

AZURE2

Generated by Doxygen 1.9.7

1 AZURE2	1
1.1 Dependencies	1
1.2 Build	2
1.2.1 Basic Building	2
1.2.2 Options	2
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	9
4.1 Class List	9
5 File Index	13
5.1 File List	13
6 Namespace Documentation	17
6.1 ROOT Namespace Reference	17
6.2 ROOT::Minuit2 Namespace Reference	17
7 Class Documentation	19
7.1 AboutAZURE2Dialog Class Reference	19
7.1.1 Detailed Description	19
7.1.2 Constructor & Destructor Documentation	19
7.1.2.1 AboutAZURE2Dialog()	19
7.2 AChannel Class Reference	20
7.2.1 Detailed Description	20
7.2.2 Constructor & Destructor Documentation	20
7.2.2.1 AChannel() [1/2]	20
7.2.2.2 AChannel() [2/2]	20
7.2.3 Member Function Documentation	21
7.2.3.1 GetBoundaryCondition()	21
7.2.3.2 GetL()	21
7.2.3.3 GetPairNum()	21
7.2.3.4 GetRadType()	21
7.2.3.5 GetS()	21
7.2.3.6 SetBoundaryCondition()	22
7.3 AddLevelDialog Class Reference	22
7.3.1 Detailed Description	22
7.3.2 Constructor & Destructor Documentation	22
7.3.2.1 AddLevelDialog()	22
7.3.3 Member Data Documentation	23

7.3.3.1 energyText	23
7.3.3.2 jValueText	23
7.3.3.3 piValueCombo	23
7.4 AddPairDialog Class Reference	23
7.4.1 Detailed Description	24
7.4.2 Constructor & Destructor Documentation	24
7.4.2.1 AddPairDialog()	24
7.4.3 Member Function Documentation	24
7.4.3.1 updateLightParticle	24
7.4.4 Member Data Documentation	24
7.4.4.1 channelRadiusText	24
7.4.4.2 e1Check	25
7.4.4.3 e2Check	25
7.4.4.4 excitationEnergyText	25
7.4.4.5 heavyJText	25
7.4.4.6 heavyMText	25
7.4.4.7 heavyPiCombo	25
7.4.4.8 heavyZText	25
7.4.4.9 lightJText	25
7.4.4.10 lightMText	26
7.4.4.11 lightPiCombo	26
7.4.4.12 lightZText	26
7.4.4.13 multBox	26
7.4.4.14 pairTypeCombo	26
7.4.4.15 seperationEnergyText	26
7.5 AddSegDataDialog Class Reference	27
7.5.1 Detailed Description	27
7.5.2 Constructor & Destructor Documentation	28
7.5.2.1 AddSegDataDialog()	28
7.5.3 Member Function Documentation	28
7.5.3.1 dataTypeChanged	28
7.5.3.2 setChooseFile	28
7.5.3.3 varyNormChanged	28
7.5.4 Member Data Documentation	28
7.5.4.1 dataFileText	28
7.5.4.2 dataNormErrorLabel	28
7.5.4.3 dataNormErrorText	29
7.5.4.4 dataNormText	29
7.5.4.5 dataTypeCombo	29
7.5.4.6 entrancePairIndexSpin	29
7.5.4.7 exitPairIndexSpin	29
7.5.4.8 highAngleText	29

7.5.4.9 highEnergyText	29
7.5.4.10 lowAngleText	29
7.5.4.11 lowEnergyText	30
7.5.4.12 phaseJValueLabel	30
7.5.4.13 phaseJValueText	30
7.5.4.14 phaseLValueLabel	30
7.5.4.15 phaseLValueText	30
7.5.4.16 totalCaptureLabel	30
7.5.4.17 varyNormCheck	30
7.6 AddSegTestDialog Class Reference	31
7.6.1 Detailed Description	31
7.6.2 Constructor & Destructor Documentation	31
7.6.2.1 AddSegTestDialog()	31
7.6.3 Member Function Documentation	32
7.6.3.1 dataTypeChanged	32
7.6.4 Member Data Documentation	32
7.6.4.1 angDistLabel	32
7.6.4.2 angDistSpin	32
7.6.4.3 angleStepText	32
7.6.4.4 dataTypeCombo	32
7.6.4.5 energyStepText	32
7.6.4.6 entrancePairIndexSpin	32
7.6.4.7 exitPairIndexSpin	33
7.6.4.8 highAngleText	33
7.6.4.9 highEnergyText	33
7.6.4.10 lowAngleText	33
7.6.4.11 lowEnergyText	33
7.6.4.12 phaseJValueLabel	33
7.6.4.13 phaseJValueText	33
7.6.4.14 phaseLValueLabel	33
7.6.4.15 phaseLValueText	34
7.6.4.16 totalCaptureLabel	34
7.7 AddTargetIntDialog Class Reference	34
7.7.1 Detailed Description	35
7.7.2 Constructor & Destructor Documentation	35
7.7.2.1 AddTargetIntDialog()	35
7.7.3 Member Function Documentation	35
7.7.3.1 convolutionCheckChanged	35
7.7.3.2 createParameterItem()	35
7.7.3.3 createQCoefficientItem()	36
7.7.3.4 parameterChanged	36
7.7.3.5 parameterSpinChanged	36

7.7.3.6 qCoefficientChanged	36
7.7.3.7 qCoefficientCheckChanged	36
7.7.3.8 qCoefficientSpinChanged	36
7.7.3.9 targetIntCheckChanged	37
7.7.4 Member Data Documentation	37
7.7.4.1 densityText	37
7.7.4.2 isConvolutionCheck	37
7.7.4.3 isQCoefficientCheck	37
7.7.4.4 isTargetIntegrationCheck	37
7.7.4.5 numParametersSpin	37
7.7.4.6 numPointsSpin	37
7.7.4.7 numQCoefficientSpin	38
7.7.4.8 parametersTable	38
7.7.4.9 qCoefficientTable	38
7.7.4.10 segmentsListText	38
7.7.4.11 sigmaText	38
7.7.4.12 stoppingPowerEqText	38
7.7.4.13 tempParameters	38
7.7.4.14 tempQCoefficients	39
7.8 ALevel Class Reference	39
7.8.1 Detailed Description	40
7.8.2 Constructor & Destructor Documentation	40
7.8.2.1 ALevel() [1/2]	40
7.8.2.2 ALevel() [2/2]	40
7.8.3 Member Function Documentation	40
7.8.3.1 AddECConversionFactor()	40
7.8.3.2 AddGamma() [1/2]	40
7.8.3.3 AddGamma() [2/2]	41
7.8.3.4 AddNFIntegral()	41
7.8.3.5 ChannelFixed()	41
7.8.3.6 EnergyFixed()	41
7.8.3.7 GetBigGamma()	41
7.8.3.8 GetE()	42
7.8.3.9 GetECConversionFactor()	42
7.8.3.10 GetECMultMask()	42
7.8.3.11 GetECPairNum()	42
7.8.3.12 GetExternalGamma()	42
7.8.3.13 GetFitE()	42
7.8.3.14 GetFitGamma()	43
7.8.3.15 GetGamma()	43
7.8.3.16 GetNFIntegral()	43
7.8.3.17 GetShiftFunction()	43

7.8.3.18 GetSqrtNFFactor()	43
7.8.3.19 GetTransformE()	43
7.8.3.20 GetTransformGamma()	44
7.8.3.21 GetTransformIterations()	44
7.8.3.22 IsECLLevel()	44
7.8.3.23 IsInRMatrix()	44
7.8.3.24 NumNFIntegrals()	44
7.8.3.25 SetBigGamma()	44
7.8.3.26 SetE()	45
7.8.3.27 SetECPParams()	45
7.8.3.28 SetExternalGamma()	45
7.8.3.29 SetFitE()	45
7.8.3.30 SetFitGamma()	45
7.8.3.31 SetGamma()	46
7.8.3.32 SetShiftFunction()	46
7.8.3.33 SetSqrtNFFactor()	46
7.8.3.34 SetTransformE()	46
7.8.3.35 SetTransformGamma()	46
7.8.3.36 SetTransformIterations()	47
7.9 AMatrixFunc Class Reference	47
7.9.1 Detailed Description	48
7.9.2 Constructor & Destructor Documentation	48
7.9.2.1 AMatrixFunc()	48
7.9.3 Member Function Documentation	49
7.9.3.1 AddAInvMatrixElement()	49
7.9.3.2 AddAMatrix()	49
7.9.3.3 CalculateCrossSection()	49
7.9.3.4 CalculateTMatrix()	49
7.9.3.5 ClearMatrices()	49
7.9.3.6 compound()	50
7.9.3.7 configure()	50
7.9.3.8 FillMatrices()	50
7.9.3.9 GetAMatrixElement()	50
7.9.3.10 GetJSpecAInvMatrix()	50
7.9.3.11 InvertMatrices()	51
7.10 AngCoeff Class Reference	51
7.10.1 Detailed Description	51
7.10.2 Member Function Documentation	51
7.10.2.1 ClebGord()	51
7.10.2.2 Racah()	52
7.11 AZURECalc Class Reference	52
7.11.1 Detailed Description	52

7.11.2 Constructor & Destructor Documentation	53
7.11.2.1 AZURECalc()	53
7.11.2.2 ~AZURECalc()	53
7.11.3 Member Function Documentation	53
7.11.3.1 compound()	53
7.11.3.2 configure()	53
7.11.3.3 data()	53
7.11.3.4 operator()()	54
7.11.3.5 SetErrorDef()	54
7.11.3.6 Up()	54
7.12 AZUREFBuffer Class Reference	54
7.12.1 Detailed Description	55
7.12.2 Constructor & Destructor Documentation	55
7.12.2.1 AZUREFBuffer()	55
7.12.2.2 ~AZUREFBuffer()	55
7.12.3 Member Function Documentation	55
7.12.3.1 GetEntranceKey()	55
7.12.3.2 GetExitKey()	55
7.12.3.3 GetFBuffer()	56
7.12.3.4 IsAngDist()	56
7.13 AZUREMain Class Reference	56
7.13.1 Detailed Description	56
7.13.2 Constructor & Destructor Documentation	57
7.13.2.1 AZUREMain()	57
7.13.2.2 ~AZUREMain()	57
7.13.3 Member Function Documentation	57
7.13.3.1 compound()	57
7.13.3.2 configure()	57
7.13.3.3 data()	57
7.13.3.4 operator()()	58
7.14 AZUREMainThread Class Reference	58
7.14.1 Detailed Description	58
7.14.2 Constructor & Destructor Documentation	59
7.14.2.1 AZUREMainThread()	59
7.14.3 Member Function Documentation	59
7.14.3.1 configure()	59
7.14.3.2 readyToRun	59
7.14.3.3 run()	59
7.14.3.4 stopAZURE	59
7.15 AZUREMainThreadWorker Class Reference	59
7.15.1 Detailed Description	60
7.15.2 Constructor & Destructor Documentation	60

7.15.2.1 AZUREMainThreadWorker()	60
7.15.3 Member Function Documentation	60
7.15.3.1 done	60
7.15.3.2 run	60
7.16 AZUREOutput Class Reference	60
7.16.1 Detailed Description	61
7.16.2 Constructor & Destructor Documentation	61
7.16.2.1 AZUREOutput()	61
7.16.2.2 ~AZUREOutput()	61
7.16.3 Member Function Documentation	61
7.16.3.1 AddAZUREFBuffer()	61
7.16.3.2 GetAZUREFBuffer()	62
7.16.3.3 GetOutputDir()	62
7.16.3.4 IsAZUREFBuffer()	62
7.16.3.5 IsExtrap()	62
7.16.3.6 NumAZUREFBuffers()	62
7.16.3.7 operator>()	63
7.16.3.8 SetExtrap()	63
7.17 AZUREParams Class Reference	63
7.17.1 Detailed Description	63
7.17.2 Member Function Documentation	64
7.17.2.1 GetMinuitParams()	64
7.17.2.2 ReadUserParameters()	64
7.17.2.3 WriteParameterErrors()	64
7.17.2.4 WriteUserParameters()	64
7.18 AZUREPlot Class Reference	65
7.18.1 Detailed Description	65
7.18.2 Constructor & Destructor Documentation	65
7.18.2.1 AZUREPlot()	65
7.18.3 Member Function Documentation	65
7.18.3.1 clearEntries()	65
7.18.3.2 draw	66
7.18.3.3 exportPlot	66
7.18.3.4 print	66
7.18.3.5 setXAxisLog()	66
7.18.3.6 setXAxisType()	66
7.18.3.7 setYAxisLog()	66
7.18.3.8 setYAxisType()	66
7.18.3.9 update	67
7.19 AZURESetup Class Reference	67
7.19.1 Detailed Description	67
7.19.2 Constructor & Destructor Documentation	67

7.19.2.1 AZURESetup()	67
7.19.3 Member Function Documentation	68
7.19.3.1 DeleteThread	68
7.19.3.2 GetConfig()	68
7.19.3.3 open()	68
7.19.3.4 SaveAndRun	68
7.20 AZUREZoomer Class Reference	68
7.20.1 Detailed Description	69
7.20.2 Constructor & Destructor Documentation	69
7.20.2.1 AZUREZoomer()	69
7.20.3 Member Function Documentation	69
7.20.3.1 trackerTextF()	69
7.21 ChannelDetails Class Reference	69
7.21.1 Detailed Description	70
7.21.2 Constructor & Destructor Documentation	70
7.21.2.1 ChannelDetails()	70
7.21.3 Member Function Documentation	70
7.21.3.1 setNormParam()	70
7.21.4 Member Data Documentation	70
7.21.4.1 details	70
7.21.4.2 reducedWidthText	70
7.22 ChannelsData Struct Reference	71
7.22.1 Detailed Description	71
7.22.2 Member Data Documentation	71
7.22.2.1 isFixed	71
7.22.2.2 levelIndex	71
7.22.2.3 IValue	71
7.22.2.4 pairIndex	71
7.22.2.5 radType	72
7.22.2.6 reducedWidth	72
7.22.2.7 SIZE	72
7.22.2.8 sValue	72
7.23 ChannelsModel Class Reference	72
7.23.1 Detailed Description	73
7.23.2 Constructor & Destructor Documentation	73
7.23.2.1 ChannelsModel()	73
7.23.3 Member Function Documentation	73
7.23.3.1 columnCount()	73
7.23.3.2 data()	73
7.23.3.3 flags()	74
7.23.3.4 getChannels()	74
7.23.3.5 getSpinLabel()	74

7.23.3.6 headerData()	74
7.23.3.7 insertRows()	74
7.23.3.8 isChannel()	74
7.23.3.9 removeRows()	75
7.23.3.10 rowCount()	75
7.23.3.11 setData()	75
7.23.3.12 setPairsModel()	75
7.24 ChooseFileButton Class Reference	75
7.24.1 Detailed Description	76
7.24.2 Constructor & Destructor Documentation	76
7.24.2.1 ChooseFileButton()	76
7.24.3 Member Function Documentation	76
7.24.3.1 click	76
7.24.3.2 clicked	76
7.24.3.3 setLineEdit()	77
7.25 CNuc Class Reference	77
7.25.1 Detailed Description	78
7.25.2 Member Function Documentation	78
7.25.2.1 AddJGroup()	78
7.25.2.2 AddPair()	78
7.25.2.3 CalcAngularDists()	78
7.25.2.4 CalcBoundaryConditions()	78
7.25.2.5 CalcExternalWidth()	79
7.25.2.6 CalcShiftFunctions()	79
7.25.2.7 Clone()	79
7.25.2.8 Fill()	79
7.25.2.9 FillCompoundFromParams()	79
7.25.2.10 FillMnParams()	80
7.25.2.11 GetJGroup()	80
7.25.2.12 GetMaxLValue()	80
7.25.2.13 GetPair()	80
7.25.2.14 GetPairNumFromKey()	80
7.25.2.15 Initialize()	81
7.25.2.16 IsJGroup()	81
7.25.2.17 IsPair()	81
7.25.2.18 IsPairKey()	81
7.25.2.19 NumJGroups()	81
7.25.2.20 NumPairs()	82
7.25.2.21 ParseExternalCapture()	82
7.25.2.22 PrintAngularDists()	82
7.25.2.23 PrintBoundaryConditions()	82
7.25.2.24 PrintNuc()	82

7.25.2.25 PrintPathways()	83
7.25.2.26 PrintTransformParams()	83
7.25.2.27 SetMaxLValue()	83
7.25.2.28 SortPathways()	83
7.25.2.29 TransformIn()	83
7.25.2.30 TransformOut()	84
7.26 Config Class Reference	84
7.26.1 Detailed Description	85
7.26.2 Member Enumeration Documentation	85
7.26.2.1 CheckFileFlags	85
7.26.2.2 ParameterFlags	86
7.26.3 Constructor & Destructor Documentation	86
7.26.3.1 Config()	86
7.26.4 Member Function Documentation	87
7.26.4.1 CheckForInputFiles()	87
7.26.4.2 ReadConfigFile()	87
7.26.4.3 Reset()	87
7.26.5 Member Data Documentation	87
7.26.5.1 checkdir	87
7.26.5.2 chiVariance	87
7.26.5.3 configfile	88
7.26.5.4 fileCheckMask	88
7.26.5.5 integralsfile	88
7.26.5.6 maxLOrder	88
7.26.5.7 outputdir	88
7.26.5.8 outStream	88
7.26.5.9 paramfile	89
7.26.5.10 paramMask	89
7.26.5.11 rateParams	89
7.26.5.12 screenCheckMask	89
7.26.5.13 stopFlag	89
7.27 CoulFunc Class Reference	89
7.27.1 Detailed Description	90
7.27.2 Constructor & Destructor Documentation	90
7.27.2.1 CoulFunc()	90
7.27.3 Member Function Documentation	90
7.27.3.1 coulLast()	90
7.27.3.2 energyLast()	91
7.27.3.3 lLast()	91
7.27.3.4 operator>()	91
7.27.3.5 Penetrability()	91
7.27.3.6 PEShift()	91

7.27.3.7 PShift_dE()	92
7.27.3.8 radiusLast()	92
7.27.3.9 redmass()	92
7.27.3.10 setLast()	92
7.27.3.11 z1()	92
7.27.3.12 z2()	93
7.28 Coulomb_wave_functions Class Reference	93
7.28.1 Detailed Description	93
7.28.2 Constructor & Destructor Documentation	93
7.28.2.1 Coulomb_wave_functions()	93
7.28.2.2 ~Coulomb_wave_functions()	94
7.28.3 Member Function Documentation	94
7.28.3.1 F_dF()	94
7.28.3.2 F_dF_init()	94
7.28.3.3 G_dG()	94
7.28.3.4 H_dH()	94
7.28.3.5 H_dH_scaled()	95
7.28.4 Member Data Documentation	95
7.28.4.1 eta	95
7.28.4.2 is_it_normalized	95
7.28.4.3 l	95
7.29 CoulWaves Struct Reference	95
7.29.1 Detailed Description	96
7.29.2 Member Data Documentation	96
7.29.2.1 dF	96
7.29.2.2 dG	96
7.29.2.3 F	96
7.29.2.4 G	96
7.30 DataLine Class Reference	97
7.30.1 Detailed Description	97
7.30.2 Constructor & Destructor Documentation	97
7.30.2.1 DataLine()	97
7.30.3 Member Function Documentation	97
7.30.3.1 angle()	97
7.30.3.2 crossSection()	97
7.30.3.3 energy()	98
7.30.3.4 error()	98
7.31 Decay Class Reference	98
7.31.1 Detailed Description	98
7.31.2 Constructor & Destructor Documentation	99
7.31.2.1 Decay()	99
7.31.3 Member Function Documentation	99

7.31.3.1 AddKGroup()	99
7.31.3.2 AddKLGroup()	99
7.31.3.3 GetKGroup()	99
7.31.3.4 GetKLGroup()	99
7.31.3.5 GetPairNum()	100
7.31.3.6 IsKGroup()	100
7.31.3.7 IsKLGroup()	100
7.31.3.8 NumKGroups()	100
7.31.3.9 NumKLGroups()	100
7.32 Directories Class Reference	101
7.32.1 Detailed Description	101
7.32.2 Constructor & Destructor Documentation	101
7.32.2.1 Directories()	101
7.32.3 Member Data Documentation	101
7.32.3.1 checksDir	101
7.32.3.2 outputDir	101
7.33 ECIntegral Class Reference	102
7.33.1 Detailed Description	102
7.33.2 Constructor & Destructor Documentation	102
7.33.2.1 ECIntegral()	102
7.33.2.2 ~ECIntegral()	102
7.33.3 Member Function Documentation	103
7.33.3.1 operator>()	103
7.34 ECMGroup Class Reference	103
7.34.1 Detailed Description	104
7.34.2 Constructor & Destructor Documentation	104
7.34.2.1 ECMGroup() [1/2]	104
7.34.2.2 ECMGroup() [2/2]	104
7.34.3 Member Function Documentation	105
7.34.3.1 GetChanCapDecay()	105
7.34.3.2 GetChanCapKGroup()	105
7.34.3.3 GetChanCapMGroup()	105
7.34.3.4 GetFinalChannel()	105
7.34.3.5 GetIntChannelNum()	105
7.34.3.6 GetJ()	105
7.34.3.7 GetJGroupNum()	106
7.34.3.8 GetL()	106
7.34.3.9 GetLevelNum()	106
7.34.3.10 GetMult()	106
7.34.3.11 GetRadType()	106
7.34.3.12 GetStatSpinFactor()	106
7.34.3.13 IsChannelCapture()	107

7.34.3.14 SetStatSpinFactor()	107
7.35 EData Class Reference	107
7.35.1 Detailed Description	108
7.35.2 Constructor & Destructor Documentation	108
7.35.2.1 EData()	108
7.35.3 Member Function Documentation	108
7.35.3.1 AddSegment()	108
7.35.3.2 AddTargetEffect()	109
7.35.3.3 begin()	109
7.35.3.4 CalcCoulombAmplitude()	109
7.35.3.5 CalcEDependentValues()	109
7.35.3.6 CalcLegendreP()	109
7.35.3.7 CalculateECAmplitudes()	110
7.35.3.8 Clone()	110
7.35.3.9 DeleteLastSegment()	110
7.35.3.10 end()	110
7.35.3.11 Fill()	110
7.35.3.12 FillMnParams()	111
7.35.3.13 FillNormsFromParams()	111
7.35.3.14 GetNormParamOffset()	111
7.35.3.15 GetSegment()	111
7.35.3.16 GetSegmentFromKey()	111
7.35.3.17 GetSegments()	111
7.35.3.18 GetTargetEffect()	112
7.35.3.19 Initialize()	112
7.35.3.20 IsErrorAnalysis()	112
7.35.3.21 IsFit()	112
7.35.3.22 IsSegmentKey()	112
7.35.3.23 Iterate()	113
7.35.3.24 Iterations()	113
7.35.3.25 MakePoints()	113
7.35.3.26 MapData()	113
7.35.3.27 NumSegments()	113
7.35.3.28 NumTargetEffects()	113
7.35.3.29 PrintCoulombAmplitude()	114
7.35.3.30 PrintData()	114
7.35.3.31 PrintEDependentValues()	114
7.35.3.32 PrintLegendreP()	114
7.35.3.33 ReadTargetEffectsFile()	114
7.35.3.34 ResetIterations()	115
7.35.3.35 SetErrorAnalysis()	115
7.35.3.36 SetFit()	115

7.35.3.37 SetNormParamOffset()	115
7.35.3.38 WriteOutputFiles()	115
7.36 EDatalterator Class Reference	116
7.36.1 Detailed Description	116
7.36.2 Constructor & Destructor Documentation	116
7.36.2.1 EDatalterator() [1/2]	116
7.36.2.2 EDatalterator() [2/2]	116
7.36.3 Member Function Documentation	117
7.36.3.1 operator!=(())	117
7.36.3.2 operator++() [1/2]	117
7.36.3.3 operator++() [2/2]	117
7.36.3.4 operator==(())	117
7.36.3.5 point()	117
7.36.3.6 segment()	118
7.36.3.7 SetEnd()	118
7.37 EditChecksDialog Class Reference	118
7.37.1 Detailed Description	119
7.37.2 Constructor & Destructor Documentation	119
7.37.2.1 EditChecksDialog()	119
7.37.3 Member Data Documentation	119
7.37.3.1 angDistsCheckCombo	119
7.37.3.2 boundaryCheckCombo	119
7.37.3.3 compoundCheckCombo	119
7.37.3.4 coulAmpCheckCombo	119
7.37.3.5 dataCheckCombo	119
7.37.3.6 legendreCheckCombo	120
7.37.3.7 lMatrixCheckCombo	120
7.37.3.8 pathwaysCheckCombo	120
7.38 EditDirsDialog Class Reference	120
7.38.1 Detailed Description	120
7.38.2 Constructor & Destructor Documentation	121
7.38.2.1 EditDirsDialog()	121
7.38.3 Member Data Documentation	121
7.38.3.1 checksDirectoryText	121
7.38.3.2 outputDirectoryText	121
7.39 EditOptionsDialog Class Reference	121
7.39.1 Detailed Description	122
7.39.2 Constructor & Destructor Documentation	122
7.39.2.1 EditOptionsDialog()	122
7.39.3 Member Data Documentation	122
7.39.3.1 ignoreExternalsCheck	122
7.39.3.2 noTransformCheck	122

7.39.3.3 useBruneCheck	122
7.39.3.4 useGSLCoulCheck	122
7.39.3.5 useRMCCheck	123
7.40 EffectiveCharge Class Reference	123
7.40.1 Detailed Description	123
7.40.2 Constructor & Destructor Documentation	123
7.40.2.1 EffectiveCharge()	123
7.40.3 Member Function Documentation	124
7.40.3.1 operator>()	124
7.41 EigenFunc Class Reference	124
7.41.1 Detailed Description	124
7.41.2 Constructor & Destructor Documentation	124
7.41.2.1 EigenFunc() [1/2]	124
7.41.2.2 EigenFunc() [2/2]	125
7.41.3 Member Function Documentation	125
7.41.3.1 eigenvalues()	125
7.41.3.2 eigenvectors()	125
7.42 EnergyMap Struct Reference	125
7.42.1 Detailed Description	126
7.42.2 Member Data Documentation	126
7.42.2.1 point	126
7.42.2.2 segment	126
7.43 EPoint Class Reference	126
7.43.1 Detailed Description	128
7.43.2 Constructor & Destructor Documentation	128
7.43.2.1 EPoint() [1/3]	128
7.43.2.2 EPoint() [2/3]	129
7.43.2.3 EPoint() [3/3]	129
7.43.3 Member Function Documentation	129
7.43.3.1 AddECAmplitude()	129
7.43.3.2 AddExpCoulombPhase()	129
7.43.3.3 AddExpHardSpherePhase()	130
7.43.3.4 AddLegendreP()	130
7.43.3.5 AddLocalMappedPoint()	130
7.43.3.6 AddLoElement()	130
7.43.3.7 AddSqrtPenetrability()	130
7.43.3.8 AddSubPoint()	131
7.43.3.9 CalcCoulombAmplitude()	131
7.43.3.10 CalcEDependentValues()	131
7.43.3.11 CalcLegendreP()	131
7.43.3.12 Calculate()	131
7.43.3.13 CalculateECAmplitudes()	132

7.43.3.14 ClearLocalMappedPoints()	132
7.43.3.15 ConvertCrossSection()	132
7.43.3.16 ConvertDecayEnergy()	132
7.43.3.17 ConvertLabAngle() [1/2]	132
7.43.3.18 ConvertLabAngle() [2/2]	133
7.43.3.19 ConvertLabEnergy()	133
7.43.3.20 GetAngularDist()	133
7.43.3.21 GetCMAngle()	133
7.43.3.22 GetCMCrossSection()	133
7.43.3.23 GetCMCrossSectionError()	134
7.43.3.24 GetCMEnergy()	134
7.43.3.25 GetCoulombAmplitude()	134
7.43.3.26 GetECAmplitude()	134
7.43.3.27 GetEntranceKey()	134
7.43.3.28 GetExcitationEnergy()	134
7.43.3.29 GetExitKey()	135
7.43.3.30 GetExpCoulombPhase()	135
7.43.3.31 GetExpHardSpherePhase()	135
7.43.3.32 GetFitCrossSection()	135
7.43.3.33 GetGeometricalFactor()	135
7.43.3.34 GetJ()	136
7.43.3.35 GetL()	136
7.43.3.36 GetLabAngle()	136
7.43.3.37 GetLabCrossSection()	136
7.43.3.38 GetLabCrossSectionError()	136
7.43.3.39 GetLabEnergy()	136
7.43.3.40 GetLegendreP()	137
7.43.3.41 GetLocalMappedPoint()	137
7.43.3.42 GetLoElement()	137
7.43.3.43 GetMap()	137
7.43.3.44 GetMappedPoints()	137
7.43.3.45 GetMaxAngDistOrder()	138
7.43.3.46 GetMaxLOrder()	138
7.43.3.47 GetNumAngularDists()	138
7.43.3.48 GetParentData()	138
7.43.3.49 GetSFactorConversion()	138
7.43.3.50 GetSqrtPenetrability()	138
7.43.3.51 GetStoppingPower()	139
7.43.3.52 GetSubPoint()	139
7.43.3.53 GetSubPoints()	139
7.43.3.54 GetTargetEffectNum()	139
7.43.3.55 GetTargetThickness()	139

7.43.3.56 Initialize()	139
7.43.3.57 IntegrateTargetEffect()	140
7.43.3.58 IsAngularDist()	140
7.43.3.59 IsDifferential()	140
7.43.3.60 IsMapped()	140
7.43.3.61 IsPhase()	140
7.43.3.62 IsTargetEffect()	140
7.43.3.63 NumLocalMappedPoints()	141
7.43.3.64 NumSubPoints()	141
7.43.3.65 SetAngularDists()	141
7.43.3.66 SetCoulombAmplitude()	141
7.43.3.67 SetExitKey()	141
7.43.3.68 SetFitCrossSection()	141
7.43.3.69 SetGeometricalFactor()	142
7.43.3.70 SetMap()	142
7.43.3.71 SetParentData()	142
7.43.3.72 SetSFactorConversion()	142
7.43.3.73 SetStoppingPower()	142
7.43.3.74 SetTargetEffectNum()	143
7.43.3.75 SetTargetThickness()	143
7.44 Equation Class Reference	143
7.44.1 Detailed Description	143
7.44.2 Constructor & Destructor Documentation	144
7.44.2.1 Equation() [1/4]	144
7.44.2.2 Equation() [2/4]	144
7.44.2.3 Equation() [3/4]	144
7.44.2.4 Equation() [4/4]	144
7.44.3 Member Function Documentation	145
7.44.3.1 Evaluate()	145
7.44.3.2 GetParameters()	145
7.44.3.3 Initialize()	145
7.44.3.4 SetParameter()	145
7.45 ESegment Class Reference	146
7.45.1 Detailed Description	147
7.45.2 Constructor & Destructor Documentation	147
7.45.2.1 ESegment() [1/2]	147
7.45.2.2 ESegment() [2/2]	147
7.45.3 Member Function Documentation	147
7.45.3.1 AddPoint()	147
7.45.3.2 Fill()	147
7.45.3.3 GetAStep()	148
7.45.3.4 GetDataFile()	148

