNAL(clicked(QLineEdit*)),this,SLOT(setChooseDirectory(QLineEdit*)));
  directoryLayout->addWidget(chooseButton,0,2);
  directoryLayout->addWidget(new QLabel(tr("Checks Directory:")),1,0,Qt::AlignRight);
  checksDirectoryText = new QLineEdit;
  directoryLayout->addWidget(checksDirectoryText,1,1);
  chooseButton=new ChooseFileButton(tr("Choose..."));
  chooseButton->setLineEdit(checksDirectoryText);
  connect(chooseButton,SIGNAL(clicked(QLineEdit*)),this,SLOT(setChooseDirectory(QLineEdit*)));
  directoryLayout->addWidget(chooseButton,1,2);
  directoryBox->setLayout(directoryLayout);

  cancelButton = new QPushButton(tr("Cancel"));
  okButton = new QPushButton(tr("Accept"));
  okButton->setDefault(true);
  connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
  connect(cancelButton,SIGNAL(clicked()),this,SLOT(reject()));

  QHBoxLayout *buttonBox = new QHBoxLayout;
  buttonBox->addWidget(cancelButton);
  buttonBox->addWidget(okButton);

  QVBoxLayout *mainLayout = new QVBoxLayout;
  mainLayout->addWidget(directoryBox);
  mainLayout->addLayout(buttonBox);

  setWindowTitle(tr("Edit Directory Configuration"));

  setLayout(mainLayout);
}

void EditDirsDialog::setChooseDirectory(QLineEdit *lineEdit) {
  QString filename = QFileDialog::getExistingDirectory(this);
  if(!filename.isEmpty()) {
    lineEdit->setText(QDir::fromNativeSeparators(filename)+'/');
  }
}
```

8.103 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditOptionsDialog.cpp File Reference

```
#include "EditOptionsDialog.h"
#include <QCheckBox>
#include <QVBoxLayout>
#include <QPushButton>
#include <QGroupBox>
```

8.104 EditOptionsDialog.cpp

Go to the documentation of this file.

```cpp
#include "EditOptionsDialog.h"

#include <QCheckBox>
#include <QVBoxLayout>
#include <QPushButton>
#include <QGroupBox>

EditOptionsDialog::EditOptionsDialog(QWidget* parent) : QDialog(parent) {

  useGSLCoulCheck = new QCheckBox(tr("Use GSL Coulomb functions"));
  useBruneCheck = new QCheckBox(tr("Use Brune formalism"));
  ignoreExternalsCheck = new QCheckBox(tr("Ignore external width\nif internal width is zeroed"));
  useRMCCheck = new QCheckBox(tr("Use RMC capture formalism\n(neutron capture only)"));
  noTransformCheck = new QCheckBox(tr("Do not perform parameter\ntransformations"));
  // noLongWavelengthCheck = new QCheckBox(tr("Do not use long wavelength\n"
  //                         "approximation for EL external capture"));

  connect(useBruneCheck,SIGNAL(stateChanged(int)),this,SLOT(useBruneCheckChanged(int)));
  connect(useRMCCheck,SIGNAL(stateChanged(int)),this,SLOT(useRMCCheckChanged(int)));

  QGroupBox* optionsBox = new QGroupBox(tr("AZURE2 Options"));
  QVBoxLayout* optionsLayout = new QVBoxLayout;
  optionsLayout->addWidget(useGSLCoulCheck);
  optionsLayout->addWidget(useBruneCheck);
  optionsLayout->addWidget(ignoreExternalsCheck);
  optionsLayout->addWidget(useRMCCheck);
  optionsLayout->addWidget(noTransformCheck);
  //optionsLayout->addWidget(noLongWavelengthCheck);
  optionsBox->setLayout(optionsLayout);

  cancelButton = new QPushButton(tr("Cancel"));
  okButton = new QPushButton(tr("Accept"));
  okButton->setDefault(true);
  connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
  connect(cancelButton,SIGNAL(clicked()),this,SLOT(reject()));

  QHBoxLayout *buttonBox = new QHBoxLayout;
  buttonBox->addWidget(cancelButton);
  buttonBox->addWidget(okButton);
  QVBoxLayout *mainLayout = new QVBoxLayout;
  mainLayout->addWidget(optionsBox);
  mainLayout->addLayout(buttonBox);
  setWindowTitle(tr("Edit Options"));
  setLayout(mainLayout);
}

void EditOptionsDialog::useBruneCheckChanged(int state) {
  if(state==Qt::Checked) {
    useRMCCheck->setChecked(false);
    useRMCCheck->setEnabled(false);
  } else useRMCCheck->setEnabled(true);
}

void EditOptionsDialog::useRMCCheckChanged(int state) {
  if(state==Qt::Checked) {
    useBruneCheck->setChecked(false);
    useBruneCheck->setEnabled(false);
  } else useBruneCheck->setEnabled(true);
}
```

8.105 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InfoDialog.cpp File Reference

```
#include <QTextEdit>
#include <QPushButton>
#include <QVBoxLayout>
#include ¨InfoDialog.h¨
```

8.106 InfoDialog.cpp

Go to the documentation of this file.

```
#include <QTextEdit>
#include <QPushButton>
#include <QVBoxLayout>

#include "InfoDialog.h"

InfoDialog::InfoDialog(const QString& string,
                       QWidget* parent,
                       QString title) : QDialog(parent) {
  setWindowTitle(QString("Documentation for %1 Tab").arg(title));
  setMinimumSize(600,400);

  QTextEdit* textEdit = new QTextEdit;
  textEdit->setReadOnly(true);
  textEdit->setAcceptRichText(false);
  textEdit->setHtml(string);

  QPushButton*  okButton = new QPushButton(tr("OK"),this);
  okButton->setMaximumSize(80,30);

  QVBoxLayout* layout = new QVBoxLayout(this);
  layout->addWidget(textEdit);
  layout->addWidget(okButton);
  layout->setAlignment(okButton,Qt::AlignHCenter);

  connect(okButton,SIGNAL(clicked()),this,SLOT(close()));

  setLayout(layout);
}
```

8.107 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InTabDocs.cpp File Reference

```
#include "PairsTab.h"
#include "LevelsTab.h"
#include "SegmentsTab.h"
#include "TargetIntTab.h"
#include "RunTab.h"
#include "PlotTab.h"
```

Functions

- std::vector< QString > setInfoStrings (QString infoString1="", QString infoString2="", QString infoString3="", QString infoString4="", QString infoString5="")

8.107.1 Function Documentation

8.107.1.1 setInfoStrings()

```
std::vector< QString > setInfoStrings (
            QString infoString1 = "",
            QString infoString2 = "",
            QString infoString3 = "",
            QString infoString4 = "",
            QString infoString5 = "" )
```

Definition at line 8 of file InTabDocs.cpp.

8.108 InTabDocs.cpp

Go to the documentation of this file.

```
#include "PairsTab.h"
#include "LevelsTab.h"
#include "SegmentsTab.h"
#include "TargetIntTab.h"
#include "RunTab.h"
#include "PlotTab.h"

std::vector<QString> setInfoStrings(QString infoString1 = "",
                    QString infoString2 = "",
                    QString infoString3 = "",
                    QString infoString4 = "",
                    QString infoString5 = "") {
  std::vector<QString> stringVector;
  if(infoString1!="") stringVector.push_back(infoString1);
  if(infoString2!="") stringVector.push_back(infoString2);
  if(infoString3!="") stringVector.push_back(infoString3);
  if(infoString4!="") stringVector.push_back(infoString4);
  if(infoString5!="") stringVector.push_back(infoString5);
  return stringVector;
};

const std::vector<QString> PairsTab::infoText =
  setInfoStrings("<ul><li><p>This tab is used to define the particle pairs of the reaction. These are the two particles that fuse together to form or result from the decay of the compound nucleus. The theory is limited to two particle interactions. &gamma;-ray and &beta;-decays are currently only supported as decay pairs and must be distinguished by the different drop down selection when the pair is created. Since these decay pairs are specialized, some of the information fields are automatically filled in and may not be edited. &gamma;-ray decays are limited to bound states.</p></ul>"
"<ul><li><p>Use the &#43; or &#45; buttons on the lower left corner to add or delete a decay pair respectively.</p></ul>"
"<ul><li><p>When a particle pair is created it is assigned a number that is displayed on the left hand side of the particle pair row. These numbers are referenced when creating segments on the <i>Segments</i> tab.</p></ul>"
"<ul><li><p>The tabs are meant to be filled out starting with this tab and moving to the right.  The information provided in this tab is the basis for several automatic calculations that are performed in the subsequent tabs, especially the <i>Levels and Channels</i> tab. If changes are later made to values in this tab after subsequent information has been filled out in other tabs, those changes will be applied automatically, possibly resulting in significant changes in the other tabs.</p></ul>");

const std::vector<QString> LevelsTab::infoText =
  setInfoStrings("<ul><li><p>This tab is used to create the different levels of the compound nucleus considered in the calculation. Levels can be created by clicking the &#43; button on the lower left corner. A level can be deleted by highlighting the level and then clicking the &#45; button. Levels may be above or below the particle separation energies. If a level is below a particle separation energy, the program automatically recognizes this, changing the label from a partial width to an ANC for the appropriate channels.</p></ul>"
"<ul><li><p>The energetically and momentum allowed reaction channels are determined automatically by the code for each level based on the information provided in the <i>Particle Pairs</i> tab and the spin-party and energy of the entered levels. If a change is made to values in the <i>Particle Pairs</i> tab, these changes are automatically applied to the levels calculated in this tab.</p></ul>"
"<ul><li><p>Levels can be included (excluded) from the calculation by checking (unchecking) the box on the far left for each level. A level can have its energy fixed by checking the box marked under the <b>Fix?</b> column. Partial widths or ANCs can be fixed by checking the <b>Fix?</b> box under the Selected Channels window.</p></ul>"
"<ul><li><p>&gamma;-ray decays are limited to bound states.</p></ul>"
"<ul><li><p>For &gamma;-ray calculations, the user should define a level corresponding to the each &gamma;-ray particle pair defined in the <i>Particle Pairs</i> tab. This allows the user to set the value of the ANCs and the &gamma;-ray decay widths for each of these bound states. The energies of these levels need to be the same as the excitation energies of the &gamma;-ray particle pairs. </p></ul>"
"<ul><li><p>Relative interferences for channels can be specified by changing the sign of the corresponding partial width or ANC.</p></ul>"
"<ul><li><p>As one of the general features of <i>R</i>-matrix theory, the number of levels must be truncated both in number and in spin parity to some finite amount. One result of this is an ambiguity in the number of angular momentum terms that are summed in the hard sphere phase terms that are included in scattering and in the external capture calculations. To define these angular momenta, the code only uses those that are present in the channels of the levels defined in this tab. The user may therefore have to include dummy levels in the calculation. These dummy levels simply need to be created at an arbitrary energy with the spin-parities that are not present in the real levels under consideration. The energy can be fixed and the partial widths may be set to zero. The affects should be investigated for each allowed spin parity combination.</p></ul>"
"<ul><li><p>The user needs to be careful that the maximum orbital momentum is high enough to have at least one channels for each particle pair for each level if it allowed or the code will crash. </p></ul>");

const std::vector<QString> SegmentsTab::infoText =
  setInfoStrings("<ul><li><p>This tab is used to define the different data sets that will be considered in the minimization analysis and to define regions for pure calculations.</p></ul>"
"<ul><li><p>In the upper half of the tab, the data that will be considered when a <b>Calculate Segments With Data</b> option is executed under the <i>Calculate</i> tab are designated by assigning data segments. Data segments are also used as a convenient way to sort the input data so that it may be more easily viewed graphically on the <i>Plot</i> tab. A data set is identified with a reaction by
```

```
assigning an entrance and exit pair using the corresponding numbers assigned to each particle pair in
the <i>Particle Pairs</i> tab. The four allowed data types are angle integrated cross section,
differential cross section, phase shifts, and angle integrated total cross sections. High and Low
values of energy and angle are used to selected data that is inclusive between the ranges that are
defined. For example, to plot an excitation curve for differential cross section data, a range of
energy can be entered and a single angle can be specified by giving the same angle in the Low Angle
and High Angle boxes. On execution, the program then looks in the specified data file and pulls only
those data points that meet the specified ranges.</p></ul>"
"<ul><li><p>In the lower half the tab segments can be made that are used when the <b>Calculate
Segments Without Data</b> option is executed under the <i>Calculate</i> tab. These segments define
regions in energy or angle where a pure calculation is made. The energy and angle inputs now represent
the upper and lower energy that will be calculated. Additional energy and angle step sizes must also
be defined in order to specify the spacing of the calculation. These segments may also be used to
extract angular distribution coefficents.</p></ul>"
"<ul><li><p>Like the particle pairs under the <i>Particle Pairs</i> tab, each segment is also assigned
a numerical value. These values can be referenced in the <i>Experimental Effects</i> tab.</p></ul>"
"<ul><li><p>Data files should be created as text files with four columns of space or tab delimitation.
The four columns are ordered from left to right as energy, angle, cross section, cross section
uncertainty. The frame of reference for data files is the laboratory system in forward kinematics
(i.e. light particle projectile, heavy particle target).</p></ul>"
"<ul><li><p>Experimental uncertainties are often categorized into systematic and statistical. The
statistical uncertainties are different for each data point and these are the uncertainties assumed to
be provided in the data file. Systematic uncertainties often apply to a data set as a whole. The
percent systematic uncertainty for a data segment can be specified when creating the segment and the
<b>Vary Norm?</b> box should be checked. For convenience a normalization can be applied to the cross
section. The percent uncertainty is then taken relative to this normalization. The user needs to be
careful in how they define these segments when a rigorous statistical analysis is being performed in
order to avoid double counting of uncertainties. </p></ul>");

const std::vector<QString> TargetIntTab::infoText =
  setInfoStrings("<ul><li><p>This tab is used to apply experimental effect corrections to input data.
Often experimentally reported cross sections still retain some effects due to beam energy loss in
targets, energy resolution, and/or geometric effects of the setup.</p></ul>"
"<ul><li><p>The experimental effect corrections are only of the most basic form. It is assumed that
the user may need to modify the code for their particular analysis.</p></ul>"
"<ul><li><p>It is important to note that there are issues with using both the target convolution and
target integration routines at the same time. The user should be very careful if this is attempted.
</p></ul>"
"<ul><li><p>Note that the experimental effects segments apply to both the data and calculation
segments at the same time. The user may need to remember to select or deselect the experimental effect
segments depending on the kind of calculation that is to be performed.</p></ul>"
"<ul><li><p>Modeling of experimental effects including Gaussian energy convolution, energy loss using
stopping cross section curves, and geometrical attenuation coefficients are included. The parameters
characterizing these corrections can be specified in an experimental effect and that experimental
effect can then be associated with a segment created in the <i>Segments</i> tab using the segment
numbers. When creating an experimental effect, segment numbers can be entered into the box labeled
<b>Segments List</b>. These numbers can be comma delimited or a range can be specified by putting the
lower segment number followed by a dash and then the upper segment number.</p></ul>"
"<ul><li><p>Examples: ``5, 8, 13” and ``4-12” and ``3,6,7-14” are all valid. </p></ul>");

const std::vector<QString> RunTab::infoText =
  setInfoStrings("<ul><li><p>This tab controls the execution of the code. The code offers several
modes of operation including <b>Calculate Segments From Data</b>, <b>Fit Segments From Data</b>,
<b>Calculate Segments Without Data</b>, <b>Perform MINOS Error Analysis</b>, and <b>Calculate Reaction
Rate</b> that can be selected from the drop down menu labeled <b>Calculation Type</b>.</p></ul>"
"<ul><li><p>When a new calculation is first performed, the starting parameter values must be taken
from those entered using the proceeding tabs. Therefore, for a new calculation the Create New
Parameters option must always be selected. If external capture calculations are necessary, the Create
new Integrals File option must also be selected. For initial calculations the <b>Calculate Segments
From Data</b> option should be selected. This will only make a calculation based on the input
parameters that are initially given, no fitting will be performed. This option is useful for
preliminary testing and to adjust initial parameters by hand in preparation for a fit.</p></ul>"
"<ul><li><p>The integrations necessary for the external capture calculations can be quite time
consuming. For this reason their results are stored in a file called <code>ECint.dat</code>. This file
needs to only be created once and then subsequent calculations can be made using the results by
selecting the Use option and then selecting the <code>ECint.dat</code> file from the user’s specified
output directory. However, if the input data, the channel radius or experimental effects are modified
or a level of a new spin-parity is added or deleted, this file needs to be recalculated.</p></ul>"
"<ul><li><p>After a fit has been performed using the <b>Fit Segments From Data</b> mode, the final
<i>R</i>-matrix parameters are stored in the files <code>param.sav</code> and
<code>normalizations.out</code>.  By selecting the <code>param.sav</code> file from the working output
directory the fit can then be reproduced. If the <code>param.sav</code> file is selected the
<code>normalizations.out</code> file is automatically selected from the same directory. In this way
the results from the fit can be used to make extrapolations using the <b>Calculate Segments Without
Data</b> option or calculate the resulting reaction rate using <b>Calculate Reaction
Rate</b>.</p></ul>"
"<ul><li><p>The observable parameters resulting from the fit are output in the file
<code>parameters.out</code>. If the user wishes to use these parameters as new starting values for a
fit they must be copied over by hand into the <i>Levels and Channels</i> tab.</p></ul>"
"<ul><li><p>Results of the MINOS uncertainty analysis are output in the files
<code>param.errors</code> and <code>covariance_matrix.out</code>.</p></ul>"
"<ul><li><p>Results of the reaction rate calculation are output in <code>reactionrates.out</code>. The
reaction rate calculation uses GSL adaptive step size integration. For calculations that involve
external capture, this means that the code must also calculate external capture integrals for each of
the energy points ``on the fly” since these points are not determined in advance. This may result in
very long computation times for these reaction rates. Instead, the user may wish to create an array of
finely energy spaced data points using the Segments Without Data section of the <i>Segments</i> tab
```

```
and perform the numerical integration on their own. It should also be noted that numerical integration
of narrow resonances may not be performed properly even with adaptive step size integration. For this
reason, reaction rate calculations using the code should only be performed for broad structures.
Narrow resonance contributions should be added separately using the usual narrow resonance reaction
rate approximation. </p></ul>");

const std::vector<QString> PlotTab::infoText =
  setInfoStrings("<ul><li><p>This tab is used to plot cross sections (or corresponding S-factors)
using the segments defined in the <i>Segments</i> tab. A segment can be plotted by clicking on the
short segment label from the list on the left and then clicking the <b>Draw</b> button in the lower
left corner. The segment is unselected by clicking on it a second time. Multiple segments may be
plotted at once by clicking on each a single time and then clicking the <b>Draw</b> button. A segment
is selected when the background color changes.</p></ul>");
```

8.109 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/LevelsModel.cpp File Reference

```
#include ¨LevelsModel.h¨
```

8.110 LevelsModel.cpp

Go to the documentation of this file.

```
#include "LevelsModel.h"

LevelsModel::LevelsModel(QObject *parent) : QAbstractTableModel(parent) {
}

int LevelsModel::rowCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return levelsList.size();
}

int LevelsModel::columnCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return LevelsData::SIZE;
}

QVariant LevelsModel::data(const QModelIndex &index, int role) const {
  if(!index.isValid()) return QVariant();

  if(index.row() >= levelsList.size() || index.row() < 0)  return QVariant();

  if (role == Qt::DisplayRole) {
    LevelsData level = levelsList.at(index.row());
    if(index.column() == 2) return getSpinLabel(level);
    else if(index.column() == 3) return level.piValue;
    else if(index.column() == 4) return level.energy;
  } else if (role == Qt::EditRole) {
    LevelsData level = levelsList.at(index.row());
    if(index.column() == 2) return level.jValue;
    else if(index.column() == 3) return level.piValue;
    else if(index.column() == 4) return level.energy;
  } else if(role==Qt::TextAlignmentRole) return Qt::AlignCenter;
  else if (role==Qt::CheckStateRole && (index.column()==0||index.column()==1)) {
    LevelsData level = levelsList.at(index.row());
    if(index.column()==0) {
      if(level.isActive==1) return Qt::Checked;
      else return Qt::Unchecked;
    } else {
      if(level.isFixed==1) return Qt::Checked;
      else return Qt::Unchecked;
    }
  }
  return QVariant();
}

QVariant LevelsModel::headerData(int section, Qt::Orientation orientation, int role) const {
  if(role!= Qt::DisplayRole) return QVariant();
  if(orientation == Qt::Horizontal) {
    switch(section) {
    case 0:
      return tr("Include?");
```

```
    case 1:
      return tr("Fix?");
    case 2:
      return tr("Level\nSpin");
    case 3:
      return tr("Parity");
    case 4:
      return tr("Energy\n[MeV]");
    default:
      return QVariant();
    }
  } else if(orientation == Qt::Vertical) {
    return section+1;
  }
  return QVariant();
}

bool LevelsModel::setData(const QModelIndex &index, const QVariant &value, int role) {
  if (index.isValid()) {
    if(role == Qt::EditRole ) {
      int row = index.row();
      LevelsData tempData = levelsList.value(row);
      if(index.column() == 0) tempData.isActive=value.toInt();
      else if(index.column() == 1) tempData.isFixed=value.toInt();
      else if(index.column() == 2) tempData.jValue=value.toDouble();
      else if(index.column() == 3) tempData.piValue=value.toInt();
      else if(index.column() == 4) tempData.energy=value.toDouble();
      else return false;

      levelsList.replace(row,tempData);
      emit(dataChanged(index,index));
      return true;
    } else if(role== Qt::CheckStateRole) {
      int row = index.row();
      LevelsData tempData = levelsList.value(row);
      if(index.column()==0) {
    if(value==Qt::Checked) tempData.isActive=1;
    else tempData.isActive=0;
      } else if(index.column()==1) {
    if(value==Qt::Checked) tempData.isFixed=1;
    else tempData.isFixed=0;
      } else return false;

      levelsList.replace(row,tempData);
      emit(dataChanged(index,index));
      return true;
    }
  }
  return false;
}

bool LevelsModel::insertRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginInsertRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows; row++) {
      LevelsData tempData={1,0,0.0,-1,0.0};
      levelsList.insert(position,tempData);
    }
    endInsertRows();
  }
  return true;
}

bool LevelsModel::removeRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginRemoveRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows;++row) {
      levelsList.removeAt(position);
    }
    endRemoveRows();
  }
  return true;
}

Qt::ItemFlags LevelsModel::flags(const QModelIndex &index) const {
  if (!index.isValid()) return Qt::ItemIsEnabled;
  if(index.column()==0 || index.column() ==1) return QAbstractTableModel::flags(index) |
Qt::ItemIsUserCheckable;
  return QAbstractTableModel::flags(index);
}

int LevelsModel::isLevel(const LevelsData &level) const {
  int foundLevel=-1;
  for(int i=0;i<levelsList.size();i++) {
    LevelsData tempLevel=levelsList.value(i);
```

```
    if(tempLevel.jValue==level.jValue&&
       tempLevel.piValue==level.piValue&&
       tempLevel.energy==level.energy) {
      foundLevel=i;
      break;
    }
  }
  return foundLevel;
}

QString LevelsModel::getSpinLabel(const LevelsData &level) const {
  QString tempSpin;
  if(((int)(level.jValue*2))%2!=0&&level.jValue!=0.)
tempSpin=QString("%1/2").arg((int)(level.jValue*2));
  else tempSpin=QString("%1").arg(level.jValue);
  if(level.piValue==-1) return QString("%1-").arg(tempSpin);
  else return QString("%1+").arg(tempSpin);
}
```

8.111 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/LevelsTab.cpp File Reference

#include <QHeaderView>
#include ¨LevelsTab.h¨
#include ¨LevelsHeaderView.h¨
#include ¨AddLevelDialog.h¨
#include ¨RichTextDelegate.h¨
#include ¨InfoDialog.h¨
#include <iostream>

8.112 LevelsTab.cpp

Go to the documentation of this file.

```
#include <QHeaderView>

#include "LevelsTab.h"
#include "LevelsHeaderView.h"
#include "AddLevelDialog.h"
#include "RichTextDelegate.h"
#include "InfoDialog.h"
#include <iostream>

LevelsTab::LevelsTab(QWidget *parent) : QWidget(parent) {
  levelsModel=new LevelsModel(this);
  levelsModelProxy = new QSortFilterProxyModel(this);
  levelsModelProxy->setSourceModel(levelsModel);
  levelsModelProxy->setDynamicSortFilter(true);
  levelsView=new QTableView;
  levelsView->setHorizontalHeader(new LevelsHeaderView(Qt::Horizontal, levelsView));
  levelsView->setModel(levelsModelProxy);
  levelsView->horizontalHeader()->setSortIndicator(4,Qt::AscendingOrder);
  levelsView->setSortingEnabled(true);
  levelsView->verticalHeader()->hide();
  levelsView->horizontalHeader()->setHighlightSections(false);
  levelsView->setColumnWidth(0,60);
  levelsView->setColumnWidth(1,40);
  levelsView->horizontalHeader()->setSectionResizeMode(0,QHeaderView::Fixed);
  levelsView->horizontalHeader()->setSectionResizeMode(1,QHeaderView::Fixed);
  levelsView->horizontalHeader()->setSectionResizeMode(2,QHeaderView::Stretch);
  levelsView->horizontalHeader()->setSectionResizeMode(4,QHeaderView::Stretch);
  levelsView->setColumnHidden(3,true);
  levelsView->setSelectionBehavior(QAbstractItemView::SelectRows);
  levelsView->setSelectionMode(QAbstractItemView::SingleSelection);
  levelsView->setEditTriggers(QAbstractItemView::NoEditTriggers);
  levelsView->setShowGrid(false);

connect(levelsView->selectionModel(),SIGNAL(selectionChanged(QItemSelection,QItemSelection)),this,SLOT(updateButtons(QIt

connect(levelsView->selectionModel(),SIGNAL(selectionChanged(QItemSelection,QItemSelection)),this,SLOT(updateFilter(QIte
```

```
    connect(levelsView,SIGNAL(doubleClicked(QModelIndex)),this,SLOT(editLevel()));

    maxLSpin = new QSpinBox;
    maxLSpin->setMinimum(0);
    maxLSpin->setMaximum(10);
    maxLSpin->setSingleStep(1);
    maxLSpin->setValue(2);
    QLabel *maxLLabel = new QLabel(tr("Maximum Orbital Momentum"));
    maxMultSpin = new QSpinBox;
    maxMultSpin->setMinimum(1);
    maxMultSpin->setMaximum(10);
    maxMultSpin->setSingleStep(1);
    maxMultSpin->setValue(2);
    QLabel *maxMultLabel = new QLabel(tr("Maximum Gamma Multipolarity"));
    maxNumMultSpin = new QSpinBox;
    maxNumMultSpin->setMinimum(1);
    maxNumMultSpin->setMaximum(10);
    maxNumMultSpin->setSingleStep(1);
    maxNumMultSpin->setValue(2);
    QLabel *maxNumMultLabel = new QLabel(tr("Maximum Gamma Multipolarities\nPer Decay"));
    connect(maxLSpin,SIGNAL(valueChanged(int)),this,SLOT(updateChannelsPairAddedEdited()));
    connect(maxMultSpin,SIGNAL(valueChanged(int)),this,SLOT(updateChannelsPairAddedEdited()));
    connect(maxNumMultSpin,SIGNAL(valueChanged(int)),this,SLOT(updateChannelsPairAddedEdited()));

    channelsModel = new ChannelsModel(this);
    proxyModel = new QSortFilterProxyModel(this);
    proxyModel->setSourceModel(channelsModel);
    proxyModel->setDynamicSortFilter(true);
    proxyModel->setFilterKeyColumn(1);
    proxyModel->setFilterRegExp("-1");
    proxyModel->sort(1,Qt::AscendingOrder);
    channelsView = new QTableView;
    channelsView->setModel(proxyModel);
    channelsView->verticalHeader()->hide();
    channelsView->horizontalHeader()->setHighlightSections(false);
    channelsView->setColumnWidth(0,40);
    channelsView->setColumnWidth(2,160);
    channelsView->horizontalHeader()->setSectionResizeMode(0,QHeaderView::Fixed);
    channelsView->horizontalHeader()->setSectionResizeMode(2,QHeaderView::Fixed);
    channelsView->horizontalHeader()->setSectionResizeMode(3,QHeaderView::Stretch);
    channelsView->horizontalHeader()->setSectionResizeMode(4,QHeaderView::Stretch);
    channelsView->setColumnHidden(1,true);
    channelsView->setColumnHidden(5,true);
    channelsView->setColumnHidden(6,true);
    channelsView->setItemDelegateForColumn(2,new RichTextDelegate());
    channelsView->setItemDelegateForColumn(3,new RichTextDelegate());
    channelsView->setItemDelegateForColumn(4,new RichTextDelegate());
    channelsView->setSelectionBehavior(QAbstractItemView::SelectRows);
    channelsView->setSelectionMode(QAbstractItemView::SingleSelection);
    channelsView->setEditTriggers(QAbstractItemView::NoEditTriggers);
    channelsView->setShowGrid(false);
  connect(channelsView->selectionModel(),SIGNAL(selectionChanged(QItemSelection,QItemSelection)),this,SLOT(updateDetails(

    channelDetails=new ChannelDetails(this);
    channelDetails->hide();
    connect(channelDetails->reducedWidthText,SIGNAL(textEdited(const QString&)),this,SLOT(updateReducedWidth(const QString&)));

    addLevelButton = new QPushButton(tr("+"));
    addLevelButton->setMaximumSize(28,28);
    connect(addLevelButton,SIGNAL(clicked()),this,SLOT(addLevel()));
    removeLevelButton = new QPushButton(tr("-"));
    removeLevelButton->setMaximumSize(28,28);
    removeLevelButton->setEnabled(false);
    connect(removeLevelButton,SIGNAL(clicked()),this,SLOT(removeLevel()));

    /*
    mapper = new QSignalMapper(this);
    connect(mapper,SIGNAL(mapped(int)),this,SLOT(showInfo(int)));

    infoButton[0] = new QPushButton(this);
    infoButton[0]->setMaximumSize(28,28);
    infoButton[0]->setIcon(style()->standardIcon(QStyle::SP_MessageBoxInformation));
    mapper->setMapping(infoButton[0],1);
    connect(infoButton[0],SIGNAL(clicked()),mapper,SLOT(map()));
    */

    QGridLayout *buttonBox = new QGridLayout;
    buttonBox->addWidget(addLevelButton,0,0);
    buttonBox->addWidget(removeLevelButton,0,1);
    buttonBox->addItem(new QSpacerItem(28,28),0,2);
    //buttonBox->addWidget(infoButton[0],0,3);
    buttonBox->setColumnStretch(0,0);
    buttonBox->setColumnStretch(1,0);
    buttonBox->setColumnStretch(2,1);
    buttonBox->setColumnStretch(3,0);
```

```
#ifdef MACX_SPACING
  buttonBox->setHorizontalSpacing(11);
#else
  buttonBox->setHorizontalSpacing(0);
#endif

  QGroupBox *levelsBox=new QGroupBox(tr("Compound Nucleus Levels"));
  QGridLayout *levelsLayout=new QGridLayout;
  levelsLayout->setContentsMargins(6,6,6,6);
  levelsLayout->addWidget(levelsView,0,0);
  levelsLayout->addLayout(buttonBox,1,0);
#ifdef MACX_SPACING
  levelsLayout->setVerticalSpacing(0);
#endif
  levelsBox->setLayout(levelsLayout);

  QGroupBox *configBox = new QGroupBox(tr("Channel Configuration"));
  QGridLayout *configLayout = new QGridLayout;
  configLayout->addWidget(maxLSpin,0,0);
  configLayout->addWidget(maxLLabel,0,1);
  configLayout->addWidget(maxMultSpin,1,0);
  configLayout->addWidget(maxMultLabel,1,1);
  configLayout->addWidget(maxNumMultSpin,2,0);
  configLayout->addWidget(maxNumMultLabel,2,1);
  configLayout->setColumnStretch(0,0);
  configLayout->setColumnStretch(1,1);
  configBox->setLayout(configLayout);

  QGroupBox *channelsBox=new QGroupBox(tr("Channels In Selected Level"));
  QGridLayout *channelsLayout=new QGridLayout;
  channelsLayout->setContentsMargins(6,6,6,6);
  channelsLayout->addWidget(channelsView,0,0);
#ifdef MACX_SPACING
  channelsLayout->addItem(new QSpacerItem(40,40),1,0);
  channelsLayout->setVerticalSpacing(0);
#else
  channelsLayout->addItem(new QSpacerItem(34,34),1,0);
#endif
  channelsBox->setLayout(channelsLayout);

  QGridLayout *detailsBox = new QGridLayout;
  QLabel *detailsLabel = new QLabel(tr("Channel Details (select from list to view):"));
  detailsLabel->setAlignment(Qt::AlignHCenter);
  detailsBox->addWidget(detailsLabel,0,0);
  detailsBox->addWidget(channelDetails,1,0);
  detailsBox->setRowStretch(0,0);
  detailsBox->setRowStretch(1,1);

  QGridLayout *mainLayout = new QGridLayout;
  mainLayout->addWidget(levelsBox,0,0,2,1);
  mainLayout->addWidget(channelsBox,0,1,2,1);
  mainLayout->addWidget(configBox,0,2,1,1);
  mainLayout->addLayout(detailsBox,1,2,1,1);
  mainLayout->setColumnStretch(0,1);
  mainLayout->setColumnStretch(1,1);
  mainLayout->setColumnStretch(2,1);

  setLayout(mainLayout);
}

void LevelsTab::setPairsModel(PairsModel *model) {
  pairsModel=model;
  channelsModel->setPairsModel(model);
}

void LevelsTab::addLevel() {
  AddLevelDialog aDialog;
  if(aDialog.exec()) {
    LevelsData newLevel;
    newLevel.isActive=1;
    newLevel.isFixed=0;
    newLevel.jValue=(aDialog.jValueText->text()).toDouble();
    if(aDialog.piValueCombo->currentIndex()==0) newLevel.piValue=-1;
    else newLevel.piValue=1;
    newLevel.energy=(aDialog.energyText->text()).toDouble();
    addLevel(newLevel,false);
  }
}

void LevelsTab::addLevel(LevelsData level, bool fromFile) {
  QList<LevelsData> levels = levelsModel->getLevels();
  if(levelsModel->isLevel(level)==-1) {
    levelsModel->insertRows(levels.size(),1,QModelIndex());
    QModelIndex index = levelsModel->index(levels.size(),0,QModelIndex());
    levelsModel->setData(index,level.isActive,Qt::EditRole);
    index = levelsModel->index(levels.size(),1,QModelIndex());
    levelsModel->setData(index,level.isFixed,Qt::EditRole);
```

```
    index = levelsModel->index(levels.size(),2,QModelIndex());
    levelsModel->setData(index,level.jValue,Qt::EditRole);
    index = levelsModel->index(levels.size(),3,QModelIndex());
    levelsModel->setData(index,level.piValue,Qt::EditRole);
    index = levelsModel->index(levels.size(),4,QModelIndex());
    levelsModel->setData(index,level.energy,Qt::EditRole);
    levelsView->resizeRowsToContents();
    if(!fromFile) updateChannelsLevelAdded(levels.size());
  } else {
    QMessageBox::information(this,tr("Duplicate Level"),tr("This level already exists."));
  }
}

void LevelsTab::removeLevel() {
  QItemSelectionModel *selectionModel = levelsView->selectionModel();
  QModelIndexList indexes = selectionModel->selectedRows();
  QModelIndex index=levelsModelProxy->mapToSource(indexes[0]);

  levelsModel->removeRows(index.row(),1,QModelIndex());
  updateChannelsLevelDeleted(index.row());
  selectionModel->clearSelection();
}

void LevelsTab::editLevel() {
  QItemSelectionModel *selectionModel = levelsView->selectionModel();
  QModelIndexList indexes = selectionModel->selectedRows();
  QModelIndex index=levelsModelProxy->mapToSource(indexes[0]);

  QModelIndex i=levelsModel->index(index.row(),2,QModelIndex());
  QVariant var=levelsModel->data(i,Qt::EditRole);
  QString jValue=var.toString();
  i=levelsModel->index(index.row(),3,QModelIndex());
  var=levelsModel->data(i,Qt::EditRole);
  int piValue=var.toInt();
  i=levelsModel->index(index.row(),4,QModelIndex());
  var=levelsModel->data(i,Qt::EditRole);
  QString energy=var.toString();

  AddLevelDialog aDialog;
  aDialog.setWindowTitle(tr("Edit a Level"));
  aDialog.jValueText->setText(jValue);
  if(piValue==-1) aDialog.piValueCombo->setCurrentIndex(0);
  else aDialog.piValueCombo->setCurrentIndex(1);
  aDialog.energyText->setText(energy);

  if(aDialog.exec()) {
    QString newJValue=aDialog.jValueText->text();
    if(newJValue!=jValue) {
      i=levelsModel->index(index.row(),2,QModelIndex());
      levelsModel->setData(i,newJValue,Qt::EditRole);
    }
    int newPiValue;
    if(aDialog.piValueCombo->currentIndex()==0) newPiValue=-1;
    else newPiValue =1;
    if(newPiValue!=piValue) {
      i=levelsModel->index(index.row(),3,QModelIndex());
      levelsModel->setData(i,newPiValue,Qt::EditRole);
    }
    QString newEnergy=aDialog.energyText->text();
    if(newEnergy!=energy) {
      i=levelsModel->index(index.row(),4,QModelIndex());
      levelsModel->setData(i,newEnergy,Qt::EditRole);
    }
    updateChannelsLevelEdited(index.row());
  }
}

void LevelsTab::updateButtons(const QItemSelection &selection) {
  QModelIndexList indexes=selection.indexes();

  if(indexes.isEmpty()) {
    removeLevelButton->setEnabled(false);
  } else {
    removeLevelButton->setEnabled(true);
  }
}

void LevelsTab::updateChannelsLevelAdded(int levelIndex) {
  QList<ChannelsData> newChannels = calculateChannels(levelIndex);
  channelsModel->insertRows(0,newChannels.size(),QModelIndex());
  for(int i=0;i<newChannels.size();i++) {
    QModelIndex index = channelsModel->index(i,0,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).isFixed,Qt::EditRole);
    index = channelsModel->index(i,1,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).levelIndex,Qt::EditRole);
    index = channelsModel->index(i,2,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).pairIndex,Qt::EditRole);
```

```
    index = channelsModel->index(i,3,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).sValue,Qt::EditRole);
    index = channelsModel->index(i,4,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).lValue,Qt::EditRole);
    index = channelsModel->index(i,5,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).radType,Qt::EditRole);
    index = channelsModel->index(i,6,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).reducedWidth,Qt::EditRole);
  }
  channelsView->resizeRowsToContents();
}

void LevelsTab::updateChannelsLevelDeleted(int levelIndex) {
  QList<ChannelsData> channels=channelsModel->getChannels();
  int deleted=0;
  for(int i=0;i<channels.size();i++) {
    if(channels.at(i).levelIndex==levelIndex) {
      channelsModel->removeRows(i-deleted,1,QModelIndex());
      deleted++;
    }
  }
  channels=channelsModel->getChannels();
  for(int i=0;i<channels.size();i++) {
    if(channels.at(i).levelIndex>levelIndex) {
      QModelIndex index=channelsModel->index(i,1,QModelIndex());
      channelsModel->setData(index,channels.at(i).levelIndex-1,Qt::EditRole);
    }
  }
}

void LevelsTab::updateChannelsLevelEdited(int levelIndex) {
  QList<ChannelsData> channels=channelsModel->getChannels();
  int deleted=0;
  for(int i=0;i<channels.size();i++) {
    if(channels.at(i).levelIndex==levelIndex) {
      channelsModel->removeRows(i-deleted,1,QModelIndex());
      deleted++;
    }
  }
  QList<ChannelsData> newChannels = calculateChannels(levelIndex);
  channelsModel->insertRows(0,newChannels.size(),QModelIndex());
  for(int i=0;i<newChannels.size();i++) {
    QModelIndex index = channelsModel->index(i,1,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).levelIndex,Qt::EditRole);
    index = channelsModel->index(i,2,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).pairIndex,Qt::EditRole);
    index = channelsModel->index(i,3,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).sValue,Qt::EditRole);
    index = channelsModel->index(i,4,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).lValue,Qt::EditRole);
    index = channelsModel->index(i,5,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).radType,Qt::EditRole);
    for(int ii=0;ii<channels.size();ii++) {
      if(channels.at(ii).levelIndex==newChannels.at(i).levelIndex&&
     channels.at(ii).pairIndex==newChannels.at(i).pairIndex&&
     channels.at(ii).sValue==newChannels.at(i).sValue&&
     channels.at(ii).lValue==newChannels.at(i).lValue&&
     channels.at(ii).radType==newChannels.at(i).radType) {
    newChannels[i].isFixed=channels.at(ii).isFixed;
    newChannels[i].reducedWidth=channels.at(ii).reducedWidth;
    break;
      }
    }
    index = channelsModel->index(i,6,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).reducedWidth,Qt::EditRole);
    index = channelsModel->index(i,0,QModelIndex());
    channelsModel->setData(index,newChannels.at(i).isFixed,Qt::EditRole);
  }
  channelsView->resizeRowsToContents();
}

QList<ChannelsData> LevelsTab::calculateChannels(int levelIndex) {
  QList<ChannelsData> channels;
  QList<PairsData> pairs = pairsModel->getPairs();
  LevelsData level = (levelsModel->getLevels()).at(levelIndex);

  int maxL=maxLSpin->value();
  int maxMult=maxMultSpin->value();
  int maxNumMult=maxNumMultSpin->value();

  for(int i=0;i<pairs.size();i++) {
    PairsData pair=pairs.at(i);
    if(pair.pairType==0) {
      for(double s=fabs(pair.heavyJ-pair.lightJ);s<=pair.heavyJ+pair.lightJ;s+=1.0) {
    for(double l=fabs(s-level.jValue);l<=s+level.jValue;l+=1.0) {
      if(int(l*2.0)%2==0&&pair.lightPi*pair.heavyPi*pow(-1,int(l))==level.piValue&&int(l)<=maxL) {
        ChannelsData channel={0,levelIndex,i,s,int(l),'P',0.0};
```

```
        channels.push_back(channel);
      }
    }
      }
    } else if(pair.pairType==20) {
      if(fabs(pair.heavyJ-level.jValue)==0.&&pair.heavyPi==level.piValue) {
    ChannelsData gtChannel = {0,levelIndex,i,0.,1,'G',0.0};
    ChannelsData fChannel = {0,levelIndex,i,0.,0,'F',0.0};
    channels.push_back(gtChannel);
    channels.push_back(fChannel);
      } else if(fabs(pair.heavyJ-level.jValue)==1.&&pair.heavyPi==level.piValue) {
    ChannelsData gtChannel = {0,levelIndex,i,1.,1,'G',0.0};
    channels.push_back(gtChannel);
      }
    } else {
      int numMult=1;
      for(int l=1;l<=maxMult;l++) {
    if(fabs(l-pair.heavyJ)<=level.jValue&&level.jValue<=l+pair.heavyJ&&numMult<=maxNumMult) {
      QChar radType;
      int parityChange=pair.heavyPi*level.piValue;
      if(l%2!=0) {
        if(parityChange==-1) radType='E';
        else radType='M';
      } else {
        if(parityChange==-1) radType='M';
        else radType='E';
      }
      ChannelsData channel={0,levelIndex,i,pair.heavyJ,l,radType,0.0};
      channels.push_back(channel);
      numMult++;
    }
      }
    }
  }
  return channels;
}

void LevelsTab::updateFilter(const QItemSelection &selection) {
  QModelIndexList indexes = selection.indexes();
  if(indexes.isEmpty())
    proxyModel->setFilterRegExp("-1");
  else {
    QModelIndex index = levelsModelProxy->mapToSource(indexes.at(0));
    int row=index.row();
    proxyModel->setFilterRegExp(QString("\\b%1\\b").arg(row));
  }
  channelsView->resizeRowsToContents();
}

void LevelsTab::updateChannelsPairAddedEdited() {
  QList<LevelsData> levels=levelsModel->getLevels();
  QList<ChannelsData> channels=channelsModel->getChannels();
  channelsModel->removeRows(0,channels.size(),QModelIndex());
  for(int levelIndex=0;levelIndex<levels.size();levelIndex++) {
    QList<ChannelsData> newChannels=calculateChannels(levelIndex);
    channelsModel->insertRows(0,newChannels.size(),QModelIndex());
    for(int i=0;i<newChannels.size();i++) {
      QModelIndex index = channelsModel->index(i,1,QModelIndex());
      channelsModel->setData(index,newChannels.at(i).levelIndex,Qt::EditRole);
      index = channelsModel->index(i,2,QModelIndex());
      channelsModel->setData(index,newChannels.at(i).pairIndex,Qt::EditRole);
      index = channelsModel->index(i,3,QModelIndex());
      channelsModel->setData(index,newChannels.at(i).sValue,Qt::EditRole);
      index = channelsModel->index(i,4,QModelIndex());
      channelsModel->setData(index,newChannels.at(i).lValue,Qt::EditRole);
      index = channelsModel->index(i,5,QModelIndex());
      channelsModel->setData(index,newChannels.at(i).radType,Qt::EditRole);
      for(int ii=0;ii<channels.size();ii++) {
    if(channels.at(ii).levelIndex==newChannels.at(i).levelIndex&&
       channels.at(ii).pairIndex==newChannels.at(i).pairIndex&&
       channels.at(ii).sValue==newChannels.at(i).sValue&&
       channels.at(ii).lValue==newChannels.at(i).lValue&&
       channels.at(ii).radType==newChannels.at(i).radType) {
      newChannels[i].reducedWidth=channels.at(ii).reducedWidth;
      newChannels[i].isFixed=channels.at(ii).isFixed;
      break;
    }
      }
      index = channelsModel->index(i,6,QModelIndex());
      channelsModel->setData(index,newChannels.at(i).reducedWidth,Qt::EditRole);
      index = channelsModel->index(i,0,QModelIndex());
      channelsModel->setData(index,newChannels.at(i).isFixed,Qt::EditRole);
    }
  }
  channelsView->resizeRowsToContents();
}

```

```
void LevelsTab::updateChannelsPairRemoved(int pairIndex) {
  QList<ChannelsData> channels=channelsModel->getChannels();
  int deleted=0;
  for(int i=0;i<channels.size();i++) {
    if(channels.at(i).pairIndex==pairIndex) {
      channelsModel->removeRows(i-deleted,1,QModelIndex());
      deleted++;
    }
  }
  channels=channelsModel->getChannels();
  for(int i=0;i<channels.size();i++) {
    if(channels.at(i).pairIndex>pairIndex)  {
      QModelIndex index=channelsModel->index(i,2,QModelIndex());
      channelsModel->setData(index,channels.at(i).pairIndex-1,Qt::EditRole);
    }
  }
  channelsView->resizeRowsToContents();
}

void LevelsTab::updateDetails(const QItemSelection &selection) {
  QModelIndexList indexes=selection.indexes();

  if(indexes.isEmpty()) {
    channelDetails->hide();
  } else {
    QModelIndex index=proxyModel->mapToSource(indexes.at(0));

    QModelIndex i=channelsModel->index(index.row(),1,QModelIndex());
    QVariant var=channelsModel->data(i,Qt::EditRole);
    int levelIndex = var.toInt();
    i=channelsModel->index(index.row(),2,QModelIndex());
    var=channelsModel->data(i,Qt::EditRole);
    int pairIndex = var.toInt();

    ChannelsData channel=channelsModel->getChannels().at(index.row());
    LevelsData level=levelsModel->getLevels().at(levelIndex);
    PairsData pair=pairsModel->getPairs().at(pairIndex);

    QString details="";
    QTextStream stm(&details,QIODevice::Append);
    stm « QString("%1 MeV level with spin %2\n   transitioning via pair key #%3").arg(level.energy).arg(levelsModel->getSpinLabel(level)).arg(pairIndex+1)
    « endl;
    if(channel.radType=='P') {
      stm « QString("Channel configuration is\n   s = %1, l = %2").arg(channelsModel->getSpinLabel(channel)).arg(channel.lValue)
      « endl « endl;
      stm « qSetFieldWidth(21) « right « "Light Particle Spin: "
      « qSetFieldWidth(0) « left « QString("%1").arg(pairsModel->getSpinLabel(pair,0)) « endl;
      stm « qSetFieldWidth(21) « right « "Light Particle Z: "
      « qSetFieldWidth(0) « left « QString("%1").arg(pair.lightZ) « endl;
      stm « qSetFieldWidth(21) « right « "Light Particle M: "
      « qSetFieldWidth(0) « left « QString("%1").arg(pair.lightM) « endl;
      stm « qSetFieldWidth(21) « right « "Light Particle G: "
      « qSetFieldWidth(0) « left « QString("%1").arg(pair.lightG) « endl;
    } else if(channel.radType=='G' || channel.radType=='F') {
      if(channel.radType=='G')
    stm « QString("Channel is Gamow-Teller beta decay") « endl « endl;
      else
    stm « QString("Channel is Fermi beta decay") « endl « endl;
      stm « qSetFieldWidth(21) « right « "Fermion Charge: "
      « qSetFieldWidth(0) « left « QString("%1").arg(pair.lightZ) « endl;
    } else {
      stm « QString("Capture gamma is %1%2 radiation").arg(channel.radType).arg(channel.lValue) « endl;
      if(((channel.radType=='E'&&channel.lValue==1)&&
      (pair.ecMultMask & (1«0)))||
     ((channel.radType=='M'&&channel.lValue==1)&&
      (pair.ecMultMask & (1«1)))||
     ((channel.radType=='E'&&channel.lValue==2)&&
      (pair.ecMultMask & (1«2))))
    stm « "Capture is internal and external" « endl;
      else stm « "Capture is internal only" « endl;
      stm « endl;
    }
    stm « qSetFieldWidth(21) « right « "Heavy Particle Spin: "
    « qSetFieldWidth(0) « left «QString("%1").arg(pairsModel->getSpinLabel(pair,1)) « endl;
    stm « qSetFieldWidth(21) « right « "Heavy Particle Z: "
    « qSetFieldWidth(0) « left «QString("%1").arg(pair.heavyZ) « endl;
    stm « qSetFieldWidth(21) « right « "Heavy Particle M: "
    « qSetFieldWidth(0) « left «QString("%1").arg(pair.heavyM) « endl;
    stm « qSetFieldWidth(21) « right « "Heavy Particle G: "
    « qSetFieldWidth(0) « left «QString("%1").arg(pair.heavyG) « endl;
    if(channel.radType!='G'&&channel.radType!='F')
      stm « qSetFieldWidth(21) « right « "Excitation Energy: "
      « qSetFieldWidth(0) « left «QString("%1").arg(pair.excitationEnergy) « endl;
    if(channel.radType!='M'&&channel.radType!='E') {
```

```
      stm « qSetFieldWidth(21) « right « "Separation Energy: "
      « qSetFieldWidth(0) « left «QString("%1").arg(pair.seperationEnergy) « endl;
      stm « qSetFieldWidth(21) « right « "Channel Radius: "
      « qSetFieldWidth(0) « left «QString("%1").arg(pair.channelRadius) « endl;
    }
    stm.flush();
    channelDetails->details->setText(details);
    if(level.energy<(pair.seperationEnergy+pair.excitationEnergy)&&pair.pairType==0)
channelDetails->setNormParam(1);
    else if (pair.pairType==10&&level.energy==pair.excitationEnergy&&level.jValue==pair.heavyJ&&
         level.piValue==pair.heavyPi&&channel.radType=='M'&&channel.lValue==1)
      channelDetails->setNormParam(2);
    else if (pair.pairType==10&&level.energy==pair.excitationEnergy&&level.jValue==pair.heavyJ&&
         level.piValue==pair.heavyPi&&channel.radType=='E'&&channel.lValue==2)
      channelDetails->setNormParam(3);
    else if(pair.pairType==20) channelDetails->setNormParam(4);
    else channelDetails->setNormParam(0);
    channelDetails->reducedWidthText->setText(QString("%1").arg(channel.reducedWidth));
    channelDetails->show();
  }
}

void LevelsTab::updateReducedWidth(const QString &string) {
  QItemSelectionModel *selectionModel=channelsView->selectionModel();
  QModelIndex index=proxyModel->mapToSource(selectionModel->selectedRows().at(0));

  if(index.isValid()) {
    double reducedWidth=string.toDouble();
    QModelIndex i = channelsModel->index(index.row(),6,QModelIndex());
    channelsModel->setData(i,reducedWidth,Qt::EditRole);
  }
}

bool LevelsTab::writeNuclearFile(QTextStream& outStream) {
  QList<PairsData> pairs=pairsModel->getPairs();
  QList<LevelsData> levels=levelsModel->getLevels();
  QList<ChannelsData> channels=channelsModel->getChannels();

  outStream.setFieldAlignment(QTextStream::AlignRight);

  double lowJ=0;
  double highJ=0;
  for(int la=0;la<levels.size();la++) {
    double tempJ=levels.at(la).jValue;
    if(la==0) {
      lowJ=tempJ;
      highJ=tempJ;
    } else if(tempJ<=lowJ) lowJ=tempJ;
    else if(tempJ>=highJ) highJ=tempJ;
  }
  QList<QList<LevelsData> > sortedLevels;
  QList<QList<int> > levelsMap;
  for(double j=lowJ;j<=highJ;j+=0.5) {
    for(int pi=-1;pi<=1;pi+=2) {
      for(int la=0;la<levels.size();la++) {
    if(levels.at(la).jValue==j&&levels.at(la).piValue==pi) {
      if(sortedLevels.size()==0||(sortedLevels.at(sortedLevels.size()-1).at(0).jValue!=j||
                      sortedLevels.at(sortedLevels.size()-1).at(0).piValue!=pi)) {
        QList<LevelsData> tempLevelList;
        sortedLevels.append(tempLevelList);
        QList<int> tempKeyList;
        levelsMap.append(tempKeyList);
      }
      if(sortedLevels.at(sortedLevels.size()-1).size()==0) {
        sortedLevels[sortedLevels.size()-1].append(levels.at(la));
        levelsMap[levelsMap.size()-1].append(la);
      } else for(int mu=0;mu<sortedLevels.at(sortedLevels.size()-1).size();mu++) {
        if(levels.at(la).energy<=sortedLevels.at(sortedLevels.size()-1).at(mu).energy) {
          sortedLevels[sortedLevels.size()-1].insert(mu,levels.at(la));
          levelsMap[levelsMap.size()-1].insert(mu,la);
          break;
        }
        else if(mu==sortedLevels.at(sortedLevels.size()-1).size()-1) {
          sortedLevels[sortedLevels.size()-1].append(levels.at(la));
          levelsMap[levelsMap.size()-1].append(la);
          break;
        }
      }
    }
      }
    }
  }
  int levelId=1;
  for(int i=0;i<sortedLevels.size();i++){
    for(int ii=0;ii<sortedLevels.at(i).size();ii++) {
      for(int ch=0;ch<channels.size();ch++) {
    if(channels.at(ch).levelIndex==levelsMap.at(i).at(ii)) {
```

```
      outStream « qSetFieldWidth(4)  « sortedLevels.at(i).at(ii).jValue
            « qSetFieldWidth(5)  « sortedLevels.at(i).at(ii).piValue
            « qSetFieldWidth(13) « sortedLevels.at(i).at(ii).energy
            « qSetFieldWidth(5)  « sortedLevels.at(i).at(ii).isFixed
            « qSetFieldWidth(5)  « "1"
            « qSetFieldWidth(5)  « channels.at(ch).pairIndex+1
            « qSetFieldWidth(5)  « int(channels.at(ch).sValue*2)
            « qSetFieldWidth(5)  « int(channels.at(ch).lValue*2)
            « qSetFieldWidth(5)  « levelId
            « qSetFieldWidth(5)  « sortedLevels.at(i).at(ii).isActive
            « qSetFieldWidth(5)  « channels.at(ch).isFixed
            « qSetFieldWidth(20) « channels.at(ch).reducedWidth
            « qSetFieldWidth(5)  « pairs.at(channels.at(ch).pairIndex).lightJ
            « qSetFieldWidth(5)  « pairs.at(channels.at(ch).pairIndex).lightPi
            « qSetFieldWidth(5)  « pairs.at(channels.at(ch).pairIndex).heavyJ
            « qSetFieldWidth(5)  « pairs.at(channels.at(ch).pairIndex).heavyPi
            « qSetFieldWidth(13) « pairs.at(channels.at(ch).pairIndex).excitationEnergy
            « qSetFieldWidth(8)  « pairs.at(channels.at(ch).pairIndex).lightM
            « qSetFieldWidth(8)  « pairs.at(channels.at(ch).pairIndex).heavyM
            « qSetFieldWidth(5)  « pairs.at(channels.at(ch).pairIndex).lightZ
            « qSetFieldWidth(5)  « pairs.at(channels.at(ch).pairIndex).heavyZ
            « qSetFieldWidth(13) « pairs.at(0).seperationEnergy
            « qSetFieldWidth(13) « pairs.at(channels.at(ch).pairIndex).seperationEnergy
            « "    0    0          0.0"
            « qSetFieldWidth(6)  « pairs.at(channels.at(ch).pairIndex).pairType
            « qSetFieldWidth(8)  « pairs.at(channels.at(ch).pairIndex).channelRadius
            « qSetFieldWidth(13)  « pairs.at(channels.at(ch).pairIndex).lightG
            « qSetFieldWidth(13)  « pairs.at(channels.at(ch).pairIndex).heavyG
            « qSetFieldWidth(8)  « pairs.at(channels.at(ch).pairIndex).ecMultMask
            « qSetFieldWidth(0)  « endl;
    }
      }
      outStream « endl;
      levelId++;
    }
  }

  outStream.setFieldAlignment(QTextStream::AlignLeft);

  return true;
}

bool LevelsTab::readNuclearFile(QTextStream &inStream) {

  double levelJ;
  int levelPi;
  double levelEnergy;
  int levelFix;
  int aa;
  int ir;
  double channelS;
  int channelL;
  int levelId;
  int levelYN;
  int channelFix;
  double channelReducedWidth;
  double lightJ;
  int lightPi;
  double heavyJ;
  int heavyPi;
  double excitationEnergy;
  double lightM;
  double heavyM;
  int lightZ;
  int heavyZ;
  double seperationEnergyIn;
  double seperationEnergyOut;
  int pairType;
  double channelRadius;
  double lightG;
  double heavyG;
  double dummyDouble;
  int dummyInt;
  int ecMultMask;

  maxLSpin->blockSignals(true);
  maxMultSpin->blockSignals(true);
  maxNumMultSpin->blockSignals(true);
  maxLSpin->setValue(0);
  maxMultSpin->setValue(0);
  maxNumMultSpin->setValue(0);

  int maxLValue=0;
  int maxMultValue=0;
  int maxNumMultValue=0;
  bool firstLine=true;;
  int lastPair;
```

```
  int currentPair=0;
  int thisNumMult=0;

  QString line("");
  while(!inStream.atEnd()&&line.trimmed()!=QString("</levels>")) {
    line=inStream.readLine();
    if(line.trimmed().isEmpty()) continue;
    if(!inStream.atEnd()&&line.trimmed()!=QString("</levels>")) {
      QTextStream in(&line);
      in » levelJ » levelPi » levelEnergy » levelFix » aa » ir » channelS » channelL » levelId »
levelYN » channelFix
     » channelReducedWidth » lightJ » lightPi » heavyJ » heavyPi » excitationEnergy
     » lightM » heavyM » lightZ » heavyZ » seperationEnergyIn » seperationEnergyOut
     » dummyInt » dummyInt » dummyDouble » pairType » channelRadius » lightG » heavyG;
      if(in.status()!=QTextStream::Ok) return false;
      in » ecMultMask;
      if(in.status()!=QTextStream::Ok) ecMultMask=0;
      if(firstLine) {
    lastPair=ir;
    firstLine=false;
      } else lastPair=currentPair;
      currentPair=ir;
      if(lastPair==currentPair) thisNumMult++;
      else thisNumMult=1;
      if(pairType==0) {
    if(channelL/2>maxLValue) maxLValue=channelL/2;
      } else if (pairType==10) {
    if(channelL/2>maxMultValue) maxMultValue=channelL/2;
    if(thisNumMult>maxNumMultValue) maxNumMultValue=thisNumMult;
      }

      PairsData newPair={lightJ,lightPi,lightZ,lightM,lightG,heavyJ,heavyPi,heavyZ,heavyM,
             heavyG,excitationEnergy,seperationEnergyOut,channelRadius,pairType,ecMultMask};
      int pairIndex=ir-1;
      if(pairsModel->numPairs()<ir) {
    emit(readNewPair(newPair,pairIndex,true));
      } else if(pairsModel->isPair(newPair) == -1) {
    emit(readExistingPair(newPair,pairIndex,true));
      }
      LevelsData newLevel = {levelYN,levelFix,levelJ,levelPi,levelEnergy};
      int levelIndex = levelsModel->isLevel(newLevel);
      if(levelIndex == -1) {
    addLevel(newLevel,true);
    levelIndex = levelsModel->isLevel(newLevel);
      }
      QChar radType;
      if(pairType==0) radType='P';
      else if(pairType==20) {
    if(channelL==0) radType = 'F';
    else radType = 'G';
      }
      else {
    int parityChange=heavyPi*levelPi;
    if((channelL/2)%2!=0) {
      if(parityChange==-1) radType='E';
      else radType='M';
    } else {
      if(parityChange==-1) radType='M';
      else radType='E';
    }
      }

      int channelIndex=channelsModel->getChannels().size();
      channelsModel->insertRows(channelIndex,1,QModelIndex());
      QModelIndex index = channelsModel->index(channelIndex,0,QModelIndex());
      channelsModel->setData(index,channelFix,Qt::EditRole);
      index = channelsModel->index(channelIndex,1,QModelIndex());
      channelsModel->setData(index,levelIndex,Qt::EditRole);
      index = channelsModel->index(channelIndex,2,QModelIndex());
      channelsModel->setData(index,pairIndex,Qt::EditRole);
      index = channelsModel->index(channelIndex,3,QModelIndex());
      channelsModel->setData(index,channelS/2.0,Qt::EditRole);
      index = channelsModel->index(channelIndex,4,QModelIndex());
      channelsModel->setData(index,channelL/2,Qt::EditRole);
      index = channelsModel->index(channelIndex,5,QModelIndex());
      channelsModel->setData(index,radType,Qt::EditRole);
      index = channelsModel->index(channelIndex,6,QModelIndex());
      channelsModel->setData(index,channelReducedWidth,Qt::EditRole);
      channelsView->resizeRowsToContents();
    }
  }
  if(line.trimmed()!=QString("</levels>")) return false;

  if(maxLValue>maxLSpin->value()) maxLSpin->setValue(maxLValue);
  if(maxMultValue>maxMultSpin->value()) maxMultSpin->setValue(maxMultValue);
  if(maxNumMultValue>maxNumMultSpin->value()) maxNumMultSpin->setValue(maxNumMultValue);
  maxLSpin->blockSignals(false);
```

```
  maxMultSpin->blockSignals(false);
  maxNumMultSpin->blockSignals(false);
  levelsView->resizeRowsToContents();

  return true;
}

void LevelsTab::reset() {
  pairsModel->removeRows(0,pairsModel->getPairs().size(),QModelIndex());
  levelsModel->removeRows(0,levelsModel->getLevels().size(),QModelIndex());
  channelsModel->removeRows(0,channelsModel->getChannels().size(),QModelIndex());
  maxLSpin->blockSignals(true);
  maxMultSpin->blockSignals(true);
  maxNumMultSpin->blockSignals(true);
  maxLSpin->setValue(2);
  maxMultSpin->setValue(2);
  maxNumMultSpin->setValue(2);
  maxLSpin->blockSignals(false);
  maxMultSpin->blockSignals(false);
  maxNumMultSpin->blockSignals(false);
}

void LevelsTab::showInfo(int which,QString title) {
  if(which<infoText.size()) {
    if(!infoDialog[which]) {
      infoDialog[which] = new InfoDialog(infoText[which],this,title);
      infoDialog[which]->setAttribute(Qt:: WA_DeleteOnClose);
      infoDialog[which]->show();
    } else infoDialog[which]->raise();
  }
}
```

8.113 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/main.cpp File Reference

```
#include <QMainWindow>
#include <QApplication>
#include <QResource>
#include <iostream>
#include "AZURESetup.h"
```

Functions

- void initResource ()
- int start_gui (int argc, char *argv[ ])

8.113.1 Function Documentation

8.113.1.1 initResource()

```
void initResource ( ) [inline]
```

Definition at line 7 of file main.cpp.

8.113.1.2 start_gui()

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

Definition at line 9 of file main.cpp.

8.114 main.cpp

Go to the documentation of this file.
```
#include <QMainWindow>
#include <QApplication>
#include <QResource>
#include <iostream>
#include "AZURESetup.h"

inline void initResource() { Q_INIT_RESOURCE(AZURESetup); }

int start_gui(int argc, char *argv[]) {

#ifdef Q_OS_MACX
    if ( QSysInfo::MacintoshVersion > QSysInfo::MV_10_8 )
    {
        // fix Mac OS X 10.9 (mavericks) font issue
        // https://bugreports.qt-project.org/browse/QTBUG-32789
        QFont::insertSubstitution(".Lucida Grande UI", "Lucida Grande");
    }
#endif

  QApplication app(argc, argv);

#ifdef WIN_SPACING
  QFont font = app.font();
  font.setPointSizeF(10.5);
  app.setFont(font);
#endif

  initResource();
  QCoreApplication::setOrganizationName("jina");
  QCoreApplication::setApplicationName("azure2");

  AZURESetup azureSetup;
  azureSetup.show();

  QString filename="";
  for(int i=1;i<argc;i++)
    if(strncmp(argv[i],"--",2)!=0) {
      filename=argv[i];
      break;
    }
  if(!filename.trimmed().isEmpty()) azureSetup.open(filename);

  return app.exec();
}
```

8.115 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PairsModel.cpp File Reference

```
#include ¨PairsModel.h¨
#include ¨ElementMap.h¨
#include <math.h>
```

8.116 PairsModel.cpp

Go to the documentation of this file.
```
#include "PairsModel.h"
#include "ElementMap.h"
#include <math.h>

PairsModel::PairsModel(QObject *parent) : QAbstractTableModel(parent) {
}

int PairsModel::rowCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return pairsList.size();
}
```

```

int PairsModel::columnCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return PairsData::SIZE;
}

QVariant PairsModel::data(const QModelIndex &index, int role) const {
  if(!index.isValid()) return QVariant();

  if(index.row() >= pairsList.size() || index.row() < 0)  return QVariant();

  if (role == Qt::DisplayRole) {
    PairsData pair = pairsList.at(index.row());
    if(index.column() == 2) return getSpinLabel(pair,0);
    else if(index.column() == 3) return pair.lightPi;
    else if(index.column() == 0) return getParticleLabel(pair,0);
    else if(index.column() == 1) return pair.lightM;
    else if(index.column() == 4) return pair.lightG;
    else if(index.column() == 7) return getSpinLabel(pair,1);
    else if(index.column() == 8) return pair.heavyPi;
    else if(index.column() == 5) return getParticleLabel(pair,1);
    else if(index.column() == 6) return pair.heavyM;
    else if(index.column() == 9) return pair.heavyG;
    else if(index.column() == 10) return pair.excitationEnergy;
    else if(index.column() == 11) return pair.seperationEnergy;
    else if(index.column() == 12) return pair.channelRadius;
    else if(index.column() == 13) {
      if(pair.pairType == 10) return "Particle, Gamma";
      else if(pair.pairType == 20) return "Beta Decay";
      else return "Particle, Particle";
    } else if(index.column() == 14) return pair.ecMultMask;
  } else if (role == Qt::EditRole) {
    PairsData pair = pairsList.at(index.row());
    if(index.column() == 0) return pair.lightJ;
    else if(index.column() == 1) return pair.lightPi;
    else if(index.column() == 2) return pair.lightZ;
    else if(index.column() == 3) return pair.lightM;
    else if(index.column() == 4) return pair.lightG;
    else if(index.column() == 5) return pair.heavyJ;
    else if(index.column() == 6) return pair.heavyPi;
    else if(index.column() == 7) return pair.heavyZ;
    else if(index.column() == 8) return pair.heavyM;
    else if(index.column() == 9) return pair.heavyG;
    else if(index.column() == 10) return pair.excitationEnergy;
    else if(index.column() == 11) return pair.seperationEnergy;
    else if(index.column() == 12) return pair.channelRadius;
    else if(index.column() == 13) return pair.pairType;
    else if(index.column() == 14) return pair.ecMultMask;
  } else if(role == Qt::TextAlignmentRole) return Qt::AlignCenter;

  return QVariant();
}

QVariant PairsModel::headerData(int section, Qt::Orientation orientation, int role) const {
  if(role!= Qt::DisplayRole) return QVariant();
  if(orientation == Qt::Horizontal) {
    switch(section) {
    case 2:
      return tr("Light\nSpin");
    case 3:
      return tr("Light\nParity");
    case 0:
      return tr("Light\n Particle");
    case 1:
      return tr("Light\nM");
    case 4:
      return tr("Light\ng-Factor");
    case 7:
      return tr("Heavy\nSpin");
    case 8:
      return tr("Heavy\nParity");
    case 5:
      return tr("Heavy\nParticle");
    case 6:
      return tr("Heavy\nM");
    case 9:
      return tr("Heavy\ng-Factor");
    case 10:
      return tr("Excitation\nEnergy");
    case 11:
      return tr("Separation\nEnergy");
    case 12:
      return tr("Channel\nRadius");
    case 13:
      return tr("Pair\nType");
    case 14:
      return tr("EC\nMultipolarities");
```

```
    default:
      return QVariant();
    }
  } else if(orientation == Qt::Vertical) {
    return section+1;
  }
  return QVariant();
}

bool PairsModel::setData(const QModelIndex &index, const QVariant &value, int role) {
  if (index.isValid() && role == Qt::EditRole ) {
    int row = index.row();
    PairsData tempData = pairsList.value(row);
    if (index.column() == 0) tempData.lightJ=value.toDouble();
    else if(index.column() == 1) tempData.lightPi=value.toInt();
    else if(index.column() == 2) tempData.lightZ=value.toInt();
    else if(index.column() == 3) tempData.lightM=value.toDouble();
    else if(index.column() == 4) tempData.lightG=value.toDouble();
    else if(index.column() == 5) tempData.heavyJ=value.toDouble();
    else if(index.column() == 6) tempData.heavyPi=value.toInt();
    else if(index.column() == 7) tempData.heavyZ=value.toInt();
    else if(index.column() == 8) tempData.heavyM=value.toDouble();
    else if(index.column() == 9) tempData.heavyG=value.toDouble();
    else if(index.column() == 10) tempData.excitationEnergy=value.toDouble();
    else if(index.column() == 11) tempData.seperationEnergy=value.toDouble();
    else if(index.column() == 12) tempData.channelRadius=value.toDouble();
    else if(index.column() == 13) tempData.pairType=value.toInt();
    else if(index.column() == 14) tempData.ecMultMask=value.toInt();
    else return false;

    pairsList.replace(row,tempData);
    emit(dataChanged(index,index));
    return true;
  }
  return false;
}

bool PairsModel::insertRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginInsertRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows; row++) {
      PairsData tempData;
      pairsList.insert(position,tempData);
    }
    endInsertRows();
  }
  return true;
}

bool PairsModel::removeRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginRemoveRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows;++row) {
      pairsList.removeAt(position);
    }
    endRemoveRows();
  }
  return true;
}

int PairsModel::isPair(const PairsData &pair) const {
  int foundPair=-1;
  for(int i=0;i<pairsList.size();i++) {
    PairsData tempPair=pairsList.value(i);
    if(tempPair.lightJ==pair.lightJ&&
       tempPair.lightPi==pair.lightPi&&
       tempPair.lightZ==pair.lightZ&&
       tempPair.lightM==pair.lightM&&
       tempPair.lightG==pair.lightG&&
       tempPair.heavyJ==pair.heavyJ&&
       tempPair.heavyPi==pair.heavyPi&&
       tempPair.heavyZ==pair.heavyZ&&
       tempPair.heavyM==pair.heavyM&&
       tempPair.heavyG==pair.heavyG&&
       tempPair.seperationEnergy==pair.seperationEnergy&&
       tempPair.excitationEnergy==pair.excitationEnergy&&
       tempPair.channelRadius==pair.channelRadius&&
       tempPair.pairType==pair.pairType&&
       tempPair.ecMultMask==pair.ecMultMask) {
      foundPair=i;
      break;
    }
  }
  return foundPair;
}
```

```
QString PairsModel::getParticleLabel(const PairsData &pair, int which) const {
  if(which!=-1) {
    if(pair.pairType==10&&which==0) return "<center>&gamma;</center>";
    else if(pair.pairType==20&&which==0) {
      if(pair.lightZ<0) return "<center>&beta;<sup>-</sup></center>";
      else return "<center>&beta;<sup>+</sup></center>";
    } else {
      int tempZ;
      int tempM;
      if(which==0) {
    tempZ=pair.lightZ;
    tempM=round(pair.lightM);
      } else {
    tempZ=pair.heavyZ;
    tempM=round(pair.heavyM);
      }
      std::map<int, QString>::const_iterator it=elementMap.find(tempZ);
      if(it!=elementMap.end()) {
    if(tempM==1) {
      if(tempZ==1) return "<center><i>p</i></center>";
      else return QString("<center><i>%1</i></center>").arg(it->second);
    } else if(tempZ==2&&tempM==4) return "<center>&alpha;</center>";
    else return QString("<center><sup>%1</sup>%2</center>").arg(tempM).arg(it->second);
      } else return "?";
    }
  } else {
    QString lightLabel;
    std::map<int, QString>::const_iterator it=elementMap.find(pair.lightZ);
    if(pair.pairType==10) lightLabel="&gamma;";
    else if(pair.pairType==20) {
      if(pair.lightZ<0.) lightLabel="&beta;<sup>-</sup>";
      else lightLabel="&beta;<sup>+</sup>";
    } else if(it!=elementMap.end()) {
      if(round(pair.lightM)==1) {
    if(pair.lightZ==1) lightLabel="<i>p</i>";
    else lightLabel=QString("<i>%1</i>").arg(it->second);
      } else if(pair.lightZ==2&&round(pair.lightM)==4) lightLabel="&alpha;";
      else lightLabel=QString("<sup>%1</sup>%2").arg(round(pair.lightM)).arg(it->second);
    } else lightLabel="?";
    QString heavyLabel;
    it=elementMap.find(pair.heavyZ);
    if(it!=elementMap.end()) {
      if(round(pair.heavyM)==1) {
    if(pair.heavyZ==1) heavyLabel="<i>p</i>";
    else heavyLabel=QString("<i>%1</i>").arg(it->second);
      } else if(pair.heavyZ==2&&round(pair.heavyM)==4) heavyLabel="&alpha;";
      else heavyLabel=QString("<sup>%1</sup>%2").arg(round(pair.heavyM)).arg(it->second);
    } else heavyLabel="?";
    if(pair.pairType==20) return QString("<center>%1(%2) [%3 MeV]</center>").arg(heavyLabel).arg(lightLabel).arg(pair.excitationEnergy,0,'f',3);
    else return QString("<center>%1+%2 [%3 MeV]</center>").arg(heavyLabel).arg(lightLabel).arg(pair.excitationEnergy,0,'f',3);
  }
}

QString PairsModel::getReactionLabel(const PairsData &firstPair, const PairsData &secondPair) {
  QString lightLabel[2];
  std::map<int, QString>::const_iterator it=elementMap.find(firstPair.lightZ);
  if(firstPair.pairType==10) lightLabel[0]="&gamma;";
  else if(firstPair.pairType==20) {
    if(firstPair.lightZ<0) lightLabel[0]="&beta;<sup>-</sup>";
    else lightLabel[0]="&beta;<sup>+</sup>";
  } else if(it!=elementMap.end()) {
    if(round(firstPair.lightM)==1) {
      if(firstPair.lightZ==1) lightLabel[0]="<i>p</i>";
      else lightLabel[0]=QString("<i>%1</i>").arg(it->second);
    } else if(firstPair.lightZ==2&&round(firstPair.lightM)==4) lightLabel[0]="&alpha;";
    else lightLabel[0]=QString("<sup>%1</sup>%2").arg(round(firstPair.lightM)).arg(it->second);
  } else lightLabel[0]="?";
  it=elementMap.find(secondPair.lightZ);
  if(secondPair.pairType==10) lightLabel[1]="&gamma;";
  else if(secondPair.pairType==20) {
    if(secondPair.lightZ<0) lightLabel[1]="&beta;<sup>-</sup>";
    else lightLabel[1]="&beta;<sup>+</sup>";
  } else if(it!=elementMap.end()) {
    if(round(secondPair.lightM)==1) {
      if(secondPair.lightZ==1) lightLabel[1]="<i>p</i>";
      else lightLabel[1]=QString("<i>%1</i>").arg(it->second);
    } else if(secondPair.lightZ==2&&round(secondPair.lightM)==4) lightLabel[1]="&alpha;";
    else lightLabel[1]=QString("<sup>%1</sup>%2").arg(round(secondPair.lightM)).arg(it->second);
  } else lightLabel[1]="?";
  QString heavyLabel[2];
  it=elementMap.find(firstPair.heavyZ);
  if(it!=elementMap.end()) {
    if(round(firstPair.heavyM)==1) {
      if(firstPair.heavyZ==1) heavyLabel[0]="<i>p</i>";
```

```
      else heavyLabel[0]=QString("<i>%1</i>").arg(it->second);
    } else if(firstPair.heavyZ==2&&round(firstPair.heavyM)==4) heavyLabel[0]="&alpha;";
    else heavyLabel[0]=QString("<sup>%1</sup>%2").arg(round(firstPair.heavyM)).arg(it->second);
  } else heavyLabel[0]="?";
  it=elementMap.find(secondPair.heavyZ);
  if(it!=elementMap.end()) {
    if(round(secondPair.heavyM)==1) {
      if(secondPair.heavyZ==1) heavyLabel[1]="<i>p</i>";
      else heavyLabel[1]=QString("<i>%1</i>").arg(it->second);
    } else if(secondPair.heavyZ==2&&round(secondPair.heavyM)==4) heavyLabel[1]="&alpha;";
    else heavyLabel[1]=QString("<sup>%1</sup>%2").arg(round(secondPair.heavyM)).arg(it->second);
  } else heavyLabel[1]="?";
  if(firstPair.pairType==20) return QString("%1(%2%3)%4 [%5
MeV]").arg(heavyLabel[0]).arg(lightLabel[0]).arg(lightLabel[1]).arg(heavyLabel[1]).arg(secondPair.excitationEnergy,0,'f'
  else return QString("%1(%2,%3)%4 [%5
MeV]").arg(heavyLabel[0]).arg(lightLabel[0]).arg(lightLabel[1]).arg(heavyLabel[1]).arg(secondPair.excitationEnergy,0,'f'
}

QString PairsModel::getReactionLabelTotalCapture(const PairsData &firstPair) {
  QString lightLabel[2];
  std::map<int, QString>::const_iterator it=elementMap.find(firstPair.lightZ);
  if(firstPair.pairType==10) lightLabel[0]="&gamma;";
  else if(firstPair.pairType==20) {
    if(firstPair.lightZ<0) lightLabel[0]="&beta;<sup>-</sup>";
    else lightLabel[0]="&beta;<sup>+</sup>";
  } else if(it!=elementMap.end()) {
    if(round(firstPair.lightM)==1) {
      if(firstPair.lightZ==1) lightLabel[0]="<i>p</i>";
      else lightLabel[0]=QString("<i>%1</i>").arg(it->second);
    } else if(firstPair.lightZ==2&&round(firstPair.lightM)==4) lightLabel[0]="&alpha;";
    else lightLabel[0]=QString("<sup>%1</sup>%2").arg(round(firstPair.lightM)).arg(it->second);
  } else lightLabel[0]="?";
  lightLabel[1]="&gamma;";
  QString heavyLabel[2];
  it=elementMap.find(firstPair.heavyZ);
  if(it!=elementMap.end()) {
    if(round(firstPair.heavyM)==1) {
      if(firstPair.heavyZ==1) heavyLabel[0]="<i>p</i>";
      else heavyLabel[0]=QString("<i>%1</i>").arg(it->second);
    } else if(firstPair.heavyZ==2&&round(firstPair.heavyM)==4) heavyLabel[0]="&alpha;";
    else heavyLabel[0]=QString("<sup>%1</sup>%2").arg(round(firstPair.heavyM)).arg(it->second);
  } else heavyLabel[0]="?";
  int i;
  for(i=0;i<pairsList.size();i++)
    if(pairsList[i].pairType==10) break;
  if(i==pairsList.size()) heavyLabel[1]="?";
  else {
    PairsData secondPair = pairsList[i];
    it=elementMap.find(secondPair.heavyZ);
    if(it!=elementMap.end()) {
      if(round(secondPair.heavyM)==1) {
    if(secondPair.heavyZ==1) heavyLabel[1]="<i>p</i>";
    else heavyLabel[1]=QString("<i>%1</i>").arg(it->second);
      } else if(secondPair.heavyZ==2&&round(secondPair.heavyM)==4) heavyLabel[1]="&alpha;";
      else heavyLabel[1]=QString("<sup>%1</sup>%2").arg(round(secondPair.heavyM)).arg(it->second);
    } else heavyLabel[1]="?";
  }
  if(firstPair.pairType==20) return QString("%1(%2%3)%4
[TOTAL]").arg(heavyLabel[0]).arg(lightLabel[0]).arg(lightLabel[1]).arg(heavyLabel[1]);
  else return QString("%1(%2,%3)%4
[TOTAL]").arg(heavyLabel[0]).arg(lightLabel[0]).arg(lightLabel[1]).arg(heavyLabel[1]);
}

QString PairsModel::getSpinLabel(const PairsData &pair, int which) const {
  double tempJ;
  int tempPi;
  if(which==0) {
    tempJ=pair.lightJ;
    tempPi=pair.lightPi;
  } else {
    tempJ=pair.heavyJ;
    tempPi=pair.heavyPi;
  }
  QString tempSpin;
  if(((int)(tempJ*2))%2!=0&&tempJ!=0.) tempSpin=QString("%1/2").arg((int)(tempJ*2));
  else tempSpin=QString("%1").arg(tempJ);
  if(tempPi==-1) return QString("%1-").arg(tempSpin);
  else return QString("%1+").arg(tempSpin);
}
```

8.117 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PairsTab.cpp File Reference

```
#include <QHeaderView>
#include <QGridLayout>
#include <QComboBox>
#include <QLineEdit>
#include <QCheckBox>
#include <QTextStream>
#include <QMessageBox>
#include "PairsTab.h"
#include "RichTextDelegate.h"
#include "InfoDialog.h"
#include <iostream>
```

8.118 PairsTab.cpp

Go to the documentation of this file.