7.45.3.5 GetEntranceKey()	148
7.45.3.6 GetEStep()	148
7.45.3.7 GetExitKey()	148
7.45.3.8 GetJ()	148
7.45.3.9 GetL()	149
7.45.3.10 GetMaxAngDistOrder()	149
7.45.3.11 GetMaxAngle()	149
7.45.3.12 GetMaxEnergy()	149
7.45.3.13 GetMinAngle()	149
7.45.3.14 GetMinEnergy()	149
7.45.3.15 GetNominalNorm()	150
7.45.3.16 GetNorm()	150
7.45.3.17 GetNormError()	150
7.45.3.18 GetPoint()	150
7.45.3.19 GetPoints()	150
7.45.3.20 GetSegmentChiSquared()	150
7.45.3.21 GetSegmentKey()	151
7.45.3.22 GetTargetEffectNum()	151
7.45.3.23 IsAngularDist()	151
7.45.3.24 IsDifferential()	151
7.45.3.25 IsInSegment()	151
7.45.3.26 IsPhase()	151
7.45.3.27 IsTargetEffect()	152
7.45.3.28 IsTotalCapture()	152
7.45.3.29 IsVaryNorm()	152
7.45.3.30 NumPoints()	152
7.45.3.31 SetExitKey()	152
7.45.3.32 SetIsTotalCapture()	152
7.45.3.33 SetNorm()	153
7.45.3.34 SetSegmentChiSquared()	153
7.45.3.35 SetSegmentKey()	153
7.45.3.36 SetTargetEffectNum()	153
7.45.3.37 SetVaryNorm()	153
7.46 ExtrapolLine Class Reference	154
7.46.1 Detailed Description	154
7.46.2 Constructor & Destructor Documentation	154
7.46.2.1 ExtrapolLine()	154
7.46.3 Member Function Documentation	154
7.46.3.1 aStep()	154
7.46.3.2 entranceKey()	155
7.46.3.3 eStep()	155
7.46.3.4 exitKey()	155

7.46.3.5 isActive()	155
7.46.3.6 isDiff()	155
7.46.3.7 maxA()	155
7.46.3.8 maxAngDistOrder()	156
7.46.3.9 maxE()	156
7.46.3.10 minA()	156
7.46.3.11 minE()	156
7.46.3.12 phaseJ()	156
7.46.3.13 phaseL()	156
7.47 FilteredTextEdit Class Reference	157
7.47.1 Detailed Description	157
7.47.2 Constructor & Destructor Documentation	157
7.47.2.1 FilteredTextEdit()	157
7.47.3 Member Function Documentation	157
7.47.3.1 IsMouseFiltered()	157
7.47.3.2 mouseDoubleClickEvent()	158
7.47.3.3 mousePressEvent()	158
7.47.3.4 SetMouseFiltered()	158
7.47.3.5 write	158
7.48 GenericFunction Class Reference	158
7.48.1 Detailed Description	159
7.48.2 Constructor & Destructor Documentation	159
7.48.2.1 GenericFunction() [1/2]	159
7.48.2.2 GenericFunction() [2/2]	159
7.48.3 Member Function Documentation	159
7.48.3.1 Evaluate()	159
7.49 GenMatrixFunc Class Reference	160
7.49.1 Detailed Description	160
7.49.2 Constructor & Destructor Documentation	161
7.49.2.1 GenMatrixFunc()	161
7.49.2.2 ~GenMatrixFunc()	161
7.49.3 Member Function Documentation	161
7.49.3.1 AddECTMatrixElement()	161
7.49.3.2 AddTMatrixElement()	161
7.49.3.3 AddToTempTMatrix()	161
7.49.3.4 CalculateCrossSection()	162
7.49.3.5 CalculateTMatrix()	162
7.49.3.6 ClearMatrices()	162
7.49.3.7 ClearTempTMatrices()	162
7.49.3.8 compound()	162
7.49.3.9 configure()	162
7.49.3.10 FillMatrices()	163

7.49.3.11 GetECTMatrixElement()	163
7.49.3.12 GetTempTMatrix()	163
7.49.3.13 GetTMatrixElement()	163
7.49.3.14 InvertMatrices()	163
7.49.3.15 IsTempTMatrix()	164
7.49.3.16 NewTempTMatrix()	164
7.49.3.17 NumTempTMatrices()	164
7.49.4 Member Data Documentation	164
7.49.4.1 ec_tmatrix_	164
7.49.4.2 tmatrix_	164
7.50 gsl_reactionrate_params Struct Reference	165
7.50.1 Detailed Description	165
7.50.2 Constructor & Destructor Documentation	165
7.50.2.1 gsl_reactionrate_params()	165
7.50.3 Member Data Documentation	165
7.50.3.1 compound	165
7.50.3.2 configure	165
7.50.3.3 entranceKey	165
7.50.3.4 exitKey	166
7.50.3.5 temperature	166
7.51 GSLEException Class Reference	166
7.51.1 Detailed Description	166
7.51.2 Constructor & Destructor Documentation	167
7.51.2.1 GSLEException()	167
7.51.2.2 ~GSLEException()	167
7.51.3 Member Function Documentation	167
7.51.3.1 GSLErrorHandler()	167
7.51.3.2 what()	167
7.52 InfoDialog Class Reference	167
7.52.1 Detailed Description	168
7.52.2 Constructor & Destructor Documentation	168
7.52.2.1 InfoDialog()	168
7.53 IntegratedFermiFunc Class Reference	168
7.53.1 Detailed Description	168
7.53.2 Constructor & Destructor Documentation	169
7.53.2.1 IntegratedFermiFunc()	169
7.53.3 Member Function Documentation	169
7.53.3.1 operator>()	169
7.54 Interference Class Reference	169
7.54.1 Detailed Description	170
7.54.2 Constructor & Destructor Documentation	170
7.54.2.1 Interference()	170

7.54.3 Member Function Documentation	170
7.54.3.1 GetInterferenceType()	170
7.54.3.2 GetM1()	170
7.54.3.3 GetM2()	170
7.54.3.4 GetZ1Z2()	171
7.55 JGroup Class Reference	171
7.55.1 Detailed Description	171
7.55.2 Constructor & Destructor Documentation	172
7.55.2.1 JGroup() [1/2]	172
7.55.2.2 JGroup() [2/2]	172
7.55.3 Member Function Documentation	172
7.55.3.1 AddChannel()	172
7.55.3.2 AddLevel()	172
7.55.3.3 GetChannel()	172
7.55.3.4 GetJ()	173
7.55.3.5 GetLevel()	173
7.55.3.6 GetPi()	173
7.55.3.7 IsChannel()	173
7.55.3.8 IsInRMatrix()	173
7.55.3.9 IsLevel()	174
7.55.3.10 NumChannels()	174
7.55.3.11 NumLevels()	174
7.56 KGroup Class Reference	174
7.56.1 Detailed Description	175
7.56.2 Constructor & Destructor Documentation	175
7.56.2.1 KGroup()	175
7.56.3 Member Function Documentation	175
7.56.3.1 AddECMGroup()	175
7.56.3.2 AddMGroup()	175
7.56.3.3 GetECMGroup()	175
7.56.3.4 GetMGroup()	176
7.56.3.5 GetS()	176
7.56.3.6 GetSp()	176
7.56.3.7 IsMGroup()	176
7.56.3.8 NumECMGroups()	176
7.56.3.9 NumMGroups()	176
7.57 KLGroup Class Reference	177
7.57.1 Detailed Description	177
7.57.2 Constructor & Destructor Documentation	177
7.57.2.1 KLGroup()	177
7.57.3 Member Function Documentation	177
7.57.3.1 AddInterference()	177

7.57.3.2 GetInterference()	178
7.57.3.3 GetK()	178
7.57.3.4 GetLOrder()	178
7.57.3.5 IsInterference()	178
7.57.3.6 NumInterferences()	178
7.58 LevelsData Struct Reference	179
7.58.1 Detailed Description	179
7.58.2 Member Data Documentation	179
7.58.2.1 energy	179
7.58.2.2 isActive	179
7.58.2.3 isFixed	179
7.58.2.4 jValue	179
7.58.2.5 piValue	180
7.58.2.6 SIZE	180
7.59 LevelsHeaderView Class Reference	180
7.59.1 Detailed Description	180
7.59.2 Constructor & Destructor Documentation	181
7.59.2.1 LevelsHeaderView()	181
7.59.3 Member Function Documentation	181
7.59.3.1 mouseMoveEvent()	181
7.59.3.2 mousePressEvent()	181
7.59.3.3 mouseReleaseEvent()	181
7.60 LevelsModel Class Reference	181
7.60.1 Detailed Description	182
7.60.2 Constructor & Destructor Documentation	182
7.60.2.1 LevelsModel()	182
7.60.3 Member Function Documentation	182
7.60.3.1 columnCount()	182
7.60.3.2 data()	182
7.60.3.3 flags()	183
7.60.3.4 getLevels()	183
7.60.3.5 getSpinLabel()	183
7.60.3.6 headerData()	183
7.60.3.7 insertRows()	183
7.60.3.8 isLevel()	183
7.60.3.9 removeRows()	184
7.60.3.10 rowCount()	184
7.60.3.11 setData()	184
7.61 LevelsTab Class Reference	184
7.61.1 Detailed Description	185
7.61.2 Constructor & Destructor Documentation	185
7.61.2.1 LevelsTab()	185

7.61.3 Member Function Documentation	185
7.61.3.1 addLevel [1/2]	185
7.61.3.2 addLevel [2/2]	185
7.61.3.3 calculateChannels()	186
7.61.3.4 editLevel	186
7.61.3.5 readExistingPair	186
7.61.3.6 readNewPair	186
7.61.3.7 readNuclearFile()	186
7.61.3.8 removeLevel	186
7.61.3.9 reset()	186
7.61.3.10 setPairsModel()	187
7.61.3.11 showInfo	187
7.61.3.12 updateButtons	187
7.61.3.13 updateChannelsLevelAdded()	187
7.61.3.14 updateChannelsLevelDeleted()	187
7.61.3.15 updateChannelsLevelEdited()	187
7.61.3.16 updateChannelsPairAddedEdited	187
7.61.3.17 updateChannelsPairRemoved	188
7.61.3.18 updateDetails	188
7.61.3.19 updateFilter	188
7.61.3.20 updateReducedWidth	188
7.61.3.21 writeNuclearFile()	188
7.62 MatrixInv Class Reference	188
7.62.1 Detailed Description	189
7.62.2 Constructor & Destructor Documentation	189
7.62.2.1 MatrixInv()	189
7.62.3 Member Function Documentation	189
7.62.3.1 inverse()	189
7.63 MGroup Class Reference	189
7.63.1 Detailed Description	190
7.63.2 Constructor & Destructor Documentation	190
7.63.2.1 MGroup()	190
7.63.3 Member Function Documentation	190
7.63.3.1 GetChNum()	190
7.63.3.2 GetChpNum()	190
7.63.3.3 GetJNum()	191
7.63.3.4 GetStatSpinFactor()	191
7.63.3.5 SetStatSpinFactor()	191
7.64 NFIIntegral Class Reference	191
7.64.1 Detailed Description	192
7.64.2 Constructor & Destructor Documentation	192
7.64.2.1 NFIIntegral()	192

7.64.2.2 <code>~NFIintegral()</code>	192
7.64.3 Member Function Documentation	192
7.64.3.1 <code>chanRad()</code>	192
7.64.3.2 <code>operator>()</code>	192
7.64.3.3 <code>totalSepE()</code>	193
7.65 NucLine Class Reference	193
7.65.1 Detailed Description	194
7.65.2 Constructor & Destructor Documentation	194
7.65.2.1 <code>NucLine()</code>	194
7.65.3 Member Function Documentation	194
7.65.3.1 <code>aa()</code>	194
7.65.3.2 <code>channelFix()</code>	194
7.65.3.3 <code>chRad()</code>	194
7.65.3.4 <code>e2()</code>	195
7.65.3.5 <code>e3()</code>	195
7.65.3.6 <code>ecMultMask()</code>	195
7.65.3.7 <code>entranceSepE()</code>	195
7.65.3.8 <code>g1()</code>	195
7.65.3.9 <code>g2()</code>	195
7.65.3.10 <code>gamma()</code>	196
7.65.3.11 <code>ir()</code>	196
7.65.3.12 <code>isActive()</code>	196
7.65.3.13 <code>j1()</code>	196
7.65.3.14 <code>j2()</code>	196
7.65.3.15 <code>j3()</code>	196
7.65.3.16 <code>l()</code>	197
7.65.3.17 <code>levelE()</code>	197
7.65.3.18 <code>levelFix()</code>	197
7.65.3.19 <code>levelID()</code>	197
7.65.3.20 <code>levelJ()</code>	197
7.65.3.21 <code>levelPi()</code>	197
7.65.3.22 <code>m1()</code>	198
7.65.3.23 <code>m2()</code>	198
7.65.3.24 <code>pi1()</code>	198
7.65.3.25 <code>pi2()</code>	198
7.65.3.26 <code>pi3()</code>	198
7.65.3.27 <code>pType()</code>	198
7.65.3.28 <code>s()</code>	199
7.65.3.29 <code>sepE()</code>	199
7.65.3.30 <code>z1()</code>	199
7.65.3.31 <code>z2()</code>	199
7.66 ODE_integration Class Reference	199

7.66.1 Detailed Description	200
7.66.2 Constructor & Destructor Documentation	200
7.66.2.1 ODE_integration()	200
7.66.3 Member Function Documentation	200
7.66.3.1 operator()()	200
7.67 PairsData Struct Reference	200
7.67.1 Detailed Description	201
7.67.2 Member Data Documentation	201
7.67.2.1 channelRadius	201
7.67.2.2 ecMultMask	201
7.67.2.3 excitationEnergy	201
7.67.2.4 heavyG	201
7.67.2.5 heavyJ	201
7.67.2.6 heavyM	201
7.67.2.7 heavyPi	202
7.67.2.8 heavyZ	202
7.67.2.9 lightG	202
7.67.2.10 lightJ	202
7.67.2.11 lightM	202
7.67.2.12 lightPi	202
7.67.2.13 lightZ	202
7.67.2.14 pairType	202
7.67.2.15 seperationEnergy	203
7.67.2.16 SIZE	203
7.68 PairsModel Class Reference	203
7.68.1 Detailed Description	203
7.68.2 Constructor & Destructor Documentation	204
7.68.2.1 PairsModel()	204
7.68.3 Member Function Documentation	204
7.68.3.1 columnCount()	204
7.68.3.2 data()	204
7.68.3.3 getPairs()	204
7.68.3.4 getParticleLabel()	204
7.68.3.5 getReactionLabel()	204
7.68.3.6 getReactionLabelTotalCapture()	205
7.68.3.7 getSpinLabel()	205
7.68.3.8 headerData()	205
7.68.3.9 insertRows()	205
7.68.3.10 isPair()	205
7.68.3.11 numPairs()	205
7.68.3.12 removeRows()	206
7.68.3.13 rowCount()	206

7.68.3.14 setData()	206
7.69 PairsTab Class Reference	206
7.69.1 Detailed Description	207
7.69.2 Constructor & Destructor Documentation	207
7.69.2.1 PairsTab()	207
7.69.3 Member Function Documentation	207
7.69.3.1 addPair [1/2]	207
7.69.3.2 addPair [2/2]	207
7.69.3.3 editPair [1/2]	207
7.69.3.4 editPair [2/2]	208
7.69.3.5 getPairsModel()	208
7.69.3.6 pairAdded	208
7.69.3.7 pairEdited	208
7.69.3.8 pairRemoved	208
7.69.3.9 parseOldECSection()	208
7.69.3.10 removePair	208
7.69.3.11 showInfo	209
7.69.3.12 updateButtons	209
7.70 PlotEntry Class Reference	209
7.70.1 Detailed Description	209
7.70.2 Constructor & Destructor Documentation	210
7.70.2.1 PlotEntry()	210
7.70.2.2 ~PlotEntry()	210
7.70.3 Member Function Documentation	210
7.70.3.1 attach()	210
7.70.3.2 detach()	210
7.70.3.3 readData()	210
7.70.3.4 type()	210
7.70.4 Friends And Related Symbol Documentation	211
7.70.4.1 AZUREPlot	211
7.71 PlotPoint Struct Reference	211
7.71.1 Detailed Description	211
7.71.2 Member Data Documentation	211
7.71.2.1 angle	211
7.71.2.2 dataCrossSection	211
7.71.2.3 dataErrorCrossSection	212
7.71.2.4 dataErrorSFactor	212
7.71.2.5 dataSFactor	212
7.71.2.6 energy	212
7.71.2.7 excitationEnergy	212
7.71.2.8 fitCrossSection	212
7.71.2.9 fitSFactor	212

7.72 PlotTab Class Reference	213
7.72.1 Detailed Description	213
7.72.2 Constructor & Destructor Documentation	213
7.72.2.1 PlotTab()	213
7.72.3 Member Function Documentation	214
7.72.3.1 draw	214
7.72.3.2 getDataSegments()	214
7.72.3.3 getTestSegments()	214
7.72.3.4 reset()	214
7.72.3.5 showInfo	214
7.72.3.6 xAxisLogScaleChanged	214
7.72.3.7 xAxisTypeChanged	214
7.72.3.8 yAxisLogScaleChanged	215
7.72.3.9 yAxisTypeChanged	215
7.72.4 Friends And Related Symbol Documentation	215
7.72.4.1 AZUREPlot	215
7.73 PPair Class Reference	215
7.73.1 Detailed Description	216
7.73.2 Constructor & Destructor Documentation	216
7.73.2.1 PPair()	216
7.73.3 Member Function Documentation	216
7.73.3.1 AddDecay()	216
7.73.3.2 GetChRad()	216
7.73.3.3 GetDecay()	216
7.73.3.4 GetExE()	217
7.73.3.5 GetG()	217
7.73.3.6 GetI1I2Factor()	217
7.73.3.7 GetJ()	217
7.73.3.8 GetM()	217
7.73.3.9 GetPairKey()	217
7.73.3.10 GetPi()	218
7.73.3.11 GetPType()	218
7.73.3.12 GetRedMass()	218
7.73.3.13 GetSepE()	218
7.73.3.14 GetZ()	218
7.73.3.15 IsDecay() [1/2]	218
7.73.3.16 IsDecay() [2/2]	219
7.73.3.17 IsEntrance()	219
7.73.3.18 NumDecays()	219
7.73.3.19 SetEntrance()	219
7.74 RateData Class Reference	219
7.74.1 Detailed Description	220

7.74.2 Constructor & Destructor Documentation	220
7.74.2.1 RateData()	220
7.74.3 Member Function Documentation	220
7.74.3.1 operator<()	220
7.74.4 Member Data Documentation	221
7.74.4.1 rate	221
7.74.4.2 temperature	221
7.75 RateParams Struct Reference	221
7.75.1 Detailed Description	222
7.75.2 Member Data Documentation	222
7.75.2.1 entrancePair	222
7.75.2.2 exitPair	222
7.75.2.3 maxTemp	222
7.75.2.4 minTemp	222
7.75.2.5 temperatureFile	222
7.75.2.6 tempStep	223
7.75.2.7 useFile	223
7.76 ReactionRate Class Reference	223
7.76.1 Detailed Description	223
7.76.2 Constructor & Destructor Documentation	224
7.76.2.1 ReactionRate()	224
7.76.3 Member Function Documentation	224
7.76.3.1 CalculateFileRates()	224
7.76.3.2 CalculateRates()	224
7.76.3.3 compound()	224
7.76.3.4 configure()	224
7.76.3.5 entranceKey()	225
7.76.3.6 exitKey()	225
7.76.3.7 WriteRates()	225
7.77 RichTextDelegate Class Reference	225
7.77.1 Detailed Description	226
7.77.2 Member Function Documentation	226
7.77.2.1 paint()	226
7.77.2.2 sizeHint()	226
7.78 RMatrixFunc Class Reference	226
7.78.1 Detailed Description	228
7.78.2 Constructor & Destructor Documentation	228
7.78.2.1 RMatrixFunc()	228
7.78.3 Member Function Documentation	228
7.78.3.1 AddRLInvMatrix()	228
7.78.3.2 AddRLInvRMatrixElement()	228
7.78.3.3 AddRLMatrixElement()	228

7.78.3.4 AddRMatrixElement()	229
7.78.3.5 CalculateCrossSection()	229
7.78.3.6 CalculateTMatrix()	229
7.78.3.7 ClearMatrices()	229
7.78.3.8 compound()	229
7.78.3.9 configure()	230
7.78.3.10 FillMatrices()	230
7.78.3.11 GetJSpecRLMatrix()	230
7.78.3.12 GetRLInvMatrixElement()	230
7.78.3.13 GetRLInvRMatrixElement()	230
7.78.3.14 GetRLMatrixElement()	231
7.78.3.15 GetRMatrixElement()	231
7.78.3.16 InvertMatrices()	231
7.79 RunTab Class Reference	231
7.79.1 Detailed Description	232
7.79.2 Constructor & Destructor Documentation	232
7.79.2.1 RunTab()	232
7.79.3 Member Function Documentation	232
7.79.3.1 reset()	232
7.79.3.2 showInfo	232
7.79.4 Friends And Related Symbol Documentation	233
7.79.4.1 AZUREMainThread	233
7.79.4.2 AZURESetup	233
7.80 SegDataProxyModel Class Reference	233
7.80.1 Detailed Description	233
7.80.2 Constructor & Destructor Documentation	233
7.80.2.1 SegDataProxyModel()	233
7.80.3 Member Function Documentation	234
7.80.3.1 data()	234
7.81 SegLine Class Reference	234
7.81.1 Detailed Description	234
7.81.2 Constructor & Destructor Documentation	235
7.81.2.1 SegLine()	235
7.81.3 Member Function Documentation	235
7.81.3.1 dataFile()	235
7.81.3.2 dataNorm()	235
7.81.3.3 dataNormError()	235
7.81.3.4 entranceKey()	235
7.81.3.5 exitKey()	236
7.81.3.6 isActive()	236
7.81.3.7 isDiff()	236
7.81.3.8 maxA()	236

7.81.3.9 maxE()	236
7.81.3.10 minA()	236
7.81.3.11 minE()	237
7.81.3.12 phaseJ()	237
7.81.3.13 phaseL()	237
7.81.3.14 varyNorm()	237
7.82 SegmentsDataData Struct Reference	237
7.82.1 Detailed Description	238
7.82.2 Member Data Documentation	238
7.82.2.1 dataFile	238
7.82.2.2 dataNorm	238
7.82.2.3 dataNormError	238
7.82.2.4 dataType	239
7.82.2.5 entrancePairIndex	239
7.82.2.6 exitPairIndex	239
7.82.2.7 highAngle	239
7.82.2.8 highEnergy	239
7.82.2.9 isActive	239
7.82.2.10 lowAngle	239
7.82.2.11 lowEnergy	239
7.82.2.12 phaseJ	240
7.82.2.13 phaseL	240
7.82.2.14 SIZE	240
7.82.2.15 varyNorm	240
7.83 SegmentsDataModel Class Reference	240
7.83.1 Detailed Description	241
7.83.2 Constructor & Destructor Documentation	241
7.83.2.1 SegmentsDataModel()	241
7.83.3 Member Function Documentation	241
7.83.3.1 columnCount()	241
7.83.3.2 data()	241
7.83.3.3 flags()	242
7.83.3.4 getLines()	242
7.83.3.5 getReactionLabel()	242
7.83.3.6 headerData()	242
7.83.3.7 insertRows()	242
7.83.3.8 isSegDataLine()	242
7.83.3.9 removeRows()	243
7.83.3.10 rowCount()	243
7.83.3.11 setData()	243
7.83.3.12 setPairsModel()	243
7.84 SegmentsTab Class Reference	243

7.84.1 Detailed Description	244
7.84.2 Constructor & Destructor Documentation	244
7.84.2.1 SegmentsTab()	244
7.84.3 Member Function Documentation	244
7.84.3.1 addSegDataLine [1/2]	244
7.84.3.2 addSegDataLine [2/2]	245
7.84.3.3 addSegTestLine [1/2]	245
7.84.3.4 addSegTestLine [2/2]	245
7.84.3.5 deleteSegDataLine	245
7.84.3.6 deleteSegTestLine	245
7.84.3.7 editSegDataLine	245
7.84.3.8 editSegTestLine	245
7.84.3.9 getSegmentsDataModel()	246
7.84.3.10 getSegmentsTestModel()	246
7.84.3.11 moveSegDataLineDown	246
7.84.3.12 moveSegDataLineUp	246
7.84.3.13 moveSegTestLineDown	246
7.84.3.14 moveSegTestLineUp	246
7.84.3.15 readSegDataFile	246
7.84.3.16 readSegTestFile	247
7.84.3.17 reset()	247
7.84.3.18 setPairsModel	247
7.84.3.19 showInfo	247
7.84.3.20 updateSegDataButtons	247
7.84.3.21 updateSegTestButtons	247
7.84.3.22 writeSegDataFile	247
7.84.3.23 writeSegTestFile	248
7.85 SegmentsTestData Struct Reference	248
7.85.1 Detailed Description	248
7.85.2 Member Data Documentation	248
7.85.2.1 angleStep	248
7.85.2.2 dataType	249
7.85.2.3 energyStep	249
7.85.2.4 entrancePairIndex	249
7.85.2.5 exitPairIndex	249
7.85.2.6 highAngle	249
7.85.2.7 highEnergy	249
7.85.2.8 isActive	249
7.85.2.9 lowAngle	249
7.85.2.10 lowEnergy	250
7.85.2.11 maxAngDistOrder	250
7.85.2.12 phaseJ	250

7.85.2.13 phaseL	250
7.85.2.14 SIZE	250
7.86 SegmentsTestModel Class Reference	250
7.86.1 Detailed Description	251
7.86.2 Constructor & Destructor Documentation	251
7.86.2.1 SegmentsTestModel()	251
7.86.3 Member Function Documentation	251
7.86.3.1 columnCount()	251
7.86.3.2 data()	251
7.86.3.3 flags()	252
7.86.3.4 getLines()	252
7.86.3.5 getReactionLabel()	252
7.86.3.6 headerData()	252
7.86.3.7 insertRows()	252
7.86.3.8 isSegTestLine()	252
7.86.3.9 removeRows()	253
7.86.3.10 rowCount()	253
7.86.3.11 setData()	253
7.86.3.12 setPairsModel()	253
7.87 SegPairs Struct Reference	253
7.87.1 Detailed Description	253
7.87.2 Member Data Documentation	254
7.87.2.1 firstPair	254
7.87.2.2 secondPair	254
7.88 SegTestProxyModel Class Reference	254
7.88.1 Detailed Description	254
7.88.2 Constructor & Destructor Documentation	254
7.88.2.1 SegTestProxyModel()	254
7.88.3 Member Function Documentation	255
7.88.3.1 data()	255
7.88.3.2 filterAcceptsRow()	255
7.89 ShftFunc Class Reference	255
7.89.1 Detailed Description	255
7.89.2 Constructor & Destructor Documentation	256
7.89.2.1 ShftFunc()	256
7.89.2.2 ~ShftFunc()	256
7.89.3 Member Function Documentation	256
7.89.3.1 EnergyDerivative()	256
7.89.3.2 operator>()	256
7.90 SyntaxError Class Reference	257
7.90.1 Detailed Description	257
7.90.2 Constructor & Destructor Documentation	257

7.90.2.1 SyntaxError()	257
7.90.2.2 ~SyntaxError()	257
7.90.3 Member Function Documentation	258
7.90.3.1 what()	258
7.91 TargetEffect Class Reference	258
7.91.1 Detailed Description	258
7.91.2 Constructor & Destructor Documentation	259
7.91.2.1 TargetEffect()	259
7.91.3 Member Function Documentation	259
7.91.3.1 GetConvolutionFactor()	259
7.91.3.2 GetDensity()	259
7.91.3.3 GetQCoefficient()	259
7.91.3.4 GetSegmentsList()	259
7.91.3.5 GetSigma()	260
7.91.3.6 GetStoppingPowerEq()	260
7.91.3.7 IsActive()	260
7.91.3.8 IsConvolution()	260
7.91.3.9 IsQCoefficients()	260
7.91.3.10 IsTargetIntegration()	260
7.91.3.11 NumQCoefficients()	261
7.91.3.12 NumSubPoints()	261
7.91.3.13 SetNumSubPoints()	261
7.91.3.14 SetSigma()	261
7.91.3.15 TargetThickness()	261
7.91.4 Member Data Documentation	262
7.91.4.1 convolutionRange	262
7.92 TargetIntData Struct Reference	262
7.92.1 Detailed Description	262
7.92.2 Member Data Documentation	262
7.92.2.1 density	262
7.92.2.2 isActive	263
7.92.2.3 isConvolution	263
7.92.2.4 isQCoefficients	263
7.92.2.5 isTargetIntegration	263
7.92.2.6 numParameters	263
7.92.2.7 numPoints	263
7.92.2.8 parameters	263
7.92.2.9 qCoefficients	263
7.92.2.10 segmentsList	264
7.92.2.11 sigma	264
7.92.2.12 SIZE	264
7.92.2.13 stoppingPowerEq	264

7.93 TargetIntModel Class Reference	264
7.93.1 Detailed Description	265
7.93.2 Constructor & Destructor Documentation	265
7.93.2.1 TargetIntModel()	265
7.93.3 Member Function Documentation	265
7.93.3.1 columnCount()	265
7.93.3.2 data()	265
7.93.3.3 flags()	265
7.93.3.4 getLines()	265
7.93.3.5 headerData()	266
7.93.3.6 insertRows()	266
7.93.3.7 removeRows()	266
7.93.3.8 rowCount()	266
7.93.3.9 setData()	266
7.94 TargetIntTab Class Reference	267
7.94.1 Detailed Description	267
7.94.2 Constructor & Destructor Documentation	267
7.94.2.1 TargetIntTab()	267
7.94.3 Member Function Documentation	267
7.94.3.1 addLine [1/2]	267
7.94.3.2 addLine [2/2]	268
7.94.3.3 deleteLine	268
7.94.3.4 editLine	268
7.94.3.5 getTargetIntModel()	268
7.94.3.6 readFile()	268
7.94.3.7 reset()	268
7.94.3.8 showInfo	268
7.94.3.9 updateButtons	269
7.94.3.10 writeFile()	269
7.95 TempTMatrix Struct Reference	269
7.95.1 Detailed Description	269
7.95.2 Member Data Documentation	270
7.95.2.1 jValue	270
7.95.2.2 lpValue	270
7.95.2.3 lValue	270
7.95.2.4 TMatrix	270
7.96 TextEditBuffer Class Reference	270
7.96.1 Detailed Description	271
7.96.2 Constructor & Destructor Documentation	271
7.96.2.1 TextEditBuffer()	271
7.96.3 Member Function Documentation	271
7.96.3.1 overflow()	271

7.96.3.2 sync()	271
7.96.3.3 updateLog	272
7.97 WhitFunc Class Reference	272
7.97.1 Detailed Description	272
7.97.2 Constructor & Destructor Documentation	272
7.97.2.1 WhitFunc()	272
7.97.3 Member Function Documentation	273
7.97.3.1 operator>()	273
7.97.3.2 redmass()	273
7.97.3.3 z1()	273
7.97.3.4 z2()	273
8 File Documentation	275
8.1 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/complex_functions.H File Reference	275
8.1.1 Macro Definition Documentation	276
8.1.1.1 SIGN	276
8.1.2 Function Documentation	276
8.1.2.1 exp_l_omega_chi_calc()	276
8.1.2.2 expm1()	277
8.1.2.3 inf_norm()	277
8.1.2.4 isfinite()	277
8.1.2.5 log1p()	277
8.1.2.6 log_Cl_eta_calc()	277
8.1.2.7 log_cut_constant_AS_calc()	277
8.1.2.8 log_cut_constant_CFa_calc()	278
8.1.2.9 log_cut_constant_CFb_calc()	278
8.1.2.10 log_Gamma()	278
8.1.2.11 operator"!=([1/4]	278
8.1.2.12 operator"!=([2/4]	278
8.1.2.13 operator"!=([3/4]	278
8.1.2.14 operator"!=([4/4]	279
8.1.2.15 operator*() [1/4]	279
8.1.2.16 operator*() [2/4]	279
8.1.2.17 operator*() [3/4]	279
8.1.2.18 operator*() [4/4]	279
8.1.2.19 operator+() [1/4]	279
8.1.2.20 operator+() [2/4]	280
8.1.2.21 operator+() [3/4]	280
8.1.2.22 operator+() [4/4]	280
8.1.2.23 operator-() [1/4]	280
8.1.2.24 operator-() [2/4]	280
8.1.2.25 operator-() [3/4]	280

8.1.2.26 operator-() [4 / 4]	281
8.1.2.27 operator/() [1 / 4]	281
8.1.2.28 operator/() [2 / 4]	281
8.1.2.29 operator/() [3 / 4]	281
8.1.2.30 operator/() [4 / 4]	281
8.1.2.31 operator==() [1 / 4]	281
8.1.2.32 operator==() [2 / 4]	282
8.1.2.33 operator==() [3 / 4]	282
8.1.2.34 operator==() [4 / 4]	282
8.1.2.35 sigma_I_calc()	282
8.1.2.36 sin_chi_calc()	282
8.1.3 Variable Documentation	282
8.1.3.1 precision	282
8.1.3.2 sqrt_precision	283
8.2 complex_functions.H	283
8.3 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/cwfcomp.H File Reference	285
8.4 cwfcomp.H	285
8.5 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/ode_int.H File Reference	288
8.6 ode_int.H	288
8.7 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/complex_functions.cpp File Reference	289
8.7.1 Function Documentation	289
8.7.1.1 exp_I_omega_chi_calc()	289
8.7.1.2 expm1()	289
8.7.1.3 log1p()	290
8.7.1.4 log_Cl_eta_calc()	290
8.7.1.5 log_cut_constant_AS_calc()	290
8.7.1.6 log_cut_constant_CFa_calc()	290
8.7.1.7 log_cut_constant_CFb_calc()	290
8.7.1.8 log_Gamma()	290
8.7.1.9 sigma_I_calc()	291
8.7.1.10 sin_chi_calc()	291
8.8 complex_functions.cpp	291
8.9 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/cwfcomp.cpp File Reference	296
8.10 cwfcomp.cpp	296
8.11 /Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/ode_int.cpp File Reference	315
8.12 ode_int.cpp	315
8.13 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AboutAZURE2Dialog.h File Reference	318
8.14 AboutAZURE2Dialog.h	318
8.15 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddLevelDialog.h File Reference	318
8.16 AddLevelDialog.h	319
8.17 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddPairDialog.h File Reference	319
8.18 AddPairDialog.h	319

8.19 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddSegDataDialog.h File Reference	320
8.20 AddSegDataDialog.h	320
8.21 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddSegTestDialog.h File Reference	321
8.22 AddSegTestDialog.h	321
8.23 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddTargetIntDialog.h File Reference	322
8.24 AddTargetIntDialog.h	322
8.25 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREMainThread.h File Reference	323
8.26 AZUREMainThread.h	323
8.27 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREPlot.h File Reference	324
8.28 AZUREPlot.h	325
8.29 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZURESetup.h File Reference	326
8.30 AZURESetup.h	326
8.31 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelDetails.h File Reference	328
8.32 ChannelDetails.h	328
8.33 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelsModel.h File Reference	328
8.34 ChannelsModel.h	329
8.35 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChooseFileButton.h File Reference	329
8.36 ChooseFileButton.h	329
8.37 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditChecksDialog.h File Reference	330
8.38 EditChecksDialog.h	330
8.39 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditDirsDialog.h File Reference	331
8.40 EditDirsDialog.h	331
8.41 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditOptionsDialog.h File Reference	331
8.42 EditOptionsDialog.h	332
8.43 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ElementMap.h File Reference	332
8.44 ElementMap.h	332
8.45 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/FilteredTextEdit.h File Reference	334
8.46 FilteredTextEdit.h	334
8.47 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/InfoDialog.h File Reference	335
8.48 InfoDialog.h	335
8.49 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsHeaderView.h File Reference	335
8.50 LevelsHeaderView.h	336
8.51 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsModel.h File Reference	336
8.52 LevelsModel.h	336
8.53 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsTab.h File Reference	337
8.54 LevelsTab.h	337
8.55 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsModel.h File Reference	338
8.56 PairsModel.h	339
8.57 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsTab.h File Reference	339
8.58 PairsTab.h	340
8.59 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PlotTab.h File Reference	340
8.60 PlotTab.h	341

8.61 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RichTextDelegate.h File Reference	342
8.62 RichTextDelegate.h	342
8.63 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RunTab.h File Reference	342
8.64 RunTab.h	342
8.65 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsDataModel.h File Reference	343
8.66 SegmentsDataModel.h	344
8.67 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTab.h File Reference	344
8.68 SegmentsTab.h	345
8.69 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTestModel.h File Reference	346
8.70 SegmentsTestModel.h	346
8.71 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntModel.h File Reference	347
8.71.1 Function Documentation	347
8.71.1.1 Q_DECLARE_METATYPE()	347
8.72 TargetIntModel.h	347
8.73 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntTab.h File Reference	348
8.74 TargetIntTab.h	348
8.75 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TextEditBuffer.h File Reference	349
8.76 TextEditBuffer.h	349
8.77 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AboutAZURE2Dialog.cpp File Reference	350
8.78 AboutAZURE2Dialog.cpp	350
8.79 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddLevelDialog.cpp File Reference	351
8.80 AddLevelDialog.cpp	351
8.81 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddPairDialog.cpp File Reference	351
8.82 AddPairDialog.cpp	352
8.83 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddSegDataDialog.cpp File Reference	354
8.84 AddSegDataDialog.cpp	354
8.85 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddSegTestDialog.cpp File Reference	357
8.86 AddSegTestDialog.cpp	357
8.87 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddTargetIntDialog.cpp File Reference	359
8.88 AddTargetIntDialog.cpp	359
8.89 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZUREPlot.cpp File Reference	362
8.90 AZUREPlot.cpp	363
8.91 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZURESetup.cpp File Reference	367
8.91.1 Function Documentation	368
8.91.1.1 checkExternalCapture()	368
8.91.1.2 exitMessage()	368
8.91.1.3 readSegmentFile()	369
8.91.1.4 startMessage()	369
8.92 AZURESetup.cpp	369
8.93 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChannelDetails.cpp File Reference	381
8.94 ChannelDetails.cpp	381
8.95 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChannelsModel.cpp File Reference	382

8.96 ChannelsModel.cpp	382
8.97 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChooseFileButton.cpp File Reference	385
8.98 ChooseFileButton.cpp	385
8.99 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditChecksDialog.cpp File Reference	385
8.100 EditChecksDialog.cpp	385
8.101 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditDirsDialog.cpp File Reference	386
8.102 EditDirsDialog.cpp	387
8.103 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditOptionsDialog.cpp File Reference	387
8.104 EditOptionsDialog.cpp	387
8.105 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InfoDialog.cpp File Reference	388
8.106 InfoDialog.cpp	388
8.107 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InTabDocs.cpp File Reference	389
8.107.1 Function Documentation	389
8.107.1.1 setInfoStrings()	389
8.108 InTabDocs.cpp	390
8.109 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/LevelsModel.cpp File Reference	392
8.110 LevelsModel.cpp	392
8.111 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/LevelsTab.cpp File Reference	394
8.112 LevelsTab.cpp	394
8.113 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/main.cpp File Reference	404
8.113.1 Function Documentation	404
8.113.1.1 initResource()	404
8.113.1.2 start_gui()	404
8.114 main.cpp	405
8.115 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PairsModel.cpp File Reference	405
8.116 PairsModel.cpp	405
8.117 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PairsTab.cpp File Reference	410
8.118 PairsTab.cpp	410
8.119 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PlotTab.cpp File Reference	415
8.120 PlotTab.cpp	415
8.121 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RichTextDelegate.cpp File Reference	418
8.122 RichTextDelegate.cpp	418
8.123 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RunTab.cpp File Reference	419
8.124 RunTab.cpp	419
8.125 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsDataModel.cpp File Reference	422
8.126 SegmentsDataModel.cpp	423
8.127 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTab.cpp File Reference	426
8.128 SegmentsTab.cpp	426
8.129 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTestModel.cpp File Reference	437
8.130 SegmentsTestModel.cpp	437
8.131 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntModel.cpp File Reference	440
8.132 TargetIntModel.cpp	440

8.133 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntTab.cpp File Reference	442
8.134 TargetIntTab.cpp	442
8.135 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AChannel.h File Reference	447
8.136 AChannel.h	447
8.137 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ALevel.h File Reference	448
8.138 ALevel.h	448
8.139 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AMatrixFunc.h File Reference	449
8.140 AMatrixFunc.h	449
8.141 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AngCoeff.h File Reference	449
8.142 AngCoeff.h	450
8.143 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZURECalc.h File Reference	450
8.144 AZURECalc.h	450
8.145 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREFBuffer.h File Reference	451
8.146 AZUREFBuffer.h	451
8.147 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREMain.h File Reference	452
8.148 AZUREMain.h	452
8.149 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREOutput.h File Reference	452
8.150 AZUREOutput.h	453
8.151 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREParams.h File Reference	453
8.152 AZUREParams.h	453
8.153 /Users/kuba/Desktop/R-Matrix/AZURE2/include/CNuc.h File Reference	454
8.153.1 Function Documentation	454
8.153.1.1 DoubleFactorial()	454
8.154 CNuc.h	454
8.155 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Config.h File Reference	455
8.156 Config.h	455
8.157 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Constants.h File Reference	456
8.157.1 Typedef Documentation	457
8.157.1.1 complex	457
8.157.1.2 matrix_c	457
8.157.1.3 matrix_r	457
8.157.1.4 vector_c	457
8.157.1.5 vector_matrix_c	457
8.157.1.6 vector_matrix_r	458
8.157.1.7 vector_r	458
8.157.2 Variable Documentation	458
8.157.2.1 avagadroNum	458
8.157.2.2 boltzConst	458
8.157.2.3 fstruc	458
8.157.2.4 hbarc	458
8.157.2.5 isE1	458
8.157.2.6 isE2	459

8.157.2.7 isM1	459
8.157.2.8 lightSpeedInCmPerS	459
8.157.2.9 maxECMult	459
8.157.2.10 nuclearMagnetron	459
8.157.2.11 pi	459
8.157.2.12 uconv	459
8.158 Constants.h	460
8.159 /Users/kuba/Desktop/R-Matrix/AZURE2/include/CoulFunc.h File Reference	460
8.160 CoulFunc.h	460
8.161 /Users/kuba/Desktop/R-Matrix/AZURE2/include/DataLine.h File Reference	461
8.162 DataLine.h	461
8.163 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Decay.h File Reference	462
8.164 Decay.h	462
8.165 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ECIntegral.h File Reference	462
8.166 ECIntegral.h	463
8.167 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ECMGroup.h File Reference	463
8.168 ECMGroup.h	464
8.169 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EData.h File Reference	464
8.170 EData.h	465
8.171 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EDatalterator.h File Reference	465
8.171.1 Typedef Documentation	466
8.171.1.1 EPointIterator	466
8.171.1.2 EPointMapIterator	466
8.171.1.3 ESegmentIterator	466
8.172 EDatalterator.h	466
8.173 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EffectiveCharge.h File Reference	467
8.174 EffectiveCharge.h	467
8.175 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EigenFunc.h File Reference	467
8.176 EigenFunc.h	467
8.177 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EPoint.h File Reference	468
8.178 EPoint.h	468
8.179 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Equation.h File Reference	470
8.180 Equation.h	470
8.181 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ESegment.h File Reference	471
8.182 ESegment.h	471
8.183 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ExtrapLine.h File Reference	472
8.184 ExtrapLine.h	473
8.185 /Users/kuba/Desktop/R-Matrix/AZURE2/include/GenMatrixFunc.h File Reference	473
8.186 GenMatrixFunc.h	474
8.187 /Users/kuba/Desktop/R-Matrix/AZURE2/include/GSLEException.h File Reference	474
8.188 GSLEException.h	475
8.189 /Users/kuba/Desktop/R-Matrix/AZURE2/include/IntegratedFermiFunc.h File Reference	475

8.190 IntegratedFermiFunc.h	475
8.191 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Interference.h File Reference	476
8.192 Interference.h	476
8.193 /Users/kuba/Desktop/R-Matrix/AZURE2/include/JGroup.h File Reference	476
8.194 JGroup.h	477
8.195 /Users/kuba/Desktop/R-Matrix/AZURE2/include/KGroup.h File Reference	477
8.196 KGroup.h	477
8.197 /Users/kuba/Desktop/R-Matrix/AZURE2/include/KLGroup.h File Reference	478
8.198 KLGroup.h	478
8.199 /Users/kuba/Desktop/R-Matrix/AZURE2/include/MatrixInv.h File Reference	478
8.200 MatrixInv.h	479
8.201 /Users/kuba/Desktop/R-Matrix/AZURE2/include/MGroup.h File Reference	479
8.202 MGroup.h	479
8.203 /Users/kuba/Desktop/R-Matrix/AZURE2/include/NFIntegral.h File Reference	480
8.204 NFIntegral.h	480
8.205 /Users/kuba/Desktop/R-Matrix/AZURE2/include/NucLine.h File Reference	480
8.206 NucLine.h	481
8.207 /Users/kuba/Desktop/R-Matrix/AZURE2/include/PPair.h File Reference	482
8.208 PPair.h	482
8.209 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ReactionRate.h File Reference	482
8.209.1 Function Documentation	483
8.209.1.1 gsl_reactionrate_integration()	483
8.210 ReactionRate.h	483
8.211 /Users/kuba/Desktop/R-Matrix/AZURE2/include/RMatrixFunc.h File Reference	484
8.212 RMatrixFunc.h	484
8.213 /Users/kuba/Desktop/R-Matrix/AZURE2/include/SegLine.h File Reference	484
8.214 SegLine.h	485
8.215 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ShftFunc.h File Reference	485
8.216 ShftFunc.h	486
8.217 /Users/kuba/Desktop/R-Matrix/AZURE2/include/TargetEffect.h File Reference	486
8.218 TargetEffect.h	486
8.219 /Users/kuba/Desktop/R-Matrix/AZURE2/include/WhitFunc.h File Reference	487
8.219.1 Function Documentation	487
8.219.1.1 gsl_whit_function()	487
8.220 WhitFunc.h	488
8.221 /Users/kuba/Desktop/R-Matrix/AZURE2/README.md File Reference	488
8.222 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AChannel.cpp File Reference	488
8.223 AChannel.cpp	488
8.224 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ALevel.cpp File Reference	489
8.225 ALevel.cpp	489
8.226 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AMatrixFunc.cpp File Reference	491
8.227 AMatrixFunc.cpp	492

8.228 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AngCoeff.cpp File Reference	494
8.229 AngCoeff.cpp	494
8.230 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZURE2.cpp File Reference	495
8.230.1 Function Documentation	496
8.230.1.1 checkExternalCapture()	496
8.230.1.2 commandShell()	496
8.230.1.3 exitMessage()	496
8.230.1.4 getExternalCaptureFile()	496
8.230.1.5 getParameterFile()	496
8.230.1.6 getRateParams()	497
8.230.1.7 getTemperatureFile()	497
8.230.1.8 main()	497
8.230.1.9 parseOptions()	497
8.230.1.10 printHelp()	497
8.230.1.11 processCommand()	498
8.230.1.12 readSegmentFile()	498
8.230.1.13 startMessage()	498
8.230.1.14 welcomeMessage()	498
8.231 AZURE2.cpp	499
8.232 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZURECalc.cpp File Reference	506
8.233 AZURECalc.cpp	506
8.234 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREMain.cpp File Reference	507
8.235 AZUREMain.cpp	507
8.236 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREOutput.cpp File Reference	511
8.237 AZUREOutput.cpp	511
8.238 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREParams.cpp File Reference	512
8.239 AZUREParams.cpp	512
8.240 /Users/kuba/Desktop/R-Matrix/AZURE2/src/CNuc.cpp File Reference	513
8.241 CNuc.cpp	513
8.242 /Users/kuba/Desktop/R-Matrix/AZURE2/src/Config.cpp File Reference	531
8.243 Config.cpp	531
8.244 /Users/kuba/Desktop/R-Matrix/AZURE2/src/CoulFunc.cpp File Reference	533
8.245 CoulFunc.cpp	533
8.246 /Users/kuba/Desktop/R-Matrix/AZURE2/src/Decay.cpp File Reference	535
8.247 Decay.cpp	535
8.248 /Users/kuba/Desktop/R-Matrix/AZURE2/src/DoubleFactorial.cpp File Reference	536
8.248.1 Function Documentation	536
8.248.1.1 DoubleFactorial()	536
8.249 DoubleFactorial.cpp	536
8.250 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ECIntegral.cpp File Reference	536
8.250.1 Function Documentation	536
8.250.1.1 DoubleFactorial()	536

8.251	ECIntegral.cpp	537
8.252	/Users/kuba/Desktop/R-Matrix/AZURE2/src/ECMGroup.cpp File Reference	539
8.253	ECMGroup.cpp	539
8.254	/Users/kuba/Desktop/R-Matrix/AZURE2/src/EData.cpp File Reference	540
8.255	EData.cpp	540
8.256	/Users/kuba/Desktop/R-Matrix/AZURE2/src/EDatalterator.cpp File Reference	552
8.257	EDatalterator.cpp	552
8.258	/Users/kuba/Desktop/R-Matrix/AZURE2/src/EffectiveCharge.cpp File Reference	553
8.258.1	Function Documentation	553
8.258.1.1	DoubleFactorial()	553
8.259	EffectiveCharge.cpp	554
8.260	/Users/kuba/Desktop/R-Matrix/AZURE2/src/EigenFunc.cpp File Reference	554
8.261	EigenFunc.cpp	554
8.262	/Users/kuba/Desktop/R-Matrix/AZURE2/src/EPoint.cpp File Reference	555
8.263	EPoint.cpp	556
8.264	/Users/kuba/Desktop/R-Matrix/AZURE2/src/Equation.cpp File Reference	566
8.265	Equation.cpp	566
8.266	/Users/kuba/Desktop/R-Matrix/AZURE2/src/ESegment.cpp File Reference	571
8.267	ESegment.cpp	571
8.268	/Users/kuba/Desktop/R-Matrix/AZURE2/src/GenMatrixFunc.cpp File Reference	574
8.269	GenMatrixFunc.cpp	574
8.270	/Users/kuba/Desktop/R-Matrix/AZURE2/src/GSLEException.cpp File Reference	578
8.271	GSLEException.cpp	578
8.272	/Users/kuba/Desktop/R-Matrix/AZURE2/src/IntegratedFermiFunc.cpp File Reference	578
8.273	IntegratedFermiFunc.cpp	578
8.274	/Users/kuba/Desktop/R-Matrix/AZURE2/src/Interference.cpp File Reference	579
8.275	Interference.cpp	579
8.276	/Users/kuba/Desktop/R-Matrix/AZURE2/src/JGroup.cpp File Reference	580
8.277	JGroup.cpp	580
8.278	/Users/kuba/Desktop/R-Matrix/AZURE2/src/KGroup.cpp File Reference	581
8.279	KGroup.cpp	581
8.280	/Users/kuba/Desktop/R-Matrix/AZURE2/src/KLGroup.cpp File Reference	582
8.281	KLGroup.cpp	582
8.282	/Users/kuba/Desktop/R-Matrix/AZURE2/src/MatrixInv.cpp File Reference	582
8.283	MatrixInv.cpp	583
8.284	/Users/kuba/Desktop/R-Matrix/AZURE2/src/MGroup.cpp File Reference	583
8.285	MGroup.cpp	583
8.286	/Users/kuba/Desktop/R-Matrix/AZURE2/src/NFIntegral.cpp File Reference	584
8.287	NFIntegral.cpp	584
8.288	/Users/kuba/Desktop/R-Matrix/AZURE2/src/PPair.cpp File Reference	584
8.289	PPair.cpp	584
8.290	/Users/kuba/Desktop/R-Matrix/AZURE2/src/ReactionRate.cpp File Reference	586

8.290.1 Function Documentation	586
8.290.1.1 gsl_reactionrate_integrand()	586
8.290.1.2 gsl_reactionrate_integration()	587
8.291 ReactionRate.cpp	587
8.292 /Users/kuba/Desktop/R-Matrix/AZURE2/src/RMatrixFunc.cpp File Reference	589
8.293 RMatrixFunc.cpp	589
8.294 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ShftFunc.cpp File Reference	592
8.295 ShftFunc.cpp	593
8.296 /Users/kuba/Desktop/R-Matrix/AZURE2/src/TargetEffect.cpp File Reference	593
8.297 TargetEffect.cpp	593
Index	597

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 environment 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 qtscrip5-dev

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

ROOT	17
ROOT::Minuit2	17

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

AChannel	20
ALevel	39
AngCoeff	51
AZUREFBuffer	54
AZUREMain	56
AZUREOutput	60
AZUREParams	63
ChannelsData	71
CNuc	77
Config	84
CoulFunc	89
Coulomb_wave_functions	93
CoulWaves	95
DataLine	97
Decay	98
Directories	101
ECIntegral	102
ECMGroup	103
EData	107
EDataIterator	116
EffectiveCharge	123
EigenFunc	124
EnergyMap	125
EPoint	126
Equation	143
ESegment	146
std::exception	
GSLEException	166
SyntaxError	257
ExtrapLine	154
ROOT::Minuit2::FCNBase	
AZURECalc	52
GenericFunction	158
GenMatrixFunc	160
AMatrixFunc	47

RMatrixFunc	226
gsl_reactionrate_params	165
IntegratedFermiFunc	168
Interference	169
JGroup	171
KGroup	174
KLGroup	177
LevelsData	179
MatrixInv	188
MGroup	189
NFIntegral	191
NucLine	193
ODE_integration	199
PairsData	200
PlotEntry	209
PlotPoint	211
PPair	215
QAbstractTableModel	
ChannelsModel	72
LevelsModel	181
PairsModel	203
SegmentsDataModel	240
SegmentsTestModel	250
TargetIntModel	264
QDialog	
AboutAZURE2Dialog	19
AddLevelDialog	22
AddPairDialog	23
AddSegDataDialog	27
AddSegTestDialog	31
AddTargetIntDialog	34
EditChecksDialog	118
EditDirsDialog	120
EditOptionsDialog	121
InfoDialog	167
QHeaderView	
LevelsHeaderView	180
QMainWindow	
AZURESetup	67
QObject	
AZUREMainThreadWorker	59
QPushButton	
ChooseFileButton	75
QSortFilterProxyModel	
SegDataProxyModel	233
SegTestProxyModel	254
QStyledItemDelegate	
RichTextDelegate	225
QTextEdit	
FilteredTextEdit	157
QThread	
AZUREMainThread	58
QWidget	
ChannelDetails	69
LevelsTab	184
PairsTab	206
PlotTab	213
RunTab	231

SegmentsTab	243
TargetIntTab	267
TextEditBuffer	270
QwtPlot	
AZUREPlot	65
QwtPlotZoomer	
AZUREZoomer	68
RateData	219
RateParams	221
ReactionRate	223
SegLine	234
SegmentsDataData	237
SegmentsTestData	248
SegPairs	253
ShftFunc	255
std::streambuf	
TextEditBuffer	270
TargetEffect	258
TargetIntData	262
TempTMatrix	269
WhitFunc	272

Chapter 4

Class Index

4.1 Class List

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

AboutAZURE2Dialog	19
AChannel	
An AZURE channel object	20
AddLevelDialog	22
AddPairDialog	23
AddSegDataDialog	27
AddSegTestDialog	31
AddTargetIntDialog	34
ALevel	
An AZURE level object	39
AMatrixFunc	
A function class to calculate the T-Matrix using the A-Matrix	47
AngCoeff	
A container class for angular coupling coefficient functions	51
AZURECalc	
A function class to perform the calculation of the chi-squared value	52
AZUREFBuffer	
A container class for a pointer to a file buffer	54
AZUREMain	
The top-level AZURE function class	56
AZUREMainThread	58
AZUREMainThreadWorker	59
AZUREOutput	
A class to assist in writing AZURE output files	60
AZUREParams	
A container class to hold Minuit parameters in AZURE	63
AZUREPlot	65
AZURESetup	67
AZUREZoomer	68
ChannelDetails	69
ChannelsData	71
ChannelsModel	72
ChooseFileButton	75
CNuc	
An AZURE compound nucleus	77

Config	A configuration structure for AZURE	84
CoulFunc	A function class to calculate Coulomb functions for positive energy channels	89
Coulomb_wave_functions	93
CoulWaves	The return structure of the CoulFunc function class	95
DataLine	A class to read and store a line from a data file	97
Decay	An AZURE decay pair	98
Directories	101
ECIntegral	A function class to calculate external capture integrals	102
ECMGroup	An AZURE external reaction pathway	103
EData	An AZURE data object	107
EDataIterator	An iterator class for an EData object	116
EditChecksDialog	118
EditDirsDialog	120
EditOptionsDialog	121
EffectiveCharge	A function class for calculating effective charge without long-wavelength approximation	123
EigenFunc	A function class to solve a eigenvalue problems	124
EnergyMap	A container structure for a reference to a data point	125
EPoint	An AZURE data point	126
Equation	A class for parsing algebraic expressions	143
ESegment	An AZURE data segment	146
ExtrapLine	A class to read and store a line from the extrapolation input file	154
FilteredTextEdit	157
GenericFunction	A wrapper class for function pointers used by Equation class	158
GenMatrixFunc	A generalized function class to calculate cross sections	160
gsl_reactionrate_params	165
GSLError	166
InfoDialog	167
IntegratedFermiFunc	A function class to calculate the integrated Fermi function for beta decay	168
Interference	An AZURE $l_1, l_2, l_1', l_2', J_1, J_2$ combination	169
JGroup	An AZURE J^π group	171
KGroup	An AZURE s, s' group	174
KLGroup	An AZURE s, s', L group	177
LevelsData	179
LevelsHeaderView	180
LevelsModel	181

LevelsTab	184
MatrixInv	
A Function class to perform matrix inversion	188
MGroup	
An AZURE internal reaction pathway	189
NFIntegral	
A function class to calculate the channel integrals in the denominator of the $N_f^{1/2}$ term	191
NucLine	
A class to read and store a line from a nuclear input file	193
ODE_integration	199
PairsData	200
PairsModel	203
PairsTab	206
PlotEntry	209
PlotPoint	211
PlotTab	213
PPair	
An AZURE Particle Pair	215
RateData	
A container structure for a reaction rate	219
RateParams	
A structure holding the reaction rate calculation configuration	221
ReactionRate	
A function class to calculate the reaction rate	223
RichTextDelegate	225
RMatrixFunc	
A function class to calculate the T-Matrix using the R-Matrix	226
RunTab	231
SegDataProxyModel	233
SegLine	
A class to read and store a line from the data segments input file	234
SegmentsDataData	237
SegmentsDataModel	240
SegmentsTab	243
SegmentsTestData	248
SegmentsTestModel	250
SegPairs	253
SegTestProxyModel	254
ShiftFunc	
A function class for negative energy shift functions	255
SyntaxError	
An exception class thrown by the Equation class	257
TargetEffect	
An AZURE target effect entry	258
TargetIntData	262
TargetIntModel	264
TargetIntTab	267
TempTMatrix	
A temporary T-Matrix structure	269
TextEditBuffer	270
WhitFunc	
A function class to calculate Whittaker functions for negative energy channels	272

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

/Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/complex_functions.H	275
/Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/cwfcomp.H	285
/Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/ode_int.H	288
/Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/complex_functions.cpp	289
/Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/cwfcomp.cpp	296
/Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/ode_int.cpp	315
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AboutAZURE2Dialog.h	318
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddLevelDialog.h	318
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddPairDialog.h	319
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddSegDataDialog.h	320
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddSegTestDialog.h	321
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddTargetIntDialog.h	322
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREMainThread.h	323
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZUREPlot.h	324
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZURESetup.h	326
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelDetails.h	328
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelsModel.h	328
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChooseFileButton.h	329
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditChecksDialog.h	330
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditDirsDialog.h	331
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditOptionsDialog.h	331
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ElementMap.h	332
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/FilteredTextEdit.h	334
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/InfoDialog.h	335
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsHeaderView.h	335
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsModel.h	336
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsTab.h	337
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsModel.h	338
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsTab.h	339
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PlotTab.h	340
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RichTextDelegate.h	342
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RunTab.h	342
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsDataModel.h	343
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTab.h	344
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTestModel.h	346

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntModel.h	347
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntTab.h	348
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TextEditBuffer.h	349
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AboutAZURE2Dialog.cpp	350
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddLevelDialog.cpp	351
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddPairDialog.cpp	351
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddSegDataDialog.cpp	354
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddSegTestDialog.cpp	357
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddTargetIntDialog.cpp	359
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZUREPlot.cpp	362
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZURESetup.cpp	367
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChannelDetails.cpp	381
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChannelsModel.cpp	382
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChooseFileButton.cpp	385
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditChecksDialog.cpp	385
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditDirsDialog.cpp	386
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditOptionsDialog.cpp	387
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InfoDialog.cpp	388
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InTabDocs.cpp	389
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/LevelsModel.cpp	392
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/LevelsTab.cpp	394
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/main.cpp	404
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PairsModel.cpp	405
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PairsTab.cpp	410
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PlotTab.cpp	415
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RichTextDelegate.cpp	418
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RunTab.cpp	419
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsDataModel.cpp	422
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTab.cpp	426
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTestModel.cpp	437
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntModel.cpp	440
/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntTab.cpp	442
/Users/kuba/Desktop/R-Matrix/AZURE2/include/ACHannel.h	447
/Users/kuba/Desktop/R-Matrix/AZURE2/include/ALevel.h	448
/Users/kuba/Desktop/R-Matrix/AZURE2/include/AMatrixFunc.h	449
/Users/kuba/Desktop/R-Matrix/AZURE2/include/AngCoeff.h	449
/Users/kuba/Desktop/R-Matrix/AZURE2/include/AZURECalc.h	450
/Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREFBuffer.h	451
/Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREMain.h	452
/Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREOutput.h	452
/Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREParams.h	453
/Users/kuba/Desktop/R-Matrix/AZURE2/include/CNuc.h	454
/Users/kuba/Desktop/R-Matrix/AZURE2/include/Config.h	455
/Users/kuba/Desktop/R-Matrix/AZURE2/include/Constants.h	456
/Users/kuba/Desktop/R-Matrix/AZURE2/include/CoulFunc.h	460
/Users/kuba/Desktop/R-Matrix/AZURE2/include/DataLine.h	461
/Users/kuba/Desktop/R-Matrix/AZURE2/include/Decay.h	462
/Users/kuba/Desktop/R-Matrix/AZURE2/include/ECIntegral.h	462
/Users/kuba/Desktop/R-Matrix/AZURE2/include/ECMGroup.h	463
/Users/kuba/Desktop/R-Matrix/AZURE2/include/EData.h	464
/Users/kuba/Desktop/R-Matrix/AZURE2/include/EDataIterator.h	465
/Users/kuba/Desktop/R-Matrix/AZURE2/include/EffectiveCharge.h	467
/Users/kuba/Desktop/R-Matrix/AZURE2/include/EigenFunc.h	467
/Users/kuba/Desktop/R-Matrix/AZURE2/include/EPoint.h	468
/Users/kuba/Desktop/R-Matrix/AZURE2/include/Equation.h	470
/Users/kuba/Desktop/R-Matrix/AZURE2/include/ESegment.h	471
/Users/kuba/Desktop/R-Matrix/AZURE2/include/ExtrapLine.h	472
/Users/kuba/Desktop/R-Matrix/AZURE2/include/GenMatrixFunc.h	473