```
#include <QHeaderView>
#include <QGridLayout>
#include <QComboBox>
#include <QLineEdit>
#include <QCheckBox>
#include <QTextStream>
#include <QMessageBox>

#include "PairsTab.h"
#include "RichTextDelegate.h"
#include "InfoDialog.h"
#include <iostream>

PairsTab::PairsTab(QWidget *parent) : QWidget(parent) {
  pairsModel = new PairsModel(this);

  pairsView = new QTableView;
  pairsView->setModel(pairsModel);
  pairsView->verticalHeader()->setHighlightSections(false);
  pairsView->horizontalHeader()->setHighlightSections(false);
  pairsView->setColumnHidden(1,true);
  pairsView->setColumnHidden(3,true);
  pairsView->setColumnHidden(4,true);
  pairsView->setColumnHidden(6,true);
  pairsView->setColumnHidden(8,true);
  pairsView->setColumnHidden(9,true);
  pairsView->setColumnHidden(13,true);
  pairsView->setColumnHidden(14,true);
  RichTextDelegate *rt = new RichTextDelegate();
  pairsView->setItemDelegateForColumn(0,rt);
  pairsView->setItemDelegateForColumn(5,rt);
  pairsView->setSelectionBehavior(QAbstractItemView::SelectRows);
  pairsView->setSelectionMode(QAbstractItemView::SingleSelection);
  pairsView->setEditTriggers(QAbstractItemView::NoEditTriggers);
  pairsView->setShowGrid(false);

connect(pairsView->selectionModel(),SIGNAL(selectionChanged(QItemSelection,QItemSelection)),this,SLOT(updateButtons(QIt
  connect(pairsView,SIGNAL(doubleClicked(QModelIndex)),this,SLOT(editPair()));

  for(int i = 0; i<PairsData::SIZE;i++)
pairsView->horizontalHeader()->setSectionResizeMode(i,QHeaderView::Stretch);

  addButton=new QPushButton(tr("+"));
  addButton->setMaximumSize(28,28);
  connect(addButton,SIGNAL(clicked()),this,SLOT(addPair()));
  deleteButton = new QPushButton(tr("-"));
  deleteButton->setMaximumSize(28,28);
  deleteButton->setEnabled(false);
  connect(deleteButton,SIGNAL(clicked()),this,SLOT(removePair()));

  /*
  mapper = new QSignalMapper(this);
```

```
  connect(mapper,SIGNAL(mapped(int)),this,SLOT(showInfo(int)));

  infoButton[0] = new QPushButton(this);
  infoButton[0]->setMaximumSize(28,28);
  infoButton[0]->setIcon(style()->standardIcon(QStyle::SP_MessageBoxInformation));
  mapper->setMapping(infoButton[0],0);
  connect(infoButton[0],SIGNAL(clicked()),mapper,SLOT(map()));
  */

  QGridLayout *buttonBox = new QGridLayout;
  buttonBox->addWidget(addButton,0,0);
  buttonBox->addWidget(deleteButton,0,1);
  buttonBox->addItem(new QSpacerItem(28,28),0,2);
  // buttonBox->addWidget(infoButton[0],0,3);
  buttonBox->setColumnStretch(0,0);
  buttonBox->setColumnStretch(1,0);
  buttonBox->setColumnStretch(2,1);
  buttonBox->setColumnStretch(3,0);
#ifdef MACX_SPACING
  buttonBox->setHorizontalSpacing(11);
#else
  buttonBox->setHorizontalSpacing(0);
#endif

  QVBoxLayout *layout = new QVBoxLayout();
  layout->addWidget(pairsView);
  layout->addLayout(buttonBox);
  setLayout(layout);
}

PairsModel *PairsTab::getPairsModel() {
  return pairsModel;
}

void PairsTab::addPair() {
  AddPairDialog aDialog;
  if(aDialog.exec()) {
    if(!((pairsModel->getPairs()).size()==0&&aDialog.pairTypeCombo->currentIndex()!=0)) {
      PairsData newPair;
      newPair.lightJ=(aDialog.lightJText->text()).toDouble();
      if(aDialog.lightPiCombo->currentIndex() == 0) newPair.lightPi=-1;
      else newPair.lightPi=1;
      newPair.lightZ=(aDialog.lightZText->text()).toInt();
      newPair.lightM=(aDialog.lightMText->text()).toDouble();
      newPair.lightG=0.;//(aDialog.lightGText->text()).toDouble();
      newPair.heavyJ=(aDialog.heavyJText->text()).toDouble();
      if(aDialog.heavyPiCombo->currentIndex() == 0) newPair.heavyPi=-1;
      else newPair.heavyPi=1;
      newPair.heavyZ=(aDialog.heavyZText->text()).toInt();
      newPair.heavyM=(aDialog.heavyMText->text()).toDouble();
      newPair.heavyG=0.;//(aDialog.heavyGText->text()).toDouble();
      newPair.seperationEnergy=(aDialog.seperationEnergyText->text()).toDouble();
      newPair.excitationEnergy=(aDialog.excitationEnergyText->text()).toDouble();
      newPair.channelRadius=(aDialog.channelRadiusText->text()).toDouble();
      if(aDialog.pairTypeCombo->currentIndex() == 1) newPair.pairType=10;
      else if(aDialog.pairTypeCombo->currentIndex() == 2) newPair.pairType=20;
      else newPair.pairType=0;
      unsigned int newMask=0;
      if(aDialog.e1Check->isChecked()) newMask |= (1«0);
      //if(aDialog.m1Check->isChecked()) newMask |= (1«1);
      if(aDialog.e2Check->isChecked()) newMask |= (1«2);
      newPair.ecMultMask=newMask;
      addPair(newPair,pairsModel->numPairs(),false);
    } else {
      QMessageBox::information(this, tr("Pair Type Error"),
                               tr("The first pair must be a particle,particle pair."));
    }
  }
}

void PairsTab::addPair(PairsData pair,int pairIndex,bool fromFile) {
  if(pairsModel->isPair(pair)==-1) {
    pairsModel->insertRows(pairsModel->numPairs(),pairIndex+1-pairsModel->numPairs(),QModelIndex());

    QModelIndex index = pairsModel->index(pairIndex,0,QModelIndex());
    pairsModel->setData(index,pair.lightJ,Qt::EditRole);
    index = pairsModel->index(pairIndex,1,QModelIndex());
    pairsModel->setData(index,pair.lightPi,Qt::EditRole);
    index = pairsModel->index(pairIndex,2,QModelIndex());
    pairsModel->setData(index,pair.lightZ,Qt::EditRole);
    index = pairsModel->index(pairIndex,3,QModelIndex());
    pairsModel->setData(index,pair.lightM,Qt::EditRole);
    index = pairsModel->index(pairIndex,4,QModelIndex());
    pairsModel->setData(index,pair.lightG,Qt::EditRole);
    index = pairsModel->index(pairIndex,5,QModelIndex());
    pairsModel->setData(index,pair.heavyJ,Qt::EditRole);
    index = pairsModel->index(pairIndex,6,QModelIndex());
```

```
    pairsModel->setData(index,pair.heavyPi,Qt::EditRole);
    index = pairsModel->index(pairIndex,7,QModelIndex());
    pairsModel->setData(index,pair.heavyZ,Qt::EditRole);
    index = pairsModel->index(pairIndex,8,QModelIndex());
    pairsModel->setData(index,pair.heavyM,Qt::EditRole);
    index = pairsModel->index(pairIndex,9,QModelIndex());
    pairsModel->setData(index,pair.heavyG,Qt::EditRole);
    index = pairsModel->index(pairIndex,10,QModelIndex());
    pairsModel->setData(index,pair.excitationEnergy,Qt::EditRole);
    index = pairsModel->index(pairIndex,11,QModelIndex());
    pairsModel->setData(index,pair.seperationEnergy,Qt::EditRole);
    index = pairsModel->index(pairIndex,12,QModelIndex());
    pairsModel->setData(index,pair.channelRadius,Qt::EditRole);
    index = pairsModel->index(pairIndex,13,QModelIndex());
    pairsModel->setData(index,pair.pairType,Qt::EditRole);
    index = pairsModel->index(pairIndex,14,QModelIndex());
    pairsModel->setData(index,pair.ecMultMask,Qt::EditRole);

    pairsView->resizeRowsToContents();
    if(!fromFile) emit(pairAdded(pairIndex));
  } else {
    QMessageBox::information(this,tr("Duplicate Pair"),
                 tr("This pair already exists."));
  }
}

void PairsTab::removePair() {
  QItemSelectionModel *selectionModel = pairsView->selectionModel();
  QModelIndexList indexes = selectionModel->selectedRows();
  QModelIndex index=indexes[0];
  bool previousIsGamma=false;
  if(index.row()==0&&(pairsModel->getPairs()).size()!=1) {
    QModelIndex previousIndex=pairsModel->index(index.row()+1,13,QModelIndex());
    QVariant previousPairType=pairsModel->data(previousIndex,Qt::EditRole);
    if (previousPairType.toInt()==10) previousIsGamma=true;
  }
  if(!previousIsGamma) {
    pairsModel->removeRows(index.row(),1,QModelIndex());
    emit(pairRemoved(index.row()));
  } else {
    QMessageBox::information(this,tr("Entrance Channel Error"),
                 tr("This delete will result in a forbidden particle,gamma entrance pair."));
  }
}

void PairsTab::editPair() {
  QItemSelectionModel *selectionModel = pairsView->selectionModel();
  QModelIndexList indexes = selectionModel->selectedRows();

  QModelIndex index=indexes[0];
  QModelIndex i = pairsModel->index(index.row(), 0, QModelIndex());
  QVariant var = pairsModel->data(i, Qt::EditRole);
  QString lightJ = var.toString();
  i = pairsModel->index(index.row(), 1, QModelIndex());
  var =pairsModel->data(i, Qt::EditRole);
  int lightPi = var.toInt();
  i = pairsModel->index(index.row(), 2, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  QString lightZ = var.toString();
  i = pairsModel->index(index.row(), 3, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  QString lightM = var.toString();
  i = pairsModel->index(index.row(), 4, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  QString lightG = var.toString();
  i = pairsModel->index(index.row(), 5, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  QString  heavyJ = var.toString();
  i = pairsModel->index(index.row(), 6, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  int  heavyPi = var.toInt();
  i = pairsModel->index(index.row(), 7, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  QString heavyZ = var.toString();
  i = pairsModel->index(index.row(), 8, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  QString heavyM = var.toString();
  i = pairsModel->index(index.row(), 9, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  QString heavyG = var.toString();
  i = pairsModel->index(index.row(), 10, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  QString excitationEnergy = var.toString();
  i = pairsModel->index(index.row(), 11, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  QString seperationEnergy = var.toString();
  i = pairsModel->index(index.row(), 12, QModelIndex());
```

```
  var = pairsModel->data(i, Qt::EditRole);
  QString channelRadius = var.toString();
  i = pairsModel->index(index.row(), 13, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  int pairType = var.toInt();
  i = pairsModel->index(index.row(), 14, QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  int ecMultMask = var.toInt();


  AddPairDialog aDialog;
  aDialog.setWindowTitle(tr("Edit a Particle Pair"));
  aDialog.lightJText->setText(lightJ);
  if(lightPi==-1) aDialog.lightPiCombo->setCurrentIndex(0);
  else aDialog.lightPiCombo->setCurrentIndex(1);
  aDialog.lightZText->setText(lightZ);
  aDialog.lightMText->setText(lightM);
  //aDialog.lightGText->setText(lightG);
  aDialog.heavyJText->setText(heavyJ);
  if(heavyPi==-1) aDialog.heavyPiCombo->setCurrentIndex(0);
  else aDialog.heavyPiCombo->setCurrentIndex(1);
  aDialog.heavyZText->setText(heavyZ);
  aDialog.heavyMText->setText(heavyM);
  //aDialog.heavyGText->setText(heavyG);
  aDialog.excitationEnergyText->setText(excitationEnergy);
  aDialog.seperationEnergyText->setText(seperationEnergy);
  aDialog.channelRadiusText->setText(channelRadius);
  if(pairType == 10) aDialog.pairTypeCombo->setCurrentIndex(1);
  else if(pairType == 20) aDialog.pairTypeCombo->setCurrentIndex(2);
  else aDialog.pairTypeCombo->setCurrentIndex(0);
  if(ecMultMask&(1«0)) aDialog.e1Check->setChecked(true);
  else aDialog.e1Check->setChecked(false);
  //if(ecMultMask&(1«1)) aDialog.m1Check->setChecked(true);
  //else aDialog.m1Check->setChecked(false);
  if(ecMultMask&(1«2)) aDialog.e2Check->setChecked(true);
  else aDialog.e2Check->setChecked(false);

  if (aDialog.exec()) {
    if(!(index.row()==0&&aDialog.pairTypeCombo->currentIndex()!=0)) {
      PairsData pair;
      pair.lightJ = aDialog.lightJText->text().toDouble();
      if(aDialog.lightPiCombo->currentIndex()==0) pair.lightPi=-1;
      else pair.lightPi=1;
      pair.lightZ = aDialog.lightZText->text().toInt();
      pair.lightM = aDialog.lightMText->text().toDouble();
      pair.lightG = 0.;//aDialog.lightGText->text().toDouble();
      pair.heavyJ = aDialog.heavyJText->text().toDouble();
      if(aDialog.heavyPiCombo->currentIndex()==0) pair.heavyPi=-1;
      else pair.heavyPi=1;
      pair.heavyZ = aDialog.heavyZText->text().toInt();
      pair.heavyM = aDialog.heavyMText->text().toDouble();
      pair.heavyG = 0.;//aDialog.heavyGText->text().toDouble();
      pair.excitationEnergy = aDialog.excitationEnergyText->text().toDouble();
      pair.seperationEnergy = aDialog.seperationEnergyText->text().toDouble();
      pair.channelRadius = aDialog.channelRadiusText->text().toDouble();
      if(aDialog.pairTypeCombo->currentIndex()==1) pair.pairType=10;
      else if(aDialog.pairTypeCombo->currentIndex()==2) pair.pairType=20;
      else pair.pairType=0;
      unsigned char newECMultMask=0;
      if(aDialog.e1Check->isChecked()) newECMultMask |= (1«0);
      //if(aDialog.m1Check->isChecked()) newECMultMask |= (1«1);
      if(aDialog.e2Check->isChecked()) newECMultMask |= (1«2);
      pair.ecMultMask=newECMultMask;
      editPair(pair,index.row(),false);
    } else {
      QMessageBox::information(this,tr("Pair Type Error"),
                    tr("The first pair must be a particle,particle pair."));
    }
  }
}

void PairsTab::editPair(PairsData pair,int pairIndex,bool fromFile) {
  QModelIndex i = pairsModel->index(pairIndex,0,QModelIndex());
  QVariant var = pairsModel->data(i, Qt::EditRole);
  if (pair.lightJ != var.toDouble()) pairsModel->setData(i,pair.lightJ, Qt::EditRole);
  i = pairsModel->index(pairIndex,1,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.lightPi != var.toInt()) pairsModel->setData(i,pair.lightPi, Qt::EditRole);
  i = pairsModel->index(pairIndex,2,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.lightZ != var.toInt()) pairsModel->setData(i,pair.lightZ, Qt::EditRole);
  i = pairsModel->index(pairIndex,3,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.lightM != var.toDouble()) pairsModel->setData(i,pair.lightM, Qt::EditRole);
  i = pairsModel->index(pairIndex,4,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.lightG != var.toDouble()) pairsModel->setData(i,pair.lightG, Qt::EditRole);
```

```
  i = pairsModel->index(pairIndex,5,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.heavyJ != var.toDouble()) pairsModel->setData(i,pair.heavyJ, Qt::EditRole);
  i = pairsModel->index(pairIndex,6,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.heavyPi != var.toInt()) pairsModel->setData(i,pair.heavyPi, Qt::EditRole);
  i = pairsModel->index(pairIndex,7,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.heavyZ != var.toInt()) pairsModel->setData(i,pair.heavyZ, Qt::EditRole);
  i = pairsModel->index(pairIndex,8,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.heavyM != var.toDouble()) pairsModel->setData(i,pair.heavyM, Qt::EditRole);
  i = pairsModel->index(pairIndex,9,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.heavyG != var.toDouble()) pairsModel->setData(i,pair.heavyG, Qt::EditRole);
  i = pairsModel->index(pairIndex,10,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.excitationEnergy != var.toDouble()) pairsModel->setData(i,pair.excitationEnergy,
Qt::EditRole);
  i = pairsModel->index(pairIndex,11,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.seperationEnergy != var.toDouble()) pairsModel->setData(i,pair.seperationEnergy,
Qt::EditRole);
  i = pairsModel->index(pairIndex,12,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.channelRadius != var.toDouble()) pairsModel->setData(i,pair.channelRadius, Qt::EditRole);
  i = pairsModel->index(pairIndex,13,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.pairType != var.toInt()) pairsModel->setData(i,pair.pairType, Qt::EditRole);
  i = pairsModel->index(pairIndex,14,QModelIndex());
  var = pairsModel->data(i, Qt::EditRole);
  if (pair.pairType != var.toInt()) pairsModel->setData(i,pair.ecMultMask, Qt::EditRole);

  if(!fromFile) emit(pairEdited(pairIndex));
}

void PairsTab::updateButtons(const QItemSelection &selection) {
  QModelIndexList indexes = selection.indexes();

  if (!indexes.isEmpty()) {
    deleteButton->setEnabled(true);
  } else {
    deleteButton->setEnabled(false);
  }
}

bool PairsTab::parseOldECSection(QTextStream& inStream) {
  int isActive;
  int exitPairIndex;
  double minJ;
  double maxJ;
  int multMask;
  QString line("");
  while(line.trimmed()!=QString("</externalCapture>")&&!inStream.atEnd()) {
    line = inStream.readLine();
    if(line.trimmed().isEmpty()) continue;
    if(line.trimmed()!=QString("</externalCapture>")&&!inStream.atEnd()) {
      QTextStream in(&line);
      in » isActive » exitPairIndex » minJ » maxJ » multMask;
      if(in.status()!=QTextStream::Ok) return false;
      if(exitPairIndex-1<pairsModel->getPairs().size()) {
    QModelIndex i = pairsModel->index(exitPairIndex-1,14,QModelIndex());
    QVariant var = pairsModel->data(i, Qt::EditRole);
    pairsModel->setData(i,multMask, Qt::EditRole);
      }
    }
  }
  if(line.trimmed()!=QString("</externalCapture>")) return false;
  return true;
}

void PairsTab::showInfo(int which, QString title) {
  if(which<infoText.size()) {
    if(!infoDialog[which]) {
      infoDialog[which] = new InfoDialog(infoText[which],this,title);
      infoDialog[which]->setAttribute(Qt:: WA_DeleteOnClose);
      infoDialog[which]->show();
    } else infoDialog[which]->raise();
  }
}
```

8.119 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PlotTab.cpp File Reference

```
#include <QVBoxLayout>
#include <QGridLayout>
#include <QGroupBox>
#include <QComboBox>
#include <QRadioButton>
#include <QCheckBox>
#include <QPushButton>
#include <QListView>
#include "PlotTab.h"
#include "Config.h"
#include "AZUREPlot.h"
#include "SegmentsDataModel.h"
#include "SegmentsTestModel.h"
#include "RichTextDelegate.h"
#include "InfoDialog.h"
#include <iostream>
```

8.120 PlotTab.cpp

Go to the documentation of this file.

```
#include <QVBoxLayout>
#include <QGridLayout>
#include <QGroupBox>
#include <QComboBox>
#include <QRadioButton>
#include <QCheckBox>
#include <QPushButton>
#include <QCheckBox>
#include <QListView>

#include "PlotTab.h"
#include "Config.h"
#include "AZUREPlot.h"
#include "SegmentsDataModel.h"
#include "SegmentsTestModel.h"
#include "RichTextDelegate.h"
#include "InfoDialog.h"
#include <iostream>

QVariant SegTestProxyModel::data(const QModelIndex& index, int role) const {
  if (index.isValid() && role == Qt::DisplayRole) {
    QModelIndex sourceIndex = mapToSource(index);
    return QString("#%1: %2").arg(sourceIndex.row()+1).arg(static_cast<SegmentsTestModel*>(sourceModel())->getReactionLabel(sourceIndex));
  }
  return QVariant();
}


bool SegTestProxyModel::filterAcceptsRow(int source_row, const QModelIndex &source_parent) const {
  if(QSortFilterProxyModel::filterAcceptsRow(source_row,source_parent)) {
    SegmentsTestModel* model = static_cast<SegmentsTestModel*>(sourceModel());
    QModelIndex source_index = model->index(source_row, 9, source_parent);
    int dataType = model->data(source_index,Qt::EditRole).toInt();
    if(dataType!=3) return true;
    return false;
  }
  return false;
}

QVariant SegDataProxyModel::data(const QModelIndex& index, int role) const {
  if (index.isValid() && role == Qt::DisplayRole) {
    QModelIndex sourceIndex = mapToSource(index);
    return QString("#%1: %2").arg(sourceIndex.row()+1).arg(static_cast<SegmentsDataModel*>(sourceModel())->getReactionLabel(sourceIndex));
  }
```

```
  return QVariant();
}



PlotTab::PlotTab(Config& config, SegmentsDataModel* dataModel, SegmentsTestModel* testModel, QWidget* parent) :
  configure(config), QWidget(parent)  {
  azurePlot = new AZUREPlot(this,this);

  segDataProxyModel = new SegDataProxyModel(this);
  segDataProxyModel->setSourceModel(dataModel);
  segDataProxyModel->setDynamicSortFilter(true);
  segDataProxyModel->setFilterKeyColumn(0);
  segDataProxyModel->setFilterRole(Qt::CheckStateRole);
  segDataProxyModel->setFilterRegExp(QString("%1").arg(Qt::Checked));
  segTestProxyModel = new SegTestProxyModel(this);
  segTestProxyModel->setSourceModel(testModel);
  segTestProxyModel->setDynamicSortFilter(true);
  segTestProxyModel->setFilterKeyColumn(0);
  segTestProxyModel->setFilterRole(Qt::CheckStateRole);
  segTestProxyModel->setFilterRegExp(QString("%1").arg(Qt::Checked));

  QVBoxLayout* rightLayout = new QVBoxLayout;

  QGridLayout *topLayout = new QGridLayout;

  QGroupBox *xAxisBox = new QGroupBox(tr("X-Axis Configuration"));
  xAxisTypeCombo=new QComboBox;
  xAxisTypeCombo->addItem(tr("CoM Energy"));
  xAxisTypeCombo->addItem(tr("Excitation Energy"));
  xAxisTypeCombo->addItem(tr("CoM Angle"));
  connect(xAxisTypeCombo,SIGNAL(activated(int)),this,SLOT(xAxisTypeChanged()));
  xAxisTypeCombo->setCurrentIndex(0);
  azurePlot->setXAxisType(0);
#ifdef MACX_SPACING
  xAxisIsLogCheck = new QCheckBox(tr("Use Log Scale"));
#else
  xAxisIsLogCheck = new QCheckBox(tr("Use Logarithmic Scale"));
#endif
  connect(xAxisIsLogCheck,SIGNAL(toggled(bool)),this,SLOT(xAxisLogScaleChanged(bool)));
  QHBoxLayout* xAxisLayout = new QHBoxLayout;
  xAxisLayout->setContentsMargins(5,5,5,5);
  xAxisLayout->addWidget(xAxisTypeCombo);
  xAxisLayout->addWidget(xAxisIsLogCheck);
  xAxisBox->setLayout(xAxisLayout);
  QGroupBox *yAxisBox = new QGroupBox(tr("Y-Axis Configuration"));
  yAxisXSButton = new QRadioButton(tr("Cross Section"));
  connect(yAxisXSButton,SIGNAL(toggled(bool)),this,SLOT(yAxisTypeChanged()));
  yAxisXSButton->setChecked(true);
  yAxisSFButton = new QRadioButton(tr("S-Factor"));
  connect(yAxisSFButton,SIGNAL(toggled(bool)),this,SLOT(yAxisTypeChanged()));
#ifdef MACX_SPACING
  yAxisIsLogCheck = new QCheckBox(tr("Use Log Scale"));
#else
  yAxisIsLogCheck = new QCheckBox(tr("Use Logarithmic Scale"));
#endif
  connect(yAxisIsLogCheck,SIGNAL(toggled(bool)),this,SLOT(yAxisLogScaleChanged(bool)));
  yAxisIsLogCheck->setChecked(true);
  QHBoxLayout* yAxisLayout = new QHBoxLayout;
  yAxisLayout->setContentsMargins(5,5,5,5);
  yAxisLayout->addWidget(yAxisXSButton);
  yAxisLayout->addWidget(yAxisSFButton);
  yAxisLayout->addWidget(yAxisIsLogCheck);
  yAxisBox->setLayout(yAxisLayout);

  topLayout->addWidget(xAxisBox,0,0);
  topLayout->addWidget(yAxisBox,0,1);

  rightLayout->addLayout(topLayout);
  rightLayout->addWidget(azurePlot);

  dataSegmentSelectorList = new QListView;
  testSegmentSelectorList = new QListView;
  dataSegmentSelectorList->setAttribute(Qt::WA_MacShowFocusRect, 0);
  testSegmentSelectorList->setAttribute(Qt::WA_MacShowFocusRect, 0);
  dataSegmentSelectorList->setModel(segDataProxyModel);
  testSegmentSelectorList->setModel(segTestProxyModel);
  dataSegmentSelectorList->setItemDelegate(new RichTextDelegate());
  testSegmentSelectorList->setItemDelegate(new RichTextDelegate());
  dataSegmentSelectorList->setSelectionMode(QAbstractItemView::MultiSelection);
  testSegmentSelectorList->setSelectionMode(QAbstractItemView::MultiSelection);
  dataSegmentSelectorList->setResizeMode(QListView::Adjust);
  testSegmentSelectorList->setResizeMode(QListView::Adjust);

  QGroupBox *dataSegmentSelectorBox = new QGroupBox(tr("Segments From Data"));
  QGridLayout *dataSegmentSelectorLayout = new QGridLayout;
```

```
  dataSegmentSelectorLayout->setContentsMargins(5,5,5,5);
  QGroupBox *testSegmentSelectorBox = new QGroupBox(tr("Segments Without Data"));
  QGridLayout *testSegmentSelectorLayout = new QGridLayout;
  testSegmentSelectorLayout->setContentsMargins(5,5,5,5);

  dataSegmentSelectorLayout->addWidget(dataSegmentSelectorList,0,0);
  testSegmentSelectorLayout->addWidget(testSegmentSelectorList,0,0);
  dataSegmentSelectorBox->setLayout(dataSegmentSelectorLayout);
  testSegmentSelectorBox->setLayout(testSegmentSelectorLayout);

  refreshButton = new QPushButton(tr("&Draw"));
  connect(refreshButton,SIGNAL(clicked()),this,SLOT(draw()));
  exportButton = new QPushButton(tr("Export..."));
  connect(exportButton,SIGNAL(clicked()),azurePlot,SLOT(exportPlot()));
  printButton = new QPushButton(tr("Print..."));
  connect(printButton,SIGNAL(clicked()),azurePlot,SLOT(print()));
  QHBoxLayout *buttonLayout = new QHBoxLayout;
  buttonLayout->addWidget(refreshButton);
  buttonLayout->addWidget(exportButton);
  buttonLayout->addWidget(printButton);

  QVBoxLayout* leftLayout = new QVBoxLayout;
  leftLayout->addWidget(dataSegmentSelectorBox);
  leftLayout->addWidget(testSegmentSelectorBox);
  leftLayout->addLayout(buttonLayout);

  QGridLayout* mainLayout = new QGridLayout;
  mainLayout->addLayout(leftLayout,0,0);
  mainLayout->addLayout(rightLayout,0,1);
  mainLayout->setColumnStretch(1,1);

  setLayout(mainLayout);
}

QList<PlotEntry*> PlotTab::getDataSegments() {
  QList<PlotEntry*> dataSegmentPlotEntries;
  QModelIndexList indexes = dataSegmentSelectorList->selectionModel()->selectedIndexes();
  for(int i = 0; i< indexes.size(); i++) {
    int sourceRow = segDataProxyModel->mapToSource(indexes[i]).row();
    QModelIndex sourceIndex =
      segDataProxyModel->mapToSource(segDataProxyModel->index(indexes[i].row(),1,QModelIndex()));
    int entranceKey = segDataProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
    sourceIndex =
segDataProxyModel->mapToSource(segDataProxyModel->index(indexes[i].row(),2,QModelIndex()));
    int exitKey = segDataProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
    sourceIndex =
segDataProxyModel->mapToSource(segDataProxyModel->index(indexes[i].row(),7,QModelIndex()));
    int dataType = segDataProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
    QString filename = (dataType==3) ?

QString::fromStdString(configure.outputdir)+QString("AZUREOut_aa=%1_TOTAL_CAPTURE.out").arg(entranceKey)
:

QString::fromStdString(configure.outputdir)+QString("AZUREOut_aa=%1_R=%2.out").arg(entranceKey).arg(exitKey);
    int numPreviousInBlock = 0;
    for(int j =0; j<indexes[i].row(); j++) {
      sourceIndex = segDataProxyModel->mapToSource(segDataProxyModel->index(j,1,QModelIndex()));
      int previousEntranceKey =
segDataProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
      sourceIndex = segDataProxyModel->mapToSource(segDataProxyModel->index(j,2,QModelIndex()));
      int previousExitKey = segDataProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
      if(previousEntranceKey==entranceKey&&previousExitKey==exitKey) numPreviousInBlock++;
    }
    PlotEntry* newPlotEntry = new PlotEntry(0,entranceKey,exitKey,numPreviousInBlock,filename);
    dataSegmentPlotEntries.push_back(newPlotEntry);
  }
  return dataSegmentPlotEntries;
}

QList<PlotEntry*> PlotTab::getTestSegments() {
  QList<PlotEntry*> testSegmentPlotEntries;
  QModelIndexList indexes = testSegmentSelectorList->selectionModel()->selectedIndexes();
  for(int i = 0; i< indexes.size(); i++) {
    int sourceRow = segTestProxyModel->mapToSource(indexes[i]).row();
    QModelIndex sourceIndex =
      segTestProxyModel->mapToSource(segTestProxyModel->index(indexes[i].row(),1,QModelIndex()));
    int entranceKey = segTestProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
    sourceIndex =
segTestProxyModel->mapToSource(segTestProxyModel->index(indexes[i].row(),2,QModelIndex()));
    int exitKey = segTestProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
    sourceIndex =
segTestProxyModel->mapToSource(segTestProxyModel->index(indexes[i].row(),9,QModelIndex()));
    int dataType = segTestProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
    QString filename = (dataType==4) ?

QString::fromStdString(configure.outputdir)+QString("AZUREOut_aa=%1_TOTAL_CAPTURE.extrap").arg(entranceKey)
:
```

```
QString::fromStdString(configure.outputdir)+QString("AZUREOut_aa=%1_R=%2.extrap").arg(entranceKey).arg(exitKey);
    int numPreviousInBlock = 0;
    for(int j =0; j<indexes[i].row(); j++) {
      sourceIndex = segTestProxyModel->mapToSource(segTestProxyModel->index(j,1,QModelIndex()));
      int previousEntranceKey =
segTestProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
      sourceIndex = segTestProxyModel->mapToSource(segTestProxyModel->index(j,2,QModelIndex()));
      int previousExitKey = segTestProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
      if(previousEntranceKey==entranceKey&&previousExitKey==exitKey) numPreviousInBlock++;
    }
    PlotEntry* newPlotEntry = new PlotEntry(1,entranceKey,exitKey,numPreviousInBlock,filename);
    testSegmentPlotEntries.push_back(newPlotEntry);
  }
  return testSegmentPlotEntries;
}

void PlotTab::draw() {
  QList<PlotEntry*> entries = getDataSegments();
  entries.append(getTestSegments());
  azurePlot->draw(entries);
}

void PlotTab::xAxisTypeChanged() {
  azurePlot->setXAxisType(xAxisTypeCombo->currentIndex());
}

void PlotTab::yAxisTypeChanged() {
  if(yAxisXSButton->isChecked())
    azurePlot->setYAxisType(0);
  else if(yAxisSFButton->isChecked())
    azurePlot->setYAxisType(1);
}

void PlotTab::xAxisLogScaleChanged(bool checked) {
  azurePlot->setXAxisLog(checked);
}

void PlotTab::yAxisLogScaleChanged(bool checked) {
  azurePlot->setYAxisLog(checked);
}

void PlotTab::reset() {
  azurePlot->clearEntries();
  xAxisTypeCombo->setCurrentIndex(0);
  xAxisIsLogCheck->setChecked(false);
  yAxisXSButton->setChecked(true);
  yAxisIsLogCheck->setChecked(true);
}

void PlotTab::showInfo(int which,QString title) {
  if(which<infoText.size()) {
    if(!infoDialog[which]) {
      infoDialog[which] = new InfoDialog(infoText[which],this,title);
      infoDialog[which]->setAttribute(Qt:: WA_DeleteOnClose);
      infoDialog[which]->show();
    } else infoDialog[which]->raise();
  }
}
```

8.121 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RichText↩ Delegate.cpp File Reference

```
#include <QTextDocument>
#include <QPainter>
#include <QAbstractTextDocumentLayout>
#include <QApplication>
#include <QPushButton>
#include "RichTextDelegate.h"
```

8.122 RichTextDelegate.cpp

Go to the documentation of this file.

```
#include <QTextDocument>
#include <QPainter>
#include <QAbstractTextDocumentLayout>
#include <QApplication>
#include <QPushButton>

#include "RichTextDelegate.h"

void RichTextDelegate::paint(QPainter *painter, const QStyleOptionViewItem &option, const QModelIndex
&index) const {
  QTextDocument document;
  QString value = index.data(Qt::DisplayRole).toString();

  document.setHtml(value);
  QStyleOptionViewItemV4 opt(option);
  initStyleOption(&opt, index);

  document.setTextWidth(opt.rect.width());

  opt.text=QString();
  QStyle *style = opt.widget ? opt.widget->style() : QApplication::style();
  style->drawControl(QStyle::CE_ItemViewItem, &opt, painter);

  QAbstractTextDocumentLayout::PaintContext ctx;
  if (opt.state & QStyle::State_Selected) {
    if(opt.state & QStyle::State_Active)
      ctx.palette.setColor(QPalette::Text, opt.palette.color(QPalette::Active,
QPalette::HighlightedText));
    else ctx.palette.setColor(QPalette::Text, opt.palette.color(QPalette::Inactive,
QPalette::HighlightedText));
  }

  painter->translate(opt.rect.topLeft());
  document.documentLayout()->draw(painter,ctx);
  painter->translate(-opt.rect.topLeft());
}

QSize RichTextDelegate::sizeHint(const QStyleOptionViewItem & option, const QModelIndex & index )
const {
  QTextDocument document;
  QString value = index.data(Qt::DisplayRole).toString();
  document.setHtml(value);
  document.setTextWidth(option.rect.width());
  return QSize(option.rect.width(),document.size().height());
}

```

8.123 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RunTab.cpp File Reference

```
#include <QTextEdit>
#include <QComboBox>
#include <QGridLayout>
#include <QLabel>
#include <QSpacerItem>
#include <QRadioButton>
#include <QGroupBox>
#include <QFileDialog>
#include "RunTab.h"
#include "ChooseFileButton.h"
#include "FilteredTextEdit.h"
#include "InfoDialog.h"
```

8.124 RunTab.cpp

Go to the documentation of this file.

```
#include <QTextEdit>
```

```
#include <QComboBox>
#include <QGridLayout>
#include <QLabel>
#include <QSpacerItem>
#include <QRadioButton>
#include <QGroupBox>
#include <QFileDialog>

#include "RunTab.h"
#include "ChooseFileButton.h"
#include "FilteredTextEdit.h"
#include "InfoDialog.h"

RunTab::RunTab(QWidget* parent) : QWidget(parent) {
  calcType = new QComboBox;
  calcType->addItem(tr("Calculate Segments From Data"));
  calcType->addItem(tr("Fit Segments From Data"));
  calcType->addItem(tr("Calculate Segments Without Data"));
  calcType->addItem(tr("Perform MINOS Error Analysis"));
  calcType->addItem(tr("Calculate Reaction Rate"));
  connect(calcType,SIGNAL(currentIndexChanged(int)),this,SLOT(calculationTypeChanged(int)));

  chiVarianceText = new QLineEdit;
  chiVarianceText->setText("1.0");
  chiVarianceText->setEnabled(false);
  chiVarianceText->setMinimumWidth(50);
  chiVarianceText->setMaximumWidth(50);

  calcButton = new QPushButton(tr("Save and &Run"));
  stopAZUREButton = new QPushButton(tr("Stop AZURE2"));
  stopAZUREButton->setEnabled(false);

  QGridLayout* calcLayout = new QGridLayout;
  calcLayout->addWidget(new QLabel(tr("Calculation Type:")),0,0,Qt::AlignRight);
  calcLayout->addWidget(calcType,0,1);
  calcLayout->setColumnStretch(1,0);
  calcLayout->addItem(new QSpacerItem(28,28),0,2);
  calcLayout->setColumnStretch(2,1);
  calcLayout->addWidget(new QLabel(tr("Chi-Squared Variance:")),0,3,Qt::AlignRight);
  calcLayout->addWidget(chiVarianceText,0,4);
  calcLayout->setColumnStretch(4,0);
  calcLayout->addWidget(calcButton,0,5);
  calcLayout->setColumnStretch(5,0);
  calcLayout->addWidget(stopAZUREButton,0,6);
  calcLayout->setColumnStretch(6,0);

  paramFileText = new QLineEdit;
  paramFileText->setEnabled(false);
  newParamFileButton = new QRadioButton(tr("Create New Parameters File"));
  newParamFileButton->setMinimumWidth(200);
  newParamFileButton->setMaximumWidth(200);
  oldParamFileButton = new QRadioButton(tr("Use: "));
  connect(oldParamFileButton,SIGNAL(toggled(bool)),this,SLOT(paramFileButtonChanged(bool)));
  newParamFileButton->setChecked(true);

  QGroupBox* paramFileGroup = new QGroupBox(tr("Parameters File"));
  QHBoxLayout* paramFileLayout = new QHBoxLayout;
  paramFileLayout->setContentsMargins(5,5,5,5);
  paramFileLayout->addWidget(newParamFileButton);
  paramFileLayout->addWidget(oldParamFileButton);
  paramFileLayout->addWidget(paramFileText);
  paramFileChoose=new ChooseFileButton(tr("Choose..."));
  paramFileChoose->setEnabled(false);
  paramFileChoose->setLineEdit(paramFileText);
  connect(paramFileChoose,SIGNAL(clicked(QLineEdit*)),this,SLOT(setChooseFile(QLineEdit*)));
  paramFileLayout->addWidget(paramFileChoose);
  paramFileGroup->setLayout(paramFileLayout);

  integralsFileText = new QLineEdit;
  integralsFileText->setEnabled(false);
  newIntegralsFileButton = new QRadioButton(tr("Create New Integrals File"));
  newIntegralsFileButton->setMinimumWidth(200);
  newIntegralsFileButton->setMaximumWidth(200);
  oldIntegralsFileButton = new QRadioButton(tr("Use: "));
  connect(oldIntegralsFileButton,SIGNAL(toggled(bool)),this,SLOT(integralsFileButtonChanged(bool)));
  newIntegralsFileButton->setChecked(true);

  integralsFileGroup = new QGroupBox(tr("External Capture Integrals File"));
  QHBoxLayout* integralsFileLayout = new QHBoxLayout;
  integralsFileLayout->setContentsMargins(5,5,5,5);
  integralsFileLayout->addWidget(newIntegralsFileButton);
  integralsFileLayout->addWidget(oldIntegralsFileButton);
  integralsFileLayout->addWidget(integralsFileText);
  integralsFileChoose=new ChooseFileButton(tr("Choose..."));
  integralsFileChoose->setEnabled(false);
  integralsFileChoose->setLineEdit(integralsFileText);
  connect(integralsFileChoose,SIGNAL(clicked(QLineEdit*)),this,SLOT(setChooseFile(QLineEdit*)));
```

```
  integralsFileLayout->addWidget(integralsFileChoose);
  integralsFileGroup->setLayout(integralsFileLayout);

  gridTempButton = new QRadioButton(tr("Create Temperatures"));
  rateEntranceKey = new QLineEdit;
  rateEntranceKey->setMinimumWidth(50);
  rateEntranceKey->setMaximumWidth(50);
  rateExitKey = new QLineEdit;
  rateExitKey->setMinimumWidth(50);
  rateExitKey->setMaximumWidth(50);
  minTempText = new QLineEdit;
  minTempText->setMinimumWidth(50);
  minTempText->setMaximumWidth(50);
  maxTempText = new QLineEdit;
  maxTempText->setMinimumWidth(50);
  maxTempText->setMaximumWidth(50);
  tempStepText = new QLineEdit;
  tempStepText->setMinimumWidth(50);
  tempStepText->setMaximumWidth(50);
  fileTempButton = new QRadioButton(tr("Use Temperature File: "));
  connect(fileTempButton,SIGNAL(toggled(bool)),this,SLOT(fileTempButtonChanged(bool)));
  fileTempText = new QLineEdit;
  fileTempText->setEnabled(false);
  gridTempButton->setChecked(true);

  rateParamsGroup = new QGroupBox(tr("Reaction Rate Parameters"));
  rateParamsGroup->hide();
  QGridLayout* keyLayout = new QGridLayout;
  keyLayout->addWidget(new QLabel(tr("Entrance Key:")),0,0,Qt::AlignRight);
  keyLayout->addWidget(rateEntranceKey,0,1);
  keyLayout->addWidget(new QLabel(tr("Exit Key:")),0,2,Qt::AlignRight);
  keyLayout->addWidget(rateExitKey,0,3);
  keyLayout->addItem(new QSpacerItem(28,28),0,4);
  keyLayout->setColumnStretch(4,1);
  QGridLayout* gridTempLayout = new QGridLayout;
  gridTempLayout->addWidget(gridTempButton,0,0);
  gridTempLayout->setColumnStretch(0,0);
  gridTempLayout->addItem(new QSpacerItem(28,28),0,1);
  gridTempLayout->setColumnStretch(1,1);
  gridTempLayout->addWidget(new QLabel(tr("Minimum Temperature (GK):")),0,2,Qt::AlignRight);
  gridTempLayout->addWidget(minTempText,0,3);
  gridTempLayout->setColumnStretch(3,0);
  gridTempLayout->addWidget(new QLabel(tr("Maximum Temperature (GK):")),0,4,Qt::AlignRight);
  gridTempLayout->addWidget(maxTempText,0,5);
  gridTempLayout->setColumnStretch(5,0);
  gridTempLayout->addWidget(new QLabel(tr("Temperature Step (GK):")),0,6,Qt::AlignRight);
  gridTempLayout->addWidget(tempStepText,0,7);
  gridTempLayout->setColumnStretch(7,0);
  QHBoxLayout *fileTempLayout = new QHBoxLayout;
  fileTempLayout->addWidget(fileTempButton);
  fileTempLayout->addWidget(fileTempText);
  rateParamsChoose=new ChooseFileButton(tr("Choose..."));
  rateParamsChoose->setEnabled(false);
  rateParamsChoose->setLineEdit(fileTempText);
  connect(rateParamsChoose,SIGNAL(clicked(QLineEdit*)),this,SLOT(setChooseFile(QLineEdit*)));
  fileTempLayout->addWidget(rateParamsChoose);

  QVBoxLayout* rateParamsLayout = new QVBoxLayout;
  rateParamsLayout->setContentsMargins(5,5,5,5);
  rateParamsLayout->addLayout(keyLayout);
  rateParamsLayout->addLayout(gridTempLayout);
  rateParamsLayout->addLayout(fileTempLayout);
  rateParamsGroup->setLayout(rateParamsLayout);

  runtimeText = new FilteredTextEdit;
  runtimeText->setReadOnly(true);
  runtimeText->setAcceptRichText(false);

  QVBoxLayout* mainLayout = new QVBoxLayout;
  mainLayout->addLayout(calcLayout);
  mainLayout->addWidget(paramFileGroup);
  mainLayout->addWidget(integralsFileGroup);
  mainLayout->addWidget(rateParamsGroup);
  mainLayout->addWidget(runtimeText);
  setLayout(mainLayout);
}


void RunTab::calculationTypeChanged(int index) {
  if(index==4) {
    integralsFileGroup->hide();
    rateParamsGroup->show();
  } else {
    rateParamsGroup->hide();
    integralsFileGroup->show();
  }
  if(index==3) chiVarianceText->setEnabled(true);
```

```
  else {
    chiVarianceText->setText("1.0");
    chiVarianceText->setEnabled(false);
  }
}

void RunTab::paramFileButtonChanged(bool checked) {
  if(checked) {
    paramFileText->setEnabled(true);
    paramFileChoose->setEnabled(true);
  } else {
    paramFileText->setEnabled(false);
    paramFileChoose->setEnabled(false);
  }
}

void RunTab::integralsFileButtonChanged(bool checked) {
  if(checked) {
    integralsFileText->setEnabled(true);
    integralsFileChoose->setEnabled(true);
  } else {
    integralsFileText->setEnabled(false);
    integralsFileChoose->setEnabled(false);
  }
}

void RunTab::fileTempButtonChanged(bool checked) {
  if(checked) {
    fileTempText->setEnabled(true);
    rateParamsChoose->setEnabled(true);
    minTempText->setEnabled(false);
    maxTempText->setEnabled(false);
    tempStepText->setEnabled(false);
  } else {
    fileTempText->setEnabled(false);
    rateParamsChoose->setEnabled(false);
    minTempText->setEnabled(true);
    maxTempText->setEnabled(true);
    tempStepText->setEnabled(true);
  }
}

void RunTab::setChooseFile(QLineEdit *lineEdit) {
  QString filename = QFileDialog::getOpenFileName(this);
  if(!filename.isEmpty()) {
    lineEdit->setText(QDir::fromNativeSeparators(filename));
  }
}

void RunTab::reset() {
  calcType->setCurrentIndex(0);
  newParamFileButton->setChecked(true);
  paramFileText->setText("");
  newIntegralsFileButton->setChecked(true);
  integralsFileText->setText("");
  rateEntranceKey->setText("");
  rateExitKey->setText("");
  gridTempButton->setChecked(true);
  minTempText->setText("");
  maxTempText->setText("");
  tempStepText->setText("");
  fileTempText->setText("");
  runtimeText->clear();
}

void RunTab::showInfo(int which, QString title) {
  if(which<infoText.size()) {
    if(!infoDialog[which]) {
      infoDialog[which] = new InfoDialog(infoText[which],this,title);
      infoDialog[which]->setAttribute(Qt:: WA_DeleteOnClose);
      infoDialog[which]->show();
    } else infoDialog[which]->raise();
  }
}
```

8.125 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsDataModel.cpp File Reference

```
#include "SegmentsDataModel.h"
#include "PairsModel.h"
```

```
#include <QColor>
```

8.126 SegmentsDataModel.cpp

Go to the documentation of this file.

```
#include "SegmentsDataModel.h"
#include "PairsModel.h"
#include <QColor>

SegmentsDataModel::SegmentsDataModel(QObject *parent) : QAbstractTableModel(parent) {
}

int SegmentsDataModel::rowCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return segDataLineList.size();
}

int SegmentsDataModel::columnCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return SegmentsDataData::SIZE;
}

QVariant SegmentsDataModel::data(const QModelIndex &index, int role) const {
  if(!index.isValid()) return QVariant();

  if(index.row() >= segDataLineList.size() || index.row() < 0)  return QVariant();

  if (role == Qt::DisplayRole) {
    SegmentsDataData line = segDataLineList.at(index.row());
    if(index.column() == 1) {
      if(line.dataType==3) {
    int i = 0;
    QList<PairsData> pairsList = pairsModel->getPairs();
    for(i=0;i<pairsList.size();i++)
      if(pairsList[i].pairType==10) break;
    if(pairsList.size()>=line.entrancePairIndex&&i<pairsList.size()) {
      PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
      return QString("<center>%1</center>").arg(pairsModel->getReactionLabelTotalCapture(firstPair));
    } else return QString("<center><font
style='color:red;font-weight:bold;'>UNDEFINED</font></center>");
      } else {
if(pairsModel->getPairs().size()>=line.entrancePairIndex&&pairsModel->getPairs().size()>=line.exitPairIndex)
{
      PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
      PairsData secondPair=pairsModel->getPairs().at(line.exitPairIndex-1);
      return QString("<center>%1</center>").arg(pairsModel->getReactionLabel(firstPair,secondPair));
    } else return QString("<center><font
style='color:red;font-weight:bold;'>UNDEFINED</font></center>");
      }
    } else if(index.column() == 2) return QVariant();
    else if(index.column() == 3) {
      if(line.lowEnergy==line.highEnergy) return line.lowEnergy;
      else return QString("%1-%2").arg(line.lowEnergy).arg(line.highEnergy);
    } else if(index.column() == 4) return QVariant();
    else if(index.column() == 5) {
      if(line.lowAngle==line.highAngle) return line.lowAngle;
      else return QString("%1-%2").arg(line.lowAngle).arg(line.highAngle);
    } else if(index.column() == 6) return QVariant();
    else if(index.column() == 7) {
      if(line.dataType==2) {
    QChar orbital;
    switch (line.phaseL) {
    case 0:
      orbital='s';
      break;
    case 1:
      orbital='p';
      break;
    case 2:
      orbital='d';
      break;
    case 3:
      orbital='f';
      break;
    case 4:
      orbital='g';
      break;
    case 5:
      orbital='h';
```

```
      break;
    case 6:
      orbital='i';
      break;
    default:
      orbital='?';
    }
    QString tempSpin;
    if(((int)(line.phaseJ*2))%2!=0&&line.phaseJ!=0.) tempSpin=QString("%1/2").arg((int)(line.phaseJ*2));
    else tempSpin=QString("%1").arg(line.phaseJ);
    return QString("<center>Phase Shift [%1<sub>%2</sub>]</center>").arg(orbital).arg(tempSpin);
      } else if(line.dataType==1) return QString(tr("<center>Differential</center>"));
      else if(line.dataType==3) return QString(tr("<center>Total Capture</center>"));
      else return QString(tr("<center>Angle Integrated</center>"));
    } else if(index.column() == 8) return line.dataFile;
    else if(index.column() == 9) {
      if(line.varyNorm==1) return QString("<center><font style='color:red;font-weight:bold;'>%1</font></center>").arg(line.dataNorm,0,'g',2);
      else return QString("<center>%1</center>").arg(line.dataNorm,0,'g',2);
    }  else if(index.column() == 10) return QVariant();
    else if(index.column() == 11) {
      if(line.varyNorm==1) return QString(tr("YES"));
      else return QString(tr("NO"));
    } else if(index.column() == 12) {
      return QVariant();
    } else if(index.column() == 13) {
      return QVariant();
    }
  } else if (role == Qt::EditRole) {
    SegmentsDataData line = segDataLineList.at(index.row());
    if(index.column() == 1) return line.entrancePairIndex;
    else if(index.column() == 2) return line.exitPairIndex;
    else if(index.column() == 3) return line.lowEnergy;
    else if(index.column() == 4) return line.highEnergy;
    else if(index.column() == 5) return line.lowAngle;
    else if(index.column() == 6) return line.highAngle;
    else if(index.column() == 7) return line.dataType;
    else if(index.column() == 8) return line.dataFile;
    else if(index.column() == 9) return line.dataNorm;
    else if(index.column() == 10) return line.dataNormError;
    else if(index.column() == 11) return line.varyNorm;
    else if(index.column() == 12) return line.phaseJ;
    else if(index.column() == 13) return line.phaseL;
  } else if (role==Qt::CheckStateRole && index.column()==0) {
    SegmentsDataData line = segDataLineList.at(index.row());
    if(line.isActive==1) return Qt::Checked;
    else return Qt::Unchecked;
  } else if(role == Qt::TextAlignmentRole) return Qt::AlignCenter;

  return QVariant();
}

QVariant SegmentsDataModel::headerData(int section, Qt::Orientation orientation, int role) const {
  if(role!= Qt::DisplayRole) return QVariant();
  if(orientation == Qt::Horizontal) {
    switch(section) {
    case 0:
      return tr("");
    case 1:
      return tr("Reaction");
    case 2:
      return QVariant();
    case 3:
      return tr("Energy\nRange");
    case 4:
      return QVariant();
    case 5:
      return tr("Angle\nRange");
    case 6:
      return QVariant();
    case 7:
      return tr("Data Type");
    case 8:
      return tr("Data File");
    case 9:
      return tr("Data\nNorm.");
    case 10:
      return tr("Data\nNorm. Err.");
    case 11:
      return tr("Vary\nNorm.?");
    case 12:
      return QVariant();
    case 13:
      return QVariant();
    default:
      return QVariant();
```

```
    }
  } else if(orientation == Qt::Vertical) {
    return section+1;
  }
  return QVariant();
}

bool SegmentsDataModel::setData(const QModelIndex &index, const QVariant &value, int role) {
  if (index.isValid() && role == Qt::EditRole ) {
    int row = index.row();
    SegmentsDataData tempData = segDataLineList.value(row);
    if(index.column() == 0) tempData.isActive=value.toInt();
    else if(index.column() == 1) tempData.entrancePairIndex=value.toInt();
    else if(index.column() == 2) tempData.exitPairIndex=value.toInt();
    else if(index.column() == 3) tempData.lowEnergy=value.toDouble();
    else if(index.column() == 4) tempData.highEnergy=value.toDouble();
    else if(index.column() == 5) tempData.lowAngle=value.toDouble();
    else if(index.column() == 6) tempData.highAngle=value.toDouble();
    else if(index.column() == 7) tempData.dataType=value.toInt();
    else if(index.column() == 8) tempData.dataFile=value.toString();
    else if(index.column() == 9) tempData.dataNorm=value.toDouble();
    else if(index.column() == 10) tempData.dataNormError=value.toDouble();
    else if(index.column() == 11) tempData.varyNorm=value.toInt();
    else if(index.column() == 12) tempData.phaseJ=value.toDouble();
    else if(index.column() == 13) tempData.phaseL=value.toInt();
    else return false;

    segDataLineList.replace(row,tempData);
    emit(dataChanged(index,index));
    return true;
  } else if(role== Qt::CheckStateRole) {
    int row = index.row();
    SegmentsDataData tempData = segDataLineList.value(row);
    if(index.column()==0) {
      if(value==Qt::Checked) tempData.isActive=1;
      else tempData.isActive=0;
    } else return false;
    segDataLineList.replace(row,tempData);
    emit(dataChanged(index,index));
    return true;
  }
  return false;
}

bool SegmentsDataModel::insertRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginInsertRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows; row++) {
      SegmentsDataData tempData;
      segDataLineList.insert(position,tempData);
    }
    endInsertRows();
  }
  return true;
}

bool SegmentsDataModel::removeRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginRemoveRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows;++row) {
      segDataLineList.removeAt(position);
    }
    endRemoveRows();
  }
  return true;
}

Qt::ItemFlags SegmentsDataModel::flags(const QModelIndex &index) const {
  if (!index.isValid()) return Qt::ItemIsEnabled;
  if(index.column()==0) return QAbstractTableModel::flags(index) | Qt::ItemIsUserCheckable;
  return QAbstractTableModel::flags(index);
}

int SegmentsDataModel::isSegDataLine(const SegmentsDataData &line) const {
  int foundLine=-1;
  for(int i=0;i<segDataLineList.size();i++) {
    SegmentsDataData tempLine=segDataLineList.value(i);
    if(tempLine.entrancePairIndex==line.entrancePairIndex&&
       tempLine.exitPairIndex==line.exitPairIndex&&
       tempLine.lowEnergy==line.lowEnergy&&
       tempLine.highEnergy==line.highEnergy&&
       tempLine.lowAngle==line.lowAngle&&
       tempLine.highAngle==line.highAngle&&
       tempLine.dataType==line.dataType&&
       tempLine.dataFile==line.dataFile&&
```

```
        tempLine.dataNorm==line.dataNorm&&
        tempLine.dataNormError==line.dataNormError&&
        tempLine.varyNorm==line.varyNorm&&
        tempLine.phaseJ==line.phaseJ&&
        tempLine.phaseL==line.phaseL) {
       foundLine=i;
       break;
     }
   }
   return foundLine;
 }

 void SegmentsDataModel::setPairsModel(PairsModel* model) {
   pairsModel=model;
 }

 QString SegmentsDataModel::getReactionLabel(const QModelIndex &index) {
   SegmentsDataData line = segDataLineList.at(index.row());
   if(line.dataType==3) {
     int i = 0;
     QList<PairsData> pairsList = pairsModel->getPairs();
     for(i=0;i<pairsList.size();i++)
       if(pairsList[i].pairType==10) break;
     if(pairsList.size()>=line.entrancePairIndex&&i<pairsList.size()) {
       PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
       return pairsModel->getReactionLabelTotalCapture(firstPair);
     }
     return QString("<font style='color:red;font-weight:bold;'>UNDEFINED</font>");
   } else {
     int numPairs = pairsModel->getPairs().size();
     if(line.entrancePairIndex-1>=numPairs ||
        line.exitPairIndex-1>=numPairs) return QString("<font
 style='color:red;font-weight:bold'>UNDEFINED</font>");
     PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
     PairsData secondPair=pairsModel->getPairs().at(line.exitPairIndex-1);
     return pairsModel->getReactionLabel(firstPair,secondPair);
   }
 }
```

8.127 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTab.cpp File Reference

```
#include <QGridLayout>
#include <QMessageBox>
#include <QGroupBox>
#include <QHeaderView>
#include <QTextStream>
#include "SegmentsTab.h"
#include "RichTextDelegate.h"
#include "InfoDialog.h"
```

8.128 SegmentsTab.cpp

Go to the documentation of this file.

```
#include <QGridLayout>
#include <QMessageBox>
#include <QGridLayout>
#include <QGroupBox>
#include <QHeaderView>
#include <QTextStream>

#include "SegmentsTab.h"
#include "RichTextDelegate.h"
#include "InfoDialog.h"

SegmentsTab::SegmentsTab(QWidget *parent) : QWidget(parent) {
  segmentsDataModel = new SegmentsDataModel;
  segmentsDataView = new QTableView;
  segmentsDataView->setModel(segmentsDataModel);
```

```
RichTextDelegate *rt = new RichTextDelegate();
segmentsDataView->setItemDelegateForColumn(1,rt);
segmentsDataView->setColumnHidden(2,true);
segmentsDataView->setColumnHidden(4,true);
segmentsDataView->setColumnHidden(6,true);
segmentsDataView->horizontalHeader()->setStretchLastSection(true);
segmentsDataView->horizontalHeader()->setHighlightSections(false);
segmentsDataView->setSelectionBehavior(QAbstractItemView::SelectRows);
segmentsDataView->setSelectionMode(QAbstractItemView::SingleSelection);
segmentsDataView->setEditTriggers(QAbstractItemView::NoEditTriggers);
segmentsDataView->setShowGrid(false);
segmentsDataView->setColumnWidth(0,27);
segmentsDataView->horizontalHeader()->setSectionResizeMode(0,QHeaderView::Fixed);
segmentsDataView->horizontalHeader()->setSectionResizeMode(1,QHeaderView::Fixed);
segmentsDataView->horizontalHeader()->setSectionResizeMode(2,QHeaderView::Fixed);
for(int i = 1; i<3;i++) segmentsDataView->setColumnWidth(i,200);
for(int i = 3; i<7;i++) segmentsDataView->setColumnWidth(i,120);
segmentsDataView->setColumnWidth(7,140);
segmentsDataView->setItemDelegateForColumn(7,rt);
segmentsDataView->setColumnWidth(8,220);
segmentsDataView->setItemDelegateForColumn(9,rt);
segmentsDataView->setColumnHidden(10,true);
segmentsDataView->setColumnHidden(11,true);
segmentsDataView->setColumnHidden(12,true);
segmentsDataView->setColumnHidden(13,true);

connect(segmentsDataView->selectionModel(),SIGNAL(selectionChanged(QItemSelection,QItemSelection)),this,SLOT(updateSegDa
connect(segmentsDataView,SIGNAL(doubleClicked(QModelIndex)),this,SLOT(editSegDataLine()));

segmentsTestModel = new SegmentsTestModel;
segmentsTestView = new QTableView;
segmentsTestView->setModel(segmentsTestModel);
segmentsTestView->setItemDelegateForColumn(1,rt);
segmentsTestView->setColumnHidden(2,true);
segmentsTestView->setColumnHidden(4,true);
segmentsTestView->setColumnHidden(7,true);
segmentsTestView->horizontalHeader()->setStretchLastSection(true);
segmentsTestView->horizontalHeader()->setHighlightSections(false);
segmentsTestView->setSelectionBehavior(QAbstractItemView::SelectRows);
segmentsTestView->setSelectionMode(QAbstractItemView::SingleSelection);
segmentsTestView->setEditTriggers(QAbstractItemView::NoEditTriggers);
segmentsTestView->setShowGrid(false);
segmentsTestView->setColumnWidth(0,27);
segmentsTestView->horizontalHeader()->setSectionResizeMode(0,QHeaderView::Fixed);
segmentsTestView->horizontalHeader()->setSectionResizeMode(1,QHeaderView::Fixed);
segmentsTestView->horizontalHeader()->setSectionResizeMode(2,QHeaderView::Fixed);
for(int i = 1; i<3;i++) segmentsTestView->setColumnWidth(i,200);
segmentsTestView->setColumnWidth(3,120);
segmentsTestView->setColumnWidth(5,90);
segmentsTestView->setColumnWidth(6,120);
for(int i = 8; i<SegmentsTestData::SIZE;i++) segmentsTestView->setColumnWidth(i,90);
segmentsTestView->setItemDelegateForColumn(9,rt);
segmentsTestView->setColumnHidden(10,true);
segmentsTestView->setColumnHidden(11,true);
segmentsTestView->setColumnHidden(12,true);

connect(segmentsTestView->selectionModel(),SIGNAL(selectionChanged(QItemSelection,QItemSelection)),this,SLOT(updateSegTe
connect(segmentsTestView,SIGNAL(doubleClicked(QModelIndex)),this,SLOT(editSegTestLine()));

//segDataAddButton=new QPushButton(tr("Add Line"));
//segDataDeleteButton = new QPushButton(tr("Delete Line"));
segDataAddButton=new QPushButton(tr("+"));
segDataAddButton->setMaximumSize(28,28);
segDataDeleteButton = new QPushButton(tr("-"));
segDataDeleteButton->setEnabled(false);
segDataDeleteButton->setMaximumSize(28,28);
segDataUpButton = new QPushButton;
segDataUpButton->setIcon(style()->standardIcon(QStyle::SP_ArrowUp));
segDataUpButton->setEnabled(false);
segDataUpButton->setMaximumSize(28,28);
segDataDownButton = new QPushButton;
segDataDownButton->setIcon(style()->standardIcon(QStyle::SP_ArrowDown));
segDataDownButton->setEnabled(false);
segDataDownButton->setMaximumSize(28,28);
connect(segDataAddButton,SIGNAL(clicked()),this,SLOT(addSegDataLine()));
connect(segDataDeleteButton,SIGNAL(clicked()),this,SLOT(deleteSegDataLine()));
connect(segDataUpButton,SIGNAL(clicked()),this,SLOT(moveSegDataLineUp()));
connect(segDataDownButton,SIGNAL(clicked()),this,SLOT(moveSegDataLineDown()));

segTestAddButton=new QPushButton(tr("+"));
segTestAddButton->setMaximumSize(28,28);
segTestDeleteButton = new QPushButton(tr("-"));
segTestDeleteButton->setMaximumSize(28,28);
segTestDeleteButton->setEnabled(false);
segTestUpButton = new QPushButton;
segTestUpButton->setIcon(style()->standardIcon(QStyle::SP_ArrowUp));
segTestUpButton->setEnabled(false);
```

```
  segTestUpButton->setMaximumSize(28,28);
  segTestDownButton = new QPushButton;
  segTestDownButton->setIcon(style()->standardIcon(QStyle::SP_ArrowDown));
  segTestDownButton->setEnabled(false);
  segTestDownButton->setMaximumSize(28,28);
  connect(segTestAddButton,SIGNAL(clicked()),this,SLOT(addSegTestLine()));
  connect(segTestDeleteButton,SIGNAL(clicked()),this,SLOT(deleteSegTestLine()));
  connect(segTestUpButton,SIGNAL(clicked()),this,SLOT(moveSegTestLineUp()));
  connect(segTestDownButton,SIGNAL(clicked()),this,SLOT(moveSegTestLineDown()));

  QGroupBox *segDataBox = new QGroupBox(tr("Segments From Data"));
  QGridLayout *segDataLayout = new QGridLayout;
  segDataLayout->addWidget(segmentsDataView,0,0);
  QGridLayout *segDataButtonBox = new QGridLayout;
  segDataButtonBox->addWidget(segDataAddButton,0,0);
  segDataButtonBox->addWidget(segDataDeleteButton,0,1);
  segDataButtonBox->addItem(new QSpacerItem(28,28),0,2);
  segDataButtonBox->addWidget(segDataUpButton,0,3);
  segDataButtonBox->addWidget(segDataDownButton,0,4);
  segDataButtonBox->setColumnStretch(0,0);
  segDataButtonBox->setColumnStretch(1,0);
  segDataButtonBox->setColumnStretch(2,1);
  segDataButtonBox->setColumnStretch(3,0);
  segDataButtonBox->setColumnStretch(4,0);
#ifdef MACX_SPACING
  segDataButtonBox->setHorizontalSpacing(11);
#else
  segDataButtonBox->setHorizontalSpacing(0);
#endif
  segDataLayout->addLayout(segDataButtonBox,1,0);
  segDataBox->setLayout(segDataLayout);

  QGroupBox *segTestBox = new QGroupBox(tr("Segments Without Data"));
  QGridLayout *segTestLayout = new QGridLayout;
  segTestLayout->addWidget(segmentsTestView,0,0);
  QGridLayout *segTestButtonBox = new QGridLayout;
  segTestButtonBox->addWidget(segTestAddButton,0,0);
  segTestButtonBox->addWidget(segTestDeleteButton,0,1);
  segTestButtonBox->addItem(new QSpacerItem(28,28),0,2);
  segTestButtonBox->addWidget(segTestUpButton,0,3);
  segTestButtonBox->addWidget(segTestDownButton,0,4);
  segTestButtonBox->setColumnStretch(0,0);
  segTestButtonBox->setColumnStretch(1,0);
  segTestButtonBox->setColumnStretch(2,1);
  segTestButtonBox->setColumnStretch(3,0);
  segTestButtonBox->setColumnStretch(4,0);
#ifdef MACX_SPACING
  segTestButtonBox->setHorizontalSpacing(11);
#else
  segTestButtonBox->setHorizontalSpacing(0);
#endif
  segTestLayout->addLayout(segTestButtonBox,1,0);
  segTestBox->setLayout(segTestLayout);

  QGridLayout *mainLayout= new QGridLayout;
  mainLayout->addWidget(segDataBox,0,0);
  mainLayout->addWidget(segTestBox,1,0);

  setLayout(mainLayout);
}

SegmentsTestModel* SegmentsTab::getSegmentsTestModel() {
  return segmentsTestModel;
}

SegmentsDataModel* SegmentsTab::getSegmentsDataModel() {
  return segmentsDataModel;
}

void SegmentsTab::deleteSegDataLine() {
  QItemSelectionModel *selectionModel = segmentsDataView->selectionModel();
  QModelIndexList indexes = selectionModel->selectedRows();
  QModelIndex index=indexes.at(0);

  segmentsDataModel->removeRows(index.row(),1,QModelIndex());
  updateSegDataButtons(selectionModel->selection());
}

void SegmentsTab::deleteSegTestLine() {
  QItemSelectionModel *selectionModel = segmentsTestView->selectionModel();
  QModelIndexList indexes = selectionModel->selectedRows();
  QModelIndex index=indexes.at(0);

  segmentsTestModel->removeRows(index.row(),1,QModelIndex());
  updateSegTestButtons(selectionModel->selection());
}

```

```
void SegmentsTab::addSegDataLine() {
  AddSegDataDialog aDialog;
  if(aDialog.exec()) {
    SegmentsDataData newLine;
    newLine.isActive=1;
    newLine.entrancePairIndex=aDialog.entrancePairIndexSpin->value();
    if(aDialog.dataTypeCombo->currentIndex()==3)
      newLine.exitPairIndex=-1;
    else newLine.exitPairIndex=aDialog.exitPairIndexSpin->value();
    newLine.lowEnergy=aDialog.lowEnergyText->text().toDouble();
    newLine.highEnergy=aDialog.highEnergyText->text().toDouble();
    newLine.lowAngle=aDialog.lowAngleText->text().toDouble();
    newLine.highAngle=aDialog.highAngleText->text().toDouble();
    newLine.dataType=aDialog.dataTypeCombo->currentIndex();
    newLine.dataFile=aDialog.dataFileText->text();
    newLine.dataNorm=aDialog.dataNormText->text().toDouble();
    newLine.dataNormError=aDialog.dataNormErrorText->text().toDouble();
    if(aDialog.varyNormCheck->isChecked()) newLine.varyNorm=1;
    else newLine.varyNorm=0;
    newLine.phaseJ=aDialog.phaseJValueText->text().toDouble();
    newLine.phaseL=aDialog.phaseLValueText->text().toInt();
    addSegDataLine(newLine);
  }
}

void SegmentsTab::addSegDataLine(SegmentsDataData line) {
  QList<SegmentsDataData> lines = segmentsDataModel->getLines();
  if(segmentsDataModel->isSegDataLine(line)==-1) {
    segmentsDataModel->insertRows(lines.size(),1,QModelIndex());
    QModelIndex index = segmentsDataModel->index(lines.size(),0,QModelIndex());
    segmentsDataModel->setData(index,line.isActive,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),1,QModelIndex());
    segmentsDataModel->setData(index,line.entrancePairIndex,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),2,QModelIndex());
    segmentsDataModel->setData(index,line.exitPairIndex,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),3,QModelIndex());
    segmentsDataModel->setData(index,line.lowEnergy,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),4,QModelIndex());
    segmentsDataModel->setData(index,line.highEnergy,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),5,QModelIndex());
    segmentsDataModel->setData(index,line.lowAngle,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),6,QModelIndex());
    segmentsDataModel->setData(index,line.highAngle,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),7,QModelIndex());
    segmentsDataModel->setData(index,line.dataType,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),8,QModelIndex());
    segmentsDataModel->setData(index,line.dataFile,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),9,QModelIndex());
    segmentsDataModel->setData(index,line.dataNorm,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),10,QModelIndex());
    segmentsDataModel->setData(index,line.dataNormError,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),11,QModelIndex());
    segmentsDataModel->setData(index,line.varyNorm,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),12,QModelIndex());
    segmentsDataModel->setData(index,line.phaseJ,Qt::EditRole);
    index = segmentsDataModel->index(lines.size(),13,QModelIndex());
    segmentsDataModel->setData(index,line.phaseL,Qt::EditRole);
    segmentsDataView->resizeRowToContents(lines.size());
    updateSegDataButtons(segmentsDataView->selectionModel()->selection());
  } else {
    QMessageBox::information(this,tr("Duplicate Line"),tr("This line already exists."));
  }
}

void SegmentsTab::addSegTestLine() {
  AddSegTestDialog aDialog;
  if(aDialog.exec()) {
    SegmentsTestData newLine;
    newLine.isActive=1;
    newLine.entrancePairIndex=aDialog.entrancePairIndexSpin->value();
    if(aDialog.dataTypeCombo->currentIndex()==4)
      newLine.exitPairIndex=-1;
    else newLine.exitPairIndex=aDialog.exitPairIndexSpin->value();
    newLine.lowEnergy=aDialog.lowEnergyText->text().toDouble();
    newLine.highEnergy=aDialog.highEnergyText->text().toDouble();
    newLine.energyStep=aDialog.energyStepText->text().toDouble();
    newLine.lowAngle=aDialog.lowAngleText->text().toDouble();
    newLine.highAngle=aDialog.highAngleText->text().toDouble();
    newLine.angleStep=aDialog.angleStepText->text().toDouble();
    newLine.dataType=aDialog.dataTypeCombo->currentIndex();
    newLine.phaseJ=aDialog.phaseJValueText->text().toDouble();
    newLine.phaseL=aDialog.phaseLValueText->text().toInt();
    newLine.maxAngDistOrder=aDialog.angDistSpin->value();
    addSegTestLine(newLine);
   }
}

```