/Users/kuba/Desktop/R-Matrix/AZURE2/include/GSLEException.h	474
/Users/kuba/Desktop/R-Matrix/AZURE2/include/IntegratedFermiFunc.h	475
/Users/kuba/Desktop/R-Matrix/AZURE2/include/Interference.h	476
/Users/kuba/Desktop/R-Matrix/AZURE2/include/JGroup.h	476
/Users/kuba/Desktop/R-Matrix/AZURE2/include/KGroup.h	477
/Users/kuba/Desktop/R-Matrix/AZURE2/include/KLGroup.h	478
/Users/kuba/Desktop/R-Matrix/AZURE2/include/MatrixInv.h	478
/Users/kuba/Desktop/R-Matrix/AZURE2/include/MGroup.h	479
/Users/kuba/Desktop/R-Matrix/AZURE2/include/NFIntegral.h	480
/Users/kuba/Desktop/R-Matrix/AZURE2/include/NucLine.h	480
/Users/kuba/Desktop/R-Matrix/AZURE2/include/PPair.h	482
/Users/kuba/Desktop/R-Matrix/AZURE2/include/ReactionRate.h	482
/Users/kuba/Desktop/R-Matrix/AZURE2/include/RMatrixFunc.h	484
/Users/kuba/Desktop/R-Matrix/AZURE2/include/SegLine.h	484
/Users/kuba/Desktop/R-Matrix/AZURE2/include/ShftFunc.h	485
/Users/kuba/Desktop/R-Matrix/AZURE2/include/TargetEffect.h	486
/Users/kuba/Desktop/R-Matrix/AZURE2/include/WhitFunc.h	487
/Users/kuba/Desktop/R-Matrix/AZURE2/src/AChannel.cpp	488
/Users/kuba/Desktop/R-Matrix/AZURE2/src/ALevel.cpp	489
/Users/kuba/Desktop/R-Matrix/AZURE2/src/AMatrixFunc.cpp	491
/Users/kuba/Desktop/R-Matrix/AZURE2/src/AngCoeff.cpp	494
/Users/kuba/Desktop/R-Matrix/AZURE2/src/AZURE2.cpp	495
/Users/kuba/Desktop/R-Matrix/AZURE2/src/AZURECalc.cpp	506
/Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREMain.cpp	507
/Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREOutput.cpp	511
/Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREParams.cpp	512
/Users/kuba/Desktop/R-Matrix/AZURE2/src/CNuc.cpp	513
/Users/kuba/Desktop/R-Matrix/AZURE2/src/Config.cpp	531
/Users/kuba/Desktop/R-Matrix/AZURE2/src/CoulFunc.cpp	533
/Users/kuba/Desktop/R-Matrix/AZURE2/src/Decay.cpp	535
/Users/kuba/Desktop/R-Matrix/AZURE2/src/DoubleFactorial.cpp	536
/Users/kuba/Desktop/R-Matrix/AZURE2/src/ECIntegral.cpp	536
/Users/kuba/Desktop/R-Matrix/AZURE2/src/ECMGroup.cpp	539
/Users/kuba/Desktop/R-Matrix/AZURE2/src/EData.cpp	540
/Users/kuba/Desktop/R-Matrix/AZURE2/src/EDataIterator.cpp	552
/Users/kuba/Desktop/R-Matrix/AZURE2/src/EffectiveCharge.cpp	553
/Users/kuba/Desktop/R-Matrix/AZURE2/src/EigenFunc.cpp	554
/Users/kuba/Desktop/R-Matrix/AZURE2/src/EPoint.cpp	555
/Users/kuba/Desktop/R-Matrix/AZURE2/src/Equation.cpp	566
/Users/kuba/Desktop/R-Matrix/AZURE2/src/ESegment.cpp	571
/Users/kuba/Desktop/R-Matrix/AZURE2/src/GenMatrixFunc.cpp	574
/Users/kuba/Desktop/R-Matrix/AZURE2/src/GSLEException.cpp	578
/Users/kuba/Desktop/R-Matrix/AZURE2/src/IntegratedFermiFunc.cpp	578
/Users/kuba/Desktop/R-Matrix/AZURE2/src/Interference.cpp	579
/Users/kuba/Desktop/R-Matrix/AZURE2/src/JGroup.cpp	580
/Users/kuba/Desktop/R-Matrix/AZURE2/src/KGroup.cpp	581
/Users/kuba/Desktop/R-Matrix/AZURE2/src/KLGroup.cpp	582
/Users/kuba/Desktop/R-Matrix/AZURE2/src/MatrixInv.cpp	582
/Users/kuba/Desktop/R-Matrix/AZURE2/src/MGroup.cpp	583
/Users/kuba/Desktop/R-Matrix/AZURE2/src/NFIntegral.cpp	584
/Users/kuba/Desktop/R-Matrix/AZURE2/src/PPair.cpp	584
/Users/kuba/Desktop/R-Matrix/AZURE2/src/ReactionRate.cpp	586
/Users/kuba/Desktop/R-Matrix/AZURE2/src/RMatrixFunc.cpp	589
/Users/kuba/Desktop/R-Matrix/AZURE2/src/ShftFunc.cpp	592
/Users/kuba/Desktop/R-Matrix/AZURE2/src/TargetEffect.cpp	593

Chapter 6

Namespace Documentation

6.1 ROOT Namespace Reference

Namespaces

- namespace [Minuit2](#)

6.2 ROOT::Minuit2 Namespace Reference

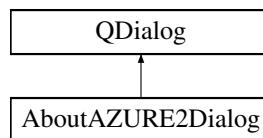
Chapter 7

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 specific 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](#).

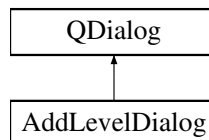
The documentation for this class was generated from the following files:

- [/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](#).

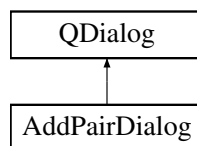
The documentation for this class was generated from the following files:

- [/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](#).

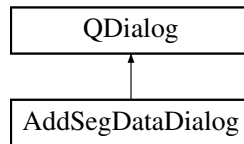
The documentation for this class was generated from the following files:

- /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](#).

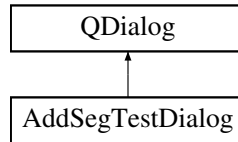
The documentation for this class was generated from the following files:

- [/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](#).

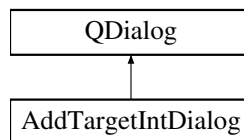
The documentation for this class was generated from the following files:

- [/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 IsECLLevel()

```
bool ALevel::IsECLLevel ( ) 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:

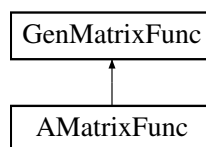
- [/Users/kuba/Desktop/R-Matrix/AZURE2/include/ALevel.h](#)
- [/Users/kuba/Desktop/R-Matrix/AZURE2/src/ALevel.cpp](#)

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 accessible to child class.
- `matrix_c ec_tmatrix_`
Vector of external T-matrix elements accessible 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:

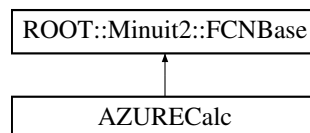
- /Users/kuba/Desktop/R-Matrix/AZURE2/include/[AngCoeff.h](#)
- /Users/kuba/Desktop/R-Matrix/AZURE2/src/[AngCoeff.cpp](#)

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 actual 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](#).

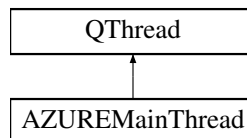
The documentation for this class was generated from the following files:

- /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](#).

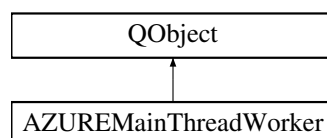
The documentation for this class was generated from the following file:

- [/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](#).

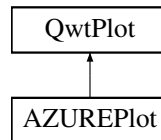
The documentation for this class was generated from the following files:

- /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](#).

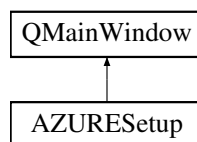
The documentation for this class was generated from the following files:

- [/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](#).

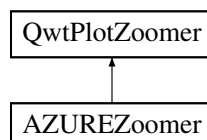
The documentation for this class was generated from the following files:

- [/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](#).

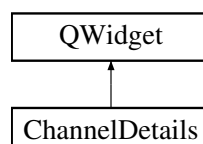
The documentation for this class was generated from the following files:

- /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](#).

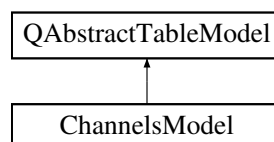
The documentation for this struct was generated from the following file:

- [/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](#).

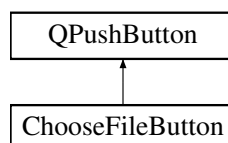
The documentation for this class was generated from the following files:

- [/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_DIST](#)
 =(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_DIST	
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 function 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 PShift_dE()

```
double CoulFunc::PShift_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 IsKGroup()

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

Tests a specific k, L 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 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 NumKGroups()

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

Returns the number of k, L combinations in the [KGroup](#) 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 initial 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)
- 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-sphere 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)
- [EDatalerator](#) begin ()
- [EDatalerator](#) 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 an 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 EDatalterator Class Reference

An iterator class for an [EData](#) object.

```
#include <EDataIterator.h>
```

Public Member Functions

- [EDatalterator](#) (std::vector< [ESegment](#) > *)
- [EDatalterator](#) (const [EDatalterator](#) &it)
- [EDatalterator](#) & operator++ ()
- [EDatalterator](#) operator++ (int)
- bool operator== (const [EDatalterator](#) &)
- bool operator!= (const [EDatalterator](#) &)
- [EDatalterator](#) & SetEnd ()
- [ESegmentIterator](#) & segment ()
- [EPointIterator](#) & point ()

7.36.1 Detailed Description

An iterator class for an [EData](#) object.

The [EDatalterator](#) class is used to loop through all data segments and points contained in an [EData](#) object.

Definition at line 20 of file [EDatalterator.h](#).

7.36.2 Constructor & Destructor Documentation

7.36.2.1 EDatalterator() [1/2]

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

The [EDatalterator](#) object is created with reference to a vector of [ESegment](#) objects.

Definition at line 9 of file [EDatalterator.cpp](#).

7.36.2.2 EDatalterator() [2/2]

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

The copy constructor creates a new [EDatalterator](#) object with reference to an already existing object.

Definition at line 20 of file [EDatalterator.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 [EDatalterator.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 [EDatalterator.cpp](#).

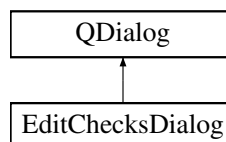
The documentation for this class was generated from the following files:

- [/Users/kuba/Desktop/R-Matrix/AZURE2/include/EDatalterator.h](#)
- [/Users/kuba/Desktop/R-Matrix/AZURE2/src/EDatalterator.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 * [IMatrixCheckCombo](#)
- 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](#).

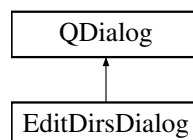
The documentation for this class was generated from the following files:

- [/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](#).

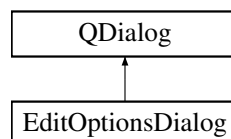
The documentation for this class was generated from the following files:

- [/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 calculates 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 initialization 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 initialization 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 simultaneously. 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 shifts.

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
```

Returns 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. Initialization is done before the fitting process to calculate all energy dependent quantities that do not 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 involved 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 specifying 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](#).

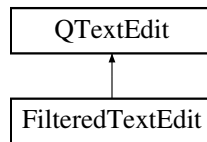
The documentation for this class was generated from the following file:

- [/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 mouseClickedEvent()

```
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:

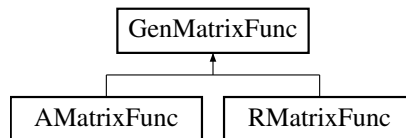
- [/Users/kuba/Desktop/R-Matrix/AZURE2/include/Equation.h](#)

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 accessible to child class.
- [matrix_c](#) [ec_tmatrix_](#)
Vector of external T-matrix elements accessible 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 is implemented in the child class.

Implemented in [AMatrixFunc](#), and [RMatrixFunc](#).

7.49.3.6 ClearMatrices()

```
virtual void GenMatrixFunc::ClearMatrices ( ) [pure virtual]
```

This virtual function is 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 is 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 is 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 is 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 is 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 accessible 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 accessible 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](#).

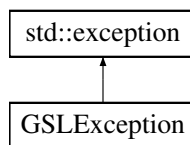
The documentation for this struct was generated from the following file:

- [/Users/kuba/Desktop/R-Matrix/AZURE2/src/ReactionRate.cpp](#)

7.51 GSLEException Class Reference

```
#include <GSLEException.h>
```

Inheritance diagram for GSLEException:



Public Member Functions

- [GSLEException](#) (std::string message, std::string line="", std::string file="")
- [~GSLEException](#) () 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 [GSLEException](#) class is an exception class thrown by the [CoulFunc](#) class. It should not be used directly.

Definition at line 13 of file [GSLEException.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](#).

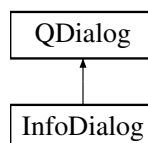
The documentation for this class was generated from the following files:

- [/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 $m_0 c^2$), 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 differential 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 specific [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](#).

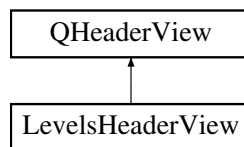
The documentation for this struct was generated from the following file:

- [/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](#).

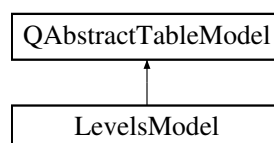
The documentation for this class was generated from the following file:

- [/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](#).

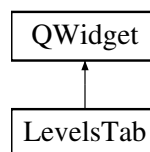
The documentation for this class was generated from the following files:

- [/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/lnTabDocs.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 accessible 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 IFinal, 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 separation 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 deprecated 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 deprecated 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 deprecated 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 deprecated 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](#).

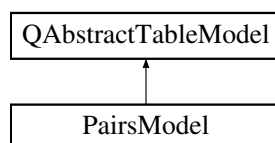
The documentation for this struct was generated from the following file:

- [/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](#).

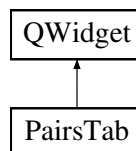
The documentation for this class was generated from the following files:

- [/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](#).

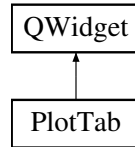
The documentation for this struct was generated from the following file:

- [/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/lnTabDocs.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 an 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 separation 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](#).

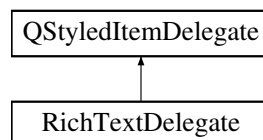
The documentation for this class was generated from the following files:

- [/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:

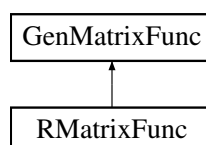
- [/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RichTextDelegate.h](#)
- [/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RichTextDelegate.cpp](#)

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 accessible to child class.
- [matrix_c](#) [ec_tmatrix_](#)
Vector of external T-matrix elements accessible 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 is 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](#).

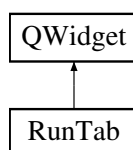
The documentation for this class was generated from the following files:

- [/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](#).

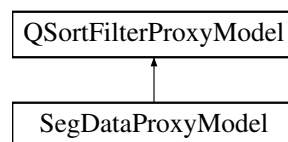
The documentation for this class was generated from the following files:

- [/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 if 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](#).

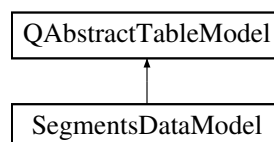
The documentation for this struct was generated from the following file:

- [/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](#).

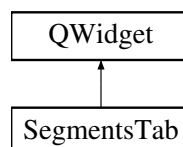
The documentation for this class was generated from the following files:

- [/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](#).

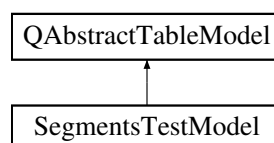
The documentation for this struct was generated from the following file:

- [/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](#).

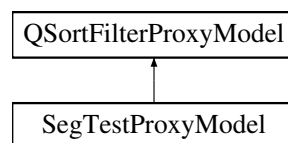
The documentation for this struct was generated from the following files:

- [/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 calculate 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:

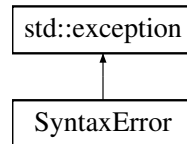
- [/Users/kuba/Desktop/R-Matrix/AZURE2/include/ShftFunc.h](#)
- [/Users/kuba/Desktop/R-Matrix/AZURE2/src/ShftFunc.cpp](#)

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.

Experimental 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 AZURE-Setup2.

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::istream` 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/cm². 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](#).

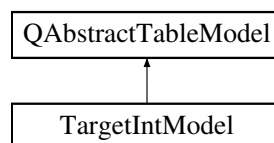
The documentation for this struct was generated from the following file:

- [/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](#).

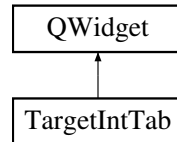
The documentation for this class was generated from the following files:

- [/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 & outputStream )
```

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/IntTabDocs.cpp](#)
- [/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntTab.cpp](#)

7.95 TempTMatrix Struct Reference

A temporary 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 temporary 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 primarily 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](#).

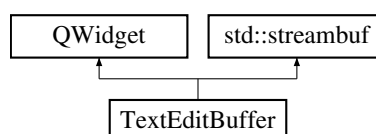
The documentation for this struct was generated from the following file:

- [/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_l_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_l_omega_chi_calc()

```
std::complex< double > exp_l_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     return (finite (x) && finite (y));
00028 }
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 }
```

```

00070
00071 inline std::complex<double> operator / (const int n,const std::complex<double> &z)
00072 {
00073     return (static_cast<double> (n)/z);
00074 }
00075
00076 inline std::complex<double> operator + (const std::complex<double> &z,const unsigned int n)
00077 {
00078     return (z+static_cast<double> (n));
00079 }
00080
00081 inline std::complex<double> operator - (const std::complex<double> &z,const unsigned int n)
00082 {
00083     return (z-static_cast<double> (n));
00084 }
00085
00086 inline std::complex<double> operator * (const std::complex<double> &z,const unsigned int n)
00087 {
00088     return (z*static_cast<double> (n));
00089 }
00090
00091 inline std::complex<double> operator / (const std::complex<double> &z,const unsigned int n)
00092 {
00093     return (z/static_cast<double> (n));
00094 }
00095
00096 inline std::complex<double> operator + (const unsigned int n,const std::complex<double> &z)
00097 {
00098     return (static_cast<double> (n)+z);
00099 }
00100
00101 inline std::complex<double> operator - (const unsigned int n,const std::complex<double> &z)
00102 {
00103     return (static_cast<double> (n)-z);
00104 }
00105
00106 inline std::complex<double> operator * (const unsigned int n,const std::complex<double> &z)
00107 {
00108     return (static_cast<double> (n)*z);
00109 }
00110
00111 inline std::complex<double> operator / (const unsigned int n,const std::complex<double> &z)
00112 {
00113     return (static_cast<double> (n)/z);
00114 }
00115
00116 inline bool operator == (const std::complex<double> &z,const int n)
00117 {
00118     return (z == static_cast<double> (n));
00119 }
00120
00121 inline bool operator != (const std::complex<double> &z,const int n)
00122 {
00123     return (z != static_cast<double> (n));
00124 }
00125
00126 inline bool operator == (const int n,const std::complex<double> &z)
00127 {
00128     return (static_cast<double> (n) == z);
00129 }
00130
00131 inline bool operator != (const int n,const std::complex<double> &z)
00132 {
00133     return (static_cast<double> (n) != z);
00134 }
00135
00136 inline bool operator == (const std::complex<double> &z,const unsigned int n)
00137 {
00138     return (z == static_cast<double> (n));
00139 }
00140
00141 inline bool operator != (const std::complex<double> &z,const unsigned int n)
00142 {
00143     return (z != static_cast<double> (n));
00144 }
00145
00146 inline bool operator == (const unsigned int n,const std::complex<double> &z)
00147 {
00148     return (static_cast<double> (n) == z);
00149 }
00150
00151 inline bool operator != (const unsigned int n,const std::complex<double> &z)
00152 {
00153     return (static_cast<double> (n) != z);
00154 }
00155
00156 extern std::complex<double> expm1 (const std::complex<double> &z);

```

```

00157
00158 extern std::complex<double> log1p (const std::complex<double> &z);
00159
00160 extern std::complex<double> log_Gamma (const std::complex<double> &z);
00161
00162 extern std::complex<double> sigma_l_calc (const std::complex<double> &l,const std::complex<double>
&eta);
00163
00164 extern std::complex<double> log_Cl_eta_calc (const std::complex<double> &l,const std::complex<double>
&eta);
00165
00166 extern std::complex<double> log_cut_constant_AS_calc (const int omega,const std::complex<double>
&l,const std::complex<double> &eta);
00167
00168 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);
00169
00170 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);
00171
00172 extern std::complex<double> sin_chi_calc (const std::complex<double> &l,const std::complex<double>
&eta);
00173
00174 extern std::complex<double> exp_I_omega_chi_calc (const int omega,const std::complex<double> &l,const
std::complex<double> &eta);
00175
00176 #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.](#)

```

00001 #ifndef CWFCOMP_H
00002 #define CWFCOMP_H
00003
00004 #include "ode_int.H"
00005
00006 // Class to calculate the Coulomb wave functions
00007 // -----
00008
00009 class Coulomb_wave_functions
00010 {
00011 public:
00012
00013 // Constructor.
00014 // -----
00015 // Constants are defined in the constructor,
00016 // plus a pointer to class ODE_integration, ODE_ptr, to integrate numerically the regular Coulomb
wave function.
00017 //
00018 // Variables:
00019 // -----
00020 // is_it_normalized_c : true if one wants normalized functions, i.e. the standard normalization,
00021 // 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.
00022 // l_c : orbital angular momentum.
00023 // eta_c : Sommerfeld parameter.
00024
00025 Coulomb_wave_functions (const bool is_it_normalized_c,const std::complex<double> &l_c,const
std::complex<double> &eta_c)
00026 : l (l_c),

```

```

00027         is_it_normalized (is_it_normalized_c),
00028         eta (eta_c),
00029         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))) &&
00030         (rint (real (1 + l_c + std::complex<double> (-imag (eta_c), real (eta_c)))) <= 0.0)),
00031         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))) &&
00032         (rint (real (1 + l_c - std::complex<double> (-imag (eta_c), real (eta_c)))) <=
0.0)),
00033         sigma_l (sigma_l_calc (l_c, eta_c)),
00034         log_Cl_eta (log_Cl_eta_calc (l_c, eta_c)),
00035         Cl_eta (exp (log_Cl_eta_calc (l_c, eta_c))),
00036         exp_I_chi (exp_I_omega_chi_calc (l, l_c, eta_c)),
00037         exp_minus_I_chi (exp_I_omega_chi_calc (-l, l_c, eta_c)),
00038         one_over_sin_chi (1.0/sin_chi_calc (l_c, eta_c)),
00039         log_cut_constant_CFa_plus (log_cut_constant_CFa_calc (is_it_normalized_c, l, l_c, eta_c)),
00040         log_cut_constant_CFa_minus (log_cut_constant_CFa_calc (is_it_normalized_c, -l, l_c, eta_c)),
00041         cut_constant_CFa_plus (exp (log_cut_constant_CFa_calc (is_it_normalized_c, l, l_c, eta_c))),
00042         cut_constant_CFa_minus (exp (log_cut_constant_CFa_calc (is_it_normalized_c, -l, l_c, eta_c))),
00043         log_cut_constant_CFa_plus (log_cut_constant_CFa_calc (is_it_normalized_c, l, l_c, eta_c)),
00044         log_cut_constant_CFa_minus (log_cut_constant_CFa_calc (is_it_normalized_c, -l, l_c, eta_c)),
00045         log_cut_constant_AS_plus (log_cut_constant_AS_calc (l, l_c, eta_c)),
00046         log_cut_constant_AS_minus (log_cut_constant_AS_calc (-l, l_c, eta_c)),
00047         cut_constant_CFa_plus (exp (log_cut_constant_CFa_calc (is_it_normalized_c, l, l_c, eta_c))),
00048         cut_constant_CFa_minus (exp (log_cut_constant_CFa_calc (is_it_normalized_c, -l, l_c, eta_c))),
00049         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))),
00050         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))),
00051         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)))),
00052         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)))),
00053         turning_point (std::max (1.0, abs (eta_c) + sqrt (abs (l_c*(l_c+1.0)) + abs (eta_c*eta_c))),
00054         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),
00055         cwf_minus_eta_ptr (0), cwf_lp_ptr (0), prec_first_order_expansion (0.1*sqrt_precision)
00056     {
00057         ODE_ptr = new class ODE_integration (l, 2.0*eta);
00058
00059         debut = 0.0;
00060
00061         if (real (l) >= 0.0)
00062         {
00063             F_debut = 0.0;
00064             dF_debut = (l == 0) ? ((is_it_normalized) ? (Cl_eta) : (1.0)) : (0.0);
00065         }
00066     }
00067
00068     ~Coulomb_wave_functions (void)
00069     {
00070         delete cwf_real_ptr;
00071
00072         delete cwf_real_l_plus_ptr;
00073         delete cwf_real_l_minus_ptr;
00074
00075         delete cwf_real_eta_plus_ptr;
00076         delete cwf_real_eta_minus_ptr;
00077
00078         delete cwf_minus_eta_ptr;
00079         delete cwf_lp_ptr;
00080
00081         delete ODE_ptr;
00082     }
00083
00084     void F_dF_init (const std::complex<double> &z, const std::complex<double> &F, const
std::complex<double> &dF);
00085
00086     void F_dF (const std::complex<double> &z, std::complex<double> &F, std::complex<double> &dF);
00087     void G_dG (const std::complex<double> &z, std::complex<double> &G, std::complex<double> &dG);
00088     void H_dH (const int omega, const std::complex<double> &z, std::complex<double>
&H, std::complex<double> &dH);
00089     void H_dH_scaled (const int omega, const std::complex<double> &z, std::complex<double>
&H, std::complex<double> &dH);
00090
00091     const std::complex<double> l, eta; // Angular momentum and Sommerfeld parameter.
00092
00093     const bool is_it_normalized;
00094     // true if  $F(z) \sim C(l, \eta) z^{l+1}$  in 0, false if  $F(z) \sim z^{l+1}$  in 0.
00095
00096 private:
00097
00098     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);
00099
00099     std::complex<double> continued_fraction_f (const std::complex<double> &z, const int omega);
00100     std::complex<double> continued_fraction_h (const std::complex<double> &z, const int omega);

```

```

00102
00103     void F_dF_power_series (const std::complex<double> &z, std::complex<double> &F, std::complex<double>
00104                             &dF);
00105     void asymptotic_expansion_F_dF (const std::complex<double> &z, std::complex<double>
00106                                     &F, std::complex<double> &dF, bool &is_it_successful);
00107     void asymptotic_expansion_H_dH_scaled (const int omega, const std::complex<double> &one_over_z,
00108                                             std::complex<double> &H_scaled, std::complex<double> &dH_scaled, bool
00109                                             &is_it_successful);
00110     void F_dF_direct_integration (const std::complex<double> &z, std::complex<double>
00111                                   &F, std::complex<double> &dF, bool &is_it_successful);
00112     void H_dH_direct_integration (const int omega, const std::complex<double> &z, std::complex<double>
00113                                   &H, std::complex<double> &dH);
00114     void partial_derivatives (const bool is_it_regular, const bool is_it_eta, const double x, double
00115                               &d_chi_Bx, double &d_chi_dBx);
00116     void first_order_expansions (const bool is_it_regular, const std::complex<double>
00117                                  &z, std::complex<double> &B, std::complex<double> &dB);
00118     void H_dH_from_first_order_expansions (const int omega, const std::complex<double>
00119                                              &z, std::complex<double> &H, std::complex<double> &dH);
00120     void H_dH_with_F_dF_and_CF (const int omega, const std::complex<double> &z, std::complex<double>
00121                                  &H, std::complex<double> &dH);
00122     void H_dH_with_expansion (const int omega, const std::complex<double> &z, std::complex<double>
00123                               &H, std::complex<double> &dH, bool &is_it_successful);
00124     void F_dF_with_symmetry_relations (const std::complex<double> &z, std::complex<double>
00125                                         &F, std::complex<double> &dF);
00126
00127     const bool neg_int_omega_one, neg_int_omega_minus_one;
00128     // neg_int_omega_one : true if 1+l+i.eta is negative integer, false if not.
00129     // neg_int_omega_minus_one : true if 1+l-i.eta is negative integer, false if not.
00130
00131     const std::complex<double> log_C1_eta, C1_eta, sigma_l; // log[C(1,eta)], C(1,eta), sigma(1,eta)
00132
00133     const std::complex<double>
00134     cut_constant_CFa_plus, cut_constant_CFa_minus, cut_constant_Cfb_plus, cut_constant_Cfb_minus;
00135     const std::complex<double>
00136     log_cut_constant_CFa_plus, log_cut_constant_CFa_minus, log_cut_constant_Cfb_plus, log_cut_constant_Cfb_minus;
00137     const std::complex<double> log_cut_constant_AS_plus, log_cut_constant_AS_minus;
00138     // cut constants and their logs for continued fractions (CFa and Cfb) and asymptotic series (AS).
00139     // plus, minus is for omega = 1 or -1.
00140     // See functions log_cut_constant_AS_calc, log_cut_constant_CFa_calc and log_cut_constant_Cfb_calc.
00141
00142     const std::complex<double> exp_I_chi, exp_minus_I_chi, one_over_sin_chi;
00143     // exp (i.chi), exp (-i.chi), 1/sin (chi) with chi = sigma(1,eta) - sigma(-1,eta) - (1+1/2).Pi .
00144     // They are used to calculate H+/H- from F(1,eta,z) and F(-1,eta,z) if |Im[l]| >= 1 and |z| <= 1 .
00145
00146     const std::complex<double>
00147     sym_constant_arg_neg, sym_constant_arg_pos, log_sym_constant_arg_neg, log_sym_constant_arg_pos;
00148     // Multiplicative constants and their logs used in the following reflection formulas :
00149     // F(1,eta,z) = -F(1,-eta,-z).exp[-Pi.(eta-i.l)] if arg (z) > 0 and is_it_normalized is true, so
00150     sym_constant_arg_pos = -exp[-Pi.(eta-i.l)],
00151     // F(1,eta,z) = -F(1,-eta,-z).exp[-Pi.(eta+i.l)] if arg (z) <= 0 and is_it_normalized is true, so
00152     sym_constant_arg_neg = -exp[-Pi.(eta+i.l)],
00153     // F(1,eta,z) = -F(1,-eta,-z).exp[i.Pi.l] if arg (z) > 0 and is_it_normalized is false, so
00154     sym_constant_arg_pos = -exp[i.Pi.l],
00155     // F(1,eta,z) = -F(1,-eta,-z).exp[-i.Pi.l] if arg (z) <= 0 and is_it_normalized is false, so
00156     sym_constant_arg_neg = -exp[-i.Pi.l].
00157
00158     const double turning_point, prec_first_order_expansion; // turning_point : max (1, |eta| +
00159                                                             sqrt[|l(l+1)| + |eta|^2]).
00160     // prec_first_order_expansion :
00161     0.1*sqrt_precision. It is the precision used for first_order_expansions.
00162
00163     bool is_H_dir_int_naive; // true if one integrates H+/H- forward without considering |H+/H-|, false
00164                             if not. It is false except in continued_fraction_h.
00165
00166     std::complex<double> debut, F_debut, dF_debut;
00167     // Coulomb wave functions and derivative at z = debut.
00168     // It is used to integrate the Coulomb wave function faster,
00169     // as debut is usually close to the argument of the Coulomb wave function so that the integration is
00170     // quicker and more stable.
00171
00172     class ODE_integration *ODE_ptr; // pointer to class ODE_integration to integrate numerically the
00173                                     Coulomb equation.
00174
00175     class Coulomb_wave_functions
00176     *cwf_real_ptr, *cwf_real_l_plus_ptr, *cwf_real_l_minus_ptr, *cwf_real_eta_plus_ptr, *cwf_real_eta_minus_ptr;
00177     // pointers to classes Coulomb_wave_functions of parameters (l_r, eta_r) (one has eta_r = Re[eta],
00178     eta_i = Im[eta], l_r = Re[l] and l_i = Im[l]),
00179     // (l_r +/- epsilon[l], eta_r) and (l_r, eta_r +/- epsilon[eta]).
00180     // They are first put to zero and allocated in the program when they are needed.
00181     // They are used for the first order expansion method when |l_i| < 1, |eta_i| < 1 and |Im[z]| <
00182     Re[z] with Re[z] > 0.
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000