```
void SegmentsTab::addSegTestLine(SegmentsTestData line) {
  QList<SegmentsTestData> lines = segmentsTestModel->getLines();
  if(segmentsTestModel->isSegTestLine(line)==-1) {
    segmentsTestModel->insertRows(lines.size(),1,QModelIndex());
    QModelIndex index = segmentsTestModel->index(lines.size(),0,QModelIndex());
    segmentsTestModel->setData(index,line.isActive,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),1,QModelIndex());
    segmentsTestModel->setData(index,line.entrancePairIndex,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),2,QModelIndex());
    segmentsTestModel->setData(index,line.exitPairIndex,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),3,QModelIndex());
    segmentsTestModel->setData(index,line.lowEnergy,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),4,QModelIndex());
    segmentsTestModel->setData(index,line.highEnergy,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),5,QModelIndex());
    segmentsTestModel->setData(index,line.energyStep,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),6,QModelIndex());
    segmentsTestModel->setData(index,line.lowAngle,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),7,QModelIndex());
    segmentsTestModel->setData(index,line.highAngle,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),8,QModelIndex());
    segmentsTestModel->setData(index,line.angleStep,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),9,QModelIndex());
    segmentsTestModel->setData(index,line.dataType,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),10,QModelIndex());
    segmentsTestModel->setData(index,line.phaseJ,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),11,QModelIndex());
    segmentsTestModel->setData(index,line.phaseL,Qt::EditRole);
    index = segmentsTestModel->index(lines.size(),12,QModelIndex());
    segmentsTestModel->setData(index,line.maxAngDistOrder,Qt::EditRole);
    segmentsTestView->resizeRowToContents(lines.size());
    updateSegTestButtons(segmentsTestView->selectionModel()->selection());
  } else {
    QMessageBox::information(this,tr("Duplicate Line"),tr("This line already exists."));
  }
}

void SegmentsTab::editSegDataLine() {
  QItemSelectionModel *selectionModel = segmentsDataView->selectionModel();
  QModelIndexList indexes = selectionModel->selectedRows();
  QModelIndex index=indexes[0];

  QModelIndex i=segmentsDataModel->index(index.row(),1,QModelIndex());
  QVariant var=segmentsDataModel->data(i,Qt::EditRole);
  int entrancePairIndex=var.toInt();
  i=segmentsDataModel->index(index.row(),2,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  int exitPairIndex=var.toInt();
  i=segmentsDataModel->index(index.row(),3,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  QString lowEnergy=var.toString();
  i=segmentsDataModel->index(index.row(),4,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  QString highEnergy=var.toString();
  i=segmentsDataModel->index(index.row(),5,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  QString lowAngle=var.toString();
  i=segmentsDataModel->index(index.row(),6,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  QString highAngle=var.toString();
  i=segmentsDataModel->index(index.row(),7,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  int dataType=var.toInt();
  i=segmentsDataModel->index(index.row(),8,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  QString dataFile=var.toString();
  i=segmentsDataModel->index(index.row(),9,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  QString dataNorm=var.toString();
  i=segmentsDataModel->index(index.row(),10,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  QString dataNormError=var.toString();
  i=segmentsDataModel->index(index.row(),11,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  int varyNorm=var.toInt();
  i=segmentsDataModel->index(index.row(),12,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  QString phaseJ=var.toString();
  i=segmentsDataModel->index(index.row(),13,QModelIndex());
  var=segmentsDataModel->data(i,Qt::EditRole);
  QString phaseL=var.toString();


  AddSegDataDialog aDialog;
  aDialog.setWindowTitle(tr("Edit a Segment From Data"));
  aDialog.entrancePairIndexSpin->setValue(entrancePairIndex);
  aDialog.exitPairIndexSpin->setValue(exitPairIndex);
```

```
  aDialog.lowEnergyText->setText(lowEnergy);
  aDialog.highEnergyText->setText(highEnergy);
  aDialog.lowAngleText->setText(lowAngle);
  aDialog.highAngleText->setText(highAngle);
  aDialog.dataTypeCombo->setCurrentIndex(dataType);
  aDialog.dataFileText->setText(dataFile);
  aDialog.dataNormText->setText(dataNorm);
  aDialog.dataNormErrorText->setText(dataNormError);
  if(varyNorm==1) aDialog.varyNormCheck->setChecked(true);
  else aDialog.varyNormCheck->setChecked(false);
  aDialog.phaseJValueText->setText(phaseJ);
  aDialog.phaseLValueText->setText(phaseL);

  if(aDialog.exec()) {
    int newEntrancePairIndex=aDialog.entrancePairIndexSpin->value();
    if(newEntrancePairIndex!=entrancePairIndex) {
      i=segmentsDataModel->index(index.row(),1,QModelIndex());
      segmentsDataModel->setData(i,newEntrancePairIndex,Qt::EditRole);
    }
    int newExitPairIndex= (aDialog.dataTypeCombo->currentIndex()==3) ? -1 :
      aDialog.exitPairIndexSpin->value();
    if(newExitPairIndex!=exitPairIndex) {
      i=segmentsDataModel->index(index.row(),2,QModelIndex());
      segmentsDataModel->setData(i,newExitPairIndex,Qt::EditRole);
    }
    QString newLowEnergy=aDialog.lowEnergyText->text();
    if(newLowEnergy!=lowEnergy) {
      i=segmentsDataModel->index(index.row(),3,QModelIndex());
      segmentsDataModel->setData(i,newLowEnergy,Qt::EditRole);
    }
    QString newHighEnergy=aDialog.highEnergyText->text();
    if(newHighEnergy!=highEnergy) {
      i=segmentsDataModel->index(index.row(),4,QModelIndex());
      segmentsDataModel->setData(i,newHighEnergy,Qt::EditRole);
    }
    QString newLowAngle=aDialog.lowAngleText->text();
    if(newLowAngle!=lowAngle) {
      i=segmentsDataModel->index(index.row(),5,QModelIndex());
      segmentsDataModel->setData(i,newLowAngle,Qt::EditRole);
    }
    QString newHighAngle=aDialog.highAngleText->text();
    if(newHighAngle!=highAngle) {
      i=segmentsDataModel->index(index.row(),6,QModelIndex());
      segmentsDataModel->setData(i,newHighAngle,Qt::EditRole);
    }
    int newDataType=aDialog.dataTypeCombo->currentIndex();
    if(newDataType!=dataType) {
      i=segmentsDataModel->index(index.row(),7,QModelIndex());
      segmentsDataModel->setData(i,newDataType,Qt::EditRole);
    }
    QString newDataFile=aDialog.dataFileText->text();
    if(newDataFile!=dataFile) {
      i=segmentsDataModel->index(index.row(),8,QModelIndex());
      segmentsDataModel->setData(i,newDataFile,Qt::EditRole);
    }
    QString newDataNorm=aDialog.dataNormText->text();
    if(newDataNorm!=dataNorm) {
      i=segmentsDataModel->index(index.row(),9,QModelIndex());
      segmentsDataModel->setData(i,newDataNorm,Qt::EditRole);
    }
     QString newDataNormError=aDialog.dataNormErrorText->text();
    if(newDataNormError!=dataNormError) {
      i=segmentsDataModel->index(index.row(),10,QModelIndex());
      segmentsDataModel->setData(i,newDataNormError,Qt::EditRole);
    }
    int newVaryNorm=0;
    if(aDialog.varyNormCheck->isChecked()) newVaryNorm=1;
    if(newVaryNorm!=varyNorm) {
      i=segmentsDataModel->index(index.row(),11,QModelIndex());
      segmentsDataModel->setData(i,newVaryNorm,Qt::EditRole);
    }
    QString newPhaseJ=aDialog.phaseJValueText->text();
    if(newPhaseJ!=phaseJ) {
      i=segmentsDataModel->index(index.row(),12,QModelIndex());
      segmentsDataModel->setData(i,newPhaseJ,Qt::EditRole);
    }
    QString newPhaseL=aDialog.phaseLValueText->text();
    if(newPhaseL!=phaseL) {
      i=segmentsDataModel->index(index.row(),13,QModelIndex());
      segmentsDataModel->setData(i,newPhaseL,Qt::EditRole);
    }
  }
}

void SegmentsTab::editSegTestLine() {
  QItemSelectionModel *selectionModel = segmentsTestView->selectionModel();
  QModelIndexList indexes = selectionModel->selectedRows();
```

```
QModelIndex index=indexes[0];

QModelIndex i=segmentsTestModel->index(index.row(),1,QModelIndex());
QVariant var=segmentsTestModel->data(i,Qt::EditRole);
int entrancePairIndex=var.toInt();
i=segmentsTestModel->index(index.row(),2,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
int exitPairIndex=var.toInt();
i=segmentsTestModel->index(index.row(),3,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
QString lowEnergy=var.toString();
i=segmentsTestModel->index(index.row(),4,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
QString highEnergy=var.toString();
i=segmentsTestModel->index(index.row(),5,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
QString energyStep=var.toString();
i=segmentsTestModel->index(index.row(),6,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
QString lowAngle=var.toString();
i=segmentsTestModel->index(index.row(),7,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
QString highAngle=var.toString();
i=segmentsTestModel->index(index.row(),8,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
QString angleStep=var.toString();
i=segmentsTestModel->index(index.row(),9,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
int dataType=var.toInt();
i=segmentsTestModel->index(index.row(),10,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
QString phaseJ=var.toString();
i=segmentsTestModel->index(index.row(),11,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
QString phaseL=var.toString();
i=segmentsTestModel->index(index.row(),12,QModelIndex());
var=segmentsTestModel->data(i,Qt::EditRole);
int maxAngDistOrder=var.toInt();

AddSegTestDialog aDialog;
aDialog.setWindowTitle(tr("Edit a Segment Without Data"));
aDialog.entrancePairIndexSpin->setValue(entrancePairIndex);
aDialog.exitPairIndexSpin->setValue(exitPairIndex);
aDialog.lowEnergyText->setText(lowEnergy);
aDialog.highEnergyText->setText(highEnergy);
aDialog.energyStepText->setText(energyStep);
aDialog.lowAngleText->setText(lowAngle);
aDialog.highAngleText->setText(highAngle);
aDialog.angleStepText->setText(angleStep);
aDialog.dataTypeCombo->setCurrentIndex(dataType);
aDialog.phaseJValueText->setText(phaseJ);
aDialog.phaseLValueText->setText(phaseL);
aDialog.angDistSpin->setValue(maxAngDistOrder);


if(aDialog.exec()) {
  int newEntrancePairIndex=aDialog.entrancePairIndexSpin->value();
  if(newEntrancePairIndex!=entrancePairIndex) {
    i=segmentsTestModel->index(index.row(),1,QModelIndex());
    segmentsTestModel->setData(i,newEntrancePairIndex,Qt::EditRole);
  }
  int newExitPairIndex= (aDialog.dataTypeCombo->currentIndex()==4) ? -1 :
    aDialog.exitPairIndexSpin->value();
  if(newExitPairIndex!=exitPairIndex) {
    i=segmentsTestModel->index(index.row(),2,QModelIndex());
    segmentsTestModel->setData(i,newExitPairIndex,Qt::EditRole);
  }
  QString newLowEnergy=aDialog.lowEnergyText->text();
  if(newLowEnergy!=lowEnergy) {
    i=segmentsTestModel->index(index.row(),3,QModelIndex());
    segmentsTestModel->setData(i,newLowEnergy,Qt::EditRole);
  }
  QString newHighEnergy=aDialog.highEnergyText->text();
  if(newHighEnergy!=highEnergy) {
    i=segmentsTestModel->index(index.row(),4,QModelIndex());
    segmentsTestModel->setData(i,newHighEnergy,Qt::EditRole);
  }
  QString newEnergyStep=aDialog.energyStepText->text();
  if(newEnergyStep!=energyStep) {
    i=segmentsTestModel->index(index.row(),5,QModelIndex());
    segmentsTestModel->setData(i,newEnergyStep,Qt::EditRole);
  }
  QString newLowAngle=aDialog.lowAngleText->text();
  if(newLowAngle!=lowAngle) {
    i=segmentsTestModel->index(index.row(),6,QModelIndex());
    segmentsTestModel->setData(i,newLowAngle,Qt::EditRole);
  }
```

```
    QString newHighAngle=aDialog.highAngleText->text();
    if(newHighAngle!=highAngle) {
      i=segmentsTestModel->index(index.row(),7,QModelIndex());
      segmentsTestModel->setData(i,newHighAngle,Qt::EditRole);
    }
    QString newAngleStep=aDialog.angleStepText->text();
    if(newAngleStep!=angleStep) {
      i=segmentsTestModel->index(index.row(),8,QModelIndex());
      segmentsTestModel->setData(i,newAngleStep,Qt::EditRole);
    }
    int newDataType = aDialog.dataTypeCombo->currentIndex();
    if(newDataType!=dataType) {
      i=segmentsTestModel->index(index.row(),9,QModelIndex());
      segmentsTestModel->setData(i,newDataType,Qt::EditRole);
    }
    QString newPhaseJ=aDialog.phaseJValueText->text();
    if(newPhaseJ!=phaseJ) {
      i=segmentsTestModel->index(index.row(),10,QModelIndex());
      segmentsTestModel->setData(i,newPhaseJ,Qt::EditRole);
    }
    QString newPhaseL=aDialog.phaseLValueText->text();
    if(newPhaseL!=phaseL) {
      i=segmentsTestModel->index(index.row(),11,QModelIndex());
      segmentsTestModel->setData(i,newPhaseL,Qt::EditRole);
    }
    int newMaxAngDistOrder=aDialog.angDistSpin->value();
    if(newMaxAngDistOrder!=maxAngDistOrder) {
      i=segmentsTestModel->index(index.row(),12,QModelIndex());
      segmentsTestModel->setData(i,newMaxAngDistOrder,Qt::EditRole);
    }
  }
}

void SegmentsTab::moveSegDataLineUp() {
  moveSegDataLine(1);
}

void SegmentsTab::moveSegDataLineDown() {
  moveSegDataLine(0);
}

void SegmentsTab::moveSegDataLine(unsigned int upDown) {
  QItemSelectionModel *selectionModel = segmentsDataView->selectionModel();
  QModelIndexList selectionList = selectionModel->selectedRows();
  QModelIndex selectionIndex=selectionList.at(0);

  int previous = selectionIndex.row();
  int future;
  if(upDown==0) future = previous+1;
  else future = previous-1;
  SegmentsDataData line = segmentsDataModel->getLines().at(previous);
  segmentsDataModel->removeRows(previous,1,QModelIndex());
  segmentsDataModel->insertRows(future,1,QModelIndex());
  QModelIndex index = segmentsDataModel->index(future,0,QModelIndex());
  segmentsDataModel->setData(index,line.isActive,Qt::EditRole);
  index = segmentsDataModel->index(future,1,QModelIndex());
  segmentsDataModel->setData(index,line.entrancePairIndex,Qt::EditRole);
  index = segmentsDataModel->index(future,2,QModelIndex());
  segmentsDataModel->setData(index,line.exitPairIndex,Qt::EditRole);
  index = segmentsDataModel->index(future,3,QModelIndex());
  segmentsDataModel->setData(index,line.lowEnergy,Qt::EditRole);
  index = segmentsDataModel->index(future,4,QModelIndex());
  segmentsDataModel->setData(index,line.highEnergy,Qt::EditRole);
  index = segmentsDataModel->index(future,5,QModelIndex());
  segmentsDataModel->setData(index,line.lowAngle,Qt::EditRole);
  index = segmentsDataModel->index(future,6,QModelIndex());
  segmentsDataModel->setData(index,line.highAngle,Qt::EditRole);
  index = segmentsDataModel->index(future,7,QModelIndex());
  segmentsDataModel->setData(index,line.dataType,Qt::EditRole);
  index = segmentsDataModel->index(future,8,QModelIndex());
  segmentsDataModel->setData(index,line.dataFile,Qt::EditRole);
  index = segmentsDataModel->index(future,9,QModelIndex());
  segmentsDataModel->setData(index,line.dataNorm,Qt::EditRole);
  index = segmentsDataModel->index(future,10,QModelIndex());
  segmentsDataModel->setData(index,line.dataNormError,Qt::EditRole);
  index = segmentsDataModel->index(future,11,QModelIndex());
  segmentsDataModel->setData(index,line.varyNorm,Qt::EditRole);
  index = segmentsDataModel->index(future,12,QModelIndex());
  segmentsDataModel->setData(index,line.phaseJ,Qt::EditRole);
  index = segmentsDataModel->index(future,13,QModelIndex());
  segmentsDataModel->setData(index,line.phaseL,Qt::EditRole);
  segmentsDataView->resizeRowToContents(future);

  selectionModel->select(segmentsDataModel->index(future,0,QModelIndex()),
             QItemSelectionModel::ClearAndSelect | QItemSelectionModel::Rows);
}

```

```
void SegmentsTab::moveSegTestLineUp() {
  moveSegTestLine(1);
}

void SegmentsTab::moveSegTestLineDown() {
  moveSegTestLine(0);
}

void SegmentsTab::moveSegTestLine(unsigned int upDown) {
  QItemSelectionModel *selectionModel = segmentsTestView->selectionModel();
  QModelIndexList selectionList = selectionModel->selectedRows();
  QModelIndex selectionIndex=selectionList.at(0);

  int previous = selectionIndex.row();
  int future;
  if(upDown==0) future = previous+1;
  else future = previous-1;

  SegmentsTestData line = segmentsTestModel->getLines().at(previous);
  segmentsTestModel->removeRows(previous,1,QModelIndex());
  segmentsTestModel->insertRows(future,1,QModelIndex());
  QModelIndex index = segmentsTestModel->index(future,0,QModelIndex());
  segmentsTestModel->setData(index,line.isActive,Qt::EditRole);
  index = segmentsTestModel->index(future,1,QModelIndex());
  segmentsTestModel->setData(index,line.entrancePairIndex,Qt::EditRole);
  index = segmentsTestModel->index(future,2,QModelIndex());
  segmentsTestModel->setData(index,line.exitPairIndex,Qt::EditRole);
  index = segmentsTestModel->index(future,3,QModelIndex());
  segmentsTestModel->setData(index,line.lowEnergy,Qt::EditRole);
  index = segmentsTestModel->index(future,4,QModelIndex());
  segmentsTestModel->setData(index,line.highEnergy,Qt::EditRole);
  index = segmentsTestModel->index(future,5,QModelIndex());
  segmentsTestModel->setData(index,line.energyStep,Qt::EditRole);
  index = segmentsTestModel->index(future,6,QModelIndex());
  segmentsTestModel->setData(index,line.lowAngle,Qt::EditRole);
  index = segmentsTestModel->index(future,7,QModelIndex());
  segmentsTestModel->setData(index,line.highAngle,Qt::EditRole);
  index = segmentsTestModel->index(future,8,QModelIndex());
  segmentsTestModel->setData(index,line.angleStep,Qt::EditRole);
  index = segmentsTestModel->index(future,9,QModelIndex());
  segmentsTestModel->setData(index,line.dataType,Qt::EditRole);
  index = segmentsTestModel->index(future,10,QModelIndex());
  segmentsTestModel->setData(index,line.phaseJ,Qt::EditRole);
  index = segmentsTestModel->index(future,11,QModelIndex());
  segmentsTestModel->setData(index,line.phaseL,Qt::EditRole);
  index = segmentsTestModel->index(future,12,QModelIndex());
  segmentsTestModel->setData(index,line.maxAngDistOrder,Qt::EditRole);
  segmentsTestView->resizeRowToContents(future);

  selectionModel->select(segmentsTestModel->index(future,0,QModelIndex()),
              QItemSelectionModel::ClearAndSelect | QItemSelectionModel::Rows);
}

void SegmentsTab::updateSegDataButtons(const QItemSelection &selection) {
  QModelIndexList indexes=selection.indexes();

  if(indexes.isEmpty()) {
    segDataDeleteButton->setEnabled(false);
    segDataUpButton->setEnabled(false);
    segDataDownButton->setEnabled(false);
  } else {
    segDataDeleteButton->setEnabled(true);
    if(indexes.at(0).row()==0) segDataUpButton->setEnabled(false);
    else segDataUpButton->setEnabled(true);
    if(indexes.at(0).row()==segmentsDataModel->rowCount(QModelIndex())-1)
 segDataDownButton->setEnabled(false);
    else segDataDownButton->setEnabled(true);
  }
}

void SegmentsTab::updateSegTestButtons(const QItemSelection &selection) {
  QModelIndexList indexes=selection.indexes();

  if(indexes.isEmpty()) {
    segTestDeleteButton->setEnabled(false);
    segTestUpButton->setEnabled(false);
    segTestDownButton->setEnabled(false);
  } else {
    segTestDeleteButton->setEnabled(true);
    if(indexes.at(0).row()==0) segTestUpButton->setEnabled(false);
    else segTestUpButton->setEnabled(true);
    if(indexes.at(0).row()==segmentsTestModel->rowCount(QModelIndex())-1)
 segTestDownButton->setEnabled(false);
    else segTestDownButton->setEnabled(true);
  }
}

```

```
bool SegmentsTab::readSegDataFile(QTextStream& inStream) {

  int isActive;
  int entrancePairIndex;
  int exitPairIndex;
  double lowEnergy;
  double highEnergy;
  double lowAngle;
  double highAngle;
  int dataType;
  QString dataFile;
  double dataNorm;
  double dataNormError;
  int varyNorm;
  double phaseJ;
  int phaseL;

  QString line("");
  while(!inStream.atEnd()&&line.trimmed()!=QString("</segmentsData>")) {
    line=inStream.readLine();
    if(line.trimmed().isEmpty()) continue;
    if(!inStream.atEnd()&&line.trimmed()!=QString("</segmentsData>")) {
      QTextStream in(&line);
      in » isActive » entrancePairIndex » exitPairIndex » lowEnergy » highEnergy » lowAngle » highAngle
     » dataType;
      if(dataType==2) in»phaseJ»phaseL;
      else {
    phaseJ=0.;
    phaseL=0;
      }
      in » dataNorm » varyNorm;
      QString restOfLine=in.readLine();
      QTextStream stm(&restOfLine);
      stm»dataNormError;
      if(stm.status()!=QTextStream::Ok) {
    dataNormError=0.;
    dataFile=restOfLine.trimmed();
      } else dataFile=stm.readLine().trimmed();
      if(in.status()!=QTextStream::Ok) return false;
      SegmentsDataData newLine = {isActive,entrancePairIndex,exitPairIndex,lowEnergy,highEnergy,lowAngle,
                    highAngle,dataType,dataFile,dataNorm,dataNormError,varyNorm,phaseJ,phaseL};
      addSegDataLine(newLine);
    }
  }
  if(line.trimmed()!=QString("</segmentsData>")) return false;
  return true;
}

bool SegmentsTab::writeSegDataFile(QTextStream& outStream) {

  QList<SegmentsDataData> lines = segmentsDataModel->getLines();

  for(int i = 0; i<lines.size(); i++) {
    outStream « qSetFieldWidth(15) « lines.at(i).isActive
          « qSetFieldWidth(15) « lines.at(i).entrancePairIndex
          « qSetFieldWidth(15) « lines.at(i).exitPairIndex
          « qSetFieldWidth(15) « lines.at(i).lowEnergy
          « qSetFieldWidth(15) « lines.at(i).highEnergy
          « qSetFieldWidth(15) « lines.at(i).lowAngle
          « qSetFieldWidth(15) « lines.at(i).highAngle
          « qSetFieldWidth(15) « lines.at(i).dataType;
    if(lines.at(i).dataType == 2) outStream  « qSetFieldWidth(15) « lines.at(i).phaseJ
                     « qSetFieldWidth(15) « lines.at(i).phaseL;
    outStream « qSetFieldWidth(15) « lines.at(i).dataNorm
          « qSetFieldWidth(15) « lines.at(i).varyNorm
          « qSetFieldWidth(15) « lines.at(i).dataNormError
          « qSetFieldWidth(0) « lines.at(i).dataFile « endl;
  }

  return true;
}

bool SegmentsTab::readSegTestFile(QTextStream& inStream) {

  int isActive;
  int entrancePairIndex;
  int exitPairIndex;
  double lowEnergy;
  double highEnergy;
  double energyStep;
  double lowAngle;
  double highAngle;
  double angleStep;
  int dataType;
```

```
  double phaseJ;
  int phaseL;
  int maxAngDistOrder;

  QString line("");
  while(!inStream.atEnd()&&line.trimmed()!=QString("</segmentsTest>")) {
    line = inStream.readLine();
    if(line.trimmed().isEmpty()) continue;
    if(!inStream.atEnd()&&line.trimmed()!=QString("</segmentsTest>")) {
      QTextStream in(&line);
      in » isActive » entrancePairIndex » exitPairIndex » lowEnergy » highEnergy » energyStep » lowAngle » highAngle » angleStep
     » dataType;
      if(dataType==2) {
    in » phaseJ » phaseL;
      } else {
    phaseJ=0.;
    phaseL = 0;
      }
      if(dataType==3) {
    in » maxAngDistOrder;
      } else {
    maxAngDistOrder=0;
      }
      if(in.status()!=QTextStream::Ok) return false;
      SegmentsTestData newLine = {isActive,entrancePairIndex,exitPairIndex,lowEnergy,highEnergy,energyStep,lowAngle,
                  highAngle,angleStep,dataType,phaseJ,phaseL,maxAngDistOrder};
      addSegTestLine(newLine);
    }
  }
  if(line.trimmed()!=QString("</segmentsTest>")) return false;
  return true;
}

bool SegmentsTab::writeSegTestFile(QTextStream& outStream) {

  QList<SegmentsTestData> lines = segmentsTestModel->getLines();

  for(int i = 0; i<lines.size(); i++) {
    outStream « qSetFieldWidth(15) « lines.at(i).isActive
          « qSetFieldWidth(15) « lines.at(i).entrancePairIndex
          « qSetFieldWidth(15) « lines.at(i).exitPairIndex
          « qSetFieldWidth(15) « lines.at(i).lowEnergy
          « qSetFieldWidth(15) « lines.at(i).highEnergy
          « qSetFieldWidth(15) « lines.at(i).energyStep
          « qSetFieldWidth(15) « lines.at(i).lowAngle
          « qSetFieldWidth(15) « lines.at(i).highAngle
          « qSetFieldWidth(15) « lines.at(i).angleStep;
    if(lines.at(i).dataType==2) {
      outStream « qSetFieldWidth(15) « lines.at(i).dataType
        « qSetFieldWidth(15) « lines.at(i).phaseJ
        « qSetFieldWidth(0) « lines.at(i).phaseL
        « endl;
    } else if(lines.at(i).dataType==3)  {
      outStream  « qSetFieldWidth(15) « lines.at(i).dataType
         « qSetFieldWidth(0) « lines.at(i).maxAngDistOrder
         « endl;
    } else outStream « qSetFieldWidth(0) « lines.at(i).dataType
             « endl;
  }

  return true;
}

void SegmentsTab::reset() {
  segmentsDataModel->removeRows(0,segmentsDataModel->getLines().size(),QModelIndex());
  segmentsTestModel->removeRows(0,segmentsTestModel->getLines().size(),QModelIndex());
}

void SegmentsTab::showInfo(int which,QString title) {
  if(which<infoText.size()) {
    if(!infoDialog[which]) {
      infoDialog[which] = new InfoDialog(infoText[which],this,title);
      infoDialog[which]->setAttribute(Qt:: WA_DeleteOnClose);
      infoDialog[which]->show();
    } else infoDialog[which]->raise();
  }
}
```

8.129 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTest↩Model.cpp File Reference

```
#include ¨SegmentsTestModel.h¨
#include ¨PairsModel.h¨
```

8.130 SegmentsTestModel.cpp

Go to the documentation of this file.

```
#include "SegmentsTestModel.h"
#include "PairsModel.h"

SegmentsTestModel::SegmentsTestModel(QObject *parent) : QAbstractTableModel(parent) {
}

int SegmentsTestModel::rowCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return segTestLineList.size();
}

int SegmentsTestModel::columnCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return SegmentsTestData::SIZE;
}

QVariant SegmentsTestModel::data(const QModelIndex &index, int role) const {
  if(!index.isValid()) return QVariant();

  if(index.row() >= segTestLineList.size() || index.row() < 0)  return QVariant();

  if (role == Qt::DisplayRole) {
    SegmentsTestData line = segTestLineList.at(index.row());
    if(index.column() == 1) {
      if(line.dataType==4) {
    int i = 0;
    QList<PairsData> pairsList = pairsModel->getPairs();
    for(i=0;i<pairsList.size();i++)
      if(pairsList[i].pairType==10) break;
    if(pairsList.size()>=line.entrancePairIndex&&i<pairsList.size()) {
      PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
      return QString("<center>%1</center>").arg(pairsModel->getReactionLabelTotalCapture(firstPair));
    } else return QString("<center><font style='color:red;font-weight:bold;'>UNDEFINED</font></center>");
      } else {
if(pairsModel->getPairs().size()>=line.entrancePairIndex&&pairsModel->getPairs().size()>=line.exitPairIndex) {
      PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
      PairsData secondPair=pairsModel->getPairs().at(line.exitPairIndex-1);
      return QString("<center>%1</center>").arg(pairsModel->getReactionLabel(firstPair,secondPair));
    } else return QString("<center><font style='color:red;font-weight:bold'>UNDEFINED</font></center>");
      }
    } else if(index.column() == 2) return QVariant();
    else if(index.column() == 3) {
      if(line.lowEnergy==line.highEnergy) return line.lowEnergy;
      else return QString("%1-%2").arg(line.lowEnergy).arg(line.highEnergy);
    } else if(index.column() == 4) return QVariant();
    else if(index.column() == 5) return line.energyStep;
    else if(index.column() == 6) {
      if(line.lowAngle==line.highAngle) return line.lowAngle;
      else return QString("%1-%2").arg(line.lowAngle).arg(line.highAngle);
    } else if(index.column() == 7) return QVariant();
    else if(index.column() == 8) return line.angleStep;
    else if(index.column() == 9) {
      if(line.dataType==3) return QString(tr("<center>Angular Distribution</center>"));
      else if(line.dataType==2) {
    QChar orbital;
    switch (line.phaseL) {
    case 0:
      orbital='s';
      break;
    case 1:
      orbital='p';
      break;
    case 2:
```

```
          orbital='d';
          break;
        case 3:
          orbital='f';
          break;
        case 4:
          orbital='g';
          break;
        case 5:
          orbital='h';
          break;
        case 6:
          orbital='i';
          break;
        default:
          orbital='?';
        }
        QString tempSpin;
        if(((int)(line.phaseJ*2))%2!=0&&line.phaseJ!=0.)
  tempSpin=QString("%1/2").arg((int)(line.phaseJ*2));
        else tempSpin=QString("%1").arg(line.phaseJ);
        return QString("<center>Phase Shift [%1<sub>%2</sub>]</center>").arg(orbital).arg(tempSpin);
          } else if(line.dataType==1) return QString(tr("<center>Differential</center>"));
          else if(line.dataType==4) return QString(tr("<center>Total Capture</center>"));
          else return QString(tr("<center>Angle Integrated</center>"));
        }
        else if(index.column() == 10) return QVariant();
        else if(index.column() == 11) return QVariant();
        else if(index.column() == 12) return QVariant();
      } else if (role == Qt::EditRole) {
        SegmentsTestData line = segTestLineList.at(index.row());
        if(index.column() == 1) return line.entrancePairIndex;
        else if(index.column() == 2) return line.exitPairIndex;
        else if(index.column() == 3) return line.lowEnergy;
        else if(index.column() == 4) return line.highEnergy;
        else if(index.column() == 5) return line.energyStep;
        else if(index.column() == 6) return line.lowAngle;
        else if(index.column() == 7) return line.highAngle;
        else if(index.column() == 8) return line.angleStep;
        else if(index.column() == 9) return line.dataType;
        else if(index.column() == 10) return line.phaseJ;
        else if(index.column() == 11) return line.phaseL;
        else if(index.column() == 12) return line.maxAngDistOrder;
      } else if (role==Qt::CheckStateRole && index.column()==0) {
        SegmentsTestData line = segTestLineList.at(index.row());
        if(line.isActive==1) return Qt::Checked;
        else return Qt::Unchecked;
      } else if(role == Qt::TextAlignmentRole) return Qt::AlignCenter;

      return QVariant();
  }

  QVariant SegmentsTestModel::headerData(int section, Qt::Orientation orientation, int role) const {
      if(role!= Qt::DisplayRole) return QVariant();
      if(orientation == Qt::Horizontal) {
        switch(section) {
        case 0:
          return tr("");
        case 1:
          return tr("Reaction");
        case 2:
          return QVariant();
        case 3:
          return tr("Energy\nRange");
        case 4:
          return QVariant();
        case 5:
          return tr("Energy\nStep");
        case 6:
          return tr("Angle\nRange");
        case 7:
          return QVariant();
        case 8:
          return tr("Angle\nStep");
        case 9:
          return tr("Data Type");
        case 10:
          return QVariant();
        case 11:
          return QVariant();
        case 12:
          return QVariant();
        default:
          return QVariant();
        }
      } else if(orientation == Qt::Vertical) {
        return section+1;
```

```
  }
  return QVariant();
}

bool SegmentsTestModel::setData(const QModelIndex &index, const QVariant &value, int role) {
  if (index.isValid() && role == Qt::EditRole ) {
    int row = index.row();
    SegmentsTestData tempData = segTestLineList.value(row);
    if(index.column() == 0) tempData.isActive=value.toInt();
    else if(index.column() == 1) tempData.entrancePairIndex=value.toInt();
    else if(index.column() == 2) tempData.exitPairIndex=value.toInt();
    else if(index.column() == 3) tempData.lowEnergy=value.toDouble();
    else if(index.column() == 4) tempData.highEnergy=value.toDouble();
    else if(index.column() == 5) tempData.energyStep=value.toDouble();
    else if(index.column() == 6) tempData.lowAngle=value.toDouble();
    else if(index.column() == 7) tempData.highAngle=value.toDouble();
    else if(index.column() == 8) tempData.angleStep=value.toDouble();
    else if(index.column() == 9) tempData.dataType=value.toInt();
    else if(index.column() == 10) tempData.phaseJ=value.toDouble();
    else if(index.column() == 11) tempData.phaseL=value.toInt();
    else if(index.column() == 12) tempData.maxAngDistOrder=value.toInt();
    else return false;

    segTestLineList.replace(row,tempData);
    emit(dataChanged(index,index));
    return true;
  } else if(role== Qt::CheckStateRole) {
    int row = index.row();
    SegmentsTestData tempData = segTestLineList.value(row);
    if(index.column()==0) {
      if(value==Qt::Checked) tempData.isActive=1;
      else tempData.isActive=0;
    } else return false;
    segTestLineList.replace(row,tempData);
    emit(dataChanged(index,index));
    return true;
  }
  return false;
}

bool SegmentsTestModel::insertRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginInsertRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows; row++) {
      SegmentsTestData tempData;
      segTestLineList.insert(position,tempData);
    }
    endInsertRows();
  }
  return true;
}

bool SegmentsTestModel::removeRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginRemoveRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows;++row) {
      segTestLineList.removeAt(position);
    }
    endRemoveRows();
  }
  return true;
}

Qt::ItemFlags SegmentsTestModel::flags(const QModelIndex &index) const {
  if (!index.isValid()) return Qt::ItemIsEnabled;
  if(index.column()==0) return QAbstractTableModel::flags(index) | Qt::ItemIsUserCheckable;
  return QAbstractTableModel::flags(index);
}

int SegmentsTestModel::isSegTestLine(const SegmentsTestData &line) const {
  int foundLine=-1;
  for(int i=0;i<segTestLineList.size();i++) {
    SegmentsTestData tempLine=segTestLineList.value(i);
    if(tempLine.entrancePairIndex==line.entrancePairIndex&&
       tempLine.exitPairIndex==line.exitPairIndex&&
       tempLine.lowEnergy==line.lowEnergy&&
       tempLine.highEnergy==line.highEnergy&&
       tempLine.energyStep==line.energyStep&&
       tempLine.lowAngle==line.lowAngle&&
       tempLine.highAngle==line.highAngle&&
       tempLine.angleStep==line.angleStep&&
       tempLine.dataType==line.dataType&&
       tempLine.phaseJ==line.phaseJ&&
       tempLine.phaseL==line.phaseL&&
       tempLine.maxAngDistOrder==line.maxAngDistOrder) {
```

```
      foundLine=i;
      break;
    }
  }
  return foundLine;
}

void SegmentsTestModel::setPairsModel(PairsModel* model) {
  pairsModel=model;
}

QString SegmentsTestModel::getReactionLabel(const QModelIndex &index) {
  SegmentsTestData line = segTestLineList.at(index.row());
  if(line.dataType==4) {
    int i = 0;
    QList<PairsData> pairsList = pairsModel->getPairs();
    for(i=0;i<pairsList.size();i++)
      if(pairsList[i].pairType==10) break;
    if(pairsList.size()>=line.entrancePairIndex&&i<pairsList.size()) {
      PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
      return pairsModel->getReactionLabelTotalCapture(firstPair);
    }
    return QString("<font style='color:red;font-weight:bold;'>UNDEFINED</font>");
  } else {
    int numPairs = pairsModel->getPairs().size();
    if(line.entrancePairIndex-1>=numPairs ||
       line.exitPairIndex-1>=numPairs) return QString("<font style='color:red;font-weight:bold'>UNDEFINED</font>");
    PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
    PairsData secondPair=pairsModel->getPairs().at(line.exitPairIndex-1);
    return pairsModel->getReactionLabel(firstPair,secondPair);
  }
}
```

8.131 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntModel.cpp File Reference

```
#include <QVariant>
#include "TargetIntModel.h"
```

8.132 TargetIntModel.cpp

Go to the documentation of this file.

```
#include <QVariant>

#include "TargetIntModel.h"

TargetIntModel::TargetIntModel(QObject *parent) : QAbstractTableModel(parent) {
}

int TargetIntModel::rowCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return targetIntList.size();
}

int TargetIntModel::columnCount(const QModelIndex &parent) const {
  Q_UNUSED(parent);
  return TargetIntData::SIZE;
}

QVariant TargetIntModel::data(const QModelIndex &index, int role) const {
  if(!index.isValid()) return QVariant();
  if(index.row()>=targetIntList.size() || index.row() < 0) return QVariant();
  if(role == Qt::DisplayRole) {
    TargetIntData targetInt=targetIntList.at(index.row());
    if(index.column() == 1) return targetInt.segmentsList;
    else if(index.column() == 2) {
      if(targetInt.isTargetIntegration||targetInt.isConvolution) return targetInt.numPoints;
      else return QString(tr("N/A"));
    }
    else if(index.column() == 3) {
      if(targetInt.isConvolution) return QString(tr("YES"));
```

```
      else return QString(tr("NO"));
    } else if (index.column() == 4) return targetInt.sigma;
    else if(index.column() == 5) {
      if(targetInt.isTargetIntegration) return QString(tr("YES"));
      else return QString(tr("NO"));
    } else if(index.column() == 6) return targetInt.density;
    else if(index.column() == 7) return targetInt.stoppingPowerEq;
    else if(index.column() == 8) return targetInt.numParameters;
    else if(index.column() == 9) return QVariant();
    else if(index.column() == 10) {
      if(targetInt.isQCoefficients) return QString(tr("YES"));
      else return QString(tr("NO"));
    } else if(index.column() == 11) return QVariant();
  } else if(role == Qt::EditRole) {
    TargetIntData targetInt=targetIntList.at(index.row());
    if(index.column()==1) return targetInt.segmentsList;
    if(index.column()==2) return targetInt.numPoints;
    if(index.column()==3) return targetInt.isConvolution;
    if(index.column()==4) return targetInt.sigma;
    if(index.column()==5) return targetInt.isTargetIntegration;
    if(index.column()==6) return targetInt.density;
    if(index.column()==7) return targetInt.stoppingPowerEq;
    if(index.column()==8) return targetInt.numParameters;
    if(index.column()==9) return QVariant::fromValue<QList<double> >(targetInt.parameters);
    if(index.column()==10) return  targetInt.isQCoefficients;
    if(index.column()==11) return QVariant::fromValue<QList<double> >(targetInt.qCoefficients);
  } else if (role==Qt::CheckStateRole && index.column()==0) {
    TargetIntData targetInt=targetIntList.at(index.row());
    if(targetInt.isActive==1) return Qt::Checked;
    else return Qt::Unchecked;
  } else if(role == Qt::TextAlignmentRole)  return Qt::AlignCenter;
  return QVariant();
}

QVariant TargetIntModel::headerData(int section, Qt::Orientation orientation, int role) const {
  if(role!=Qt::DisplayRole) return QVariant();
  if(orientation == Qt::Horizontal) {
    switch(section) {
    case 0:
      return tr("");
    case 1:
      return tr("Segment List");
    case 2:
      return tr("Number of Integration Points");
    case 3:
      return tr("Convolution Active?");
    case 4:
      return tr("Gaussian Sigma");
    case 5:
      return tr("Target Integration Active?");
    case 6:
      return tr("Target Density");
    case 7:
      return tr("Stopping Power Equation");
    case 8:
      return tr("Number of Parameters");
    case 9:
      return tr("Parameters List");
    case 10:
      return tr("Use Q-Coefficients?");
    case 11:
      return tr("Q-Coefficient List");
    default:
      return QVariant();
    }
  } else if(orientation == Qt::Vertical) return section+1;
  return QVariant();
}

bool TargetIntModel::setData(const QModelIndex &index, const QVariant &value, int role) {
  if (index.isValid() && role == Qt::EditRole ) {
    int row = index.row();
    TargetIntData tempData = targetIntList.value(row);
    if(index.column() == 0) tempData.isActive = value.toInt();
    else if(index.column() == 1) tempData.segmentsList = value.toString();
    else if(index.column() == 2) tempData.numPoints = value.toInt();
    else if(index.column() == 3) tempData.isConvolution = value.toBool();
    else if(index.column() == 4) tempData.sigma = value.toDouble();
    else if(index.column() == 5) tempData.isTargetIntegration = value.toBool();
    else if(index.column() == 6) tempData.density = value.toDouble();
    else if(index.column() == 7) tempData.stoppingPowerEq = value.toString();
    else if(index.column() == 8) tempData.numParameters = value.toInt();
    else if(index.column() == 9) tempData.parameters = value.value<QList<double> >();
    else if(index.column() == 10) tempData.isQCoefficients = value.toBool();
    else if(index.column() == 11) tempData.qCoefficients = value.value<QList<double> >();
    else return false;
    targetIntList.replace(row,tempData);
```

```
    emit(dataChanged(index,index));
    return true;
  } else if(role == Qt::CheckStateRole) {
    int row = index.row();
    TargetIntData tempData = targetIntList.value(row);
    if(index.column() == 0) {
      if(value==Qt::Checked) tempData.isActive=1;
      else tempData.isActive=0;
    } else return false;
    targetIntList.replace(row,tempData);
    emit(dataChanged(index,index));
    return true;
  }
  return false;
}

bool TargetIntModel::insertRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginInsertRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows; row++) {
     TargetIntData tempData;
      targetIntList.insert(position,tempData);
    }
    endInsertRows();
  }
  return true;
}

bool TargetIntModel::removeRows(int position, int rows, const QModelIndex &index) {
  Q_UNUSED(index);
  if(rows>0) {
    beginRemoveRows(QModelIndex(),position,position+rows-1);
    for(int row=0; row<rows;++row) {
      targetIntList.removeAt(position);
    }
    endRemoveRows();
  }
  return true;
}

Qt::ItemFlags TargetIntModel::flags(const QModelIndex &index) const {
  if (!index.isValid()) return Qt::ItemIsEnabled;
  if(index.column()==0) return QAbstractTableModel::flags(index) | Qt::ItemIsUserCheckable;
  return QAbstractTableModel::flags(index);
}
```

8.133 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntTab.cpp File Reference

#include <QGridLayout>
#include <QCheckBox>
#include <QSpacerItem>
#include <QLineEdit>
#include <QSpinBox>
#include <QPushButton>
#include <QTextStream>
#include <QHeaderView>
#include "TargetIntTab.h"
#include "InfoDialog.h"

8.134 TargetIntTab.cpp

Go to the documentation of this file.

```
#include <QGridLayout>
#include <QCheckBox>
#include <QSpacerItem>
#include <QLineEdit>
```

```
#include <QSpinBox>
#include <QPushButton>
#include <QTextStream>
#include <QHeaderView>

#include "TargetIntTab.h"
#include "InfoDialog.h"


TargetIntTab::TargetIntTab(QWidget *parent) : QWidget(parent) {
  targetIntModel = new TargetIntModel(this);
  targetIntView = new QTableView;
  targetIntView->setModel(targetIntModel);
  targetIntView->verticalHeader()->setHighlightSections(false);
  targetIntView->horizontalHeader()->setHighlightSections(false);
  targetIntView->setColumnHidden(4,true);
  targetIntView->setColumnHidden(6,true);
  targetIntView->setColumnHidden(7,true);
  targetIntView->setColumnHidden(8,true);
  targetIntView->setColumnHidden(9,true);
  targetIntView->setColumnHidden(11,true);
  targetIntView->setColumnWidth(0,27);
  targetIntView->horizontalHeader()->setSectionResizeMode(0,QHeaderView::Fixed);
  targetIntView->horizontalHeader()->setSectionResizeMode(1,QHeaderView::Stretch);
  targetIntView->horizontalHeader()->setSectionResizeMode(2,QHeaderView::Stretch);
  targetIntView->horizontalHeader()->setSectionResizeMode(3,QHeaderView::Stretch);
  targetIntView->horizontalHeader()->setSectionResizeMode(5,QHeaderView::Stretch);
  targetIntView->horizontalHeader()->setSectionResizeMode(10,QHeaderView::Stretch);
  targetIntView->setSelectionBehavior(QAbstractItemView::SelectRows);
  targetIntView->setSelectionMode(QAbstractItemView::SingleSelection);
  targetIntView->setEditTriggers(QAbstractItemView::NoEditTriggers);
  targetIntView->setShowGrid(false);

connect(targetIntView->selectionModel(),SIGNAL(selectionChanged(QItemSelection,QItemSelection)),this,SLOT(updateButtons
  connect(targetIntView,SIGNAL(doubleClicked(QModelIndex)),this,SLOT(editLine()));

  addButton = new QPushButton(tr("+"));
  addButton->setMaximumSize(28,28);
  connect(addButton,SIGNAL(clicked()),this,SLOT(addLine()));
  deleteButton = new QPushButton(tr("-"));
  deleteButton->setMaximumSize(28,28);
  deleteButton->setEnabled(false);
  connect(deleteButton,SIGNAL(clicked()),this,SLOT(deleteLine()));

  QGridLayout *buttonBox = new QGridLayout;
  buttonBox->addWidget(addButton,0,0);
  buttonBox->addWidget(deleteButton,0,1);
  buttonBox->addItem(new QSpacerItem(28,28),0,2);
  buttonBox->setColumnStretch(0,0);
  buttonBox->setColumnStretch(1,0);
  buttonBox->setColumnStretch(2,1);
#ifdef MACX_SPACING
  buttonBox->setHorizontalSpacing(11);
#else
  buttonBox->setHorizontalSpacing(0);
#endif

  QGridLayout *mainLayout = new QGridLayout;
  mainLayout->addWidget(targetIntView,0,0);
  mainLayout->addLayout(buttonBox,1,0);
  mainLayout->setRowStretch(0,1);
  mainLayout->setRowStretch(1,0);

  setLayout(mainLayout);
}

TargetIntModel* TargetIntTab::getTargetIntModel() {
  return targetIntModel;
}

void TargetIntTab::addLine() {
  AddTargetIntDialog aDialog;
  if(aDialog.exec()) {
    TargetIntData newLine;
    newLine.isActive=1;
    newLine.segmentsList=aDialog.segmentsListText->text();
    newLine.numPoints=aDialog.numPointsSpin->value();
    if(aDialog.isConvolutionCheck->isChecked())
      newLine.isConvolution=true;
    else newLine.isConvolution=false;
    newLine.sigma=aDialog.sigmaText->text().toDouble();
    if(aDialog.isTargetIntegrationCheck->isChecked())
      newLine.isTargetIntegration=true;
    else newLine.isTargetIntegration=false;
    newLine.density=aDialog.densityText->text().toDouble();
    newLine.stoppingPowerEq=aDialog.stoppingPowerEqText->text();
    newLine.numParameters=aDialog.numParametersSpin->value();
```

```
    for(int i=0;i<newLine.numParameters;i++)
      newLine.parameters.append(aDialog.tempParameters.at(i));
    newLine.isQCoefficients = (aDialog.isQCoefficientCheck->isChecked()) ? (true) : (false);
    for(int i=0;i<aDialog.numQCoefficientSpin->value();i++)
      newLine.qCoefficients.append(aDialog.tempQCoefficients.at(i));
    addLine(newLine);
  }
}

void TargetIntTab::addLine(TargetIntData line) {
  QList<TargetIntData> lines = targetIntModel->getLines();
  targetIntModel->insertRows(lines.size(),1,QModelIndex());
  QModelIndex index = targetIntModel->index(lines.size(),0,QModelIndex());
  targetIntModel->setData(index,line.isActive,Qt::EditRole);
  index = targetIntModel->index(lines.size(),1,QModelIndex());
  targetIntModel->setData(index,line.segmentsList,Qt::EditRole);
  index = targetIntModel->index(lines.size(),2,QModelIndex());
  targetIntModel->setData(index,line.numPoints,Qt::EditRole);
  index = targetIntModel->index(lines.size(),3,QModelIndex());
  targetIntModel->setData(index,line.isConvolution,Qt::EditRole);
  index = targetIntModel->index(lines.size(),4,QModelIndex());
  targetIntModel->setData(index,line.sigma,Qt::EditRole);
  index = targetIntModel->index(lines.size(),5,QModelIndex());
  targetIntModel->setData(index,line.isTargetIntegration,Qt::EditRole);
  index = targetIntModel->index(lines.size(),6,QModelIndex());
  targetIntModel->setData(index,line.density,Qt::EditRole);
  index = targetIntModel->index(lines.size(),7,QModelIndex());
  targetIntModel->setData(index,line.stoppingPowerEq,Qt::EditRole);
  index = targetIntModel->index(lines.size(),8,QModelIndex());
  targetIntModel->setData(index,line.numParameters,Qt::EditRole);
  index = targetIntModel->index(lines.size(),9,QModelIndex());
  targetIntModel->setData(index,QVariant::fromValue<QList<double> >(line.parameters),Qt::EditRole);
  index = targetIntModel->index(lines.size(),10,QModelIndex());
  targetIntModel->setData(index,line.isQCoefficients,Qt::EditRole);
  index = targetIntModel->index(lines.size(),11,QModelIndex());
  targetIntModel->setData(index,QVariant::fromValue<QList<double> >(line.qCoefficients),Qt::EditRole);

  targetIntView->resizeRowsToContents();
}

void TargetIntTab::editLine() {
  QItemSelectionModel *selectionModel = targetIntView->selectionModel();
  QModelIndexList indexes = selectionModel->selectedRows();
  QModelIndex index = indexes[0];

  QModelIndex i=targetIntModel->index(index.row(),1,QModelIndex());
  QVariant var=targetIntModel->data(i,Qt::EditRole);
  QString segmentsList=var.toString();
  i=targetIntModel->index(index.row(),2,QModelIndex());
  var=targetIntModel->data(i,Qt::EditRole);
  int numPoints=var.toInt();
  i=targetIntModel->index(index.row(),3,QModelIndex());
  var=targetIntModel->data(i,Qt::EditRole);
  bool isConvolution = var.toBool();
  i=targetIntModel->index(index.row(),4,QModelIndex());
  var=targetIntModel->data(i,Qt::EditRole);
  QString sigma = var.toString();
  i=targetIntModel->index(index.row(),5,QModelIndex());
  var=targetIntModel->data(i,Qt::EditRole);
  bool isTargetIntegration = var.toBool();
  i=targetIntModel->index(index.row(),6,QModelIndex());
  var=targetIntModel->data(i,Qt::EditRole);
  QString density = var.toString();
  i=targetIntModel->index(index.row(),7,QModelIndex());
  var=targetIntModel->data(i,Qt::EditRole);
  QString stoppingPowerEq = var.toString();
  i=targetIntModel->index(index.row(),8,QModelIndex());
  var=targetIntModel->data(i,Qt::EditRole);
  int numParameters = var.toInt();
  i=targetIntModel->index(index.row(),9,QModelIndex());
  var=targetIntModel->data(i,Qt::EditRole);
  QList<double> parameters = var.value<QList<double> >();
  i=targetIntModel->index(index.row(),10,QModelIndex());
  var=targetIntModel->data(i,Qt::EditRole);
  bool isQCoefficient = var.toBool();
  i=targetIntModel->index(index.row(),11,QModelIndex());
  var=targetIntModel->data(i,Qt::EditRole);
  QList<double> qCoefficients = var.value<QList<double> >();

  AddTargetIntDialog aDialog;
  aDialog.setWindowTitle(tr("Edit an Experimental Effect Line"));
  aDialog.segmentsListText->setText(segmentsList);
  aDialog.numPointsSpin->setValue(numPoints);
  if(isConvolution)
    aDialog.isConvolutionCheck->setChecked(true);
  else aDialog.isConvolutionCheck->setChecked(false);
  aDialog.sigmaText->setText(sigma);
```

```
  if(isTargetIntegration) aDialog.isTargetIntegrationCheck->setChecked(true);
  else aDialog.isTargetIntegrationCheck->setChecked(false);
  aDialog.densityText->setText(density);
  aDialog.stoppingPowerEqText->setText(stoppingPowerEq);
  aDialog.tempParameters=parameters;
  aDialog.numParametersSpin->setValue(numParameters);
  if(isQCoefficient) aDialog.isQCoefficientCheck->setChecked(true);
  else aDialog.isQCoefficientCheck->setChecked(false);
  aDialog.tempQCoefficients=qCoefficients;
  aDialog.numQCoefficientSpin->setValue(qCoefficients.size());

  if(aDialog.exec()) {
    QString newSegmentsList = aDialog.segmentsListText->text();
    if(segmentsList!=newSegmentsList) {
      i=targetIntModel->index(index.row(),1,QModelIndex());
      targetIntModel->setData(i,newSegmentsList,Qt::EditRole);
    }
    int newNumPoints = aDialog.numPointsSpin->value();
    if(numPoints!=newNumPoints) {
      i=targetIntModel->index(index.row(),2,QModelIndex());
      targetIntModel->setData(i,newNumPoints,Qt::EditRole);
    }
    bool newIsConvolution=false;
    if(aDialog.isConvolutionCheck->isChecked()) newIsConvolution=true;
    if(isConvolution!=newIsConvolution) {
      i=targetIntModel->index(index.row(),3,QModelIndex());
      targetIntModel->setData(i,newIsConvolution,Qt::EditRole);
    }
    QString newSigma = aDialog.sigmaText->text();
    if(sigma!=newSigma) {
      i=targetIntModel->index(index.row(),4,QModelIndex());
      targetIntModel->setData(i,newSigma,Qt::EditRole);
    }
    bool newIsTargetIntegration=false;
    if(aDialog.isTargetIntegrationCheck->isChecked()) newIsTargetIntegration=true;
    if(isTargetIntegration!=newIsTargetIntegration) {
      i=targetIntModel->index(index.row(),5,QModelIndex());
      targetIntModel->setData(i,newIsTargetIntegration,Qt::EditRole);
    }
    QString newDensity = aDialog.densityText->text();
    if(density!=newDensity) {
      i=targetIntModel->index(index.row(),6,QModelIndex());
      targetIntModel->setData(i,newDensity,Qt::EditRole);
    }
    QString newStoppingPowerEq=aDialog.stoppingPowerEqText->text();
    if(stoppingPowerEq!=newStoppingPowerEq) {
      i=targetIntModel->index(index.row(),7,QModelIndex());
      targetIntModel->setData(i,newStoppingPowerEq,Qt::EditRole);
    }
    int newNumParameters=aDialog.numParametersSpin->value();
    if(numParameters!=newNumParameters) {
      i=targetIntModel->index(index.row(),8,QModelIndex());
      targetIntModel->setData(i,newNumParameters,Qt::EditRole);
    }
    QList<double> newParameters = aDialog.tempParameters;
    if(parameters!=newParameters||numParameters!=newNumParameters) {
      parameters.clear();
      for(int j=0;j<newNumParameters;j++)
    parameters.append(newParameters.at(j));
      i=targetIntModel->index(index.row(),9,QModelIndex());
      targetIntModel->setData(i,QVariant::fromValue<QList<double> >(parameters),Qt::EditRole);
    }
    bool newIsQCoefficient=false;
    if(aDialog.isQCoefficientCheck->isChecked()) newIsQCoefficient=true;
    if(isQCoefficient!=newIsQCoefficient) {
      i=targetIntModel->index(index.row(),10,QModelIndex());
      targetIntModel->setData(i,newIsQCoefficient,Qt::EditRole);
    }
    QList<double> newQCoefficients = aDialog.tempQCoefficients;
    if(qCoefficients!=newQCoefficients||qCoefficients.size()!=aDialog.numQCoefficientSpin->value()) {
      qCoefficients.clear();
      for(int j=0;j<aDialog.numQCoefficientSpin->value();j++)
    qCoefficients.append(newQCoefficients.at(j));
      i=targetIntModel->index(index.row(),11,QModelIndex());
      targetIntModel->setData(i,QVariant::fromValue<QList<double> >(qCoefficients),Qt::EditRole);
    }
  }
}

void TargetIntTab::deleteLine() {
  QItemSelectionModel *selectionModel = targetIntView->selectionModel();
  QModelIndexList indexes = selectionModel->selectedRows();
  QModelIndex index=indexes.at(0);

  targetIntModel->removeRows(index.row(),1,QModelIndex());
}
```

```
void TargetIntTab::updateButtons(const QItemSelection &selection) {
  QModelIndexList indexes=selection.indexes();

  if(indexes.isEmpty()) {
    deleteButton->setEnabled(false);
  } else {
    deleteButton->setEnabled(true);
  }
}

bool TargetIntTab::writeFile(QTextStream& outStream) {

  QList<TargetIntData> lines = targetIntModel->getLines();

  for(int i=0;i<lines.size();i++) {
    outStream «  qSetFieldWidth(15) « lines.at(i).isActive
          «  qSetFieldWidth(15) « '\"'+lines[i].segmentsList.remove(' ')+'\"'
          «  qSetFieldWidth(15) « lines.at(i).numPoints;
    if(lines.at(i).isConvolution) outStream « qSetFieldWidth(15) « '1';
    else outStream « qSetFieldWidth(15) « '0';
    outStream « qSetFieldWidth(15) « lines.at(i).sigma;
    if(lines.at(i).isTargetIntegration) outStream « qSetFieldWidth(15) « '1';
    else outStream « qSetFieldWidth(15) « '0';
    outStream « qSetFieldWidth(15) « lines.at(i).density
          « qSetFieldWidth(0) « " \""+lines[i].stoppingPowerEq.remove(' ')+"\" "
          « qSetFieldWidth(0) « lines.at(i).numParameters « qSetFieldWidth(0) « ' ';
    for(int j=0;j<lines.at(i).numParameters;j++)
      outStream « lines.at(i).parameters.at(j) «  qSetFieldWidth(0) « ' ';
    if(lines.at(i).isQCoefficients) outStream « qSetFieldWidth(0) « "              1";
    else outStream « qSetFieldWidth(0) « "              0";
    outStream « qSetFieldWidth(0) « "              " «  lines.at(i).qCoefficients.size() « ' ';
    for(int j=0;j<lines.at(i).qCoefficients.size();j++)
      outStream « qSetFieldWidth(0) «  lines.at(i).qCoefficients.at(j) « ' ';
    outStream«endl;
  }

  return true;
}

bool TargetIntTab::readFile(QTextStream& inStream) {

  int isActive;
  QString segmentsList;
  int numPoints;
  int isConvolution;
  double sigma;
  int isTargetIntegration;
  double density;
  QString stoppingPowerEq;
  int numParameters;
  QList<double> parameters;
  int numQCoefficients;
  QList<double> qCoefficients;
  int isQCoefficient;

  QString line("");
  while(!inStream.atEnd()&&line.trimmed()!=QString("</targetInt>")) {
    line=inStream.readLine();
    if(line.trimmed().isEmpty()) continue;
    if(!inStream.atEnd()&&line.trimmed()!=QString("</targetInt>")) {
      parameters.clear();
      qCoefficients.clear();
      QTextStream in(&line);
      in » isActive » segmentsList » numPoints » isConvolution » sigma » isTargetIntegration
     » density » stoppingPowerEq » numParameters;
      if(in.status()!=QTextStream::Ok) return false;
      int i=0;
      while(i<numParameters) {
    double tempParameter;
    in » tempParameter;
    parameters.append(tempParameter);
    i++;
      }
      if(in.status()!=QTextStream::Ok) return false;
      in » isQCoefficient;
      if(in.status()==QTextStream::Ok) {
    in » numQCoefficients;
    i=0;
    while(i<numQCoefficients) {
      double tempQCoefficient;
      in » tempQCoefficient;
      qCoefficients.append(tempQCoefficient);
      i++;
    }
    if(in.status()!=QTextStream::Ok) return false;
      } else {
    isQCoefficient=0;
```

```
      }
      bool tempIsQCoefficient=false;
      if(isQCoefficient==1) tempIsQCoefficient=true;
      bool tempIsConvolution=false;
      if(isConvolution==1) tempIsConvolution=true;
      bool tempIsTargetIntegration=false;
      if(isTargetIntegration==1) tempIsTargetIntegration=true;
      TargetIntData newLine =
{isActive,segmentsList.remove('\"'),numPoints,tempIsConvolution,sigma,tempIsTargetIntegration,
density,stoppingPowerEq.remove('\"'),numParameters,parameters,tempIsQCoefficient,qCoefficients};
      addLine(newLine);
    }
  }
  targetIntView->resizeRowsToContents();
  if(line.trimmed()!=QString("</targetInt>")) return false;
  return true;
}

void TargetIntTab::reset() {
  targetIntModel->removeRows(0,targetIntModel->getLines().size(),QModelIndex());
}

void TargetIntTab::showInfo(int which,QString title) {
  if(which<infoText.size()) {
    if(!infoDialog[which]) {
      infoDialog[which] = new InfoDialog(infoText[which],this,title);
      infoDialog[which]->setAttribute(Qt:: WA_DeleteOnClose);
      infoDialog[which]->show();
    } else infoDialog[which]->raise();
  }
}
```

8.135 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AChannel.h File Reference

Classes

- class AChannel

  An AZURE channel object.

8.136 AChannel.h

Go to the documentation of this file.

```
#ifndef ACHANNEL_H
#define ACHANNEL_H

class NucLine;


class AChannel {
 public:
  AChannel(NucLine, int);
  AChannel(int, double, int, char);
  int GetPairNum() const;
  int GetL() const;
  double GetS() const;
  double GetBoundaryCondition() const;
  char GetRadType() const;
  void SetBoundaryCondition(double);
 private:
  int l_;
  int pair_;
  double s_;
  char radtype_;
  double boundary_condition_;
};

#endif
```

8.137 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ALevel.h File Reference

`#include "Constants.h"`

Classes

- class ALevel

    An AZURE level object.

8.138 ALevel.h

Go to the documentation of this file.

```
#ifndef ALEVEL_H
#define ALEVEL_H

#include "Constants.h"

class NucLine;


class ALevel {
 public:
  ALevel(NucLine);
  ALevel(double);
  bool IsInRMatrix() const;
  bool EnergyFixed() const;
  bool ChannelFixed(int) const;
  bool IsECLevel() const;
  int NumNFIntegrals() const;
  int GetTransformIterations() const;
  int GetECPairNum() const;
  unsigned char GetECMultMask() const;
  double GetE() const;
  double GetGamma(int) const;
  double GetFitGamma(int) const;
  double GetFitE() const;
  double GetNFIntegral(int) const;
  double GetSqrtNFFactor() const;
  double GetECConversionFactor(int) const;
  double GetTransformGamma(int) const;
  double GetTransformE() const;
  double GetBigGamma(int) const;
  double GetShiftFunction(int) const;
  complex GetExternalGamma(int) const;
  void AddGamma( NucLine);
  void AddGamma(double);
  void SetGamma(int,double);
  void SetE(double);
  void SetFitGamma(int,double);
  void SetFitE(double);
  void AddNFIntegral(double);
  void SetSqrtNFFactor(double);
  void AddECConversionFactor(double);
  void SetTransformGamma(int,double);
  void SetTransformE(double);
  void SetBigGamma(int,double);
  void SetTransformIterations(int);
  void SetExternalGamma(int,complex);
  void SetShiftFunction(int,double);
  void SetECParams(int,unsigned char);
 private:
  bool isinrmatrix_;
  bool energyfixed_;
  bool isECLevel_;
  int transform_iter_;
  int ecPairNum_;
  unsigned char ecMultMask_;
  double level_e_;
  double fitlevel_e_;
  double sqrt_nf_factor_;
```

```
  double transform_e_;
  std::vector<bool> channelfixed_;
  vector_r gammas_;
  vector_r fitgammas_;
  vector_r nf_integrals_;
  vector_r ec_conv_factors_;
  vector_r transform_gammas_;
  vector_r big_gammas_;
  vector_r shifts_;
  vector_c external_gammas_;
};

#endif
```

8.139 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AMatrixFunc.h File Reference

`#include ¨GenMatrixFunc.h¨`

Classes

- class AMatrixFunc

  A function class to calculate the T-Matrix using the A-Matrix.

8.140 AMatrixFunc.h

Go to the documentation of this file.

```
#ifndef AMATRIXFUNC_H
#define AMATRIXFUNC_H

#include "GenMatrixFunc.h"


class AMatrixFunc : public GenMatrixFunc {
 public:
  AMatrixFunc(CNuc*, const Config &configure);
  CNuc *compound() const {return compound_;};
  const Config &configure() const {return configure_;};

  void ClearMatrices();
  void FillMatrices(EPoint*);
  void InvertMatrices();
  void CalculateTMatrix(EPoint*);
  void CalculateCrossSection();

  complex GetAMatrixElement(int,int,int) const;
  matrix_c *GetJSpecAInvMatrix(int);
  void AddAInvMatrixElement(int,int,int,complex);
  void AddAMatrix(matrix_c);
 private:
  const Config &configure_;
  CNuc *compound_;
  vector_matrix_c a_inv_matrices_;
  vector_matrix_c a_matrices_;
};

#endif
```

8.141 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AngCoeff.h File Reference

Classes

- class AngCoeff

  A container class for angular coupling coefficient functions.

8.142 AngCoeff.h

Go to the documentation of this file.

```
#ifndef ANGCOEFF_H
#define ANGCOEFF_H


class AngCoeff {
 public:
  static double ClebGord(double,double,double,double,double,double);
  static double Racah(double,double,double,double,double,double);
};

#endif
```

8.143 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZURECalc.h File Reference

```
#include ¨Minuit2/FCNBase.h¨
#include ¨Constants.h¨
#include <vector>
```

Classes

- class AZURECalc

  A function class to perform the calculation of the chi-squared value.

8.144 AZURECalc.h

Go to the documentation of this file.

```
#ifndef AZURECALC_H
#define AZURECALC_H

#include "Minuit2/FCNBase.h"
#include "Constants.h"
#include <vector>

class Config;
class EData;
class CNuc;


class AZURECalc : public ROOT::Minuit2::FCNBase {
 public:
  AZURECalc(EData* data,CNuc* compound, const Config& configure) : configure_(configure) {
    data_=data;
    compound_=compound;
  };

  ~AZURECalc() {};
  virtual double Up() const {return theErrorDef;};
  virtual double operator()(const vector_r&) const;

  const Config &configure() const {return configure_;};
  EData *data() const {return data_;};
  CNuc *compound() const {return compound_;};

  void SetErrorDef(double def) {theErrorDef=def;};
 private:
  const Config &configure_;
  EData *data_;
  CNuc *compound_;
  double theErrorDef;
};

#endif
```

8.145 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREFBuffer.h File Reference

```
#include <fstream>
#include <string>
#include <assert.h>
```

Classes

- class AZUREFBuffer

  A container class for a pointer to a file buffer.

8.146 AZUREFBuffer.h

Go to the documentation of this file.

```
#ifndef AZUREFBUFFER_H
#define AZUREFBUFFER_H

#include <fstream>
#include <string>
#include <assert.h>


class AZUREFBuffer {
 public:
 AZUREFBuffer(int entranceKey,int exitKey,std::string outputdir,bool isExtrap,bool isAngDist) :
  isAngDist_(isAngDist) {
    char filename[256];
    entrancekey_=entranceKey;
    exitkey_=exitKey;
    if(exitkey_==-1) {
      if(!isExtrap) sprintf(filename,"%sAZUREOut_aa=%d_TOTAL_CAPTURE.out",outputdir.c_str(),entranceKey);
      else sprintf(filename,"%sAZUREOut_aa=%d_TOTAL_CAPTURE.extrap",outputdir.c_str(),entranceKey);
    } else {
      if(!isExtrap) sprintf(filename,"%sAZUREOut_aa=%d_R=%d.out",outputdir.c_str(),entranceKey,exitKey);
      else {
    if(!isAngDist) sprintf(filename,"%sAZUREOut_aa=%d_R=%d.extrap",outputdir.c_str(),entranceKey,exitKey);
    else sprintf(filename,"%sAZUREOut_aa=%d_R=%d.acoeff",outputdir.c_str(),entranceKey,exitKey);
      }
    }
    fbuffer_=new std::filebuf;
    fbuffer_->open(filename,std::ios::out);
    assert(fbuffer_->is_open());
  };
  ~AZUREFBuffer() {
    fbuffer_->close();
    delete fbuffer_;
  };
  bool IsAngDist() const {return isAngDist_;};
  int GetEntranceKey() const {return entrancekey_;};
  int GetExitKey() const {return exitkey_;}
  std::filebuf *GetFBuffer() {return fbuffer_;};
 private:
  bool isAngDist_;
  int entrancekey_;
  int exitkey_;
  std::filebuf *fbuffer_;
};

#endif
```

8.147 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREMain.h File Reference

```
#include ¨CNuc.h¨
#include ¨EData.h¨
```

Classes

- class AZUREMain

  The top-level AZURE function class.

8.148 AZUREMain.h

Go to the documentation of this file.

```
#ifndef AZUREMAIN_H
#define AZUREMAIN_H

#include "CNuc.h"
#include "EData.h"


class AZUREMain {
 public:
  AZUREMain(const Config &configure) : configure_(configure) {
    compound_ = new CNuc;
    data_ = new EData;
  };
  ~AZUREMain() {
    delete compound_;
    delete data_;
  };
  int operator()();
  const Config &configure() const {return configure_;};
  CNuc *compound() const {return compound_;};
  EData *data() const {return data_;};
 private:
  const Config &configure_;
  CNuc *compound_;
  EData *data_;
};

#endif
```

8.149 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREOutput.h File Reference

```
#include ¨AZUREFBuffer.h¨
#include <vector>
```

Classes

- class AZUREOutput

  A class to assist in writing AZURE output files.

8.150 AZUREOutput.h

Go to the documentation of this file.

```
#ifndef AZUREOUTPUT_H
#define AZUREOUTPUT_H

#include "AZUREFBuffer.h"
#include <vector>

class AZUREOutput {
 public:
  AZUREOutput(std::string);
  ~AZUREOutput();
  bool IsExtrap() const;
  std::filebuf *operator()(int entranceKey, int exitKey, bool isAngDist=false);
  int NumAZUREFBuffers() const;
  int IsAZUREFBuffer(int,int,bool);
  std::string GetOutputDir() const;
  void AddAZUREFBuffer(AZUREFBuffer*);
  void SetExtrap();
  AZUREFBuffer *GetAZUREFBuffer(int);
 private:
  bool is_extrap_;
  std::string outputdir_;
  std::vector<AZUREFBuffer*> azurefbuffers_;
};

#endif
```

8.151 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREParams.h File Reference

```
#include <Minuit2/MnUserParameters.h>
#include <iostream>
#include <fstream>
#include <iomanip>
#include "Constants.h"
```

Classes

- class AZUREParams

  A container class to hold Minuit parameters in AZURE.

8.152 AZUREParams.h

Go to the documentation of this file.

```
#ifndef AZUREPARAMS_H
#define AZUREPARAMS_H

#include <Minuit2/MnUserParameters.h>
#include <iostream>
#include <fstream>
#include <iomanip>
#include "Constants.h"

class Config;

class AZUREParams {
 public:
  ROOT::Minuit2::MnUserParameters &GetMinuitParams();
  void ReadUserParameters(const Config&);
  void WriteUserParameters(const Config&,bool);
  void WriteParameterErrors(const std::vector<std::pair<double,double> >&,const Config&);
 private:
  ROOT::Minuit2::MnUserParameters params_;
};

#endif
```

8.153 /Users/kuba/Desktop/R-Matrix/AZURE2/include/CNuc.h File Reference

```
#include <string>
#include <map>
#include ¨JGroup.h¨
#include ¨PPair.h¨
```

Classes

- class CNuc
  An AZURE compound nucleus.

Namespaces

- namespace ROOT
- namespace ROOT::Minuit2

Functions

- double DoubleFactorial (int)

8.153.1 Function Documentation

8.153.1.1 DoubleFactorial()

```
double DoubleFactorial (
              int a )
```

Definition at line 1 of file DoubleFactorial.cpp.

8.154 CNuc.h

Go to the documentation of this file.

```
#ifndef CNUC_H
#define CNUC_H

#include <string>
#include <map>
#include "JGroup.h"
#include "PPair.h"

namespace ROOT {
  namespace Minuit2 {
    class MnUserParameters;
  }
}
class Config;


class CNuc {
public:
  bool IsPairKey(int);
  int NumPairs() const;
```

```
  int NumJGroups() const;
  int IsPair(PPair);
  int IsJGroup(JGroup);
  int GetPairNumFromKey(int);
  int Fill(const Config&);
  void ParseExternalCapture(const Config&,std::map<int,int>&);
  int GetMaxLValue() const;
  void Initialize(const Config&);
  void AddPair(PPair);
  void AddJGroup(JGroup);
  void PrintNuc(const Config&);
  void TransformIn(const Config&);
  void SortPathways(const Config&);
  void PrintPathways(const Config&);
  void CalcBoundaryConditions(const Config&);
  void PrintBoundaryConditions(const Config&);
  void CalcAngularDists(int);
  void PrintAngularDists(const Config&);
  void FillMnParams(ROOT::Minuit2::MnUserParameters&);
  void FillCompoundFromParams(const vector_r &);
  void TransformOut(const Config&);
  void PrintTransformParams(const Config&);
  void SetMaxLValue(int);
  void CalcShiftFunctions(const Config&);
  complex CalcExternalWidth(JGroup*, ALevel*,AChannel*,bool,const Config&);
  PPair *GetPair(int);
  JGroup *GetJGroup(int);
  CNuc *Clone() const;
private:
  std::vector<PPair> pairs_;
  std::vector<JGroup> jgroups_;
  int maxLValue_;
};

extern double DoubleFactorial(int);

#endif
```

8.155 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Config.h File Reference

```
#include <string>
#include <fstream>
```

Classes

- struct RateParams

  A structure holding the reaction rate calculation configuration.

- class Config

  A configuration structure for AZURE.

8.156 Config.h

Go to the documentation of this file.