```

```

00163     class Coulomb_wave_functions *cwf_minus_eta_ptr,*cwf_lp_ptr;
00164     // pointers to classes Coulomb_wave_functions of parameters (l,-eta), (lp = -l-1,eta) and (l_r +/-
precision,eta).
00165     // They are first put to zero and allocated in the program when they are needed.
00166     // 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)
00167     // and to calculate H+/H- from F(l,eta,z) and F(lp,eta,z) if |Im[l]| >= 1 and |z| <= 1.
00168 };
00169
00170 #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.](#)

```

00001 #ifndef ODE_INT_H
00002 #define ODE_INT_H
00003
00004 #include "complex_functions.H"
00005
00006 // Direct integration of the Coulomb equation
00007 // -----
00008 // One uses the Burlisch-Stoer-Henrici method, where one integrates on different meshes
00009 // with the Henrici method, and then uses the Richardson method to get the final result by
extrapolation.
00010 // Numerical Recipes, Chap. 16.4 .
00011
00012 class ODE_integration
00013 {
00014 public:
00015     ODE_integration (const std::complex<double> &l_1,
00016                     const std::complex<double> &two_eta_1)
00017         : l (l_1), ll_plus_one (l_1*(l_1+1.0)), two_eta (two_eta_1)
00018     {
00019         for (int n = 0 ; n < 8 ; n++)
00020             for (int i = 0 ; i < n ; i++)
00021             {
00022                 interpolation_term_tab[n][i] = 1.0;
00023
00024                 for (int j = 0 ; j < n ; j++)
00025                     if (i != j)
00026                         interpolation_term_tab[n][i] *= (i+1.0)/(i-j);
00027             }
00028
00029             for (unsigned int k = 0 ; k < 8 ; k++) m_tab[k] = 2*(k+1);
00030             for (unsigned int k = 0 ; k < 8 ; k++) one_over_m_tab[k] = 1.0/static_cast<double> (m_tab[k]);
00031     }
00032
00033     void operator() (const std::complex<double> &r0,const std::complex<double> &u0,const
std::complex<double> &du0,
00034                     const std::complex<double> &r,std::complex<double> &u,std::complex<double> &du) const;
00035
00036 private:
00037     std::complex<double> extrapolation_in_zero (const unsigned int n,const std::complex<double> *f)
const;
00038     std::complex<double> F_r_u (const std::complex<double> &r,const std::complex<double> &u) const;
00039     void integration_Henrici (const unsigned int m,const std::complex<double> &h,
00040                             const std::complex<double> &r0,const std::complex<double> &u0,const
std::complex<double> &du0,
00041                             const std::complex<double> &r,std::complex<double> &u,std::complex<double> &du) const;
00042

```



```

00043     const std::complex<double> l,l_plus_one; // angular momentum,l(l+1).
00044     const std::complex<double> two_eta; // 2.eta, with eta the Sommerfeld parameter.
00045
00046     unsigned int m_tab[8]; // integers used in the extrapolation method.
00047     double one_over_m_tab[8],interpolation_term_tab[8][8]; // doubles used in the extrapolation method.
00048 };
00049
00050
00051 #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_l_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_l_omega_chi_calc()`

```

std::complex< double > exp_l_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.](#)

```
00001 #include "complex_functions.H"
00002
00003 // Precise evaluation of exp[z]-1 for z complex
00004 // -----
00005 // When |Re[z]| >= 1 or |Im[z]| >= 1, one uses directly the standard exp function as it is precise.
00006 // Otherwise, numerical cancellations can occur.
00007 // So, one uses the always stable formula exp[z]-1 = expml(x) - 2.exp(x).sin^2(y/2) + i.exp(x).sin(y)
00008 // with x = Re[z] and y = Im[z]. expml(x) gives a precise evaluation of exp(x)-1 for x double.
00009
00010 std::complex<double> expml (const std::complex<double> &z)
00011 {
00012     const double x = real (z), y = imag (z);
00013
00014     if ((std::abs (x) >= 1.0) || (std::abs (y) >= 1.0)) return (exp (z) - 1.0);
00015
00016     const double expml_x = expml (x), exp_x = 1.0 + expml_x, sin_y_over_two = sin (0.5*y), sin_y = sin (y);
00017
00018     return std::complex<double> (expml_x - 2.0*exp_x*sin_y_over_two*sin_y_over_two, exp_x*sin_y);
00019 }
00020
00021
00022
00023 // Precise evaluation of log[1+z] for z complex
00024 // -----
00025 // When |Re[z]| >= 1 or |Im[z]| >= 1, one uses directly the standard log function as it is precise.
00026 // Otherwise, numerical cancellations can occur.
00027 // So, one uses the always stable formula log[1+z] = loglp(x) + loglp([y/(1+x)]^2)/2 + i.atan2(y,1+x)
00028 // with x = Re[z] and y = Im[z]. loglp(x) gives a precise evaluation of log(1+x) for x double.
00029 // atan2(x,y) gives the arc tangent of y/x so it is in ]-Pi:Pi].
00030
00031 std::complex<double> loglp (const std::complex<double> &z)
00032 {
00033     const double x = real (z), y = imag (z);
00034
00035     const double xpl = 1.0 + x, abs_x = std::abs (x), abs_y = std::abs (y);
00036
00037     if ((abs_x >= 1.0) || (abs_y >= 1.0)) return log (1.0 + z);
00038
00039     const double y_over_xpl = y/xpl;
00040
00041     return std::complex<double> (loglp (x) + 0.5*loglp (y_over_xpl*y_over_xpl), atan2 (y,xpl));
00042 }
00043
00044
00045
00046
00047 // Logarithm of Gamma[z], z anywhere in the complex plane except in the Gamma function poles.
00048 // -----
00049 // If z is not finite or is a negative integer, the program returns an error message and stops.
00050 // The Lanczos method is used. Precision : < 2E-10 in theory, < 1E-12 in almost every case.
00051 // The method works for Re[z] > 0.
00052 // If Re[z] <= 0, one uses the formula Gamma[z].Gamma[1-z] = Pi/sin (Pi.z).
```

```

00053 // log[sin(Pi.z)] is calculated with the Kolbig method (K.S. Kolbig, Comp. Phys. Comm., Vol. 4, p.221
(1972)) :
00054 // 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.
00055 // If y > 110, log[sin(Pi.z)] = -i.Pi.z + log[0.5] + i.Pi/2 numerically so that no overflow can occur.
00056 // If z = x+iy and y < 0, log[Gamma(z)] = [log[Gamma(z*)]]*, so that one can use the previous formula
with z*.
00057 //
00058 // Variables:
00059 // -----
00060 // z,z_p_0p5,z_p_5p5 : argument of the Gamma function, z+0.5, z+5.5
00061 // sqrt_2Pi,log_Pi : sqrt(2.Pi), log(Pi).
00062 // sum : Rational function in the Lanczos method.
00063 // log_Gamma_z : log[Gamma(z)] value.
00064 // c : table containing the seven coefficients in the expansion used in the Lanczos method.
00065 // eps : z = n + eps so 0 <= Re[eps] < 1 and n integer.
00066 // log_const : log[0.5] + i.Pi/2
00067
00068 std::complex<double> log_Gamma (const std::complex<double> &z)
00069 {
00070     if (!isfinite (z)) std::cout<<"z is not finite in log_Gamma."<<std::endl, abort ();
00071
00072     const double x = real (z),y = imag (z);
00073
00074     if ((z == rint (x)) && (x <= 0)) std::cout<<"z is negative integer in log_Gamma."<<std::endl, abort
());
00075
00076     if (x > 0.0)
00077     {
00078         const std::complex<double> z_p_0p5 = z + 0.5, z_p_5p5 = z + 5.5;
00079         const double sqrt_2Pi = 2.5066282746310005;
00080         const double c[7] = {1.000000000190015,
00081             7.618009172947146E+1,
00082             -8.650532032941677E+1,
00083             2.401409824083091E+1,
00084             -1.231739572450155,
00085             0.1208650973866179E-2,
00086             -0.5395239384953000E-5};
00087
00088         std::complex<double> sum = c[0];
00089         for (int i = 1 ; i < 7 ; i++) sum += c[i]/(z + i);
00090         sum *= sqrt_2Pi;
00091
00092         const std::complex<double> log_Gamma_z = log (sum) - log (z) + z_p_0p5*log (z_p_5p5) - z_p_5p5;
00093
00094         return log_Gamma_z;
00095     }
00096     else if (y >= 0.0)
00097     {
00098         const int n = (x < rint (x)) ? (static_cast<int> (rint (x)) - 1) : (static_cast<int> (rint (x)));
00099         const double log_Pi = 1.1447298858494002;
00100         const std::complex<double> log_const (-M_LN2,M_PI_2),i_Pi (0.0,M_PI);
00101         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);
00102         const std::complex<double> log_Gamma_z = log_Pi - log_sin_Pi_z - log_Gamma (1.0 - z);
00103
00104         return log_Gamma_z;
00105     }
00106     else
00107         return conj (log_Gamma (conj (z)));
00108 }
00109
00110
00111
00112 // Coulomb phase shift.
00113 // -----
00114 // It is given by the formula [Gamma[1+l+I.eta] - Gamma[1+l-I.eta]]/[2i].
00115 // 0 is returned if 1+l+/-I.eta is a negative integer.
00116 //
00117 // Variables:
00118 // -----
00119 // l : orbital angular momentum l.
00120 // eta : Sommerfeld parameter.
00121 // Ieta,one_over_two_I : i.eta,1/[2.i] .
00122 // arg_plus,arg_minus : 1+l+i.eta, 1+l-i.eta.
00123 // log_Gamma_plus,log_Gamma_minus : logs of Gamma[1+l+I.eta], Gamma[1+l-I.eta].
00124 // sigma_l : returned result.
00125
00126 std::complex<double> sigma_l_calc (const std::complex<double> &l,const std::complex<double> &eta)
00127 {
00128     const std::complex<double> Ieta(-imag (eta),real (eta)),one_over_two_I(0,-0.5);
00129     const std::complex<double> arg_plus = 1.0 + l + Ieta, arg_minus = 1.0 + l - Ieta;
00130
00131     if ((rint (real (arg_plus)) == arg_plus) && (rint (real (arg_plus)) <= 0.0)) return 0.0;
00132     if ((rint (real (arg_minus)) == arg_minus) && (rint (real (arg_minus)) <= 0.0)) return 0.0;
00133
00134     const std::complex<double> log_Gamma_plus = log_Gamma (arg_plus),log_Gamma_minus = log_Gamma

```

```

    (arg_minus);
00135     const std::complex<double> sigma_l = (log_Gamma_plus - log_Gamma_minus)*one_over_two_I;
00136
00137     return sigma_l;
00138 }
00139
00140
00141
00142
00143
00144
00145 // log of C(l,eta)
00146 // -----
00147 // It is given by the formula  $l \cdot \log[2] - \eta \cdot \pi/2 + (\log[\Gamma[1+l+i\eta]] + \log[\Gamma[1+l-i\eta]])/2.0 - \log[\Gamma[2l+2]]$ .
00148 // 0 is returned if  $1+l \pm i\eta$  is a negative integer.
00149 //  $2l+2$  should not be a negative integer : one has to use  $-l-1$  instead of  $l$  in this case.
00150 //
00151 // Variables:
00152 // -----
00153 // l : orbital angular momentum l.
00154 // eta : Sommerfeld parameter.
00155 // Ieta : i.eta .
00156 // arg_plus,arg_minus :  $1+l+i\eta$ ,  $1+l-i\eta$ .
00157 // log_Gamma_plus,log_Gamma_minus,log_Gamma_2l_plus_2 : logs of  $\Gamma[1+l+i\eta]$ ,  $\Gamma[1+l-i\eta]$ ,  $\Gamma[2l+2]$ .
00158 // log_Cl_eta : returned result.
00159
00160 std::complex<double> log_Cl_eta_calc (const std::complex<double> &l,const std::complex<double> &eta)
00161 {
00162     const std::complex<double> Ieta(-imag (eta),real (eta));
00163     const std::complex<double> arg_plus = 1.0 + l + Ieta, arg_minus = 1.0 + l - Ieta;
00164
00165     if ((rint (real (arg_plus)) == arg_plus) && (rint (real (arg_plus)) <= 0.0)) return 0.0;
00166     if ((rint (real (arg_minus)) == arg_minus) && (rint (real (arg_minus)) <= 0.0)) return 0.0;
00167
00168     const std::complex<double> log_Gamma_plus = log_Gamma (arg_plus),log_Gamma_minus = log_Gamma
00169 (arg_minus),log_Gamma_2l_plus_2 = log_Gamma (2.0*l + 2.0);
00170     const std::complex<double> log_Cl_eta = l*M_LN2 - M_PI_2*eta + 0.5*(log_Gamma_plus +
00171 log_Gamma_minus) - log_Gamma_2l_plus_2;
00172
00173     return log_Cl_eta;
00174 }
00175
00176 // Cut constant log for the asymptotic series.
00177 // -----
00178 // The asymptotic series and  $H[\omega]$  behave differently near the negative real axis.
00179 // Then, if one is in the bad quadrant of  $H[\omega]$ , one has to take into account the cut directly.
00180 // One is in the bad quadrant of  $H[\omega]$  if  $\text{Re}[z] < 0.0$  and  $\text{sign}(\text{Im}[z]) = -\omega$ .
00181 //
00182 //  $H[\omega] = [H[\omega] \text{ from asymptotic function formula}] + (1 - \exp(2.i.\pi.(i.\eta - l.\omega))).[H[-\omega] \text{ from asymptotic function formula}]$ 
00183 //
00184 //
00185 // The cut constant log is then  $\log [1 - \exp(2.i.\pi.(i.\eta - l.\omega))]$ .
00186 // Its returned imaginary part is not necessarily in  $]-\pi;\pi]$ 
00187 //
00188 //
00189 // Variables:
00190 // -----
00191 // omega : 1 or -1.
00192 // l : orbital angular momentum l.
00193 // eta : Sommerfeld parameter.
00194 // Ieta : i.eta
00195 // l_int,Ieta_int : closest integers to  $\text{Re}[l]$ , $\text{Re}[i.\eta]$ 
00196 // eps :  $(Ieta - Ieta\_int) - (l.\omega - l\_int.\omega)$ 
00197 // two_I_Pi, two_I_Pi_eps :  $2.i.\pi$ ,  $2.i.\pi.\text{eps}$  .
00198 // log_cut_constant : returned result.
00199
00200 std::complex<double> log_cut_constant_AS_calc (const int omega,const std::complex<double> &l,const
00201 std::complex<double> &eta)
00202 {
00203     const std::complex<double> Ieta(-imag (eta),real (eta));
00204     const double l_int = rint (real (l)), Ieta_int = rint (real (Ieta));
00205     const std::complex<double> eps = (Ieta - Ieta_int) - omega*(l - l_int);
00206
00207     const std::complex<double> two_I_Pi(0,2.0*M_PI),two_I_Pi_eps = two_I_Pi*eps;
00208
00209     if (real (two_I_Pi_eps) > -0.1)
00210     {
00211         const std::complex<double> log_cut_constant = log (expm1 (-two_I_Pi_eps)) + two_I_Pi_eps;
00212         return log_cut_constant;
00213     }
00214     else

```

```

00215 {
00216     const std::complex<double> log_cut_constant = log1p (-exp (two_I_Pi_eps));
00217
00218     return log_cut_constant;
00219 }
00220 }
00221
00222
00223
00224
00225
00226 // Cut constant log for continued fractions : H[omega] from H[omega, not corrected] case.
00227 // -----
00228 // The continued fraction has no cut on the negative real axis, whereas H[omega] has one.
00229 // Then, if one is in the bad quadrant of H[omega], one has to take into account the cut directly.
00230 // One is in the bad quadrant of H[omega] if Re[z] < 0.0 and sign(Im[z]) = -omega.
00231 //
00232 // H[omega] = H[omega, not corrected] - cut_constant.F .
00233 //
00234 // The cut constant is 2i.omega.norm.(exp (2.i.Pi.[l.omega - i.eta]) - 1), and one takes its log.
00235 // The imaginary part of the log is not necessarily in ]-Pi:Pi].
00236 // Norm is 1.0 for normalized wave functions, C(l,eta)^2 for unnormalized wave functions.
00237 //
00238 //
00239 // Variables:
00240 // -----
00241 // is_it_normalized : true if one wants normalized functions, i.e. the standard normalization,
00242 //                     false if one wants F -> F/C(l,eta) and H+/H-/G -> H+/H-/G.C(l,eta), to avoid
00243 //                     overflows for |eta| » 1 and |z| small.
00244 // omega : 1 or -1.
00245 // l : orbital angular momentum l.
00246 // eta : Sommerfeld parameter.
00247 // Ieta : i.eta .
00248 // l_int, Ieta_int : closest integers to Re[l], Re[i.eta]
00249 // eps : (l.omega - l_int.omega) - (Ieta - Ieta_int)
00250 // log_norm : log[C(l,eta)^2] if is_it_normalized is false, 0.0 if it is true.
00251 // two_I_Pi, two_I_Pi_eps : 2.i.Pi, 2.i.Pi.eps .
00252 // log_two_I_omega : log[2.i.omega] = log[2] + i.omega.Pi/2 .
00253 // log_cut_constant : returned result.
00254
00255 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)
00256 {
00257     const std::complex<double> Ieta(-imag (eta), real (eta));
00258     const double l_int = rint (real (l)), Ieta_int = rint (real (Ieta));
00259     const std::complex<double> eps = omega*(l - l_int) - (Ieta - Ieta_int);
00260     const std::complex<double> log_norm = (!is_it_normalized) ? (2.0*log_Cl_eta_calc (l,eta)) : (0.0);
00261     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);
00262
00263     if (real (two_I_Pi_eps) < 0.1)
00264     {
00265         const std::complex<double> log_cut_constant = log_two_I_omega + log (expm1 (two_I_Pi_eps)) +
log_norm;
00266
00267         return log_cut_constant;
00268     }
00269     else
00270     {
00271         const std::complex<double> log_cut_constant = log_two_I_omega + log1p (-exp (-two_I_Pi_eps)) +
two_I_Pi_eps + log_norm;
00272
00273         return log_cut_constant;
00274     }
00275 }
00276
00277
00278
00279 // Cut constant log for continued fractions : H[omega] from H[-omega, not corrected] case.
00280 // -----
00281 // The continued fraction has no cut on the negative real axis, whereas H[-omega] has one.
00282 // Then, if one is in the bad quadrant of H[-omega], one has to take into account the cut directly.
00283 // One is in the bad quadrant of H[-omega] if Re[z] < 0.0 and sign(Im[z]) = omega.
00284 //
00285 // H[omega] = H[-omega, not corrected] - cut_constant.F .
00286 //
00287 // The cut constant is 2i.omega.norm.exp (-2.i.Pi.[l.omega + i.eta]), and one takes its log.
00288 // The returned imaginary part of the log is not necessarily in ]-Pi:Pi].
00289 // Norm is 1.0 for normalized wave functions, C(l,eta)^2 for unnormalized wave functions.
00290 //
00291 //
00292 // Variables:
00293 // -----
00294 // is_it_normalized : true if one wants normalized functions, i.e. the standard normalization,
00295 //                     false if one wants F -> F/C(l,eta) and H+/H-/G -> H+/H-/G.C(l,eta), to avoid
00296 //                     overflows for |eta| » 1 and |z| small.

```

```

00296 // omega : 1 or -1.
00297 // l : orbital angular momentum l.
00298 // eta : Sommerfeld parameter = Coulomb_constant.Z.(2.mu/hbar^2)/(2k).
00299 // Ieta : i.eta .
00300 // l_int,Ieta_int : closest integers to Re[l],Re[i.eta]
00301 // eps : (l.omega - l_int.omega) + (Ieta - Ieta_int)
00302 // two_I_Pi : 2.i.Pi .
00303 // log_norm : log[C(l,eta)^2] if is_it_normalized is false, 0.0 if it is true.
00304 // log_two_I_omega : log[2.i.omega] = log[2] + i.omega.Pi/2 .
00305 // log_cut_constant : returned result.
00306
00307 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)
00308 {
00309     const std::complex<double> Ieta(-imag (eta),real (eta));
00310     const double l_int = rint (real (l)), Ieta_int = rint (real (Ieta));
00311     const std::complex<double> eps = omega*(l - l_int) + (Ieta - Ieta_int);
00312     const std::complex<double> log_norm = (!is_it_normalized) ? (2.0*log_Cl_eta_calc (l,eta)) : (0.0);
00313     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;
00314
00315     return log_cut_constant;
00316 }
00317
00318 // Sin (chi) calculation
00319 // -----
00320 // If 2l is integer, 0.0 is returned as chi is zero.
00321 // If not, one calculates sin (chi) with chi = sigma(l,eta) - sigma(-l-1,eta) - (l+1/2).Pi .
00322 // One uses the stable formula sin (chi) = -(2l+1).C(l,eta).C(-l-1,eta).
00323 //
00324 // Variables
00325 // -----
00326 // l : orbital angular momentum l.
00327 // eta : Sommerfeld parameter.
00328 // sin_chi : sin (chi)
00329
00330 std::complex<double> sin_chi_calc (const std::complex<double> &l,const std::complex<double> &eta)
00331 {
00332     if (rint (real (2.0*l)) == 2.0*l) return 0.0;
00333
00334     const std::complex<double> sin_chi = -(2*l+1)*exp (log_Cl_eta_calc (l,eta) + log_Cl_eta_calc
(-l-1,eta));
00335
00336     return sin_chi;
00337 }
00338
00339
00340 // exp (i.omega.chi) calculation.
00341 // -----
00342 // One calculates exp (i.omega.chi), with chi = sigma(l,eta) - sigma(l,-eta) - (l+1/2).Pi .
00343 // If 2l is integer, 1.0 is returned as chi is zero.
00344 // If not, one first calculates sin (chi) with the previous routine.
00345 // 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.
00346 // 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)]],
00347 // with chi given by sigma(l,eta) - sigma(l,-eta) - (l+1/2).Pi .
00348 //
00349 // Variables
00350 // -----
00351 // omega : 1 or -1
00352 // l : orbital angular momentum l.
00353 // eta : Sommerfeld parameter = Coulomb_constant.Z.(2.mu/hbar^2)/(2k).
00354 // I_omega : i.omega
00355 // sin_chi : sin (chi)
00356 // chi : sigma(l,eta) - sigma(l,-eta) - (l+1/2).Pi .
00357 // cos_chi : sign[Re[cos (chi)]]*sqrt[1 - [sin(chi)]^2]
00358 // exp_I_omega_chi : exp[i.omega.chi], returned result.
00359
00360 std::complex<double> exp_I_omega_chi_calc (const int omega,const std::complex<double> &l,const
std::complex<double> &eta)
00361 {
00362     if (rint (real (2.0*l)) == 2.0*l) return 1.0;
00363
00364     const std::complex<double> I_omega(0,omega),sin_chi = sin_chi_calc (l,eta);
00365     const std::complex<double> chi = sigma_l_calc (l,eta) - sigma_l_calc (-l-1,eta) - (l+0.5)*M_PI;
00366
00367     if (abs (sin_chi) > 0.5)
00368     {
00369         const std::complex<double> exp_I_omega_chi = exp (I_omega*chi);
00370
00371         return exp_I_omega_chi;
00372     }
00373     else
00374     {
00375         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;

```

```

00376
00377     return exp_I_omega_chi;
00378 }
00379 }

```

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.](#)

```

00001 #include "cwfcomp.H"
00002
00003 // Calculation of h(omega) = H(omega)/H(omega) with a continued fraction.
00004 // -----
00005 // One calculates the ratio h = H'/H with the continued fraction of the associated hypergeometric
confluent function.
00006 // One uses Lentz's method.
00007 // One has : h = [b[0] + a[1]/b[1] + a[2]/b[2] + ... a[n]/b[n] + ...].i.omega/z with :
00008 // 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 .
00009 //
00010 // 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.
00011 // If l=0 and |z| <= 0.5, the direct integration is still done as it is stable.
00012 // If l+1+/-i.eta is negative integer, it has to be done otherwise it is too long even if |z| is not
exceedingly small.
00013 // One first considers Re[z] >= 0.
00014 // 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).
00015 // Then, one calculates H[omega],H[omega]' at the starting point, and one integrates numerically
H[omega] until z.
00016 // h(omega)(z) is then H[omega](z)/H[omega](z).
00017 // If Re[z] < 0, one calculates H[-omega],H[-omega]' with l, eta -> -eta, and z -> -z.
00018 // One uses the Coulomb wave functions class cwf_minus_eta_ptr defined with l and -eta.
00019 // The ratio h(omega,l,eta,z) is then equal to -h(-omega,l,-eta,-z).
00020 // 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
00021 // to increase in modulus. Then, one always integrates forward. Forward integration is enforced
putting
00022 // is_H_dir_int_naive to true. It is put to false again at the end of the calculation.
00023 //
00024 // Variables:
00025 // -----
00026 // z : variable of the Coulomb wave function.
00027 // omega : 1 for the outgoing wave function ratio H+/H+, -1 for the incoming wave function ratio
H-/H-.
00028 // large,small : 1E50,1E-50. They are used in the case of vanishing denominators or numerators.
00029 // 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).
00030 // a,c : 1 + l + i.omega.eta, i.omega.eta - l.
00031 // b0,Cn,Dn,an,bn,Delta_n : variables used in the Lentz method.
00032 // n,nml : index of a[n] and b[n], n-1
00033 // bn_plus_an_Dn : bn + an.Dn. Dn = 1/[bn + an.Dn] or 1E50 if bn + an.Dn is zero.
00034 // bn_plus_an_over_Cn : bn + an/Cn. Cn = bn + an/Dn or 1E-50 if bn + an/Dn is zero.
00035 // hn : value of the continuous fraction during the iteration process.
00036 // test : test of convergence of hn.
00037 // cwf : reference to *this if Re[z] >= 0, reference to cwf_minus_eta_ptr if Re[z] < 0
00038 // cwf_minus_eta_ptr is allocated first if it is zero, in the case of Re[z] < 0 .
00039 // It is used to integrate the Coulomb wave functions with l,eta for Re[z] >= 0 or l,-eta for
Re[z] < 0.
00040 // z00 : 2 + i.sign[Re[z]].(Im[z] + 2.sign[Im[z]]).
00041 // z01 : 0.6 + i.0.6.sign[Re[z]].sign[Im[z]].
00042 // abs_z : |z|
00043 // z_start,F_start,dF_start : starting point of the direct integration,
F(1,+/-eta,z_start),F'(1,+/-eta,z_start)
00044 // 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.
00045 // h : value of H(omega)/H(omega).
00046 // 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.

```



```

00047
00048 std::complex<double> Coulomb_wave_functions::continued_fraction_h (const std::complex<double> &z,const
int omega)
00049 {
00050     const double small = 1E-50,large = 1E50,abs_z = abs (z);
00051     const std::complex<double> I_omega(0.0,omega),two_I_omega(0.0,2.0*omega),I_omega_eta = I_omega*eta;
00052     const std::complex<double> a = I_omega_eta + 1 + 1.0,c = I_omega_eta - 1,z_minus_eta = z -
eta,two_z_minus_eta = 2.0*z_minus_eta;
00053
00054     std::complex<double> b0 = z_minus_eta,hn = (b0 != 0.0) ? (b0) : (1E-50), Cn = hn, Dn = 0.0;
00055     int n = 1;
00056     double test;
00057     do
00058     {
00059         const int nml = n-1;
00060         const std::complex<double> an = (a + nml)*(c + nml),bn = two_z_minus_eta +
n*two_I_omega,bn_plus_an_Dn = bn + an*Dn,bn_plus_an_over_Cn = bn + an/Cn;
00061
00062         Dn = (bn_plus_an_Dn != 0.0) ? (1.0/bn_plus_an_Dn) : (large);
00063         Cn = (bn_plus_an_over_Cn != 0.0) ? (bn_plus_an_over_Cn) : (small);
00064
00065         const std::complex<double> Delta_n = Dn*Cn;
00066         hn *= Delta_n;
00067         test = inf_norm (1.0 - Delta_n);
00068
00069         if ((n++ > 100000) && ((l == 0.0) || (abs_z > 0.5) || neg_int_omega_one ||
neg_int_omega_minus_one))
00070         {
00071             if ((real (z) < 0.0) && (cwf_minus_eta_ptr == 0)) cwf_minus_eta_ptr = new class
Coulomb_wave_functions (is_it_normalized,l,-eta);
00072             class Coulomb_wave_functions &cwf = (real (z) < 0.0) ? (*cwf_minus_eta_ptr) : (*this);
00073
00074             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)));
00075             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;
00076             std::complex<double> F_start,dF_start,H,dH;
00077             cwf.F_dF (z_start,F_start,dF_start);
00078             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;
00079             cwf.debut = debut_cwf, cwf.F_debut = F_debut_cwf, cwf.dF_debut = dF_debut_cwf;
00080             return (SIGN (real (z))*dH/H);
00081         }
00082     }
00083     while (test > 1E-15);
00084
00085     const std::complex<double> h = hn*I_omega/z;
00086     return h;
00087 }
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098 // Calculation of H(omega) and dH(omega)/dz (scaled) with asymptotic series
00099 // -----
00100 // H[omega](z) = exp[i.omega.[z - eta.log[2z] - 1.Pi/2 + sigma(l,eta)]] .S[omega](z) for Re[z] >= 0.
00101 // H[omega]'(z) = exp[i.omega.[z - eta.log[2z] - 1.Pi/2 + sigma(l,eta)]] .[S[omega]'(z) + i.omega.(1 -
eta/z).S[omega](z)] for Re[z] >= 0.
00102 //
00103 // S[omega](z) is the asymptotic series and S[omega]'(z) its derivative calculated in
asymptotic_series.
00104 // If they did not converge, one leaves the routine.
00105 //
00106 // The negative cut is taken into account if log [cut_constant_AS] is finite, i.e. cut_constant_AS is
not exactly zero :
00107 //
00108 // If Re[z] < 0.0 and omega.Im[z] < 0.0 :
00109 // -----
00110 // H[omega] = H[omega][ASd] + cut_constant_AS_plus.H[-omega][ASd].
00111 // H[omega][ASd] is given by directly by the asymptotic series.
00112 // 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.
00113 //
00114 // 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]]] .
00115 //
00116 // In the case of overflows or underflows, one uses logs.
00117 //
00118 //
00119 // Variables :
00120 // -----

```

```

00121 // omega : 1 if one calculates H+(z) and H' (z), -1 if one calculates H-(z) and H' (z).
00122 // one_over_z : 1/z.
00123 // z : variable of the Coulomb wave function.
00124 // 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]]).
00125 // is_it_successful : true if the asymptotic expansions converged, false if not
00126 // sum,dsum : {S[omega](z),S[-omega](z)} and {S[omega]'(z),S[-omega]'(z)}
00127 // I_omega,two_I_omega : i.omega, 2i.omega.
00128 // I_omega_one_minus_eta_over_z : i.omega.(1 - eta/z).
00129 // phase_shift : i.omega.[-1.Pi/2 + sigma(1,eta)]
00130 // exp_phase_shift : exp (i.omega.(-1.Pi/2 + sigma(1,eta))).
00131 // log_cut_constant_AS : log_cut_constant_AS_plus if omega = 1, log_cut_constant_AS_minus if omega
=-1.
00132 // 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.
00133 // factor : exp (log_cut_constant_AS - 2.i.omega.(z - eta.log(2z)) - phase_shift).
00134
00135 void Coulomb_wave_functions::asymptotic_expansion_H_dH_scaled (const int omega,const
std::complex<double> &one_over_z,
00136 std::complex<double> &H_scaled,std::complex<double> &dH_scaled,bool
&is_it_successful)
00137 {
00138     std::complex<double> sum[2],dsum[2];
00139
00140     asymptotic_series (omega,one_over_z,sum,dsum,is_it_successful);
00141     if (!is_it_successful) return;
00142
00143     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);
00144
00145     const std::complex<double> phase_shift = I_omega*(sigma_1 - 1*M_PI_2),exp_phase_shift = exp
(phase_shift);
00146
00147     H_scaled = sum[0]*exp_phase_shift;
00148     dH_scaled = (dsum[0] + sum[0]*I_omega_one_minus_eta_over_z)*exp_phase_shift;
00149
00150     const std::complex<double> log_cut_constant_AS = (omega == 1) ? (log_cut_constant_AS_plus) :
(log_cut_constant_AS_minus);
00151
00152     if (one_over_z != 0.0)
00153     {
00154         const std::complex<double> z = 1.0/one_over_z;
00155
00156         if (isfinite (log_cut_constant_AS) && (real (z) < 0.0) && (SIGN (imag (z)) == -omega))
00157         {
00158             const std::complex<double> factor = exp (-two_I_omega*(z - eta*(M_LN2 + log (z))) - phase_shift
+ log_cut_constant_AS);
00159
00160             H_scaled += sum[1]*factor;
00161             dH_scaled += (dsum[1] - sum[1]*I_omega_one_minus_eta_over_z)*factor;
00162         }
00163     }
00164
00165     if (!is_it_normalized)
00166     {
00167         if ((Cl_eta == 0.0) || (!isfinite (Cl_eta)))
00168             H_scaled = exp (log_Cl_eta + log (H_scaled)),dH_scaled = exp (log_Cl_eta + log (dH_scaled));
00169         else
00170             H_scaled *= Cl_eta,dH_scaled *= Cl_eta;
00171     }
00172 }
00173
00174
00175
00176
00177
00178
00179 // Calculation of H[omega](z) and H[omega]'(z) by direct integration.
00180 // -----
00181 // To calculate H[omega](z) and H'[omega](z), one integrates numerically H[omega]"(z) = [1(1+1)/z^2 +
2.eta/z - 1].H[omega](z)
00182 // starting from debut,H_debut = H[omega](debut) and dH_debut = H'[omega](debut).
00183 // There is no branch cut problem as Re[z] >= 0.
00184 // The starting point comes for the regular function from the stored values debut, F_debut and
dF_debut.
00185 // If debut = 0, one puts debut = debut_omega = z/|2z| and calculates F(debut) and F'(debut) with
power series.
00186 // Then, the starting point {debut_omega,H[omega](debut),H'[omega](debut)} is calculated
00187 // from {debut,F(debut),F'(debut)} and the continued fraction h[omega](debut).
00188 // The first order expansions method is used if one is very close to the real axes of l,eta and z
(Re[z] > 0).
00189 // The step of the integration is (z - debut)/N_num, with N_num = [|z - debut|/min
(0.1,10.turning_point)] + 1.
00190 // The value of min (0.1,10/turning_point) gives a smaller step when turning_point increases,
00191 // as calculations become there more difficult as |H[omega]| typically varies faster in this case.
00192 // The intermediates points are called z_aft. They go from debut to z,
00193 // and (debut_omega,H_debut,dH_debut) is put to {z_aft,H[omega](z_aft),H'[omega](z_aft)} at each step.

```

```

00194 // If |H[omega]| increases along the path, the integration is stable.
00195 // If is_H_dir_int_naive is true, one has to integrate forward, as this integration is used to
    calculate the continued fraction.
00196 // If not, and if |H[omega]| decreases,
00197 // 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).
00198 // H'[omega](z_aft)/H[omega](z_aft) is given by the continued fraction formula.
00199 // One then obtains by direct integration C.H[omega](debut) and C.H'[omega](debut).
00200 // Knowing H_debut, one deduces H[omega](z) = 1/C and H'[omega](z) = f(z_aft).H[omega](z).
00201 // Increase or decrease is known using the Taylor expansion of H[omega] near debut_omega in z up to
    second order.
00202 //
00203 // If H(z_aft) is not finite, one stops the integration.
00204 //
00205 // Variables
00206 // -----
00207 // omega : 1 for H+,H+', -1 for H-,H-'.
00208 // z : variable of the Coulomb wave function.
00209 // H,dH : wave function H[omega] and derivative H'[omega] to calculate.
00210 // x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
00211 // debut_omega, H_debut,dH_debut : starting point or the integration. debut_omega = debut at the
    beginning.
00212 // ODE : reference to pointer ODE_ptr, which performs direct integration from one point to another.
00213 // step_abs : length of the integration step from z to debut. It is min (0.1,10/turning_point)
00214 // N_num : number of integrations to do from debut to z. It is |z - debut|/step_abs + 1.
00215 // step_num : std::complex step from debut to z. It is (z - debut)/N_num .
00216 // ll_plus_one,two_eta : 1(l+1), 2.eta. They are used in the Taylor expansion test.
00217 // 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.
00218 // 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).
00219 //                                     They are used in the Taylor expansion test.
00220 // h : continued fraction h[omega] = H[omega]'/H[omega] in z.
00221 // H_debut_not_normed,dH_debut_not_normed : unnormalized regular wave function and derivative in
    debut_omega,
00222 //                                     after integration from the starting point {z_aft, 1.0,
    H'[omega](z_aft)/H[omega](z_aft)}.
00223 //                                     Then, H[omega](z_aft) = H_debut/H_debut_not_normed and
    H'[omega](z_aft) = h[omega].H[omega](z_aft) .
00224
00225 void Coulomb_wave_functions::H_dH_direct_integration (const int omega,const std::complex<double>
    &z,std::complex<double> &H,std::complex<double> &dH)
00226 {
00227     const double x = real (z),y = imag (z),l_r = real (l),l_i = imag (l),eta_r = real (eta),eta_i = imag
    (eta);
00228
00229     if (debut == 0.0)
00230     {
00231         debut = 0.5*z/abs (z);
00232         F_dF_power_series (debut,F_debut,dF_debut);
00233     }
00234
00235     std::complex<double> debut_omega = debut,H_debut,dH_debut;
00236
00237     if ((y != 0.0) || (eta_i != 0.0) || (l_i != 0.0))
00238         && (std::abs (y) < sqrt_precision*std::min (1.0,x)) && (std::abs (eta_i) < sqrt_precision) &&
    (std::abs (l_i) < sqrt_precision)
00239         && (!neg_int_omega_one && !neg_int_omega_minus_one))
00240         first_order_expansions (omega,debut_omega,H_debut,dH_debut);
00241     else
00242         H_dH_with_F_dF_and_CF (omega,debut_omega,H_debut,dH_debut);
00243
00244     const class ODE_integration &ODE = *ODE_ptr;
00245     const double step_abs = std::min(0.1,10.0/turning_point);
00246     const unsigned int N_num = static_cast<unsigned int> ((abs (z-debut_omega)/step_abs) + 1);
00247     const std::complex<double> step_num = (z - debut_omega)/static_cast<double> (N_num),ll_plus_one =
    1*(1+1.0),two_eta = 2.0*eta;
00248
00249     for (unsigned int i = N_num-1 ; i <= N_num ; i--)
00250     {
00251         const std::complex<double> z_aft = z - i*step_num,one_over_debut = 1.0/debut,log_H_debut_der =
    dH_debut/H_debut;
00252         const std::complex<double> d2H_debut_over_H_debut = (ll_plus_one*one_over_debut +
    two_eta)*one_over_debut - 1.0;
00253
00254         if (is_H_dir_int_naive)
00255             ODE (debut_omega,H_debut,dH_debut,z_aft,H,dH);
00256         else if (abs (1.0 + step_num*(log_H_debut_der + 0.5*step_num*d2H_debut_over_H_debut)) < 1.0)
00257         {
00258             const std::complex<double> h = continued_fraction_h (z_aft,omega);
00259             std::complex<double> H_debut_not_normed,dH_debut_not_normed;
00260
00261             ODE (z_aft,1.0,h,debut_omega,H_debut_not_normed,dH_debut_not_normed);
00262             H = H_debut/H_debut_not_normed;
00263             dH = h*H;
00264         }
00265         else ODE (debut_omega,H_debut,dH_debut,z_aft,H,dH);

```

```

00266
00267     if (!isfinite (H) || !isfinite (dH)) std::cout<<"Numerical failure encountered in
H_dH_direct_integration."<<std::endl,exit (1);
00268
00269     debut_omega = z_aft,H_debut = H,dH_debut = dH;
00270 }
00271 }
00272
00273
00274
00275
00276
00277
00278
00279 // Calculation of H[omega](z), H'[omega](z) with the first order expansion method.
00280 // -----
00281 // When imaginary parts of l,eta,z are much smaller than their real parts but not all zero, with Re[z]
> 0,
00282 // 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.
00283 //
00284 // 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.
00285 //
00286 // Then, H[omega](z) = G(z) + i.omega.norm.F(z) and H'[omega](z) = G'(z) + i.omega.norm.F'(z),
00287 // with norm = 1 if the wave functions are normalized and C(l,eta)^2 if not.
00288 // One uses logs if norm underflows or overflows.
00289 //
00290 // Variables:
00291 // -----
00292 // omega : 1 if one calculates H+(z),H+'(z), -1 if one calculates H-(z),H-'(z).
00293 // z : variable of the Coulomb wave function.
00294 // H,dH : wave function H[omega] and derivative H'[omega] to calculate.
00295 // F,dF,G,dG : regular and irregular wave functions and derivatives in z, calculated with first order
expansions.
00296 // I_omega : i.omega
00297 // norm_functions,log_norm : 1 if the wave functions are normalized and C(l,eta)^2 if not, its log.
00298
00299
00300 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)
00301 {
00302     std::complex<double> F,dF,G,dG;
00303     first_order_expansions (true,z,F,dF);
00304     first_order_expansions (false,z,G,dG);
00305
00306     const std::complex<double> I_omega(0,omega),norm_functions = (!is_it_normalized) ? (Cl_eta*Cl_eta) :
(1.0);
00307
00308     if ((norm_functions == 0.0) || (!isfinite (norm_functions)))
00309     {
00310         const std::complex<double> log_norm = (!is_it_normalized) ? (2.0*log_Cl_eta) : (0.0);
00311
00312         H = G + I_omega*exp (log (F) + log_norm);
00313         dH = dG + I_omega*exp (log (dF) + log_norm);
00314     }
00315     else
00316     {
00317         H = G + I_omega*norm_functions*F;
00318         dH = dG + I_omega*norm_functions*dF;
00319     }
00320 }
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334 // Calculation of H(omega) and H(omega)' with F(z), F'(z) and continued fractions.
00335 // -----
00336 // 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
00337 // linearly independent of F.
00338 // 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].
00339 //
00340 // If not, one calculates h[sign(Im[z])](z), and one has f = F'(z)/F(z).
00341 // One chooses sign(Im[z]), as h converges fastest for z in this region.
00342 // If |f - h[sign(Im[z])]|oo >= 1, F and H[sign] are numerically linearly independent so the continued
fraction is meaningful.

```

```

00343 // Then,  $h[-\text{sign}(\text{Im}[z])](z)$  is not needed and is put to f, the worst value it can have. If not,
//  $h[-\text{sign}(\text{Im}[z])](z)$  is needed and calculated.
00344 // Then,  $h[\omega]$  and  $h[-\omega]$  take their values from  $h[\text{sign}(\text{Im}[z])]$  and  $h[-\text{sign}(\text{Im}[z])]$ .
00345 //
00346 // If  $|f - h[\omega]|_{\infty} > |f - h[-\omega]|_{\infty}$ , one uses the continued fraction  $h[\omega]$ .
00347 // If  $\text{Re}[z] > 0$  or  $\text{sign}(\text{Im}[z]) = \omega$  (good quadrants),  $H[\omega] = 1/F/(f - h[\omega])$  and  $H'[\omega] =$ 
 $h[\omega].H[\omega]$ .
00348 // If not, one has to take the branch cut into account, as one is in the bad quadrant of  $H[\omega]$  :
00349 //  $H[\omega] = 1/F/(f - h[\omega]) - \text{cut\_constant}.F$  and  $H'[\omega] = h[\omega].H[\omega] - \text{cut\_constant}.F'$ 
.
00350 //  $\text{cut\_constant}$  is  $\text{cut\_constant\_CFa\_plus}$  if  $\omega = 1$ , and  $\text{cut\_constant\_CFa\_minus}$  if  $\omega = -1$ .
00351 // If  $\log[\text{cut\_constant}]$  is not finite, it means that  $\text{cut\_constant}$  is exactly zero if it is defined,
so that there is no branch cut to consider.
00352 //
00353 // If  $|f - h[\omega]|_{\infty} < |f - h[-\omega]|_{\infty}$ , one uses the continued fraction  $h[-\omega]$ ,
00354 // calculating  $H[\omega]$  and  $H'[\omega]$  using  $H[\omega] = H[-\omega] + \text{constant}.F$  .
00355 //  $\text{constant}$  is  $2.i.\omega.\text{norm}$  if  $\text{Re}[z] > 0$  or  $\text{sign}(\text{Im}[z]) = -\omega$  (good quadrants), and  $\text{cut\_constant}$ 
if not.
00356 //  $\text{cut\_constant}$  is  $\text{cut\_constant\_CFb\_plus}$  if  $\omega = 1$ ,  $\text{cut\_constant\_CFb\_minus}$  if  $\omega = -1$ .
00357 //  $\text{norm}$  is 1 if  $\text{is\_it\_normalized}$  is true,  $C(1,\eta)^2$  if not.
00358 //
00359 // If  $\text{cut\_constant}$  underflows or overflows, one uses logs of  $F, F'$  and  $\log\_cut\_constant$  for the
calculation.
00360 //
00361 // Variables:
00362 // -----
00363 //  $\omega$  : 1 if one calculates  $H(z)$  and  $H'(z)$ , -1 if one calculates  $H(-z)$  and  $H'(-z)$ .
00364 //  $z$  : variable of the Coulomb wave function.
00365 //  $F, dF$ : regular Coulomb wave function and derivative.
00366 //  $H, dH$  :  $H(z)$  and  $H'(z)$  if  $\omega=1$ ,  $H(-z)$  and  $H'(-z)$  if  $\omega=-1$ .
00367 //  $x, y$  :  $\text{Re}[z], \text{Im}[z]$ .
00368 //  $f, \text{two\_I\_}\omega$  : ratio  $F'(z)/F(z)$ ,  $2.i.\omega$  .
00369 //  $h\_sign, h\_minus\_sign$  : continuous fractions  $h[-\text{SIGN}(y)]$  and  $h[\text{SIGN}(y)]$ . See before for their
calculations
00370 //  $h\_omega, h\_minus\_omega$  : continuous fraction  $h[\omega]$  and  $h[-\omega]$ .
00371 //  $\text{cut\_constant\_CFa}, \log\_cut\_constant\_CFa$  : cut constant for  $H[\omega]$ , so in its bad quadrant
 $H[\omega](z) = 1/F/(f - h[\omega]) - \text{cut\_constant}.F$  and its log.
00372 //  $\text{cut\_constant}$  is  $\text{cut\_constant\_CFa\_plus}$  if  $\omega = 1$ ,
 $\text{cut\_constant\_CFa\_minus}$  if  $\omega = -1$ .
00373 // If the log is not finite,  $\text{cut\_constant}$  has to be strictly
zero if it is defined so there is no branch cut to consider.
00374 //  $H\_minus\_omega, dH\_minus\_omega$  :  $H(-z)$  and  $H'(-z)$  if  $\omega=1$ ,  $H(z)$  and  $H'(z)$  if  $\omega=-1$  in good
quadrants.
00375 // In bad quadrants, they are not equal to the previous functions, but
combined with branch cut formulas, one calculates them.
00376 //  $\text{cut\_constant\_CFb}, \text{constant}, \text{norm\_functions}$  :  $\text{cut\_constant}$  is  $\text{cut\_constant\_CFb\_plus}$  if  $\omega = 1$ ,
 $\text{cut\_constant\_CFb\_minus}$  if  $\omega = -1$ .
00377 //  $\text{constant}$  is  $2.i.\omega.\text{norm\_functions}$  in good quadrants,
 $\text{cut\_constant}$  in bad quadrants.
00378 //  $\text{norm\_functions}$  is 1 for normalized functions,  $C(1,\eta)^2$ 
for unnormalized wave functions.
00379 // One has :  $H[-\omega](z) = 1/F/(f - h[-\omega]) +$ 
 $\text{constant}.F$  in all quadrants.
00380 //  $\log\_cut\_constant\_CFb, \log\_constant, \log\_norm$  : logs of previous values.
00381 //  $\log\_two\_I\_omega$  :  $\log[2.i.\omega] = \log[2] + i.\omega.\text{Pi}/2$  .
00382
00383 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)
00384 {
00385     std::complex<double> F, dF;
00386     F_dF (z, F, dF);
00387
00388     const double x = real (z), y = imag (z);
00389     const std::complex<double> f = dF/F, two_I_omega(0.0, 2.0*omega);
00390
00391     if ((neg_int_omega_one && (omega == -1)) || ((neg_int_omega_minus_one && (omega == 1))))
00392     {
00393         const std::complex<double> h_omega = continued_fraction_h (z, omega);
00394          $H = 1.0/(F*(f - h\_omega)), dH = H*h\_omega;$ 
00395     }
00396     else if (neg_int_omega_one || neg_int_omega_minus_one)
00397     {
00398          $H = F, dH = dF;$ 
00399     }
00400     {
00401         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);
00402         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);
00403         if (inf_norm (f - h_omega) > inf_norm (f - h_minus_omega))
00404         {
00405              $H = 1.0/(F*(f - h\_omega)), dH = H*h\_omega;$ 
00406         }
00407         const std::complex<double> log_cut_constant_CFa = (omega == 1) ? (log_cut_constant_CFa_plus) :
(log_cut_constant_CFa_minus);
00408
00409         if ((isfinite (log_cut_constant_CFa)) && (x < 0.0) && (SIGN (y) == -omega))

```

```

00410     {
00411         const std::complex<double> cut_constant_CFa = (omega == 1) ? (cut_constant_CFa_plus) :
(cut_constant_CFa_minus);
00412
00413         if ((cut_constant_CFa == 0.0) || (!isfinite (cut_constant_CFa)))
00414             H -= exp (log_cut_constant_CFa + log (F)), dH -= exp (log_cut_constant_CFa + log (dF));
00415         else
00416             H -= cut_constant_CFa*F, dH -= cut_constant_CFa*dF;
00417     }
00418 }
00419 else
00420 {
00421     const std::complex<double> H_minus_omega = 1.0/(F*(f - h_minus_omega)), dH_minus_omega =
H_minus_omega*h_minus_omega;
00422     const std::complex<double> norm_functions = (!is_it_normalized) ? (Cl_eta*Cl_eta) : (1.0);
00423     const std::complex<double> cut_constant_Cfb = (omega == 1) ? (cut_constant_Cfb_plus) :
(cut_constant_Cfb_minus);
00424     const std::complex<double> constant = ((x < 0.0) && (SIGN (y) == omega)) ? (cut_constant_Cfb) :
(two_I_omega*norm_functions);
00425
00426     if ((constant == 0.0) || (!isfinite (constant)))
00427     {
00428         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);
00429         const std::complex<double> log_cut_constant_Cfb = (omega == 1) ? (log_cut_constant_Cfb_plus) :
(log_cut_constant_Cfb_minus);
00430         const std::complex<double> log_constant = ((x < 0.0) && (SIGN (y) == omega)) ?
(log_cut_constant_Cfb) : (log_two_I_omega + log_norm);
00431
00432         H = exp (log_constant + log (F)) + H_minus_omega, dH = exp (log_constant + log (dF)) +
dH_minus_omega;
00433     }
00434     else H = constant*F + H_minus_omega, dH = constant*dF + dH_minus_omega;
00435 }
00436 }
00437 }
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450 // Calculation of H[omega], H'[omega] for std::complex l with the expansion formula.
00451 // -----
00452 // When 2l is non-integer, one can expand H[omega] with F[l, eta, z] and F[-l-1, eta, z].
00453 // 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,
00454 // 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.
00455 // chi is sigma(l, eta) - sigma(-l-1, eta) - (l+0.5).Pi, and Fp is F(-l-1, eta).
00456 // Fp is calculated using a class Coulomb_wave_functions with parameters -l-1 and eta.
00457 // To avoid numerical imprecisions, sin (chi) is calculated with the stable formula
-(2l+1).C(l, eta).C(-l-1, eta).
00458 // The validity of this expansion is checked with the wronskian of F and Fp, which must be correct up
to precision :
00459 // F'.Fp - Fp'.F = sin (chi) if wave functions are normalized, Fp'.F - F'.Fp = 2l + 1 if not.
00460 // If the wronskians are numerically correct, one does the calculation and is_it_successful is put to
true.
00461 // If not, one puts is_it_successful to false and quits the routine.
00462 // If C(l, eta)^2 underflows or overflows, one uses logs of F, F' and C(l, eta)^2 for the calculation.
00463 //
00464 // Variables
00465 // -----
00466 // omega : 1 if one calculates H+(z) and H+(z), -1 if one calculates H-(z) and H-(z).
00467 // z : variable of the Coulomb wave function.
00468 // F, dF: regular Coulomb wave function and derivative in l, eta and z.
00469 // H, dH : H+(z) and H+(z) if omega=1, H-(z) and H-(z) if omega=-1.
00470 // 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.
00471 // Fp, dFp : F(-l-1, eta, z), F'(-l-1, eta, z).
00472 // exp_I_omega_chi : exp[i.omega.chi]
00473 // one_over_2lpl : 1/(2l + 1)
00474 // Cl_eta_2, exp_I_omega_chi_over_sin_chi : C(l, eta)^2, exp[i.omega.chi]/sin (chi)
00475 // F_Cl_eta_2, dF_Cl_eta_2 : F(z).C(l, eta)^2, F'(z).C(l, eta)^2.
00476
00477 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)
00478 {
00479     if (cwf_lp_ptr == 0) cwf_lp_ptr = new class Coulomb_wave_functions (is_it_normalized, -l-1, eta);
00480
00481     std::complex<double> F, dF, Fp, dFp;

```

```

00482 F_dF (z,F,dF);
00483 cwf_lp_ptr->F_dF (z,Fp,dFp);
00484
00485 const std::complex<double> exp_I_omega_chi = (omega == 1) ? (exp_I_chi) : (exp_minus_I_chi);
00486
00487 if (is_it_normalized)
00488 {
00489     if (inf_norm ((dFp*F - dF*Fp)*one_over_sin_chi - 1.0) < precision)
00490     {
00491         H = (exp_I_omega_chi*F - Fp)*one_over_sin_chi;
00492         dH = (exp_I_omega_chi*dF - dFp)*one_over_sin_chi;
00493     }
00494     else {is_it_successful = false; return;}
00495 }
00496 else
00497 {
00498     const std::complex<double> one_over_2lp1 = 1.0/(2*1+1);
00499
00500     if (inf_norm ((dF*Fp - dFp*F)*one_over_2lp1 - 1.0) < precision)
00501     {
00502         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;
00503         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);
00504         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);
00505
00506         H = exp_I_omega_chi_over_sin_chi*F_Cl_eta_2 + Fp*one_over_2lp1;
00507         dH = exp_I_omega_chi_over_sin_chi*dF_Cl_eta_2 + dFp*one_over_2lp1;
00508     }
00509     else {is_it_successful = false; return;}
00510 }
00511
00512 is_it_successful = true;
00513 }
00514
00515
00516
00517
00518
00519
00520
00521 // Calculation of F and F' by power series.
00522 // -----
00523 // It is used only when |z| <= 0.5, to avoid numerical inaccuracies.
00524 //
00525 // F(z) = norm.z^(l+1).\sum a[n], n in [0:+oo[, where :
00526 // a[0] = 1.0.
00527 // a[1] = z.eta/(l+1).
00528 // a[n] = (2.z.eta.a[n-1] - a[n-2].(z^2))/(n.(n+2l+1)),n >= 2.
00529 //
00530 // The z = 0 case is treated first. It is defined only for Re[l] > 0 or l = 0. The program aborts for
other cases.
00531 // Norm is C(l,eta) if one uses normalized wave functions, 1.0 if not.
00532 // So, one multiplies by C(l,eta) at the end if one uses normalized functions.
00533 // 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.
00534 //
00535 // Variables:
00536 // -----
00537 // z : variable of the Coulomb wave function.
00538 // F,dF : regular wave function and derivative
00539 // z_square,z_two_eta,z_pow_l_plus_one : z^2, 2.z.eta, z^(l+1).
00540 // n : index of the power series term. It begins at two.
00541 // an,an_minus_one,an_minus_two : a[n], a[n-1], a[n-2].
00542 // The test of convergence is |(n+1-l).a[n-2]|oo + |(n+1).a[n-1]|oo, as one of the two can be zero
even before convergence.
00543
00544 void Coulomb_wave_functions::F_dF_power_series (const std::complex<double> &z,std::complex<double>
&F,std::complex<double> &dF)
00545 {
00546     if (z == 0.0)
00547     {
00548         if (l == 0) F = 0.0,dF = (is_it_normalized) ? (Cl_eta) : (1.0);
00549         else if (real (l) > 0) F = dF = 0.0;
00550         else std::cout<<"F(z=0) and/or F'(z=0) are undefined."<<std::endl, abort ();
00551     }
00552     else
00553     {
00554         const std::complex<double> z_square = z*z,z_two_eta = 2.0*eta*z;
00555
00556         int n = 2;
00557         std::complex<double> an_minus_two = 1.0,an_minus_one = z*eta/(1+1.0);
00558
00559         F = an_minus_two + an_minus_one;
00560         dF = (1+1.0)*an_minus_two + (1+2.0)*an_minus_one;
00561

```

```

00562     while (inf_norm (an_minus_two*(n+1-1.0)) + inf_norm (an_minus_one*(n+1)) > precision)
00563     {
00564         const std::complex<double> an = (z_two_eta*an_minus_one - an_minus_two*z_square)/(n*(n + 1 + 1 +
1.0));
00565
00566         F += an;
00567         dF += an*(n + 1 + 1.0);
00568
00569         n++;
00570         an_minus_two = an_minus_one;
00571         an_minus_one = an;
00572     }
00573
00574     const std::complex<double> z_pow_l_plus_one = pow (z,1+1.0);
00575     F *= z_pow_l_plus_one;
00576     dF *= z_pow_l_plus_one/z;
00577
00578     if (is_it_normalized)
00579     {
00580         if ((Cl_eta == 0.0) || (!isfinite (Cl_eta))) F = exp (log_Cl_eta + log (F)), dF = exp (log_Cl_eta
+ log (dF));
00581         else F *= Cl_eta, dF *= Cl_eta;
00582     }
00583 }
00584 }
00585
00586
00587
00588
00589
00590
00591
00592 // Calculation of f = F'/F with a continued fraction.
00593 // -----
00594 // One calculates the ratio f = F'/F with the continued fraction of the associated hypergeometric
confluent function.
00595 // One uses Lentz's method.
00596 // One has : f = [b[0] + a[1]/b[1] + a[2]/b[2] + ... a[n]/b[n] + ...]/z with :
00597 // b[0] = 1 + 1 + i.omega.z, a[n] = -2.i.omega.[1 + 1 + i.omega.eta] + (n-1).[-2.i.omega.z], b[n] = 2l
+ 2 + 2.i.omega.z + n-1.
00598 // omega is 1 or -1, and theoretically the result is the same.
00599 // If they are not equal numerically, omega = sign[-Im [z]] gives usually the best result.
00600 // If 1+l+i.omega.eta is a negative integer, f[omega] is finite and must be used.
00601 //
00602 // Variables:
00603 // -----
00604 // z : variable of the Coulomb wave function.
00605 // omega : 1 or -1. Both values should be tried to test stability.
00606 // large,small : 1E50,1E-50. They are used in the case of vanishing denominators or numerators.
00607 // I_omega,a,b : i.omega, 1 + 1 + i.omega.eta, 2l + 2.
00608 // 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
00609 // b0,Cn,Dn,an,bn,Delta_n : variables used in the Lentz method.
00610 // n,nml : index of a[n] and b[n], n-1
00611 // bn_plus_an_Dn : bn + an.Dn. Dn = 1/[bn + an.Dn] or large if bn + an.Dn is zero.
00612 // bn_plus_an_over_Cn : bn + an/Cn. Cn = bn + an/Dn or small if bn + an/Dn is zero.
00613 // fn : value of the continuous fraction during the iteration process.
00614 // test : test of convergence of fn.
00615 // f : value of F'(z)/F(z).
00616
00617 std::complex<double> Coulomb_wave_functions::continued_fraction_f (const std::complex<double> &z,const
int omega)
00618 {
00619     const double small = 1E-50,large = 1E50;
00620
00621     const std::complex<double> I_omega(0.0,omega);
00622     const std::complex<double> a = I_omega*eta + 1 + 1.0,b = 2*1 + 2;
00623     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;
00624
00625     const std::complex<double> b0 = 1 + 1.0 + I_omega*z;
00626     std::complex<double> fn = (b0 != 0.0) ? (b0) : (small), Cn = fn, Dn = 0.0;
00627
00628     int n = 1;
00629     double test;
00630     do
00631     {
00632         const int nml = n-1;
00633         const std::complex<double> an = minus_two_I_omega_a_z + nml*minus_two_I_omega_z;
00634         const std::complex<double> bn = b_plus_two_I_omega_z + nml;
00635
00636         const std::complex<double> bn_plus_an_Dn = bn + an*Dn,bn_plus_an_over_Cn = bn + an/Cn;
00637
00638         Dn = (bn_plus_an_Dn != 0.0) ? (1.0/bn_plus_an_Dn) : (large);
00639         Cn = (bn_plus_an_over_Cn != 0.0) ? (bn_plus_an_over_Cn) : (small);
00640
00641         const std::complex<double> Delta_n = Dn*Cn;

```