```
#ifndef CONFIG_H
#define CONFIG_H

#include <string>
#include <fstream>


struct RateParams {
  bool useFile;
  std::string temperatureFile;
  int entrancePair;
  int exitPair;
```

```
  double minTemp;
  double maxTemp;
  double tempStep;
};


class Config {
 public:
  Config(std::ostream& stream);
  void Reset();
  enum ParameterFlags {
    USE_AMATRIX                 =(1«0),
    PERFORM_ERROR_ANALYSIS      =(1«1),
    PERFORM_FIT                 =(1«2),
    CALCULATE_WITH_DATA         =(1«3),
    USE_PREVIOUS_PARAMETERS     =(1«4),
    USE_EXTERNAL_CAPTURE        =(1«5),
    USE_PREVIOUS_INTEGRALS      =(1«6),
    CALCULATE_REACTION_RATE     =(1«7),
    TRANSFORM_PARAMETERS        =(1«8),
    USE_BRUNE_FORMALISM         =(1«9),
    IGNORE_ZERO_WIDTHS          =(1«10),
    USE_RMC_FORMALISM           =(1«11),
    USE_GSL_COULOMB_FUNC        =(1«12),
    USE_LONGWAVELENGTH_APPROX   =(1«13)
  };
  enum CheckFileFlags {
    CHECK_COMPOUND_NUCLEUS      =(1«0),
    CHECK_PATHWAYS              =(1«1),
    CHECK_DATA                  =(1«2),
    CHECK_ENERGY_DEP            =(1«3),
    CHECK_LEGENDRE              =(1«4),
    CHECK_BOUNDARY_CONDITIONS   =(1«5),
    CHECK_ANGULAR_DISTS         =(1«6),
    CHECK_COUL_AMPLITUDES       =(1«7)
  };
  std::ostream &outStream;
  std::string configfile;
  bool stopFlag;
  unsigned int paramMask;
  unsigned int screenCheckMask;
  unsigned int fileCheckMask;
  double chiVariance;
  std::string outputdir;
  std::string checkdir;
  std::string paramfile;
  std::string integralsfile;
  RateParams  rateParams;
  static const int maxLOrder=20;
  int ReadConfigFile();
#ifndef NO_STAT
  int CheckForInputFiles();
#endif
};

#endif
```

8.157 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Constants.h File Reference

```
#include <complex>
#include <vector>
```

Typedefs

- typedef std::complex< double > complex
- typedef std::vector< double > vector_r
- typedef std::vector< std::complex< double > > vector_c
- typedef std::vector< std::vector< double > > matrix_r
- typedef std::vector< std::vector< std::complex< double > > > matrix_c
- typedef std::vector< std::vector< std::vector< double > > > vector_matrix_r
- typedef std::vector< std::vector< std::vector< std::complex< double > > > > vector_matrix_c

Variables

- const double pi =3.141592650
- const double hbarc =197.32696310
- const double uconv =931.4940880
- const double fstruc =1.00/137.0359996790
- const double boltzConst =8.6171e-2
- const double lightSpeedInCmPerS =29979245800.
- const double avagadroNum =6.02214179e23
- const double nuclearMagneton =0.105155
- const unsigned char isE1 = 1 << 0
- const unsigned char isM1 = 1 << 1
- const unsigned char isE2 = 1 << 2
- const int maxECMult =2

8.157.1 Typedef Documentation

8.157.1.1 complex

```
typedef std::complex<double> complex
```

Definition at line 20 of file Constants.h.

8.157.1.2 matrix_c

```
typedef std::vector<std::vector<std::complex<double> > > matrix_c
```

Definition at line 24 of file Constants.h.

8.157.1.3 matrix_r

```
typedef std::vector<std::vector<double> > matrix_r
```

Definition at line 23 of file Constants.h.

8.157.1.4 vector_c

```
typedef std::vector<std::complex<double> > vector_c
```

Definition at line 22 of file Constants.h.

8.157.1.5 vector_matrix_c

```
typedef std::vector<std::vector<std::vector<std::complex<double> > > > vector_matrix_c
```

Definition at line 26 of file Constants.h.

8.157.1.6 vector_matrix_r

```
typedef std::vector<std::vector<std::vector<double> > > vector_matrix_r
```

Definition at line 25 of file Constants.h.

8.157.1.7 vector_r

```
typedef std::vector<double> vector_r
```

Definition at line 21 of file Constants.h.

8.157.2 Variable Documentation

8.157.2.1 avagadroNum

```
const double avagadroNum =6.02214179e23
```

Definition at line 13 of file Constants.h.

8.157.2.2 boltzConst

```
const double boltzConst =8.6171e-2
```

Definition at line 11 of file Constants.h.

8.157.2.3 fstruc

```
const double fstruc =1.00/137.0359996790
```

Definition at line 10 of file Constants.h.

8.157.2.4 hbarc

```
const double hbarc =197.32696310
```

Definition at line 8 of file Constants.h.

8.157.2.5 isE1

```
const unsigned char isE1 = 1 << 0
```

Definition at line 15 of file Constants.h.

8.157.2.6 isE2

```
const unsigned char isE2 = 1 << 2
```

Definition at line 17 of file Constants.h.

8.157.2.7 isM1

```
const unsigned char isM1 = 1 << 1
```

Definition at line 16 of file Constants.h.

8.157.2.8 lightSpeedInCmPerS

```
const double lightSpeedInCmPerS =29979245800.
```

Definition at line 12 of file Constants.h.

8.157.2.9 maxECMult

```
const int maxECMult =2
```

Definition at line 18 of file Constants.h.

8.157.2.10 nuclearMagneton

```
const double nuclearMagneton =0.105155
```

Definition at line 14 of file Constants.h.

8.157.2.11 pi

```
const double pi =3.141592650
```

Definition at line 7 of file Constants.h.

8.157.2.12 uconv

```
const double uconv =931.4940880
```

Definition at line 9 of file Constants.h.

8.158 Constants.h

Go to the documentation of this file.
```
#ifndef CONSTANTS_H
#define CONSTANTS_H

#include <complex>
#include <vector>

const double pi=3.141592650;
const double hbarc=197.32696310;
const double uconv=931.4940880;
const double fstruc=1.00/137.0359996790;
const double boltzConst=8.6171e-2;
const double lightSpeedInCmPerS=29979245800.;
const double avagadroNum=6.02214179e23;
const double nuclearMagneton=0.105155;
const unsigned char isE1 = 1 « 0;
const unsigned char isM1 = 1 « 1;
const unsigned char isE2 = 1 « 2;
const int maxECMult=2;

typedef std::complex<double> complex;
typedef std::vector<double> vector_r;
typedef std::vector<std::complex<double> > vector_c;
typedef std::vector<std::vector<double> > matrix_r;
typedef std::vector<std::vector<std::complex<double> > > matrix_c;
typedef std::vector<std::vector<std::vector<double> > > vector_matrix_r;
typedef std::vector<std::vector<std::vector<std::complex<double> > > > vector_matrix_c;

#endif
```

8.159 /Users/kuba/Desktop/R-Matrix/AZURE2/include/CoulFunc.h File Reference

Classes

- struct CoulWaves

  The return structure of the CoulFunc function class.
- class CoulFunc

  A function class to calculate Coulomb functions for positive energy channels.

8.160 CoulFunc.h

Go to the documentation of this file.
```
#ifndef COULFUNC_H
#define COULFUNC_H

class PPair;


struct CoulWaves {
  double F;
  double dF;
  double G;
  double dG;
};


class CoulFunc {
 public:
  CoulFunc(PPair *pPair, bool useGSLFunctions);
  int z1() const;
  int z2() const;
  double redmass() const;
  int lLast() const;
  double radiusLast() const;
  double energyLast() const;
```

```
  struct CoulWaves coulLast() const;
  void setLast(int, double, double, CoulWaves);
  CoulWaves operator()(int,double,double);
  double Penetrability(int,double,double);
  double PEShift(int,double,double);
  double PEShift_dE(int,double,double);;
 private:
  static double thisPEShift(double,void*);
  typedef struct DEShiftParams {
    CoulFunc *coulFunc;
    int lValue;
    double radius;
  } DEShiftParams;
  DEShiftParams dEShiftParams_;
  bool useGSLFunctions_;
  int z1_;
  int z2_;
  int lLast_;
  double redmass_;
  double radiusLast_;
  double energyLast_;
  struct CoulWaves coulLast_;
};

#endif
```

8.161 /Users/kuba/Desktop/R-Matrix/AZURE2/include/DataLine.h File Reference

```
#include <fstream>
```

Classes

- class DataLine

  A class to read and store a line from a data file.

8.162 DataLine.h

Go to the documentation of this file.

```
#ifndef DATALINE_H
#define DATALINE_H

#include <fstream>


class DataLine {
 public:
  DataLine(std::ifstream &stream) {
    stream » energy_ » angle_ » crossSection_ » error_;
  };
  double angle() const {return angle_;};
  double energy() const {return energy_;};
  double crossSection() const {return crossSection_;};
  double error() const {return error_;};
 private:
  double angle_;
  double energy_;
  double crossSection_;
  double error_;
};

#endif
```

8.163 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Decay.h File Reference

```
#include ¨KGroup.h¨
#include ¨KLGroup.h¨
```

Classes

- class Decay

  An AZURE decay pair.

8.164 Decay.h

Go to the documentation of this file.

```
#ifndef DECAY_H
#define DECAY_H

#include "KGroup.h"
#include "KLGroup.h"


class Decay {
 public:
  Decay(int);
  int GetPairNum() const;
  int NumKGroups() const;
  int NumKLGroups() const;
  int IsKGroup(KGroup);
  int IsKLGroup(KLGroup) ;
  void AddKGroup(KGroup);
  void AddKLGroup(KLGroup);
  KGroup *GetKGroup(int);
  KLGroup *GetKLGroup(int);
 private:
  int pair_;
  std::vector<KGroup> kgroups_;
  std::vector<KLGroup> klgroups_;
};


#endif
```

8.165 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ECIntegral.h File Reference

```
#include ¨CoulFunc.h¨
#include ¨WhitFunc.h¨
#include ¨Config.h¨
```

Classes

- class ECIntegral

  A function class to calculate external capture integrals.

8.166 ECIntegral.h

Go to the documentation of this file.
```
#ifndef ECINTEGRAL_H
#define ECINTEGRAL_H

#include "CoulFunc.h"
#include "WhitFunc.h"
#include "Config.h"

class EffectiveCharge;


class ECIntegral {
 public:
  ECIntegral(PPair *pPair, const Config& configure) {
    params_.coulFunc = new CoulFunc(pPair,!!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
    params_.whitFunc = new WhitFunc(pPair);
    params_.useLongWavelengthApprox = !!(configure.paramMask&Config::USE_LONGWAVELENGTH_APPROX);
    pair_ = pPair;
  };
  ~ECIntegral() {
    delete params_.coulFunc;
    delete params_.whitFunc;
  };
  complex operator()(int,int,double,double,double,double,int,char,double,double,bool);
 private:
  void ResetIntegrals() {FW_=0.;GW_=0.;};
  void Integrate(double);
  static double FWIntegrand(double,void*);
  static double GWIntegrand(double,void*);
  static double WWIntegrand(double,void*);
  CoulFunc *coulfunction() const {return params_.coulFunc;};
  WhitFunc *whitfunction() const {return params_.whitFunc;};
  PPair *pair() const {return pair_;};
  double FW() const {return FW_;};
  double GW() const {return GW_;};
  typedef struct Params {
    EffectiveCharge* effectiveCharge;
    CoulFunc *coulFunc;
    WhitFunc *whitFunc;
    int liValue;
    int lfValue;
    int multLValue;
    double pairEnergy;
    double bindingEnergy;
    bool useLongWavelengthApprox;
  } Params;
  Params params_;
  PPair *pair_;
  double FW_;
  double GW_;
};




#endif
```

8.167 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ECMGroup.h File Reference

`#include "Constants.h"`

Classes

- class ECMGroup

  An AZURE external reaction pathway.

8.168 ECMGroup.h

Go to the documentation of this file.

```
#ifndef ECMGROUP_H
#define ECMGROUP_H

#include "Constants.h"


class ECMGroup {
 public:
  ECMGroup(char, int, int, double, int, int, int);
  ECMGroup(char, int, int, double, int, int, int, int, int, int, int);
  bool IsChannelCapture() const;
  char GetRadType() const;
  int GetMult() const;
  int GetL() const;
  int GetFinalChannel() const;
  int GetJGroupNum() const;
  int GetLevelNum() const;
  int GetChanCapDecay() const;
  int GetChanCapKGroup() const;
  int GetChanCapMGroup() const;
  int GetIntChannelNum() const;
  double GetJ() const;
  double GetStatSpinFactor() const;
  void SetStatSpinFactor(double);
 private:
  char radtype_;
  int mult_;
  int li_;
  int chf_;
  int jGroupNum_;
  int levelNum_;
  bool ischancap_;
  int chdecay_;
  int chkgroup_;
  int chmgroup_;
  int internalChannel_;
  double ji_;
  double statspinfactor_;
  complex tmatrix_;
};

#endif
```

8.169 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EData.h File Reference

```
#include ¨ESegment.h¨
#include ¨TargetEffect.h¨
#include ¨EDataIterator.h¨
```

Classes

- class EData

  An AZURE data object.

Namespaces

- namespace ROOT
- namespace ROOT::Minuit2

8.170 EData.h

Go to the documentation of this file.

```
#ifndef EDATA_H
#define EDATA_H

#include "ESegment.h"
#include "TargetEffect.h"
#include "EDataIterator.h"

class CNuc;
namespace ROOT {
  namespace Minuit2 {
    class MnUserParameters;
  }
}


class EData {
 public:
  EData();
  int NumSegments() const;
  int Fill(const Config&,CNuc*);
  int MakePoints(const Config&,CNuc*);
  int Iterations() const;
  int NumTargetEffects() const;
  int GetNormParamOffset() const;
  int ReadTargetEffectsFile(const Config&,CNuc*);
  bool IsFit() const;
  bool IsErrorAnalysis() const;
  bool IsSegmentKey(int);
  void SetFit(bool);
  void SetErrorAnalysis(bool);
  void Iterate();
  void ResetIterations();
  int Initialize(CNuc*,const Config&);
  void AddSegment(ESegment);
  void PrintData(const Config&);
  void CalcLegendreP(int);
  void PrintLegendreP(const Config&);
  int CalcEDependentValues(CNuc*,const Config&);
  void PrintEDependentValues(const Config&,CNuc*);
  void CalcCoulombAmplitude(CNuc*);
  void PrintCoulombAmplitude(const Config&,CNuc*);
  void WriteOutputFiles(const Config&,bool=false);
  int CalculateECAmplitudes(CNuc*,const Config&);
  void MapData();
  void AddTargetEffect(TargetEffect);
  void SetNormParamOffset(int);
  void FillMnParams(ROOT::Minuit2::MnUserParameters&);
  void FillNormsFromParams(const vector_r &);
  void DeleteLastSegment();
  ESegment *GetSegment(int);
  ESegment *GetSegmentFromKey(int);
  EData *Clone() const;
  TargetEffect *GetTargetEffect(int);
  EDataIterator begin();
  EDataIterator end();
  std::vector<ESegment>& GetSegments();
 private:
  std::vector<TargetEffect> targetEffects_;
  std::vector<ESegment> segments_;
  int iterations_;
  int normParamOffset_;
  bool isFit_;
  bool isErrorAnalysis_;
};

#endif
```

8.171 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EDataIterator.h File Reference

`#include <vector>`

Classes

- class EDataIterator

  An iterator class for an EData object.

Typedefs

- typedef std::vector< EPoint >::iterator EPointIterator
- typedef std::vector< EPoint * >::iterator EPointMapIterator
- typedef std::vector< ESegment >::iterator ESegmentIterator

8.171.1 Typedef Documentation

8.171.1.1 EPointIterator

```
typedef std::vector<EPoint>::iterator EPointIterator
```

Definition at line 9 of file EDataIterator.h.

8.171.1.2 EPointMapIterator

```
typedef std::vector<EPoint*>::iterator EPointMapIterator
```

Definition at line 10 of file EDataIterator.h.

8.171.1.3 ESegmentIterator

```
typedef std::vector<ESegment>::iterator ESegmentIterator
```

Definition at line 11 of file EDataIterator.h.

8.172 EDataIterator.h

Go to the documentation of this file.

```
00001 #ifndef EDATAITERATOR_H
00002 #define EDATAITERATOR_H
00003 
00004 #include <vector>
00005 
00006 class EPoint;
00007 class ESegment;
00008 
00009 typedef std::vector<EPoint>::iterator EPointIterator;
00010 typedef std::vector<EPoint*>::iterator EPointMapIterator;
00011 typedef std::vector<ESegment>::iterator ESegmentIterator;
00012 
00014 
00020 class EDataIterator {
00021  public:
00022   EDataIterator(std::vector<ESegment>*);
00023   EDataIterator(const EDataIterator& it);
00024   EDataIterator& operator++();
00025   EDataIterator operator++(int);
00026   bool operator==(const EDataIterator&);
00027   bool operator!=(const EDataIterator&);
00028   EDataIterator& SetEnd();
00029   ESegmentIterator& segment();
00030   EPointIterator& point();
00031 
00032  private:
00033   std::vector<ESegment>* segments_;
00034   ESegmentIterator segmentIterator_;
00035   EPointIterator pointIterator_;
00036 };
00037 
00038 #endif
```

8.173 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EffectiveCharge.h File Reference

Classes

- class EffectiveCharge

  A function class for calculating effective charge without long-wavelength approximation.

8.174 EffectiveCharge.h

Go to the documentation of this file.

```
#ifndef EFFECTIVECHARGE_H
#define EFFECTIVECHARGE_H

class PPair;


class EffectiveCharge {
 public:
  EffectiveCharge(PPair*,double, int);
  double operator()(double);
 private:
  static double Integrand(double,void*);
  int z1_;
  int z2_;
  int L_;
  double m1_;
  double m2_;
  double energy_;
};

#endif
```

8.175 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EigenFunc.h File Reference

```
#include ¨Constants.h¨
```

Classes

- class EigenFunc

  A function class to solve a eigenvalue problems.

8.176 EigenFunc.h

Go to the documentation of this file.

```
#ifndef EIGENFUNC_H
#define EIGENFUNC_H

#include "Constants.h"


class EigenFunc {
 public:
  EigenFunc(const matrix_r&);
  EigenFunc(const matrix_r&, const std::vector<vector_r >&);
  const vector_r& eigenvalues() const {return eigenvalues_;};
  const matrix_r& eigenvectors() const {return eigenvectors_;};
 private:
  vector_r eigenvalues_;
  matrix_r eigenvectors_;
};

#endif
```

8.177 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EPoint.h File Reference

```
#include ¨Constants.h¨
```

Classes

- struct EnergyMap

    A container structure for a reference to a data point.
- class EPoint

    An AZURE data point.

8.178 EPoint.h

Go to the documentation of this file.

```
#ifndef EPOINT_H
#define EPOINT_H

#include "Constants.h"


struct EnergyMap {
  int segment;
  int point;
};

class ESegment;
class EData;
class CNuc;
class PPair;
class TargetEffect;
class DataLine;
class Config;


class EPoint {
 public:
  EPoint(DataLine, ESegment*);
  EPoint(double, double, ESegment*);
  EPoint(double, double, int, int, bool, bool, bool, double, int, int);
  bool IsDifferential() const;
  bool IsPhase() const;
  bool IsMapped() const;
  bool IsTargetEffect() const;
  bool IsAngularDist() const;
  int GetEntranceKey() const;
  int GetExitKey() const;
  int GetMaxLOrder() const;
  int GetL() const;
  int NumLocalMappedPoints() const;
  int NumSubPoints() const;
  int GetTargetEffectNum() const;
  int GetMaxAngDistOrder() const;
  int GetNumAngularDists() const;
  double GetLabAngle() const;
  double GetCMAngle() const;
  double GetLabEnergy() const;
  double GetCMEnergy() const;
  double GetExcitationEnergy() const;
  double GetLegendreP(int) const;
  double GetLabCrossSection() const;
  double GetCMCrossSection() const;
  double GetLabCrossSectionError() const;
  double GetCMCrossSectionError() const;
  double GetGeometricalFactor() const;
  double GetFitCrossSection() const;
  double GetSFactorConversion() const;
  double GetSqrtPenetrability(int,int) const;
  double GetJ() const;
  double GetStoppingPower() const;
```

```
  double GetTargetThickness() const;
  double GetAngularDist(int) const;
  complex GetLoElement(int,int) const;
  complex GetExpCoulombPhase(int,int) const;
  complex GetExpHardSpherePhase(int,int) const;
  complex GetCoulombAmplitude() const;
  complex GetECAmplitude(int,int) const;
  EnergyMap GetMap() const;
  void Initialize(CNuc*,const Config&);
  void ConvertLabEnergy(PPair*);
  void ConvertDecayEnergy(PPair*);
  void ConvertLabAngle(PPair*);
  void ConvertLabAngle(PPair*,PPair*,const Config&);
  void ConvertCrossSection(PPair*,PPair*);
  void AddLegendreP(double);
  void SetGeometricalFactor(double);
  void SetFitCrossSection(double);
  void SetSFactorConversion(double);
  void SetExitKey(int);
  void CalcLegendreP(int,TargetEffect*);
  void CalcEDependentValues(CNuc*,const Config&);
  void AddLoElement(int,int,complex);
  void AddSqrtPenetrability(int,int,double);
  void AddExpCoulombPhase(int,int,complex);
  void AddExpHardSpherePhase(int,int,complex);
  void CalcCoulombAmplitude(CNuc*);
  void SetCoulombAmplitude(complex);
  void CalculateECAmplitudes(CNuc*,const Config&);
  void AddECAmplitude(int,int,complex);
  void Calculate(CNuc*,const Config &configure,EPoint* parent=NULL, int subPointNum=0);
  void SetMap(int,int);
  void AddLocalMappedPoint(EPoint*);
  void ClearLocalMappedPoints();
  void SetTargetEffectNum(int);
  void AddSubPoint(EPoint);
  void IntegrateTargetEffect();
  void SetParentData(EData*);
  void SetStoppingPower(double);
  void SetTargetThickness(double);
  void SetAngularDists(vector_r);
  EData *GetParentData() const;
  EPoint* GetLocalMappedPoint(int) const;
  EPoint* GetSubPoint(int);
  std::vector<EPoint>& GetSubPoints();
  std::vector<EPoint*>& GetMappedPoints();
 private:
  bool is_differential_;
  bool is_phase_;
  bool is_mapped_;
  bool is_ang_dist_;
  int entrance_key_;
  int exit_key_;
  int l_value_;
  int targetEffectNum_;
  int max_ang_dist_order_;
  double cm_angle_;
  double lab_angle_;
  double cm_energy_;
  double lab_energy_;
  double excitation_energy_;
  double cm_crosssection_;
  double cm_dcrosssection_;
  double lab_crosssection_;
  double lab_dcrosssection_;
  double geofactor_;
  double fitcrosssection_;
  double sfactorconv_;
  double j_value_;
  double stoppingPower_;
  double targetThickness_;
  struct EnergyMap energy_map_;
  complex coulombamplitude_;
  vector_r legendreP_;
  vector_r angularDists_;
  matrix_c lo_elements_;
  matrix_r penetrabilities_;
  matrix_c coulombphase_;
  matrix_c hardspherephase_;
  matrix_c ec_amplitudes_;
  std::vector<EPoint*> local_mapped_points_;
  std::vector<EPoint> integrationPoints_;
  EData* parentData_;
};

#endif
```

8.179 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Equation.h File Reference

```
#include <string>
#include <vector>
#include <exception>
#include <sstream>
#include <map>
```

Classes

- class GenericFunction

  A wrapper class for function pointers used by Equation class.
- class Equation

  A class for parsing algebraic expressions.
- class SyntaxError

  An exception class thrown by the Equation class.

8.180 Equation.h

Go to the documentation of this file.

```
#ifndef EQUATION_H
#define EQUATION_H

#include <string>
#include <vector>
#include <exception>
#include <sstream>
#include <map>

class Config;


class GenericFunction {
 public:
  GenericFunction() {};
  GenericFunction(double (*function)(double)) :
   function_(function) {};
  double Evaluate(double value) const {
    return (*function_)(value);
  };
 private:
  double (*function_)(double);
};


class Equation {
 public:
  Equation();
  Equation(std::string equation, int numParams,const Config&);
  Equation(std::string equation,std::vector<double> parameters,const Config&);
  Equation(std::string equation,double parameters[],size_t arraySize,const Config&);
  void Initialize(std::string equation, int numParams, const Config &);
  void SetParameter(unsigned int index, double value, const Config&);
  std::vector<double> GetParameters() const;
  double Evaluate(const Config&, double x=0.0) const;
 private:
  enum Associativity {LEFT,RIGHT};
  enum TokenType {NUMBER=1,OPERATOR=2,VARIABLE=4,PARAMETER=8,LEFTPAR=16,RIGHTPAR=32,FUNCTION=64};
  enum OperatorType {ADD=0,SUBTRACT=0,MULT=1,DIVIDE=1,POWER=2,BADTYPE=10};
  typedef std::pair<TokenType,std::string> TokenPair;
  void BuildFunctionList();
  void Parse(const Config&);
  bool IsOperator(char) const;
  bool IsDigit(char) const;
  unsigned int FindFunction(unsigned int &position);
  TokenPair GetToken(unsigned int &position, const Config&);
```

```
  OperatorType GetOperatorType(char) const;
  Associativity GetOperatorAssociativity(char) const;
  std::string BinaryOperation(double left, double right, char op, const Config&) const;
  double FunctionOperation(TokenPair token, double x, const Config&) const;
  double GetTokenValue(TokenPair token, double x,const Config&) const;
  std::string infixEquation_;
  std::vector<TokenPair> output_;
  std::vector<double> parameters_;
  std::vector<Equation> subEquations_;
  std::map<std::string,GenericFunction> functionList_;
};


class SyntaxError : public std::exception {
 public:
  SyntaxError(std::string equation, int type, int position=-1) {
    std::string typeMessage = GetTypeMessage(type);
    std::ostringstream stm;
    stm « "Syntax Error in " « equation «  ':' «  typeMessage « std::endl;
    if(position!=-1) {
      stm « "                    "; for(int i=0;i<position;i++) stm « ' '; stm « '^';
    }
    messageString_=stm.str();
    message_=messageString_.c_str();
  };
  ~SyntaxError() throw() {
    delete[] message_;
  };
  virtual const char* what() const throw() {
    return message_;
  };
 private:
  std::string GetTypeMessage(int type) {
    switch(type) {
      case 0: return " Unknown token type.";
      case 1: return " Parameter out of range.";
      case 2: return " Mismatched Parentheses.";
      case 3: return " Unexpected token.";
      default: return " Unknown error.";
    }
  };
  std::string messageString_;
  const char* message_;
};


#endif
```

8.181 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ESegment.h File Reference

#include "EPoint.h"

Classes

- class ESegment

    An AZURE data segment.

8.182 ESegment.h

Go to the documentation of this file.

```
#ifndef ESEGMENT_H
#define ESEGMENT_H

#include "EPoint.h"

class EData;
class ExtrapLine;
```

```
class SegLine;


class ESegment {
 public:
  ESegment(SegLine);
  ESegment(ExtrapLine);
  bool IsInSegment(EPoint);
  bool IsDifferential() const;
  bool IsPhase() const;
  bool IsTargetEffect() const;
  bool IsVaryNorm() const;
  bool IsAngularDist() const;
  int IsTotalCapture() const;
  int NumPoints() const;
  int GetEntranceKey() const;
  int GetExitKey() const;
  int Fill(CNuc*,EData*,const Config&);
  int GetL() const;
  int GetTargetEffectNum() const;
  int GetSegmentKey() const;
  int GetMaxAngDistOrder() const;
  double GetMinEnergy() const;
  double GetMaxEnergy() const;
  double GetMinAngle() const;
  double GetMaxAngle() const;
  double GetSegmentChiSquared() const;
  double GetEStep() const;
  double GetAStep() const;
  double GetJ() const;
  double GetNorm() const;
  double GetNominalNorm() const;
  double GetNormError() const;
  std::string GetDataFile() const;
  void AddPoint(EPoint);
  void SetSegmentChiSquared(double);
  void SetTargetEffectNum(int);
  void SetSegmentKey(int);
  void SetNorm(double);
  void SetExitKey(int);
  void SetIsTotalCapture(int);
  void SetVaryNorm(bool);
  EPoint *GetPoint(int);
  std::vector<EPoint>& GetPoints();
 private:
  bool isdifferential_;
  bool isphase_;
  bool isTargetEffect_;
  bool varyNorm_;
  bool isAngDist_;
  int isTotalCapture_;
  int entrancekey_;
  int exitkey_;
  int l_;
  int targetEffectNum_;
  int segmentKey_;
  int maxAngDistOrder_;
  double min_e_;
  double max_e_;
  double min_a_;
  double max_a_;
  double e_step_;
  double a_step_;
  double segment_chi_squared_;
  double j_;
  double dataNorm_;
  double dataNormNominal_;
  double dataNormError_;
  std::string datafile_;
  std::vector<EPoint> points_;
};

#endif
```

8.183 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ExtrapLine.h File Reference

```
#include <iostream>
#include <string>
```

Classes

- class ExtrapLine

  A class to read and store a line from the extrapolation input file.

8.184 ExtrapLine.h

Go to the documentation of this file.

```
#ifndef EXTRAPLINE_H
#define EXTRAPLINE_H

#include <iostream>
#include <string>


class ExtrapLine {
 public:
  ExtrapLine(std::istream &stream) {
    stream » isActive_ » entranceKey_ » exitKey_ » minE_
       » maxE_ » eStep_ » minA_ » maxA_ » aStep_ » isDiff_;
    if(isDiff_==2) stream » phaseJ_ » phaseL_;
    else if(isDiff_==3) stream » maxAngDistOrder_;

  };
  int isActive() const {return isActive_;};
  int entranceKey() const {return entranceKey_;};
  int exitKey() const {return exitKey_;};
  double minE() const {return minE_;};
  double maxE() const {return maxE_;};
  double minA() const {return minA_;};
  double maxA() const {return maxA_;};
  double eStep() const {return eStep_;};
  double aStep() const {return aStep_;};
  int isDiff() const {return isDiff_;};
  double phaseJ() const {return phaseJ_;};
  int phaseL() const {return phaseL_;};
  int maxAngDistOrder() const {return maxAngDistOrder_;};
 private:
  int isActive_;
  int entranceKey_;
  int exitKey_;
  double minE_;
  double maxE_;
  double minA_;
  double maxA_;
  double eStep_;
  double aStep_;
  int isDiff_;
  double phaseJ_;
  int phaseL_;
  int maxAngDistOrder_;
};

#endif
```

8.185 /Users/kuba/Desktop/R-Matrix/AZURE2/include/GenMatrixFunc.h File Reference

```
#include "Constants.h"
#include "Config.h"
```

Classes

- struct TempTMatrix

  A temporaray T-Matrix structure.

- class GenMatrixFunc

  A generalized function class to calculate cross sections.

8.186 GenMatrixFunc.h

Go to the documentation of this file.

```
#ifndef GENMATRIXFUNC_H
#define GENMATRIXFUNC_H

#include "Constants.h"
#include "Config.h"

class EPoint;
class CNuc;


struct TempTMatrix {
  double jValue;
  int lValue;
  int lpValue;
  complex TMatrix;
};



class GenMatrixFunc {
 public:
  GenMatrixFunc() {};
  virtual ~GenMatrixFunc(){};
  virtual void ClearMatrices()=0;
  virtual void FillMatrices(EPoint*)=0;
  virtual void InvertMatrices()=0;
  virtual void CalculateTMatrix(EPoint*)=0;
  void CalculateCrossSection(EPoint*);
  void NewTempTMatrix(TempTMatrix);
  void AddToTempTMatrix(int,complex);
  void ClearTempTMatrices();
  void AddTMatrixElement(int,int,complex,int decayNum=1);
  void AddECTMatrixElement(int,int,complex);
  int IsTempTMatrix(double,int,int);
  int NumTempTMatrices() const;
  TempTMatrix *GetTempTMatrix(int);
  complex GetTMatrixElement(int,int,int decayNum=1) const;
  complex GetECTMatrixElement(int,int) const;

  virtual CNuc *compound() const = 0;
  virtual const Config& configure() const = 0;
 protected:
  std::vector<matrix_c> tmatrix_;
  matrix_c ec_tmatrix_;
 private:
  std::vector<TempTMatrix> temp_t_matrices_;
};

#endif
```

8.187 /Users/kuba/Desktop/R-Matrix/AZURE2/include/GSLException.h File Reference

```
#include <iostream>
#include <exception>
#include <sstream>
```

Classes

- class GSLException

8.188 GSLException.h

Go to the documentation of this file.

```
#ifndef GSLEXCEPTION_H
#define GSLEXCEPTION_H

#include <iostream>
#include <exception>
#include <sstream>

class GSLException : public std::exception {
 public:
  GSLException(std::string message, std::string line = "", std::string file = "") {
    std::ostringstream stm;
    if(line != "" && file != "") {
      stm « "Exception thrown from line " « line « " of file " « file « " with message: " « std::endl
      « message;
    } else {
      stm « "Exception thrown with message: " « std::endl
      « message;
    }
    messageString_=stm.str();
    message_=messageString_.c_str();
  };
  ~GSLException() throw() {
  };
  virtual const char* what() const throw() {
    return message_;
  };
  static void GSLErrorHandler(const char*, const char*, int, int);
 private:
  std::string messageString_;
  const char* message_;
};

#endif
```

8.189 /Users/kuba/Desktop/R-Matrix/AZURE2/include/IntegratedFermiFunc.h File Reference

Classes

- class IntegratedFermiFunc

  A function class to calculate the integrated Fermi function for beta decay.

8.190 IntegratedFermiFunc.h

Go to the documentation of this file.

```
#ifndef INTEGRATEDFERMIFUNC_H
#define INTEGRATEDFERMIFUNC_H


class IntegratedFermiFunc {
 public:
  IntegratedFermiFunc(int,double V0 = 0.);
  double operator()(double,double,double);
 private:
  static const double alpha_;
  static const double pi_;
  static const double electronMass_;
  static const double hbarc_;
  static double Integrand(double, void*);
  typedef struct Params_ {
    int charge;
    double gamma0;
    double Z;
    double radius;
    double W0;
    double GammaDenom2;
    double V0;
```

```
  } Params_;
  int charge_;
  double V0_;
};

#endif
```

8.191 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Interference.h File Reference

```
#include <string>
```

Classes

- class Interference

  An AZURE $l_1, l_2, l_1', l_2', J_1, J_2$ *combination.*

8.192 Interference.h

Go to the documentation of this file.

```
#ifndef INTERFERENCE_H
#define INTERFERENCE_H

#include <string>


class Interference {
 public:
 Interference(int, int, double, std::string);
  std::string GetInterferenceType() const;
  int GetM1() const;
  int GetM2() const;
  double GetZ1Z2() const;
 private:
  int m1_;
  int m2_;
  double z1z2_;
  std::string intertype_;
};

#endif
```

8.193 /Users/kuba/Desktop/R-Matrix/AZURE2/include/JGroup.h File Reference

```
#include "Constants.h"
#include "ALevel.h"
#include "AChannel.h"
```

Classes

- class JGroup

  An AZURE J^{π} *group.*

8.194 JGroup.h

Go to the documentation of this file.

```
#ifndef JGROUP_H
#define JGROUP_H

#include "Constants.h"
#include "ALevel.h"
#include "AChannel.h"

class NucLine;


class JGroup {
 public:
  JGroup(NucLine);
  JGroup(double,int);
  bool IsInRMatrix() const;
  int IsLevel(ALevel);
  int GetPi() const;
  int NumLevels() const;
  int NumChannels();
  int IsChannel(AChannel);
  double GetJ() const;
  void AddLevel(ALevel);
  void AddChannel(AChannel);
  AChannel *GetChannel(int);
  ALevel *GetLevel(int);
 private:
  bool isinrmatrix_;
  int pi_;
  double j_;
  std::vector<ALevel> levels_;
  std::vector<AChannel> channels_;
};

#endif
```

8.195 /Users/kuba/Desktop/R-Matrix/AZURE2/include/KGroup.h File Reference

```
#include "MGroup.h"
#include "ECMGroup.h"
```

Classes

- class KGroup

  An AZURE s, s' group.

8.196 KGroup.h

Go to the documentation of this file.

```
#ifndef KGROUP_H
#define KGROUP_H

#include "MGroup.h"
#include "ECMGroup.h"


class KGroup {
 public:
  KGroup(double, double);
  int NumMGroups() const;
  int NumECMGroups() const;
  int IsMGroup(MGroup);
```

```
  double GetS() const;
  double GetSp() const;
  void AddMGroup(MGroup);
  void AddECMGroup(ECMGroup);
  MGroup *GetMGroup(int);
  ECMGroup *GetECMGroup(int);
 private:
  double s_;
  double sp_;
  std::vector<MGroup> mgroups_;
  std::vector<ECMGroup> ec_mgroups_;
};

#endif
```

8.197 /Users/kuba/Desktop/R-Matrix/AZURE2/include/KLGroup.h File Reference

```
#include <vector>
#include "Interference.h"
```

Classes

- class KLGroup

  An AZURE s, s', L group.

8.198 KLGroup.h

Go to the documentation of this file.

```
#ifndef KLGROUP_H
#define KLGROUP_H

#include <vector>
#include "Interference.h"


class KLGroup {
 public:
  KLGroup(int,int);
  int GetK() const;
  int GetLOrder() const;
  int NumInterferences() const;
  int IsInterference(Interference);
  void AddInterference(Interference);
  Interference *GetInterference(int);
 private:
  int k_;
  int lorder_;
  std::vector<Interference> interferences_;
};

#endif
```

8.199 /Users/kuba/Desktop/R-Matrix/AZURE2/include/MatrixInv.h File Reference

```
#include "Constants.h"
```

Classes

- class MatrixInv

    A Function class to perform matrix inversion.

8.200 MatrixInv.h

Go to the documentation of this file.

```
#ifndef MATRIXINV_H
#define MATRIXINV_H

#include "Constants.h"


class MatrixInv {
 public:
  MatrixInv(const matrix_c&);
  const matrix_c& inverse() const {return inverse_;};
 private:
 matrix_c inverse_;
};

#endif
```

8.201 /Users/kuba/Desktop/R-Matrix/AZURE2/include/MGroup.h File Reference

```
#include "Constants.h"
```

Classes

- class MGroup

    An AZURE internal reaction pathway.

8.202 MGroup.h

Go to the documentation of this file.

```
#ifndef MGROUP_H
#define MGROUP_H

#include "Constants.h"


class MGroup {
 public:
  MGroup(int, int, int);
  int GetChNum() const;
  int GetChpNum() const;
  int GetJNum() const;
  double GetStatSpinFactor() const;
  void SetStatSpinFactor(double);
 private:
  int jnum_;
  int ch_;
  int chp_;
  double statspinfactor_;
  complex tmatrix_;
};

#endif
```

8.203 /Users/kuba/Desktop/R-Matrix/AZURE2/include/NFIntegral.h File Reference

```
#include "PPair.h"
#include "WhitFunc.h"
#include <math.h>
```

Classes

- class NFIntegral

  A function class to calculate the channel integrals in the denominator of the $N_f^{1/2}$ term.

8.204 NFIntegral.h

Go to the documentation of this file.

```
#ifndef NFINTEGRAL_H
#define NFINTEGRAL_H

#include "PPair.h"
#include "WhitFunc.h"
#include <math.h>


class NFIntegral {
 public:
  NFIntegral(PPair* pPair) {
    params_.whitFunc = new WhitFunc(pPair);
    chanrad_ = pPair->GetChRad();
    total_sep_e_ = pPair->GetSepE()+pPair->GetExE();
  };
  ~NFIntegral() {
    delete params_.whitFunc;
  };
  double operator()(int lFinal,double levelEnergy);
  double chanRad() const {return chanrad_;};
  double totalSepE() const {return total_sep_e_;};
 private:
  static double Integrand(double,void*);
  typedef struct Params {
    WhitFunc *whitFunc;
    int lfValue;
    double bindingEnergy;
    double whitChRadSquaredValue;
  } Params;
  Params params_;
  double chanrad_;
  double total_sep_e_;
};

#endif
```

8.205 /Users/kuba/Desktop/R-Matrix/AZURE2/include/NucLine.h File Reference

```
#include <iostream>
```

Classes

- class NucLine

  A class to read and store a line from a nuclear input file.

8.206 NucLine.h

Go to the documentation of this file.

```
#ifndef NUCLINE_H
#define NUCLINE_H

#include <iostream>


class NucLine {
 public:
  NucLine(std::istream &stream) {
    stream » levelJ_ » levelPi_ » levelE_ » levelFix_ » aa_ » ir_
      » s_ » l_ » levelID_ » isActive_ » channelFix_ » gamma_ » j1_ » pi1_
      » j2_ » pi2_ » e2_ » m1_ » m2_ » z1_ » z2_
      » entranceSepE_ » sepE_ » j3_ » pi3_ » e3_
      » pType_ » chRad_ » g1_ » g2_ » ecMultMask_;
    s_/=2.;
    l_/=2;
  };
  double levelJ() const {return levelJ_;};
  int levelPi() const {return levelPi_;};
  double levelE() const {return levelE_;};
  int levelFix() const {return levelFix_;};
  int aa() const {return aa_;};
  int ir() const {return ir_;};
  double s() const {return s_;};
  int l() const {return l_;};
  int levelID() const {return levelID_;};
  int isActive() const {return isActive_;};
  int channelFix() const {return channelFix_;};
  double gamma() const {return gamma_;};
  double j1() const {return j1_;};
  int pi1() const {return pi1_;};
  double j2() const {return j2_;};
  int pi2() const {return pi2_;};
  double e2() const {return e2_;};
  double m1() const {return m1_;};
  double m2() const {return m2_;};
  int z1() const {return z1_;};
  int z2() const {return z2_;};
  double entranceSepE() const {return entranceSepE_;};
  double sepE() const {return sepE_;};
  int j3() const {return j3_;};
  int pi3() const {return pi3_;};
  double e3() const {return e3_;};
  int pType() const {return pType_;};
  double chRad() const {return chRad_;};
  double g1() const {return g1_;};
  double g2() const {return g2_;};
  double ecMultMask() const {return ecMultMask_;};
 private:
  double levelJ_;
  int levelPi_;
  double levelE_;
  int levelFix_;
  int aa_;
  int ir_;
  double s_;
  int l_;
  int levelID_;
  int isActive_;
  int channelFix_;
  double gamma_;
  double j1_;
  int pi1_;
  double j2_;
  int pi2_;
  double e2_;
  double m1_;
  double m2_;
  int z1_;
  int z2_;
  double entranceSepE_;
  double sepE_;
  int j3_;
  int pi3_;
  double e3_;
  int pType_;
  double chRad_;
  double g1_;
  double g2_;
  unsigned int ecMultMask_;
};

#endif
```

8.207 /Users/kuba/Desktop/R-Matrix/AZURE2/include/PPair.h File Reference

```
#include "Decay.h"
```

Classes

- class PPair

    An AZURE Particle Pair.

8.208 PPair.h

Go to the documentation of this file.

```
#ifndef PPAIR_H
#define PPAIR_H

#include "Decay.h"

class NucLine;


class PPair {
public:
  PPair(NucLine);
  bool IsEntrance() const;
  int GetZ(int) const;
  int GetPi(int) const;
  int GetPType() const;
  int NumDecays() const;
  int IsDecay(Decay);
  int IsDecay(int);
  int GetPairKey() const;
  double GetM(int) const;
  double GetG(int) const;
  double GetJ(int) const;
  double GetExE() const;
  double GetSepE() const;
  double GetChRad() const;
  double GetRedMass() const;
  double GetI1I2Factor() const;
  void AddDecay(Decay);
  void SetEntrance();
  Decay *GetDecay(int);
private:
  bool entrance_;
  bool ec_entrance_;
  int pair_z_[2];
  int pair_pi_[2];
  int pair_ptype_;
  int pair_key_;
  double pair_m_[2];
  double pair_g_[2];
  double pair_j_[2];
  double pair_ex_e_;
  double pair_sep_e_;
  double pair_ch_rad_;
  double red_mass_;
  double i1i2factor_;
  std::vector<Decay> decays_;
};

#endif
```

8.209 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ReactionRate.h File Reference

```
#include "Constants.h"
```

Classes

- class RateData

  A container structure for a reaction rate.
- class ReactionRate

  A function class to calculate the reaction rate.

Functions

- double gsl_reactionrate_integration (double, CNuc ∗, const Config &, int, int)

8.209.1 Function Documentation

8.209.1.1 gsl_reactionrate_integration()

```
double gsl_reactionrate_integration (
            double temperature,
            CNuc * compound,
            const Config & configure,
            int entranceKey,
            int exitKey )
```

Definition at line 42 of file ReactionRate.cpp.

8.210 ReactionRate.h

Go to the documentation of this file.

```
#ifndef REACTIONRATE_H
#define REACTIONRATE_H

#include "Constants.h"

class CNuc;

extern double gsl_reactionrate_integration(double,CNuc*,const Config&,int,int);


class RateData {
 public:
 RateData(double t, double r) :
  temperature(t),rate(r) {};
  bool operator<(const RateData& right) const {
    return temperature < right.temperature;
  };
  double temperature;
  double rate;
};


class ReactionRate {
 public:
  ReactionRate(CNuc*, const vector_r&, const Config &, int, int);
  CNuc *compound() const {return compound_;};
  const Config &configure() const {return configure_;};
  int entranceKey() const {return entrance_key_;};
  int exitKey() const {return exit_key_;};
  void CalculateRates();
  void CalculateFileRates();
  void WriteRates();
 private:
  int entrance_key_;
  int exit_key_;
  CNuc *compound_;
  const Config &configure_;
  std::vector<RateData> rates_;
};



#endif
```

8.211 /Users/kuba/Desktop/R-Matrix/AZURE2/include/RMatrixFunc.h File Reference

```
#include ¨GenMatrixFunc.h¨
```

Classes

- class RMatrixFunc

  A function class to calculate the T-Matrix using the R-Matrix.

8.212 RMatrixFunc.h

Go to the documentation of this file.

```
#ifndef RMATRIXFUNC_H
#define RMATRIXFUNC_H

#include "GenMatrixFunc.h"


class RMatrixFunc : public GenMatrixFunc {
 public:
  RMatrixFunc(CNuc*,const Config&);
  CNuc *compound() const {return compound_;};
  const Config &configure() const {return configure_;};

  void ClearMatrices();
  void FillMatrices(EPoint*);
  void InvertMatrices();
  void CalculateTMatrix(EPoint*);
  void CalculateCrossSection();

  complex GetRMatrixElement(int,int,int) const;
  complex GetRLMatrixElement(int,int,int) const;
  complex GetRLInvMatrixElement(int,int,int) const;
  complex GetRLInvRMatrixElement(int,int,int) const;
  matrix_c *GetJSpecRLMatrix(int);
  void AddRMatrixElement(int,int,int,complex);
  void AddRLMatrixElement(int,int,int,complex);
  void AddRLInvMatrix(matrix_c);
  void AddRLInvRMatrixElement(int,int,int,complex);
 private:
  CNuc *compound_;
  const Config& configure_;
  vector_matrix_c r_matrices_;
  vector_matrix_c rl_matrices_;
  vector_matrix_c rl_inv_matrices_;
  vector_matrix_c rl_inv_r_matrices_;
};

#endif
```

8.213 /Users/kuba/Desktop/R-Matrix/AZURE2/include/SegLine.h File Reference

```
#include <iostream>
#include <string>
```

Classes

- class SegLine

  A class to read and store a line from the data segments input file.

8.214 SegLine.h

Go to the documentation of this file.

```
#ifndef SEGLINE_H
#define SEGLINE_H

#include <iostream>
#include <string>


class SegLine {
 public:
  SegLine(std::istream &stream) {
    stream » isActive_ »  entranceKey_ » exitKey_ » minE_ » maxE_ » minA_ » maxA_ » isDiff_;
    if(isDiff_==2) stream » phaseJ_ » phaseL_;
    stream » dataNorm_ » varyNorm_ » dataNormError_;
    std::string dummyString;
    getline(stream,dummyString);
    int p2 = dummyString.find_last_not_of(" \n\t\r");
    if (p2 != std::string::npos) {
      int p1 = dummyString.find_first_not_of(" \n\t\r");
      if (p1 == std::string::npos) p1 = 0;
      dataFile_=dummyString.substr(p1,(p2-p1)+1);
    } else dataFile_=std::string();
  };
  int isActive() const {return isActive_;};
  int entranceKey() const {return entranceKey_;};
  int exitKey() const {return exitKey_;};
  double minE() const {return minE_;};
  double maxE() const {return maxE_;};
  double minA() const {return minA_;};
  double maxA() const {return maxA_;};
  int isDiff() const {return isDiff_;};
  std::string dataFile() const {return dataFile_;};
  double dataNorm() const {return dataNorm_;};
  double dataNormError() const {return dataNormError_;};
  int varyNorm() const {return varyNorm_;};
  double phaseJ() const {return phaseJ_;};
  int phaseL() const {return phaseL_;};
 private:
  int isActive_;
  int entranceKey_;
  int exitKey_;
  double minE_;
  double maxE_;
  double minA_;
  double maxA_;
  int isDiff_;
  std::string dataFile_;
  double dataNorm_;
  double dataNormError_;
  int varyNorm_;
  double phaseJ_;
  int phaseL_;
};

#endif
```

8.215 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ShftFunc.h File Reference

```
#include "PPair.h"
#include "WhitFunc.h"
```

Classes

- class ShftFunc

  A function class for negative energy shift functions.

8.216 ShftFunc.h

Go to the documentation of this file.

```
#ifndef SHIFTFUNC_H
#define SHIFTFUNC_H

#include "PPair.h"
#include "WhitFunc.h"

class ShftFunc;


class ShftFunc {
 public:
  ShftFunc(PPair* pPair) {
    totalSepE_=pPair->GetSepE()+pPair->GetExE();
    radius_=(double) pPair->GetChRad();
    params_.whitFunc= new WhitFunc(pPair);
  };
  ~ShftFunc() {
    delete params_.whitFunc;
  };
  double operator()(int l,double energy);
  double EnergyDerivative(int l,double energy);
 private:
  double totalSepE() const {return totalSepE_;};
  double radius() const {return radius_;};
  static double thisShftFunc(double,void*);
  static double theWhitFunc(double,void*);
  typedef struct Params {
    int lValue;
    double bindingEnergy;
    WhitFunc *whitFunc;
  } Params;
  Params params_;
  double totalSepE_;
  double radius_;
};

#endif
```

8.217 /Users/kuba/Desktop/R-Matrix/AZURE2/include/TargetEffect.h File Reference

```
#include <string>
#include <fstream>
#include <vector>
#include ¨Constants.h¨
#include ¨Equation.h¨
```

Classes

- class TargetEffect

  An AZURE target effect entry.

8.218 TargetEffect.h

Go to the documentation of this file.

```
#ifndef TARGETEFFECT_H
#define TARGETEFFECT_H

#include <string>
#include <fstream>
```

```
#include <vector>
#include "Constants.h"
#include "Equation.h"
class TargetEffect {
 public:
  TargetEffect(std::istream &, const Config&);
  bool IsActive() const;
  bool IsConvolution() const;
  bool IsTargetIntegration() const;
  bool IsQCoefficients() const;
  int NumSubPoints() const;
  int NumQCoefficients() const;
  double GetSigma() const;
  double GetDensity() const;
  double TargetThickness(double,const Config&);
  double GetConvolutionFactor(double, double) const;
  double GetQCoefficient(int) const;
  void SetSigma(double);
  void SetNumSubPoints(int);
  std::vector<int> GetSegmentsList() const;
  Equation *GetStoppingPowerEq();
  static constexpr double convolutionRange=3.;
 private:
  bool isConvolution_;
  bool isTargetIntegration_;
  bool isActive_;
  bool isQCoefficients_;
  int numIntegrationPoints_;
  double sigma_;
  double density_;
  Equation stoppingPowerEq_;
  std::string segmentsList_;
  vector_r qCoefficients_;
};
#endif
```

8.219 /Users/kuba/Desktop/R-Matrix/AZURE2/include/WhitFunc.h File Reference

```
#include "PPair.h"
#include <gsl/gsl_sf_hyperg.h>
#include "Constants.h"
```

Classes

- class WhitFunc

  A function class to calculate Whittaker functions for negative energy channels.

Functions

- double gsl_whit_function (int, double, double, double, int, int)

8.219.1 Function Documentation

8.219.1.1 gsl_whit_function()

```
double gsl_whit_function (
              int ,
              double ,
              double ,
              double ,
              int ,
              int  )
```

8.220 WhitFunc.h

Go to the documentation of this file.

```
#ifndef WHITFUNC_H
#define WHITFUNC_H

#include "PPair.h"
#include <gsl/gsl_sf_hyperg.h>
#include "Constants.h"

extern double gsl_whit_function(int,double,double,double,int,int);


class WhitFunc {
  public:
  WhitFunc(PPair *pPair) {
    z1_=pPair->GetZ(1);
    z2_=pPair->GetZ(2);
    redmass_=(double)pPair->GetRedMass();
  };
  int z1() const {
    return z1_;
  };
  int z2() const {
    return  z2_;
  };
  double redmass() const {
    return redmass_;
  };
  double operator()(int l, double radius, double energy) const {
    const double k=-sqrt(uconv/2.)*fstruc*z1()*z2()*sqrt(redmass()/energy);
    const double m=l+0.5;
    const double z=2.0*sqrt(2.0*uconv)/hbarc*radius*sqrt(redmass()*energy);

    const double a=m-k+0.5;
    const double b=1.0+2.0*m;

    return exp(-z/2.0)*pow(z,m+0.50)*gsl_sf_hyperg_U(a,b,z);
  };
  private:
    int z1_;
    int z2_;
    double redmass_;
};

#endif
```

8.221 /Users/kuba/Desktop/R-Matrix/AZURE2/README.md File Reference

8.222 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AChannel.cpp File Reference

```
#include ¨AChannel.h¨
#include ¨NucLine.h¨
#include <math.h>
#include <assert.h>
```

8.223 AChannel.cpp

Go to the documentation of this file.

```
#include "AChannel.h"
#include "NucLine.h"
#include <math.h>
```

```
#include <assert.h>


AChannel::AChannel(NucLine nucLine, int pairNum) {
  l_=nucLine.l();
  s_=nucLine.s();
  pair_=pairNum;
  if(nucLine.pType()==0) {
    radtype_='P';
  } else if(nucLine.pType()==10) {
    if(nucLine.levelPi()*nucLine.pi2()==pow(-1,nucLine.l())) radtype_='E';
    else radtype_='M';
  } else if(nucLine.pType()==20) {
    if(nucLine.l()==0) radtype_='F';
    else radtype_='G';
  }
}

AChannel::AChannel(int lValue, double sValue, int pairNum, char radType) {
  l_=lValue;
  s_=sValue;
  pair_=pairNum;
  radtype_=radType;
};

int AChannel::GetPairNum() const {
  return pair_;
}

int AChannel::GetL() const {
  return l_;
}

double AChannel::GetS() const {
  return s_;
}

double AChannel::GetBoundaryCondition() const {
  return boundary_condition_;
}

char AChannel::GetRadType() const {
  return radtype_;
}

void AChannel::SetBoundaryCondition(double boundaryCondition) {
  boundary_condition_=boundaryCondition;
}

```

8.224 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ALevel.cpp File Reference

```
#include "ALevel.h"
#include "NucLine.h"
```

8.225 ALevel.cpp

Go to the documentation of this file.

```
#include "ALevel.h"
#include "NucLine.h"

ALevel::ALevel(NucLine nucLine) :
  level_e_(nucLine.levelE()),fitlevel_e_(0.0), isinrmatrix_(true), sqrt_nf_factor_(1.0), isECLevel_(false),
  ecMultMask_(0), ecPairNum_(0) {
  if(nucLine.levelFix()==1) energyfixed_=true;
  else energyfixed_=false;
}

ALevel::ALevel(double energy) :
```

```
  energyfixed_(true),level_e_(energy),fitlevel_e_(0.0), isinrmatrix_(false), sqrt_nf_factor_(1.0),
isECLevel_(false),
  ecMultMask_(0), ecPairNum_(0) {};

bool ALevel::EnergyFixed() const {
  return energyfixed_;
}

bool ALevel::IsInRMatrix() const {
  return isinrmatrix_;
}

bool ALevel::ChannelFixed(int channelNum) const {
  return channelfixed_[channelNum-1];
}

bool ALevel::IsECLevel() const {
  return isECLevel_;
}

int ALevel::NumNFIntegrals() const {
  return nf_integrals_.size();
}

int ALevel::GetTransformIterations() const {
  return transform_iter_;
}

int ALevel::GetECPairNum() const {
  return ecPairNum_;
}

unsigned char ALevel::GetECMultMask() const {
  return ecMultMask_;
}

double ALevel::GetE() const {
  return level_e_;
}

double ALevel::GetGamma(int channelNum) const {
  return gammas_[channelNum-1];
}

double ALevel::GetFitGamma(int channelNum) const {
  return fitgammas_[channelNum-1];
}

double ALevel::GetFitE() const {
  return fitlevel_e_;
}

double ALevel::GetNFIntegral(int channelNum) const {
  return nf_integrals_[channelNum-1];
}

double ALevel::GetSqrtNFFactor() const {
  return sqrt_nf_factor_;
}

double ALevel::GetECConversionFactor(int channelNum) const {
  return ec_conv_factors_[channelNum-1];
}

double ALevel::GetTransformGamma(int channelNum) const {
  return transform_gammas_[channelNum-1];
}

double ALevel::GetTransformE() const {
  return transform_e_;
}

double ALevel::GetBigGamma(int channelNum) const {
  return big_gammas_[channelNum-1];
}

double ALevel::GetShiftFunction(int channelNum) const {
  return shifts_[channelNum-1];
}

complex ALevel::GetExternalGamma(int channelNum) const {
  return external_gammas_[channelNum-1];
}

void ALevel::AddGamma(NucLine nucLine) {
  double b=nucLine.gamma();
  gammas_.push_back(b);
```

```
  fitgammas_.push_back(0.0);
  transform_gammas_.push_back(0.0);
  big_gammas_.push_back(0.0);
  external_gammas_.push_back(complex(0.0,0.0));
  if(nucLine.channelFix()==1) channelfixed_.push_back(true);
  else channelfixed_.push_back(false);
  shifts_.push_back(0.0);
}

void ALevel::AddGamma(double reducedWidth) {
  gammas_.push_back(reducedWidth);
  fitgammas_.push_back(0.0);
  transform_gammas_.push_back(0.0);
  big_gammas_.push_back(0.0);
  external_gammas_.push_back(complex(0.0,0.0));
  channelfixed_.push_back(false);
}

void ALevel::SetGamma(int channelNum, double reducedWidth) {
  gammas_[channelNum-1]=reducedWidth;
}

void ALevel::SetE(double energy) {
  level_e_=energy;
}

void ALevel::SetFitGamma(int channelNum,double reducedWidth) {
  fitgammas_[channelNum-1]=reducedWidth;
}

void ALevel::SetFitE(double energy) {
  fitlevel_e_=energy;
}

void ALevel::AddNFIntegral(double integral) {
  nf_integrals_.push_back(integral);
}

void ALevel::SetSqrtNFFactor(double term) {
  sqrt_nf_factor_=term;
}

void ALevel::AddECConversionFactor(double conversionFactor) {
  ec_conv_factors_.push_back(conversionFactor);
}

void ALevel::SetTransformGamma(int channelNum,double reducedWidth) {
  transform_gammas_[channelNum-1]=reducedWidth;
}

void ALevel::SetTransformE(double energy) {
  transform_e_=energy;
}

void ALevel::SetBigGamma(int channelNum, double partialWidth) {
  big_gammas_[channelNum-1]=partialWidth;
}

void ALevel::SetTransformIterations(int iterations) {
  transform_iter_=iterations;
}

void ALevel::SetExternalGamma(int channelNum, complex reducedWidth) {
  external_gammas_[channelNum-1]=reducedWidth;
}

void ALevel::SetShiftFunction(int channelNum, double shiftFunction) {
  shifts_[channelNum-1]=shiftFunction;
}

void ALevel::SetECParams(int pairNum, unsigned char multMask) {
  isECLevel_=true;
  ecPairNum_=pairNum;
  ecMultMask_=multMask;
}
```

8.226 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AMatrixFunc.cpp File Reference

```
#include "AMatrixFunc.h"
#include "CNuc.h"
```

```
#include ¨Config.h¨
#include ¨EPoint.h¨
#include ¨MatrixInv.h¨
#include <assert.h>
```

8.227 AMatrixFunc.cpp

Go to the documentation of this file.

```
#include "AMatrixFunc.h"
#include "CNuc.h"
#include "Config.h"
#include "EPoint.h"
#include "MatrixInv.h"
#include <assert.h>

AMatrixFunc::AMatrixFunc(CNuc* compound, const Config &configure) :
  compound_(compound), configure_(configure) {}

complex AMatrixFunc::GetAMatrixElement(int jGroupNum, int lambdaNum, int muNum) const {
  return a_matrices_[jGroupNum-1][lambdaNum-1][muNum-1];
}

matrix_c *AMatrixFunc::GetJSpecAInvMatrix(int jGroupNum) {
  matrix_c *b=&a_inv_matrices_[jGroupNum-1];
  return b;
}

void AMatrixFunc::ClearMatrices() {
  a_inv_matrices_.clear();
  a_matrices_.clear();
  tmatrix_.clear();
  ec_tmatrix_.clear();
}

void AMatrixFunc::FillMatrices (EPoint *point) {
  double inEnergy;
  if(compound()->
     GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->
     GetPType()==20)
    inEnergy=point->GetCMEnergy()+
      compound()->
      GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->
      GetSepE()+
      compound()->
      GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->
      GetExE();
  else inEnergy = point->GetCMEnergy()+
     compound()->GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->GetSepE()+
     compound()->GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->GetExE();
  for(int j=1;j<=compound()->NumJGroups();j++) {
    if(compound()->GetJGroup(j)->IsInRMatrix()) {
      for(int la=1;la<=compound()->GetJGroup(j)->NumLevels();la++) {
    if(compound()->GetJGroup(j)->GetLevel(la)->IsInRMatrix()) {
      ALevel *level=compound()->GetJGroup(j)->GetLevel(la);
      for(int lap=1;lap<=compound()->GetJGroup(j)->NumLevels();lap++) {
        if(compound()->GetJGroup(j)->GetLevel(lap)->IsInRMatrix()) {
          ALevel *levelp=compound()->GetJGroup(j)->GetLevel(lap);
          complex sum(0.0,0.0);
          for(int ch=1;ch<=compound()->GetJGroup(j)->NumChannels();ch++) {
        double gammaCh=level->GetFitGamma(ch);
        double gammaChp=levelp->GetFitGamma(ch);
        complex loElement=point->GetLoElement(j,ch);
        sum+=gammaCh*gammaChp*loElement;
        if((compound()->GetJGroup(j)->GetChannel(ch)->GetRadType() == 'M' ||
            compound()->GetJGroup(j)->GetChannel(ch)->GetRadType() == 'E' ) &&
           la==lap &&
           (configure().paramMask & Config::USE_RMC_FORMALISM))
          sum+=complex(0.0,1.0)*gammaCh*gammaChp;
        if((configure().paramMask & Config::USE_BRUNE_FORMALISM) &&
compound()->GetJGroup(j)->GetChannel(ch)->GetRadType()=='P') {
          sum+=gammaCh*gammaChp*compound()->GetJGroup(j)->GetChannel(ch)->GetBoundaryCondition();
          if(la==lap) sum-=gammaCh*gammaChp*level->GetShiftFunction(ch);
          else sum-=gammaCh*gammaChp*

(level->GetShiftFunction(ch)*(inEnergy-levelp->GetFitE())-levelp->GetShiftFunction(ch)*(inEnergy-level->GetFitE()))/
             (level->GetFitE()-levelp->GetFitE());
        }
          }
```

```
          if(la==lap) {
        double resenergy=level->GetFitE();
        this->AddAInvMatrixElement(j,la,lap,resenergy-inEnergy-sum);
          } else this->AddAInvMatrixElement(j,la,lap,-sum);
        }
      }
    }
      }
    }
  }
}

void AMatrixFunc::InvertMatrices() {
  for(int j=1;j<=compound()->NumJGroups();j++) {
    if(compound()->GetJGroup(j)->IsInRMatrix()) {
      matrix_c *theAInvMatrix = this->GetJSpecAInvMatrix(j);
      MatrixInv matrixInv(*theAInvMatrix);
      this->AddAMatrix(matrixInv.inverse());
    }
  }
}

void AMatrixFunc::CalculateTMatrix(EPoint *point) {
  int aa=compound()->GetPairNumFromKey(point->GetEntranceKey());
  int irEnd;
  int irStart;
  bool isRMC=false;
  if((configure().paramMask & Config::USE_RMC_FORMALISM) &&
     compound()->GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->GetPType()==10) {
    irStart=1;
    irEnd=compound()->GetPair(aa)->NumDecays();
    isRMC=true;
  } else {
    irStart=0;
    while(irStart<compound()->GetPair(aa)->NumDecays()) {
      irStart++;
      if(compound()->GetPair(aa)->GetDecay(irStart)->GetPairNum()==
     compound()->GetPairNumFromKey(point->GetExitKey())) break;
    }
    irEnd=irStart;
  }
  for(int ir=irStart;ir<=irEnd;ir++) {
    Decay *theDecay=compound()->GetPair(aa)->GetDecay(ir);
    for(int k=1;k<=theDecay->NumKGroups();k++) {
      for(int m=1;m<=theDecay->GetKGroup(k)->NumMGroups();m++) {
    MGroup *theMGroup=theDecay->GetKGroup(k)->GetMGroup(m);
    JGroup *theJGroup=compound()->GetJGroup(theMGroup->GetJNum());
    AChannel *entranceChannel=theJGroup->GetChannel(theMGroup->GetChNum());
    AChannel *exitChannel=theJGroup->GetChannel(theMGroup->GetChpNum());
    complex uphase=point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChNum())*
      point->GetExpHardSpherePhase(theMGroup->GetJNum(),theMGroup->GetChNum())*
      point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChpNum())*
      point->GetExpHardSpherePhase(theMGroup->GetJNum(),theMGroup->GetChpNum());
    complex umatrix(0.,0.);
    for(int la=1;la<=compound()->GetJGroup(theMGroup->GetJNum())->NumLevels();la++) {
      if(compound()->GetJGroup(theMGroup->GetJNum())->GetLevel(la)->IsInRMatrix()) {
        ALevel *level=compound()->GetJGroup(theMGroup->GetJNum())->GetLevel(la);
        for(int lap=1;lap<=compound()->GetJGroup(theMGroup->GetJNum())->NumLevels();lap++) {
          if(compound()->GetJGroup(theMGroup->GetJNum())->GetLevel(lap)->IsInRMatrix()) {
        ALevel *levelp=compound()->GetJGroup(theMGroup->GetJNum())->GetLevel(lap);
        umatrix+=2.0*complex(0.0,1.0)*
          point->GetSqrtPenetrability(theMGroup->GetJNum(),theMGroup->GetChNum())*
          point->GetSqrtPenetrability(theMGroup->GetJNum(),theMGroup->GetChpNum())*
          level->GetFitGamma(theMGroup->GetChNum())*
          levelp->GetFitGamma(theMGroup->GetChpNum())*
          this->GetAMatrixElement(theMGroup->GetJNum(),la,lap);
          }
        }
      }
    }
    complex tphase=point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChNum())*
      point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChNum());
    complex tmatrix;
    if(isRMC) this->AddTMatrixElement(k,m,complex(0.0,-1.0)*umatrix,ir);
    else {
      if(theMGroup->GetChNum()==theMGroup->GetChpNum()) {
        tmatrix=tphase-uphase*(1.0+umatrix);
      } else tmatrix=-uphase*umatrix;
      this->AddTMatrixElement(k,m,tmatrix);
    }
      }
      for(int m=1;m<=theDecay->GetKGroup(k)->NumECMGroups();m++) {
    ECMGroup *theECMGroup=theDecay->GetKGroup(k)->GetECMGroup(m);
    ALevel *finalLevel=compound()->GetJGroup(theECMGroup->GetJGroupNum())
      ->GetLevel(theECMGroup->GetLevelNum());
    double ecNormParam=finalLevel->GetFitGamma(theECMGroup->GetFinalChannel())*
      finalLevel->GetSqrtNFFactor()*finalLevel->GetECConversionFactor(theECMGroup->GetFinalChannel());
```

```
    complex tmatrix=ecNormParam*point->GetECAmplitude(k,m);
    if(theECMGroup->IsChannelCapture()) {
      int internalChannel=theECMGroup->GetIntChannelNum();
      MGroup *chanMGroup=compound()->GetPair(aa)->GetDecay(theECMGroup->GetChanCapDecay())
        ->GetKGroup(theECMGroup->GetChanCapKGroup())->GetMGroup(theECMGroup->GetChanCapMGroup());
      AChannel *chanEntranceChannel=compound()->GetJGroup(chanMGroup->GetJNum())
        ->GetChannel(chanMGroup->GetChNum());
      AChannel *chanExitChannel=compound()->GetJGroup(chanMGroup->GetJNum())
        ->GetChannel(chanMGroup->GetChpNum());
      complex umatrix(0.,0.);
      for(int la=1;la<=compound()->GetJGroup(chanMGroup->GetJNum())->NumLevels();la++) {
        if(compound()->GetJGroup(chanMGroup->GetJNum())->GetLevel(la)->IsInRMatrix()) {
          ALevel *level=compound()->GetJGroup(chanMGroup->GetJNum())->GetLevel(la);
          if(internalChannel && (configure().paramMask & Config::IGNORE_ZERO_WIDTHS))
        if(fabs(level->GetFitGamma(internalChannel))<1.0e-8) continue;
          for(int lap=1;lap<=compound()->GetJGroup(chanMGroup->GetJNum())->NumLevels();lap++) {
        if(compound()->GetJGroup(chanMGroup->GetJNum())->GetLevel(lap)->IsInRMatrix()) {
          ALevel *levelp=compound()->GetJGroup(chanMGroup->GetJNum())->GetLevel(lap);
          if(internalChannel && (configure().paramMask & Config::IGNORE_ZERO_WIDTHS))
            if(fabs(levelp->GetFitGamma(internalChannel))<1.0e-8) continue;
          umatrix+=2.0*complex(0.0,1.0)*
            point->GetSqrtPenetrability(chanMGroup->GetJNum(),chanMGroup->GetChNum())*
            level->GetFitGamma(chanMGroup->GetChNum())*
            levelp->GetFitGamma(chanMGroup->GetChpNum())*
            this->GetAMatrixElement(chanMGroup->GetJNum(),la,lap);
        }
          }
        }
      }
      tmatrix=tmatrix*umatrix;
    }
    this->AddECTMatrixElement(k,m,tmatrix);
      }
    }
  }
}

void AMatrixFunc::AddAInvMatrixElement(int jGroupNum, int lambdaNum, int muNum, complex
      aMatrixElement) {
  matrix_c e;
  vector_c f;
  while(jGroupNum>a_inv_matrices_.size()) a_inv_matrices_.push_back(e);
  while(lambdaNum>a_inv_matrices_[jGroupNum-1].size()) a_inv_matrices_[jGroupNum-1].push_back(f);
  a_inv_matrices_[jGroupNum-1][lambdaNum-1].push_back(aMatrixElement);
  assert(muNum=a_inv_matrices_[jGroupNum-1][lambdaNum-1].size());
}

void AMatrixFunc::AddAMatrix(matrix_c aMatrix) {
  a_matrices_.push_back(aMatrix);
}
```

8.228 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AngCoeff.cpp File Reference

```
#include "AngCoeff.h"
#include <gsl/gsl_sf_coupling.h>
#include <math.h>
```

8.229 AngCoeff.cpp

Go to the documentation of this file.

```
#include "AngCoeff.h"
#include <gsl/gsl_sf_coupling.h>
#include <math.h>

double AngCoeff::ClebGord(double j1, double j2, double j3, double m1, double m2, double m3) {
  m3=-m3;
  int j1x2=(int)(2*j1);
  int j2x2=(int)(2*j2);
  int j3x2=(int)(2*j3);
  int m1x2=(int)(2*m1);
```

```
  int m2x2=(int)(2*m2);
  int m3x2=(int)(2*m3);

  double w3j=gsl_sf_coupling_3j(j1x2,j2x2,j3x2,m1x2,m2x2,m3x2);

  return pow(-1.0,j1-j2-m3)*sqrt(2.0*j3+1.)*w3j;
}

double AngCoeff::Racah(double j1, double j2, double l2, double l1, double j3, double l3) {

  int j1x2=(int)(2*j1);
  int j2x2=(int)(2*j2);
  int j3x2=(int)(2*j3);
  int l1x2=(int)(2*l1);
  int l2x2=(int)(2*l2);
  int l3x2=(int)(2*l3);

  double w6j=gsl_sf_coupling_6j(j1x2,j2x2,j3x2,l1x2,l2x2,l3x2);

  return pow(-1.0,j1+j2+l2+l1)*w6j;
}
```

8.230 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZURE2.cpp File Reference

```
#include "AZUREMain.h"
#include "Config.h"
#include "NucLine.h"
#include "SegLine.h"
#include "ExtrapLine.h"
#include "GSLException.h"
#include <stdlib.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <gsl/gsl_errno.h>
#include <readline/readline.h>
#include <readline/history.h>
#include <string.h>
```

Classes

- struct SegPairs

Functions

- void welcomeMessage (const Config &configure)
- void exitMessage (const Config &configure)
- void printHelp ()
- bool parseOptions (int argc, char *argv[ ], Config &configure)
- int commandShell (const Config &configure)
- void processCommand (int command, Config &configure)
- void getParameterFile (bool useReadline, Config &configure)
- bool readSegmentFile (const Config &configure, std::vector< SegPairs > &segPairs)
- void getTemperatureFile (bool useReadline, Config &configure)
- void getRateParams (Config &configure, std::vector< SegPairs > &segPairs, bool useReadline)
- bool checkExternalCapture (Config &configure, const std::vector< SegPairs > &segPairs)
- void getExternalCaptureFile (bool useReadline, Config &configure)
- void startMessage (const Config &configure)
- int main (int argc, char *argv[ ])

8.230.1 Function Documentation

8.230.1.1 checkExternalCapture()

```
bool checkExternalCapture (
            Config & configure,
            const std::vector< SegPairs > & segPairs )
```

This function checks the external capture file against a vector of segment key pairs. Only if the calculation includes external capture segments is the user prompted for an integrals file. The appropriate configure flag is set here.

Definition at line 438 of file AZURE2.cpp.

8.230.1.2 commandShell()

```
int commandShell (
            const Config & configure )
```

This function handles the command shell in AZURE2. The function will not terminate until the user enters a valid integer option. Upon successful entry, the integer option is returned.

Definition at line 143 of file AZURE2.cpp.

8.230.1.3 exitMessage()

```
void exitMessage (
            const Config & configure )
```

This function prints a message upon successful termination of the program.

Definition at line 56 of file AZURE2.cpp.

8.230.1.4 getExternalCaptureFile()

```
void getExternalCaptureFile (
            bool useReadline,
            Config & configure )
```

This function promps the user for an external capture integrals file, checks for it's validity, and stores the path.

Definition at line 497 of file AZURE2.cpp.

8.230.1.5 getParameterFile()

```
void getParameterFile (
            bool useReadline,
            Config & configure )
```

This function prompts for a parameter file and sets the corresponding configure flags and variables based on the user response.

Definition at line 206 of file AZURE2.cpp.

8.230.1.6 getRateParams()

```
void getRateParams (
            Config & configure,
            std::vector< SegPairs > & segPairs,
            bool useReadline )
```

This function prompts the user for the required parameters if reaction rate is to be calculated.

Definition at line 351 of file AZURE2.cpp.

8.230.1.7 getTemperatureFile()

```
void getTemperatureFile (
            bool useReadline,
            Config & configure )
```

If reaction rate is desired, the user may specify a file containing temperatures for the calculation. This function prompts for that file name, checks for access, and stores path.

Definition at line 312 of file AZURE2.cpp.

8.230.1.8 main()

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

This is the main function. All the above initialization functions are called from here. The AZUREMain oject is created and called.

Definition at line 556 of file AZURE2.cpp.

8.230.1.9 parseOptions()

```
bool parseOptions (
            int argc,
            char * argv[],
            Config & configure )
```

This function parses the command line options given, and sets the appropriate variables in the Config structure. It also reads and parses the configuration file if the appropriate environment variable is set.

Definition at line 89 of file AZURE2.cpp.

8.230.1.10 printHelp()

```
void printHelp ( )
```

This function prints the response to the –help command which consists of available runtime options.

Definition at line 66 of file AZURE2.cpp.

8.230.1.11 processCommand()

```
void processCommand (
            int command,
            Config & configure )
```

This function takes the returned option from the commandShell function and sets the appropriate flags in the Config structure.

Definition at line 175 of file AZURE2.cpp.

8.230.1.12 readSegmentFile()

```
bool readSegmentFile (
            const Config & configure,
            std::vector< SegPairs > & segPairs )
```

This function reads the segment file, and stores the active entrance and exit pair keys for cross reference with the External capture file. Only if an active external capture segment is required is the user prompted for an external integrals file.

Definition at line 247 of file AZURE2.cpp.

8.230.1.13 startMessage()

```
void startMessage (
            const Config & configure )
```

This function prints a brief start message describing the type of calculation that will be performed.

Definition at line 539 of file AZURE2.cpp.

8.230.1.14 welcomeMessage()

```
void welcomeMessage (
            const Config & configure )
```

This function displays the welcome banner.

Definition at line 40 of file AZURE2.cpp.

8.231 AZURE2.cpp

Go to the documentation of this file.