```

00642     fn *= Delta_n;
00643     test = inf_norm (1.0 - Delta_n);
00644     n++;
00645 }
00646 while (test > 1E-15);
00647
00648 const std::complex<double> f = fn/z;
00649
00650 return f;
00651 }
00652
00653
00654
00655
00656
00657
00658
00659
00660 // Calculation of F(z) and F'(z) with asymptotic series
00661 // -----
00662 // F(z) = [H+(z) - H-(z)]/[2.i.norm].
00663 // F'(z) = [H+'(z) - H-'(z)]/[2.i.norm].
00664 // In this routine, Re[z] >= 0, so there is no branch cut problem.
00665 //
00666 // H+(z) = exp[i.(z - eta.log[2z] - 1.Pi/2 + sigma(1,eta))].S+(z) .
00667 // H-(z) = exp[-i.(z - eta.log[2z] - 1.Pi/2 + sigma(1,eta))].S-(z) .
00668 //
00669 // H+'(z) = exp[i.(z - eta.log[2z] - 1.Pi/2 + sigma(1,eta))].[S+'(z) + S+(z).i.(1 - eta/z)] .
00670 // H-'(z) = exp[-i.(z - eta.log[2z] - 1.Pi/2 + sigma(1,eta))].[S-'(z) - S-(z).i.(1 - eta/z)] .
00671 //
00672 //
00673 // S+ and S- and derivatives are calculated in asymptotic_series. If is_it_successful is true, the
00674 // series are meaningful. If not, one leaves the routine.
00675 // Norm is C(1,eta) if one uses normalized wave functions, 1.0 if not.
00676 // If there is overflow or underflow for C(1,eta) in this last case, one uses logs of F,F' and
00677 // C(1,eta) for the calculation.
00678 //
00679 // Variables :
00680 // -----
00681 // z : variable of the Coulomb wave function.
00682 // one_over_z : 1/z
00683 // F,dF : regular wave function and derivative to calculate.
00684 // is_it_successful : true if the calculation converged, i.e. the series are good up to precision and
00685 // the wronskian of H+,H- up to precision, false if not.
00686 // sum,dsum : asymptotic series. sum[0] = S+(z), sum[1] = S-(z), dsum[0] = S+'(z), dsum[1] = S-'(z).
00687 // I_one_minus_eta_over_z : i*(1 - eta/z).
00688 // exp_phase_shift_plus : exp (i*(z - eta.log(2z) - 1.Pi/2 + sigma(1,eta))).
00689 // exp_phase_shift_minus : exp (-i*(z - eta.log(2z) - 1.Pi/2 + sigma(1,eta))).
00690 // H_plus,dH_plus,H_minus,dH_minus : exp (+/-i*(z - eta.log(2z) - 1.Pi/2 + sigma(1,eta))).S(+/-)(z)
00691 // and derivatives.
00692
00693 void Coulomb_wave_functions::asymptotic_expansion_F_dF (const std::complex<double>
00694 &z,std::complex<double> &F,std::complex<double> &dF,bool &is_it_successful)
00695 {
00696     std::complex<double> sum[2],dsum[2];
00697
00698     const std::complex<double> one_over_z = 1.0/z;
00699
00700     asymptotic_series (1,one_over_z,sum,dsum,is_it_successful);
00701     if (!is_it_successful) return;
00702
00703     const std::complex<double> I(0,1),one_over_two_I(0.0,-0.5),I_one_minus_eta_over_z = I*(1.0 -
00704 eta*one_over_z);
00705
00706     const std::complex<double> exp_phase_shift_plus = exp (I*(z - eta*(M_LN2 + log (z)) - 1*M_PI_2 +
00707 sigma_1));
00708     const std::complex<double> exp_phase_shift_minus = 1.0/exp_phase_shift_plus;
00709
00710     const std::complex<double> H_plus = sum[0]*exp_phase_shift_plus,dH_plus = (dsum[0] +
00711 sum[0]*I_one_minus_eta_over_z)*exp_phase_shift_plus;
00712     const std::complex<double> H_minus = sum[1]*exp_phase_shift_minus,dH_minus = (dsum[1] -
00713 sum[1]*I_one_minus_eta_over_z)*exp_phase_shift_minus;
00714
00715     F = (H_plus - H_minus)*one_over_two_I;
00716     dF = (dH_plus - dH_minus)*one_over_two_I;
00717
00718     if (!is_it_normalized)
00719     {
00720         if ((Cl_eta == 0.0) || (!isfinite (Cl_eta))) F = exp (log (F) - log_Cl_eta),dF = exp (log (dF) -
00721 log_Cl_eta);
00722         else F /= Cl_eta,dF /= Cl_eta;
00723     }
00724 }
00725
00726
00727 }
00728
00729

```

```

00719
00720
00721
00722
00723
00724 // Calculation of F(z) and F'(z) by direct integration.
00725 // -----
00726 // To calculate F(z) and F'(z), one integrates numerically  $F''(z) = [1(1+1)/z^2 + 2.\text{eta}/z - 1].F(z)$ 
00727 // starting from debut,  $F_{\text{debut}} = F(\text{debut})$  and  $dF_{\text{debut}} = F'(\text{debut})$ .
00728 // One always has  $\text{Re}[z] \geq 0.0$ , so there is no branch cut problem.
00729 // If  $z = \text{debut}$ , the previous values are returned.
00730 // The starting point come from the stored values debut,  $F_{\text{debut}}$  and  $dF_{\text{debut}}$ .
00731 // If debut = 0, one puts  $\text{debut} = z/|2z|$  and calculates  $F(\text{debut})$  and  $F'(\text{debut})$  with power series.
00732 // The step of the integration is  $(z - \text{debut})/N_{\text{num}}$ , with  $N_{\text{num}} = [|z - \text{debut}|/\min$ 
     $(0.1, 10.\text{turning\_point})] + 1$ .
00733 // The value of  $\min(0.1, 10/\text{turning\_point})$  gives a smaller step when  $\text{turning\_point}$  increases,
00734 // as calculations become there more difficult as  $|F|$  typically varies faster in this case.
00735 // The intermediates points are called  $z_{\text{aft}}$ . They go from debut to  $z$ , and  $(\text{debut}, F_{\text{debut}}, dF_{\text{debut}})$  is
    put to  $\{z_{\text{aft}}, F(z_{\text{aft}}), F'(z_{\text{aft}})\}$  at each step.
00736 // If  $|F|$  increases along the path, the integration is stable.
00737 // If it decreases, and if  $z_{\text{aft}}$  decreases in modulus or one does not integrate with constant argument
    (i.e.  $\theta$  constant in  $z = |z|. \exp[i.\theta]$ )
00738 // for  $\text{Re}[1] > -1$ , one reintegrates  $F(z)$  from  $\text{debut} = z/|2z|$ , as integration is usually stable at
    constant argument for  $\text{Re}[1] > -1$ .
00739 // Increase or decrease is known using the Taylor expansion of  $F$  near debut in  $z$  up to second order.
00740 // If one integrates with constant argument and  $|F|$  decreases,
00741 // one integrates backwards from  $z_{\text{aft}}$  to debut with the knowledge of  $f(z_{\text{aft}}) = F'(z_{\text{aft}})/F(z_{\text{aft}})$ .
00742 //  $F'(z_{\text{aft}})/F(z_{\text{aft}})$  is given by the continued fraction formula.
00743 // One then obtains by direct integration  $C.F(\text{debut})$  and  $C.F'(\text{debut})$ .
00744 // Knowing  $F_{\text{debut}}$ , one deduces  $F(z) = 1/C$  and  $F'(z) = f(z_{\text{aft}}).F(z)$ .
00745 // If  $1+l+i.\text{omega}.\text{eta}$  is a negative integer,  $f[\text{omega}]$  is finite and is used.
00746 // Otherwise,  $f(z_{\text{aft}})$  is calculated with  $\text{omega} = 1$  and  $-1$ . If they are equal up to precision,
     $f(\text{omega})$  is correct and used.
00747 //  $\text{omega}$  is chosen so  $\text{Re}[-2.i.\text{omega}.z] < 0$ ,
00748 // for which the anomalous convergence phenomenon of Gautschi of  $f$  is the smallest (W. Gautschi, Math.
    Comp. Vol. 31 p.994).
00749 // If not, but  $|\text{norm}.F| < 0.1$ , one still has to use  $f[\text{omega}]$ , as it is probably correct as  $F$  is the
    minimal solution, and also one has no other way to calculate  $F$ .
00750 // If  $|\text{norm}.F| > 0.1$  in this case, one stops the procedure and  $F$  will be calculated from  $H+$  and  $H-$ ,
    given by direct integration and continued fraction formulae.
00751 // In this case,  $\text{is\_it\_successful}$  is put to false, and otherwise the integration worked and it is put
    to true.
00752 // Norm is 1.0 if one uses normalized wave functions,  $C(1,\text{eta})$  if not.
00753 //
00754 // If  $F(z_{\text{aft}})$  is not finite, one stops the integration and  $\text{is\_it\_successful}$  is put to false.
00755 //
00756 // Variables
00757 // -----
00758 //  $z$  : variable of the Coulomb wave function.
00759 //  $F, dF$  : regular wave function and derivative to calculate.
00760 //  $\text{is\_it\_successful}$  : false is the calculation is unstable,
00761 // i.e.  $|F| > 0.1$  decreasing on the integration path and  $f(\text{omega})$  is not equal to  $f(-\text{omega})$  up to
    precision, true if not.
00762 // ODE : reference to pointer ODE_ptr, which performs direct integration from one point to another.
00763 //  $\text{step\_abs}$  : length of the integration step from  $z$  to debut. It is  $\min(0.1, 10/\text{turning\_point})$ 
00764 //  $N_{\text{num}}$  : number of integrations to do from debut to  $z$ . It is  $|z - \text{debut}|/\text{step\_abs} + 1$ .
00765 //  $\text{step\_num}$  :  $\text{std::complex}$  step from debut to  $z$ . It is  $(z - \text{debut})/N_{\text{num}}$ .
00766 //  $1l\_plus\_one, two\_eta$  :  $1(1+1)$ ,  $2.\text{eta}$ . They are used in the Taylor expansion test.
00767 //  $z_{\text{aft}}$  : next point in which  $F$  is calculated by direct integration. It is  $z - i.\text{step\_num}$ , with  $i$ 
    from  $N_{\text{num}}-1$  to 0.
00768 //  $\text{one\_over\_debut}, \log\_F_{\text{debut\_der}}, d2F_{\text{debut\_over\_F\_debut}}$  :  $1/\text{debut}$ ,  $F'(\text{debut})/F(\text{debut})$ ,
     $F''(\text{debut})/F(\text{debut})$ . They are used in the Taylor expansion test.
00769 //  $\text{ratio} = \text{debut}/z$ . It is used to know if one integrates with constant argument or if  $|z_{\text{aft}}|$ 
    increases.
00770 // If not and  $|F|$  increases and  $\text{Re}[1] > -1$ , one reintegrates from  $z/|2z|$ .
00771 //  $f_{\text{omega}}, f_{\text{minus\_omega}}$  : continued fractions  $F'/F$  with  $\text{omega} = \text{sign}(-\text{Im}[z_{\text{aft}}])$  and  $-\text{omega}$ .
00772 //  $fp, fm$  : continued fractions  $F'/F$  with  $\text{omega} = 1$  or  $-1$  used in the case of  $1+l \pm i.\text{eta}$  negative
    integer, as continued fractions are finite in this case.
00773 //  $\text{norm\_functions}$  :  $C(1,\text{eta})^2$  for unnormalized functions, 1 for normalized functions.
00774 //  $F_{\text{debut\_not\_normed}}, dF_{\text{debut\_not\_normed}}$  : unnormed regular wave function and derivative in debut,
    after integration from the starting point  $\{z_{\text{aft}}, 1.0,$ 
00775  $F'(z_{\text{aft}})/F(z_{\text{aft}})\}$ .
00776 // Then,  $F(z_{\text{aft}}) = F_{\text{debut}}/F_{\text{debut\_not\_normed}}$  and  $F'(z_{\text{aft}})$ 
     $= f[\text{omega}].F$ .
00777
00778 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)
00779 {
00780     if (z == debut) {F = F_debut, dF = dF_debut, is_it_successful = true; return;}
00781     if (debut == 0.0) debut = 0.5*z/abs(z), F_dF_power_series(debut, F_debut, dF_debut);
00782
00783     const class ODE_integration &ODE = *ODE_ptr;
00784     const double step_abs = std::min(0.1, 10.0/turning_point);
00785     const unsigned int N_num = static_cast<unsigned int>(rint(abs(z-debut)/step_abs) + 1);
00786     const std::complex<double> step_num = (z - debut)/static_cast<double>(N_num), 1l_plus_one =
    1*(1+1.0), two_eta = 2.0*eta;
00787     const std::complex<double> norm_functions = (!is_it_normalized) ? (Cl_eta*Cl_eta) : (1.0);

```

```

00788
00789     for (unsigned int i = N_num-1 ; i <= N_num ; i--)
00790     {
00791         const std::complex<double> z_aft = z - i*step_num, one_over_debut = 1.0/debut, log_F_debut_der =
dF_debut/F_debut;
00792         const std::complex<double> d2F_debut_over_F_debut = (1l_plus_one*one_over_debut +
two_eta)*one_over_debut - 1.0;
00793
00794         if (abs (1.0 + step_num*(log_F_debut_der + 0.5*step_num*d2F_debut_over_F_debut)) < 1.0)
00795         {
00796             const std::complex<double> ratio = debut/z;
00797             if ((real (1) > -1.0) && ((std::abs (imag (ratio)) > precision) || (real (ratio) > 1.0)))
00798             {debut = 0.0, F_dF_direct_integration (z,F,dF,is_it_successful); return;}
00799
00800             std::complex<double> F_debut_not_normed,dF_debut_not_normed;
00801
00802             if (neg_int_omega_one)
00803             {
00804                 const std::complex<double> fp = continued_fraction_f (z_aft,1);
00805                 ODE (z_aft,1.0,fp,debut,F_debut_not_normed,dF_debut_not_normed);
00806                 F = F_debut/F_debut_not_normed;
00807                 dF = fp*F;
00808             }
00809             else if (neg_int_omega_minus_one)
00810             {
00811                 const std::complex<double> fm = continued_fraction_f (z_aft,-1);
00812                 ODE (z_aft,1.0,fm,debut,F_debut_not_normed,dF_debut_not_normed);
00813                 F = F_debut/F_debut_not_normed;
00814                 dF = fm*F;
00815             }
00816             else
00817             {
00818                 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)));
00819
00820                 if ((abs (F*norm_functions) > 0.1) && (abs (f_minus_omega/f_omega - 1.0) > precision))
00821                 {is_it_successful = false; return;}
00822
00823                 ODE (z_aft,1.0,f_omega,debut,F_debut_not_normed,dF_debut_not_normed);
00824                 F = F_debut/F_debut_not_normed;
00825                 dF = f_omega*F;
00826             }
00827             }
00828             else ODE (debut,F_debut,dF_debut,z_aft,F,dF);
00829
00830             debut = z_aft,F_debut = F,dF_debut = dF;
00831
00832             if (!isfinite (F) || !isfinite (dF)) std::cout<<"Numerical failure encountered in
F_dF_direct_integration."<<std::endl,exit (1);
00833         }
00834         is_it_successful = true;
00835     }
00836
00837
00838
00839     // Regular wave function and derivative from symmetry relations.
00840     // -----
00841     // If |z| > 0.5 and Re[z] < 0, one calculates F(1,eta,z), F'(1,eta,z) from F(1,-eta,-z), F'(1,-eta,-z)
using the formulas :
00842     // F(1,eta,z) = -F(1,-eta,-z).exp[-Pi.(eta-i.1)], F'(1,eta,z) = F'(1,-eta,-z).exp[-Pi.(eta-i.1)] if
arg (z) > 0 and is_it_normalized is true,
00843     // F(1,eta,z) = -F(1,-eta,-z).exp[-Pi.(eta+i.1)], F'(1,eta,z) = F'(1,-eta,-z).exp[-Pi.(eta+i.1)] if
arg (z) <= 0 and is_it_normalized is true,
00844     // F(1,eta,z) = -F(1,-eta,-z).exp[i.Pi.1], F'(1,eta,z) = F'(1,-eta,-z).exp[i.Pi.1] if arg (z) > 0
and is_it_normalized is false,
00845     // F(1,eta,z) = -F(1,-eta,-z).exp[-i.Pi.1], F'(1,eta,z) = F'(1,-eta,-z).exp[-i.Pi.1] if arg (z) <= 0
and is_it_normalized is false.
00846     //
00847     // F(1,-eta,-z) is calculated using the class cwf_minus_eta_ptr defined with (1,-eta).
00848     // The debut point of the class cwf_minus_eta_ptr is initialized with {debut,F_debut,dF_debut} and
previous relations
00849     // if cwf_minus_eta_ptr->debut and -debut are different and debut non zero.
00850     //
00851     // If the normalization constant underflows or overflows, one uses logs.
00852     //
00853     // Variables
00854     // -----
00855     // z : variable of the Coulomb wave function.
00856     // F,dF : regular wave function and derivative to calculate.
00857     // arg_debut, arg_z : argument angles of debut and z.
00858     // sym_constant_debut, log_sym_constant_debut : constant C so F(1,-eta,-debut) = C.F(1,eta,debut), its
log.
00859     // sym_constant, log_sym_constant : constant C so F(1,eta,z) = C.F(1,-eta,-z), its log.
00860
00861     void Coulomb_wave_functions::F_dF_with_symmetry_relations (const std::complex<double>
&z, std::complex<double> &F, std::complex<double> &dF)
00862     {

```

```

00863     if (cwf_minus_eta_ptr == 0) cwf_minus_eta_ptr = new class Coulomb_wave_functions
(is_it_normalized,1,-eta);
00864
00865     if ((debut != 0.0) && (cwf_minus_eta_ptr->debut != -debut))
00866     {
00867         const double arg_debut = arg (debut);
00868         const std::complex<double> sym_constant_debut = (arg_debut <= 0.0) ? (1.0/sym_constant_arg_neg) :
(1.0/sym_constant_arg_pos);
00869
00870         cwf_minus_eta_ptr->debut = -debut;
00871
00872         if ((sym_constant_debut == 0.0) || (!isfinite (sym_constant_debut)))
00873         {
00874             const std::complex<double> log_sym_constant_debut = (arg (debut) <= 0.0) ?
(-log_sym_constant_arg_neg) : (-log_sym_constant_arg_pos);
00875
00876             cwf_minus_eta_ptr->F_debut = exp (log_sym_constant_debut + log (F_debut));
00877             cwf_minus_eta_ptr->dF_debut = -exp (log_sym_constant_debut + log (dF_debut));
00878         }
00879         else
00880         {
00881             cwf_minus_eta_ptr->F_debut = F_debut*sym_constant_debut;
00882             cwf_minus_eta_ptr->dF_debut = -dF_debut*sym_constant_debut;
00883         }
00884     }
00885
00886     const double arg_z = arg (z);
00887     const std::complex<double> sym_constant = (arg_z <= 0.0) ? (sym_constant_arg_neg) :
(sym_constant_arg_pos);
00888
00889     cwf_minus_eta_ptr->F_dF (-z,F,dF);
00890
00891     if ((sym_constant == 0.0) || (!isfinite (sym_constant)))
00892     {
00893         const std::complex<double> log_sym_constant = (arg_z <= 0.0) ? (log_sym_constant_arg_neg) :
(log_sym_constant_arg_pos);
00894
00895         F = exp (log_sym_constant + log (F));
00896         dF = -exp (log_sym_constant + log (dF));
00897     }
00898     else
00899     {
00900         F *= sym_constant;
00901         dF *= -sym_constant;
00902     }
00903 }
00904
00905
00906
00907
00908
00909 // Calculation of the asymptotic series
00910 // -----
00911 // Asymptotic expansion:
00912 //  $S(+/-)(z) = 1.0 + \sum_{n=1}^{\infty} a[n]$  with
a[n+1]=a[n].(n.(n+1+/-2i.eta)+i.eta.(i.eta+/-1)-1(1+1))/[+/-2i.(n+1)]/z, n >= 0 and a[0] = 1.
00913 // This expansion diverges : it is only useful with the smallest term summation method.
00914 // 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.
00915 // Practically, one stops when test < precision (it worked) or when test is not finite (it failed).
00916 // After that, one tests the series with the wronskian of H[omega] and H[-omega].
00917 //
00918 // Variables :
00919 // -----
00920 // z is the variable of the Coulomb wave function.
00921 // omega : 1 if one calculates H+(z) and H' (z), -1 if one calculates H-(z) and H' (z).
00922 // one_over_z : 1/z.
00923 // sum : sum[0] is the series in H(omega), sum[1] the one in H(-omega).
00924 // dsum : dsum[0] is the series in H' (omega), dsum[1] the one in H' (-omega).
00925 // is_it_successful : true if the series converged and |wronskian - 2.i.omega|oo is smaller than
precision, false if not.
00926 // test : test of convergence of the asymptotic series. It is max(|a[n]|oo,|n.a[n]/z|oo).
00927 // sign : if i=0, it is omega, if i=1, it is -omega.
00928 // Ieta,two_I_eta_sign : i.eta, 2.i.eta if sign = 1, -2.i.eta if sign = -1.
00929 // Ieta,Ieta_plus_sign_minus_ll_plus_one : i.eta(i.eta+1) - 1(1+1) if sign=1, i.eta(i.eta-1) - 1(1+1)
if sign=-1.
00930 // n,an_sign : index of the series, a[n+1]
00931 // 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.
00932 // sum_term, dsum_term : an_sign, (n+1).an_sign/z
00933 // two_I_omega : 2.i.omega .
00934
00935 void Coulomb_wave_functions::asymptotic_series (const int omega,const std::complex<double>
&one_over_z,
00936                                     std::complex<double> sum[],std::complex<double> dsum[],bool &is_it_successful)
00937 {
00938     sum[0] = sum[1] = 1.0;

```

```

00939     dsum[0] = dsum[1] = 0.0;
00940
00941     double test;
00942
00943     for (unsigned int i = 0 ; i <= 1 ; i++)
00944     {
00945         const int sign = (i == 0) ? (omega) : (-omega);
00946         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) - 1*(1+1.0);
00947
00948         int n = 0;
00949         std::complex<double> an_sign = 1.0;
00950
00951         do
00952         {
00953             const double n_plus_one = n + 1.0;
00954             const std::complex<double> sign_one_over_two_I_n_plus_one(0,-sign*0.5/n_plus_one);
00955
00956             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;
00957
00958             const std::complex<double> sum_term = an_sign,dsum_term = n_plus_one*an_sign*one_over_z;
00959
00960             sum[i] += sum_term;
00961             dsum[i] -= dsum_term;
00962
00963             test = std::max (inf_norm (sum_term),inf_norm (dsum_term));
00964             n++;
00965         }
00966         while ((test > precision) && (isfinite (test)));
00967
00968         if (!isfinite (test)) {is_it_successful = false; return;}
00969     }
00970
00971     const std::complex<double> two_I_omega(0.0,2.0*omega);
00972     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);
00973 }
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983 // Numerical partial derivatives according to l or eta.
00984 // -----
00985 // One calculates here the partial derivatives according to l or eta
00986 // of the Coulomb wave functions F(x) or G(x) and of their derivatives with x F'(x) or G'(x),
00987 // with l_r=Re[l] and eta_r=Re[eta].
00988 // One considers here the parameters l_r and eta_r with the argument x.
00989 // For this, one uses the standard formula : df/d_chi(x) = [f(x,chi+eps) - f(x,chi-eps)]/[2.eps],
00990 // with chi = l_r or eta_r and eps = prec_first_order_expansion*chi. f is either F, F', G or G'.
00991 // One then calculates F or G, F' or G' with chi+eps and chi-eps.
00992 // 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.
00993 // All functions are real, so double values are returned, taking the real part of std::complex
variables.
00994 //
00995 // Variables
00996 // -----
00997 // is_it_regular : true if one calculates derivatives of F(z),F'(z), false if one calculates
derivatives of G(z),G'(z).
00998 // is_it_eta : true if one calculates the partial derivatives according to eta,
00999 // false if one calculates the partial derivatives according to l.
01000 // 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.
01001 // If one calculates the partial derivatives according to eta, they are
*cwf_real_eta_plus_ptr and *cwf_real_eta_minus_ptr.
01002 // If one calculates the partial derivatives according to l, they are
*cwf_real_l_plus_ptr and *cwf_real_l_minus_ptr.
01003 // x : Re[z], z the argument of the wave function.
01004 // 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.
01005 // l_r,eta_r,chi_r : Re[l],Re[eta], Re[eta] is is_it_eta is true, Re[l] if not.
01006 // chi_r_plus,chi_r_minus : chi_r+eps, chi_r-eps.
01007 // 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,
01008 // H+(x,chi+eps),H+(x,chi+eps),H+(x,chi-eps),H+(x,chi-eps) if
is_it_regular is false.
01009 // 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,
01010 // G(x,chi+eps),G'(x,chi+eps),G(x,chi-eps),G'(x,chi-eps) if
is_it_regular is false.
01011
01012 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)
01013 {
01014     const double l_r = real (l),eta_r = real (eta),chi_r = (is_it_eta) ? (eta_r) : (l_r);
01015     const double chi_r_plus = (chi_r != 0.0) ? (chi_r*(1.0 + prec_first_order_expansion)) :
(prec_first_order_expansion);
01016     const double chi_r_minus = (chi_r != 0.0) ? (chi_r*(1.0 - prec_first_order_expansion)) :
(-prec_first_order_expansion);
01017
01018     if (is_it_eta)
01019     {
01020         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);
01021         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);
01022     }
01023     else
01024     {
01025         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);
01026         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);
01027     }
01028
01029     class Coulomb_wave_functions &cwf_plus = (is_it_eta) ? (*cwf_real_eta_plus_ptr) :
(*cwf_real_l_plus_ptr);
01030     class Coulomb_wave_functions &cwf_minus = (is_it_eta) ? (*cwf_real_eta_minus_ptr) :
(*cwf_real_l_minus_ptr);
01031
01032     std::complex<double> A_plus,dA_plus,A_minus,dA_minus;
01033
01034     if (is_it_regular)
01035     {
01036         cwf_plus.F_dF (x,A_plus,dA_plus);
01037         cwf_minus.F_dF (x,A_minus,dA_minus);
01038     }
01039     else
01040     {
01041         cwf_plus.H_dH (l,x,A_plus,dA_plus);
01042         cwf_minus.H_dH (l,x,A_minus,dA_minus);
01043     }
01044
01045     const double B_plus = real (A_plus),B_minus = real (A_minus),dB_plus = real (dA_plus),dB_minus =
real (dA_minus);
01046
01047     d_chi_Bx = (B_plus - B_minus)/(chi_r_plus - chi_r_minus);
01048     d_chi_dBx = (dB_plus - dB_minus)/(chi_r_plus - chi_r_minus);
01049 }
01050
01051
01052
01053
01054
01055
01056
01057
01058 // Calculation of F(z),F'(z) or G(z),G'(z) with the first order expansion method.
01059 // -----
01060 // When imaginary parts of l,eta,z are much smaller than their real parts but not all zero, with Re[z]
> 0,
01061 // 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.
01062 // Re[z] < 0 is not considered, as G(z) is std::complex with Im[z] = Im[l] = Im[eta] = 0.
01063 // So, with z = x+1.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).
01064 //
01065 // One considers here the parameters l_r and eta_r with the argument x, and the parameters l and eta
with the argument z.
01066 //
01067 // One then has F(x),F'(x) from usual relations and one takes their real parts.
01068 // One also has H+(x),H+'(x) from usual relations and G(x) = Re[H+(x)],G'(x) = Re[H+'(x)].
01069 //
01070 // 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:
01071 //
01072 // (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).
01073 // (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).
01074 //
01075 // Variables:
01076 // -----
01077 // is_it_regular : true if one calculates F(z),F'(z), false if one calculates G(z),G'(z).
01078 // z : variable of the Coulomb wave function.
01079 // B,dB : F(z),F'(z) if is_it_regular is true, G(z),G'(z) if it is false.
01080 // x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
01081 // A_x,dA_x : F(x),F'(x) if is_it_regular is true, H+(x),H+'(x) if it is false.
01082 // Bx,dBx : F(x),F'(x) if is_it_regular is true, G(x),G'(x) if it is false.
01083 // cwf_real : class Coulomb_wave_functions of parameters l_r and eta_r.

```

```

01084 // one_over_x,two_eta_r : 1/x, 2.eta_r.
01085 // 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)
01086 // They are initialized at zero, and not calculated if they are
multiplied by zero after.
01087 // 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),
01088
01089
01090 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)
01091 {
01092     const double x = real (z),y = imag (z),l_r = real (l),l_i = imag (l),eta_r = real (eta),eta_i = imag
(eta);
01093
01094     if (cwf_real_ptr == 0) cwf_real_ptr = new class Coulomb_wave_functions (is_it_normalized,l_r,eta_r);
01095     class Coulomb_wave_functions &cwf_real = *cwf_real_ptr;
01096
01097     std::complex<double> A_x,dA_x;
01098     if (is_it_regular)
01099         cwf_real.F_dF (x,A_x,dA_x);
01100     else
01101         cwf_real.H_dH (l,x,A_x,dA_x);
01102
01103     const double Bx = real (A_x),dBx = real (dA_x);
01104
01105     double d_l_Bx = 0.0,d_l_dBx = 0.0,d_eta_Bx = 0.0,d_eta_dBx = 0.0;
01106     if (l_i != 0.0) partial_derivatives (is_it_regular,false,x,d_l_Bx,d_l_dBx);
01107     if (eta_i != 0.0) partial_derivatives (is_it_regular,true,x,d_eta_Bx,d_eta_dBx);
01108
01109     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;
01110     const double d2Bx = Coulomb_eq_x*Bx;
01111
01112     B = std::complex<double> (Bx,y*dBx + l_i*d_l_Bx + eta_i*d_eta_Bx);
01113     dB = std::complex<double> (dBx,y*d2Bx + l_i*d_l_dBx + eta_i*d_eta_dBx);
01114 }
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128 // Regular wave function and derivative.
01129 // -----
01130 // One calculates F(z) and F'(z), so F(z) ~ z^{l+1} for z -> 0 if is_it_normalized is false,
01131 // F(z) ~ C(l,eta).z^{l+1} for z -> 0 if is_it_normalized is true.
01132 // If |z| <= 0.5, one uses the power series.
01133 //
01134 // If |z| > 0.5 and Re[z] < 0, one calculates F(z) from F[l,-eta,-z] with
F_dF_with_symmetry_relations.
01135 //
01136 // If |z| > 0.5 and Re[z] >= 0, and l+l+/-i.eta no negative integer, one first tries the asymptotic
series formula.
01137 // If it failed, one integrates directly the regular Coulomb wave function with
F_dF_direct_integration.
01138 // If l+l+/-i.eta is a negative integer, this is the only available method besides power series so it
is accepted.
01139 // If l+l+/-i.eta is no negative integer but it failed again,
01140 // one calculates the Coulomb wave function H[omega] with H_dH_direct_integration and omega =
sign(Im[z]).
01141 // omega is chosen so one cannot encounter the branch cut of h[omega].
01142 // H[-omega] is calculated from H[omega] and continued fractions h[omega] and h[-omega].
01143 // One then has F(z) = (H[omega] - H[-omega])/(2.i.omega.norm), F'(z) = (H'[omega] -
H'[-omega])/(2.i.omega.norm),
01144 // with norm = 1 if the wave functions are normalized and C(l,eta)^2 if not.
01145 // The formula is stable as one uses this case only when |F(z)| > 0.1 .
01146 //
01147 // One takes only real parts if l, eta and z are real.
01148 // At the end of the function, one puts {debut,F_debut,dF_debut} equal to {z,F,dF}.
01149 //
01150 // Variables:
01151 // -----
01152 // z : variable of the Coulomb wave function.
01153 // F,dF : regular Coulomb wave function in z and derivative in z.
01154 // x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
01155 // is_it_successful : true if the asymptotic expansions converges, false if not (after
asymptotic_expansion_F_dF).
01156 // true if the direct integration worked, false if not (after
F_dF_direct_integration).
01157 // omega : sign[Im[z]]. H[omega](z) is calculated if the direct integration of F failed.

```