```
/*
 * AZURE2.cpp
 * Ethan Uberseder, University of Notre Dame, 2011
 *
 * This file contains the main() function, as well as other C functions used to
 * initialize the program.  The functions defined here are (mostly) proceedural
 * C, and used to initialize the AZUREMain object from command line input.  All
 * actual calculations are managed by the AZUREMain object, and (almost) all
 * subsequent routines/classes follow object-oriented C++ programming conventions.
 */


#include "AZUREMain.h"
#include "Config.h"
#include "NucLine.h"
#include "SegLine.h"
#include "ExtrapLine.h"
#include "GSLException.h"
#include <stdlib.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <gsl/gsl_errno.h>
#ifndef NO_READLINE
#include <readline/readline.h>
#include <readline/history.h>
#endif
#include <string.h>

#ifdef GUI_BUILD
extern int start_gui(int argc, char *argv[]);
#endif
struct SegPairs {int firstPair; int secondPair;};

void welcomeMessage(const Config& configure) {
  configure.outStream « std::endl
        « "O--------------------------O-------------------------------O" « std::endl
        « "| #### #### #  # ###  ####  ##  | Version 1.0              |" « std::endl
        « "| #  #    # #  # #  # #    #  # |                          |" « std::endl
        « "| ####   #  #  # ###  ##      #  |                          |" « std::endl
        « "| #  #  #   #  # # #  #     #   | Joint Institute for      |" « std::endl
        « "| #  # ####  ##  #  # #### #### | Nuclear Astrophysics     |" « std::endl
        « "O--------------------------O-------------------------------O" « std::endl
        « std::endl;
}

void exitMessage(const Config& configure) {
  configure.outStream « std::endl
        « "Thanks for using AZURE2." « std::endl;
}

void printHelp() {
  std::cout  « "Syntax: AZURE2 <options> configfile" « std::endl « std::endl
         « "Options:" « std::endl
         « std::setw(25) « std::left « "\t--no-gui:" « std::setw(0) « "Do not use graphical setup utility (if built)." « std::endl
         « std::setw(25) « std::left « "\t" « std::setw(0) « "If this flag is not set all other options are ignored," « std::endl
         « std::setw(25) « std::left « "\t" « std::setw(0) « "and configuration occurs within the setup utility." « std::endl
#ifndef NO_READLINE
         « std::setw(25) « std::left « "\t--no-readline:" « std::setw(0) « "Do not use readline package." «  std::endl
#endif
         « std::setw(25) « std::left « "\t--no-transform:" « std::setw(0) « "Do not perform initial parameter transformations." « std::endl
         « std::setw(25) « std::left « "\t--no-long-wavelenth:" « std::setw(0) « "Do not use long wavelength approximation for EL capture." « std::endl
         « std::setw(25) « std::left « "\t--use-brune:" « std::setw(0) « "Use the alternative level matrix of C.R. Brune." « std::endl
         « std::setw(25) « std::left « "\t--ignore-externals:" « std::setw(0) « "Ignore external resonant capture amplitude if internal width is zero." « std::endl
         « std::setw(25) « std::left « "\t--use-rmc:" « std::setw(0) « "Use Reich-Moore approximation for capture (neutron capture only)." « std::endl
         « std::setw(25) « std::left « "\t--gsl-coul:" « std::setw(0) « "Use GSL Coulomb functions (faster, but less accurate)." « std::endl;
}

bool parseOptions(int argc, char *argv[], Config& configure) {
#ifndef NO_READLINE
  bool useReadline=true;
```

```
#else
  bool useReadline = false;
#endif
  configure.configfile="";
  std::vector<std::string> options;
  for(int i=1;i<argc;i++) {
    std::string arg=argv[i];
    if(!arg.empty()&&arg.substr(0,2)=="--") options.push_back(arg);
    else configure.configfile=arg;
  }
  char* optionsFile = getenv("AZURE_OPTIONS_FILE");
  if(optionsFile) {
    std::ifstream in(optionsFile);
    if(in) {
      while(!in.eof()) {
    std::string newOption;
    getline(in,newOption);
    if(!in.eof()) {
      std::string trimmedOption="";
      for(std::string::iterator it = newOption.begin();it<newOption.end();it++)
        if(*it!=' '&&*it!='\t'&&*it!='\n')
          trimmedOption+=*it;
      if(!trimmedOption.empty()&&trimmedOption.substr(0,2)=="--") options.push_back(trimmedOption);
    }
      }
      in.close();
    } else configure.outStream « "AZURE_OPTIONS_FILE variable set, but file not readable." « std::endl;
  }
  for(std::vector<std::string>::iterator it = options.begin();it<options.end();it++) {
#ifndef NO_READLINE
    if(*it=="--no-readline") useReadline=false;
    else if(*it=="--no-transform") configure.paramMask &= ~Config::TRANSFORM_PARAMETERS;
#else
    if (*it == "--no-transform") configure.paramMask &= ~Config::TRANSFORM_PARAMETERS;
#endif
    else if(*it=="--no-long-wavelength") configure.paramMask &= ~Config::USE_LONGWAVELENGTH_APPROX;
    else if(*it=="--use-brune") configure.paramMask |= Config::USE_BRUNE_FORMALISM;
    else if(*it=="--gsl-coul") configure.paramMask |= Config::USE_GSL_COULOMB_FUNC;
    else if(*it=="--ignore-externals") configure.paramMask |= Config::IGNORE_ZERO_WIDTHS;
    else if(*it=="--use-rmc") configure.paramMask |= Config::USE_RMC_FORMALISM;
    else if(*it=="--no-gui") continue;
    else configure.outStream « "WARNING: Unknown option " « *it « '.' « std::endl;
  }
  return useReadline;
}

int commandShell(const Config& configure) {
  int command=0;

  configure.outStream « "Please select from the following options: " « std::endl
        « "\t1. Calculate Segments From Data" « std::endl
        « "\t2. Fit Segments From Data" « std::endl
        « "\t3. Calculate Segments Without Data" « std::endl
        « "\t4. Perform MINOS Error Analysis" « std::endl
        « "\t5. Calculate Reaction Rate" « std::endl
        « "\t6. Exit" « std::endl;

  while(command<1||command>6) {
    configure.outStream « "azure2: ";
    std::string inString;
    getline(std::cin,inString);
    if(inString.empty()) continue;
    std::istringstream in;
    in.str(inString);
    if(!(in»command))
      configure.outStream « "Please enter an integer." « std::endl;
    else if(command<1||command>6)
      configure.outStream « "Invalid option.  Please try again."
        « std::endl;
  }
  return command;
}

void processCommand(int command, Config& configure) {
  if(command==2) configure.paramMask |= Config::PERFORM_FIT;
  else if(command==3) configure.paramMask &= ~Config::CALCULATE_WITH_DATA;
  else if(command==4) {
    bool goodAnswer=false;
    while (!goodAnswer) {
      configure.outStream « std::setw(30) « "Allowed Chi-Squared Variance: ";
      std::string inString;
      getline(std::cin,inString);
      std::istringstream stm;
      stm.str(inString);
      if(!(stm»configure.chiVariance) || configure.chiVariance<0.)
    configure.outStream « "Please enter a positive number." « std::endl;
```

```
      else goodAnswer=true;
    }
    configure.paramMask |= Config::PERFORM_FIT;
    configure.paramMask |= Config::PERFORM_ERROR_ANALYSIS;
  } else if(command==5) {
    configure.paramMask &= ~Config::CALCULATE_WITH_DATA;
    configure.paramMask |= Config::CALCULATE_REACTION_RATE;
  } else if(command==6) {
    exitMessage(configure);
    exit(0);
  }
}

void getParameterFile(bool useReadline, Config& configure) {
  bool validInfile=false;
  configure.outStream « std::endl;
  if(!useReadline) configure.outStream « "External Parameter File (leave blank for new file): ";
  while(!validInfile) {
    std::string inFile;
    if(!useReadline) getline(std::cin,inFile);
#ifndef NO_READLINE
    else {
      char *line = readline("External Parameter File (leave blank for new file): ");
      inFile=line;
      size_t endpos = inFile.find_last_not_of(" \t");
      if( std::string::npos != endpos ) inFile = inFile.substr( 0, endpos+1 );
      if(line && *line) add_history(line);
      free(line);
    }
#endif
    if(!inFile.empty()) {
      std::ifstream in;
      in.open(inFile.c_str());
      if(in) {
    validInfile=true;
    configure.paramMask |= Config::USE_PREVIOUS_PARAMETERS;
    configure.paramfile=inFile;
    in.close();
      }
      in.clear();
    } else validInfile=true;
    if(!validInfile) {
      configure.outStream « "Cannot Read From " « inFile « ". Please reenter file." « std::endl;
      if(!useReadline) configure.outStream « "External Parameter File (leave blank for new file): ";
    }
  }
}

bool readSegmentFile(const Config& configure,std::vector<SegPairs>& segPairs) {
  std::ifstream in;
  std::string startTag,stopTag;
  if(configure.paramMask & Config::CALCULATE_WITH_DATA) {
    startTag="<segmentsData>";
    stopTag="</segmentsData>";
  } else {
    startTag="<segmentsTest>";
    stopTag="</segmentsTest>";
  }
  in.open(configure.configfile.c_str());
  if(in) {
    std::string line="";
    while(line!=startTag&&!in.eof()) getline(in,line);
    if(line==startTag) {
      line="";
      while(line!=stopTag&&!in.eof()) {
    getline(in,line);
    bool empty=true;
    for(unsigned int i=0;i<line.size();++i)
      if(line[i]!=' '&&line[i]!='\t') {
        empty=false;
        break;
      }
    if(empty==true) continue;
    if(line!=stopTag&&!in.eof()) {
      std::istringstream stm;
      stm.str(line);
      if(configure.paramMask & Config::CALCULATE_WITH_DATA) {
        SegLine segment(stm);
        if(!(stm.rdstate() & (std::stringstream::failbit |
std::stringstream::badbit))&&segment.isActive()==1) {
          SegPairs tempSet={segment.entranceKey(),segment.exitKey()};
          segPairs.push_back(tempSet);
        }
      } else {
        ExtrapLine segment(stm);
        if(!(stm.rdstate() & (std::stringstream::failbit |
std::stringstream::badbit))&&segment.isActive()==1) {
```

```
          SegPairs tempSet={segment.entranceKey(),segment.exitKey()};
          segPairs.push_back(tempSet);
        }
      }
    }
      }
      if(line!=stopTag) {
    configure.outStream « "Problem reading segments. Check configuration file." « std::endl;
    return false;
      }
    } else {
      configure.outStream « "Problem reading segments. Check configuration file." « std::endl;
      return false;
    }
    in.close();
  } else {
    configure.outStream « "Cannot read segments. Check configuration file." « std::endl;
    return false;
  }
  in.clear();
  return true;
}

void getTemperatureFile(bool useReadline, Config& configure) {
  bool validInfile=false;
  if(!useReadline) configure.outStream « std::setw(38) « "Temperature File Name: ";
  while(!validInfile) {
    std::string inFile;
    if(!useReadline) getline(std::cin,inFile);
#ifndef NO_READLINE
    else {
      char *line = readline("               Temperature File Name: ");
      inFile=line;
      size_t endpos = inFile.find_last_not_of(" \t");
      if( std::string::npos != endpos ) inFile = inFile.substr( 0, endpos+1 );
      if(line && *line) add_history(line);
      free(line);
    }
#endif
    if(!inFile.empty()) {
      std::ifstream in;
      in.open(inFile.c_str());
      if(in) {
    validInfile=true;
    configure.rateParams.temperatureFile=inFile;
    in.close();
      }
      in.clear();
    }
    if(!validInfile) {
      if(inFile.empty()) configure.outStream « "Please enter a file name." « std::endl;
      else configure.outStream « "Cannot Read From " « inFile « ". Please reenter file." « std::endl;
      if(!useReadline) configure.outStream « "               Temperature File Name: ";
    }
  }
}

void getRateParams(Config& configure, std::vector<SegPairs>& segPairs,bool useReadline) {
  configure.rateParams.entrancePair=0;
  configure.rateParams.exitPair=0;
  configure.rateParams.minTemp=-1.;
  configure.rateParams.maxTemp=-1.;
  configure.rateParams.tempStep=-1.;
  while(configure.rateParams.entrancePair==configure.rateParams.exitPair){
    while(!configure.rateParams.entrancePair) {
      configure.outStream « std::setw(38) « "Reaction Rate Entrance Pair: ";
      std::string inString;
      getline(std::cin,inString);
      std::istringstream stm;
      stm.str(inString);
      if(!(stm » configure.rateParams.entrancePair) || configure.rateParams.entrancePair==0)
    configure.outStream « "Please enter an integer greater than zero." « std::endl ;
    }
    while(!configure.rateParams.exitPair) {
      configure.outStream « std::setw(38) « "Reaction Rate Exit Pair: ";
      std::string inString;
      getline(std::cin,inString);
      std::istringstream stm;
      stm.str(inString);
      if(!(stm » configure.rateParams.exitPair) || configure.rateParams.exitPair==0)
    configure.outStream « "Please enter an integer greater than zero." « std::endl;
    }
    if(configure.rateParams.entrancePair==configure.rateParams.exitPair) {
      configure.outStream « "Cannot calculate rate for elastic scattering." « std::endl;
      configure.rateParams.entrancePair=0;configure.rateParams.exitPair=0;
    }
  }
```

```
  SegPairs tempSet={configure.rateParams.entrancePair,configure.rateParams.exitPair};
  segPairs.push_back(tempSet);
  bool goodAnswer = false;
  std::string fileAnswer="";
  while(!goodAnswer) {
    configure.outStream « std::setw(38) « "Use temperatures from file (yes/no): ";
    getline(std::cin,fileAnswer);
    std::string trimmedAnswer;
    for(int i =0;i<fileAnswer.length();i++)
      if(fileAnswer[i]!=' '&&fileAnswer[i]!='\t'&&fileAnswer[i]!='\n')
    trimmedAnswer+=fileAnswer[i];
    if(trimmedAnswer!="yes"&&trimmedAnswer!="no")
     configure.outStream « "Please type 'yes' or 'no'." «std::endl;
    else {
      goodAnswer = true;
      fileAnswer=trimmedAnswer;
    }
  }
  configure.rateParams.useFile = (fileAnswer=="yes") ? (true) : (false);
  if(configure.rateParams.useFile) getTemperatureFile(useReadline,configure);
  else {
    while(configure.rateParams.minTemp<0.) {
      configure.outStream « std::setw(38) « "Reaction Rate Min Temp [GK]: ";
      std::string inString;
      getline(std::cin,inString);
      std::istringstream stm;
      stm.str(inString);
      if(!(stm » configure.rateParams.minTemp) || configure.rateParams.minTemp<0.)
    configure.outStream « "Please enter a positive number." « std::endl;
    }
    while(configure.rateParams.maxTemp<0.) {
      configure.outStream « std::setw(38) « "Reaction Rate Max Temp [GK]: ";
      std::string inString;
      getline(std::cin,inString);
      std::istringstream stm;
      stm.str(inString);
      if(!(stm » configure.rateParams.maxTemp) || configure.rateParams.maxTemp<0.)
    configure.outStream « "Please enter a positive number." « std::endl;
    }
    while(configure.rateParams.tempStep<0.) {
      configure.outStream « std::setw(38) « "Reaction Rate Temp Step [GK]: ";
      std::string inString;
      getline(std::cin,inString);
      std::istringstream stm;
      stm.str(inString);
      if(!(stm » configure.rateParams.tempStep) || configure.rateParams.tempStep<0.)
    configure.outStream « "Please enter a positive number." « std::endl;
    }
  }
}

bool checkExternalCapture(Config& configure, const std::vector<SegPairs>& segPairs) {
  std::ifstream in;
  in.open(configure.configfile.c_str());
  if(!in) {
    configure.outStream « "Cannot read nuclear information. Check configuration file." « std::endl;
    return false;
  }
  std::string line="";
  while(line!="<levels>"&&!in.eof()) getline(in,line);
  if(line!="<levels>") {
    configure.outStream « "Cannot read nuclear information. Check configuration file." « std::endl;
    return false;
  }
  line="";
  while(line!="</levels>"&&!in.eof()&&
    !(configure.paramMask & Config::USE_EXTERNAL_CAPTURE)) {
    getline(in,line);
    bool empty=true;
    for(unsigned int i=0;i<line.size();++i)
      if(line[i]!=' '&&line[i]!='\t') {
    empty=false;
    break;
      }
    if(empty==true) continue;
    if(line!="</levels>"&&!in.eof()) {
      std::istringstream stm;
      stm.str(line);
      NucLine tempNucLine(stm);
      if((stm.rdstate() & (std::stringstream::failbit | std::stringstream::badbit))) {
    configure.outStream « "Problem reading nuclear information. Check configuration file." «
std::endl;
    return false;
      }
      if(tempNucLine.ecMultMask()!=0) {
    for(int i=0;i<segPairs.size();i++) {
      if(tempNucLine.ir()==segPairs[i].secondPair||
```

```
         segPairs[i].secondPair==-1) {
        configure.paramMask |= Config::USE_EXTERNAL_CAPTURE;
        break;
      }
    }
      }
    }
  }
  in.close();
  in.clear();
  if((configure.paramMask & Config::USE_EXTERNAL_CAPTURE)&&
     (configure.paramMask & Config::USE_RMC_FORMALISM)) {
    configure.outStream « "WARNING: External capture is not compatible with Reich-Moore.  Ignoring external capture."
            « std::endl;
    configure.paramMask &= ~Config::USE_EXTERNAL_CAPTURE;
  }
  return true;
}

void getExternalCaptureFile(bool useReadline, Config& configure) {
  if((configure.paramMask & Config::USE_EXTERNAL_CAPTURE)&&
     !(configure.paramMask & Config::CALCULATE_REACTION_RATE)) {
    configure.outStream « std::endl;
    if(!useReadline) configure.outStream « "External Capture Amplitude File (leave blank for new file): ";
    bool validInfile=false;
    while(!validInfile) {
      std::string inFile;
      if(!useReadline) getline(std::cin,inFile);
#ifndef NO_READLINE
      else {
        char *line = readline("External Capture Amplitude File (leave blank for new file): ");
        inFile=line;
        size_t endpos = inFile.find_last_not_of(" \t");
        if( std::string::npos != endpos ) inFile = inFile.substr( 0, endpos+1 );
        if(line && *line) add_history(line);
        free(line);
      }
#endif
      if(!inFile.empty()) {
    std::ifstream in;
    in.open(inFile.c_str());
    if(in) {
      validInfile=true;
      configure.paramMask |= Config::USE_PREVIOUS_INTEGRALS;
      configure.integralsfile=inFile;
      in.close();
    }
    in.clear();
      } else validInfile=true;
      if(!validInfile) {
    configure.outStream « "Cannot Read From " « inFile « ". Please reenter file." « std::endl;
    if(!useReadline) configure.outStream « "External Capture Amplitude File (leave blank for new file): ";
      }
    }
  }
}

void startMessage(const Config& configure) {
  if(configure.paramMask & Config::PERFORM_ERROR_ANALYSIS)
    configure.outStream « "Calling AZURE2 for MINOS Error Analysis..." « std::endl;
  else if(configure.paramMask & Config::CALCULATE_REACTION_RATE)
    configure.outStream « "Calling AZURE2 for reaction rate calculation..." « std::endl;
  else if(!(configure.paramMask & Config::CALCULATE_WITH_DATA))
    configure.outStream « "Calling AZURE2 for calculation of segments without data..." « std::endl;
  else if(configure.paramMask & Config::PERFORM_FIT)
    configure.outStream « "Calling AZURE2 for fitting of segments from data..." « std::endl;
  else  configure.outStream « "Calling AZURE2 for calculation of segments from data..." « std::endl;
}

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

  //Check for --help option first.  If set, print help and exit.
  for(int i=1;i<argc;i++)
    if(strcmp(argv[i],"--help")==0) {
      printHelp();
      return 0;
    }

  //Set GSL Error Handler
  gsl_set_error_handler (&GSLException::GSLErrorHandler);

  //If GUI is built, look for --no-gui option.  If not set, hand control to GUI.
#ifdef GUI_BUILD
  bool useGUI=true;
```

```
  for(int i=1;i<argc;i++)
    if(strcmp(argv[i],"--no-gui")==0) useGUI=false;
  if(useGUI) return start_gui(argc,argv);
#endif

  //Create new configuration structure, and parse the command line parameters
  Config configure(std::cout);
  bool useReadline = parseOptions(argc,argv,configure);

  //Read the parameters from the runtime configuration file
  if(configure.configfile.empty()) {
    configure.outStream « "A valid configuration file must be specified." « std::endl
          « "\tSyntax: AZURE2 <options> configfile" « std::endl;
    return -1;
  }
  if(configure.ReadConfigFile()==-1) {
    configure.outStream « "Could not open " « configure.configfile « ".  Check that file exists."
          « std::endl;
    return -1;
  }
#ifndef NO_STAT
  else if(configure.CheckForInputFiles() == -1) return -1;
#endif
  if((configure.paramMask & Config::USE_RMC_FORMALISM) && (configure.paramMask & Config::USE_BRUNE_FORMALISM)) {
    configure.outStream « "WARNING: --use-brune is incompatible with --use-rmc. Ignoring --use-brune." « std::endl;
    configure.paramMask &= ~Config::USE_BRUNE_FORMALISM;
  }
  if((configure.paramMask & Config::USE_BRUNE_FORMALISM)||(configure.paramMask & Config::IGNORE_ZERO_WIDTHS)) {
    if(!(configure.paramMask & Config::USE_AMATRIX))
      configure.outStream « "WARNING: R-Matrix specified but --ignore-externals and --use-brune options require A-Matrix.  A-Matrix will be used."
        « std::endl;
    configure.paramMask |= Config::USE_AMATRIX;
  }

  //Print welcome message
  welcomeMessage(configure);

  //Read and process command, setting appropriate configuration flags
  processCommand(commandShell(configure),configure);

  //Open history file for readline
#ifndef NO_READLINE
  if(useReadline) read_history("./.azure_history");
#endif

  //Read the external parameter file to be used, if any
  getParameterFile(useReadline,configure);

  //Parse the segment files for entrance,exit pairs
  std::vector<SegPairs> segPairs;
  if(!(configure.paramMask & Config::CALCULATE_REACTION_RATE)) {
    if(!readSegmentFile(configure,segPairs)) exit(1);
  } else getRateParams(configure,segPairs,useReadline);

  //Check if the entrance,exit pairs are in the external capture file
  // If so, external capture will be needed
  if(!checkExternalCapture(configure,segPairs)) exit(1);

  //Read the external capture file name to be used, if any
  getExternalCaptureFile(useReadline,configure);

  //Create instance of main AZURE function, print start message,
  // and execute
  AZUREMain azureMain(configure);
  configure.outStream « std::endl; startMessage(configure);
  int returnValue = azureMain();

  //Print exit message
  exitMessage(configure);

  //Write readline history file
#ifndef NO_READLINE
  if(useReadline) write_history("./.azure_history");
#endif

  return returnValue;
}
```

8.232 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZURECalc.cpp File Reference

```
#include "AZURECalc.h"
#include "Config.h"
#include "CNuc.h"
#include "EData.h"
#include <iostream>
#include <iomanip>
```

8.233 AZURECalc.cpp

Go to the documentation of this file.

```
#include "AZURECalc.h"
#include "Config.h"
#include "CNuc.h"
#include "EData.h"
#include <iostream>
#include <iomanip>

double AZURECalc::operator()(const vector_r& p) const {

  int thisIteration=data()->Iterations();
  data()->Iterate();
  bool isFit=data()->IsFit();

  CNuc * localCompound = NULL;
  EData *localData = NULL;
  if(isFit) {
    localCompound = compound()->Clone();
    localData = data()->Clone();
  } else {
    localCompound = compound();
    localData = data();
  }

  //Fill Compound Nucleus From Minuit Parameters
  localCompound->FillCompoundFromParams(p);
  localData->FillNormsFromParams(p);
  if(configure().paramMask & Config::USE_BRUNE_FORMALISM)
localCompound->CalcShiftFunctions(configure());

  //loop over segments and points
  double chiSquared=0.0;
  double segmentChiSquared=0.0;
  ESegmentIterator firstSumIterator = localData->GetSegments().end();
  ESegmentIterator lastSumIterator = localData->GetSegments().end();
  for(EDataIterator data=localData->begin();data!=localData->end();data++) {
    if(data.segment()->GetPoints().begin()==data.point()) {
      segmentChiSquared=0.0;
      if(data.segment()->IsTotalCapture()) {
    firstSumIterator=data.segment();
    lastSumIterator=data.segment()+data.segment()->IsTotalCapture()-1;
      }
    }
    if(!data.point()->IsMapped()) data.point()->Calculate(localCompound,configure());
    if(firstSumIterator!=localData->GetSegments().end()&&
       data.segment()!=lastSumIterator) continue;
    double fitCrossSection=data.point()->GetFitCrossSection();
    ESegmentIterator thisSegment = data.segment();
    if(data.segment()==lastSumIterator) {
      int pointIndex=data.point()-data.segment()->GetPoints().begin()+1;
      for(ESegmentIterator it=firstSumIterator;it<data.segment();it++)
    fitCrossSection+=it->GetPoint(pointIndex)->GetFitCrossSection();
      thisSegment = firstSumIterator;
    }
    double dataNorm=thisSegment->GetNorm();
    double CrossSection=data.point()->GetCMCrossSection()*dataNorm;
    double CrossSectionError=data.point()->GetCMCrossSectionError()*dataNorm;
    double chi=(fitCrossSection-CrossSection)/CrossSectionError;
    double pointChiSquared=pow(chi,2.0);
    segmentChiSquared+=pointChiSquared;
    if(data.segment()->GetPoints().end()-1==data.point()) {
      if(!isFit) thisSegment->SetSegmentChiSquared(segmentChiSquared);
```

```
      if(data.segment()==lastSumIterator) {
    firstSumIterator=localData->GetSegments().end();
    lastSumIterator=localData->GetSegments().end();
      }
      double dataNormNominal=thisSegment->GetNominalNorm();
      double dataNormError=dataNormNominal/100.*thisSegment->GetNormError();
      if(dataNormError!=0.)
    segmentChiSquared += pow((dataNorm-dataNormNominal)/dataNormError,2.0);
      chiSquared+=segmentChiSquared;
    }
  }

  if(!localData->IsErrorAnalysis()&&thisIteration!=0) {
    if(thisIteration%100==0) configure().outStream
                   « "\r\tIteration: " « std::setw(6) « thisIteration
                   « " Chi-Squared: " « chiSquared;  configure().outStream.flush();

    if(thisIteration%1000==0) {
      localData->WriteOutputFiles(configure(),isFit);
      localCompound->TransformOut(configure());
      localCompound->PrintTransformParams(configure());
    }
  }
  if(isFit) {
    delete localCompound;
    delete localData;
  }
  if(configure().stopFlag&&isFit) return 0.;
  else return chiSquared;
}
```

8.234 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREMain.cpp File Reference

#include "AZURECalc.h"
#include "AZUREMain.h"
#include "AZUREParams.h"
#include "Config.h"
#include "ReactionRate.h"
#include "Minuit2/MnPrint.h"
#include "GSLException.h"
#include <Minuit2/FunctionMinimum.h>
#include <Minuit2/MnMigrad.h>
#include <Minuit2/MnMinos.h>

8.235 AZUREMain.cpp

Go to the documentation of this file.

```
#include "AZURECalc.h"
#include "AZUREMain.h"
#include "AZUREParams.h"
#include "Config.h"
#include "ReactionRate.h"
#include "Minuit2/MnPrint.h"
#include "GSLException.h"
#include <Minuit2/FunctionMinimum.h>
#include <Minuit2/MnMigrad.h>
#include <Minuit2/MnMinos.h>

int AZUREMain::operator()(){
  //Fill compound nucleus from nucfile
  configure().outStream « "Filling Compound Nucleus..." « std::endl;
  if(compound()->Fill(configure())==-1) {
    configure().outStream « "Could not fill compound nucleus from file."
          « std::endl;
    return -1;
  } else if(compound()->NumPairs()==0 || compound()->NumJGroups()==0) {
    configure().outStream « "No nuclear data exists. Calculation not possible." « std::endl;
```

```
    return -1;
  }
  if((configure().screenCheckMask|configure().fileCheckMask) &
     Config::CHECK_COMPOUND_NUCLEUS) compound()->PrintNuc(configure());

  if(!(configure().paramMask & Config::CALCULATE_REACTION_RATE)) {
    //Fill the data object from the segments and data file
    //  Compound object is passed to the function for pair key verification and
    //  center of mass conversions, s-factor conversions, etc.
    configure().outStream « "Filling Data Structures..." « std::endl;
    if(configure().paramMask & Config::CALCULATE_WITH_DATA) {
      if(data()->Fill(configure(),compound())==-1) {
    configure().outStream « "Could not fill data object from file." « std::endl;
    return -1;
      } else if(data()->NumSegments()==0) {
    configure().outStream « "There is no data provided." « std::endl;
    return -1;
      }
    } else {
      if(data()->MakePoints(configure(),compound())==-1) {
    configure().outStream « "Could not fill data object from file." « std::endl;
    return -1;
      } else if(data()->NumSegments()==0) {
    configure().outStream « "Extrapolation segments produce no data." « std::endl;
    return -1;
      }
    }
    if((configure().fileCheckMask|configure().screenCheckMask) & Config::CHECK_DATA)
      data()->PrintData(configure());
  } else {
    if(!compound()->IsPairKey(configure().rateParams.entrancePair)||!compound()->IsPairKey(configure().rateParams.exitPair))
    {
      configure().outStream « "Reaction rate pairs do not exist in compound nucleus." « std::endl;
      return -1;
    } else {
      compound()->GetPair(compound()->GetPairNumFromKey(configure().rateParams.entrancePair))->SetEntrance();
    }
  }

  //Initialize compound nucleus object
  try {
    compound()->Initialize(configure());
  } catch (GSLException e) {
    configure().outStream « e.what() « std::endl;
    configure().outStream « std::endl
                « "Calculation was aborted." « std::endl;
    return -1;
  }

  //Create new parameters for minuit, fill them from compound nucleus object and data file.
  AZUREParams params;
  compound()->FillMnParams(params.GetMinuitParams());
  data()->FillMnParams(params.GetMinuitParams());
  if(!(configure().paramMask & Config::USE_PREVIOUS_PARAMETERS)) {
    configure().outStream « "Creating New param.par File..." « std::endl;
    params.WriteUserParameters(configure(),false);
  } else {
    configure().outStream « "Reading User Parameter File..." « std::endl;
    params.ReadUserParameters(configure());
  }

  if(!(configure().paramMask & Config::CALCULATE_REACTION_RATE)) {
    //Initialize data object
    if(data()->Initialize(compound(),configure())==-1) {
      configure().outStream « std::endl
                « "Calculation was aborted." « std::endl;
      return -1;
    }

    //Declare a new instance of FCNBase
    AZURECalc theFunc(data(),compound(),configure());
    theFunc.SetErrorDef(1.0);

    if(configure().paramMask & Config::PERFORM_FIT) {
      //Call Minuit for function minimization, write minimized parameters to params
      if(configure().paramMask & Config::USE_AMATRIX) configure().outStream « "Performing A-Matrix Fit..." « std::endl;
      else configure().outStream « "Performing R-Matrix Fit..." « std::endl;
      data()->SetFit(true);
      ROOT::Minuit2::MnMigrad migrad(theFunc,params.GetMinuitParams());
      ROOT::Minuit2::FunctionMinimum min=migrad(50000);
      if(configure().paramMask & Config::PERFORM_ERROR_ANALYSIS) {
    configure().outStream « std::endl
          « "Performing parameter error analysis with Up=" «  configure().chiVariance « "." « std::endl;
```

```
    data()->SetErrorAnalysis(true);
    theFunc.SetErrorDef(configure().chiVariance);
    ROOT::Minuit2::MnMinos minos(theFunc,min);
    std::vector<std::pair<double,double> > errors;
    for(int i = 0; i<params.GetMinuitParams().Params().size(); i++) {
      configure().outStream « "\tParameter " « i+1 « "..." « std::endl;
      if(!params.GetMinuitParams().Parameter(i).IsFixed()) {
        std::pair< double, double > error=minos(i);
        errors.push_back(error);
      } else errors.push_back(std::pair< double, double > (0.,0.));
    }

        // New output of the covariance matrix
        char filename[256];
        sprintf(filename,"%scovariance_matrix.out",configure().outputdir.c_str());
        std::ofstream out;
        out.open(filename);
        if(out) {

          // Write header information
          params.GetMinuitParams()=min.UserParameters();
          out « "Parameter List" « std::endl « std::endl;
          for(int i = 0; i<params.GetMinuitParams().Params().size(); i++) {
            out « std::setw(20) « params.GetMinuitParams().GetName(i)
                « " (" « std::setw(3) « i « ")"
                « std::scientific « std::setw(20) «  params.GetMinuitParams().Value(i);
            if(params.GetMinuitParams().Parameter(i).IsFixed()) {
          out « " Fixed" « std::endl;
            }
        else {
              out « " Fitted by Minuit" « std::endl;
            }
      }

      std::cout « min.UserCovariance();

      std::vector <double> CovarianceData;
          CovarianceData=min.UserCovariance().Data();
          int parameterTable[100];
          int size=0;

          // Header for Covariance Matrix
          out « std::endl «  "Covariance Matrix" « std::endl « std::endl;
          out « std::setw(5) « " ";
          for(int i = 0; i<params.GetMinuitParams().Params().size(); i++) {
            if(!params.GetMinuitParams().Parameter(i).IsFixed()) {
              out « std::setw(15) « i ;
              parameterTable[size]=i;
              size=size+1;
            }
          }
          out « std::endl;

      std::cout « "covariance length " « CovarianceData.size() « std::endl;

          // Covariance matrix
          for(int i = 0; i<size; i++) {
            out « std::setw(5) « parameterTable[i];
            for(int j = 0; j<size; j++) {
              int k;
              if(i<=j) { k=((j+1)*j)/2+i; }
              if(i>j)  { k=((i+1)*i)/2+j; }
          std::cout « std::setw(5) « i « std::setw(5) « j « std::setw(5) « k « std::endl;
              out « std::setw(15) « CovarianceData[k];
            }
            out « std::endl;
      }

          // Header for Correlation Matrix
          out « std::endl «  "Correlation Matrix" « std::endl « std::endl;
          out « std::setw(5) « " ";
          for(int i = 0; i<params.GetMinuitParams().Params().size(); i++) {
            if(!params.GetMinuitParams().Parameter(i).IsFixed()) {
              out « std::setw(15) « i ;
            }
          }
          out « std::endl;

          // Correlation matrix
          for(int i = 0; i<size; i++) {
            out « std::setw(5) « parameterTable[i];
            for(int j = 0; j<size; j++) {
              int k;
              if(i<=j) { k=((j+1)*j)/2+i; }
              if(i>j)  { k=((i+1)*i)/2+j; }
              int jdiag=((j+2)*(j+1))/2-1;
              int idiag=((i+2)*(i+1))/2-1;
```

```
          std::cout « k « "  " « CovarianceData[k] « std::endl;
              out « std::setw(15) « std::fixed «
CovarianceData[k]/sqrt(fabs(CovarianceData[jdiag]*CovarianceData[idiag]));
            }
            out « std::endl;
      }

          out.flush();
          out.close();
        } else std::cout « "Could not save " « configure().outputdir.c_str()
                         « "covariance_matrix.out file." « std::endl;

        // ECS this line added to set the azure variables to the minimised fit parameters
        params.GetMinuitParams()=min.UserParameters();
    params.WriteParameterErrors(errors,configure());

      }
      params.GetMinuitParams()=min.UserParameters();
      params.WriteUserParameters(configure(),true);
    } else {
      if(configure().paramMask & Config::USE_AMATRIX) configure().outStream « "Performing A-Matrix
Calculation..." « std::endl;
      else configure().outStream « "Performing R-Matrix Calculation..." « std::endl;
    }
    data()->SetFit(false);
    data()->SetErrorAnalysis(false);
    double chiSquared=theFunc(params.GetMinuitParams().Params());
    if(configure().paramMask & Config::CALCULATE_WITH_DATA) {
      configure().outStream « std::endl « std::endl;
      for(ESegmentIterator segment=data()->GetSegments().begin();
      segment<data()->GetSegments().end();segment++) {
    configure().outStream « "Segment #"
          « segment->GetSegmentKey()
          « " Chi-Squared/N: "
          « segment->GetSegmentChiSquared()/segment->NumPoints()
          « std::endl;
    if(segment->IsTotalCapture()) segment+=segment->IsTotalCapture()-1;
      }
      configure().outStream « "Total Chi-Squared: "
        « chiSquared « std::endl « std::endl;
    }

    //Write Output Files
    configure().outStream « "Writing output files..." « std::endl;
    data()->WriteOutputFiles(configure());
  } else {
    //Calculate Reaction Rate
    // This uses the adaptive integration routines of GSL.  As the energy stepsize is
    // unknown until integration, AZURE is called not in segment control mode but as
    // an energy dependent function.  As every data point must be reinitialized (new energy
    // dependent terms calculated) the routine is slow.  This should be a tradeoff
    // for good accuracy.
    configure().outStream « "Performing reaction rate calculation..." « std::endl;
    ReactionRate reactionRate(compound(),params.GetMinuitParams().Params(),configure(),
                  configure().rateParams.entrancePair,configure().rateParams.exitPair);
    if(configure().paramMask & Config::USE_BRUNE_FORMALISM)
      compound()->CalcShiftFunctions(configure());
    if(configure().rateParams.useFile)
      reactionRate.CalculateFileRates();
    else
      reactionRate.CalculateRates();
    reactionRate.WriteRates();
  }

  configure().outStream « "Performing final parameter transformation..." « std::endl;
  try {
    compound()->TransformOut(configure());
  } catch (GSLException e) {
    configure().outStream « e.what() « std::endl;
    configure().outStream « "Problem with output transformation.  Aborting."
              « std::endl;
    return -1;
  }
  compound()->PrintTransformParams(configure());

  return 0;
}
```

8.236 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREOutput.cpp File Reference

```
#include "AZUREOutput.h"
```

8.237 AZUREOutput.cpp

Go to the documentation of this file.

```
#include "AZUREOutput.h"

AZUREOutput::AZUREOutput(std::string outputdir) {
  outputdir_=outputdir;
  is_extrap_=false;
}

AZUREOutput::~AZUREOutput() {
  for(int i=0;i<azurefbuffers_.size();i++) delete azurefbuffers_[i];
}

bool AZUREOutput::IsExtrap() const {
  return is_extrap_;
}

std::filebuf *AZUREOutput::operator()(int entranceKey, int exitKey, bool isAngDist) {
  int c=this->IsAZUREFBuffer(entranceKey,exitKey,isAngDist);
  if(!c) {
    AZUREFBuffer *d = new AZUREFBuffer(entranceKey,exitKey,this->GetOutputDir(),this->IsExtrap(),isAngDist);
    this->AddAZUREFBuffer(d);
    c=this->IsAZUREFBuffer(entranceKey,exitKey,isAngDist);
  }
  return this->GetAZUREFBuffer(c)->GetFBuffer();
}

int AZUREOutput::NumAZUREFBuffers() const {
  return azurefbuffers_.size();
}

int AZUREOutput::IsAZUREFBuffer(int entranceKey, int exitKey, bool isAngDist) {
  bool c=false;
  int d=0;
  while(!c&&d<this->NumAZUREFBuffers()) {
    if(this->GetAZUREFBuffer(d+1)->IsAngDist()==isAngDist&&
       this->GetAZUREFBuffer(d+1)->GetEntranceKey()==entranceKey&&
       this->GetAZUREFBuffer(d+1)->GetExitKey()==exitKey) c=true;
    d++;
  }
  if(c) return d;
  else return 0;
}

std::string AZUREOutput::GetOutputDir() const {
  return outputdir_;
};

void AZUREOutput::AddAZUREFBuffer(AZUREFBuffer *azureFBuffer) {
  azurefbuffers_.push_back(azureFBuffer);
}

void AZUREOutput::SetExtrap() {
  is_extrap_=true;
}

AZUREFBuffer *AZUREOutput::GetAZUREFBuffer(int fBufferNum) {
  AZUREFBuffer *b=azurefbuffers_[fBufferNum-1];
  return b;
}

```

8.238 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREParams.cpp File Reference

```
#include ¨AZUREParams.h¨
#include ¨Config.h¨
```

8.239 AZUREParams.cpp

Go to the documentation of this file.

```
#include "AZUREParams.h"
#include "Config.h"

ROOT::Minuit2::MnUserParameters &AZUREParams::GetMinuitParams() {
  return params_;
}

void AZUREParams::ReadUserParameters(const Config& configure) {
  std::vector<std::string> names;
  vector_r values;
  vector_r errors;
  std::vector<bool> fixed;
  std::string tempname,tempfixed,tempfixed_nows;
  double tempvalue,temperror;

  std::ifstream in;
  in.open(configure.paramfile.c_str());
  if(in) {
    while(!in.eof()) {
      in » tempname » tempvalue » temperror; getline(in,tempfixed);
      if(!in.eof()) {
    names.push_back(tempname);
    values.push_back(tempvalue);
    errors.push_back(temperror);
    tempfixed_nows.clear();
    for(int i=0;i<tempfixed.length();i++)
      if(tempfixed[i]!=' '&&tempfixed[i]!='\t')
        tempfixed_nows.push_back(tempfixed[i]);
    if(tempfixed_nows=="fixed") fixed.push_back(true);
    else fixed.push_back(false);
      }
    }
    in.close();
  } else configure.outStream « "Could not read user parameter file." « std::endl;

  for(int i=0;i<GetMinuitParams().Params().size();i++) {
    for(int ii=0;ii<names.size();ii++) {
      if(GetMinuitParams().GetName(i)==names[ii]) {
    if(GetMinuitParams().Value(i)==0.0&&values[ii]!=0.0&&
       GetMinuitParams().Parameter(i).IsFixed()) GetMinuitParams().Release(i);
    if(GetMinuitParams().Value(i)!=0.0&&values[ii]==0.0&&
       !GetMinuitParams().Parameter(i).IsFixed()) GetMinuitParams().Fix(i);
    GetMinuitParams().SetValue(i,values[ii]);
    GetMinuitParams().SetError(i,errors[ii]);
    if(fixed[ii]&&!GetMinuitParams().Parameter(i).IsFixed()) GetMinuitParams().Fix(i);
      }
    }
  }
}

void AZUREParams::WriteUserParameters(const Config& configure, bool fitParameters) {
  char filename[256];
  if(fitParameters) sprintf(filename,"%sparam.sav",configure.outputdir.c_str());
  else sprintf(filename,"%sparam.par",configure.outputdir.c_str());
  std::ofstream out;
  out.open(filename);
  if(out) {
    out.precision(7);
    for(int i=0;i<GetMinuitParams().Params().size();i++) {
      out « std::setw(20) « GetMinuitParams().GetName(i)
      « std::scientific « std::setw(20) «  GetMinuitParams().Value(i)
      « std::scientific « std::setw(20) «  GetMinuitParams().Error(i) « std::endl;
    }
    out.flush();
    out.close();
  } else configure.outStream « "Could not save param.par file." « std::endl;
}
```

```

void AZUREParams::WriteParameterErrors(const std::vector<std::pair<double,double> > &errors,const Config& configure) {
  char filename[256];
  sprintf(filename,"%sparam.errors",configure.outputdir.c_str());
  std::ofstream out;
  out.open(filename);
  if(out) {
    out.precision(7);
    for(int i=0;i<GetMinuitParams().Params().size();i++) {
      out « std::setw(20) « GetMinuitParams().GetName(i)
      « std::scientific « std::setw(20) «  GetMinuitParams().Value(i)
      « std::scientific « std::setw(20) «  fabs(errors[i].first)
      « std::scientific « std::setw(20) «  fabs(errors[i].second)
      « std::endl;
    }
    out.flush();
    out.close();
  } else configure.outStream « "Could not save param.errors file." « std::endl;
}
```

8.240 /Users/kuba/Desktop/R-Matrix/AZURE2/src/CNuc.cpp File Reference

```
#include <iostream>
#include <iomanip>
#include <sstream>
#include "AngCoeff.h"
#include "CNuc.h"
#include "Config.h"
#include "CoulFunc.h"
#include "EigenFunc.h"
#include "ECIntegral.h"
#include "NucLine.h"
#include "Minuit2/MnUserParameters.h"
#include "NFIntegral.h"
#include "ShftFunc.h"
```

8.241 CNuc.cpp

Go to the documentation of this file.

```
#include <iostream>
#include <iomanip>
#include <sstream>
#include "AngCoeff.h"
#include "CNuc.h"
#include "Config.h"
#include "CoulFunc.h"
#include "EigenFunc.h"
#include "ECIntegral.h"
#include "NucLine.h"
#include "Minuit2/MnUserParameters.h"
#include "NFIntegral.h"
#include "ShftFunc.h"

bool CNuc::IsPairKey(int key) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumPairs()) {
    if(key==this->GetPair(c+1)->GetPairKey()) b=true;
    c++;
  }
  return b;
}

int CNuc::NumPairs() const {
```

```
  return pairs_.size();
}

int CNuc::NumJGroups() const {
  return jgroups_.size();
}

int CNuc::IsPair(PPair pair) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumPairs())
    {
      if(pair.GetPairKey()==this->GetPair(c+1)->GetPairKey()) b=true;
      c++;
    }
  if(b) return c;
  else return 0;
}

int CNuc::IsJGroup(JGroup jGroup) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumJGroups())
    {
      if(jGroup.GetJ()==this->GetJGroup(c+1)->GetJ()&&
     jGroup.GetPi()==this->GetJGroup(c+1)->GetPi()) b=true;
     c++;
    }
  if(b) return c;
  else return 0;
}

int CNuc::GetPairNumFromKey(int key) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumPairs()) {
    if(key==this->GetPair(c+1)->GetPairKey()) b=true;
    c++;
  }
  if(b) return c;
  else return 0;
}

int CNuc::Fill(const Config &configure) {
  int PairNum,LevelNum,ChannelNum,JGroupNum;
  int maxLValue=0;
  std::ifstream in(configure.configfile.c_str());
  if(!in) return -1;
  std::string line = "";
  while(line!="<levels>"&&!in.eof()) getline(in,line);
  if(line!="<levels>") return -1;
  std::map<int,int> ecPairs;
  line="";
  while(!in.eof()&&line!="</levels>") {
    getline(in,line);
    bool empty=true;
    for(unsigned int i=0;i<line.size();++i)
      if(line[i]!=' '&&line[i]!='\t') {
    empty=false;
    break;
      }
    if(empty==true) continue;
    if(!in.eof()&&line!="</levels>") {
      std::istringstream stm;
      stm.str(line);
      NucLine Line(stm);
      if(stm.rdstate() & (std::stringstream::failbit | std::stringstream::badbit)) return -1;
      if(Line.l()>maxLValue&&Line.pType()==0) maxLValue=Line.l();
      if(Line.isActive()==1) {
    PPair NewPair(Line);
    PairNum=this->IsPair(NewPair);
    if(!PairNum) {
      this->AddPair(NewPair);
      PairNum=this->IsPair(NewPair);
    }
    if(Line.ecMultMask()!=0) {
      std::map<int,int>::iterator it = ecPairs.find(PairNum);
      if(it==ecPairs.end()) ecPairs[PairNum]=Line.ecMultMask();
    }
    JGroup NewJGroup(Line);
    JGroupNum=this->IsJGroup(NewJGroup);
    if(!JGroupNum) {
      this->AddJGroup(NewJGroup);
      JGroupNum=this->IsJGroup(NewJGroup);
    }
    AChannel NewChannel(Line,PairNum);
    ChannelNum=this->GetJGroup(JGroupNum)->IsChannel(NewChannel);
```

```
    if(!ChannelNum) {
      this->GetJGroup(JGroupNum)->AddChannel(NewChannel);
      ChannelNum=this->GetJGroup(JGroupNum)->IsChannel(NewChannel);
      if(this->GetJGroup(JGroupNum)->GetChannel(ChannelNum)->GetL()>maxLValue&&
         this->GetJGroup(JGroupNum)->GetChannel(ChannelNum)->GetRadType()=='P')
        maxLValue=this->GetJGroup(JGroupNum)->GetChannel(ChannelNum)->GetL();
    }
    ALevel NewLevel(Line);
    LevelNum=this->GetJGroup(JGroupNum)->IsLevel(NewLevel);
    if(!LevelNum) {
      this->GetJGroup(JGroupNum)->AddLevel(NewLevel);
      LevelNum=this->GetJGroup(JGroupNum)->IsLevel(NewLevel);
    }
    this->GetJGroup(JGroupNum)->GetLevel(LevelNum)->AddGamma(Line);
      }
    }
  }

  if(line!="</levels>") return -1;

  in.close();

  this->SetMaxLValue(maxLValue);
  if((configure.paramMask & Config::USE_EXTERNAL_CAPTURE) &&
     this->NumJGroups()>0 && this->NumPairs()>0)
    this->ParseExternalCapture(configure,ecPairs);

  return 0;
}

void CNuc::ParseExternalCapture(const Config& configure,std::map<int,int>& ecPairs) {
  for(std::map<int,int>::iterator ec=ecPairs.begin();ec!=ecPairs.end();ec++) {
    PPair *exitPair=this->GetPair(ec->first);
    if(exitPair->GetPType()!=10) {
      configure.outStream « "Final state is not a capture pair." « std::endl;
      continue;
    }
    //create new level in compound nucleus for EC state, if it doesn't exist
    double jValue=exitPair->GetJ(2);
    int parity=exitPair->GetPi(2);
    JGroup newJGroup(jValue,parity);
    int jGroupNum=this->IsJGroup(newJGroup);
    int levelNum=0;
    if(jGroupNum) {
      ALevel newLevel(exitPair->GetExE());
      levelNum=this->GetJGroup(jGroupNum)->IsLevel(newLevel);
      if(!levelNum) {
    this->GetJGroup(jGroupNum)->AddLevel(newLevel);
    levelNum=this->GetJGroup(jGroupNum)->IsLevel(newLevel);
    for(int ch=1;ch<=this->GetJGroup(jGroupNum)->NumChannels();ch++) {
      if(this->GetJGroup(jGroupNum)->GetChannel(ch)->GetRadType()=='P')
        this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddGamma(0.1);
      else this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddGamma(0.0);
    }
      }
      this->GetJGroup(jGroupNum)->GetLevel(levelNum)->SetECParams(ec->first,ec->second);
      for(int ch=1;ch<=this->GetJGroup(jGroupNum)->NumChannels();ch++) {
    PPair *theFinalPair=this->GetPair(this->GetJGroup(jGroupNum)->GetChannel(ch)->GetPairNum());
    double nfIntegralValue=0.;
    double ecConvert=0.;
    if(theFinalPair->GetPType()==0) {
      NFIntegral newNFIntegral(theFinalPair);
nfIntegralValue=newNFIntegral(this->GetJGroup(jGroupNum)->GetChannel(ch)->GetL(),exitPair->GetExE());
      WhitFunc newWhitFunc(theFinalPair);
      double whitConv=newWhitFunc(this->GetJGroup(jGroupNum)->GetChannel(ch)->GetL(),
                      theFinalPair->GetChRad(),
                      fabs(exitPair->GetExE()-theFinalPair->GetSepE()-theFinalPair->GetExE()));
ecConvert=sqrt(2.0*theFinalPair->GetRedMass()*theFinalPair->GetChRad()*uconv/pow(hbarc,2.0))/whitConv;
    }
    this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddNFIntegral(nfIntegralValue);
    this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddECConversionFactor(ecConvert);
      }
    } else {
      this->AddJGroup(newJGroup);
      jGroupNum=this->IsJGroup(newJGroup);
      ALevel newLevel(exitPair->GetExE());
      this->GetJGroup(jGroupNum)->AddLevel(newLevel);
      levelNum=this->GetJGroup(jGroupNum)->IsLevel(newLevel);
      this->GetJGroup(jGroupNum)->GetLevel(levelNum)->SetECParams(ec->first,ec->second);
      for(int ir=1;ir<=this->NumPairs();ir++) {
    if(this->GetPair(ir)->GetPType()==0) {
      double s1=this->GetPair(ir)->GetJ(1);
      double s2=this->GetPair(ir)->GetJ(2);
      int sPi=this->GetPair(ir)->GetPi(1)*this->GetPair(ir)->GetPi(2);
      for(double chS=fabs(s1-s2);chS<=s1+s2;chS+=1.) {
```

```
        for(int chL=0;chL<=this->GetMaxLValue();chL++) {
          int chPi=sPi*(int)pow(-1,chL);
          if(fabs(chS-chL)<=jValue&&jValue<=chS+chL&&chPi==parity) {
        AChannel newChannel(chL,chS,ir,'P');
        this->GetJGroup(jGroupNum)->AddChannel(newChannel);
        this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddGamma(0.1);
        NFIntegral newNFIntegral(this->GetPair(ir));
        double nfIntegralValue=newNFIntegral(chL,exitPair->GetExE());
        WhitFunc newWhitFunc(this->GetPair(ir));
        double whitConv=newWhitFunc(chL,this->GetPair(ir)->GetChRad(),
fabs(exitPair->GetExE()-this->GetPair(ir)->GetSepE()-this->GetPair(ir)->GetExE()));
        double
ecConvert=sqrt(2.0*this->GetPair(ir)->GetRedMass()*this->GetPair(ir)->GetChRad()*uconv/pow(hbarc,2.0))/whitConv;
        this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddNFIntegral(nfIntegralValue);
        this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddECConversionFactor(ecConvert);

          }
        }
      }
    }
      }
    }
  }
}

int CNuc::GetMaxLValue() const {
  return maxLValue_;
}

void CNuc::Initialize(const Config &configure) {
  //Calculate Boundary Conditions
  configure.outStream « "Calculating Boundary Conditions..." « std::endl;
  this->CalcBoundaryConditions(configure);
  if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_BOUNDARY_CONDITIONS)
    this->PrintBoundaryConditions(configure);

  //Transform Input Parameters
  if(configure.paramMask & Config::TRANSFORM_PARAMETERS) {
    configure.outStream « "Performing Input Parameter Transformation..." « std::endl;
    this->TransformIn(configure);
  }

  //Sort reaction pathways
  configure.outStream « "Sorting Reaction Pathways..." « std::endl;
  this->SortPathways(configure);
  if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_PATHWAYS)
    this->PrintPathways(configure);

  //Calculate Angular Distribution Coefficients
  configure.outStream « "Calculating Angular Distribution Coefficients..." « std::endl;
  this->CalcAngularDists(configure.maxLOrder);
  if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_ANGULAR_DISTS)
    this->PrintAngularDists(configure);

}

void CNuc::AddPair(PPair pPair) {
  pairs_.push_back(pPair);
}

void CNuc::AddJGroup(JGroup jGroup) {
  jgroups_.push_back(jGroup);
}

void CNuc::PrintNuc(const Config &configure) {
  std::streambuf *sbuffer;
  std::filebuf fbuffer;
  if(configure.fileCheckMask & Config::CHECK_COMPOUND_NUCLEUS) {
    std::string outfile=configure.checkdir+"compoundnucleus.chk";
    fbuffer.open(outfile.c_str(),std::ios::out);
    sbuffer = &fbuffer;
  } else if(configure.screenCheckMask & Config::CHECK_COMPOUND_NUCLEUS)
    sbuffer = configure.outStream.rdbuf();
  std::ostream out(sbuffer);
  if(((configure.fileCheckMask & Config::CHECK_COMPOUND_NUCLEUS)&&
      fbuffer.is_open())||
     (configure.screenCheckMask & Config::CHECK_COMPOUND_NUCLEUS)) {
    out « std::endl
    « "************************************" « std::endl
    « "*           Particle Pairs           *" « std::endl
    « "************************************" « std::endl;
    for(int i=1;i<=this->NumPairs();i++) {
      out « "Pair Number: " « i « "  Pair Key: " « this->GetPair(i)->GetPairKey() « std::endl;
      out « std::setw(30) « "Light Particle J: " « this->GetPair(i)->GetJ(1) « std::endl
      « std::setw(30) « "Light Particle Parity: " « this->GetPair(i)->GetPi(1) « std::endl
      « std::setw(30) « "Light Particle Z: " « this->GetPair(i)->GetZ(1) « std::endl
```

```
      « std::setw(30) « "Light Particle M: " « this->GetPair(i)->GetM(1) « std::endl
      « std::setw(30) « "Light Particle g: " « this->GetPair(i)->GetG(1) « std::endl
      « std::setw(30) « "Heavy Particle J: " « this->GetPair(i)->GetJ(2) « std::endl
      « std::setw(30) « "Heavy Particle Parity: " « this->GetPair(i)->GetPi(2) « std::endl
      « std::setw(30) « "Heavy Particle Z: " « this->GetPair(i)->GetZ(2) « std::endl
      « std::setw(30) « "Heavy Particle M: " « this->GetPair(i)->GetM(2) « std::endl
      « std::setw(30) « "Heavy Particle g: " « this->GetPair(i)->GetG(2) « std::endl
      « std::setw(30) « "Seperation Energy [MeV]: " « this->GetPair(i)->GetSepE() « std::endl
      « std::setw(30) « "Excitation Energy [MeV]: " « this->GetPair(i)->GetExE() « std::endl
      « std::setw(30) « "Channel Radius: " « this->GetPair(i)->GetChRad() « std::endl;
      if(this->GetPair(i)->GetPType()==0) out « std::setw(30) « "Pair Type: " « "Particle,Particle" «
std::endl;
      else if(this->GetPair(i)->GetPType()==10) out « std::setw(30) « "Pair Type: " « "Particle,Gamma"
« std::endl;
      else if(this->GetPair(i)->GetPType()==20) out « std::setw(30) « "Pair Type: " « "Beta Decay" «
std::endl;
      else out « std::setw(30) « "Pair Type: Unknown" « std::endl;
    }
    out « std::endl
    « "**************************************" « std::endl
    « "*                Levels              *" « std::endl
    « "**************************************" « std::endl
    « std::setw(11) « "J Group #"
    « std::setw(5)  «  "J"
    « std::setw(4)  « "Pi"
    « std::setw(9)  « "Level #"
    « std::setw(14) « "Energy [MeV]"
    « std::setw(11) « "Channel #"
    « std::setw(3)  « "l"
    « std::setw(5)  « "s"
    « std::setw(8)  « "Pair #"
    « std::setw(11)  « "Width"
    « std::setw(11) « "Rad. Type" « std::endl;

    for(int i=1;i<=this->NumJGroups();i++) {
      for(int ii=1;ii<=this->GetJGroup(i)->NumLevels();ii++) {
    for(int iii=1;iii<=this->GetJGroup(i)->NumChannels();iii++) {
      out « std::setw(11) « i
          « std::setw(5)  « this->GetJGroup(i)->GetJ()
          « std::setw(4)  « this->GetJGroup(i)->GetPi()
          « std::setw(9)  « ii
          « std::setw(14) « this->GetJGroup(i)->GetLevel(ii)->GetE()
          « std::setw(11) « iii
          « std::setw(3)  « this->GetJGroup(i)->GetChannel(iii)->GetL()
          « std::setw(5)  « this->GetJGroup(i)->GetChannel(iii)->GetS()
          « std::setw(8)  « this->GetJGroup(i)->GetChannel(iii)->GetPairNum()
          « std::setw(11)  « this->GetJGroup(i)->GetLevel(ii)->GetGamma(iii)
          « std::setw(11) « this->GetJGroup(i)->GetChannel(iii)->GetRadType() « std::endl;
    }
      }
      out « std::endl;
    }
  } else configure.outStream « "Could not write compound nucleus check file." « std::endl;
  out.flush();
  if(fbuffer.is_open()) fbuffer.close();
}

void CNuc::TransformIn(const Config& configure) {
  for(int j=1;j<=this->NumJGroups();j++) {
    JGroup *theJGroup=this->GetJGroup(j);
    if(theJGroup->IsInRMatrix()) {
      for(int la=1;la<=theJGroup->NumLevels();la++) {
    ALevel *theLevel=theJGroup->GetLevel(la);
    if(theLevel->IsInRMatrix()) {
      vector_r tempGammas;
      std::vector<bool> isNegative;
      vector_r penes;
      double denom=2.0;
      for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
        AChannel *theChannel=theJGroup->GetChannel(ch);
        double localEnergy=theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE()
          -this->GetPair(theChannel->GetPairNum())->GetSepE();
        double radius=this->GetPair(theChannel->GetPairNum())->GetChRad();
        if(theChannel->GetRadType()=='P') {
          if(localEnergy>0.0) {
        if(theLevel->GetGamma(ch)<0.0) isNegative.push_back(true);
        else isNegative.push_back(false);
        tempGammas.push_back(fabs(theLevel->GetGamma(ch))/1e6);
        CoulFunc theCoulombFunction(this->GetPair(theChannel->GetPairNum()),
                    !!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
        double tempPene=theCoulombFunction.Penetrability(theChannel->GetL(),
                             radius,
                             localEnergy);
        denom-=tempGammas[ch-1]/tempPene*
          theCoulombFunction.PEShift_dE(theChannel->GetL(),radius,localEnergy);
        penes.push_back(tempPene);
          } else {
```

```
        if(theLevel->GetGamma(ch)<0.0) isNegative.push_back(true);
        else isNegative.push_back(false);
        tempGammas.push_back(pow(theLevel->GetGamma(ch),2.0));
        ShftFunc theShiftFunction(this->GetPair(theChannel->GetPairNum()));
        WhitFunc newWhitFunc(this->GetPair(theChannel->GetPairNum()));
        double whitConv=newWhitFunc(theChannel->GetL(),radius,fabs(localEnergy));
        double tempPene=this->GetPair(theChannel->GetPairNum())->GetRedMass()*radius*uconv/
          pow(hbarc,2.0)/pow(whitConv,2.0);
        denom-=tempGammas[ch-1]/tempPene*
          theShiftFunction.EnergyDerivative(theChannel->GetL(),theLevel->GetE());
        penes.push_back(tempPene);
        }
        } else if(theChannel->GetRadType()=='E'||theChannel->GetRadType()=='M') {
          if(fabs(theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE())<1.e-3&&
         theJGroup->GetJ()==this->GetPair(theChannel->GetPairNum())->GetJ(2)&&
         theJGroup->GetPi()==this->GetPair(theChannel->GetPairNum())->GetPi(2)) {
        int tempSign;
        if(theLevel->GetGamma(ch)<0) tempSign=-1;
        else tempSign=1;
        double jValue=theJGroup->GetJ();
        if(int(2.*jValue)%2!=0) tempSign=-tempSign;
        double tempPene=1e-10;
        double tempGamma=theLevel->GetGamma(ch);
        if(theChannel->GetRadType()=='M'&&theChannel->GetL()==1) {
          tempPene=3.0*jValue/4.0/(jValue+1.)/nuclearMagneton/nuclearMagneton;
        } else if(theChannel->GetRadType()=='E'&&theChannel->GetL()==2) {
          tempPene=60.0*jValue*(2.*jValue-1.)/(jValue+1.)/(2.*jValue+3.);
          tempGamma=tempGamma*100*sqrt(fstruc*hbarc);
        }
        tempGammas.push_back(pow(tempGamma,2.0));
        penes.push_back(tempPene);
        if(tempSign<0) isNegative.push_back(true);
        else isNegative.push_back(false);
          } else {
        if(theLevel->GetGamma(ch)<0.0) isNegative.push_back(true);
        else isNegative.push_back(false);
        tempGammas.push_back(fabs(theLevel->GetGamma(ch))/1e6);
        double tempPene = (configure.paramMask & Config::USE_RMC_FORMALISM) ? 1.0 :
pow(fabs(localEnergy)/hbarc,2.0*theChannel->GetL()+1);
        if(tempPene<1e-16) tempPene=1e-16;
        penes.push_back(tempPene);
          }
        } else if(theChannel->GetRadType()=='F'||theChannel->GetRadType()=='G') {
          if(theLevel->GetGamma(ch)<0.0) isNegative.push_back(true);
          else isNegative.push_back(false);
          tempGammas.push_back(fabs(theLevel->GetGamma(ch)));
          penes.push_back(1.0);
        }
      }
      if(denom<0.) configure.outStream « "WARNING: Denominator less than zero in E="
                       « theLevel->GetE() « " MeV resonance transformation.  "
                       «  "Tranformation may not have been successful."
                       « std::endl;
      double nFSum=1.0;
      for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
        AChannel *theChannel=theJGroup->GetChannel(ch);
        if(theChannel->GetRadType()!='F'&&theChannel->GetRadType()!='G')
          tempGammas[ch-1]=sqrt(fabs(tempGammas[ch-1]/penes[ch-1]/denom));
        if(isNegative[ch-1]) tempGammas[ch-1]=-tempGammas[ch-1];
        theLevel->SetGamma(ch,tempGammas[ch-1]);
        if(ch<=theLevel->NumNFIntegrals()) nFSum+=2.0*
          this->GetPair(theChannel->GetPairNum())->GetChRad()*
          this->GetPair(theChannel->GetPairNum())->GetRedMass()*
          uconv/pow(hbarc,2.0)*pow(tempGammas[ch-1],2.0)*theLevel->GetNFIntegral(ch);
      }
      theLevel->SetSqrtNFFactor(1.0/sqrt(nFSum));
    }
      }
    }
  }
  for(int j=1;j<=this->NumJGroups();j++) {
    JGroup *theJGroup=this->GetJGroup(j);
    if(theJGroup->IsInRMatrix()) {
      std::vector<int> levelKeys;
      vector_r tempEnergies;
      matrix_r tempGammas;
      matrix_r shifts;
      for(int la=1;la<=theJGroup->NumLevels();la++) {
    ALevel *theLevel=theJGroup->GetLevel(la);
    if(theLevel->IsInRMatrix()) {
      levelKeys.push_back(la);
      tempEnergies.push_back(theLevel->GetE());
      vector_r tempChanVector;
      tempGammas.push_back(tempChanVector);
      shifts.push_back(tempChanVector);
      for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
        AChannel *theChannel=theJGroup->GetChannel(ch);
```

```
        double localEnergy=theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE()
          -this->GetPair(theChannel->GetPairNum())->GetSepE();
        double radius=this->GetPair(theChannel->GetPairNum())->GetChRad();
        if(theChannel->GetRadType()=='P') {
          if(localEnergy>0.0) {
        tempGammas[levelKeys.size()-1].push_back(theLevel->GetGamma(ch));
        CoulFunc theCoulombFunction(this->GetPair(theChannel->GetPairNum()),
                        !!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
        shifts[levelKeys.size()-1].push_back(theCoulombFunction.PEShift(theChannel->GetL(),
                                        radius,
                                        localEnergy));
          } else {
        tempGammas[levelKeys.size()-1].push_back(theLevel->GetGamma(ch));
        ShftFunc theShiftFunction(this->GetPair(theChannel->GetPairNum()));
        shifts[levelKeys.size()-1].push_back(theShiftFunction(theChannel->GetL(),theLevel->GetE()));
        }
        } else {
          tempGammas[levelKeys.size()-1].push_back(theLevel->GetGamma(ch));
          if((theChannel->GetRadType()=='E'||theChannel->GetRadType()=='M') &&
         (configure.paramMask & Config::USE_EXTERNAL_CAPTURE) &&
         !(fabs(theLevel->GetGamma(ch))<1.0e-8 &&
           (configure.paramMask & Config::IGNORE_ZERO_WIDTHS))) {
        complex externalWidth =
          CalcExternalWidth(theJGroup,theLevel,theChannel,true,configure);
        if(pow(tempGammas[levelKeys.size()-1][ch-1],2.0)>=pow(imag(externalWidth),2.0)) {
          if(tempGammas[levelKeys.size()-1][ch-1]<0.0)
            tempGammas[levelKeys.size()-1][ch-1]=-sqrt(pow(tempGammas[levelKeys.size()-1][ch-1],2.0)-
                                   pow(imag(externalWidth),2.0))-real(externalWidth);
          else
tempGammas[levelKeys.size()-1][ch-1]=sqrt(pow(tempGammas[levelKeys.size()-1][ch-1],2.0)-
                                 pow(imag(externalWidth),2.0))-real(externalWidth);
        } else {
          configure.outStream « "**WARNING: Imaginary portion of external width \n\tfor j=" « j « "
la="
                « la « " ch=" « ch « " is greater than total width." « std::endl;
          tempGammas[levelKeys.size()-1][ch-1]=-real(externalWidth);
        }
          }
          shifts[levelKeys.size()-1].push_back(shifts[levelKeys.size()-1][0]);
        }
      }
    }
      }
      if(!(configure.paramMask & Config::USE_BRUNE_FORMALISM)) {
    matrix_r nMatrix;
    matrix_r mMatrix;
    for(int mu=0;mu<tempEnergies.size();mu++) {
      vector_r tempLevelVector;
      nMatrix.push_back(tempLevelVector);
      mMatrix.push_back(tempLevelVector);
      for(int la=0;la<tempEnergies.size();la++) {
        if(la==mu) {
          mMatrix[mu].push_back(1.0);
          double sum=tempEnergies[la];
          for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
        if(theJGroup->GetChannel(ch)->GetRadType()=='P')
          sum+=(shifts[la][ch-1]-theJGroup->GetChannel(ch)->GetBoundaryCondition())*
            pow(tempGammas[la][ch-1],2.0);
          }
          nMatrix[mu].push_back(sum);
        } else {
          double mSum=0.0;
          double nSum=0.0;
          for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
        if(theJGroup->GetChannel(ch)->GetRadType()=='P') {
          mSum+=(shifts[mu][ch-1]-shifts[la][ch-1])/(tempEnergies[mu]-tempEnergies[la])*
            tempGammas[la][ch-1]*tempGammas[mu][ch-1];
          nSum+=((tempEnergies[mu]*shifts[la][ch-1]-tempEnergies[la]*shifts[mu][ch-1])/
             (tempEnergies[mu]-tempEnergies[la])-theJGroup->GetChannel(ch)->GetBoundaryCondition())
            *tempGammas[la][ch-1]*tempGammas[mu][ch-1];
        }
          }
          mMatrix[mu].push_back(-mSum);
          nMatrix[mu].push_back(nSum);
        }
      }
    }
    //solve eigenvalue problem
    EigenFunc eigenFunc(nMatrix,mMatrix);
    for(int la=0;la<tempEnergies.size();la++) {
      theJGroup->GetLevel(levelKeys[la])->SetE(eigenFunc.eigenvalues()[la]);
      for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
        double sum=0.0;
        for(int mu=0;mu<tempEnergies.size();mu++) {
              sum+=eigenFunc.eigenvectors()[mu][la]*tempGammas[mu][ch-1];
        }
        theJGroup->GetLevel(levelKeys[la])->SetGamma(ch,sum);
```

```
      }
    }
      } else {
    for(int la=0;la<tempEnergies.size();la++)
      for(int ch=1;ch<=theJGroup->NumChannels();ch++)
        theJGroup->GetLevel(levelKeys[la])->SetGamma(ch,tempGammas[la][ch-1]);
      }
    }
  }
}

void CNuc::SortPathways(const Config& configure) {
  int DecayNum, KGroupNum, MGroupNum;
  for(int aa=1;aa<=this->NumPairs();aa++) {
    if(!this->GetPair(aa)->IsEntrance()) continue;
    for(int ir=1;ir<=this->NumPairs();ir++) {
      if(this->GetPair(ir)->GetPType()==20) continue;
      if(this->GetPair(aa)->GetPType()==20) {
    for(int l = 0; l < 2; l++) {
      for(int j=1;j<=this->NumJGroups();j++) {
        if(!this->GetJGroup(j)->IsInRMatrix()) continue;
        for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
          if(this->GetJGroup(j)->GetChannel(ch)->GetPairNum()!=aa) continue;
          for(int chp=1;chp<=this->GetJGroup(j)->NumChannels();chp++) {
        if(this->GetJGroup(j)->GetChannel(chp)->GetPairNum()!=ir||
           this->GetJGroup(j)->GetChannel(ch)->GetL()!=l) continue;
        Decay NewDecay(ir);
        DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
        if(!DecayNum) {
          this->GetPair(aa)->AddDecay(NewDecay);
          DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
        }
        KGroup NewKGroup(l,0);
        KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
        if(!KGroupNum) {
          this->GetPair(aa)->GetDecay(DecayNum)->AddKGroup(NewKGroup);
          KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
        }
        MGroup NewMGroup(j,ch,chp);
        MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
        if(!MGroupNum) {
          this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->AddMGroup(NewMGroup);
          MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
        }
          }
        }
      }
    }
      } else if(this->GetPair(ir)->GetPType()==0) {
    for(double s=fabs(this->GetPair(aa)->GetJ(1)-this->GetPair(aa)->GetJ(2));
        s<=(this->GetPair(aa)->GetJ(1)+this->GetPair(aa)->GetJ(2));s+=1.) {
      for(double sp=fabs(this->GetPair(ir)->GetJ(1)-this->GetPair(ir)->GetJ(2));
          sp<=(this->GetPair(ir)->GetJ(1)+this->GetPair(ir)->GetJ(2));sp+=1.) {
        for(int j=1;j<=this->NumJGroups();j++) {
          if(!this->GetJGroup(j)->IsInRMatrix()) continue;
          for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
        if(this->GetJGroup(j)->GetChannel(ch)->GetPairNum()!=aa) continue;
        for(int chp=1;chp<=this->GetJGroup(j)->NumChannels();chp++) {
          if(this->GetJGroup(j)->GetChannel(chp)->GetPairNum()!=ir||
             this->GetJGroup(j)->GetChannel(ch)->GetS()!=s||
             this->GetJGroup(j)->GetChannel(chp)->GetS()!=sp) continue;
          Decay NewDecay(ir);
          DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
          if(!DecayNum) {
            this->GetPair(aa)->AddDecay(NewDecay);
            DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
          }
          KGroup NewKGroup(s,sp);
          KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
          if(!KGroupNum) {
            this->GetPair(aa)->GetDecay(DecayNum)->AddKGroup(NewKGroup);
            KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
          }
          MGroup NewMGroup(j,ch,chp);
          MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
          if(!MGroupNum) {
            this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->AddMGroup(NewMGroup);

MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
          }
          double statspinfactor=(2.*this->GetJGroup(j)->GetJ()+1.)*
            this->GetPair(this->GetJGroup(j)->GetChannel(chp)->GetPairNum())->GetI1I2Factor();
          this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->
            GetMGroup(MGroupNum)->SetStatSpinFactor(statspinfactor);
        }
          }
        }
```

```
      }
    }
  } else if(this->GetPair(ir)->GetPType()==10 && !(configure.paramMask & Config::USE_RMC_FORMALISM)) {
    for(double s=fabs(this->GetPair(aa)->GetJ(1)-this->GetPair(aa)->GetJ(2));
        s<=(this->GetPair(aa)->GetJ(1)+this->GetPair(aa)->GetJ(2));s+=1.) {
      for(int j=1;j<=this->NumJGroups();j++) {
        if(!this->GetJGroup(j)->IsInRMatrix()) continue;
        for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
          if(this->GetJGroup(j)->GetChannel(ch)->GetPairNum()!=aa) continue;
          for(int chp=1;chp<=this->GetJGroup(j)->NumChannels();chp++) {
        if(this->GetJGroup(j)->GetChannel(chp)->GetPairNum()!=ir||
           this->GetJGroup(j)->GetChannel(ch)->GetS()!=s) continue;
        Decay NewDecay(ir);
        DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
        if(!DecayNum) {
          this->GetPair(aa)->AddDecay(NewDecay);
          DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
        }
        KGroup NewKGroup(s,0);
        KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
        if(!KGroupNum) {
          this->GetPair(aa)->GetDecay(DecayNum)->AddKGroup(NewKGroup);
          KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
        }
        MGroup NewMGroup(j,ch,chp);
        MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
        if(!MGroupNum) {
          this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->AddMGroup(NewMGroup);
          MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
        }
        double statspinfactor=(2.*this->GetJGroup(j)->GetJ()+1.)*
          this->GetPair(this->GetJGroup(j)->GetChannel(chp)->GetPairNum())->GetI1I2Factor();
        this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->
          GetMGroup(MGroupNum)->SetStatSpinFactor(statspinfactor);
          }
        }
      }
    }
      }
    }
  }
  for(int aa=1;aa<=this->NumPairs();aa++) { //loop over all pairs
    PPair *entrancePair=this->GetPair(aa);
    if(entrancePair->GetPType()==20 || !entrancePair->IsEntrance()) continue;
    for(int j=1;j<=this->NumJGroups();j++) {
      JGroup *theFinalJGroup=this->GetJGroup(j);
      for(int la=1;la<=theFinalJGroup->NumLevels();la++) {
    ALevel *theFinalLevel=theFinalJGroup->GetLevel(la);
    if(!theFinalLevel->IsECLevel()) continue;
    int decayNum=entrancePair->IsDecay(theFinalLevel->GetECPairNum()); //store RESONANCE decay number to final state
    if(!decayNum) continue; //if this is a resonance decay...
    for(int k=1;k<=entrancePair->GetDecay(decayNum)->NumKGroups();k++) { //loop over all kgroups for decays to final state
      KGroup *theKGroup=entrancePair->GetDecay(decayNum)->GetKGroup(k);
      for(int chp=1;chp<=theFinalJGroup->NumChannels();chp++) { //loop over all final configurations in the capture state
        AChannel *finalChannel=theFinalJGroup->GetChannel(chp);
        if(this->GetPair(finalChannel->GetPairNum())->GetPType()!=0) continue;  //ensure the configuration is a particle pair
        int chDecayNum=entrancePair->IsDecay(finalChannel->GetPairNum());
        if(!chDecayNum) continue; //if it is actually a resonance decay...
        for(int kp=1;kp<=entrancePair->GetDecay(chDecayNum)->NumKGroups();kp++) {
          if(entrancePair->GetDecay(chDecayNum)->GetKGroup(kp)->GetS()!=theKGroup->GetS()) continue;
          for(int mp=1;mp<=entrancePair->GetDecay(chDecayNum)->GetKGroup(kp)->NumMGroups();mp++) {
        MGroup *chMGroup=entrancePair->GetDecay(chDecayNum)->GetKGroup(kp)->GetMGroup(mp);
        AChannel *chChannel=this->GetJGroup(chMGroup->GetJNum())->GetChannel(chMGroup->GetChNum());
        AChannel *chChannelp=this->GetJGroup(chMGroup->GetJNum())->GetChannel(chMGroup->GetChpNum());
        for(int multL=1;multL<=maxECMult;multL++) { //loop over all allowed gamma parities
          char radType;
if(this->GetJGroup(chMGroup->GetJNum())->GetPi()*theFinalJGroup->GetPi()==(int)pow(-1,multL)) radType='E';
          else radType='M'; //calculate radiation type
          if(!((radType=='E' && multL==1) && (theFinalLevel->GetECMultMask()&isE1)) &&
             !((radType=='M' && multL==1) && (theFinalLevel->GetECMultMask()&isM1)) &&
             !((radType=='E' && multL==2) && (theFinalLevel->GetECMultMask()&isE2)) ) continue; //allow only m1,e1,e2
          if(fabs(this->GetJGroup(chMGroup->GetJNum())->GetJ()-multL)>theFinalJGroup->GetJ()||
             theFinalJGroup->GetJ()>this->GetJGroup(chMGroup->GetJNum())->GetJ()+multL) continue;
          if(!(abs(chChannelp->GetL()-multL)<=finalChannel->GetL()&&
               finalChannel->GetL()<=chChannelp->GetL()+multL&&
               fabs(chChannelp->GetS()-finalChannel->GetL())<=theFinalJGroup->GetJ()&&
               theFinalJGroup->GetJ()<=chChannelp->GetS()+finalChannel->GetL()&&
               chChannelp->GetS()==finalChannel->GetS())&&
             !(fabs(chChannelp->GetS()-multL)<=finalChannel->GetS()&&
```

```
                finalChannel->GetS()<=chChannelp->GetS()+multL&&
                fabs(chChannelp->GetL()-finalChannel->GetS())<=theFinalJGroup->GetJ()&&
                theFinalJGroup->GetJ()<=chChannelp->GetL()+finalChannel->GetS()&&
                chChannelp->GetL()==finalChannel->GetL()&&
                radType=='M')) continue; //ensure entrance channel for dc can couple to final state

          if(chChannel==chChannelp) {
            ECMGroup newECMGroup(radType,multL,chChannel->GetL(),
                     this->GetJGroup(chMGroup->GetJNum())->GetJ(),chp,j,la);
            theKGroup->AddECMGroup(newECMGroup);
          }
          int internalChannel=0;
          for(int intCh=1;intCh<=this->GetJGroup(chMGroup->GetJNum())->NumChannels();intCh++) {
            if(this->GetJGroup(chMGroup->GetJNum())->GetChannel(intCh)->GetRadType()==radType &&
               this->GetJGroup(chMGroup->GetJNum())->GetChannel(intCh)->GetL()==multL &&

this->GetJGroup(chMGroup->GetJNum())->GetChannel(intCh)->GetPairNum()==theFinalLevel->GetECPairNum())
{
              internalChannel=intCh;
              break;
            }
          }
          ECMGroup
newECMGroup(radType,multL,chChannel->GetL(),this->GetJGroup(chMGroup->GetJNum())->GetJ(),
                      chp,j,la,chDecayNum,kp,mp,internalChannel);
          theKGroup->AddECMGroup(newECMGroup);
        }
          }
        }
      }
    }
      }
    }
  }
}

void CNuc::PrintPathways(const Config &configure) {
  std::streambuf *sbuffer;
  std::filebuf fbuffer;
  if(configure.fileCheckMask & Config::CHECK_PATHWAYS) {
    std::string outfile=configure.checkdir+"pathways.chk";
    fbuffer.open(outfile.c_str(),std::ios::out);
    sbuffer = &fbuffer;
  } else if(configure.screenCheckMask & Config::CHECK_PATHWAYS) sbuffer = configure.outStream.rdbuf();
  std::ostream out(sbuffer);
  if(((configure.fileCheckMask & Config::CHECK_PATHWAYS)&&fbuffer.is_open())
     ||(configure.screenCheckMask & Config::CHECK_PATHWAYS)) {
    out « std::endl
    « "************************************" « std::endl
    « "*     Internal Reaction Pathways     *" « std::endl
    « "************************************" « std::endl
    « std::setw(17) « "Entrance Pair #"
    « std::setw(9)  « "Decay #"
    « std::setw(14) « "Decay Pair #"
    « std::setw(12) « "K Group #"
    « std::setw(16) « "Entrance Ch. s"
    « std::setw(13) « "Decay Ch. s"
    « std::setw(11) « "M Group #"
    « std::setw(11) « "J Group #"
    « std::setw(16) « "Entrance Ch. #"
    « std::setw(12) « "Exit Ch. #" « std::endl;
    for(int i=1;i<=this->NumPairs();i++) {
      PPair *thePair=this->GetPair(i);
      for(int ii=1;ii<=this->GetPair(i)->NumDecays();ii++){
    for(int iii=1;iii<=this->GetPair(i)->GetDecay(ii)->NumKGroups();iii++) {
      for(int iiii=1;iiii<=this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->NumMGroups();iiii++) {

        out « std::setw(17) « i
        « std::setw(9)  « ii
        « std::setw(14) « this->GetPair(i)->GetDecay(ii)->GetPairNum()
        « std::setw(12) « iii
        « std::setw(16) « this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetS()
        « std::setw(13) « this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetSp()
        « std::setw(11) « iiii
        « std::setw(11) « this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetMGroup(iiii)->GetJNum()
        « std::setw(16) « this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetMGroup(iiii)->GetChNum()
        « std::setw(12) « this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetMGroup(iiii)->GetChpNum()
« std::endl;
      }
    }
    out « std::endl;
      }
    }
    out « std::endl
    « "************************************" « std::endl
    « "*     External Reaction Pathways     *" « std::endl
    « "************************************" « std::endl
```

```
    « std::setw(17) « "Entrance Pair #"
    « std::setw(9)  « "Decay #"
    « std::setw(14) « "Decay Pair #"
    « std::setw(12) « "K Group #"
    « std::setw(16) « "Entrance Ch. s"
    « std::setw(13) « "Decay Ch. s"
    « std::setw(11) « "M Group #"
    « std::setw(11) « "Mult."
    « std::setw(11) « "J_i Value"
    « std::setw(11) « "J_f Value"
    « std::setw(11) « "l_i Value"
    « std::setw(11) « "l_f Value"
    « std::setw(13) « "Type"
    « std::setw(13) « "Ch. Decay #"
    « std::setw(11) « "Ch. K #"
    « std::setw(11) « "Ch. M #"
    « std::setw(11) « "Int. Ch #"
    « std::endl;
    for(int i=1;i<=this->NumPairs();i++) {
      PPair *thePair=this->GetPair(i);
      for(int ii=1;ii<=this->GetPair(i)->NumDecays();ii++){
    for(int iii=1;iii<=this->GetPair(i)->GetDecay(ii)->NumKGroups();iii++) {
      for(int iiii=1;iiii<=this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->NumECMGroups();iiii++) {
        ECMGroup *theECMGroup=this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetECMGroup(iiii);
        JGroup *theECJGroup=this->GetJGroup(theECMGroup->GetJGroupNum());
        out « std::setw(17) « i
        « std::setw(9)  « ii
        « std::setw(14) « this->GetPair(i)->GetDecay(ii)->GetPairNum()
        « std::setw(12) « iii
        « std::setw(16) « this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetS()
        « std::setw(13) « theECJGroup->GetChannel(theECMGroup->GetFinalChannel())->GetS()
        « std::setw(11) « iiii
        « std::setw(10) « theECMGroup->GetRadType() « theECMGroup->GetMult()
        « std::setw(11) « theECMGroup->GetJ()
        « std::setw(11) « theECJGroup->GetJ()
        « std::setw(11) « theECMGroup->GetL()
        « std::setw(11) « theECJGroup->GetChannel(theECMGroup->GetFinalChannel())->GetL();
        if(theECMGroup->IsChannelCapture()) out « std::setw(13) « "Channel"
                                  « std::setw(13) « theECMGroup->GetChanCapDecay()
                                  « std::setw(11) « theECMGroup->GetChanCapKGroup()
                                  « std::setw(11) « theECMGroup->GetChanCapMGroup()
                                  « std::setw(11) « theECMGroup->GetIntChannelNum() « std::endl;
        else out « std::setw(15) « "Hard Sphere" « std::endl;
      }
    }
    out « std::endl;
      }
    }
  } else configure.outStream « "Could not write pathways check file." « std::endl;
  out.flush();
  if(fbuffer.is_open()) fbuffer.close();
}

void CNuc::CalcBoundaryConditions(const Config& configure){
  for(int j=1;j<=this->NumJGroups();j++) {
    if(this->GetJGroup(j)->IsInRMatrix()) {
      JGroup *theJGroup=this->GetJGroup(j);
      ALevel *firstLevel=theJGroup->GetLevel(1);
      if(firstLevel->IsInRMatrix()) {
    for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
      AChannel *theChannel=theJGroup->GetChannel(ch);
      PPair *thePair=this->GetPair(theChannel->GetPairNum());
      if(thePair->GetPType()==0) {
        int lValue=theChannel->GetL();
        double levelEnergy=firstLevel->GetE();
        double resonanceEnergy=levelEnergy-(thePair->GetSepE()+thePair->GetExE());
        if(resonanceEnergy<0.0) {
          ShftFunc theShiftFunction(thePair);
          theChannel->SetBoundaryCondition(theShiftFunction(lValue,levelEnergy));
        }
        else {
          CoulFunc theCoulombFunction(thePair,
                        !!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
          double radius=thePair->GetChRad();
          double boundary=theCoulombFunction.PEShift(lValue,radius,resonanceEnergy);
          theChannel->SetBoundaryCondition(boundary);
        }
      }
      else {
        double boundary=theJGroup->GetChannel(1)->GetBoundaryCondition();
        theChannel->SetBoundaryCondition(boundary);
      }
    }
      }
    }
  }
}
```

```
void CNuc::PrintBoundaryConditions(const Config &configure) {
  std::streambuf *sbuffer;
  std::filebuf fbuffer;
  if(configure.fileCheckMask & Config::CHECK_BOUNDARY_CONDITIONS) {
    std::string outfile=configure.checkdir+"boundaryconditions.chk";
    fbuffer.open(outfile.c_str(),std::ios::out);
    sbuffer = &fbuffer;
  } else if(configure.screenCheckMask & Config::CHECK_BOUNDARY_CONDITIONS) sbuffer =
configure.outStream.rdbuf();
  std::ostream out(sbuffer);
  if(((configure.fileCheckMask & Config::CHECK_BOUNDARY_CONDITIONS)&&fbuffer.is_open())||
     (configure.screenCheckMask & Config::CHECK_BOUNDARY_CONDITIONS)) {
    out « std::endl
        « "***********************************" « std::endl
        « "*         Boundary Conditions       *" « std::endl
        « "***********************************" « std::endl;
    out « std::setw(10) « "J Group #"
        « std::setw(10) « "Channel #"
        « std::setw(20) « "Boundary Condition"
        « std::endl;
    for(int j=1;j<=this->NumJGroups();j++) {
      if(this->GetJGroup(j)->IsInRMatrix()) {
    JGroup *theJGroup=this->GetJGroup(j);
    for (int ch=1;ch<=theJGroup->NumChannels();ch++) {
      AChannel *theChannel=theJGroup->GetChannel(ch);
      out « std::setw(10) « j
          « std::setw(10) « ch
          « std::setw(20) « theChannel->GetBoundaryCondition()
          « std::endl;
    }
      }
    }
  } else configure.outStream « "Could not write boundary conditions check file." « std::endl;
  out.flush();
  if(fbuffer.is_open()) fbuffer.close();
}

void CNuc::CalcAngularDists(int maxL) {
  for(int aa=1;aa<=this->NumPairs();aa++) {
    PPair *entrancePair=this->GetPair(aa);
    if(entrancePair->GetPType()==20) continue;
    for(int ir=1;ir<=this->GetPair(aa)->NumDecays();ir++) {
      Decay *theDecay=this->GetPair(aa)->GetDecay(ir);
      for(int k=1;k<=theDecay->NumKGroups();k++) {
        for(int lOrder=0;lOrder<=maxL;lOrder++) {
          for(int
m1=1;m1<=theDecay->GetKGroup(k)->NumMGroups()+theDecay->GetKGroup(k)->NumECMGroups();m1++){
            for(int
m2=1;m2<=theDecay->GetKGroup(k)->NumMGroups()+theDecay->GetKGroup(k)->NumECMGroups();m2++) {
          std::string interferenceType;
          double j1,j2,l1,l1p,l2,l2p;
          int w1p,w2p,path1,path2;
          if(m1>theDecay->GetKGroup(k)->NumMGroups()) {
        int m1_ec=m1-theDecay->GetKGroup(k)->NumMGroups();
        ECMGroup *theECMGroup1=theDecay->GetKGroup(k)->GetECMGroup(m1_ec);
        j1=theECMGroup1->GetJ();
        l1=(double) theECMGroup1->GetL();
        l1p=(double) theECMGroup1->GetMult();
        if(theECMGroup1->GetRadType()=='M') w1p=0;
        else w1p=1;
        interferenceType='E';
        path1=m1_ec;
          } else {
        JGroup *jgroup1=this->GetJGroup(theDecay->GetKGroup(k)->GetMGroup(m1)->GetJNum());
        AChannel *channel1=jgroup1->GetChannel(theDecay->GetKGroup(k)->GetMGroup(m1)->GetChNum());
        AChannel *channel1p=jgroup1->GetChannel(theDecay->GetKGroup(k)->GetMGroup(m1)->GetChpNum());
        j1=jgroup1->GetJ();
        l1=(double) channel1->GetL();
        l1p=(double) channel1p->GetL();
        if(channel1p->GetRadType()=='M'||channel1p->GetRadType()=='P') w1p=0;
        else w1p=1;
        interferenceType='R';
        path1=m1;
          }
          if(m2>theDecay->GetKGroup(k)->NumMGroups()) {
        int m2_ec=m2-theDecay->GetKGroup(k)->NumMGroups();
        ECMGroup *theECMGroup2=theDecay->GetKGroup(k)->GetECMGroup(m2_ec);
        j2=theECMGroup2->GetJ();
        l2=(double) theECMGroup2->GetL();
        l2p=(double) theECMGroup2->GetMult();
        if(theECMGroup2->GetRadType()=='M') w2p=0;
        else w2p=1;
        interferenceType+='E';
        path2=m2_ec;
          } else {
        JGroup *jgroup2=this->GetJGroup(theDecay->GetKGroup(k)->GetMGroup(m2)->GetJNum());
```

```
        AChannel *channel2=jgroup2->GetChannel(theDecay->GetKGroup(k)->GetMGroup(m2)->GetChNum());
        AChannel *channel2p=jgroup2->GetChannel(theDecay->GetKGroup(k)->GetMGroup(m2)->GetChpNum());
        j2=jgroup2->GetJ();
        l2=(double) channel2->GetL();
        l2p=(double) channel2p->GetL();
        if(channel2p->GetRadType()=='M'||channel2p->GetRadType()=='P') w2p=0;
        else w2p=1;
        interferenceType+='R';
        path2=m2;
          }
          double s=theDecay->GetKGroup(k)->GetS();
          double sp=theDecay->GetKGroup(k)->GetSp();
          if((int)(l1+l2+lOrder)%2==0&&(int)(l1p+l2p+w1p+w2p+lOrder)%2==0) {
        double z1z2=0.0;
        double z1=sqrt(2.*l1+1.)*sqrt(2.*l2+1.)*sqrt(2.*j1+1.)*sqrt(2.*j2+1.)
          *AngCoeff::ClebGord(l1,l2,lOrder,0.,0.,0.)*AngCoeff::Racah(l1,j1,l2,j2,s,lOrder);
        if(this->GetPair(theDecay->GetPairNum())->GetPType()==0) {
          double z2=sqrt(2.*l1p+1.)*sqrt(2.*l2p+1.)*sqrt(2.*j1+1.)*sqrt(2.*j2+1.)
            *AngCoeff::ClebGord(l1p,l2p,lOrder,0.,0.,0.)*AngCoeff::Racah(l1p,j1,l2p,j2,sp,lOrder);
          z1z2=pow(-1.0,sp-s)/4.*z1*z2;
        } else if(this->GetPair(theDecay->GetPairNum())->GetPType()==10) {
          double jf=this->GetPair(theDecay->GetPairNum())->GetJ(2);
          double z2=sqrt(2.*l1p+1.)*sqrt(2.*l2p+1.)*sqrt(2.*j1+1.)*sqrt(2.*j2+1.)
            *AngCoeff::ClebGord(l1p,l2p,lOrder,1.,-1.,0)*AngCoeff::Racah(l1p,j1,l2p,j2,jf,lOrder);
          z1z2=pow(-1.,1.+s-jf)/4.*z1*z2;
        }
        if(fabs(z1z2)>1e-10) {
          KLGroup NewKLGroup(k,lOrder);
          int KLGroupNum=theDecay->IsKLGroup(NewKLGroup);
          if(!KLGroupNum) {
            theDecay->AddKLGroup(NewKLGroup);
            KLGroupNum=theDecay->IsKLGroup(NewKLGroup);
          }
          Interference NewInterference(path1,path2,z1z2,interferenceType);
          int InterNum=theDecay->GetKLGroup(KLGroupNum)->IsInterference(NewInterference);
          if(!InterNum) {
            theDecay->GetKLGroup(KLGroupNum)->AddInterference(NewInterference);
            InterNum=theDecay->GetKLGroup(KLGroupNum)->IsInterference(NewInterference);
          }
        }
          }
            }
          }
    }
      }
    }
  }
}

void CNuc::PrintAngularDists(const Config &configure) {
  std::streambuf *sbuffer;
  std::filebuf fbuffer;
  if(configure.fileCheckMask & Config::CHECK_ANGULAR_DISTS) {
    std::string outfile=configure.checkdir+"angulardistributions.chk";
    fbuffer.open(outfile.c_str(),std::ios::out);
    sbuffer = &fbuffer;
  } else if(configure.screenCheckMask & Config::CHECK_ANGULAR_DISTS) sbuffer =
configure.outStream.rdbuf();
  std::ostream out(sbuffer);
  if(((configure.fileCheckMask & Config::CHECK_ANGULAR_DISTS)&&fbuffer.is_open())||
     (configure.screenCheckMask & Config::CHECK_ANGULAR_DISTS)) {
    out « std::endl
        « "************************************" « std::endl
        « "*       Angular Distributions       *" « std::endl
        « "************************************" « std::endl;
    out « std::setw(10) « "ir"
    « std::setw(10) « "k"
    « std::setw(10) « "L"
    « std::setw(10) « "m1"
    « std::setw(10) « "m2"
    « std::setw(10) « "z1z2"
    « std::setw(10) « "type"
    « std::endl;
    for(int aa=1;aa<=this->NumPairs();aa++) {
      for(int ir=1;ir<=this->GetPair(aa)->NumDecays();ir++) {
    Decay *theDecay=this->GetPair(aa)->GetDecay(ir);
    for(int kl=1;kl<=theDecay->NumKLGroups();kl++) {
      KLGroup *theKLGroup=theDecay->GetKLGroup(kl);
      for(int i=1;i<=theKLGroup->NumInterferences();i++){
        Interference *theInter=theKLGroup->GetInterference(i);
        out « std::setw(10) « theDecay->GetPairNum()
        « std::setw(10) « theKLGroup->GetK()
        « std::setw(10) « theKLGroup->GetLOrder()
        « std::setw(10) « theInter->GetM1()
        « std::setw(10) « theInter->GetM2()
        « std::setw(10) « theInter->GetZ1Z2()
        « std::setw(10) « theInter->GetInterferenceType()
```

```
            « std::endl;
          }
          out « std::endl;
        }
          }
        }
    }  else configure.outStream « "Could not write angular distributions check file." « std::endl;
    out.flush();
    if(fbuffer.is_open()) fbuffer.close();
}

void CNuc::FillMnParams(ROOT::Minuit2::MnUserParameters &p) {
  char varname[50];
  for(int j=1;j<=this->NumJGroups();j++) {
    for(int la=1;la<=this->GetJGroup(j)->NumLevels();la++) {
      ALevel *level=this->GetJGroup(j)->GetLevel(la);
      sprintf(varname,"j=%d_la=%d_energy",j,la);
      p.Add(varname,level->GetE(),0.1*level->GetE());
      bool isUnbound=false;
      for(int ir=1;ir<=this->NumPairs();ir++) {
    PPair *pair=this->GetPair(ir);
    if(pair->GetPType()==0 &&
       level->GetE()>(pair->GetSepE()+pair->GetExE())) isUnbound=true;
      }
      if(!isUnbound) p.Fix(varname);
      if(level->EnergyFixed()&&!p.Parameter(p.Index(varname)).IsFixed()) p.Fix(varname);
      for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
    sprintf(varname,"j=%d_la=%d_ch=%d_rwa",j,la,ch);
    p.Add(varname,level->GetGamma(ch),0.1*level->GetGamma(ch));
    if(level->GetGamma(ch)==0.0) p.Fix(varname);
    if(level->ChannelFixed(ch)&&!p.Parameter(p.Index(varname)).IsFixed()) p.Fix(varname);
      }
    }
  }
}

void CNuc::FillCompoundFromParams(const vector_r &p) {
  int i=0;
  for(int j=1;j<=this->NumJGroups();j++) {
    for(int la=1;la<=this->GetJGroup(j)->NumLevels();la++) {
      ALevel *level=this->GetJGroup(j)->GetLevel(la);
      level->SetFitE(p[i]);i++;
      double nFSum=1.0;
      for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
    level->SetFitGamma(ch,p[i]);
    if(ch<=level->NumNFIntegrals()) nFSum+=2.0*
      this->GetPair(this->GetJGroup(j)->GetChannel(ch)->GetPairNum())->GetChRad()*
      this->GetPair(this->GetJGroup(j)->GetChannel(ch)->GetPairNum())->GetRedMass()*
      uconv/pow(hbarc,2.0)*pow(p[i],2.0)*level->GetNFIntegral(ch);
    i++;
      }
      level->SetSqrtNFFactor(1.0/sqrt(nFSum));
    }
  }
}

void CNuc::TransformOut(const Config& configure) {
  if(!(configure.paramMask & Config::USE_BRUNE_FORMALISM)) {
    int maxIterations=1000;
    double energyTolerance=1e-6;
    for(int j=1;j<=this->NumJGroups();j++) {
      for(int la=1;la<=this->GetJGroup(j)->NumLevels();la++) {
    ALevel *theLevel=this->GetJGroup(j)->GetLevel(la);
    if(theLevel->IsInRMatrix()) {
      int iteration=1;
      int thisLevel=0;
      bool done=false;
      vector_r tempE;
      vector_r tempBoundary;
      matrix_r tempGamma;
      for(int lap=1;lap<=this->GetJGroup(j)->NumLevels();lap++) {
        if(this->GetJGroup(j)->GetLevel(lap)->IsInRMatrix()) {
          tempE.push_back(this->GetJGroup(j)->GetLevel(lap)->GetFitE());
          if(this->GetJGroup(j)->GetLevel(lap)==theLevel) thisLevel=tempE.size()-1;
          vector_r tempChanVector;
          tempGamma.push_back(tempChanVector);
          for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
        tempGamma[tempE.size()-1].push_back(this->GetJGroup(j)->GetLevel(lap)->GetFitGamma(ch));
        if(tempE.size()==1)
tempBoundary.push_back(this->GetJGroup(j)->GetChannel(ch)->GetBoundaryCondition());
          }
        }
      }
      while(iteration<=maxIterations&&!done) {
        vector_r boundaryDiff;
        for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
          double newBoundary=0.0;
```

```
       AChannel *theChannel=this->GetJGroup(j)->GetChannel(ch);
       PPair *exitPair=this->GetPair(theChannel->GetPairNum());
       double localEnergy=tempE[thisLevel]-exitPair->GetSepE()-exitPair->GetExE();
       if(theChannel->GetRadType()=='P') {
      if(localEnergy<0.0) {
        ShftFunc theShiftFunction(exitPair);
        newBoundary=theShiftFunction(theChannel->GetL(),tempE[thisLevel]);
      }
      else {
        CoulFunc theCoulombFunction(exitPair,
                        !!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
        double radius=exitPair->GetChRad();
        newBoundary=theCoulombFunction.PEShift(theChannel->GetL(),radius,localEnergy);
      }
      boundaryDiff.push_back(newBoundary-tempBoundary[ch-1]);
      tempBoundary[ch-1]=newBoundary;
        } else boundaryDiff.push_back(boundaryDiff[0]);
      }
      matrix_r cMatrix;
      for(int mu=0;mu<tempE.size();mu++) {
        vector_r tempRow;
        cMatrix.push_back(tempRow);
        for (int mup=0;mup<tempE.size();mup++) {
      double chanSum=0.0;
      for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
        if(this->GetJGroup(j)->GetChannel(ch)->GetRadType()=='P')
          chanSum+=boundaryDiff[ch-1]*tempGamma[mu][ch-1]*
            tempGamma[mup][ch-1];
      }
      if(mu==mup) cMatrix[mu].push_back(tempE[mu]-chanSum);
      else cMatrix[mu].push_back(-chanSum);
        }
      }
      EigenFunc eigenFunc(cMatrix);
      if(fabs(eigenFunc.eigenvalues()[thisLevel]-tempE[thisLevel])<=energyTolerance)
        done=true;
      matrix_r newGamma;
      for(int mu=0;mu<tempE.size();mu++) {
        vector_r tempChanVector;
        newGamma.push_back(tempChanVector);
        for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
      double gammaSum=0.0;
      for(int mup=0;mup<tempE.size();mup++) {
        gammaSum+=eigenFunc.eigenvectors()[mup][mu]*tempGamma[mup][ch-1];
      }
      newGamma[mu].push_back(gammaSum);
        }
      }
      for(int mu=0;mu<tempE.size();mu++) {
        tempE[mu]=eigenFunc.eigenvalues()[mu];
        for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
      tempGamma[mu][ch-1]=newGamma[mu][ch-1];
        }
      }
      if(!done) {
        if(iteration==maxIterations) {
      configure.outStream « "**WARNING: Could Not Transform J = "
            « this->GetJGroup(j)->GetJ();
      if(this->GetJGroup(j)->GetPi()==-1) configure.outStream « '-';
      else configure.outStream « '+';
      configure.outStream « " E = " « theLevel->GetFitE() « " MeV**" « std::endl;
      tempE[thisLevel]=theLevel->GetFitE();
      for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++)
        tempGamma[thisLevel][ch-1]=theLevel->GetFitGamma(ch);
        }
        iteration++;
      }
    }

    theLevel->SetTransformE(tempE[thisLevel]);
    theLevel->SetTransformIterations(iteration);
    double nFSum=1.0;
    for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
      AChannel *theChannel=this->GetJGroup(j)->GetChannel(ch);
      theLevel->SetTransformGamma(ch,tempGamma[thisLevel][ch-1]);
      if(ch<=theLevel->NumNFIntegrals()) nFSum+=2.0*
                     this->GetPair(theChannel->GetPairNum())->GetChRad()*
                     this->GetPair(theChannel->GetPairNum())->GetRedMass()*
                     uconv/pow(hbarc,2.0)*pow(tempGamma[thisLevel][ch-1],2.0)*
                     theLevel->GetNFIntegral(ch);
    }
    theLevel->SetSqrtNFFactor(1.0/sqrt(nFSum));
  } else {
    theLevel->SetTransformE(theLevel->GetFitE());
    theLevel->SetTransformIterations(0);
    for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++)
      theLevel->SetTransformGamma(ch,theLevel->GetFitGamma(ch));
```

```
    }
      }
    }
  } else {
    for(int j=1;j<=this->NumJGroups();j++)
      for(int la=1;la<=this->GetJGroup(j)->NumLevels();la++) {
    this->GetJGroup(j)->GetLevel(la)->SetTransformIterations(0);
    this->GetJGroup(j)->GetLevel(la)->SetTransformE(this->GetJGroup(j)->GetLevel(la)->GetFitE());
    for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++)
      this->GetJGroup(j)->GetLevel(la)->
        SetTransformGamma(ch,this->GetJGroup(j)->GetLevel(la)->GetFitGamma(ch));
      }
  }

  for(int j=1;j<=this->NumJGroups();j++) {
    JGroup *theJGroup=this->GetJGroup(j);
    for(int la=1;la<=this->GetJGroup(j)->NumLevels();la++) {
      ALevel *theLevel=this->GetJGroup(j)->GetLevel(la);
      double normSum=0.0;
      vector_r tempPene;
      for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
    AChannel *theChannel=this->GetJGroup(j)->GetChannel(ch);
    PPair *exitPair=this->GetPair(theChannel->GetPairNum());
    double localEnergy=theLevel->GetTransformE()-exitPair->GetSepE()-exitPair->GetExE();
    if(theChannel->GetRadType()=='P') {
      if(localEnergy<0.0) {
        ShftFunc theShiftFunction(exitPair);
        normSum+=theShiftFunction.EnergyDerivative(theChannel->GetL(),theLevel->GetTransformE())*
          pow(theLevel->GetTransformGamma(ch),2.0);
        WhitFunc newWhitFunc(exitPair);
        double whitConv=newWhitFunc(theChannel->GetL(),
                        exitPair->GetChRad(),
                        fabs(localEnergy));
        double pene=exitPair->GetRedMass()*exitPair->GetChRad()*uconv/
          pow(hbarc,2.0)/pow(whitConv,2.0);
        tempPene.push_back(pene);
      }
      else {
        CoulFunc theCoulombFunction(exitPair,
                        !!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
        double radius=exitPair->GetChRad();
        normSum+=theCoulombFunction.PEShift_dE(theChannel->GetL(),radius,localEnergy)*
          pow(theLevel->GetTransformGamma(ch),2.0);
        double pene=theCoulombFunction.Penetrability(theChannel->GetL(),radius,localEnergy);
        tempPene.push_back(pene);
      }
    } else if(theChannel->GetRadType()=='M'||theChannel->GetRadType()=='E') {
      if(fabs(theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE())<1.e-3&&
         theJGroup->GetJ()==this->GetPair(theChannel->GetPairNum())->GetJ(2)&&
         theJGroup->GetPi()==this->GetPair(theChannel->GetPairNum())->GetPi(2)) {
        double jValue=theJGroup->GetJ();
        double pene=1e-10;
        if(theChannel->GetRadType()=='M'&&theChannel->GetL()==1)
          pene=3.0*jValue/4.0/(jValue+1.)/nuclearMagneton/nuclearMagneton;
        else if(theChannel->GetRadType()=='E'&&theChannel->GetL()==2)
          pene=60.0*jValue*(2.*jValue-1.)/(jValue+1.)/(2.*jValue+3.);
        if((int)(2*jValue)%2!=0) pene*=-1.;
        tempPene.push_back(pene);
      } else {
        double pene = (configure.paramMask & Config::USE_RMC_FORMALISM) ? 1.0 :
pow(fabs(localEnergy)/hbarc,2.0*theChannel->GetL()+1);
        tempPene.push_back(pene);
      }
    } else tempPene.push_back(1.0);
      }
      for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
    AChannel* theChannel = this->GetJGroup(j)->GetChannel(ch);
    complex externalWidth(0.0,0.0);
    if((theChannel->GetRadType()=='M'||theChannel->GetRadType()=='E') &&
       theLevel->IsInRMatrix()&&(configure.paramMask & Config::USE_EXTERNAL_CAPTURE) &&
       !(fabs(theLevel->GetTransformGamma(ch))<1.0e-8 && (configure.paramMask &
Config::IGNORE_ZERO_WIDTHS)))
      externalWidth=CalcExternalWidth(this->GetJGroup(j),theLevel,
                      this->GetJGroup(j)->GetChannel(ch),false,configure);
    theLevel->SetExternalGamma(ch,externalWidth);
    complex totalWidth=theLevel->GetTransformGamma(ch)+externalWidth;
    int tempSign = (real(totalWidth)<0.) ? (-1) : (1);
    double bigGamma;
    if(theChannel->GetRadType()!='F'&&theChannel->GetRadType()!='G')
      bigGamma=tempSign*2.0*real(totalWidth*conj(totalWidth))*tempPene[ch-1]/
        (1.0+normSum);
    else bigGamma=real(totalWidth);
    theLevel->SetBigGamma(ch,bigGamma);
      }
    }
  }
}
```

```
void CNuc::PrintTransformParams(const Config& configure) {
  char filename[256];;
  sprintf(filename,"%sparameters.out",configure.outputdir.c_str());
  std::ofstream out;
  out.open(filename);
  if(out) {
    out « "PHYSICAL LEVEL PARAMETERS (BOUNDARY CONDITION SET TO SHIFT AT E_LEVEL)" « std::endl;
    out « std::endl;
    for(int j=1;j<=this->NumJGroups();j++) {
      JGroup *theJGroup=this->GetJGroup(j);
      for(int la=1;la<=this->GetJGroup(j)->NumLevels();la++) {
    out.precision(1);
    out « std::fixed;
    ALevel *theLevel=this->GetJGroup(j)->GetLevel(la);
    out « "J = " « std::setw(3) « this->GetJGroup(j)->GetJ();
    if(this->GetJGroup(j)->GetPi()==-1) out « '-';
    else out « '+';
    out.precision(4);
    out « "  E_level = " « std::setw(8) « theLevel->GetTransformE() « " MeV"
        « "  ITERATIONS = " « std::setw(5) « theLevel->GetTransformIterations() « std::endl;
    for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
      AChannel *theChannel=this->GetJGroup(j)->GetChannel(ch);
      PPair *exitPair=this->GetPair(theChannel->GetPairNum());
      double localEnergy=theLevel->GetTransformE()-exitPair->GetSepE()-exitPair->GetExE();
      out « "  R = " « std::setw(2) « exitPair->GetPairKey();
      if(theChannel->GetRadType()=='P') out « "  l = " « std::setw(3) « theChannel->GetL();
      else if(theChannel->GetRadType()=='F') out « "  Fermi Beta Decay ";
      else if(theChannel->GetRadType()=='G') out « "   G-T Beta Decay  ";
      else out « "  L = " « std::setw(2) « theChannel->GetRadType() « theChannel->GetL();
      out.precision(1);
      if(theChannel->GetRadType()!='G'&&theChannel->GetRadType()!='F')
        out « "  s = " « std::setw(4) « theChannel->GetS();
      out.precision(6);
      if(localEnergy<0.0&&theChannel->GetRadType()=='P') {
        out « "  C  = " « std::setw(12) « sqrt(fabs(theLevel->GetBigGamma(ch)))
        « " fm^(-1/2)";
      } else if(fabs(theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE())<1.e-3&&
         theJGroup->GetJ()==this->GetPair(theChannel->GetPairNum())->GetJ(2)&&
         theJGroup->GetPi()==this->GetPair(theChannel->GetPairNum())->GetPi(2)&&
            theChannel->GetRadType()=='M'&&theChannel->GetL()==1) {
        int tempSign = (theLevel->GetBigGamma(ch)<0) ? (-1) : (1);
        out « "  mu  = " « std::setw(12) « tempSign*sqrt(fabs(theLevel->GetBigGamma(ch)))
        « " nm        ";
      } else if(fabs(theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE())<1.e-3&&
         theJGroup->GetJ()==this->GetPair(theChannel->GetPairNum())->GetJ(2)&&
         theJGroup->GetPi()==this->GetPair(theChannel->GetPairNum())->GetPi(2)&&
            theChannel->GetRadType()=='E'&&theChannel->GetL()==2) {
        int tempSign = (theLevel->GetBigGamma(ch)<0) ? (-1) : (1);
        out « "  Q  = " « std::setw(12) «
tempSign*sqrt(fabs(theLevel->GetBigGamma(ch)))/100.0/sqrt(fstruc*hbarc)
        « " b         ";
      } else if(theChannel->GetRadType()=='F'|| theChannel->GetRadType()=='G') {
        out « "  B  = " « std::setw(12) « theLevel->GetBigGamma(ch)
        « "           ";
      } else {
        if(fabs(theLevel->GetBigGamma(ch))>=1e-3)
          out « "  G  = " « std::setw(12) « fabs(theLevel->GetBigGamma(ch))*1e3
          « " keV      ";
        else if(fabs(theLevel->GetBigGamma(ch))>=1e-6)
          out « "  G  = " « std::setw(12) « fabs(theLevel->GetBigGamma(ch))*1e6
          « " eV       ";
        else
          out « "  G  = " « std::setw(12) « fabs(theLevel->GetBigGamma(ch))*1e9
          « " meV      ";
      }
      out « "  g_int = " « std::setw(12) « theLevel->GetTransformGamma(ch);
      if(theChannel->GetRadType()!='G'&&theChannel->GetRadType()!='F') out « " MeV^(1/2) ";
      else out « "            ";
      out « "  g_ext = " « std::setw(20) « theLevel->GetExternalGamma(ch);
      if(theChannel->GetRadType()!='G'&&theChannel->GetRadType()!='F') out « " MeV^(1/2) ";
      out « std::endl;
    }
    out « std::endl;
      }
    }
  } else configure.outStream « "Could not save parameters.out file." « std::endl;
}

void CNuc::SetMaxLValue(int maxL) {
  maxLValue_=maxL;
}

void CNuc::CalcShiftFunctions(const Config& configure) {
  for(int j=1;j<=this->NumJGroups();j++) {
    if(this->GetJGroup(j)->IsInRMatrix()) {
      JGroup *theJGroup=this->GetJGroup(j);
```

```
      for(int la=1;la<=theJGroup->NumLevels();la++) {
    ALevel *theLevel=theJGroup->GetLevel(la);
    if(theLevel->IsInRMatrix()) {
      for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
        AChannel *theChannel=theJGroup->GetChannel(ch);
        PPair *thePair=this->GetPair(theChannel->GetPairNum());
        if(thePair->GetPType()==0) {
          int lValue=theChannel->GetL();
          double levelEnergy=theLevel->GetFitE();
          double resonanceEnergy=levelEnergy-(thePair->GetSepE()+thePair->GetExE());
          if(resonanceEnergy<0.0) {
        ShftFunc theShiftFunction(thePair);
        theLevel->SetShiftFunction(ch,theShiftFunction(lValue,levelEnergy));
          }
          else {
        CoulFunc theCoulombFunction(thePair,
                                !!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
        double radius=thePair->GetChRad();
        theLevel->SetShiftFunction(ch,theCoulombFunction.PEShift(lValue,radius,resonanceEnergy));
          }
        }
        else {
          theLevel->SetShiftFunction(ch,theJGroup->GetLevel(1)->GetShiftFunction(1));
        }
      }
    }
      }
    }
  }
}

complex CNuc::CalcExternalWidth(JGroup* theJGroup, ALevel* theLevel,
                                AChannel *theChannel,bool isInitial,const Config& configure) {
  complex externalWidth(0.0,0.0);
  if(theChannel->GetRadType()=='E'||(theChannel->GetRadType()=='M'&&theChannel->GetL()==1)) {
    bool isExternal=false;
    int j=0;
    int la=0;
    while(!isExternal&&j<this->NumJGroups()) {
      j++;
      la=0;
      while(!isExternal&&la<this->GetJGroup(j)->NumLevels()) {
    la++;
    if(this->GetJGroup(j)->GetLevel(la)->IsECLevel()&&
       theChannel->GetPairNum()==this->GetJGroup(j)->GetLevel(la)->GetECPairNum()) {
      isExternal=true;
    }
      }
    }
    if(isExternal) {
      JGroup *theFinalJGroup=this->GetJGroup(j);
      ALevel *theFinalLevel=theFinalJGroup->GetLevel(la);
      double theLevelEnergy;
      if(!isInitial) theLevelEnergy=theLevel->GetTransformE();
      else theLevelEnergy=theLevel->GetE();
      int multL=theChannel->GetL();
      if(((theChannel->GetRadType()=='E' && multL==1) && (theFinalLevel->GetECMultMask()&isE1)) ||
     ((theChannel->GetRadType()=='M' && multL==1) && (theFinalLevel->GetECMultMask()&isM1)) ||
     ((theChannel->GetRadType()=='E' && multL==2) && (theFinalLevel->GetECMultMask()&isE2)) ) {
//allow only m1,e1,e2
    double theFinalLevelEnergy;
    if(!isInitial) theFinalLevelEnergy=theFinalLevel->GetTransformE();
    else theFinalLevelEnergy=theFinalLevel->GetE();
    for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
      double theInitialChannelGamma;
      if(!isInitial) theInitialChannelGamma=theLevel->GetTransformGamma(ch);
      else theInitialChannelGamma=theLevel->GetGamma(ch);
      AChannel *initialChannel=theJGroup->GetChannel(ch);
      if(initialChannel->GetRadType()=='P') {
        for(int chp=1;chp<=theFinalJGroup->NumChannels();chp++) {
          double theFinalChannelGamma;
          if(!isInitial) theFinalChannelGamma=theFinalLevel->GetTransformGamma(chp);
          else theFinalChannelGamma=theFinalLevel->GetGamma(chp);
          AChannel *finalChannel=theFinalJGroup->GetChannel(chp);
          if(finalChannel->GetRadType()=='P') {
        if(finalChannel->GetPairNum()==initialChannel->GetPairNum()) {

if((abs(initialChannel->GetL()-multL)<=finalChannel->GetL()&&finalChannel->GetL()<=initialChannel->GetL()+multL&&
              fabs(initialChannel->GetS()-finalChannel->GetL())<=theFinalJGroup->GetJ()&&

theFinalJGroup->GetJ()<=initialChannel->GetS()+finalChannel->GetL()&&initialChannel->GetS()==finalChannel->GetS())||

(fabs(initialChannel->GetS()-multL)<=finalChannel->GetS()&&finalChannel->GetS()<=initialChannel->GetS()+multL&&
              fabs(initialChannel->GetL()-finalChannel->GetS())<=theFinalJGroup->GetJ()&&

theFinalJGroup->GetJ()<=initialChannel->GetL()+finalChannel->GetS()&&initialChannel->GetL()==finalChannel->GetL()&&
              theChannel->GetRadType()=='M')) {
```

```
            PPair *theFinalPair=this->GetPair(finalChannel->GetPairNum());

            ECIntegral theECIntegral(theFinalPair,configure);
            complex integrals = theECIntegral(initialChannel->GetL(),finalChannel->GetL(),
                                  initialChannel->GetS(),finalChannel->GetS(),
                                  theJGroup->GetJ(),theFinalJGroup->GetJ(),
                                  multL,theChannel->GetRadType(),
                                  theLevelEnergy,theFinalLevelEnergy,
                                  true);

            double ecNormParam=theFinalChannelGamma*
              theFinalLevel->GetSqrtNFFactor()*theFinalLevel->GetECConversionFactor(chp);
            externalWidth-=ecNormParam*theInitialChannelGamma*integrals;
          }
        }
          }
        }
      }
    }
      }
    }
  }
  return externalWidth;
}

PPair *CNuc::GetPair(int pairNum) {
  PPair *b = &pairs_[pairNum-1];
  return b;
}

JGroup *CNuc::GetJGroup(int jGroupNum) {
  JGroup *b = &jgroups_[jGroupNum-1];
  return b;
}

CNuc *CNuc::Clone() const {
  CNuc *localCompound = new CNuc(*this);
  return localCompound;
}
```

8.242 /Users/kuba/Desktop/R-Matrix/AZURE2/src/Config.cpp File Reference

```
#include ¨Config.h¨
#include <sys/stat.h>
#include <iostream>
```

8.243 Config.cpp

Go to the documentation of this file.

```
#include "Config.h"
#ifndef NO_STAT
#include <sys/stat.h>
#endif
#include <iostream>

Config::Config(std::ostream& stream) : outStream(stream) {
  Reset();
}

void Config::Reset() {
  chiVariance=1.0;
  screenCheckMask=0;
  fileCheckMask=0;
  paramMask=0;
  paramMask |=
(USE_AMATRIX|USE_BRUNE_FORMALISM|IGNORE_ZERO_WIDTHS|TRANSFORM_PARAMETERS|CALCULATE_WITH_DATA|USE_LONGWAVELENGTH_APPROX);
  stopFlag=false;
  outputdir="";
  checkdir="";
}
```

```

int Config::ReadConfigFile() {
  std::string dummy; std::string temp;
  std::ifstream in(configfile.c_str());
  if(!in) return -1;
  std::string line="";
  while(line!="<config>"&&!in.eof()) getline(in,line);
  if(line!="<config>") return -1;
  in » temp;getline(in,dummy);
  if(temp=="true") paramMask |= USE_AMATRIX;
  else paramMask &= ~USE_AMATRIX;
  getline(in,dummy);
  int poundSignPos=dummy.find_last_of('#');
  if(poundSignPos==std::string::npos) temp=dummy;
  else temp=dummy.substr(0,poundSignPos);
  int p2 = temp.find_last_not_of(" \n\t\r");
  if (p2 != std::string::npos) {
    int p1 = temp.find_first_not_of(" \n\t\r");
    if (p1 == std::string::npos) p1 = 0;
    outputdir=temp.substr(p1,(p2-p1)+1);
  } else outputdir=std::string();
  getline(in,dummy);
  poundSignPos=dummy.find_last_of('#');
  if(poundSignPos==std::string::npos) temp=dummy;
  else temp=dummy.substr(0,poundSignPos);
  p2 = temp.find_last_not_of(" \n\t\r");
  if (p2 != std::string::npos) {
    int p1 = temp.find_first_not_of(" \n\t\r");
    if (p1 == std::string::npos) p1 = 0;
    checkdir=temp.substr(p1,(p2-p1)+1);
  } else checkdir=std::string();
  in » temp;getline(in,dummy);
  if(temp=="screen") screenCheckMask |= CHECK_COMPOUND_NUCLEUS;
  else if(temp=="file") fileCheckMask |= CHECK_COMPOUND_NUCLEUS;
  in » temp;getline(in,dummy);
  if(temp=="screen") screenCheckMask |= CHECK_BOUNDARY_CONDITIONS;
  else if(temp=="file") fileCheckMask |= CHECK_BOUNDARY_CONDITIONS;
  in » temp;getline(in,dummy);
  if(temp=="screen") screenCheckMask |= CHECK_DATA;
  else if(temp=="file") fileCheckMask |= CHECK_DATA;
  in » temp;getline(in,dummy);
  if(temp=="screen") screenCheckMask |= CHECK_ENERGY_DEP;
  else if(temp=="file") fileCheckMask |= CHECK_ENERGY_DEP;
  in » temp;getline(in,dummy);
  if(temp=="screen") screenCheckMask |= CHECK_LEGENDRE;
  else if(temp=="file") fileCheckMask |= CHECK_LEGENDRE;
  in » temp;getline(in,dummy);
  if(temp=="screen") screenCheckMask |= CHECK_COUL_AMPLITUDES;
  else if(temp=="file") fileCheckMask |= CHECK_COUL_AMPLITUDES;
  in » temp;getline(in,dummy);
  if(temp=="screen") screenCheckMask |= CHECK_PATHWAYS;
  else if(temp=="file") fileCheckMask |= CHECK_PATHWAYS;
  in » temp;getline(in,dummy);
  if(temp=="screen") screenCheckMask |= CHECK_ANGULAR_DISTS;
  else if(temp=="file") fileCheckMask |= CHECK_ANGULAR_DISTS;
  line="";
  while(line!="</config>"&&!in.eof()) getline(in,line);
  if(line!="</config>") return -1;
  in.close();
  return 0;
}

#ifndef NO_STAT
int Config::CheckForInputFiles() {
  struct stat buffer;
  if(stat(outputdir.c_str(),&buffer) != 0) {
    outStream « "Could not find output directory: " « outputdir « ". Check that it exists." «
std::endl;
    return -1;
  }
  if(stat(checkdir.c_str(),&buffer) != 0) {
    outStream « "Could not find checks directory: " « checkdir « ". Check that it exists." «
std::endl;
    return -1;
  }
  return 0;
}
#endif
```

8.244 /Users/kuba/Desktop/R-Matrix/AZURE2/src/CoulFunc.cpp File Reference

```
#include ¨CoulFunc.h¨
#include ¨PPair.h¨
#include <iostream>
#include ¨cwfcomp.H¨
#include <gsl/gsl_sf_coulomb.h>
#include <gsl/gsl_deriv.h>
```

8.245 CoulFunc.cpp

Go to the documentation of this file.

```
#include "CoulFunc.h"
#include "PPair.h"
#include <iostream>
#include "cwfcomp.H"
#include <gsl/gsl_sf_coulomb.h>
#include <gsl/gsl_deriv.h>

CoulFunc::CoulFunc(PPair *pPair, bool useGSLFunctions) :
  useGSLFunctions_(useGSLFunctions) {
  z1_=pPair->GetZ(1);
  z2_=pPair->GetZ(2);
  redmass_=(double)pPair->GetRedMass();
  lLast_=0;
  radiusLast_=0.0;
  energyLast_=0.0;
  coulLast_.F=0.0;
  coulLast_.dF=0.0;
  coulLast_.G=0.0;
  coulLast_.dG=0.0;
  dEShiftParams_.coulFunc=this;
}

int CoulFunc::z1() const {
  return z1_;
}

int CoulFunc::z2() const {
  return z2_;
}

double CoulFunc::redmass() const {
  return redmass_;
}

int CoulFunc::lLast() const {
  return lLast_;
}

double CoulFunc::radiusLast() const {
  return radiusLast_;
}

double CoulFunc::energyLast() const {
  return energyLast_;
}

struct CoulWaves CoulFunc::coulLast() const {
  return coulLast_;
}

void CoulFunc::setLast(int lLast, double rLast, double eLast, CoulWaves coulLast) {
  lLast_=lLast;
  radiusLast_=rLast;
  energyLast_=eLast;
  coulLast_.F=coulLast.F;
  coulLast_.dF=coulLast.dF;
  coulLast_.G=coulLast.G;
  coulLast_.dG=coulLast.dG;
}

CoulWaves CoulFunc::operator()(int l,double radius,double energy) {
```

```
  struct CoulWaves result={0.0,0.0,0.0,0.0};
  if(l==lLast()&&radius==radiusLast()&&energy==energyLast()) {
    result=coulLast();
  } else {
    struct CoulWaves newResult;
    if(!useGSLFunctions_) {
      std::complex<double> eta(sqrt(uconv/2.)*fstruc*z1()*z2()*
                   sqrt(redmass()/energy),0.);
      std::complex<double> rho(sqrt(2.*uconv)/hbarc*radius*
                   sqrt(redmass()*energy),0.);
      std::complex<double> lValue( (double) l, 0.);
      Coulomb_wave_functions coul(true,lValue,eta);
      std::complex<double> c_F, c_dF, c_G, c_dG;
      coul.F_dF(rho,c_F,c_dF);
      coul.G_dG(rho,c_G,c_dG);
      newResult.F=real(c_F);
      newResult.dF=real(c_dF);
      newResult.G=real(c_G);
      newResult.dG=real(c_dG);
    } else {
      double eta=sqrt(uconv/2.)*fstruc*z1()*z2()*
    sqrt(redmass()/energy);
      double rho=sqrt(2.*uconv)/hbarc*radius*
    sqrt(redmass()*energy);
      double lValue=double(l);
      double eF,eG;
      gsl_sf_result F,Fp,G,Gp;
      gsl_sf_coulomb_wave_FG_e(eta,rho,lValue,0,&F,&Fp,&G,&Gp,&eF,&eG);
      newResult.F=F.val*exp(eF);
      newResult.dF=Fp.val*exp(eF);
      newResult.G=G.val*exp(eG);
      newResult.dG=Gp.val*exp(eG);
    }
    setLast(l,radius,energy,newResult);
    result=newResult;
  }
  return result;
}

double CoulFunc::Penetrability(int l,double radius,double energy)  {
  struct CoulWaves coul=this->operator()(l,radius,energy);
  double rho=sqrt(2.*uconv)/hbarc*radius*sqrt(redmass()*energy);
  return rho/(pow(coul.F,2.0)+pow(coul.G,2.0));
}

double CoulFunc::PEShift(int l,double radius,double energy)  {
  struct CoulWaves coul=this->operator()(l,radius,energy);
  double rho=sqrt(2.*uconv)/hbarc*radius*sqrt(redmass()*energy);
  if(pow(coul.F,2.0)==0.&&coul.F*coul.dF==0.) return rho*(coul.dG/coul.G);
  else return rho/(pow(coul.F,2.0)+pow(coul.G,2.0))*
    (coul.F*coul.dF+coul.G*coul.dG);
}

double CoulFunc::thisPEShift(double x, void *p) {
  DEShiftParams *params = (DEShiftParams*)p;
  CoulFunc *coulFunc=(params->coulFunc);
  int lValue=(params->lValue);
  double radius=(params->radius);

  return coulFunc->PEShift(lValue,radius,x);
}

double CoulFunc::PEShift_dE(int l,double radius,double energy)  {
  double result;
  double error;

  dEShiftParams_.radius=radius;
  dEShiftParams_.lValue=l;

  gsl_function F;
  F.function=&thisPEShift;
  F.params=&dEShiftParams_;

  gsl_deriv_central (&F, energy, 1e-6, &result, &error);

  return result;
}

```

8.246 /Users/kuba/Desktop/R-Matrix/AZURE2/src/Decay.cpp File Reference

```
#include "Decay.h"
```

8.247 Decay.cpp

Go to the documentation of this file.

```
#include "Decay.h"

Decay::Decay(int pairNum) :
  pair_(pairNum) {};

int Decay::GetPairNum() const {
  return pair_;
}

int Decay::NumKGroups() const {
  return kgroups_.size();
}

int Decay::NumKLGroups() const {
  return klgroups_.size();
}

int Decay::IsKGroup(KGroup a) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumKGroups())
    {
      if(a.GetS()==this->GetKGroup(c+1)->GetS()&&
     a.GetSp()==this->GetKGroup(c+1)->GetSp()) b=true;
     c++;
    }
  if(b) return c;
  else return 0;
}

int Decay::IsKLGroup(KLGroup a) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumKLGroups())
    {
      if(a.GetK()==this->GetKLGroup(c+1)->GetK()&&
     a.GetLOrder()==this->GetKLGroup(c+1)->GetLOrder()) b=true;
     c++;
    }
  if(b) return c;
  else return 0;
}

void Decay::AddKGroup(KGroup kGroup) {
  kgroups_.push_back(kGroup);
}

void Decay::AddKLGroup(KLGroup klGroup) {
  klgroups_.push_back(klGroup);
}

KGroup *Decay::GetKGroup(int kGroupNum) {
  KGroup *b=&kgroups_[kGroupNum-1];
  return b;
}

KLGroup *Decay::GetKLGroup(int klGroupNum) {
  KLGroup *b=&klgroups_[klGroupNum-1];
  return b;
}
```

8.248 /Users/kuba/Desktop/R-Matrix/AZURE2/src/DoubleFactorial.cpp File Reference

Functions

- double DoubleFactorial (int a)

8.248.1 Function Documentation

8.248.1.1 DoubleFactorial()

```
double DoubleFactorial (
            int a )
```

Definition at line 1 of file DoubleFactorial.cpp.

8.249 DoubleFactorial.cpp

Go to the documentation of this file.

```
00001 double DoubleFactorial(int a) {
00002   double b=1;
00003   while(a>1) {
00004     b=b*a;
00005     a-=2;
00006   }
00007   return b;
00008 }
```

8.250 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ECIntegral.cpp File Reference

```
#include "ECIntegral.h"
#include "AngCoeff.h"
#include "EffectiveCharge.h"
#include <math.h>
#include <gsl/gsl_integration.h>
#include <assert.h>
```

Functions

- double DoubleFactorial (int)

8.250.1 Function Documentation

8.250.1.1 DoubleFactorial()

```
double DoubleFactorial (
            int a )
```

Definition at line 1 of file DoubleFactorial.cpp.

8.251 ECIntegral.cpp

Go to the documentation of this file.

```
#include "ECIntegral.h"
#include "AngCoeff.h"
#include "EffectiveCharge.h"
#include <math.h>
#include <gsl/gsl_integration.h>
#include <assert.h>

extern double DoubleFactorial(int);

double ECIntegral::FWIntegrand(double x, void * p) {
  Params *params = (Params*)p;
  CoulFunc *theCoulFunc=(params->coulFunc);
  WhitFunc *theWhitFunc=(params->whitFunc);
  int liValue = (params->liValue);
  int lfValue = (params->lfValue);
  int multLValue = (params->multLValue);
  double pairEnergy = (params->pairEnergy);
  double bindingEnergy = (params->bindingEnergy);

  struct CoulWaves coul = theCoulFunc->operator()(liValue,x,pairEnergy);
  double whit = theWhitFunc->operator()(lfValue,x,bindingEnergy);
  double returnValue = coul.F*whit*pow(x,multLValue);
  return (!params->useLongWavelengthApprox) ? params->effectiveCharge->operator()(x)*returnValue :
    returnValue;
}

double ECIntegral::GWIntegrand(double x, void * p) {
  Params *params = (Params*)p;
  CoulFunc *theCoulFunc=(params->coulFunc);
  WhitFunc *theWhitFunc=(params->whitFunc);
  int liValue = (params->liValue);
  int lfValue = (params->lfValue);
  int multLValue = (params->multLValue);
  double pairEnergy = (params->pairEnergy);
  double bindingEnergy = (params->bindingEnergy);

  struct CoulWaves coul = theCoulFunc->operator()(liValue,x,pairEnergy);
  double whit = theWhitFunc->operator()(lfValue,x,bindingEnergy);
  double returnValue = coul.G*whit*pow(x,multLValue);
  return (!params->useLongWavelengthApprox) ?  params->effectiveCharge->operator()(x)*returnValue :
    returnValue;
}

double ECIntegral::WWIntegrand(double x, void * p) {
  Params *params = (Params*)p;
  WhitFunc *theWhitFunc=(params->whitFunc);
  int liValue = (params->liValue);
  int lfValue = (params->lfValue);
  int multLValue = (params->multLValue);
  double pairEnergy = (params->pairEnergy);
  double bindingEnergy = (params->bindingEnergy);

  double whitIn = theWhitFunc->operator()(liValue,x,fabs(pairEnergy));
  double whitOut= theWhitFunc->operator()(lfValue,x,fabs(bindingEnergy));
  double returnValue = whitIn*whitOut*pow(x,multLValue);
  return (!params->useLongWavelengthApprox) ? params->effectiveCharge->operator()(x)*returnValue :
    returnValue;
}

void ECIntegral::Integrate(double chanrad) {
  gsl_integration_workspace * w
    = gsl_integration_workspace_alloc (1000);

  if(params_.pairEnergy<0.0) {
    gsl_function WW;
    WW.function = &WWIntegrand;
    WW.params= &params_;

    double wwintresult,wwinterror;

    gsl_integration_qagiu(&WW,chanrad,0.0,1e-4,1000,w,&wwintresult,&wwinterror);
    GW_=wwintresult;
    FW_=0.0;
  } else {
    gsl_function FW;
    FW.function = &FWIntegrand;
    FW.params= &params_;

    gsl_function GW;
    GW.function = &GWIntegrand;
    GW.params= &params_;

```

```
    double fwintresult,fwinterror;
    double gwintresult,gwinterror;

    gsl_integration_qagiu(&FW,chanrad,0.0,1e-4,1000,w,&fwintresult,&fwinterror);
    gsl_integration_qagiu(&GW,chanrad,0.0,1e-4,1000,w,&gwintresult,&gwinterror);
    FW_=fwintresult;
    GW_=gwintresult;
  }

  gsl_integration_workspace_free (w);
}

complex ECIntegral::operator()(int theInitialLValue, int theFinalLValue,
                       double theInitialSValue, double theFinalSValue,
                       double theInitialJValue, double theFinalJValue,
                       int theLMult, char radType,
                       double inEnergy, double levelEnergy,
                       bool isChannelCapture) {
  ResetIntegrals();

  double sepEnergy = pair()->GetSepE()+pair()->GetExE();
  double outEnergy = inEnergy - sepEnergy;
  double chanRad = pair()->GetChRad();
  double redMass = pair()->GetRedMass();

  EffectiveCharge effectiveChargeFunc(pair(),inEnergy-levelEnergy,theLMult);

  params_.effectiveCharge=&effectiveChargeFunc;
  params_.liValue = theInitialLValue;
  params_.lfValue = theFinalLValue;
  params_.multLValue = theLMult;
  params_.pairEnergy = outEnergy;
  params_.bindingEnergy = fabs(levelEnergy-sepEnergy);

  if(radType=='E')
    params_.multLValue = theLMult;
  else
    params_.multLValue = 0;
  Integrate(chanRad);

  complex overlapIntegral(0.,0.);
  if(outEnergy>0.0) {
    struct CoulWaves
      coul=coulfunction()->operator()(theInitialLValue,chanRad,outEnergy);
    if(isChannelCapture) {
      complex chanExpHSP(coul.G/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)),
             -coul.F/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)));
      overlapIntegral=complex(0.0,-0.5)*
    sqrt(coulfunction()->Penetrability(theInitialLValue,chanRad,outEnergy))*
    pow(redMass*uconv/2./fabs(outEnergy),0.25)/sqrt(hbarc)*
    chanExpHSP*(GW()+complex(0.0,1.0)*FW());
    } else overlapIntegral=(coul.G/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0))*FW()
                -coul.F/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0))*GW())*
         pow(redMass*uconv/2./fabs(outEnergy),0.25)/sqrt(hbarc);
  } else {
    assert(isChannelCapture);
    double whit=whitfunction()->operator()(theInitialLValue,chanRad,fabs(outEnergy));
    overlapIntegral=complex(0.0,-0.5)*GW()/whit*
      sqrt(redMass*uconv*chanRad)/hbarc;
  }


  double effectiveCharge;
  if(radType=='E') {
    if(params_.useLongWavelengthApprox) {
      double totalM=pair()->GetM(1)+pair()->GetM(2);
      effectiveCharge=sqrt(fstruc*hbarc)*(pair()->GetZ(1)*pow(pair()->GetM(2)/totalM,theLMult)+
                      pair()->GetZ(2)*pow(-pair()->GetM(1)/totalM,theLMult));
    } else effectiveCharge = 1.;
  } else {
    effectiveCharge=redMass*1.00727638*
      (pair()->GetZ(1)/pow(pair()->GetM(1),2.)+
       pair()->GetZ(2)/pow(pair()->GetM(2),2.));
  }

  complex ecAmplitude(0.0,0.0);
  if(radType=='E') {
    ecAmplitude=complex(0.0,-1.0)*
effectiveCharge*sqrt((8.*(2.*theLMult+1.)*(theLMult+1.))/theLMult)/DoubleFactorial(2*theLMult+1)*
      pow(complex(0.,1.0),theInitialLValue+theLMult-theFinalLValue)*
AngCoeff::ClebGord(theInitialLValue,theLMult,theFinalLValue,0,0,0)*sqrt(2.*theInitialLValue+1.)*sqrt(2.*theFinalJValue+
AngCoeff::Racah(theLMult,theFinalLValue,theInitialJValue,theInitialSValue,theInitialLValue,theFinalJValue);
  } else {
```

```
    complex orbitalTerm=effectiveCharge*
      sqrt((2.*theInitialLValue+1.)*(theInitialLValue+1.)*theInitialLValue)*
AngCoeff::Racah(1.,theInitialLValue,theInitialJValue,theInitialSValue,theInitialLValue,theFinalJValue);
    complex tau=pow(std::complex<double>(-1.,0.),pair()->GetJ(1)+pair()->GetJ(2))*
      (pow(complex(-1.,0.),theFinalSValue)*
       sqrt(pair()->GetJ(1)*(pair()->GetJ(1)+1.)*(2.*pair()->GetJ(1)+1.))*
AngCoeff::Racah(theFinalSValue,pair()->GetJ(1),theInitialSValue,pair()->GetJ(1),pair()->GetJ(2),1.)*
       pair()->GetG(1)+
       pow(complex(-1.,0.),theInitialSValue)*
       sqrt(pair()->GetJ(2)*(pair()->GetJ(2)+1.)*(2.*pair()->GetJ(2)+1.))*
AngCoeff::Racah(theFinalSValue,pair()->GetJ(2),theInitialSValue,pair()->GetJ(2),pair()->GetJ(1),1.)*
       pair()->GetG(2));
    complex spinTerm=-sqrt((2.*theInitialSValue+1.)*(2.*theFinalSValue+1.))*
AngCoeff::Racah(1,theInitialSValue,theFinalJValue,theInitialLValue,theFinalSValue,theInitialJValue)*tau;
    ecAmplitude=complex(0.0,1.0)*
      sqrt(fstruc)*pow(hbarc,1.5)/(2*1.00727638*uconv)*sqrt(16/3)*sqrt(2*theFinalJValue+1.)*
      (orbitalTerm+spinTerm);
  }

  return ecAmplitude*overlapIntegral;
};
```

8.252 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ECMGroup.cpp File Reference

```
#include "ECMGroup.h"
#include <assert.h>
```

8.253 ECMGroup.cpp

Go to the documentation of this file.

```
#include "ECMGroup.h"
#include <assert.h>

ECMGroup::ECMGroup(char radType, int multipolarity, int lInitial, double jInitial, int
finalChannelNum, int ecJGroupNum, int ecLevelNum) :
  radtype_(radType), mult_(multipolarity), li_(lInitial), ji_(jInitial), chf_(finalChannelNum),
jGroupNum_(ecJGroupNum), levelNum_(ecLevelNum),
   ischancap_(false), chdecay_(0), chkgroup_(0), chmgroup_(0), internalChannel_(0),
statspinfactor_(0.0){
}

ECMGroup::ECMGroup(char radType, int multipolarity, int lInitial, double jInitial, int
finalChannelNum, int ecJGroupNum, int ecLevelNum, int decayNum, int kGroupNum, int mGroupNum, int
internalChannel) :
  radtype_(radType), mult_(multipolarity), li_(lInitial), ji_(jInitial), chf_(finalChannelNum),
jGroupNum_(ecJGroupNum), levelNum_(ecLevelNum),
  ischancap_(true), chdecay_(decayNum), chkgroup_(kGroupNum), chmgroup_(mGroupNum),
internalChannel_(internalChannel), statspinfactor_(0.0){
}

bool ECMGroup::IsChannelCapture() const {
  return ischancap_;
}

char ECMGroup::GetRadType() const {
  return radtype_;
}

int ECMGroup::GetMult() const {
  return mult_;
}

int ECMGroup::GetL() const {
  return li_;
}

int ECMGroup::GetFinalChannel() const {
```

```
  return chf_;
}

int ECMGroup::GetJGroupNum() const {
  return jGroupNum_;
}

int ECMGroup::GetLevelNum() const {
  return levelNum_;
}

int ECMGroup::GetChanCapDecay() const {
  return chdecay_;
}

int ECMGroup::GetChanCapKGroup() const {
  return chkgroup_;
}

int ECMGroup::GetChanCapMGroup() const {
  return chmgroup_;
}

int ECMGroup::GetIntChannelNum() const {
  return internalChannel_;
}

double ECMGroup::GetJ() const {
  return ji_;
}

double ECMGroup::GetStatSpinFactor() const {
  return statspinfactor_;
}

void ECMGroup::SetStatSpinFactor(double a) {
  statspinfactor_=a;
}
```

8.254 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EData.cpp File Reference

#include "AZUREOutput.h"
#include "CNuc.h"
#include "Config.h"
#include "EData.h"
#include "ExtrapLine.h"
#include "SegLine.h"
#include "Minuit2/MnUserParameters.h"
#include "GSLException.h"
#include <iostream>
#include <iomanip>
#include <omp.h>
#include <time.h>

8.255 EData.cpp

Go to the documentation of this file.

```
#include "AZUREOutput.h"
#include "CNuc.h"
#include "Config.h"
#include "EData.h"
#include "ExtrapLine.h"
#include "SegLine.h"
#include "Minuit2/MnUserParameters.h"
#include "GSLException.h"
#include <iostream>
```

```
#include <iomanip>
#include <omp.h>
#include <time.h>

EData::EData() {
  iterations_=0;
  isFit_=true;
  isErrorAnalysis_=false;
}

int EData::NumSegments() const {
  return segments_.size();
}

int EData::Fill(const Config& configure, CNuc *theCNuc) {
  std::ifstream in(configure.configfile.c_str());
  if(!in) return -1;
  std::string line="";
  while(line!="<segmentsData>"&&!in.eof()) getline(in,line);
  if(line!="<segmentsData>") return -1;
  line="";
  int numTotalSegments=0;
  while(!in.eof()&&line!="</segmentsData>") {
    getline(in,line);
    bool empty=true;
    for(unsigned int i=0;i<line.size();++i)
      if(line[i]!=' '&&line[i]!='\t') {
    empty=false;
    break;
      }
    if(empty==true) continue;
    if(!in.eof()&&line!="</segmentsData>") {
      std::istringstream stm;
      stm.str(line);
      SegLine segment(stm);
      if(stm.rdstate() & (std::stringstream::failbit | std::stringstream::badbit)) return -1;
      numTotalSegments++;
      if(segment.isActive()==1) {
    ESegment NewSegment(segment);
    if(theCNuc->IsPairKey(NewSegment.GetEntranceKey())) {
      theCNuc->GetPair(theCNuc->GetPairNumFromKey(NewSegment.GetEntranceKey()))->SetEntrance();
      bool isValidTotal=false;
      if(NewSegment.GetExitKey()==-1) {
        for(int i = 1;i<=theCNuc->NumPairs();i++) {
          if(theCNuc->GetPair(i)->GetPType()==10) {
        isValidTotal = true;
        break;
          }
        }
      }
      if(isValidTotal||theCNuc->IsPairKey(NewSegment.GetExitKey())) {
        NewSegment.SetSegmentKey(numTotalSegments);
        this->AddSegment(NewSegment);
        if(this->GetSegment(this->NumSegments())->Fill(theCNuc,this,configure)==-1) {
              configure.outStream « "WARNING: Could Not Fill Segment #" « this->NumSegments()
            « " from file." « std::endl;
          this->DeleteLastSegment();
        } else if(this->GetSegment(this->NumSegments())->NumPoints()==0) {
          configure.outStream « "WARNING: Segment #" « numTotalSegments
                  « " is empty and will not be used." « std::endl;
          this->DeleteLastSegment();
        } else {
          int thisSegmentNum = this->NumSegments();
          if(this->GetSegment(thisSegmentNum)->IsTotalCapture()) {
        int numCapturePairs=0;
        for(int i = 1;i<=theCNuc->NumPairs();i++) {
          if(theCNuc->GetPair(i)->GetPType()==10) {
            numCapturePairs++;
            if(numCapturePairs==1) {
              this->GetSegment(thisSegmentNum)->SetExitKey(theCNuc->GetPair(i)->GetPairKey());
            } else {
              ESegment newSegment(*this->GetSegment(thisSegmentNum));
              newSegment.SetExitKey(theCNuc->GetPair(i)->GetPairKey());
              newSegment.SetIsTotalCapture(0);
              newSegment.SetVaryNorm(false);
              this->AddSegment(newSegment);
            }
          }
        }
        this->GetSegment(thisSegmentNum)->SetIsTotalCapture(numCapturePairs);
          }
        }
      } else {
        if(NewSegment.GetExitKey()==-1) {
          configure.outStream « "WARNING: Total capture specified but no capture pair exists."
                  « std::endl;
        } else {
```

```
          configure.outStream « "WARNING: Pair key " « NewSegment.GetExitKey()
                  « " not in compound nucleus." « std::endl;
        }
      }
    } else configure.outStream « "WARNING: Pair key " « NewSegment.GetEntranceKey()
              « " not in compound nucleus." « std::endl;
      }
    }
  }

  if(line!="</segmentsData>") return -1;

  in.close();

  if(this->NumSegments()>0) {
    if(this->ReadTargetEffectsFile(configure,theCNuc)==-1) return -1;
    this->MapData();
  }

  return 0;
}

int EData::MakePoints(const Config& configure, CNuc *theCNuc) {
  std::ifstream in(configure.configfile.c_str());
  if(!in) return -1;
  std::string line = "";
  while(line!="<segmentsTest>"&&!in.eof()) getline(in,line);
  if(line!="<segmentsTest>") return -1;
  line="";
  int numTotalSegments=0;
  while(!in.eof()&&line!="</segmentsTest>") {
    getline(in,line);
    bool empty=true;
    for(unsigned int i=0;i<line.size();++i)
      if(line[i]!=' '&&line[i]!='\t') {
    empty=false;
    break;
      }
    if(empty==true) continue;
    if(!in.eof()&&line!="</segmentsTest>") {
      std::istringstream stm;
      stm.str(line);
      ExtrapLine segment(stm);
      if(stm.rdstate() & (std::stringstream::failbit | std::stringstream::badbit)) return -1;
      numTotalSegments++;
      if(segment.isActive()==1) {
    ESegment NewSegment(segment);
    if(theCNuc->IsPairKey(NewSegment.GetEntranceKey())) {
      bool isValidTotal=false;
      if(NewSegment.GetExitKey()==-1) {
        for(int i = 1;i<=theCNuc->NumPairs();i++) {
          if(theCNuc->GetPair(i)->GetPType()==10) {
        isValidTotal = true;
        break;
          }
        }
      }
      if(isValidTotal||theCNuc->IsPairKey(NewSegment.GetExitKey())) {
        NewSegment.SetSegmentKey(numTotalSegments);
        this->AddSegment(NewSegment);
        ESegment *theSegment=this->GetSegment(this->NumSegments());
        theCNuc->GetPair(theCNuc->GetPairNumFromKey(theSegment->GetEntranceKey()))->SetEntrance();
        PPair
*entrancePair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(theSegment->GetEntranceKey()));
        PPair *exitPair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(theSegment->GetExitKey()));
        double aStep=theSegment->GetAStep();
        double eStep=theSegment->GetEStep();
        for(double angle=theSegment->GetMinAngle();
        angle<=theSegment->GetMaxAngle();angle+=aStep) {
          for(double energy=theSegment->GetMinEnergy();
          energy<=theSegment->GetMaxEnergy();energy+=eStep) {
        EPoint NewPoint(angle,energy,theSegment);
        theSegment->AddPoint(NewPoint);
        EPoint *thePoint=theSegment->GetPoint(theSegment->NumPoints());
        thePoint->SetParentData(this);
        if(entrancePair->GetPType()==20) thePoint->ConvertDecayEnergy(exitPair);
        else thePoint->ConvertLabEnergy(entrancePair);
        if(exitPair->GetPType()==0&&theSegment->IsDifferential()&&
           !theSegment->IsPhase()&&!theSegment->IsAngularDist()) {
          if(theSegment->GetEntranceKey()==theSegment->GetExitKey()) {
            thePoint->ConvertLabAngle(entrancePair);
          } else {
            thePoint->ConvertLabAngle(entrancePair,exitPair,configure);
          }
          thePoint->ConvertCrossSection(entrancePair,exitPair);
        }
        if(eStep==0.0) break;
```

```
          }
          if(aStep==0.0) break;
        }
        if(theSegment->NumPoints()==0) {
          configure.outStream « "WARNING: Extrapolation segment #" « numTotalSegments
                  « " is empty and will not be used." « std::endl;
          this->DeleteLastSegment();
        } else {
          int thisSegmentNum = this->NumSegments();
          if(this->GetSegment(thisSegmentNum)->IsTotalCapture()) {
        int numCapturePairs=0;
        for(int i = 1;i<=theCNuc->NumPairs();i++) {
          if(theCNuc->GetPair(i)->GetPType()==10) {
            numCapturePairs++;
            if(numCapturePairs==1) {
              this->GetSegment(thisSegmentNum)->SetExitKey(theCNuc->GetPair(i)->GetPairKey());
            } else {
              ESegment newSegment(*this->GetSegment(thisSegmentNum));
              newSegment.SetExitKey(theCNuc->GetPair(i)->GetPairKey());
              newSegment.SetIsTotalCapture(0);
              newSegment.SetVaryNorm(false);
              this->AddSegment(newSegment);
            }
          }
        }
        this->GetSegment(thisSegmentNum)->SetIsTotalCapture(numCapturePairs);
          }
        }
      } else {
        if(NewSegment.GetExitKey()==-1) {
          configure.outStream « "WARNING: Total capture specified but no capture pair exists."
                  « std::endl;
        } else {
          configure.outStream « "WARNING: Pair key " « NewSegment.GetExitKey()
                  « " not in compound nucleus." « std::endl;
        }
      }
    } else configure.outStream « "WARNING: Pair key " « NewSegment.GetEntranceKey()
                   « " not in compound nucleus." « std::endl;
      }
    }
  }

  if(line!="</segmentsTest>") return -1;

  in.close();

  if(this->NumSegments()>0) {
    if(this->ReadTargetEffectsFile(configure,theCNuc)==-1) return -1;
    this->MapData();
  }

  return 0;
}

int EData::Iterations() const {
  return iterations_;
}

int EData::NumTargetEffects() const {
  return targetEffects_.size();
}

int EData::GetNormParamOffset() const {
  return normParamOffset_;
}

int EData::ReadTargetEffectsFile(const Config& configure, CNuc *compound) {
  std::ifstream in(configure.configfile.c_str());
  if(!in) return -1;
  std::string line="";
  while(line!="<targetInt>"&&!in.eof()) getline(in,line);
  if(line!="<targetInt>") return -1;
  line="";
  while(line!="</targetInt>"&&!in.eof()) {
    getline(in,line);
    bool empty=true;
    for(unsigned int i=0;i<line.size();++i)
      if(line[i]!=' '&&line[i]!='\t') {
    empty=false;
    break;
      }
    if(empty==true) continue;
    if(line!="</targetInt>"&&!in.eof()){
      std::istringstream stm;
      stm.str(line);
      TargetEffect targetEffect(stm,configure);
```

```
      if(stm.rdstate() & (std::stringstream::failbit | std::stringstream::badbit)) return -1;
      if(targetEffect.IsActive()) {
    this->AddTargetEffect(targetEffect);
    TargetEffect *thisTargetEffect=this->GetTargetEffect(this->NumTargetEffects());
    std::vector<int> segmentsList = thisTargetEffect->GetSegmentsList();
    for(int i = 1;i<=segmentsList.size();i++) {
      if(this->IsSegmentKey(segmentsList[i-1]))
        this->GetSegmentFromKey(segmentsList[i-1])->SetTargetEffectNum(this->NumTargetEffects());
    }
      }
    }
  }
  if(line!="</targetInt>") return -1;
  for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
    PPair *entrancePair = compound->GetPair(compound->GetPairNumFromKey(segment->GetEntranceKey()));
    PPair *exitPair = compound->GetPair(compound->GetPairNumFromKey(segment->GetExitKey()));
    double cmConversion;
    if(entrancePair->GetPType()==20)
      cmConversion = (exitPair->GetM(1)+exitPair->GetM(2))/exitPair->GetM(2);
    else
      cmConversion = entrancePair->GetM(2)/(entrancePair->GetM(1)+entrancePair->GetM(2));
    if(segment->IsTargetEffect()) {
      TargetEffect *targetEffect = this->GetTargetEffect(segment->GetTargetEffectNum());
      double sigma = targetEffect->GetSigma();
      targetEffect->SetSigma(cmConversion*sigma);
      for(EPointIterator point=segment->GetPoints().begin();point<segment->GetPoints().end();point++)
{
    point->SetTargetEffectNum(segment->GetTargetEffectNum());
    if(targetEffect->IsTargetIntegration()||targetEffect->IsConvolution()) {
      double forwardDepth=0.0;
      double backwardDepth=0.0;
      if(targetEffect->IsTargetIntegration()) {
        double totalM=entrancePair->GetM(1)+entrancePair->GetM(2);
        double targetThickness =
cmConversion*targetEffect->TargetThickness(point->GetLabEnergy(),configure);
        point->SetTargetThickness(targetThickness);
        if(targetEffect->IsConvolution()) {
          backwardDepth=targetThickness+targetEffect->convolutionRange*targetEffect->GetSigma();
          forwardDepth=targetEffect->convolutionRange*targetEffect->GetSigma();
        } else {
          backwardDepth=targetThickness;
          forwardDepth=0.0;
        }
      } else if(targetEffect->IsConvolution()) {
        backwardDepth=targetEffect->convolutionRange*targetEffect->GetSigma();
        forwardDepth=targetEffect->convolutionRange*targetEffect->GetSigma();
      }
      for(int i=0;i<targetEffect->NumSubPoints();i++) {
        double subEnergy=point->GetCMEnergy()+forwardDepth
          -(forwardDepth+backwardDepth)/(targetEffect->NumSubPoints())*i;
        EPoint subPoint(point->GetCMAngle(),subEnergy,&*segment);
        if(targetEffect->IsTargetIntegration()) {
          double stoppingPower=cmConversion*targetEffect->
        GetStoppingPowerEq()->Evaluate(configure,subEnergy/cmConversion);
          subPoint.SetStoppingPower(stoppingPower);
        }
        point->AddSubPoint(subPoint);
      }
    }
      }
    }
  }
  return 0;
}

bool EData::IsFit() const {
  return isFit_;
}

bool EData::IsErrorAnalysis() const {
  return isErrorAnalysis_;
}

bool EData::IsSegmentKey(int segmentKey) {
  bool isKey=false;
  for (ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
    if(segment->GetSegmentKey()==segmentKey) {
      isKey=true;
      break;
    }
  }
  return isKey;
}

void EData::SetFit(bool fit) {
  isFit_=fit;
}
```

```
void EData::SetErrorAnalysis(bool errorAnalysis) {
  isErrorAnalysis_=errorAnalysis;
}

void EData::Iterate(){
  iterations_++;
}

void EData::ResetIterations(){
  iterations_=0;
}

int EData::Initialize(CNuc *compound,const Config &configure) {
  //Calculate channel lo-matrix and channel penetrability for each channel at each local energy
  configure.outStream « "Calculating Lo-Matrix, Phases, and Penetrabilities..." « std::endl;
  if(this->CalcEDependentValues(compound,configure)==-1) return -1;
  if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_ENERGY_DEP)
    this->PrintEDependentValues(configure,compound);

  //Calculate legendre polynomials for each data point
  configure.outStream « "Calculating Legendre Polynomials..." « std::endl;
  this->CalcLegendreP(configure.maxLOrder);
  if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_LEGENDRE)
    this->PrintLegendreP(configure);

  //Calculate Coulomb Amplitudes
  configure.outStream « "Calculating Coulomb Amplitudes..." « std::endl;
  this->CalcCoulombAmplitude(compound);
  if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_COUL_AMPLITUDES) {
    this->PrintCoulombAmplitude(configure,compound);
  }

  //Calculate new ec amplitudes
  if(configure.paramMask & Config::USE_EXTERNAL_CAPTURE) {
    configure.outStream « "Calculating External Capture Amplitudes..." « std::endl;
    if(this->CalculateECAmplitudes(compound,configure)==-1) return -1;
  }
  return 0;
}

void EData::AddSegment(ESegment segment) {
  segments_.push_back(segment);
}

void EData::PrintData(const Config &configure) {
  std::streambuf *sbuffer;
  std::filebuf fbuffer;
  if(configure.fileCheckMask & Config::CHECK_DATA) {
    std::string outfile=configure.checkdir+"data.chk";
    fbuffer.open(outfile.c_str(),std::ios::out);
    sbuffer = &fbuffer;
  } else if(configure.screenCheckMask & Config::CHECK_DATA) sbuffer = configure.outStream.rdbuf();
  std::ostream out(sbuffer);
  if(((configure.fileCheckMask & Config::CHECK_DATA)&&fbuffer.is_open())||
     (configure.screenCheckMask & Config::CHECK_DATA)) {
    out « std::endl
    « "************************************" « std::endl
    « "*             Segments             *" « std::endl
    « "************************************" « std::endl;
    out « std::setw(11) « "Segment #"
    « std::setw(17) « "Segment Key #"
    « std::setw(17) « "Entrance Key #"
    « std::setw(13) « "Exit Key #"
    « std::setw(12) « "Min Energy"
    « std::setw(12) « "Max Energy"
    « std::setw(11) « "Min Angle"
    « std::setw(11) « "Max Angle"
    « std::setw(25) « "Data File"
    « std::endl;
    for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
      out « std::setw(11) « segment-GetSegments().begin()+1
      « std::setw(17) « segment->GetSegmentKey()
      « std::setw(17) « segment->GetEntranceKey()
      « std::setw(13) « segment->GetExitKey()
      « std::setw(12) « segment->GetMinEnergy()
      « std::setw(12) « segment->GetMaxEnergy()
      « std::setw(11) « segment->GetMinAngle()
      « std::setw(11) « segment->GetMaxAngle()
      « std::setw(25) « segment->GetDataFile()
      « std::endl;
    }
    out « std::endl
    « "************************************" « std::endl
    « "*               Data               *" « std::endl
    « "************************************" « std::endl;
    out « std::setw(11) « "Segment #"
```

```
    « std::setw(14) « "Data Point #"
    « std::setw(15) « "Lab Energy"
    « std::setw(15) « "CM Energy"
    « std::setw(15) « "Angle"
    « std::setw(20) « "Cross Section"
    « std::setw(22) « "Cross Section Error"
        « std::setw(12) « "Map Point"
    « std::setw(18) « "# of Subpoints"
    « std::setw(18) « "Low Sub Energy"
    « std::setw(18) « "High Sub Energy"
    « std::endl;
    for(EDataIterator data=begin();data!=end();data++) {
      out « std::setw(11) « data.segment()-GetSegments().begin()+1
      « std::setw(14) « data.point()-(data.segment()->GetPoints()).begin()+1
      « std::setw(15) « data.point()->GetLabEnergy()
      « std::setw(15) « data.point()->GetCMEnergy()
      « std::setw(15) « data.point()->GetCMAngle()
      « std::setw(20) « data.point()->GetCMCrossSection()
      « std::setw(22) « data.point()->GetCMCrossSectionError();
      if(data.point()->IsMapped()){
    EnergyMap map=data.point()->GetMap();
    char tempMap[25];
    sprintf(tempMap,"(%d,%d)",map.segment,map.point);
    out « std::setw(12) «  tempMap « std::endl;
      } else
    out « std::setw(12) « "Not Mapped"
        « std::setw(18) « data.point()->NumSubPoints();
      if(data.point()->IsTargetEffect()&&
 (data.point()->GetParentData()->GetTargetEffect(data.point()->GetTargetEffectNum())->IsConvolution()||
 data.point()->GetParentData()->GetTargetEffect(data.point()->GetTargetEffectNum())->IsTargetIntegration()))
 {
    out « std::setw(18) « data.point()->GetSubPoint(data.point()->NumSubPoints())->GetCMEnergy()
        « std::setw(18) « data.point()->GetSubPoint(1)->GetCMEnergy();
      }
      out « std::endl;
      if(data.point()==data.segment()->GetPoints().end()-1) out « std::endl;
    }
  } else configure.outStream « "Could not write data check file." « std::endl;
  out.flush();
  if(fbuffer.is_open()) fbuffer.close();
}

void EData::CalcLegendreP(int maxL) {
  for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
    TargetEffect *effect = (segment->IsTargetEffect() &&
                this->GetTargetEffect(segment->GetTargetEffectNum())->IsQCoefficients()) ?
      this->GetTargetEffect(segment->GetTargetEffectNum()) : NULL;
#pragma omp parallel for
    for(int i=1;i<=segment->NumPoints();i++) {
      EPoint* point=segment->GetPoint(i);
      point->CalcLegendreP(maxL, effect);
    }
  }
}

void EData::PrintLegendreP(const Config &configure) {
  std::streambuf *sbuffer;
  std::filebuf fbuffer;
  if(configure.fileCheckMask & Config::CHECK_LEGENDRE) {
    std::string outfile=configure.checkdir+"legendre.chk";
    fbuffer.open(outfile.c_str(),std::ios::out);
    sbuffer = &fbuffer;
  } else if(configure.screenCheckMask & Config::CHECK_LEGENDRE) sbuffer = configure.outStream.rdbuf();
  std::ostream out(sbuffer);
  if(((configure.fileCheckMask & Config::CHECK_LEGENDRE)&&fbuffer.is_open())||
     (configure.screenCheckMask & Config::CHECK_LEGENDRE)) {
    out « std::endl
    « "************************************" « std::endl
    « "*        Legendre Polynomials      *" « std::endl
    « "************************************" « std::endl;
    out « std::setw(10) « "Segment #"
    « std::setw(10) « "Point #"
    « std::setw(15) « "CM Energy"
    « std::setw(15) « "Angle"
    « std::setw(5)  « "L"
    « std::setw(15) « "Leg. Poly." « std::endl;
    for(EDataIterator data=begin();data!=end();data++) {
      for(int lOrder=0;lOrder<=data.point()->GetMaxLOrder();lOrder++) {
    out « std::setw(10) « data.segment()-GetSegments().begin()+1
        « std::setw(10) « data.point()-(data.segment()->GetPoints()).begin()+1
        « std::setw(15) « data.point()->GetCMEnergy()
        « std::setw(15) « data.point()->GetCMAngle()
        « std::setw(5)  « lOrder
        « std::setw(15) « data.point()->GetLegendreP(lOrder) « std::endl;
      }
```

```
    }
  } else configure.outStream « "Could not write legendre polynomials check file." « std::endl;
  out.flush();
  if(fbuffer.is_open()) fbuffer.close();
}

int EData::CalcEDependentValues(CNuc *theCNuc,const Config& configure) {
  for(ESegmentIterator segment=GetSegments().begin(); segment<GetSegments().end(); segment++) {
    bool localStop = false;
#pragma omp parallel for shared(localStop,configure)
    for(int i=1;i<=segment->NumPoints();i++) {
      if(configure.stopFlag||localStop) continue;
      EPoint *point = segment->GetPoint(i);
      if(!(point->IsMapped())){
    try {
      point->CalcEDependentValues(theCNuc,configure);
    } catch(GSLException e) {
#pragma omp critical
      {
        configure.outStream « e.what() « std::endl;
        localStop = true;
      }
    }
      }
    }
    if(configure.stopFlag||localStop) return -1;
  }
  return 0;
}

void EData::PrintEDependentValues(const Config &configure,CNuc *theCNuc) {
  std::streambuf *sbuffer;
  std::filebuf fbuffer;
  if(configure.fileCheckMask & Config::CHECK_ENERGY_DEP) {
    std::string outfile=configure.checkdir+"lomatrixandpene.chk";
    fbuffer.open(outfile.c_str(),std::ios::out);
    sbuffer = &fbuffer;
  } else if(configure.screenCheckMask & Config::CHECK_ENERGY_DEP) sbuffer =
configure.outStream.rdbuf();
  std::ostream out(sbuffer);
  if(((configure.fileCheckMask & Config::CHECK_ENERGY_DEP)&&fbuffer.is_open())||
     (configure.screenCheckMask & Config::CHECK_ENERGY_DEP)) {
    out « std::endl
    « "************************************" « std::endl
    « "*  Lo Matrix and Penetrabilities   *" « std::endl
    « "************************************" « std::endl;
    out « std::setw(10) « "Seg #"
    « std::setw(10) « "Point #"
    « std::setw(5)  « "j"
    « std::setw(5)  « "ch"
    « std::setw(5)  « "l"
    « std::setw(15) « "E chan"
    « std::setw(15) « "pene"
    « std::setw(25) « "Lo" « std::endl;
    for(EDataIterator data=begin();data!=end();data++) {
      double inEnergy=data.point()->GetCMEnergy()
        +theCNuc->GetPair(theCNuc->GetPairNumFromKey(data.segment()->GetEntranceKey()))->GetSepE();
      for(int j=1;j<=theCNuc->NumJGroups();j++) {
    if(theCNuc->GetJGroup(j)->IsInRMatrix()) {
      JGroup *theJGroup=theCNuc->GetJGroup(j);
      for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
        AChannel *theChannel=theJGroup->GetChannel(ch);
        PPair *thePair=theCNuc->GetPair(theChannel->GetPairNum());
        int lValue=theChannel->GetL();
        double localEnergy=inEnergy-thePair->GetSepE()-thePair->GetExE();
        out « std::setw(10) « data.segment()-GetSegments().begin()+1
        « std::setw(10) « data.point()-(data.segment()->GetPoints()).begin()+1
        « std::setw(5) « j
        « std::setw(5) « ch
        « std::setw(5) « lValue
        « std::setw(15) « localEnergy
        « std::setw(15) « data.point()->GetSqrtPenetrability(j,ch)
        « std::setw(25) « data.point()->GetLoElement(j,ch) « std::endl;
      }
    }
      }
    }
  } else configure.outStream « "Could not write lo-matrix and penetrabilities check file." «
std::endl;
  out.flush();
  if(fbuffer.is_open()) fbuffer.close();
}

void EData::CalcCoulombAmplitude(CNuc *theCNuc) {
  for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
#pragma omp parallel for
    for(int i=1;i<=segment->NumPoints();i++) {
```

```
      EPoint* point = segment->GetPoint(i);
      point->CalcCoulombAmplitude(theCNuc);
    }
  }
}

void EData::PrintCoulombAmplitude(const Config &configure,CNuc *theCNuc) {
  std::streambuf *sbuffer;
  std::filebuf fbuffer;
  if(configure.fileCheckMask & Config::CHECK_COUL_AMPLITUDES) {
    std::string outfile=configure.checkdir+"coulombamplitudes.chk";
    fbuffer.open(outfile.c_str(),std::ios::out);
    sbuffer = &fbuffer;
  } else if(configure.screenCheckMask & Config::CHECK_COUL_AMPLITUDES) sbuffer = configure.outStream.rdbuf();
  std::ostream out(sbuffer);
  if(((configure.fileCheckMask & Config::CHECK_COUL_AMPLITUDES)&&fbuffer.is_open())||
     (configure.screenCheckMask & Config::CHECK_COUL_AMPLITUDES)) {
    out « std::endl
    « "**************************************" « std::endl
    « "*          Coulomb Amplitudes         *" « std::endl
    « "**************************************" « std::endl;
    out « std::setw(10) « "segment #"
    « std::setw(10) « "point #"
    « std::setw(10) « "aa"
    « std::setw(15) « "cmenergy"
    « std::setw(15) « "angle"
    « std::setw(25) « "coulomb amplitude"
    « std::endl;
    for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
      if(segment->GetEntranceKey()==segment->GetExitKey()) {
    for(EPointIterator point=segment->GetPoints().begin();point<segment->GetPoints().end();point++) {
      out « std::setw(10) « segment-GetSegments().begin()+1
          « std::setw(10) « point-segment->GetPoints().begin()+1
          « std::setw(10) « theCNuc->GetPairNumFromKey(segment->GetEntranceKey())
          « std::setw(15) « point->GetCMEnergy()
          « std::setw(15) « point->GetCMAngle()
          « std::setw(25) « point->GetCoulombAmplitude()
          « std::endl;
    }
      }
    }
  } else configure.outStream « "Could not write coulomb amplitudes check file." « std::endl;
  out.flush();
  if(fbuffer.is_open()) fbuffer.close();
}

void EData::WriteOutputFiles(const Config &configure, bool isFit) {
  AZUREOutput output(configure.outputdir);
  std::ofstream chiOut;
  if(!isFit&&(configure.paramMask & Config::CALCULATE_WITH_DATA)) {
    std::string chiOutFile = configure.outputdir+"chiSquared.out";
    chiOut.open(chiOutFile.c_str());
  }
  if(!(configure.paramMask & Config::CALCULATE_WITH_DATA)) output.SetExtrap();
  bool isVaryNorm=false;
  double totalChiSquared=0.;
  ESegmentIterator firstSumIterator = GetSegments().end();
  for(ESegmentIterator segment=GetSegments().begin();
      segment<GetSegments().end();segment++) {
    if(segment->IsTotalCapture()) {
      firstSumIterator=segment;
      segment+=segment->IsTotalCapture()-1;
    }
    if(segment->IsVaryNorm()) isVaryNorm=true;
    int aa=segment->GetEntranceKey();
    int ir=segment->GetExitKey();
    std::filebuf* buf;
    if(firstSumIterator!=GetSegments().end()) buf=output(aa,-1);
    else {
      if(segment->IsAngularDist()&&
     !(configure.paramMask & Config::CALCULATE_WITH_DATA)) buf=output(aa,ir,true);
      else buf=output(aa,ir);
    }
    std::ostream out(buf);
    ESegmentIterator thisSegment = segment;
    if(firstSumIterator!=GetSegments().end()) thisSegment = firstSumIterator;
    for(EPointIterator point=segment->GetPoints().begin();point<segment->GetPoints().end();point++) {
      out.precision(6);
      if(segment->IsAngularDist()) {
    out « std::setw(15) « std::scientific « point->GetCMEnergy();
    for(int i = 0;i<point->GetNumAngularDists();i++) out « std::setw(15) « point->GetAngularDist(i);
    out « std::endl;
      } else {
    double fitCrossSection=point->GetFitCrossSection();
    if(firstSumIterator!=GetSegments().end()) {
      int pointIndex=point-segment->GetPoints().begin()+1;
```

```
      for(ESegmentIterator it=firstSumIterator;it&lt;segment;it++)
        fitCrossSection+=it->GetPoint(pointIndex)->GetFitCrossSection();
    }
    out « std::setw(15) « std::scientific « point->GetCMEnergy()
        « std::setw(15) « std::scientific « point->GetExcitationEnergy()
        « std::setw(15) « std::scientific « point->GetCMAngle()
        « std::setw(15) « std::scientific « fitCrossSection
        « std::setw(15) « std::scientific « fitCrossSection*point->GetSFactorConversion();
    if(!output.IsExtrap()) {
      double dataNorm=thisSegment->GetNorm();
      out « std::setw(15) « std::scientific « point->GetCMCrossSection()*dataNorm
          « std::setw(15) « std::scientific « point->GetCMCrossSectionError()*dataNorm
          « std::setw(15) « std::scientific «
point->GetCMCrossSection()*dataNorm*point->GetSFactorConversion()
          « std::setw(15) « std::scientific «
point->GetCMCrossSectionError()*dataNorm*point->GetSFactorConversion()
          « std::endl;
    } else out « std::endl;
      }
    }
    if(!isFit&&(configure.paramMask & Config::CALCULATE_WITH_DATA)) {
      totalChiSquared+=thisSegment->GetSegmentChiSquared();
      chiOut « "Segment #"
         « thisSegment->GetSegmentKey()
         « " Chi-Squared/N: "
         « thisSegment->GetSegmentChiSquared()/thisSegment->NumPoints()
         « std::endl;
    }
    out«std::endl«std::endl;out.flush();
    firstSumIterator=GetSegments().end();
  }
  if(!isFit&&(configure.paramMask & Config::CALCULATE_WITH_DATA)) {
    chiOut « "Total Chi-Squared: "
          « totalChiSquared « std::endl « std::endl;
    chiOut.flush();chiOut.close();
  }
  if(isVaryNorm) {
    std::string outputfile=configure.outputdir+"normalizations.out";
    std::ofstream out(outputfile.c_str());
    if(out) {
      out.precision(6);
      out « std::scientific;
      for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
    if(segment->IsVaryNorm()) out « std::setw(20) « "Segment Key #" « segment->GetSegmentKey()
                      « std::setw(20) « segment->GetNorm() « std::endl;
      }
      out.flush();
      out.close();
    } else configure.outStream « "Could not write normalization file." « std::endl;
  }
}

int EData::CalculateECAmplitudes(CNuc *theCNuc,const Config& configure) {
  std::ifstream in;
  std::ofstream out;
  std::string outputfile;
  if(configure.paramMask & Config::CALCULATE_WITH_DATA) outputfile=configure.outputdir+"intEC.dat";
  else outputfile=configure.outputdir+"intEC.extrap";
  if(configure.paramMask & Config::USE_PREVIOUS_INTEGRALS) in.open(configure.integralsfile.c_str());
  else {
    out.open(outputfile.c_str());
    if(!out) configure.outStream « "Could not write to EC Amplitude File." « std::endl;
  }
  int sumSegmentI=0;
  int numSumSegments=0;
  for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
    if(segment->IsTotalCapture()) {
      numSumSegments = segment->IsTotalCapture();
      sumSegmentI=0;
    }
    if(numSumSegments) sumSegmentI++;
    char segmentKeyOut[256];
    if(numSumSegments) sprintf(segmentKeyOut,"%d
(%d/%d)",segment->GetSegmentKey(),sumSegmentI,numSumSegments);
    else sprintf(segmentKeyOut,"%d",segment->GetSegmentKey());
    int aa=theCNuc->GetPairNumFromKey(segment->GetEntranceKey());
    if(theCNuc->GetPair(aa)->GetPType()==20) continue;
    if(theCNuc->GetPair(aa)->IsEntrance()) {
      PPair *entrancePair=theCNuc->GetPair(aa);
      for(int j=1;j<=theCNuc->NumJGroups();j++) {
    for(int la=1;la<=theCNuc->GetJGroup(j)->NumLevels();la++) {
      if(theCNuc->GetJGroup(j)->GetLevel(la)->IsECLevel()) {
        ALevel *ecLevel = theCNuc->GetJGroup(j)->GetLevel(la);
        int ir=theCNuc->GetPairNumFromKey(segment->GetExitKey());
        if(ecLevel->GetECPairNum()==ir) {
          if(!(configure.paramMask & Config::USE_PREVIOUS_INTEGRALS)) {
        configure.outStream « "\tSegment #" « std::setw(12) « segmentKeyOut
```

```
                  « std::setw(0) « " [                         ] 0%";configure.outStream.flush();
        int numPoints=segment->NumPoints();
        int pointIndex=0;
        time_t startTime = time(NULL);
        bool localStop = false;
#pragma omp parallel for shared(configure,localStop)
        for(int i=1;i<=numPoints;i++) {
          if(configure.stopFlag||localStop) continue;
          EPoint *point = segment->GetPoint(i);
          if(!(point->IsMapped())) {
            try {
              point->CalculateECAmplitudes(theCNuc,configure);
            } catch(GSLException e) {
#pragma omp critical
              {
            configure.outStream « e.what() « std::endl;
            localStop=true;
              }
            }
          }
          ++pointIndex;
          if(difftime(time(NULL),startTime)>0.25) {
            startTime=time(NULL);
            std::string progress=" [";
            double percent=0.;
            for(int j = 1;j<=25;j++) {
              if(pointIndex>=percent*numPoints&&percent<1.) {
            percent+=0.04;
            progress+='*';
              } else progress+=' ';
            } progress+="] ";
            configure.outStream « "\r\tSegment #" « std::setw(12) « segmentKeyOut
                    « std::setw(0) « progress « percent*100 « '%';configure.outStream.flush();
          }
        }
        if(configure.stopFlag||localStop) {
          if(out.is_open()) out.close();
          if(in.is_open()) in.close();
          return -1;
        }
        configure.outStream « "\r\tSegment #" « std::setw(12) « segmentKeyOut
                    « std::setw(0) « " [*************************] 100%" « std::endl;
          }
          for(EPointIterator point=segment->GetPoints().begin();
          point<segment->GetPoints().end();point++) {
        if(!(point->IsMapped())) {
          for(int k=1;k<=entrancePair->GetDecay(ir)->NumKGroups();k++) {
            for(int ecm=1;ecm<=entrancePair->GetDecay(ir)->GetKGroup(k)->NumECMGroups();ecm++) {
              if(!(configure.paramMask & Config::USE_PREVIOUS_INTEGRALS)) {
            if(out.is_open()) out « point->GetECAmplitude(k,ecm) « std::endl;
            for(EPointIterator subPoint=point->GetSubPoints().begin();
                subPoint<point->GetSubPoints().end();subPoint++)
              if(out.is_open()) out « subPoint->GetECAmplitude(k,ecm) « std::endl;
              } else {
            complex ecAmplitude(0.0,0.0);
            in » ecAmplitude;
            point->AddECAmplitude(k,ecm,ecAmplitude);
            for(EPointIterator subPoint=point->GetSubPoints().begin();
                subPoint<point->GetSubPoints().end();subPoint++) {
              ecAmplitude=complex(0.0,0.0);
              in » ecAmplitude;
              subPoint->AddECAmplitude(k,ecm,ecAmplitude);
            }
              }
              for(EPointMapIterator mappedPoint=point->GetMappedPoints().begin();
              mappedPoint<point->GetMappedPoints().end();mappedPoint++) {
            (*mappedPoint)->AddECAmplitude(k,ecm,point->GetECAmplitude(k,ecm));
            for(int i=1;i<=point->NumSubPoints();i++) {
              (*mappedPoint)->GetSubPoint(i)->
                AddECAmplitude(k,ecm,point->GetSubPoint(i)->GetECAmplitude(k,ecm));
            }
              }
            }
          }
        }
          }
        }
      }
    }
      }
    }
    if(sumSegmentI==numSumSegments) {
      sumSegmentI=0;
      numSumSegments=0;
    }
  }
  if(out.is_open()) {
```

```
    out.flush();
    out.close();
  }
  if(in.is_open()) in.close();
  return 0;
}

void EData::MapData() {
  for(ESegmentIterator segment=GetSegments().end()-1;
      segment>=GetSegments().begin(); segment--) {
    for(EPointIterator point=segment->GetPoints().end()-1;
    point>=segment->GetPoints().begin();point--) {
      if(point->NumLocalMappedPoints()==0) {
    for(ESegmentIterator testSegment=GetSegments().begin();
        testSegment<GetSegments().end(); testSegment++) {
      if(testSegment->GetEntranceKey()==segment->GetEntranceKey()&&
         testSegment->GetExitKey()==segment->GetExitKey()) {
        for(EPointIterator testPoint=testSegment->GetPoints().begin();
        testPoint<testSegment->GetPoints().end();testPoint++) {
          if(testPoint->GetCMEnergy()==point->GetCMEnergy()
         &&!testPoint->IsMapped()&&point!=testPoint
         &&testPoint->GetTargetEffectNum()==point->GetTargetEffectNum()) {
        point->SetMap(testSegment-GetSegments().begin()+1,
                  testPoint-testSegment->GetPoints().begin()+1);
        testPoint->AddLocalMappedPoint(&*point);
        break;
          }
        }
        if(point->IsMapped()) break;
      }
    }
      }
    }
  }
}

void EData::AddTargetEffect(TargetEffect targetEffect) {
  targetEffects_.push_back(targetEffect);
}

void EData::SetNormParamOffset(int offset) {
  normParamOffset_=offset;
}

void EData::FillMnParams(ROOT::Minuit2::MnUserParameters &p) {
  SetNormParamOffset(p.Params().size());
  char varname[50];
  for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
    if(segment->IsVaryNorm()) {
      sprintf(varname,"segment_%d_norm",segment->GetSegmentKey());
      p.Add(varname,segment->GetNorm(),segment->GetNorm()*0.05);
    }
    if(segment->IsTotalCapture()) segment+=segment->IsTotalCapture()-1;
  }
}


void EData::DeleteLastSegment() {
  segments_.pop_back();
}

void EData::FillNormsFromParams(const vector_r &p) {
  int i=GetNormParamOffset();
  for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
    if(segment->IsVaryNorm()) {
      segment->SetNorm(p[i]);
      i++;
    }
    if(segment->IsTotalCapture()) segment+=segment->IsTotalCapture()-1;
  }
}

ESegment *EData::GetSegment(int segmentNum) {
  ESegment *b=&segments_[segmentNum-1];
  return b;
}

ESegment *EData::GetSegmentFromKey(int segmentKey) {
  int segmentNumber=1;
  while(segmentNumber <= this->NumSegments()) {
    if(segmentKey==this->GetSegment(segmentNumber)->GetSegmentKey())
      break;
    else segmentNumber++;
  }
  if(segmentNumber<= this->NumSegments()) return this->GetSegment(segmentNumber);
  else return NULL;
}
```

```
EData *EData::Clone() const {
  EData *dataCopy = new EData(*this);

  for(EDataIterator data=dataCopy->begin();data!=dataCopy->end();data++) {
    data.point()->SetParentData(dataCopy);
    data.point()->ClearLocalMappedPoints();
  }
  for(EDataIterator data=dataCopy->begin();data!=dataCopy->end();data++) {
    if(data.point()->IsMapped()) {
      EnergyMap pointMap = data.point()->GetMap();
dataCopy->GetSegment(pointMap.segment)->GetPoint(pointMap.point)->AddLocalMappedPoint(&*data.point());
    }
  }
  return dataCopy;
}

TargetEffect *EData::GetTargetEffect(int effectNumber) {
  TargetEffect *temp;
  if(effectNumber<=targetEffects_.size())
    temp=&targetEffects_[effectNumber-1];
  else return temp=NULL;
  return temp;
}

EDataIterator EData::begin() {
  return EDataIterator(&segments_);
}

EDataIterator EData::end() {
  EDataIterator it(&segments_);
  return it.SetEnd();
}

std::vector<ESegment>& EData::GetSegments() {
  return segments_;
}
```

8.256 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EDataIterator.cpp File Reference

```
#include ¨EDataIterator.h¨
#include ¨ESegment.h¨
```

8.257 EDataIterator.cpp

Go to the documentation of this file.

```
#include "EDataIterator.h"
#include "ESegment.h"


EDataIterator::EDataIterator(std::vector<ESegment>* segments) :
  segments_(segments) {
  segmentIterator_=segments_->begin();
  pointIterator_=(*segmentIterator_).GetPoints().begin();
}

EDataIterator::EDataIterator(const EDataIterator& it) :
  segments_(it.segments_), segmentIterator_(it.segmentIterator_), pointIterator_(it.pointIterator_) {
}

EDataIterator& EDataIterator::operator++() {
  if(pointIterator_ < ((*segmentIterator_).GetPoints().end())-1) {
    pointIterator_++;
  } else if(segmentIterator_ < segments_->end()-1) {
    segmentIterator_++;
    pointIterator_=(*segmentIterator_).GetPoints().begin();
  } else if ((segmentIterator_ == (segments_->end()-1)) &&
         (pointIterator_ == ((*segmentIterator_).GetPoints().end()-1)))
    pointIterator_++;
  return *this;
```

```
}

EDataIterator EDataIterator::operator++(int) {
  EDataIterator temp(*this);
  operator++();
  return temp;
}

bool EDataIterator::operator==(const EDataIterator& rhs) {
  return (rhs.segmentIterator_==segmentIterator_ && rhs.pointIterator_==pointIterator_) ? (true) : (false);
}

bool EDataIterator::operator!=(const EDataIterator& rhs) {
  return (rhs.segmentIterator_==segmentIterator_ && rhs.pointIterator_==pointIterator_) ? (false) : (true);
}

EDataIterator& EDataIterator::SetEnd() {
  segmentIterator_=segments_->end()-1;
  pointIterator_=(*segmentIterator_).GetPoints().end();
  return *this;
}

ESegmentIterator& EDataIterator::segment() {
  return segmentIterator_;
}

EPointIterator& EDataIterator::point() {
  return pointIterator_;
}
```

8.258 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EffectiveCharge.cpp File Reference

```
#include "EffectiveCharge.h"
#include "PPair.h"
#include "Constants.h"
#include <gsl/gsl_sf_bessel.h>
#include <gsl/gsl_integration.h>
```

Functions

- double DoubleFactorial (int)

8.258.1 Function Documentation

8.258.1.1 DoubleFactorial()

```
double DoubleFactorial (
            int a )
```

Definition at line 1 of file DoubleFactorial.cpp.

8.259 EffectiveCharge.cpp

Go to the documentation of this file.

```
#include "EffectiveCharge.h"
#include "PPair.h"
#include "Constants.h"
#include <gsl/gsl_sf_bessel.h>
#include <gsl/gsl_integration.h>

extern double DoubleFactorial(int);

EffectiveCharge::EffectiveCharge(PPair* pair, double energy, int L) :
  z1_(pair->GetZ(1)),z2_(pair->GetZ(2)),L_(L),m1_(pair->GetM(1)),m2_(pair->GetM(2)),
  energy_(energy) {
}

double EffectiveCharge::Integrand(double x, void* p) {
  int* L = (int*)p;
  return gsl_sf_bessel_jl(*L,x)/x;
}

double EffectiveCharge::operator()(double r) {
  gsl_integration_workspace * w
    = gsl_integration_workspace_alloc (1000);
  gsl_function F;
  F.function = &Integrand;
  F.params= &L_;

  double intZ1=0.;
  double intZ2=0.;
  double intError;
  gsl_integration_qags (&F, 0., m2_/(m1_+m2_)*energy_/hbarc*r, 0., 1.e-6, 1000, w, &intZ1, &intError);
  gsl_integration_qags (&F, 0., m1_/(m1_+m2_)*energy_/hbarc*r, 0., 1.e-6, 1000, w, &intZ2, &intError);

  gsl_integration_workspace_free (w);

  return sqrt(fstruc*hbarc)*L_*DoubleFactorial(2*L_+1)/
    pow((energy_/hbarc*r),(double)L_)*(z1_*intZ1+pow(-1,L_)*z2_*intZ2);
}
```

8.260 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EigenFunc.cpp File Reference

```
#include ¨EigenFunc.h¨
#include <gsl/gsl_eigen.h>
#include <math.h>
```

8.261 EigenFunc.cpp

Go to the documentation of this file.

```
#include "EigenFunc.h"
#include <gsl/gsl_eigen.h>
#include <math.h>

EigenFunc::EigenFunc(const matrix_r &A) {
  eigenvalues_.clear();
  eigenvectors_.clear();

  gsl_matrix * m = gsl_matrix_alloc(A.size(),A.size());
  for(int i=0;i<A.size();i++) {
    for(int ii=0;ii<A.size();ii++) {
      gsl_matrix_set (m,i,ii,A[i][ii]);
    }
  }

  gsl_vector * eval = gsl_vector_alloc (A.size());
  gsl_matrix * evec = gsl_matrix_alloc (A.size(), A.size());
  gsl_eigen_symmv_workspace * w = gsl_eigen_symmv_alloc (A.size());

```

```
  gsl_eigen_symmv (m,eval,evec,w);
  gsl_eigen_symmv_sort (eval,evec,GSL_EIGEN_SORT_VAL_ASC);

  for(int i=0;i<A.size();i++) {
    eigenvalues_.push_back(gsl_vector_get(eval,i));
    vector_r tempRow;
    eigenvectors_.push_back(tempRow);
    for(int ii=0;ii<A.size();ii++) {
      eigenvectors_[i].push_back(gsl_matrix_get(evec,i,ii));
    }
  }

  gsl_vector_free (eval);
  gsl_matrix_free (evec);
  gsl_matrix_free (m);
  gsl_eigen_symmv_free (w);

}

EigenFunc::EigenFunc(const matrix_r &A, const std::vector<vector_r > &B) {
  eigenvalues_.clear();
  eigenvectors_.clear();

  gsl_matrix * n = gsl_matrix_alloc(A.size(),A.size());
  gsl_matrix * m = gsl_matrix_alloc(B.size(),B.size());
  for(int i=0;i<A.size();i++) {
    for(int ii=0;ii<A.size();ii++) {
      gsl_matrix_set (n,i,ii,A[i][ii]);
      gsl_matrix_set (m,i,ii,B[i][ii]);
    }
  }

  gsl_vector * eval = gsl_vector_alloc (A.size());
  gsl_matrix * evec = gsl_matrix_alloc (A.size(), A.size());

  gsl_eigen_gensymmv_workspace * w = gsl_eigen_gensymmv_alloc (A.size());

  gsl_eigen_gensymmv(n,m,eval,evec,w);
  gsl_eigen_gensymmv_sort (eval,evec,GSL_EIGEN_SORT_VAL_ASC);

  for(int i=0;i<A.size();i++) {
    eigenvalues_.push_back(gsl_vector_get(eval,i));
    vector_r tempRow;
    eigenvectors_.push_back(tempRow);
    for(int ii=0;ii<A.size();ii++) {
      eigenvectors_[i].push_back(gsl_matrix_get(evec,i,ii));
    }
  }

  vector_r tempNormVec;
  for(int i=0;i<A.size();i++) tempNormVec.push_back(0.0);

  for(int i=0;i<A.size();i++) {
    for(int ii=0;ii<A.size();ii++) {
      double sum=0.0;
      for(int iii=0;iii<A.size();iii++) {
    sum+=B[ii][iii]*eigenvectors_[iii][i];
      }
      tempNormVec[ii]=sum;
    }
    double sum=0.0;
    for(int ii=0;ii<A.size();ii++) {
      sum+=tempNormVec[ii]*eigenvectors_[ii][i];
    }
    for(int ii=0;ii<A.size();ii++) {
      eigenvectors_[ii][i]=eigenvectors_[ii][i]/sqrt(sum);
    }
  }

  gsl_vector_free (eval);
  gsl_matrix_free (evec);
  gsl_matrix_free (m);
  gsl_matrix_free (n);
  gsl_eigen_gensymmv_free (w);
}
```

8.262 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EPoint.cpp File Reference

```
#include "AMatrixFunc.h"
#include "AngCoeff.h"
```

```
#include "CNuc.h"
#include "Config.h"
#include "CoulFunc.h"
#include "DataLine.h"
#include "ECIntegral.h"
#include "EData.h"
#include "ESegment.h"
#include "RMatrixFunc.h"
#include "ShftFunc.h"
#include "TargetEffect.h"
#include "IntegratedFermiFunc.h"
#include <iostream>
#include <assert.h>
```

8.263 EPoint.cpp

Go to the documentation of this file.

```
#include "AMatrixFunc.h"
#include "AngCoeff.h"
#include "CNuc.h"
#include "Config.h"
#include "CoulFunc.h"
#include "DataLine.h"
#include "ECIntegral.h"
#include "EData.h"
#include "ESegment.h"
#include "RMatrixFunc.h"
#include "ShftFunc.h"
#include "TargetEffect.h"
#include "IntegratedFermiFunc.h"
#include <iostream>
#include <assert.h>

EPoint::EPoint(DataLine dataLine, ESegment *parent) {
  entrance_key_=parent->GetEntranceKey();
  exit_key_=parent->GetExitKey();
  cm_angle_=dataLine.angle();
  lab_angle_=dataLine.angle();
  cm_energy_=dataLine.energy();
  lab_energy_=dataLine.energy();
  excitation_energy_=dataLine.energy();
  cm_crosssection_=dataLine.crossSection();
  cm_dcrosssection_=dataLine.error();
  lab_crosssection_=dataLine.crossSection();
  lab_dcrosssection_=dataLine.error();
  geofactor_=0.;
  fitcrosssection_=0.;
  sfactorconv_=0.;
  is_differential_=parent->IsDifferential();
  is_phase_=parent->IsPhase();
  is_ang_dist_=parent->IsAngularDist();
  max_ang_dist_order_=parent->GetMaxAngDistOrder();
  j_value_=parent->GetJ();
  l_value_=parent->GetL();
  is_mapped_=false;
  targetEffectNum_=0;
  parentData_=NULL;
  stoppingPower_=0.0;
}

EPoint::EPoint(double angle, double energy, ESegment* parent) {
  entrance_key_=parent->GetEntranceKey();
  exit_key_=parent->GetExitKey();
  lab_angle_=angle;
  cm_angle_=angle;
  lab_energy_=energy;
  cm_energy_=energy;
  excitation_energy_=energy;
  cm_crosssection_=0.;
  cm_dcrosssection_=0.1;
  lab_crosssection_=0.;
  lab_dcrosssection_=0.1;
  geofactor_=0.;
  fitcrosssection_=0.;
```

```
  sfactorconv_=0.;
  is_differential_=parent->IsDifferential();
  is_phase_=parent->IsPhase();
  is_ang_dist_=parent->IsAngularDist();
  max_ang_dist_order_=parent->GetMaxAngDistOrder();
  j_value_=parent->GetJ();
  l_value_=parent->GetL();
  is_mapped_=false;
  targetEffectNum_=0;
  parentData_=NULL;
  stoppingPower_=0.0;
}

EPoint::EPoint(double angle, double energy, int entranceKey,
           int exitKey, bool isDifferential, bool isPhase, bool isAngularDist, double jValue, int lValue, int maxAngDistOrder) {
  entrance_key_=entranceKey;
  exit_key_=exitKey;
  lab_angle_=angle;
  cm_angle_=angle;
  lab_energy_=energy;
  cm_energy_=energy;
  excitation_energy_=energy;
  cm_crosssection_=0.;
  cm_dcrosssection_=0.1;
  lab_crosssection_=0.;
  lab_dcrosssection_=0.1;
  geofactor_=0.;
  fitcrosssection_=0.;
  sfactorconv_=0.;
  is_differential_=isDifferential;
  is_phase_=isPhase;
  is_ang_dist_=isAngularDist;
  max_ang_dist_order_=maxAngDistOrder;
  j_value_=jValue;
  l_value_=lValue;
  is_mapped_=false;
  targetEffectNum_=0;
  parentData_=NULL;
  stoppingPower_=0.0;
}

bool EPoint::IsDifferential() const {
  return is_differential_;
}

bool EPoint::IsPhase() const {
  return is_phase_;
}

bool EPoint::IsAngularDist() const {
  return is_ang_dist_;
}

bool EPoint::IsMapped() const {
  return is_mapped_;
}

bool EPoint::IsTargetEffect() const {
  if(GetTargetEffectNum()!=0) return true;
  else return false;
}

int EPoint::GetEntranceKey() const {
  return entrance_key_;
}

int EPoint::GetExitKey() const {
  return exit_key_;
}

int EPoint::GetMaxLOrder() const {
  return legendreP_.size()-1;
}

int EPoint::GetL() const {
  return l_value_;
}

int EPoint::NumLocalMappedPoints() const {
  return local_mapped_points_.size();
}

int EPoint::NumSubPoints() const {
  return integrationPoints_.size();
}
```

```
int EPoint::GetTargetEffectNum() const {
  return targetEffectNum_;
}

int EPoint::GetMaxAngDistOrder() const {
  return max_ang_dist_order_;
}

int EPoint::GetNumAngularDists() const {
  return angularDists_.size();
}

double EPoint::GetLabAngle() const {
  return lab_angle_;
}

double EPoint::GetCMAngle() const {
  return cm_angle_;
}

double EPoint::GetLabEnergy() const {
  return lab_energy_;
}

double EPoint::GetCMEnergy() const {
  return cm_energy_;
}

double EPoint::GetExcitationEnergy() const {
  return excitation_energy_;
}


double EPoint::GetLegendreP(int lOrder) const {
  return legendreP_[lOrder];
}

double EPoint::GetLabCrossSection() const {
  return lab_crosssection_;
}

double EPoint::GetCMCrossSection() const {
  return cm_crosssection_;
}

double EPoint::GetLabCrossSectionError() const {
  return lab_dcrosssection_;
}

double EPoint::GetCMCrossSectionError() const {
  return cm_dcrosssection_;
}

double EPoint::GetGeometricalFactor() const {
  return geofactor_;
}

double EPoint::GetFitCrossSection() const {
  return fitcrosssection_;
}

double EPoint::GetSFactorConversion() const {
  return sfactorconv_;
}

double EPoint::GetSqrtPenetrability(int jGroupNum, int channelNum) const {
  return penetrabilities_[jGroupNum-1][channelNum-1];
}

double EPoint::GetJ() const {
  return j_value_;
}

double EPoint::GetStoppingPower() const {
    return stoppingPower_;
}

double EPoint::GetTargetThickness() const {
  return targetThickness_;
}

double EPoint::GetAngularDist(int order) const {
  return angularDists_[order];
}

complex EPoint::GetLoElement(int jGroupNum, int channelNum) const {
  return lo_elements_[jGroupNum-1][channelNum-1];
```

```
}

complex EPoint::GetExpCoulombPhase(int jGroupNum, int channelNum) const {
  return coulombphase_[jGroupNum-1][channelNum-1];
}

complex EPoint::GetExpHardSpherePhase(int jGroupNum, int channelNum) const {
  return hardspherephase_[jGroupNum-1][channelNum-1];
}

complex EPoint::GetCoulombAmplitude() const {
  return coulombamplitude_;
}

complex EPoint::GetECAmplitude(int kGroupNum, int ecMGroupNum) const {
  return ec_amplitudes_[kGroupNum-1][ecMGroupNum-1];
}

EnergyMap EPoint::GetMap() const {
  EnergyMap thisMap;
  if(this->IsMapped()) {
    thisMap.segment=energy_map_.segment;
    thisMap.point=energy_map_.point;
  }
  else {
    thisMap.segment=0;
    thisMap.point=0;
  }
  return thisMap;
}

void EPoint::Initialize(CNuc *compound,const Config &configure) {
  this->CalcEDependentValues(compound,configure);
  if(this->IsDifferential())
    this->CalcLegendreP(configure.maxLOrder,NULL);
  this->CalcCoulombAmplitude(compound);
  if(configure.paramMask & Config::USE_EXTERNAL_CAPTURE) this->CalculateECAmplitudes(compound,configure);
}

void EPoint::ConvertLabEnergy(PPair *pPair) {
  cm_energy_=this->GetLabEnergy()*
    (pPair->GetM(2))/
    (pPair->GetM(1)+pPair->GetM(2));
  excitation_energy_=cm_energy_+pPair->GetSepE();
}

void EPoint::ConvertDecayEnergy(PPair *pPair) {
  cm_energy_=this->GetLabEnergy()/
    (pPair->GetM(2))*
    (pPair->GetM(1)+pPair->GetM(2));
  excitation_energy_=cm_energy_+pPair->GetSepE();
}

void EPoint::ConvertLabAngle(PPair *pPair) {
  cm_angle_=this->GetLabAngle()+180./pi*asin(pPair->GetM(1)/pPair->GetM(2)*sin(pi/180.*this->GetLabAngle()));
}

void EPoint::ConvertLabAngle(PPair *entrancePair, PPair *exitPair, const Config& configure) {
  double qValue=entrancePair->GetSepE()+entrancePair->GetExE()-exitPair->GetSepE()-exitPair->GetExE();
  double a13=(entrancePair->GetM(1)*exitPair->GetM(1))*this->GetLabEnergy()/(this->GetLabEnergy()+qValue)/
    (entrancePair->GetM(1)+entrancePair->GetM(2))/(exitPair->GetM(1)+exitPair->GetM(2));
  double a24=(entrancePair->GetM(2)*exitPair->GetM(2))*(1+entrancePair->GetM(1)/entrancePair->GetM(2)*qValue/(this->GetLabEnergy
    (entrancePair->GetM(1)+entrancePair->GetM(2))/(exitPair->GetM(1)+exitPair->GetM(2));

  if(a13>a24) {
    double thetaMax=asin(sqrt(a24/a13))*180./pi;
    if(thetaMax<this->GetLabAngle()) configure.outStream « std::endl « "Lab Angle (" « this->GetLabAngle()
                            « " degrees) is not kinematically possible.  Maximum angle is "
                            « thetaMax « " degrees." « std::endl;
    assert(thetaMax>=this->GetLabAngle());
  }

  double E3PerEt=a13*pow(cos(this->GetLabAngle()*pi/180.)+sqrt(a24/a13-pow(sin(this->GetLabAngle()*pi/180.),2.0)),2.0);
  double tempE3PerEt = a13*pow(cos((this->GetLabAngle()+0.001)*pi/180.)+sqrt(a24/a13-pow(sin((this->GetLabAngle()+0.001)*pi/180.),2.0)),2.0);
  double slope = sqrt(tempE3PerEt/a24)*sin((this->GetLabAngle()+0.001)*pi/180.)-sqrt(E3PerEt/a24)*sin(this->GetLabAngle()*pi/180.);
  bool switchDomain=false;
  if(slope<0.) switchDomain=true;

  cm_angle_=180./pi*asin(sqrt(E3PerEt/a24)*sin(this->GetLabAngle()*pi/180.));
```

```
  if(switchDomain) cm_angle_ = 180.- cm_angle_;
}

void EPoint::ConvertCrossSection(PPair *entrancePair, PPair *exitPair) {
  double conversionFactor;
  if(this->GetLabAngle()==0.0 || this->GetLabAngle()==180.0) {
    double m1=entrancePair->GetM(1);
    double m2=entrancePair->GetM(2);
    double m3=exitPair->GetM(1);
    double m4=exitPair->GetM(2);
    double e1=this->GetLabEnergy();
    double
qValue=entrancePair->GetSepE()+entrancePair->GetExE()-exitPair->GetSepE()-exitPair->GetExE();
    double et=e1+qValue;
    double a=m1*m4*e1/(m1+m2)/(m3+m4)/et;
    double b=m1*m3*e1/(m1+m2)/(m3+m4)/et;
    double c=m2*m3/(m1+m2)/(m3+m4) * (1+m1*qValue/m2/et);
    double d=m2*m4/(m1+m2)/(m3+m4) * (1+m1*qValue/m2/et);
    double e3et=b+d+2*pow(a*c,0.5)*cos(pi/180.*this->GetCMAngle());
    conversionFactor=pow(a*c,0.5)*pow(d/b-pow(sin(this->GetLabAngle()*pi/180.),2.0),0.5)/e3et;
  }
  else {

conversionFactor=pow(sin(pi/180.*this->GetLabAngle())/sin(pi/180.*this->GetCMAngle()),2.0)*cos(pi/180.*(this->GetCMAngle
  }
  cm_crosssection_=this->GetLabCrossSection()*conversionFactor;
  cm_dcrosssection_=this->GetLabCrossSectionError()*conversionFactor;
}

void EPoint::AddLegendreP(double polynomial) {
  legendreP_.push_back(polynomial);
}

void EPoint::SetGeometricalFactor(double geoFactor) {
  geofactor_=geoFactor;
}

void EPoint::SetFitCrossSection(double crossSection) {
  fitcrosssection_=crossSection;
}

void EPoint::SetSFactorConversion(double conversion) {
  sfactorconv_=conversion;
}

void EPoint::SetExitKey(int key) {
  exit_key_=key;
}

void EPoint::CalcLegendreP(int maxL,TargetEffect* targetEffect) {
  double x=cos(this->GetCMAngle()*pi/180.0);
  if(maxL>=0) {
    if(targetEffect && targetEffect->NumQCoefficients()>0)
      this->AddLegendreP(targetEffect->GetQCoefficient(0));
    else this->AddLegendreP(1.0);
    double polyMinusTwo=1.0;
    if(maxL>=1) {
      if(targetEffect && targetEffect->NumQCoefficients()>1)
    this->AddLegendreP(x*targetEffect->GetQCoefficient(1));
      else this->AddLegendreP(x);
      double polyMinusOne=x;
      if(maxL>=2) {
    for(int lOrder=2;lOrder<=maxL;lOrder++) {
      double poly=(2.0*lOrder-1.0)/lOrder*x*polyMinusOne-
        (lOrder-1.0)/lOrder*polyMinusTwo;
      if(targetEffect && targetEffect->NumQCoefficients()>lOrder)
        this->AddLegendreP(poly*targetEffect->GetQCoefficient(lOrder));
      else this->AddLegendreP(poly);
      polyMinusTwo=polyMinusOne;
      polyMinusOne=poly;
    }
      }
    }
  }
  for(int i=1;i<=this->NumSubPoints();i++) {
    this->GetSubPoint(i)->CalcLegendreP(maxL, targetEffect);
  }
}

void EPoint::CalcEDependentValues(CNuc *theCNuc, const Config& configure) {
  PPair *entrancePair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetEntranceKey()));
  PPair *exitPair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetExitKey()));

  double inEnergy;
  double geofactor;
  double sfactorconv;
  if(theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetEntranceKey()))->GetPType()==20)  {
```

```
    inEnergy=this->GetCMEnergy()+exitPair->GetSepE()+exitPair->GetExE();
    geofactor=1.;
    sfactorconv=1.;
  } else {
    inEnergy=this->GetCMEnergy()+entrancePair->GetSepE()+entrancePair->GetExE();
    geofactor=pi*pow(hbarc,2.)/(2*entrancePair->GetRedMass()*uconv*this->GetCMEnergy());
    sfactorconv=this->GetCMEnergy()*exp(2*pi*sqrt(uconv/2.)*fstruc*entrancePair->GetZ(1)*
                    entrancePair->GetZ(2)*sqrt(entrancePair->GetRedMass()
                                   /this->GetCMEnergy()));
  }
  this->SetGeometricalFactor(geofactor);
  this->SetSFactorConversion(sfactorconv);

  for(int j=1;j<=theCNuc->NumJGroups();j++) {
    if(theCNuc->GetJGroup(j)->IsInRMatrix()) {
      JGroup *theJGroup=theCNuc->GetJGroup(j);
      for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
    AChannel *theChannel=theJGroup->GetChannel(ch);
    PPair *thePair=theCNuc->GetPair(theChannel->GetPairNum());
    int lValue=theChannel->GetL();
    double localEnergy=inEnergy-thePair->GetSepE()-thePair->GetExE();
    if(thePair->GetPType()==0) {
      if(localEnergy<0.0) {
        ShftFunc theShiftFunction(thePair);
        double localShift=theShiftFunction(lValue,inEnergy);
        double boundary=theChannel->GetBoundaryCondition();
        complex loElement(localShift-boundary,0.0);
        this->AddLoElement(j,ch,loElement);
        this->AddSqrtPenetrability(j,ch,0.0);
        this->AddExpCoulombPhase(j,ch,1.0);
        this->AddExpHardSpherePhase(j,ch,1.0);
      } else {
        CoulFunc theCoulombFunction(thePair,!!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
        double radius=thePair->GetChRad();
        double localPene=theCoulombFunction.Penetrability(lValue,radius,localEnergy);
        double localShift=theCoulombFunction.PEShift(lValue,radius,localEnergy);
        double boundary=theChannel->GetBoundaryCondition();
        complex loElement(localShift-boundary,localPene);
        double redmass=thePair->GetRedMass();
        double eta=sqrt(uconv/2.)*fstruc*thePair->GetZ(1)*thePair->GetZ(2)*
          sqrt(redmass/localEnergy);
        complex expCP(1.0,0.0);
        for(int ll=1;ll<=theChannel->GetL();ll++)
          expCP*=complex((double)ll/sqrt(pow(eta,2.0)+pow((double)ll,2.0)),
                 eta/sqrt(pow(eta,2.0)+pow((double)ll,2.0)));
        struct CoulWaves
          coul=theCoulombFunction(lValue,radius,localEnergy);
        complex expHSP(coul.G/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)),
               -coul.F/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)));
        this->AddLoElement(j,ch,loElement);
        this->AddSqrtPenetrability(j,ch,sqrt(localPene));
        this->AddExpCoulombPhase(j,ch,expCP);
        this->AddExpHardSpherePhase(j,ch,expHSP);
      }
    } else if(thePair->GetPType()==10){
      complex loElement = complex(0.0,0.0);
      this->AddLoElement(j,ch,loElement);
      double sqrtPene = (configure.paramMask & Config::USE_RMC_FORMALISM) ? 1. :
pow(localEnergy/hbarc, (double) lValue+0.5);
      this->AddSqrtPenetrability(j,ch,sqrtPene);
      this->AddExpCoulombPhase(j,ch,1.0);
      this->AddExpHardSpherePhase(j,ch,1.0);
    } else if(thePair->GetPType()==20){
      complex loElement = complex(0.0,0.0);
      this->AddLoElement(j,ch,loElement);
      IntegratedFermiFunc fermiFunc(thePair->GetZ(1));
      double endPointE = thePair->GetSepE()-inEnergy;
      double sqrtPene = (1.+endPointE/0.510998903<=1.) ? 0. :
sqrt(fermiFunc(1.+endPointE/0.510998903,exitPair->GetZ(1)+exitPair->GetZ(2),thePair->GetChRad()));
      this->AddSqrtPenetrability(j,ch,sqrtPene);
      this->AddExpCoulombPhase(j,ch,1.0);
      this->AddExpHardSpherePhase(j,ch,1.0);
    }
      }
    }
  }
  for(int i=1;i<=this->NumSubPoints();i++) {
    this->GetSubPoint(i)->CalcEDependentValues(theCNuc,configure);
  }
  for(int i=1;i<=this->NumLocalMappedPoints();i++) {
    EPoint *mappedPoint=this->GetLocalMappedPoint(i);
    mappedPoint->geofactor_=geofactor_;
    mappedPoint->sfactorconv_=sfactorconv_;
    mappedPoint->lo_elements_=lo_elements_;
    mappedPoint->penetrabilities_=penetrabilities_;
    mappedPoint->coulombphase_=coulombphase_;
    mappedPoint->hardspherephase_=hardspherephase_;
```

```
    for(int ii=1;ii<=this->NumSubPoints();ii++) {
      EPoint *subMappedPoint=mappedPoint->GetSubPoint(ii);
      subMappedPoint->geofactor_=this->GetSubPoint(ii)->geofactor_;
      subMappedPoint->sfactorconv_=this->GetSubPoint(ii)->sfactorconv_;
      subMappedPoint->lo_elements_=this->GetSubPoint(ii)->lo_elements_;
      subMappedPoint->penetrabilities_=this->GetSubPoint(ii)->penetrabilities_;
      subMappedPoint->coulombphase_=this->GetSubPoint(ii)->coulombphase_;
      subMappedPoint->hardspherephase_=this->GetSubPoint(ii)->hardspherephase_;
    }
  }
}

void EPoint::AddLoElement(int jGroupNum, int channelNum, complex loElement) {
  vector_c d;
  while(jGroupNum>lo_elements_.size()) lo_elements_.push_back(d);
  lo_elements_[jGroupNum-1].push_back(loElement);
  assert(channelNum=lo_elements_[jGroupNum-1].size());
}

void EPoint::AddSqrtPenetrability(int jGroupNum, int channelNum, double sqrtPene) {
  vector_r d;
  while(jGroupNum>penetrabilities_.size()) penetrabilities_.push_back(d);
  penetrabilities_[jGroupNum-1].push_back(sqrtPene);
  assert(channelNum=penetrabilities_[jGroupNum-1].size());
}

void EPoint::AddExpCoulombPhase(int jGroupNum, int channelNum, complex expShift) {
  vector_c d;
  while(jGroupNum>coulombphase_.size()) coulombphase_.push_back(d);
  coulombphase_[jGroupNum-1].push_back(expShift);
  assert(channelNum=coulombphase_[jGroupNum-1].size());
}

void EPoint::AddExpHardSpherePhase(int jGroupNum, int channelNum, complex expShift) {
  vector_c d;
  while(jGroupNum>hardspherephase_.size()) hardspherephase_.push_back(d);
  hardspherephase_[jGroupNum-1].push_back(expShift);
  assert(channelNum=hardspherephase_[jGroupNum-1].size());
}

void EPoint::CalcCoulombAmplitude(CNuc *theCNuc) {
  if(this->GetEntranceKey()==this->GetExitKey()) {
    PPair *entrancePair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetEntranceKey()));
    int z1=entrancePair->GetZ(1);
    int z2=entrancePair->GetZ(2);
    double redmass=entrancePair->GetRedMass();
    double energy=this->GetCMEnergy();
    double angle=this->GetCMAngle();
    double eta=sqrt(uconv/2.)*fstruc*z1*z2*
      sqrt(redmass/energy);
    double cal=(1.0/(2.0*sqrt(pi)))*eta*(1.0/pow(sin(angle*pi/360.0),2.));
    double cex=2.0*eta*log(sin(angle*pi/360.0));
    complex calpha(cal*cos(cex),-cal*sin(cex));
    this->SetCoulombAmplitude(calpha);
  } else this->SetCoulombAmplitude(complex(0.,0.));
  for(int i=1;i<=this->NumSubPoints();i++) {
    this->GetSubPoint(i)->CalcCoulombAmplitude(theCNuc);
  }
}

void EPoint::SetCoulombAmplitude(complex amplitude) {
  coulombamplitude_=amplitude;
}

void EPoint::CalculateECAmplitudes(CNuc *theCNuc, const Config& configure) {
  int aa=theCNuc->GetPairNumFromKey(this->GetEntranceKey());
  if(theCNuc->GetPair(aa)->GetPType()==20) return;
  if(theCNuc->GetPair(aa)->IsEntrance()) {
    PPair *entrancePair=theCNuc->GetPair(aa);
    for(int j=1;j<=theCNuc->NumJGroups();j++)
      for(int la=1;la<=theCNuc->GetJGroup(j)->NumLevels();la++) {
    if(theCNuc->GetJGroup(j)->GetLevel(la)->IsECLevel()) {
      ALevel *ecLevel = theCNuc->GetJGroup(j)->GetLevel(la);
      int ir=theCNuc->GetPairNumFromKey(this->GetExitKey());
      if(ecLevel->GetECPairNum()==ir) {
        double inEnergy=this->GetCMEnergy()+entrancePair->GetSepE()+entrancePair->GetExE();
        for(int k=1;k<=entrancePair->GetDecay(ir)->NumKGroups();k++) {
          KGroup *theKGroup=entrancePair->GetDecay(ir)->GetKGroup(k);
          for(int ecm=1;ecm<=theKGroup->NumECMGroups();ecm++) {
        ECMGroup *theECMGroup=theKGroup->GetECMGroup(ecm);
        //entrance Phase Calculations;
        CoulFunc
theCoulombFunction(entrancePair,!!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
        struct CoulWaves
          coul=theCoulombFunction(theECMGroup->GetL(),entrancePair->GetChRad(),
                      this->GetCMEnergy());
        double eta=sqrt(uconv/2.)*fstruc*entrancePair->GetZ(1)*entrancePair->GetZ(2)*
```

```
          sqrt(entrancePair->GetRedMass()/this->GetCMEnergy());
        complex expCP(1.0,0.0);
        for(int ll=1;ll<=theECMGroup->GetL();ll++)
          expCP*=complex((double)ll/sqrt(pow((double)ll,2.0)+pow(eta,2.0)),
                 eta/sqrt(pow((double)ll,2.0)+pow(eta,2.0)));
        complex expHSP(coul.G/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)),
                   -coul.F/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)));

        double levelEnergy=ecLevel->GetE();
        double sqrtGammaPene=pow((inEnergy-levelEnergy)/hbarc,theECMGroup->GetMult()+0.5);

        //Initialize variables
        AChannel *theFinalChannel = theCNuc->GetJGroup(j)->GetChannel(theECMGroup->GetFinalChannel());
        PPair *theFinalPair=theCNuc->GetPair(theFinalChannel->GetPairNum());
        int theInitialLValue;
        double theInitialSValue;
        if(theECMGroup->IsChannelCapture()) {
          MGroup *theChanCapMGroup=entrancePair->GetDecay(theECMGroup->GetChanCapDecay())->
            GetKGroup(theECMGroup->GetChanCapKGroup())->GetMGroup(theECMGroup->GetChanCapMGroup());
          theInitialLValue=theCNuc->GetJGroup(theChanCapMGroup->GetJNum())->
            GetChannel(theChanCapMGroup->GetChpNum())->GetL();
          theInitialSValue=theCNuc->GetJGroup(theChanCapMGroup->GetJNum())->
            GetChannel(theChanCapMGroup->GetChpNum())->GetS();
        } else {
          theInitialLValue=theECMGroup->GetL();
          theInitialSValue=theKGroup->GetS();
        }

        ECIntegral theECIntegral(theFinalPair,configure);
        complex integrals = theECIntegral(theInitialLValue, theFinalChannel->GetL(),
                          theInitialSValue, theFinalChannel->GetS(),
                          theECMGroup->GetJ(), theCNuc->GetJGroup(j)->GetJ(),
                          theECMGroup->GetMult(), theECMGroup->GetRadType(),
                          inEnergy, levelEnergy,
                          theECMGroup->IsChannelCapture());

        //calculate the total radial integral
        complex ecAmplitude=expCP*expHSP*sqrtGammaPene*integrals;
        this->AddECAmplitude(k,ecm,ecAmplitude);
          }
        }
      }
    }
      }
    }
  }
  for(int i=1;i<=this->NumSubPoints();i++) {
    this->GetSubPoint(i)->CalculateECAmplitudes(theCNuc,configure);
  }
}

void EPoint::AddECAmplitude(int kGroupNum, int ecMGroupNum, complex ecAmplitude) {
  vector_c d;
  while(kGroupNum>ec_amplitudes_.size()) ec_amplitudes_.push_back(d);
  ec_amplitudes_[kGroupNum-1].push_back(ecAmplitude);
  assert(ecMGroupNum=ec_amplitudes_[kGroupNum-1].size());
}

void EPoint::Calculate(CNuc* theCNuc,const Config &configure, EPoint *parent, int subPointNum) {

  if(!this->IsTargetEffect()||
     (!this->GetParentData()->GetTargetEffect(this->GetTargetEffectNum())->IsConvolution()&&
      !this->GetParentData()->GetTargetEffect(this->GetTargetEffectNum())->IsTargetIntegration())) {
    GenMatrixFunc *theMatrixFunc;
    if(configure.paramMask & Config::USE_AMATRIX) theMatrixFunc=new AMatrixFunc(theCNuc,configure);
    else theMatrixFunc=new RMatrixFunc(theCNuc,configure);
    theMatrixFunc->ClearMatrices();
    theMatrixFunc->FillMatrices(this);
    theMatrixFunc->InvertMatrices();
    theMatrixFunc->CalculateTMatrix(this);
    theMatrixFunc->CalculateCrossSection(this);
    if(subPointNum&&parent) {
      for(int i=1;i<=parent->NumLocalMappedPoints();i++) {
    EPoint *mappedSubPoint = parent->GetLocalMappedPoint(i)->
      GetSubPoint(subPointNum);
    theMatrixFunc->CalculateCrossSection(mappedSubPoint);
      }
    } else {
      for(int i=1;i<=this->NumLocalMappedPoints();i++) {
    EPoint *mappedPoint = this->GetLocalMappedPoint(i);
    theMatrixFunc->CalculateCrossSection(mappedPoint);
      }
    }
    delete theMatrixFunc;
  } else {
    for(int i = 1; i<=this->NumSubPoints();i++) {
      EPoint *subPoint=this->GetSubPoint(i);
```

```
      if(this->NumLocalMappedPoints()>0)
    subPoint->Calculate(theCNuc,configure,this,i);
      else subPoint->Calculate(theCNuc,configure);
    }
    this->IntegrateTargetEffect();
    for(int i=1;i<=this->NumLocalMappedPoints();i++)
      this->GetLocalMappedPoint(i)->IntegrateTargetEffect();
  }
}

void EPoint::SetMap(int segmentNum, int pointNum) {
  is_mapped_=true;
  energy_map_.segment=segmentNum;
  energy_map_.point=pointNum;
}

void EPoint::AddLocalMappedPoint(EPoint *point) {
  local_mapped_points_.push_back(point);
}

void EPoint::ClearLocalMappedPoints() {
  local_mapped_points_.clear();
}

void EPoint::SetTargetEffectNum(int targetEffectNum) {
  targetEffectNum_=targetEffectNum;
}

void EPoint::AddSubPoint(EPoint subPoint) {
  integrationPoints_.push_back(subPoint);
}

void EPoint::IntegrateTargetEffect() {
  double yield=0.0;
  TargetEffect *targetEffect=this->GetParentData()->GetTargetEffect(this->GetTargetEffectNum());
  double energyStep=this->GetSubPoint(1)->GetCMEnergy()-this->GetSubPoint(2)->GetCMEnergy();
  if(targetEffect->IsConvolution()&&targetEffect->IsTargetIntegration()) {
    int outerLowerLimit=round((this->GetSubPoint(1)->GetCMEnergy()-this->GetCMEnergy())/energyStep)+1;
    int outerUpperLimit=outerLowerLimit-1+round(this->GetTargetThickness()/energyStep);
    double outerIntFirst=0.0;
    double outerIntEvenSum=0.0;
    double outerIntOddSum=0.0;
    double outerIntegral=0.0;
    int outerCounter=0;
    for(int i=outerLowerLimit;i<=outerUpperLimit;i++) {
      int innerLowerLimit;
      if(i-round(targetEffect->convolutionRange*targetEffect->GetSigma()/energyStep)>1)
    innerLowerLimit=i-round(targetEffect->convolutionRange*targetEffect->GetSigma()/energyStep);
      else innerLowerLimit=1;
      int innerUpperLimit;
      if(i+round(targetEffect->convolutionRange*targetEffect->GetSigma()/energyStep)-1<targetEffect->NumSubPoints())
    innerUpperLimit=i+round(targetEffect->convolutionRange*targetEffect->GetSigma()/energyStep)-1;
      else innerUpperLimit=targetEffect->NumSubPoints();
      double innerIntFirst=0.0;
      double innerIntEvenSum=0.0;
      double innerIntOddSum=0.0;
      double innerIntegral=0.0;
      double centroid=this->GetSubPoint(i)->GetCMEnergy();
      int innerCounter=0;
      for(int ii=innerLowerLimit;ii<=innerUpperLimit;ii++) {
    double thisEnergy=this->GetSubPoint(ii)->GetCMEnergy();
    double innerIntegrand=this->GetSubPoint(ii)->GetFitCrossSection()/
this->GetSubPoint(ii)->GetStoppingPower()/1e24*targetEffect->GetConvolutionFactor(thisEnergy,centroid);
    if(innerCounter==0) innerIntFirst=innerIntegrand;
    else if(innerCounter%2==0) {
      innerIntEvenSum+=innerIntegrand;
      if(innerCounter>=2)
innerIntegral=energyStep/3.0*(innerIntFirst+4.0*innerIntOddSum+2.0*innerIntEvenSum-innerIntegrand);
    } else if(innerCounter%2!=0) {
      innerIntOddSum+=innerIntegrand;
      if(innerCounter>=2)
innerIntegral=energyStep/3.0*(innerIntFirst+4.0*innerIntOddSum+2.0*innerIntEvenSum-3.0*innerIntegrand);
    }
    innerCounter++;
      }
      if(outerCounter==0) outerIntFirst=innerIntegral;
      else if(outerCounter%2==0) {
    outerIntEvenSum+=innerIntegral;
    if(outerCounter>=2) outerIntegral=energyStep/3.0*
           (outerIntFirst+4.0*outerIntOddSum+2.0*outerIntEvenSum-innerIntegral);
      } else if(outerCounter%2!=0) {
    outerIntOddSum+=innerIntegral;
    if(outerCounter>=2) outerIntegral=energyStep/3.0*
           (outerIntFirst+4.0*outerIntOddSum+2.0*outerIntEvenSum-3.0*innerIntegral);
      }
```

```
      outerCounter++;
    }
    yield=outerIntegral;
  } else if(targetEffect->IsConvolution()) {
    double intFirst=0.0;
    double intEvenSum=0.0;
    double intOddSum=0.0;
    double integral=0.0;
    double centroid=this->GetCMEnergy();
    for(int i=0;i<this->NumSubPoints();i++) {
      double thisEnergy=this->GetSubPoint(i+1)->GetCMEnergy();
      double integrand=this->GetSubPoint(i+1)->GetFitCrossSection()
    *targetEffect->GetConvolutionFactor(thisEnergy,centroid);
      if(i==0) intFirst=integrand;
      else if(i%2==0) {
    intEvenSum+=integrand;
    if(i>=2) integral=energyStep/3.0*(intFirst+4.0*intOddSum+2.0*intEvenSum-integrand);
      } else if(i%2!=0) {
      intOddSum+=integrand;
      if(i>=2) integral=energyStep/3.0*(intFirst+4.0*intOddSum+2.0*intEvenSum-3.0*integrand);
      }
    }
    yield=integral;
  } else if(targetEffect->IsTargetIntegration()) {
    double intFirst=0.0;
    double intEvenSum=0.0;
    double intOddSum=0.0;
    double integral=0.0;
    for(int i=0;i<this->NumSubPoints();i++) {
      double thisEnergy=this->GetSubPoint(i+1)->GetCMEnergy();
      double integrand=this->GetSubPoint(i+1)->GetFitCrossSection()/
    this->GetSubPoint(i+1)->GetStoppingPower()/1e24;
      if(i==0) intFirst=integrand;
      else if(i%2==0) {
    intEvenSum+=integrand;
    if(i>=2) integral=energyStep/3.0*(intFirst+4.0*intOddSum+2.0*intEvenSum-integrand);
      } else if(i%2!=0) {
      intOddSum+=integrand;
      if(i>=2) integral=energyStep/3.0*(intFirst+4.0*intOddSum+2.0*intEvenSum-3.0*integrand);
      }
    }
    yield=integral;
  }
  this->SetFitCrossSection(yield);
}

void EPoint::SetParentData(EData* parentData) {
  parentData_=parentData;
}

void EPoint::SetStoppingPower(double stoppingPower) {
 stoppingPower_=stoppingPower;
}

void EPoint::SetTargetThickness(double targetThickness) {
  targetThickness_=targetThickness;
}

void EPoint::SetAngularDists(vector_r dists) {
  angularDists_.clear();
  angularDists_=dists;
}

EData *EPoint::GetParentData() const {
  return parentData_;
}

EPoint* EPoint::GetLocalMappedPoint(int mappedPointNum) const {
  return local_mapped_points_[mappedPointNum-1];
}

EPoint* EPoint::GetSubPoint(int subPoint) {
  EPoint *tempPoint;
  if(subPoint<=integrationPoints_.size()) tempPoint=&integrationPoints_[subPoint-1];
  else tempPoint= NULL;
  return tempPoint;
}

std::vector<EPoint>& EPoint::GetSubPoints() {
  return integrationPoints_;
}

std::vector<EPoint*>& EPoint::GetMappedPoints() {
  return local_mapped_points_;
}
```

8.264 /Users/kuba/Desktop/R-Matrix/AZURE2/src/Equation.cpp File Reference

```
#include ¨Equation.h¨
#include ¨Config.h¨
#include <iostream>
#include <cmath>
#include <cstdlib>
```

8.265 Equation.cpp

Go to the documentation of this file.

```
#include "Equation.h"
#include "Config.h"
#include <iostream>
#include <cmath>
#include <cstdlib>

Equation::Equation() {
}

Equation::Equation(std::string equation, int numParams, const Config &configure) :
infixEquation_(equation) {
  if(equation.size()!=0){
    for(int i=0;i<numParams;i++) {
      double tempDouble=0.0;
      parameters_.push_back(tempDouble);
    }
    Parse(configure);
  } else {
    configure.outStream « "Error: empty equation." « std::endl;
    std::exit(-1);
  }
}

Equation::Equation(std::string equation,std::vector<double> parameters, const Config &configure) :
  infixEquation_(equation), parameters_(parameters) {
  if(equation.size()!=0){
    Parse(configure);
  } else {
    configure.outStream « "Error: empty equation." « std::endl;
    std::exit(-1);
  }
}

Equation::Equation(std::string equation,double parameters[], size_t arraySize, const Config
&configure) :
  infixEquation_(equation) {
  if(equation.size()!=0){
    parameters_=std::vector<double>(parameters,parameters+arraySize/sizeof(double));
    Parse(configure);
  } else {
    configure.outStream « "Error: empty equation." « std::endl;
    std::exit(-1);
  }
}

void Equation::Initialize(std::string equation, int numParams, const Config &configure) {
  if(equation.size()!=0){
    infixEquation_=equation;
    for(int i=0;i<numParams;i++) {
      double tempDouble=0.0;
      parameters_.push_back(tempDouble);
    }
    Parse(configure);
  } else {
    configure.outStream « "Error: empty equation." « std::endl;
    std::exit(-1);
  }
}

void Equation::BuildFunctionList() {
  functionList_["cos"]=GenericFunction(&std::cos);
  functionList_["sin"]=GenericFunction(&std::sin);
```

```
  functionList_["tan"]=GenericFunction(&std::tan);
  functionList_["asin"]=GenericFunction(&std::asin);
  functionList_["acos"]=GenericFunction(&std::acos);
  functionList_["atan"]=GenericFunction(&std::atan);
  functionList_["exp"]=GenericFunction(&std::exp);
  functionList_["e^"]=GenericFunction(&std::exp);
  functionList_["ln"]=GenericFunction(&std::log);
  functionList_["log"]=GenericFunction(&std::log10);
  functionList_["sqrt"]=GenericFunction(&std::sqrt);
}

void Equation::Parse(const Config &configure) {
  BuildFunctionList();
  try{
    unsigned int position=0;
    std::vector<TokenPair> stack;
    //Initial expectation list
    unsigned char expecting = FUNCTION|NUMBER|PARAMETER|VARIABLE|LEFTPAR;
    while(position<infixEquation_.length()) {
      //Retrieve token
      TokenPair tempPair=GetToken(position,configure);
      //If last token, enforce additional expectations.
      if(position>=infixEquation_.length())
    expecting&=FUNCTION|NUMBER|PARAMETER|VARIABLE|RIGHTPAR;
      //Check token against expectations
      if(!(expecting&tempPair.first)) throw SyntaxError(infixEquation_,3,position-1);
      //Set new expectations for next token
      if(tempPair.first==NUMBER || tempPair.first==VARIABLE ||
     tempPair.first==PARAMETER || tempPair.first==FUNCTION) expecting=OPERATOR|RIGHTPAR;
      else if(tempPair.first==OPERATOR || tempPair.first==LEFTPAR) expecting=FUNCTION|NUMBER|PARAMETER|VARIABLE|LEFTPAR;
      else if(tempPair.first==RIGHTPAR) expecting=OPERATOR|RIGHTPAR;
      //Shunting yard algorithm
      // Numbers, functions, parameters, and variable go directly to output
if(tempPair.first==NUMBER||tempPair.first==PARAMETER||tempPair.first==VARIABLE||tempPair.first==FUNCTION) {
    output_.push_back(tempPair);
      } else if(tempPair.first==OPERATOR) {
    if(stack.size()==0) stack.push_back(tempPair);
    else while(stack.size()>0) {
        //if top of stack is an operator
        if(stack[stack.size()-1].first==OPERATOR) {
          char lastChar=stack[stack.size()-1].second[0];
          //check precidence and associativity, while conditions are met move from stack to output
          if((GetOperatorType(tempPair.second[0])<=GetOperatorType(lastChar)&&
          GetOperatorAssociativity(tempPair.second[0])==LEFT)||
         (GetOperatorType(tempPair.second[0])<GetOperatorType(lastChar)&&
          GetOperatorAssociativity(tempPair.second[0])==RIGHT)) {
        output_.push_back(stack[stack.size()-1]);
        stack.pop_back();
        //if stack is empty, push new operator on stack and break loop
        if(stack.size()==0) {
          stack.push_back(tempPair);
          break;
        }
          } else {
        //if conditions aren't met, push new operator on stack and break loop
        stack.push_back(tempPair);
        break;
          }
        } else {
          //if top of stack is not operator, push new operator on stack and break loop
          stack.push_back(tempPair);
          break;
        }
      }
      } else if(tempPair.first==LEFTPAR) {
    //left parentheses go directly to the stack
    stack.push_back(tempPair);
      } else if(tempPair.first==RIGHTPAR) {
    //right parentheses initiate push of stack to output until left parenthesis is found
    while(stack.size()>0&&stack[stack.size()-1].first!=LEFTPAR) {
      output_.push_back(stack[stack.size()-1]);
      stack.pop_back();
    }
    if(stack.size()==0) {
      //if stack is empty, parentheses were mismatched
      throw SyntaxError(infixEquation_,2,position-1);
    } else if(stack[stack.size()-1].first==LEFTPAR) stack.pop_back();
      }
    }
    //push remaining stack to output after all tokens are read
    while(stack.size()>0) {
      //if left parenthesis is found, parentheses were mismatched
      if(stack[stack.size()-1].first==LEFTPAR) throw SyntaxError(infixEquation_,2);
      output_.push_back(stack[stack.size()-1]);
```

```
      stack.pop_back();
    }
  } catch (SyntaxError e) {
    configure.outStream « e.what() « std::endl;
    std::exit(-1);
  }
}
std::vector<double> Equation::GetParameters() const {
  return parameters_;
}

void Equation::SetParameter(unsigned int index, double value,const Config& configure) {
  if(index<parameters_.size()) {
    parameters_[index]=value;
    for(unsigned int i = 0; i<subEquations_.size(); i++)
subEquations_[i].SetParameter(index,value,configure);
  } else configure.outStream « "Error: Parameter index " « index « " greater than vector size." «
std::endl;
}

bool Equation::IsOperator(char c) const {
  return (c=='+'||c=='-'||c=='*'||c=='/'||c=='^') ? (true) : (false);
}

bool Equation::IsDigit(char c) const {
  return (c>='0' && c<='9') ? (true) : (false);
}

unsigned int Equation::FindFunction(unsigned int &position) {
  unsigned int length =  0;
  for(std::map<std::string,GenericFunction>::iterator it = functionList_.begin();
      it!=functionList_.end();it++) {
    std::string searchKey = it->first+'(';
    if(infixEquation_.substr(position,searchKey.length()) == searchKey) {
      length = searchKey.length()-1;
      break;
    }
  }
  return length;
}

Equation::TokenPair Equation::GetToken(unsigned int &position,const Config& configure) {
  try {
    TokenType tempType;
    std::string tempString;
    //Number (positive decimal, integer, or exponential notation)
    if(IsDigit(infixEquation_[position])||infixEquation_[position]=='.') {
      tempType=NUMBER;

while((IsDigit(infixEquation_[position])||infixEquation_[position]=='.'||infixEquation_[position]=='e'||infixEquation_[

||(infixEquation_[position]=='-'&&(infixEquation_[position-1]=='e'||infixEquation_[position-1]=='E'))

||(infixEquation_[position]=='+'&&(infixEquation_[position-1]=='e'||infixEquation_[position-1]=='E')))
          &&position<infixEquation_.length()) {
    tempString+=infixEquation_[position];
    position++;
      }
    }
    //Check for negation. Implimented as a function.  MUST be checked before minus is parsed as an
operator.
    else if(infixEquation_[position]=='-'&&(position==0||IsOperator(infixEquation_[position-1])||
                        infixEquation_[position-1]=='(')) {
      tempType=FUNCTION;
      position++;
      tempString="neg";
    }
    //Operator (+,-,*,/, or ^)
    else if(IsOperator(infixEquation_[position])) {
      tempType=OPERATOR;
      tempString+=infixEquation_[position];
      position++;
    }
    //Parameter (must be of form a0,a1,...)
    else if(infixEquation_[position]=='a'&&IsDigit(infixEquation_[position+1])) {
      tempType=PARAMETER;
      position++;
      while(IsDigit(infixEquation_[position])&&position<infixEquation_.length()) {
    tempString+=infixEquation_[position];
    position++;
      }
      std::istringstream stm;
      stm.str(tempString);
      unsigned int paramNumber;stm»paramNumber;
      if(paramNumber+1>parameters_.size()) throw SyntaxError(infixEquation_,1,position-1);
    }
    //Dependent variable
```

```
    else if(infixEquation_[position]=='x') {
      tempType=VARIABLE;
      tempString+=infixEquation_[position];
      position++;
    }
    //Left Parenthesis
    else if(infixEquation_[position]=='(') {
      tempType=LEFTPAR;
      tempString+=infixEquation_[position];
      position++;
    }
    //Right Parenthesis
    else if(infixEquation_[position]==')') {
      tempType=RIGHTPAR;
      tempString+=infixEquation_[position];
      position++;
    }
    //Functions (currently supports whatever is linked in BuildFunctionList())
    else if(FindFunction(position)) {
      tempType=FUNCTION;
      unsigned int length = FindFunction(position);
      tempString=infixEquation_.substr(position,length);
      position+=(length+1);
    }
    //Unrecognized Tokens
    else throw SyntaxError(infixEquation_,0,position);
    //Read subequation if token is function
    if(tempType==FUNCTION) {
      std::string subString;
      //for negation, read until an operator (except ^), or unmatched right paranthesis, is found
      //  and create subequation
      if(tempString=="neg") {
    int parenCount=0;
    while(((infixEquation_[position]!=')'&&
        !((IsOperator(infixEquation_[position])&&infixEquation_[position]!='^') &&
          !(infixEquation_[position-1]=='e'||infixEquation_[position-1]=='E')))||
           parenCount!=0)&&
          position<infixEquation_.length()) {
      if(infixEquation_[position]=='(') parenCount++;
      else if(infixEquation_[position]==')') parenCount--;
      subString+=infixEquation_[position];
      position++;
    }
    if(parenCount>0&&position>=infixEquation_.length()) throw SyntaxError(infixEquation_,2);
      } else {
    //for regular functions, read until closing right parenthesis is found
    // and create subequation
    int parenCount=1;
    while(parenCount>0&&position<infixEquation_.length()) {
      if(infixEquation_[position]=='(') parenCount++;
      else if(infixEquation_[position]==')') parenCount--;
      if(parenCount!=0) subString+=infixEquation_[position];
      position++;
    }
    if(parenCount>0&&position>=infixEquation_.length()) throw SyntaxError(infixEquation_,2);
      }
      Equation subEquation(subString,parameters_,configure);
      subEquations_.push_back(subEquation);
      int subIndex=subEquations_.size()-1;
      std::ostringstream stm;
      stm « subIndex;
      tempString+=stm.str();
    }
    return TokenPair(tempType,tempString);
  } catch (SyntaxError e) {
    configure.outStream « e.what() « std::endl;
    std::exit(-1);
  }
}

Equation::OperatorType Equation::GetOperatorType(char c) const {
  switch (c) {
    case '+': return ADD;
    case '-': return SUBTRACT;
    case '*': return MULT;
    case '/': return DIVIDE;
    case '^': return POWER;
    default : return BADTYPE;
  }
}

Equation::Associativity Equation::GetOperatorAssociativity(char c) const {
  switch (c) {
    case '+': return LEFT;
    case '-': return LEFT;
    case '*': return LEFT;
    case '/': return LEFT;
```

```
    case '^': return RIGHT;
    default : return LEFT;
  }
}

std::string Equation::BinaryOperation(double left, double right, char op, const Config& configure)
const {
  std::ostringstream stm;
  stm.precision(15);
  double result;
  switch(op) {
    case '+':
      result=left+right;
      break;
    case '-':
      result=left-right;
      break;
    case '*':
      result=left*right;
      break;
    case '/':
      result=left/right;
      break;
    case '^':
      if(left<0.0&&fabs(int(right)-right)>0.0) {
    configure.outStream « "Warning: Exponent results in unsupported imaginary number." « std::endl;
    result=0.0;
      } else result=pow(left,right);
      break;
    default:
      result=0.0;
  }
  stm«result;
  return stm.str();
}

double Equation::FunctionOperation(TokenPair token, double x, const Config& configure) const {
  double result=0.0;
  if(token.second.substr(0,3)=="neg") {
    int subEquationIndex;
    std::istringstream stm;
    stm.str(token.second.substr(3));
    stm»subEquationIndex;
    result=-1.*subEquations_[subEquationIndex].Evaluate(configure,x);
  } else {
    for(std::map<std::string,GenericFunction>::const_iterator it = functionList_.begin();
    it!=functionList_.end();it++) {
      if(it->first==token.second.substr(0,it->first.length())) {
    int subEquationIndex;
    std::istringstream stm;
    stm.str(token.second.substr(it->first.length()));
    stm»subEquationIndex;
    result=it->second.Evaluate(subEquations_[subEquationIndex].Evaluate(configure,x));
    break;
      }
    }
  }
  return result;
}

double Equation::GetTokenValue(TokenPair token, double x, const Config& configure) const {
  double value;
  std::istringstream stm;
  if(token.first==NUMBER) {
    stm.clear();
    stm.str(token.second);
    stm » value;
  } else if(token.first==VARIABLE) {
    value=x;
  } else if(token.first==PARAMETER) {
    unsigned int paramNumber;
    stm.clear();
    stm.str(token.second);
    stm » paramNumber;
    value=parameters_[paramNumber];
  } else if(token.first==FUNCTION)
    value=FunctionOperation(token,x,configure);
  else value=0.0;
  return value;
}

double Equation::Evaluate(const Config& configure,double x) const {
  std::vector<TokenPair> localOutput=output_;
  std::istringstream stm;
  double result=0.0;
  int i=0;
  if(localOutput.size()==1) {
```

```
    result=GetTokenValue(localOutput[0],x,configure);
  } else {
    while(localOutput.size()!=1) {
      if(localOutput[i].first==OPERATOR) {
    char op=localOutput[i].second[0];
    double left=GetTokenValue(localOutput[i-2],x,configure);
    double right=GetTokenValue(localOutput[i-1],x,configure);
    localOutput.erase(localOutput.begin()+i-2,localOutput.begin()+i+1);
    localOutput.insert(localOutput.begin()+i-2,
               TokenPair(NUMBER,BinaryOperation(left,right,op,configure)));
    i--;
      }
      else i++;
    }
    stm.clear();
    stm.str(localOutput[0].second);
    stm » result;
  }
  return result;
}

```

8.266 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ESegment.cpp File Reference

```
#include ¨CNuc.h¨
#include ¨DataLine.h¨
#include ¨EData.h¨
#include ¨ESegment.h¨
#include ¨ExtrapLine.h¨
#include ¨SegLine.h¨
```

8.267 ESegment.cpp

Go to the documentation of this file.

```
#include "CNuc.h"
#include "DataLine.h"
#include "EData.h"
#include "ESegment.h"
#include "ExtrapLine.h"
#include "SegLine.h"

ESegment::ESegment(SegLine segLine) {
  entrancekey_=segLine.entranceKey();
  exitkey_=segLine.exitKey();
  min_e_=segLine.minE();
  max_e_=segLine.maxE();
  min_a_=segLine.minA();
  max_a_=segLine.maxA();
  e_step_=0.0;
  a_step_=0.0;
  segment_chi_squared_=0.0;
  if(segLine.isDiff()==1) isdifferential_=true;
  else isdifferential_=false;
  if(segLine.isDiff()==2) {
    isphase_=true;
    j_=segLine.phaseJ();
    l_=segLine.phaseL();
  } else {
    isphase_=false;
    j_=0.0;
    l_=0;
  }
  isTotalCapture_ =  (segLine.isDiff()==3) ? 1 : 0;
  isAngDist_=false;
  maxAngDistOrder_=0;
  datafile_=segLine.dataFile();
  dataNorm_= dataNormNominal_ = segLine.dataNorm();
  dataNormError_=segLine.dataNormError();
  if(segLine.varyNorm()==1) varyNorm_=true;
```

```
  else varyNorm_=false;
  targetEffectNum_=0;
  isTargetEffect_=false;
}

ESegment::ESegment(ExtrapLine extrapLine) {
  entrancekey_=extrapLine.entranceKey();
  exitkey_=extrapLine.exitKey();
  min_e_=extrapLine.minE();
  max_e_=extrapLine.maxE();
  min_a_=extrapLine.minA();
  max_a_=extrapLine.maxA();
  e_step_=extrapLine.eStep();
  a_step_=extrapLine.aStep();
  segment_chi_squared_=0.0;
  if(extrapLine.isDiff()==1) isdifferential_=true;
  else isdifferential_=false;
  if(extrapLine.isDiff()==2) {
    isphase_=true;
    j_=extrapLine.phaseJ();
    l_=extrapLine.phaseL();
  } else {
    isphase_=false;
    j_=0.0;
    l_=0;
  }
  if(extrapLine.isDiff()==3) {
    isAngDist_=true;
    maxAngDistOrder_=extrapLine.maxAngDistOrder();
  } else {
    isAngDist_=false;
    maxAngDistOrder_=0;
  }
  isTotalCapture_ =  (extrapLine.isDiff()==4) ? 1 : 0;
  datafile_="";
  dataNorm_= dataNormNominal_ = 1.;
  dataNormError_=0.;
  varyNorm_=false;
  targetEffectNum_=0;
  isTargetEffect_=false;
}

bool ESegment::IsInSegment(EPoint point) {
  bool b=false;
  if(point.GetLabEnergy()>=this->GetMinEnergy()&&
     point.GetLabEnergy()<=this->GetMaxEnergy()) {
    if(this->IsDifferential()) {
      if(point.GetLabAngle()>=this->GetMinAngle()&&
     point.GetLabAngle()<=this->GetMaxAngle()) b=true;
    } else b=true;
  }
  return b;
}

bool ESegment::IsDifferential() const {
  return isdifferential_;
}

bool ESegment::IsPhase() const {
  return isphase_;
}

int ESegment::IsTotalCapture() const {
  return isTotalCapture_;
}

bool ESegment::IsAngularDist() const {
  return isAngDist_;
}

bool ESegment::IsTargetEffect() const {
  return isTargetEffect_;
}

bool ESegment::IsVaryNorm() const {
  return varyNorm_;
}

int ESegment::NumPoints() const {
  return points_.size();
}

int ESegment::GetEntranceKey() const {
  return entrancekey_;
}

int ESegment::GetExitKey() const {
```

```
  return exitkey_;
}

int ESegment::Fill(CNuc *theCNuc, EData *theData, const Config& configure) {
  std::string infile=this->GetDataFile();
  std::ifstream in(infile.c_str());
  if(!in) return -1;
  while(!in.eof()) {
    DataLine line(in);
    if(!in.eof()) {
      EPoint NewEPoint(line,this);
      if(this->IsInSegment(NewEPoint)) {
    this->AddPoint(NewEPoint);
    PPair *entrancePair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetEntranceKey()));
    PPair *exitPair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetExitKey()));
    this->GetPoint(this->NumPoints())->SetParentData(theData);
    if(entrancePair->GetPType()==20) this->GetPoint(this->NumPoints())->ConvertDecayEnergy(exitPair);
    else this->GetPoint(this->NumPoints())->ConvertLabEnergy(entrancePair);
    if(exitPair->GetPType()==0&&this->IsDifferential()&&!this->IsPhase()) {
      if(this->GetEntranceKey()==this->GetExitKey()) {
        this->GetPoint(this->NumPoints())->ConvertLabAngle(entrancePair);
      } else {
        this->GetPoint(this->NumPoints())->ConvertLabAngle(entrancePair,exitPair,configure);
      }
      this->GetPoint(this->NumPoints())->ConvertCrossSection(entrancePair,exitPair);
    }
      }
    }
  }
  in.close();
  return 0;
}

int ESegment::GetL() const {
  return l_;
}

int ESegment::GetTargetEffectNum() const {
  return targetEffectNum_;
}

int ESegment::GetSegmentKey() const {
  return segmentKey_;
}

int ESegment::GetMaxAngDistOrder() const {
  return maxAngDistOrder_;
}

double ESegment::GetMinEnergy() const {
  return min_e_;
}

double ESegment::GetMaxEnergy() const {
  return max_e_;
}

double ESegment::GetMinAngle() const {
  return min_a_;
}

double ESegment::GetMaxAngle() const {
  return max_a_;
}

double ESegment::GetSegmentChiSquared() const {
  return segment_chi_squared_;
}

double ESegment::GetEStep() const {
  return e_step_;
}

double ESegment::GetAStep() const {
  return a_step_;
}

double ESegment::GetJ() const {
  return j_;
}

double ESegment::GetNorm() const {
  return dataNorm_;
}

double ESegment::GetNominalNorm() const {
  return dataNormNominal_;
```

```
}

double ESegment::GetNormError() const {
  return dataNormError_;
}

std::string ESegment::GetDataFile() const {
  return datafile_;
}

void ESegment::AddPoint(EPoint point) {
  points_.push_back(point);
}

void ESegment::SetSegmentChiSquared(double chiSquared) {
  segment_chi_squared_=chiSquared;
}

void ESegment::SetTargetEffectNum(int targetEffectNum) {
  targetEffectNum_=targetEffectNum;
  isTargetEffect_=true;
}

void ESegment::SetSegmentKey (int segmentKey) {
  segmentKey_=segmentKey;
}

void ESegment::SetNorm(double norm) {
  dataNorm_=norm;
}

void ESegment::SetExitKey(int key) {
  exitkey_=key;
  for(int i=1;i<=this->NumPoints();i++)
    this->GetPoint(i)->SetExitKey(key);
}

void ESegment::SetIsTotalCapture(int num) {
  isTotalCapture_=num;
}

void ESegment::SetVaryNorm(bool varyNorm) {
  varyNorm_=varyNorm;

}
EPoint *ESegment::GetPoint(int pointNum) {
  EPoint *b=&points_[pointNum-1];
  return b;
}

std::vector<EPoint>& ESegment::GetPoints() {
  return points_;
}
```

8.268 /Users/kuba/Desktop/R-Matrix/AZURE2/src/GenMatrixFunc.cpp File Reference

```
#include ¨CNuc.h¨
#include ¨EPoint.h¨
#include ¨GenMatrixFunc.h¨
#include <assert.h>
```

8.269 GenMatrixFunc.cpp

Go to the documentation of this file.

```
#include "CNuc.h"
#include "EPoint.h"
#include "GenMatrixFunc.h"
#include <assert.h>

void GenMatrixFunc::CalculateCrossSection(EPoint *point) {
```

```
  complex sum(0.,0.);
  int aa=compound()->GetPairNumFromKey(point->GetEntranceKey());
  int ir=0;
  while(ir<compound()->GetPair(aa)->NumDecays()) {
    ir++;

if(compound()->GetPair(aa)->GetDecay(ir)->GetPairNum()==compound()->GetPairNumFromKey(point->GetExitKey()))
break;
  }
  Decay *theDecay=compound()->GetPair(aa)->GetDecay(ir);
  if(compound()->GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->GetPType()==10 &&
     (configure().paramMask & Config::USE_RMC_FORMALISM)) {
    int decayNum=0;
    while(decayNum<compound()->GetPair(aa)->NumDecays()) {
      decayNum++;
      if(compound()->GetPair(aa)->GetDecay(decayNum)->GetPairNum()==aa) break;
    }
    for(int k=1;k<=compound()->GetPair(aa)->GetDecay(decayNum)->NumKGroups();k++) {
      for(int m=1;m<=compound()->GetPair(aa)->GetDecay(decayNum)->GetKGroup(k)->NumMGroups();m++) {
    MGroup *theMGroup=compound()->GetPair(aa)->GetDecay(decayNum)->GetKGroup(k)->GetMGroup(m);
    if(theMGroup->GetChNum()==theMGroup->GetChpNum()) {
      double jValue=compound()->GetJGroup(theMGroup->GetJNum())->GetJ();
      sum+=2.*point->GetGeometricalFactor()*
        (2.*jValue+1.)*compound()->GetPair(aa)->GetI1I2Factor()*
        imag(this->GetTMatrixElement(k,m,decayNum));
    }
      }
    }
    for(int dp=1;dp<=compound()->GetPair(aa)->NumDecays();dp++) {
      if(compound()->GetPair(compound()->GetPair(aa)->GetDecay(dp)->GetPairNum())->GetPType()==0) {
    for(int k=1;k<=compound()->GetPair(aa)->GetDecay(dp)->NumKGroups();k++) {
      this->ClearTempTMatrices();
      for(int m=1;m<=compound()->GetPair(aa)->GetDecay(dp)->GetKGroup(k)->NumMGroups();m++) {
        MGroup *theMGroup=compound()->GetPair(aa)->GetDecay(dp)->GetKGroup(k)->GetMGroup(m);
        int
lValue=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChNum())->GetL();
        int
lpValue=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChpNum())->GetL();
        double jValue=compound()->GetJGroup(theMGroup->GetJNum())->GetJ();
        int tempTNum=this->IsTempTMatrix(jValue,lValue,lpValue);
        if(!tempTNum) {
          TempTMatrix temptmatrix={jValue,lValue,lpValue,this->GetTMatrixElement(k,m,dp)};
          this->NewTempTMatrix(temptmatrix);
        } else this->AddToTempTMatrix(tempTNum,this->GetTMatrixElement(k,m,dp));
      }
      for(int temp=1;temp<=this->NumTempTMatrices();temp++) {
        sum-=point->GetGeometricalFactor()*
          (2.*this->GetTempTMatrix(temp)->jValue+1.)*
          compound()->GetPair(aa)->GetI1I2Factor()*
          (this->GetTempTMatrix(temp)->TMatrix)*conj(this->GetTempTMatrix(temp)->TMatrix);
      }
    }
      }
    }
    point->SetFitCrossSection(real(sum)/100.);
  } else {
    if(!point->IsPhase()) {
      double angleIntegratedXS=0.;
      if(!point->IsDifferential()) {
    for(int k=1;k<=theDecay->NumKGroups();k++) {
      this->ClearTempTMatrices();
      for(int m=1;m<=theDecay->GetKGroup(k)->NumMGroups();m++) {
        if(compound()->GetPair(aa)->GetPType()==20) {
          sum+=25.*this->GetTMatrixElement(k,m)*conj(this->GetTMatrixElement(k,m));
        } else {
          MGroup *theMGroup=theDecay->GetKGroup(k)->GetMGroup(m);
          int
lValue=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChNum())->GetL();
          int
lpValue=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChpNum())->GetL();
          double jValue=compound()->GetJGroup(theMGroup->GetJNum())->GetJ();
          int tempTNum=this->IsTempTMatrix(jValue,lValue,lpValue);
          if(!tempTNum) {
        TempTMatrix temptmatrix={jValue,lValue,lpValue,this->GetTMatrixElement(k,m)};
        this->NewTempTMatrix(temptmatrix);
          } else this->AddToTempTMatrix(tempTNum,this->GetTMatrixElement(k,m));
        }
      }
      if(compound()->GetPair(aa)->GetPType()==20) continue;
      for(int m=1;m<=theDecay->GetKGroup(k)->NumECMGroups();m++) {
        ECMGroup *theECMGroup=theDecay->GetKGroup(k)->GetECMGroup(m);
        int lValue=theECMGroup->GetL();
        int lpValue=theECMGroup->GetMult();
        double jValue=theECMGroup->GetJ();
        int tempTNum=this->IsTempTMatrix(jValue,lValue,lpValue);
        if(!tempTNum) {
          TempTMatrix temptmatrix={jValue,lValue,lpValue,this->GetECTMatrixElement(k,m)};
```

```
          this->NewTempTMatrix(temptmatrix);
        } else this->AddToTempTMatrix(tempTNum,this->GetECTMatrixElement(k,m));
      }
      for(int temp=1;temp<=this->NumTempTMatrices();temp++) {
        sum+=point->GetGeometricalFactor()*
          (2.*this->GetTempTMatrix(temp)->jValue+1.)*
          compound()->GetPair(aa)->GetI1I2Factor()*
          (this->GetTempTMatrix(temp)->TMatrix)*conj(this->GetTempTMatrix(temp)->TMatrix);
      }
    }
    angleIntegratedXS=real(sum)/100.;
    if(!point->IsAngularDist()) {
      point->SetFitCrossSection(angleIntegratedXS);
      return;
    }
      }
      std::vector<double>
angularCoeff(std::min(point->GetMaxLOrder()+1,point->GetMaxAngDistOrder()+1),0.);
      for(int kL=1;kL<=theDecay->NumKLGroups();kL++) {
    for(int inter=1;inter<=theDecay->GetKLGroup(kL)
          ->NumInterferences();inter++) {
      Interference *theInterference=theDecay->GetKLGroup(kL)
        ->GetInterference(inter);
      complex T1(0.0,0.0),T2(0.0,0.0);
      std::string interferenceType=theInterference->GetInterferenceType();
      if(interferenceType=="RR") {
        T1=this->GetTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM1());
        T2=this->GetTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM2());
      } else if(interferenceType=="ER") {
        T1=this->GetECTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM1());
        T2=this->GetTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM2());
      } else if(interferenceType=="RE") {
        T1=this->GetTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM1());
        T2=this->GetECTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM2());
      } else if(interferenceType=="EE") {
        T1=this->GetECTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM1());
        T2=this->GetECTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM2());
      }
      int lOrder = theDecay->GetKLGroup(kL)->GetLOrder();
      sum+=theInterference->GetZ1Z2()*T1*conj(T2)*
        point->GetLegendreP(lOrder);
      if((lOrder < angularCoeff.size()) && point->IsAngularDist()) {
        double tempCoeff=angularCoeff[lOrder]+
          real(theInterference->GetZ1Z2()*T1*conj(T2))*point->GetGeometricalFactor()*
          compound()->GetPair(aa)->GetI1I2Factor()/100.*4./angleIntegratedXS;
        angularCoeff[lOrder]=tempCoeff;
      }
    }
      }
      if(point->IsAngularDist()) {
    point->SetAngularDists(angularCoeff);
    return;
      }
      complex RT=sum/pi*point->GetGeometricalFactor()*
    compound()->GetPair(aa)->GetI1I2Factor();

      complex CT(0.,0.), IT(0.,0.);
      if(aa==ir) {
    complex coulombAmplitude=point->GetCoulombAmplitude();
    CT=coulombAmplitude*conj(coulombAmplitude)*point->GetGeometricalFactor();

    sum=complex(0.,0.);
    for(int k=1;k<=theDecay->NumKGroups();k++) {
      for(int m=1;m<=theDecay->GetKGroup(k)->NumMGroups();m++) {
        MGroup *theMGroup=theDecay->GetKGroup(k)->GetMGroup(m);
        AChannel
*entranceChannel=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChNum());
        AChannel
*exitChannel=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChpNum());
        if(entranceChannel==exitChannel)
          sum+=theMGroup->GetStatSpinFactor()*
        coulombAmplitude*conj(this->GetTMatrixElement(k,m))*
        point->GetLegendreP(compound()->GetJGroup(theMGroup->GetJNum())->
                    GetChannel(theMGroup->GetChNum())->GetL());
      }
    }
    IT=complex(0.,1.)/sqrt(pi)*sum*point->GetGeometricalFactor();
      }
      point->SetFitCrossSection((real(CT)+real(RT)+real(IT))/100.);
    } else if(aa==ir) {
      double segmentJ=point->GetJ();
      int segmentL=point->GetL();
      this->ClearTempTMatrices();
      for(int k=1;k<=theDecay->NumKGroups();k++) {
    for(int m=1;m<=theDecay->GetKGroup(k)->NumMGroups();m++) {
      MGroup *theMGroup=theDecay->GetKGroup(k)->GetMGroup(m);
      double jValue=compound()->GetJGroup(theMGroup->GetJNum())->GetJ();
```

```
      AChannel *entranceChannel=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChNum());
      int lValue=entranceChannel->GetL();
      AChannel *exitChannel=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChpNum());
      if(jValue==segmentJ&&lValue==segmentL&&entranceChannel==exitChannel) {
        complex expCoulPhaseSquared=point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChNum())*
          point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChNum());
        complex theUMatrix=(expCoulPhaseSquared-this->GetTMatrixElement(k,m))/expCoulPhaseSquared;
        int tempTNum=this->IsTempTMatrix(jValue,lValue,lValue);
        if(!tempTNum) {
          TempTMatrix temptmatrix={jValue,lValue,lValue,theUMatrix};
          this->NewTempTMatrix(temptmatrix);
        } else this->AddToTempTMatrix(tempTNum,theUMatrix);
      }
    }
      }
      assert(this->NumTempTMatrices()<=1);
      double phase=0.0;
      if(this->NumTempTMatrices()==1) phase = 180.0/pi/2.0*
atan2(imag(this->GetTempTMatrix(1)->TMatrix),real(this->GetTempTMatrix(1)->TMatrix));
      //if(segmentL%2!=0&&phase<0) phase+=180.0;
      point->SetFitCrossSection(phase);
    }
  }
}


void GenMatrixFunc::NewTempTMatrix(TempTMatrix tempTMatrix) {
  temp_t_matrices_.push_back(tempTMatrix);
}

void GenMatrixFunc::AddToTempTMatrix(int tempTMatrixNum, complex tempValue) {
  this->GetTempTMatrix(tempTMatrixNum)->TMatrix+=tempValue;

}

void GenMatrixFunc::ClearTempTMatrices() {
  temp_t_matrices_.clear();
}

 void GenMatrixFunc::AddTMatrixElement(int kGroupNum ,int mGroupNum,complex tMatrixElement, int decayNum) {
  matrix_c d;
  vector_c e;
  while(decayNum>tmatrix_.size()) tmatrix_.push_back(d);
  while(kGroupNum>tmatrix_[decayNum-1].size()) tmatrix_[decayNum-1].push_back(e);
  tmatrix_[decayNum-1][kGroupNum-1].push_back(tMatrixElement);
  assert(kGroupNum==tmatrix_[decayNum-1].size());
  assert(mGroupNum==tmatrix_[decayNum-1][kGroupNum-1].size());
}

void GenMatrixFunc::AddECTMatrixElement(int kGroupNum ,int mGroupNum,complex tMatrixElement) {
  vector_c d;
  while(kGroupNum>ec_tmatrix_.size()) ec_tmatrix_.push_back(d);
  ec_tmatrix_[kGroupNum-1].push_back(tMatrixElement);
  assert(mGroupNum==ec_tmatrix_[kGroupNum-1].size());
}

int GenMatrixFunc::IsTempTMatrix(double jValue, int lValue, int lPrimeValue) {
  int d=0;
  bool e=false;
  while(!e&&d<this->NumTempTMatrices()) {
    if(jValue==this->GetTempTMatrix(d+1)->jValue&&
       lValue==this->GetTempTMatrix(d+1)->lValue&&
       lPrimeValue==this->GetTempTMatrix(d+1)->lpValue) e=true;
    d++;
  }
  if(!e) return 0;
  else return d;
}

int GenMatrixFunc::NumTempTMatrices() const {
  return temp_t_matrices_.size();
}

TempTMatrix *GenMatrixFunc::GetTempTMatrix(int tempTMatrixNum) {
  TempTMatrix *b=&temp_t_matrices_[tempTMatrixNum-1];
  return b;
}

 complex GenMatrixFunc::GetTMatrixElement(int kGroupNum, int mGroupNum, int decayNum) const {
  return tmatrix_[decayNum-1][kGroupNum-1][mGroupNum-1];
}

```

```
complex GenMatrixFunc::GetECTMatrixElement(int kGroupNum, int ecMGroupNum) const {
  return ec_tmatrix_[kGroupNum-1][ecMGroupNum-1];
}
```

8.270 /Users/kuba/Desktop/R-Matrix/AZURE2/src/GSLException.cpp File Reference

```
#include ¨GSLException.h¨
#include <gsl/gsl_errno.h>
```

8.271 GSLException.cpp

Go to the documentation of this file.

```
#include "GSLException.h"
#include <gsl/gsl_errno.h>

void GSLException::GSLErrorHandler(const char* reason,
                   const char* file,
                   int line,
                   int errorCode) {
  if(errorCode==GSL_EOVRFLW) return;
  std::ostringstream stm;
  stm « "GSL Error on line " « line « " of " « file « std::endl
      « reason « std::endl;
  std::string message = stm.str();
  throw GSLException(message);
}
```

8.272 /Users/kuba/Desktop/R-Matrix/AZURE2/src/IntegratedFermi↩ Func.cpp File Reference

```
#include ¨IntegratedFermiFunc.h¨
#include <math.h>
#include <gsl/gsl_sf_gamma.h>
#include <gsl/gsl_integration.h>
```

8.273 IntegratedFermiFunc.cpp

Go to the documentation of this file.

```
#include "IntegratedFermiFunc.h"
#include <math.h>
#include <gsl/gsl_sf_gamma.h>
#include <gsl/gsl_integration.h>

const double IntegratedFermiFunc::alpha_=1./137.036;
const double IntegratedFermiFunc::pi_=3.14159;
const double IntegratedFermiFunc::electronMass_=0.51099891;
const double IntegratedFermiFunc::hbarc_=197.327;

IntegratedFermiFunc::IntegratedFermiFunc(int charge, double V0) :
  charge_(charge), V0_(V0) {
}

double IntegratedFermiFunc::operator()(double W0,
                     double Z,
                     double radius) {
```

```
  if(W0<=1) return 0.;

  double gamma0 = sqrt(1.-alpha_*alpha_*Z*Z);
  double GammaDenom = gsl_sf_gamma(2.*gamma0+1.);
  Params_ params = {charge_,
                    gamma0,
                    Z,
                    radius,
                    W0,
                    GammaDenom*GammaDenom,
                    V0_};

  gsl_integration_workspace * w
    = gsl_integration_workspace_alloc (1000);
  gsl_function f;
  f.function = &IntegratedFermiFunc::Integrand;
  f.params = &params;

  double result, error;
  gsl_integration_qags (&f, 1., params.W0, 0., 1.e-6, 1000, w, &result, &error);

  gsl_integration_workspace_free (w);

  return result;
}

double IntegratedFermiFunc::Integrand(double x, void* p) {
  Params_* params = (Params_*)p;

  double W = (params->charge<0) ? x-params->V0*alpha_*alpha_*pow(params->Z,4./3.) :
x+params->V0*alpha_*alpha_*pow(params->Z,4./3.);
  double eta = (params->charge<0) ? alpha_*params->Z*W/sqrt(W*W-1.) :
    -alpha_*params->Z*W/sqrt(W*W-1.);
  gsl_sf_result GammaNumValue;
  gsl_sf_result GammaNumArg;
  gsl_sf_lngamma_complex_e(params->gamma0,eta,&GammaNumValue,&GammaNumArg);
  double GammaNum2 = exp(2.*GammaNumValue.val);

  double result =
    2.*(1.+params->gamma0)*
    exp(pi_*eta-2.*(1.-params->gamma0)*
    log(2.*electronMass_*params->radius*sqrt(W*W-1.)/hbarc_))*
    GammaNum2/params->GammaDenom2*sqrt(W*W-1.)*W*(params->W0-W)*(params->W0-W);

  if(params->V0!=0.) {
    result*=sqrt((W*W-1.)/(x*x-1.))*W/x;
  }

  return result;
}
```

8.274 /Users/kuba/Desktop/R-Matrix/AZURE2/src/Interference.cpp File Reference

```
#include "Interference.h"
```

8.275 Interference.cpp

Go to the documentation of this file.

```
#include "Interference.h"

Interference::Interference(int mGroupNum1, int mGroupNum2, double z1z2Coeff, std::string
interferenceType) :
  m1_(mGroupNum1), m2_(mGroupNum2),z1z2_(z1z2Coeff),intertype_(interferenceType) {};

std::string Interference::GetInterferenceType() const {
  return intertype_;
}

int Interference::GetM1() const {
  return m1_;
```

```
}

int Interference::GetM2() const {
  return m2_;
}

double Interference::GetZ1Z2() const {
  return z1z2_;
}

```

8.276 /Users/kuba/Desktop/R-Matrix/AZURE2/src/JGroup.cpp File Reference

```
#include ¨JGroup.h¨
#include ¨NucLine.h¨
```

8.277 JGroup.cpp

Go to the documentation of this file.

```
#include "JGroup.h"
#include "NucLine.h"

JGroup::JGroup(NucLine nucLine):
  pi_(nucLine.levelPi()), j_(nucLine.levelJ()), isinrmatrix_(true) {};

JGroup::JGroup(double j,int pi):
  pi_(pi),j_(j),isinrmatrix_(false) {};

bool JGroup::IsInRMatrix() const {
  return isinrmatrix_;
}

int JGroup::IsLevel(ALevel level) {
  bool b=false;
  int c=0;
  double tol=1e-3;
  while(!b&&c<this->NumLevels())
    {
      if(this->GetLevel(c+1)->GetE()-tol<=level.GetE()&&
     level.GetE()<=this->GetLevel(c+1)->GetE()+tol) b=true;
      c++;
    }
  if(b) return c;
  else return 0;
}

int JGroup::GetPi() const {
  return pi_;
}

int JGroup::NumLevels() const {
  return levels_.size();
}

int JGroup::NumChannels() {
  return channels_.size();
}

int JGroup::IsChannel(AChannel channel) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumChannels()) {
    if(channel.GetL()==this->GetChannel(c+1)->GetL()&&
       channel.GetS()==this->GetChannel(c+1)->GetS()&&
       channel.GetPairNum()==this->GetChannel(c+1)->GetPairNum()) b=true;
    c++;
  }
  if(b) return c;
  else return 0;
}

```

```
double JGroup::GetJ() const {
  return j_;
}

void JGroup::AddLevel(ALevel level) {
  levels_.push_back(level);
}

void JGroup::AddChannel(AChannel channel) {
  channels_.push_back(channel);
}

AChannel *JGroup::GetChannel(int channelNum) {
  AChannel *b=&channels_[channelNum-1];
  return b;
}

ALevel *JGroup::GetLevel(int levelNum) {
  ALevel *b=&levels_[levelNum-1];
  return b;
}
```

8.278 /Users/kuba/Desktop/R-Matrix/AZURE2/src/KGroup.cpp File Reference

```
#include ¨KGroup.h¨
```

8.279 KGroup.cpp

Go to the documentation of this file.

```
#include "KGroup.h"

KGroup::KGroup(double s, double sPrime) :
    s_(s), sp_(sPrime) {};

int KGroup::NumMGroups() const {
  return mgroups_.size();
}

int KGroup::NumECMGroups() const {
  return ec_mgroups_.size();
}

int KGroup::IsMGroup(MGroup mGroup) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumMGroups())
    {
      if(mGroup.GetChNum()==this->GetMGroup(c+1)->GetChNum()&&
     mGroup.GetChpNum()==this->GetMGroup(c+1)->GetChpNum()&&
     mGroup.GetJNum()==this->GetMGroup(c+1)->GetJNum()) b=true;
      c++;
    }
  if(b) return c;
  else return 0;
}

double KGroup::GetS() const {
  return s_;
}

double KGroup::GetSp() const {
  return sp_;
}

void KGroup::AddMGroup(MGroup mGroup) {
  mgroups_.push_back(mGroup);
}

void KGroup::AddECMGroup(ECMGroup ecMGroup) {
  ec_mgroups_.push_back(ecMGroup);
}
```

```
}
MGroup *KGroup::GetMGroup(int mGroupNum) {
  MGroup *b=&mgroups_[mGroupNum-1];
  return b;
}
ECMGroup *KGroup::GetECMGroup(int ecMGroupNum) {
  ECMGroup *b=&ec_mgroups_[ecMGroupNum-1];
  return b;
}
```

8.280 /Users/kuba/Desktop/R-Matrix/AZURE2/src/KLGroup.cpp File Reference

```
#include "KLGroup.h"
```

8.281 KLGroup.cpp

Go to the documentation of this file.

```
#include "KLGroup.h"

KLGroup::KLGroup(int kGroupNum,int lOrder) :
  k_(kGroupNum), lorder_(lOrder) {};

int KLGroup::GetK() const {
  return k_;
}

int KLGroup::GetLOrder() const {
  return lorder_;
}

int KLGroup::NumInterferences() const {
  return interferences_.size();
}

int KLGroup::IsInterference(Interference interference) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumInterferences())
    {
      if(interference.GetM1()==this->GetInterference(c+1)->GetM1()&&
     interference.GetM2()==this->GetInterference(c+1)->GetM2()&&
     interference.GetInterferenceType()==this->GetInterference(c+1)->GetInterferenceType()) b=true;
     c++;
    }
  if(b) return c;
  else return 0;
}

void KLGroup::AddInterference(Interference interference) {
  interferences_.push_back(interference);
}

Interference *KLGroup::GetInterference(int interferenceNum) {
  Interference *b=&interferences_[interferenceNum-1];
  return b;
}
```

8.282 /Users/kuba/Desktop/R-Matrix/AZURE2/src/MatrixInv.cpp File Reference

```
#include "MatrixInv.h"
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_matrix_complex_double.h>
#include <iostream>
```

8.283 MatrixInv.cpp

Go to the documentation of this file.

```
#include "MatrixInv.h"
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_matrix_complex_double.h>
#include <iostream>

MatrixInv::MatrixInv(const matrix_c &A) {
  inverse_.clear();

  gsl_complex x;
  gsl_matrix_complex * m = gsl_matrix_complex_alloc (A.size(), A.size());
  for(int i=0;i<A.size();i++) {
    for(int ii=0;ii<A.size();ii++) {
      GSL_SET_COMPLEX(&x,real(A[i][ii]),imag(A[i][ii]));
      gsl_matrix_complex_set (m,i,ii,x);
    }
  }
  int psign;
  gsl_permutation *p = gsl_permutation_alloc(A.size());
  gsl_matrix_complex * mi = gsl_matrix_complex_alloc (A.size(), A.size());
  gsl_linalg_complex_LU_decomp(m,p,&psign);
  gsl_linalg_complex_LU_invert(m,p,mi);

  vector_c AI_row;
  for(int i=0;i<A.size();i++) {
    inverse_.push_back(AI_row);
    for(int ii=0;ii<A.size();ii++) {
      x=gsl_matrix_complex_get(mi,i,ii);
      complex inv(GSL_REAL(x),GSL_IMAG(x));
      inverse_[i].push_back(inv);
    }
  }

  gsl_matrix_complex_free(m);
  gsl_matrix_complex_free(mi);
  gsl_permutation_free(p);
}
```

8.284 /Users/kuba/Desktop/R-Matrix/AZURE2/src/MGroup.cpp File Reference

```
#include "MGroup.h"
```

8.285 MGroup.cpp

Go to the documentation of this file.

```
#include "MGroup.h"

MGroup::MGroup(int jGroupNum, int channelNum, int channelPrimeNum) :
  jnum_(jGroupNum), ch_(channelNum), chp_(channelPrimeNum), statspinfactor_(0.0) {
}

int MGroup::GetChNum() const {
  return ch_;
}

int MGroup::GetChpNum() const {
  return chp_;
}

int MGroup::GetJNum() const {
  return jnum_;
}

double MGroup::GetStatSpinFactor() const {
  return statspinfactor_;
}

void MGroup::SetStatSpinFactor(double spinFactor) {
  statspinfactor_=spinFactor;
}
```

8.286 /Users/kuba/Desktop/R-Matrix/AZURE2/src/NFIntegral.cpp File Reference

`#include "NFIntegral.h"`
`#include <gsl/gsl_integration.h>`

8.287 NFIntegral.cpp

Go to the documentation of this file.

```
#include "NFIntegral.h"
#include <gsl/gsl_integration.h>

double NFIntegral::Integrand(double x, void * p) {
  Params *params = (Params*)p;
  WhitFunc *whitFunc=(params->whitFunc);
  int lfValue=(params->lfValue);
  double bindingEnergy=(params->bindingEnergy);
  double whitChRadSquaredValue=(params->whitChRadSquaredValue);

  double whit=whitFunc->operator()(lfValue,x,bindingEnergy);
  return pow(whit,2.0)/whitChRadSquaredValue;
}

double NFIntegral::operator()(int lf,double levelEnergy) {
  params_.lfValue = lf;
  params_.bindingEnergy = fabs(levelEnergy - totalSepE());

  params_.whitChRadSquaredValue =
pow(params_.whitFunc->operator()(lf,chanRad(),params_.bindingEnergy),2.0);

  gsl_integration_workspace * w
    = gsl_integration_workspace_alloc (1000);

  gsl_function F;
  F.function = &Integrand;
  F.params= &params_;

  double intresult,interror;

  gsl_integration_qagiu(&F,chanRad(),0.0,1e-4,1000,w,&intresult,&interror);

  gsl_integration_workspace_free (w);

  return intresult;
}
```

8.288 /Users/kuba/Desktop/R-Matrix/AZURE2/src/PPair.cpp File Reference

`#include "NucLine.h"`
`#include "PPair.h"`
`#include <assert.h>`

8.289 PPair.cpp

Go to the documentation of this file.

```
#include "NucLine.h"
#include "PPair.h"
#include <assert.h>

```

```
PPair::PPair(NucLine nucLine)
{
  pair_z_[0]=nucLine.z1();
  pair_z_[1]=nucLine.z2();
  pair_m_[0]=nucLine.m1();
  pair_m_[1]=nucLine.m2();
  pair_pi_[0]=nucLine.pi1();
  pair_pi_[1]=nucLine.pi2();
  pair_g_[0]=nucLine.g1();
  pair_g_[1]=nucLine.g2();
  pair_j_[0]=nucLine.j1();
  pair_j_[1]=nucLine.j2();
  pair_ex_e_=nucLine.e2();
  pair_sep_e_=nucLine.sepE();
  pair_ch_rad_=nucLine.chRad();
  pair_ptype_=nucLine.pType();
  pair_key_=nucLine.ir();
  red_mass_=nucLine.m1()*nucLine.m2()/(nucLine.m1()+nucLine.m2());
  i1i2factor_=1.0/(2.*nucLine.j1()+1.0)/(2.*nucLine.j2()+1.0);
  entrance_=false;
  ec_entrance_=false;
}

bool PPair::IsEntrance() const {
  return entrance_;
}

int PPair::GetZ(int particle) const {
  return  pair_z_[particle-1];
}

int PPair::GetPi(int particle) const {
  return  pair_pi_[particle-1];
}

int PPair::GetPType() const {
  return  pair_ptype_;
}

int PPair::NumDecays() const {
  return decays_.size();
}

int PPair::IsDecay(Decay decay) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumDecays())
    {
      if(decay.GetPairNum()==this->GetDecay(c+1)->GetPairNum()) b=true;
     c++;
    }
  if(b) return c;
  else return 0;
}

int PPair::IsDecay(int pairNum) {
  bool b=false;
  int c=0;
  while(!b&&c<this->NumDecays())
    {
      if(pairNum==this->GetDecay(c+1)->GetPairNum()) b=true;
     c++;
    }
  if(b) return c;
  else return 0;
}

int PPair::GetPairKey() const {
  return pair_key_;
}

double PPair::GetM(int particle) const {
  return pair_m_[particle-1];
}

double PPair::GetG(int particle) const {
  return pair_g_[particle-1];
}

double PPair::GetJ(int particle) const {
  return pair_j_[particle-1];
}

double PPair::GetExE() const {
  return pair_ex_e_;
}

```

```
double PPair::GetSepE() const {
  return pair_sep_e_;
}

double PPair::GetChRad()  const {
  return pair_ch_rad_;
}

double PPair::GetRedMass() const {
  return red_mass_;
}

double PPair::GetI1I2Factor() const {
  return i1i2factor_;
}

void PPair::AddDecay(Decay decay) {
  decays_.push_back(decay);
}

void PPair::SetEntrance() {
  entrance_=true;
}

Decay *PPair::GetDecay(int decayNum) {
  Decay *b=&decays_[decayNum-1];
  return b;
}

```

8.290 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ReactionRate.cpp File Reference

```
#include "CNuc.h"
#include "Config.h"
#include "EPoint.h"
#include "ReactionRate.h"
#include <iomanip>
#include <iostream>
#include <math.h>
#include <gsl/gsl_integration.h>
#include <omp.h>
#include <algorithm>
#include <time.h>
```

Classes

- struct gsl_reactionrate_params

Functions

- double gsl_reactionrate_integrand (double x, void *p)
- double gsl_reactionrate_integration (double temperature, CNuc *compound, const Config &configure, int entranceKey, int exitKey)

8.290.1 Function Documentation

8.290.1.1 gsl_reactionrate_integrand()

```
double gsl_reactionrate_integrand (
            double x,
            void * p )
```

Definition at line 22 of file ReactionRate.cpp.

8.290.1.2 gsl_reactionrate_integration()

```
double gsl_reactionrate_integration (
            double temperature,
            CNuc * compound,
            const Config & configure,
            int entranceKey,
            int exitKey )
```

Definition at line 42 of file ReactionRate.cpp.

8.291 ReactionRate.cpp

Go to the documentation of this file.

```
00001 #include "CNuc.h"
00002 #include "Config.h"
00003 #include "EPoint.h"
00004 #include "ReactionRate.h"
00005 #include <iomanip>
00006 #include <iostream>
00007 #include <math.h>
00008 #include <gsl/gsl_integration.h>
00009 #include <omp.h>
00010 #include <algorithm>
00011 #include <time.h>
00012
00013 struct gsl_reactionrate_params {
00014   gsl_reactionrate_params(const Config &config) : configure(config) {};
00015   const Config &configure;
00016   double temperature;
00017   CNuc *compound;
00018   int entranceKey;
00019   int exitKey;
00020 };
00021
00022 double gsl_reactionrate_integrand(double x, void * p) {
00023   struct gsl_reactionrate_params *params= (struct gsl_reactionrate_params *)p;
00024   CNuc *compound=params->compound;
00025   const Config &configure=params->configure;
00026   double temperature=params->temperature;
00027   int entranceKey=params->entranceKey;
00028   int exitKey=params->exitKey;
00029
00030   double crossSection;
00031   if(x<50.0&&x>0.001) {
00032     EPoint *point = new EPoint(55.0,x,entranceKey,exitKey,false,false,false,0.0,0,0);
00033     point->Initialize(compound,configure);
00034     point->Calculate(compound,configure);
00035     crossSection=point->GetFitCrossSection();
00036     delete point;
00037   } else crossSection=0.0;
00038
00039   return crossSection*x*exp(-x/temperature/boltzConst);
00040 }
00041
00042 double gsl_reactionrate_integration(double temperature,CNuc *compound,const Config& configure,
00043                         int entranceKey, int exitKey) {
00044
00045   struct gsl_reactionrate_params params(configure);
00046   params.temperature=temperature;
00047   params.compound=compound;
00048   params.entranceKey=entranceKey;
00049   params.exitKey=exitKey;
00050
00051   gsl_integration_workspace * w
00052     = gsl_integration_workspace_alloc (1000);
00053
00054   gsl_function F;
00055   F.function = &gsl_reactionrate_integrand;
00056   F.params=&params;
00057
00058   double result,error;
00059
00060   gsl_integration_qagiu(&F,0.00001,0.0,1e-4,1000,w,&result,&error);
00061
00062   double rate=1e-24*avagadroNum*lightSpeedInCmPerS*
```

```
    sqrt(8.0/pi/compound->GetPair(compound->GetPairNumFromKey(entranceKey))->GetRedMass()/uconv)/
    pow(boltzConst*temperature,1.5)*result;

  gsl_integration_workspace_free (w);

  return rate;
}

ReactionRate::ReactionRate(CNuc *compound, const vector_r &params,
                 const Config &configure, int entranceKey, int exitKey) :
  configure_(configure) {
  compound_=compound;
  compound_->FillCompoundFromParams(params);
  entrance_key_=entranceKey;
  exit_key_=exitKey;
}

void ReactionRate::CalculateRates() {
  int numSteps = (configure().rateParams.tempStep!=0.) ?
int((configure().rateParams.maxTemp-configure().rateParams.minTemp)/configure().rateParams.tempStep)+1
: 1;
  configure().outStream << std::setw(0) << "\t[                         ]
0%";configure().outStream.flush();
  int pointIndex=0;
  time_t startTime = time(NULL);
#pragma omp parallel for
  for(int i=0;i<numSteps;++i) {
    CNuc* localCompound = compound()->Clone();
    int localEntranceKey = entranceKey();
    int localExitKey = exitKey();
    const Config localConfigure = configure();
    double temp = localConfigure.rateParams.minTemp+i*localConfigure.rateParams.tempStep;

    double
rate=gsl_reactionrate_integration(temp,localCompound,localConfigure,localEntranceKey,localExitKey);
    rates_.push_back(RateData(temp,rate));
    delete localCompound;
    ++pointIndex;
    if(difftime(time(NULL),startTime)>0.25) {
      startTime=time(NULL);
      std::string progress="[";
      double percent=0.;
      for(int j = 1;j<=25;j++) {
    if(pointIndex>=percent*numSteps&&percent<1.) {
      percent+=0.04;
      progress+='*';
    } else progress+=' ';
      } progress+="] ";
      localConfigure.outStream << std::setw(0) << "\r\t" << progress << percent*100 <<
'%';localConfigure.outStream.flush();
    }
  }
  configure().outStream << std::setw(0) << "\r\t[*************************] 100%" << std::endl;
  std::sort(rates_.begin(),rates_.end());
}

void ReactionRate::CalculateFileRates() {
  std::ifstream inFile(configure().rateParams.temperatureFile.c_str());
  if(inFile) {
    while(!inFile.eof()) {
      std::string line;
      getline(inFile,line);
      if(!inFile.eof()) {
    double temp = 0.;
    std::istringstream stm;
    stm.str(line);
    if(stm >> temp) {
      rates_.push_back(RateData(temp,0.));
    }
      }
    }
    inFile.close();
    configure().outStream << std::setw(0) << "\t[                         ]
0%";configure().outStream.flush();
    int pointIndex=0;
    int numSteps = rates_.size();
    time_t startTime = time(NULL);
#pragma omp parallel for
    for(int i=0;i<numSteps;++i) {
      CNuc* localCompound = compound()->Clone();
      int localEntranceKey = entranceKey();
      int localExitKey = exitKey();
      const Config localConfigure = configure();

rates_[i].rate=gsl_reactionrate_integration(rates_[i].temperature,localCompound,localConfigure,localEntranceKey,localEx
```

```
      delete localCompound;
      ++pointIndex;
      if(difftime(time(NULL),startTime)>0.25) {
    startTime=time(NULL);
    std::string progress="[";
    double percent=0.;
    for(int j = 1;j<=25;j++) {
      if(pointIndex>=percent*numSteps&&percent<1.) {
        percent+=0.04;
        progress+='*';
      } else progress+=' ';
    } progress+="] ";
    localConfigure.outStream « std::setw(0) « "\r\t" « progress « percent*100 « '%';localConfigure.outStream.flush();
      }
    }
    configure().outStream « std::setw(0) « "\r\t[*************************] 100%" « std::endl;
  } else configure().outStream « "Couldn't open temperature file." « std::endl;
}

void ReactionRate::WriteRates() {
  std::string outputfile=configure().outputdir+"reactionrates.out";
  std::ofstream out;
  out.open(outputfile.c_str());
  if(out) {
    out « std::setw(20) « "T9" « std::setw(20) « "Rate" « std::endl;
    for(int i=0;i<rates_.size();i++) {
      out « std::setw(20) « rates_[i].temperature « std::setw(20) « rates_[i].rate « std::endl;
    }
    out.flush();
    out.close();
  } else  configure().outStream « "Could not write reaction rate file." « std::endl;
}
```

8.292 /Users/kuba/Desktop/R-Matrix/AZURE2/src/RMatrixFunc.cpp File Reference

```
#include "EPoint.h"
#include "CNuc.h"
#include "MatrixInv.h"
#include "RMatrixFunc.h"
#include <assert.h>
```

8.293 RMatrixFunc.cpp

Go to the documentation of this file.

```
#include "EPoint.h"
#include "CNuc.h"
#include "MatrixInv.h"
#include "RMatrixFunc.h"
#include <assert.h>

RMatrixFunc::RMatrixFunc(CNuc* compound, const Config &configure) :
  compound_(compound), configure_(configure) {}

complex RMatrixFunc::GetRMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum) const {
  return r_matrices_[jGroupNum-1][channelNum-1][channelPrimeNum-1];
}

complex RMatrixFunc::GetRLMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum) const {
  return rl_matrices_[jGroupNum-1][channelNum-1][channelPrimeNum-1];
}

complex RMatrixFunc::GetRLInvMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum) const {
  return rl_inv_matrices_[jGroupNum-1][channelNum-1][channelPrimeNum-1];
}

complex RMatrixFunc::GetRLInvRMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum) const
{
  return rl_inv_r_matrices_[jGroupNum-1][channelNum-1][channelPrimeNum-1];
```

```
}

matrix_c *RMatrixFunc::GetJSpecRLMatrix(int jGroupNum) {
  matrix_c *b=&rl_matrices_[jGroupNum-1];
  return b;
}

void RMatrixFunc::ClearMatrices() {
  r_matrices_.clear();
  rl_matrices_.clear();
  rl_inv_matrices_.clear();
  rl_inv_r_matrices_.clear();
  tmatrix_.clear();
  ec_tmatrix_.clear();
}

void RMatrixFunc::FillMatrices (EPoint *point) {
  double inEnergy;
  if(compound()->
     GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->
     GetPType()==20)
    inEnergy=point->GetCMEnergy()+
      compound()->
      GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->
      GetSepE()+
      compound()->
      GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->
      GetExE();
  else inEnergy=point->GetCMEnergy()+
     compound()->GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->GetSepE()+
     compound()->GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->GetExE();
  for(int j=1;j<=compound()->NumJGroups();j++) {
    if(compound()->GetJGroup(j)->IsInRMatrix()) {
      if(configure().paramMask & Config::USE_BRUNE_FORMALISM) {
    matrix_c qMatrixInverse;
    for(int la=1;la<=compound()->GetJGroup(j)->NumLevels();la++) {
      if(!compound()->GetJGroup(j)->GetLevel(la)->IsInRMatrix()) continue;
      ALevel *level=compound()->GetJGroup(j)->GetLevel(la);
      vector_c tempVector;
      for(int lap=1;lap<=compound()->GetJGroup(j)->NumLevels();lap++) {
        if(!compound()->GetJGroup(j)->GetLevel(lap)->IsInRMatrix()) continue;
        ALevel *levelp=compound()->GetJGroup(j)->GetLevel(lap);
        complex sum = (la==lap) ? complex(level->GetFitE()-inEnergy,0.0) : complex(0.0,0.0);
        for(int ch=1;ch<=compound()->GetJGroup(j)->NumChannels();ch++) {
          if(compound()->GetJGroup(j)->GetChannel(ch)->GetRadType()!='P') continue;
          double gammaCh=level->GetFitGamma(ch);
          double gammaChp=levelp->GetFitGamma(ch);
          complex channelShift=point->GetLoElement(j,ch)
        +complex(compound()->GetJGroup(j)->GetChannel(ch)->GetBoundaryCondition(),
             -1.*pow(point->GetSqrtPenetrability(j,ch),2.));
          sum-=gammaCh*gammaChp*channelShift;
          if(la==lap) sum+=gammaCh*gammaChp*level->GetShiftFunction(ch);
          else sum+=gammaCh*gammaChp*
              (level->GetShiftFunction(ch)*(inEnergy-levelp->GetFitE())
               -levelp->GetShiftFunction(ch)*(inEnergy-level->GetFitE()))/
              (level->GetFitE()-levelp->GetFitE());
        }
        tempVector.push_back(sum);
      }
      qMatrixInverse.push_back(tempVector);
    }
    MatrixInv matrixInv(qMatrixInverse);
    for(int ch=1;ch<=compound()->GetJGroup(j)->NumChannels();ch++) {
      for(int chp=1;chp<=compound()->GetJGroup(j)->NumChannels();chp++) {
        complex rTemp(0.0,0.0);;
        for(int la=1;la<=compound()->GetJGroup(j)->NumLevels();la++) {
          if(!compound()->GetJGroup(j)->GetLevel(la)->IsInRMatrix()) continue;
          complex temp(0.0,0.0);
          for(int lap=1;lap<=compound()->GetJGroup(j)->NumLevels();lap++) {
        if(!compound()->GetJGroup(j)->GetLevel(lap)->IsInRMatrix()) continue;

temp+=matrixInv.inverse()[la-1][lap-1]*compound()->GetJGroup(j)->GetLevel(lap)->GetFitGamma(chp);
          }
          rTemp+=temp*compound()->GetJGroup(j)->GetLevel(la)->GetFitGamma(ch);
        }
        this->AddRMatrixElement(j,ch,chp,rTemp);
        double tempPene = 0.;
        if(compound()->GetJGroup(j)->GetChannel(chp)->GetRadType()=='P')
          tempPene=pow(point->GetSqrtPenetrability(j,chp),2.);
        if(ch==chp) this->AddRLMatrixElement(j,ch,chp,1.-complex(0.,1.)*rTemp*tempPene);
        else this->AddRLMatrixElement(j,ch,chp,-complex(0.,1.)*rTemp*tempPene);
      }
    }
      } else {
    for(int ch=1;ch<=compound()->GetJGroup(j)->NumChannels();ch++) {
      for(int chp=1;chp<=compound()->GetJGroup(j)->NumChannels();chp++) {
        complex sum(0.0,0.0);
```

```
        for(int la=1;la<=compound()->GetJGroup(j)->NumLevels();la++) {
          if(compound()->GetJGroup(j)->GetLevel(la)->IsInRMatrix()) {
        ALevel *level=compound()->GetJGroup(j)->GetLevel(la);
        double gammaCh=level->GetFitGamma(ch);
        double gammaChp=level->GetFitGamma(chp);
        double resenergy=level->GetFitE();
        double gammaSum=0.;
        if(configure().paramMask & Config::USE_RMC_FORMALISM)
          for(int chpp=1;chpp<=compound()->GetJGroup(j)->NumChannels();chpp++)
            if(compound()->GetJGroup(j)->GetChannel(chpp)->GetRadType()=='M' ||
               compound()->GetJGroup(j)->GetChannel(chpp)->GetRadType()=='E')
              gammaSum+=pow(compound()->GetJGroup(j)->GetLevel(la)->GetFitGamma(chpp),2.0);
        sum+=gammaCh*gammaChp/(resenergy-inEnergy-complex(0.0,1.0)*gammaSum);
          }
        }
        this->AddRMatrixElement(j,ch,chp,sum);
        complex loElement =point->GetLoElement(j,chp);
        if(ch==chp) this->AddRLMatrixElement(j,ch,chp,1.0-sum*loElement);
        else this->AddRLMatrixElement(j,ch,chp,-sum*loElement);
      }
    }
      }
    }
  }
}


void RMatrixFunc::InvertMatrices() {
  for(int j=1;j<=compound()->NumJGroups();j++) {
    if(compound()->GetJGroup(j)->IsInRMatrix()) {
      matrix_c *theRLMatrix = this->GetJSpecRLMatrix(j);
      MatrixInv matrixInv(*theRLMatrix);
      this->AddRLInvMatrix(matrixInv.inverse());
      for(int ch=1;ch<=compound()->GetJGroup(j)->NumChannels();ch++) {
    for(int chp=1;chp<=compound()->GetJGroup(j)->NumChannels();chp++) {
      complex rlinvrElement(0.,0.);
      for(int chpp=1;chpp<=compound()->GetJGroup(j)->NumChannels();chpp++) {
        rlinvrElement+=this->GetRLInvMatrixElement(j,ch,chpp)*
          this->GetRMatrixElement(j,chpp,chp);
      }
      this->AddRLInvRMatrixElement(j,ch,chp,rlinvrElement);
    }
      }
    }
  }
}

void RMatrixFunc::CalculateTMatrix(EPoint *point) {
  int aa=compound()->GetPairNumFromKey(point->GetEntranceKey());
  int irEnd;
  int irStart;
  bool isRMC=false;
  if((configure().paramMask & Config::USE_RMC_FORMALISM) &&
     compound()->GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->GetPType()==10) {
    irStart=1;
    irEnd=compound()->GetPair(aa)->NumDecays();
    isRMC=true;
  } else {
    irStart=0;
    while(irStart<compound()->GetPair(aa)->NumDecays()) {
      irStart++;
      if(compound()->GetPair(aa)->GetDecay(irStart)->GetPairNum()==
     compound()->GetPairNumFromKey(point->GetExitKey())) break;
    }
    irEnd=irStart;
  }
  for(int ir=irStart;ir<=irEnd;ir++) {
    Decay *theDecay=compound()->GetPair(aa)->GetDecay(ir);
    for(int k=1;k<=theDecay->NumKGroups();k++) {
      for(int m=1;m<=theDecay->GetKGroup(k)->NumMGroups();m++) {
    MGroup *theMGroup=theDecay->GetKGroup(k)->GetMGroup(m);
    JGroup *theJGroup=compound()->GetJGroup(theMGroup->GetJNum());
    AChannel *entranceChannel=theJGroup->GetChannel(theMGroup->GetChNum());
    AChannel *exitChannel=theJGroup->GetChannel(theMGroup->GetChpNum());
    complex uphase=point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChNum())*
      point->GetExpHardSpherePhase(theMGroup->GetJNum(),theMGroup->GetChNum())*
      point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChpNum())*
      point->GetExpHardSpherePhase(theMGroup->GetJNum(),theMGroup->GetChpNum());
    complex umatrix=2.0*std::complex<double>(0.0,1.0)*
      point->GetSqrtPenetrability(theMGroup->GetJNum(),theMGroup->GetChNum())*
      point->GetSqrtPenetrability(theMGroup->GetJNum(),theMGroup->GetChpNum())*
      this->GetRLInvRMatrixElement(theMGroup->GetJNum(),
                       theMGroup->GetChpNum(),
                       theMGroup->GetChNum());
    complex tphase=point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChNum())*
      point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChNum());
    complex tmatrix;
```

```
    if(isRMC) this->AddTMatrixElement(k,m,complex(0.0,-1.0)*umatrix,ir);
    else {
      if(theMGroup->GetChNum()==theMGroup->GetChpNum()) {
        tmatrix=tphase-uphase*(1.0+umatrix);
      } else tmatrix=-uphase*umatrix;
      this->AddTMatrixElement(k,m,tmatrix);
    }
      }
      for(int m=1;m<=theDecay->GetKGroup(k)->NumECMGroups();m++) {
    ECMGroup *theECMGroup=theDecay->GetKGroup(k)->GetECMGroup(m);
    ALevel *finalLevel=compound()->GetJGroup(theECMGroup->GetJGroupNum())
      ->GetLevel(theECMGroup->GetLevelNum());
    double ecNormParam=finalLevel->GetFitGamma(theECMGroup->GetFinalChannel())*
      finalLevel->GetSqrtNFFactor()*finalLevel->GetECConversionFactor(theECMGroup->GetFinalChannel());
    complex tmatrix=ecNormParam*point->GetECAmplitude(k,m);
    if(theECMGroup->IsChannelCapture()) {
      MGroup *chanMGroup=compound()->GetPair(aa)->GetDecay(theECMGroup->GetChanCapDecay())
        ->GetKGroup(theECMGroup->GetChanCapKGroup())->GetMGroup(theECMGroup->GetChanCapMGroup());
      AChannel *chanEntranceChannel=compound()->GetJGroup(chanMGroup->GetJNum())
        ->GetChannel(chanMGroup->GetChNum());
      AChannel *chanExitChannel=compound()->GetJGroup(chanMGroup->GetJNum())
        ->GetChannel(chanMGroup->GetChpNum());
      complex umatrix=2.0*std::complex<double>(0.0,1.0)*
        point->GetSqrtPenetrability(chanMGroup->GetJNum(),chanMGroup->GetChNum())*
        this->GetRLInvRMatrixElement(chanMGroup->GetJNum(),
                     chanMGroup->GetChpNum(),
                     chanMGroup->GetChNum());
      tmatrix=tmatrix*umatrix;
    }
    this->AddECTMatrixElement(k,m,tmatrix);
      }
    }
  }
}

void RMatrixFunc::AddRMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum, complex
matrixElement) {
  matrix_c e;
  vector_c f;
  while(jGroupNum>r_matrices_.size()) r_matrices_.push_back(e);
  while(channelNum>r_matrices_[jGroupNum-1].size()) r_matrices_[jGroupNum-1].push_back(f);
  r_matrices_[jGroupNum-1][channelNum-1].push_back(matrixElement);
  assert(channelPrimeNum=r_matrices_[jGroupNum-1][channelNum-1].size());
}

void RMatrixFunc::AddRLMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum, complex
matrixElement) {
  matrix_c e;
  vector_c f;
  while(jGroupNum>rl_matrices_.size()) rl_matrices_.push_back(e);
  while(channelNum>rl_matrices_[jGroupNum-1].size()) rl_matrices_[jGroupNum-1].push_back(f);
  rl_matrices_[jGroupNum-1][channelNum-1].push_back(matrixElement);
  assert(channelPrimeNum=rl_matrices_[jGroupNum-1][channelNum-1].size());
}

void RMatrixFunc::AddRLInvRMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum, complex
matrixElement) {
  matrix_c e;
  vector_c f;
  while(jGroupNum>rl_inv_r_matrices_.size()) rl_inv_r_matrices_.push_back(e);
  while(channelNum>rl_inv_r_matrices_[jGroupNum-1].size())
rl_inv_r_matrices_[jGroupNum-1].push_back(f);
  rl_inv_r_matrices_[jGroupNum-1][channelNum-1].push_back(matrixElement);
  assert(channelPrimeNum=rl_inv_r_matrices_[jGroupNum-1][channelNum-1].size());
}

void RMatrixFunc::AddRLInvMatrix(matrix_c matrix) {
  rl_inv_matrices_.push_back(matrix);
}
```

8.294 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ShftFunc.cpp File Reference

```
#include "ShftFunc.h"
#include <math.h>
#include <gsl/gsl_deriv.h>
```

8.295 ShftFunc.cpp

Go to the documentation of this file.

```
#include "ShftFunc.h"
#include <math.h>
#include <gsl/gsl_deriv.h>

double ShftFunc::thisShftFunc(double x, void *p) {
  ShftFunc *shift = (ShftFunc*)p;
  int l= shift->params_.lValue;

  return shift->operator()(l,x);
}

double ShftFunc::theWhitFunc(double x, void * p) {
  Params *params =(Params *)p;
  int l = (params->lValue);
  double bindingenergy = (params->bindingEnergy);
  WhitFunc *whitFunc = (params->whitFunc);

  return whitFunc->operator()(l,x,bindingenergy);
}

double ShftFunc::operator()(int l, double energy) {
  params_.bindingEnergy=fabs(energy-totalSepE());
  params_.lValue=l;

  double result;
  double error;

  gsl_function F;
  F.function = &theWhitFunc;
  F.params = &params_;

  gsl_deriv_central (&F, radius(), 1e-4, &result, &error);

  return radius()*result/theWhitFunc(radius(),&params_);
}

double ShftFunc::EnergyDerivative(int l, double energy) {

  double result;
  double error;

  params_.lValue = l;

  gsl_function F;
  F.function = &thisShftFunc;
  F.params = this;

  gsl_deriv_central (&F, energy, 1e-6, &result, &error);

  return result;
}
```

8.296 /Users/kuba/Desktop/R-Matrix/AZURE2/src/TargetEffect.cpp File Reference

```
#include "TargetEffect.h"
#include <sstream>
#include <iostream>
```

8.297 TargetEffect.cpp

Go to the documentation of this file.

```
#include "TargetEffect.h"
#include <sstream>
#include <iostream>

```

```
TargetEffect::TargetEffect(std::istream &stream,const Config& configure) {
  int isActive;
  std::string segmentList;
  int numIntegrationPoints;
  int isConvolution;
  double sigma;
  int isTargetIntegration;
  double density;
  std::string stoppingPowerEq;
  int numParameters;
  vector_r parameters;
  int isQCoefficients;
  int numQCoefficients;
  vector_r qCoefficients;
  stream » isActive » segmentList » numIntegrationPoints » isConvolution
     » sigma » isTargetIntegration » density » stoppingPowerEq
     » numParameters;
  if(!stream.eof()) {
    for(int i=0;i<numParameters;i++) {
      double tempParameter;
      stream » tempParameter;
      parameters.push_back(tempParameter);
    }
    stream » isQCoefficients » numQCoefficients;
    for(int i=0;i<numQCoefficients;i++) {
      double tempQCoefficient;
      stream » tempQCoefficient;
      qCoefficients.push_back(tempQCoefficient);
    }
    isQCoefficients_ = (isQCoefficients==1) ? true : false;
    qCoefficients_=qCoefficients;
    size_t found=0;
    while(found!=std::string::npos) {
      found=segmentList.find('\"');
      if(found!=std::string::npos) segmentList.erase(found,1);
    }
    found=0;
    while(found!=std::string::npos) {
      found=stoppingPowerEq.find('\"');
      if(found!=std::string::npos) stoppingPowerEq.erase(found,1);
    }
    if(isActive==1) isActive_=true;
    else isActive_=false;
    segmentsList_=segmentList;
    numIntegrationPoints_=numIntegrationPoints;
    if(isConvolution==1) isConvolution_=true;
    else isConvolution_=false;
    sigma_=sigma;
    if(isTargetIntegration==1) isTargetIntegration_=true;
    else isTargetIntegration_=false;
    density_=density;
    if(isTargetIntegration_) {
      stoppingPowerEq_.Initialize(stoppingPowerEq,numParameters,configure);
      for(int i=0;i<numParameters;i++) {
    stoppingPowerEq_.SetParameter(i,parameters[i],configure);
      }
    }
  }
}

bool TargetEffect::IsActive() const {
  return isActive_;
}

bool TargetEffect::IsConvolution() const {
  return isConvolution_;
}

bool TargetEffect::IsTargetIntegration() const {
  return isTargetIntegration_;
}

bool TargetEffect::IsQCoefficients() const {
  return isQCoefficients_;
}

int TargetEffect::NumSubPoints() const {
  return numIntegrationPoints_;
}

int TargetEffect::NumQCoefficients() const {
  return qCoefficients_.size();
}

double TargetEffect::GetSigma() const {
  return sigma_;
}
```

```
double TargetEffect::GetDensity() const {
  return density_;
}

double TargetEffect::TargetThickness(double energy, const Config& configure)  {
  return this->GetStoppingPowerEq()->Evaluate(configure,energy)*this->GetDensity();
}

double TargetEffect::GetQCoefficient(int order)  const {
  return (qCoefficients_.size()>order) ? qCoefficients_[order] : 1.;
}

void TargetEffect::SetSigma(double sigma) {
  sigma_=sigma;
}

void TargetEffect::SetNumSubPoints(int numPoints) {
  numIntegrationPoints_=numPoints;
}

std::vector<int> TargetEffect::GetSegmentsList() const {
  std::vector<int> tempList;
  int i=0;
  int lastSegNum=0;
  bool inclusive=false;
  while(i<segmentsList_.length()) {
    if(segmentsList_[i]>=’0’&&segmentsList_[i]<=’9’) {
      std::string tempString;
      while(segmentsList_[i]!=’,’&&segmentsList_[i]!=’-’&&
        i<segmentsList_.length()) {
    tempString+=segmentsList_[i];
    i++;
      }
      std::istringstream stm;
      stm.str(tempString);
      int tempSegNum;stm»tempSegNum;
      if(inclusive==true) for(int j=lastSegNum+1;j<=tempSegNum;j++)
                tempList.push_back(j);
      else tempList.push_back(tempSegNum);
      lastSegNum=tempSegNum;
    }
    if(segmentsList_[i]==’-’) inclusive=true;
    else inclusive =false;
    i++;
  }
  return tempList;
}

Equation *TargetEffect::GetStoppingPowerEq() {
  Equation *tempEquation;
  tempEquation=&stoppingPowerEq_;
  return tempEquation;
}

double TargetEffect::GetConvolutionFactor(double energy, double centroid) const {
  double sigma=this->GetSigma();
  return pow(2.*pi,-0.5)/sigma*exp(-pow(energy-centroid,2.0)/2.0/pow(sigma,2.0));
}
```

Index