```

01158 // 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.
01159 // h_omega, h_minus_omega : log derivatives of H[omega](z) and H[-omega](z) calculated with continued
    fractions.
01160 // H_omega,dH_omega,H_minus_omega,dH_minus_omega : H[omega](z), H[-omega](z) and derivatives
01161
01162 void Coulomb_wave_functions::F_dF (const std::complex<double> &z,std::complex<double>
    &F,std::complex<double> &dF)
01163 {
01164     const double x = real (z),y = imag (z),l_r = real (1),l_i = imag (1),eta_r = real (eta),eta_i = imag
    (eta);
01165
01166     if ((y != 0.0) || (eta_i != 0.0) || (l_i != 0.0))
01167         && (std::abs (y) < sqrt_precision*std::min (1.0,x)) && (std::abs (eta_i) < sqrt_precision) &&
    (std::abs (l_i) < sqrt_precision)
01168         && (!neg_int_omega_one && !neg_int_omega_minus_one))
01169         first_order_expansions (true,z,F,dF);
01170     else if (abs (z) <= 0.5)
01171         F_dF_power_series (z,F,dF);
01172     else
01173     {
01174         if (real (z) < 0.0)
01175             F_dF_with_symmetry_relations (z,F,dF);
01176         else
01177         {
01178             bool is_it_successful = false;
01179             if (!neg_int_omega_one && !neg_int_omega_minus_one) asymptotic_expansion_F_dF
    (z,F,dF,is_it_successful);
01180
01181             if (!is_it_successful)
01182             {
01183                 F_dF_direct_integration (z,F,dF,is_it_successful);
01184
01185                 if (!neg_int_omega_one && !neg_int_omega_minus_one && !is_it_successful)
01186                 {
01187                     const int omega = SIGN(imag (z));
01188                     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);
01189                     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;
01190
01191                     std::complex<double> H_omega,dH_omega;
01192                     H_dH_direct_integration (omega,z,H_omega,dH_omega);
01193
01194                     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;
01195                     F = (H_omega - H_minus_omega)*one_over_two_I_term;
01196                     dF = (dH_omega - dH_minus_omega)*one_over_two_I_term;
01197                 }}
01198             }
01199
01200             if (!isfinite (F) || !isfinite (dF)) std::cout<<"Numerical failure encountered in
    F_dF."<<std::endl,exit (1);
01201
01202             if ((y == 0.0) && (eta_i == 0.0) && (l_i == 0.0)) F = real (F),dF = real (dF);
01203             debut = z,F_debut = F,dF_debut = dF;
01204         }
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214 // Calculation of G(z) and G'(z).
01215 // -----
01216 // One calculates the irregular Coulomb wave function from H+ and F.
01217 // If 1+l+i.omega.eta is a negative integer, G is by definition H[-omega].
01218 // If not, one uses the formulas :
01219 // G(z) = H+(z) - i.F(z), G'(z) = H+'(z) - i.F'(z) if is_it_normalized is true,
01220 // G(z) = H+(z) - i.Cl_eta^2.F(z), G'(z) = H+'(z) - i.Cl_eta^2.F'(z) if not.
01221 // There is no numerical inaccuracy as G is never a minimal solution.
01222 // One takes only real parts if l, eta and z are real.
01223 //
01224 // Variables :
01225 // -----
01226 // z : variable of the Coulomb wave function.
01227 // G,dG : irregular Coulomb wave functions and derivatives.
01228 // x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
01229 // H_plus,dH_plus : H+(z) and H+'(z).
01230 // F,dF : regular Coulomb wave functions and derivatives.
01231 // I_Cl_eta_square : i.C(l,eta)^2.
01232
01233 void Coulomb_wave_functions::G_dG (const std::complex<double> &z,std::complex<double>
    &G,std::complex<double> &dG)

```



```

01234 {
01235     const double x = real (z), y = imag (z), l_r = real (l), l_i = imag (l), eta_r = real (eta), eta_i = imag
        (eta);
01236
01237     if ((y != 0.0) || (eta_i != 0.0) || (l_i != 0.0))
01238         && (std::abs (y) < sqrt_precision*std::min (1.0,x)) && (std::abs (eta_i) < sqrt_precision) &&
        (std::abs (l_i) < sqrt_precision)
01239         && (!neg_int_omega_one && !neg_int_omega_minus_one))
01240         first_order_expansions (false,z,G,dG);
01241     else
01242     {
01243         if (neg_int_omega_one)
01244             H_dH (-1,z,G,dG);
01245         else if (neg_int_omega_minus_one)
01246             H_dH (1,z,G,dG);
01247         else
01248         {
01249             std::complex<double> F,dF;
01250             F_dF (z,F,dF);
01251
01252             std::complex<double> H_plus,dH_plus;
01253             H_dH (1,z,H_plus,dH_plus);
01254
01255             const std::complex<double> I(0.0,1.0);
01256
01257             if (is_it_normalized)
01258                 G = H_plus - I*F,dG = dH_plus - I*dF;
01259             else
01260             {
01261                 const std::complex<double> I_Cl_eta_square = I*Cl_eta*Cl_eta;
01262
01263                 if ((I_Cl_eta_square == 0.0) || (!isfinite (I_Cl_eta_square)))
01264                     G = H_plus - I*exp (2.0*log_Cl_eta + log (F)),dG = dH_plus - I*exp (2.0*log_Cl_eta + log (dF));
01265                 else
01266                     G = H_plus - I_Cl_eta_square*F,dG = dH_plus - I_Cl_eta_square*dF;
01267             }
01268
01269             if ((y == 0.0) && (eta_i == 0.0) && (l_i == 0.0)) G = real (G),dG = real (dG);
01270         }
01271     }
01272 }
01273
01274
01275
01276
01277 // Calculation of H[omega](z) and H'[omega](z).
01278 // -----
01279 // One first tries the asymptotic expansion formula if 1+l+/-i.eta is no negative integer.
01280 // On uses logs if the unscaling factor underflows or overflows.
01281 // 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,
01282 // one calculates H[omega](z) and H'[omega](z) with the first order expansion method.
01283 // If one is not in this case, one calculates F(z) and F'(z).
01284 // If |Im[l]| > 1 and |z| <= 1, one tries the expansion formula with F(1,eta,z) and F(-1-1,eta,z).
01285 // If not, or if it failed, one uses the continued fraction formula.
01286 // 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]].
01287 // norm is 1 if is_it_normalized is true, C(1,eta)^2 if not.
01288 //
01289 //
01290 // Variables :
01291 // -----
01292 // omega : 1 if one calculates H+(z) and H+(z), -1 if one calculates H-(z) and H-(z).
01293 // z : variable of the Coulomb wave function.
01294 // H,dH : H+(z) and H+(z) if omega=1, H-(z) and H-(z) if omega=-1.
01295 // is_it_successful : true if the asymptotic expansions converges, false if not (in
        asymptotic_expansion_H_dH_scaled).
01296 // true if the expansion formula with F(1,eta,z) and F(-1-1,eta,z) worked, false if
        not (in H_dH_with_expansion).
01297 // 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.
01298 // I_omega_z,log_unscale,unscale : i.omega,i.omega.[z - eta.log[2z]],exp[i.omega.[z - eta.log[2z]]]
01299 // x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
01300 // F,dF : regular Coulomb wave function and derivative in z.
01301 // omega_norm_functions : for l,eta,z real, omega.C(1,eta)^2 if is_it_normalized is false, omega if it
        is true.
01302 //
        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.
01303
01304 void Coulomb_wave_functions::H_dH (const int omega,const std::complex<double> &z,std::complex<double>
        &H,std::complex<double> &dH)
01305 {
01306     bool is_it_successful = false;
01307
01308     std::complex<double> H_scaled,dH_scaled;
01309     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);

```

```

01310
01311     if (is_it_successful)
01312     {
01313         const std::complex<double> I_omega(0,omega),log_unscale = I_omega*(z - eta*(M_LN2 + log
01314         (z))),unscale = exp (log_unscale);
01315
01316         if ((unscale == 0.0) || (!isfinite (unscale)))
01317             H = exp (log (H_scaled) + log_unscale),dH = exp (log (dH_scaled) + log_unscale);
01318         else
01319             H = H_scaled*unscale,dH = dH_scaled*unscale;
01320     }
01321     else
01322     {
01323         const double x = real (z),y = imag (z),l_r = real (l),l_i = imag (l),eta_r = real (eta),eta_i =
01324         imag (eta);
01325
01326         if ((y != 0.0) || (eta_i != 0.0) || (l_i != 0.0))
01327             && (std::abs (y) < sqrt_precision*std::min (1.0,x)) && (std::abs (eta_i) < sqrt_precision) &&
01328             (std::abs (l_i) < sqrt_precision)
01329             && (!neg_int_omega_one && !neg_int_omega_minus_one))
01330             H_dH_from_first_order_expansions (omega,z,H,dH);
01331         else
01332         {
01333             if (!neg_int_omega_one && !neg_int_omega_minus_one && (std::abs (l_i) >= 1.0) && (std::abs (z)
01334             <= 1.0)) H_dH_with_expansion (omega,z,H,dH,is_it_successful);
01335
01336             if (!is_it_successful) H_dH_with_F_dF_and_CF (omega,z,H,dH);
01337
01338             if ((y == 0.0) && (eta_i == 0.0) && (l_i == 0.0))
01339             {
01340                 std::complex<double> F,dF;
01341                 F_dF (z,F,dF);
01342
01343                 const double omega_norm_functions = (!is_it_normalized) ? (omega*real (Cl_eta)*real (Cl_eta)) :
01344                 (omega);
01345                 H = std::complex<double> (real (H),omega_norm_functions*real (F));
01346                 dH = std::complex<double> (real (dH),omega_norm_functions*real (dF));
01347             }
01348         }
01349     }
01350
01351     if (!isfinite (H) || !isfinite (dH)) std::cout<<"Numerical failure encountered in
01352     H_dH."<<std::endl,exit (1);
01353 }
01354
01355 // Calculation of the scaled H[omega](z) and H'[omega](z).
01356 // -----
01357 // They are H(omega)(z).exp[-i.omega.[z - eta.log[2z]]] and dH(omega)/dz(z).exp[-i.omega.[z -
01358 // eta.log[2z]]].
01359 // One first tries the asymptotic expansion formula if l+l+/-i.eta is no negative integer.
01360 // If it failed, and imaginary parts of l,eta,z are much smaller than their real parts but not all
01361 // zero, with Re[z] > 0,
01362 // one calculates H[omega](z) and H'[omega](z) with the first order expansion method.
01363 // If one is not in this case, one calculates F(z) and F'(z).
01364 // If |Im[l]| > 1 and |z| <= 1, one tries the expansion formula with F(l,eta,z) and F(-l-1,eta,z).
01365 // If not, or if it failed, one uses the continued fraction formula.
01366 // If l,eta and z are real, one rewrites H[omega] as H[omega] = Re[H[omega]] + I.omega.norm.Re[F], to
01367 // avoid numerical inaccuracies for Im[H[omega]].
01368 // norm is 1 if is_it_normalized is true, C(l,eta)^2 if not.
01369 // One uses logs if the scaling factor underflows or overflows.
01370 //
01371 // Variables :
01372 // -----
01373 // omega : 1 if one calculates H+(z) and H+(z) scaled, -1 if one calculates H-(z) and H-(z) scaled.
01374 // z : variable of the Coulomb wave function.
01375 // H_scaled,dH_scaled : H[omega](z).exp[-i.omega.[z - eta.log[2z]]] and H'[omega](z).exp[-i.omega.[z -
01376 // eta.log[2z]]].
01377 // is_it_successful : true if the asymptotic expansions converges, false if not (in
01378 // asymptotic_expansion_H_dH_scaled).
01379 // true if the expansion formula with F(l,eta,z) and F(-l-1,eta,z) worked, false if
01380 // not (in H_dH_with_expansion).
01381 // x,y,l_r,l_i,eta_r,eta_i : Re[z], Im[z], Re[l], Im[l], Re[eta], Im[eta].
01382 // F,dF : regular Coulomb wave function and derivative in z.
01383 // H,dH : H+(z) and H+(z) if omega=1, H-(z) and H-(z) if omega=-1.
01384 // omega_norm_functions : for l,eta,z real, omega.C(l,eta)^2 if is_it_normalized is false, omega if it
01385 // is true.
01386 // One indeed has H[omega] = G + I.omega.norm.F, with in this case G and
01387 // omega.norm.F as its real and imaginary parts.
01388 // I_omega_z,log_scale,scale : i.omega,-i.omega.[z - eta.log[2z]],exp[-i.omega.[z - eta.log[2z]]]
01389
01390 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)
01385 {
01386     bool is_it_successful = false;
01387     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);
01388
01389     if (!is_it_successful)
01390     {
01391         std::complex<double> H, dH;
01392
01393         const double x = real (z), y = imag (z), l_r = real (1), l_i = imag (1), eta_r = real (eta), eta_i =
            imag (eta);
01394
01395         if ((y != 0.0) || (eta_i != 0.0) || (l_i != 0.0))
01396             && (std::abs (y) < sqrt_precision*std::min (1.0,x)) && (std::abs (eta_i) < sqrt_precision) &&
                (std::abs (l_i) < sqrt_precision)
01397             && (!neg_int_omega_one && !neg_int_omega_minus_one))
01398             H_dH_from_first_order_expansions (omega, z, H, dH);
01399         else
01400         {
01403             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);
01404
01405             if (!is_it_successful) H_dH_with_F_dF_and_CF (omega, z, H, dH);
01406
01407             if ((y == 0.0) && (eta_i == 0.0) && (l_i == 0.0))
01408             {
01409                 std::complex<double> F, dF;
01410                 F_dF (z, F, dF);
01411
01412                 const double omega_norm_functions = (!is_it_normalized) ? (omega*real (Cl_eta)*real (Cl_eta)) :
                    (omega);
01413                 H = std::complex<double> (real (H), omega_norm_functions*real (F));
01414                 dH = std::complex<double> (real (dH), omega_norm_functions*real (dF));
01415             }
01416         }
01417
01418         const std::complex<double> I_omega(0, omega), log_scale = -I_omega*(z - eta*(M_LN2 + log (z))), scale
            = exp (log_scale);
01419
01420         if ((scale == 0.0) || (!isfinite (scale)))
01421             H_scaled = exp (log (H) + log_scale), dH_scaled = exp (log (dH) + log_scale);
01422         else
01423             H_scaled = H*scale, dH_scaled = dH*scale;
01424     }
01425
01426     if (!isfinite (H_scaled) || !isfinite (dH_scaled)) std::cout<<"Numerical failure encountered in
        H_dH_scaled."<<std::endl, exit (1);
01427 }
01428
01429 // Storage of initial conditions debut, F(debut), F'(debut)
01430 // -----
01431
01432
01433 void Coulomb_wave_functions::F_dF_init (const std::complex<double> &z, const std::complex<double>
    &F, const std::complex<double> &dF)
01434 {
01435     debut = z, F_debut = F, dF_debut = dF;
01436 }
01437

```

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.](#)

```

00001 #include "ode_int.H"
00002
00003 // Extrapolation in h=0 of a table of function values h close to h=0

```

```

00004 // -----
00005 //
00006 // Variables:
00007 // -----
00008 // n : number of points of the function f near h=0.
00009 // T : table containing the points f[h(0)]...f[h(n-1)] close to h=0.
00010 // f_in_zero : extrapolated value of the points f[h(0)]...f[h(n-1)] in h=0.
00011
00012
00013 std::complex<double> ODE_integration::extrapolation_in_zero (const unsigned int n,const
std::complex<double> T[]) const
00014 {
00015     std::complex<double> f_in_zero = 0.0;
00016
00017     for (unsigned int i = 0 ; i < n ; i++)
00018         f_in_zero += interpolation_term_tab[n][i]*T[i];
00019
00020     return f_in_zero;
00021 }
00022
00023
00024
00025 // Calculation of F(z,u(z)) in u''(z) = F(z,u(z))
00026 // -----
00027 //
00028 // F(z,u(z))=(1(1+1)/(z^2) + 2.eta/z - 1).u(z),
00029 //
00030 // Variables:
00031 // -----
00032 // z : parameter of the wave function.
00033 // u : discretized wave function in z.
00034 // one_over_z : 1.0/z
00035
00036 std::complex<double> ODE_integration::F_r_u (const std::complex<double> &z,const std::complex<double>
&u) const
00037 {
00038     if (1 == 0) return (two_eta/z - 1.0)*u;
00039
00040     const std::complex<double> one_over_z = 1.0/z;
00041
00042     return ((1l_plus_one*one_over_z + two_eta)*one_over_z - 1.0)*u;
00043 }
00044
00045
00046
00047 // Integration with discretization of u''(r)=F(r,u(r)) with the Henrici method.
00048 // -----
00049 //
00050 // See Numerical Recipes for the method.
00051 //
00052 // Initials conditions : r0,u(r0),du/dr(r0).
00053 // Obtained functions : r,u(r),du/dr(r).
00054 //
00055 // Variables:
00056 // -----
00057 // m : number of intervals between r0 and r
00058 // h : integration step (r-r0)/m .
00059 // r0,u0,du0 : r0,u(r0),du/dr(r0).
00060 // r,u,du : r,u(r),du/dr(r).
00061 // h_square : h*h
00062 // half_h = 0.5*h
00063 // delta : value used in the Henrici method.
00064
00065 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,const std::complex<double> &u,const std::complex<double> &du)
const
00066 {
00067     const std::complex<double> h_square = h*h,half_h = 0.5*h;
00068
00069     std::complex<double> delta = h*(du0 + half_h*F_r_u (r0,u0));
00070     u = u0 + delta;
00071
00072     for (unsigned int i = 1 ; i < m ; i++)
00073     {
00074         delta += h_square*F_r_u (r0 + i*h,u);
00075         u += delta;
00076     }
00077
00078     du = delta/h + half_h*F_r_u (r,u);
00079 }
00080
00081
00082
00083
00084
00085
00086

```

```

00087
00088
00089
00090
00091
00092 // Integration of  $u''(r) = F(r, u(r))$  with the Bulirsch-Stoer-Henrici method.
00093 // -----
00094 //
00095 // Initials conditions :  $r_0, u_0=u(r_0), du_0=du/dr(r_0)$ 
00096 // Obtained functions :  $r, u=u(r), du=du/dr(r)$ 
00097 //
00098 // See Numerical Recipes for the method.
00099 //
00100 // Variables:
00101 // -----
00102 //  $r_0, u_0, du_0$  :  $r_0, u(r_0), du/dr(r_0)$ .
00103 //  $r, u, du$  :  $r, u(r), du/dr(r)$ .
00104 //  $H, r\_debut, r\_end, u\_debut, du\_debut$  : length of an integration interval, debut and end of the
integration interval,  $u(r\_debut)$ ,  $u'(r\_debut)$ .
00105 //  $H$  is equal to  $r-r_0$  at the beginning and is divided by 2 each
time the extrapolation fails with 16 sub-intervals
00106 // between  $r\_debut$  and  $r\_end$ . If  $H = [r-r_0]/N$ , with  $N=2,4,8,16,\dots$ ,
00107 // the integration intervals are  $[r\_debut = r_0:r\_end = r_0+H], \dots$ ,
 $[r\_debut = r_0+(N-1).H, r\_end = r]$ .
00108 //  $u\_end, du\_end, k$  : 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  $r\_end$ ,
00109 // with  $0 \leq k \leq 7$ .
00110 //  $H\_over\_m\_tab$  :  $H/m$  for  $m=2,4,6,\dots,16$ .
00111 //  $inf\_norm\_half\_H$  :  $|H/2|_{\infty}$ . It is used to know if  $r = r\_debut$  up to numerical accuracy, as one has
 $|r-r\_debut|_{\infty} \leq |H/2|_{\infty}$  for this case only.
00112 //  $u\_extrapolated, u\_extrapolated\_next$  : values of  $u$  extrapolated from the points of the table  $u\_end$ 
with  $k$  and  $k \rightarrow k+1$  points,  $k \geq 2$ .
00113 // test : test to know if the method worked, i.e.,  $|u\_extrapolated/u\_extrapolated\_next - 1|_{\infty} <$ 
precision.
00114 //  $du\_extrapolated\_next$  : value of  $u'(r\_end)$  extrapolated from  $k$  points of the table  $du\_end$ ,  $k \geq 3$ .
00115 //  $r\_debut\_plus\_H$  :  $r\_debut+H$  at the end of integration is not necessarily  $r$  because of
numerical cancellations.
00116 // In this case,  $r\_end$  must be put equal to  $r$ .
00117
00118 void ODE_integration::operator() (const std::complex<double> &r0, const std::complex<double> &u0, const
std::complex<double> &du0,
00119 const std::complex<double> &r, std::complex<double> &u, std::complex<double> &du)
const
00120 {
00121     if (r == r0) {u = u0; du = du0; return;}
00122
00123     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;
00124     double test = 1.0;
00125
00126     while (test > precision)
00127     {
00128         std::complex<double> H_over_m_tab[8];
00129         for (unsigned int k = 0 ; k < 8 ; k++) H_over_m_tab[k] = H*one_over_m_tab[k];
00130         const double inf_norm_half_H = inf_norm (H_over_m_tab[0]);
00131
00132         while (inf_norm (r_debut - r) > inf_norm_half_H)
00133         {
00134             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);
00135
00136             integration_Henrici (2, H_over_m_tab[0], r_debut, u_debut, du_debut, r_end, u_end[0], du_end[0]);
00137             integration_Henrici (4, H_over_m_tab[1], r_debut, u_debut, du_debut, r_end, u_end[1], du_end[1]);
00138             std::complex<double> u_extrapolated = extrapolation_in_zero (2, u_end);
00139
00140             unsigned int k = 2;
00141             do
00142             {
00143                 integration_Henrici (m_tab[k], H_over_m_tab[k], r_debut, u_debut, du_debut, r_end, u_end[k], du_end[k]);
00144                 u_extrapolated_next = extrapolation_in_zero (++k, u_end);
00145                 test = inf_norm (u_extrapolated/u_extrapolated_next - 1.0);
00146                 u_extrapolated = u_extrapolated_next;
00147             }
00148             while ((test > precision) && (k < 7));
00149
00150             r_debut += H;
00151             u_debut = u_extrapolated_next;
00152             du_debut = du_extrapolated_next = extrapolation_in_zero (k, du_end);
00153         }
00154
00155         H *= 0.5;
00156         r_debut = r0;
00157         u_debut = u0;
00158         du_debut = du0;
00159     }
00160
00161     u = u_extrapolated_next;

```

```
00162     du = du_extrapolated_next;
00163 }
```

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.](#)

```
00001 #ifndef ABOUTAZURE2DIALOG_H
00002 #define ABOUTAZURE2DIALOG_H
00003
00004 #include <QDialog>
00005
00006 QT_BEGIN_NAMESPACE
00007 QT_END_NAMESPACE
00008
00009 class AboutAZURE2Dialog : public QDialog {
00010     Q_OBJECT
00011
00012 public:
00013     AboutAZURE2Dialog(QWidget *parent=0);
00014
00015 };
00016
00017 #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.](#)

```
00001 #ifndef ADDLEVELDIALOG_H
00002 #define ADDLEVELDIALOG_H
00003
00004 #include <QDialog>
00005 #include <QtWidgets>
00006
00007 QT_BEGIN_NAMESPACE
00008
00009 class QLineEdit;
00010 class QLabel;
00011
00012 QT_END_NAMESPACE
00013
00014 class AddLevelDialog : public QDialog {
00015     Q_OBJECT
00016
00017 public:
00018     AddLevelDialog(QWidget *parent=0);
00019     QLineEdit *jValueText;
00020     QComboBox *piValueCombo;
00021     QLineEdit *energyText;
00022
00023 private:
00024     QLabel *jValueLabel;
00025     QLabel *energyLabel;
00026
00027     QPushButton *okButton;
00028     QPushButton *cancelButton;
00029 };
00030
00031 #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.](#)

```
00001 #ifndef ADDPAIRDIALOG_H
00002 #define ADDPAIRDIALOG_H
00003
00004 #include <QDialog>
00005
00006 QT_BEGIN_NAMESPACE
00007
00008 class QLineEdit;
00009 class QLabel;
00010 class QCheckBox;
00011 class QGroupBox;
00012 class QComboBox;
00013 class QPushButton;
00014
00015 QT_END_NAMESPACE
00016
00017 class AddPairDialog : public QDialog {
00018     Q_OBJECT
00019
```

```

00020 public:
00021     AddPairDialog(QWidget *parent=0);
00022     QLineEdit *lightJText;
00023     QComboBox *lightPiCombo;
00024     QLineEdit *lightZText;
00025     QLineEdit *lightMText;
00026     //QLineEdit *lightGText;
00027     QLineEdit *heavyJText;
00028     QComboBox *heavyPiCombo;
00029     QLineEdit *heavyZText;
00030     QLineEdit *heavyMText;
00031     //QLineEdit *heavyGText;
00032     QLineEdit *excitationEnergyText;
00033     QLineEdit *seperationEnergyText;
00034     QLineEdit *channelRadiusText;
00035     QComboBox *pairTypeCombo;
00036     QCheckBox *e1Check;
00037     //QCheckBox *m1Check;
00038     QCheckBox *e2Check;
00039     QGroupBox *multBox;
00040
00041 public slots:
00042     void updateLightParticle(int index);
00043
00044 private:
00045     QLabel *lightJLabel;
00046     QLabel *lightPiLabel;
00047     QLabel *lightZLabel;
00048     QLabel *lightMLabel;
00049     //QLabel *lightGLabel;
00050     QLabel *heavyJLabel;
00051     QLabel *heavyPiLabel;
00052     QLabel *heavyZLabel;
00053     QLabel *heavyMLabel;
00054     //QLabel *heavyGLabel;
00055     QLabel *excitationEnergyLabel;
00056     QLabel *seperationEnergyLabel;
00057     QLabel *channelRadiusLabel;
00058     QLabel *pairTypeLabel;
00059
00060     QPushButton *okButton;
00061     QPushButton *cancelButton;
00062 };
00063
00064 #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.](#)

```

00001 #ifndef ADDSEGDATAIALOG_H
00002 #define ADDSEGDATAIALOG_H
00003

```



```

00004 #include <QDialog>
00005 #include <QLineEdit>
00006 #include <QSpinBox>
00007 #include <QComboBox>
00008 #include <QLabel>
00009 #include <QCheckBox>
00010
00011 QT_BEGIN_NAMESPACE
00012
00013 class QLabel;
00014
00015 QT_END_NAMESPACE
00016
00017 class AddSegDataDialog : public QDialog {
00018     Q_OBJECT
00019
00020 public:
00021     AddSegDataDialog(QWidget *parent=0);
00022     QSpinBox *entrancePairIndexSpin;
00023     QSpinBox *exitPairIndexSpin;
00024     QLineEdit *lowEnergyText;
00025     QLineEdit *highEnergyText;
00026     QLineEdit *lowAngleText;
00027     QLineEdit *highAngleText;
00028     QComboBox *dataTypeCombo;
00029     QLineEdit *dataFileText;
00030     QLineEdit *dataNormText;
00031     QLineEdit *dataNormErrorText;
00032     QLabel *dataNormErrorLabel;
00033     QCheckBox *varyNormCheck;
00034     QLineEdit *phaseJValueText;
00035     QLineEdit *phaseLValueText;
00036     QLabel* phaseLValueLabel;
00037     QLabel* phaseJValueLabel;
00038     QLabel* totalCaptureLabel;
00039
00040 public slots:
00041     void setChooseFile();
00042     void dataTypeChanged(int);
00043     void varyNormChanged(int);
00044
00045 private:
00046     QPushButton *okButton;
00047     QPushButton *cancelButton;
00048 };
00049
00050 #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.](#)

```

00001 #ifndef ADDSEGTESTDIALOG_H
00002 #define ADDSEGTESTDIALOG_H
00003

```

```

00004 #include <QDialog>
00005 #include <QLineEdit>
00006 #include <QSpinBox>
00007 #include <QComboBox>
00008 #include <QLabel>
00009
00010 QT_BEGIN_NAMESPACE
00011
00012 class QLabel;
00013
00014 QT_END_NAMESPACE
00015
00016 class AddSegTestDialog : public QDialog {
00017     Q_OBJECT
00018
00019 public:
00020     AddSegTestDialog(QWidget *parent=0);
00021     QSpinBox *entrancePairIndexSpin;
00022     QSpinBox *exitPairIndexSpin;
00023     QLineEdit *lowEnergyText;
00024     QLineEdit *highEnergyText;
00025     QLineEdit *energyStepText;
00026     QLineEdit *lowAngleText;
00027     QLineEdit *highAngleText;
00028     QLineEdit *angleStepText;
00029     QComboBox *dataTypeCombo;
00030     QLineEdit *phaseJValueText;
00031     QLineEdit *phaseLValueText;
00032     QLabel *phaseJValueLabel;
00033     QLabel *phaseLValueLabel;
00034     QLabel *angDistLabel;
00035     QLabel *totalCaptureLabel;
00036     QSpinBox *angDistSpin;
00037
00038 public slots:
00039     void dataTypeChanged(int);
00040
00041 private:
00042     QPushButton *okButton;
00043     QPushButton *cancelButton;
00044 };
00045
00046 #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.](#)

```

00001 #ifndef ADDTARGETINTDIALOG_H
00002 #define ADDTARGETINTDIALOG_H
00003
00004 #include <QDialog>
00005
00006 QT_BEGIN_NAMESPACE
00007
00008 class QLineEdit;
00009 class QSpinBox;
00010 class QCheckBox;
00011 class QTableWidget;
00012 class QGroupBox;
00013 class QSize;

```

```

00014
00015 QT_END_NAMESPACE
00016
00017 class AddTargetIntDialog : public QDialog {
00018     Q_OBJECT
00019
00020 public:
00021     AddTargetIntDialog(QWidget *parent=0);
00022     QCheckBox *isConvolutionCheck;
00023     QCheckBox *isTargetIntegrationCheck;
00024     QCheckBox *isQCoefficientCheck;
00025     QLineEdit *sigmaText;
00026     QLineEdit *segmentsListText;
00027     QSpinBox *numPointsSpin;
00028     QSpinBox *numParametersSpin;
00029     QSpinBox *numQCoefficientSpin;
00030     QLineEdit *densityText;
00031     QLineEdit *stoppingPowerEqText;
00032     QTableWidget *parametersTable;
00033     QTableWidget *qCoefficientTable;
00034     QList<double> tempParameters;
00035     QList<double> tempQCoefficients;
00036     void createParameterItem(int row, double value = 0.0);
00037     void createQCoefficientItem(int row, double value = 1.0);
00038
00039 public slots:
00040     void convolutionCheckChanged(bool checked);
00041     void targetIntCheckChanged(bool checked);
00042     void parameterSpinChanged(int newNumber);
00043     void parameterChanged(int row, int column);
00044     void qCoefficientCheckChanged(bool checked);
00045     void qCoefficientSpinChanged(int newNumber);
00046     void qCoefficientChanged(int row, int column);
00047
00048 private:
00049     QPushButton *okButton;
00050     QPushButton *cancelButton;
00051     QGroupBox *stoppingPowerBox;
00052     QGroupBox *qCoefficientBox;
00053 };
00054
00055 #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.](#)

```

00001 #ifndef AZUREMAINTHREAD_H
00002 #define AZUREMAINTHREAD_H
00003
00004 #include <QThread>
00005 #include <QEventLoop>
00006 #include <QPushButton>
00007
00008 #include "TextEditBuffer.h"
00009 #include "FilteredTextEdit.h"
00010 #include "RunTab.h"
00011 #include "AZUREMain.h"
00012 #include "Config.h"
00013 #include <iostream>
00014
00015 class AZUREMainThreadWorker : public QObject {
00016     Q_OBJECT
00017
00018     public:
00019     AZUREMainThreadWorker(const Config& configure) :
00020         azureMain_(configure) {};
00021     signals:
00022     void done();
00023     public slots:
00024     void run() {
00025         azureMain_();
00026         emit done();
00027     };
00028     private:
00029     AZUREMain azureMain_;
00030 };
00031
00032 class AZUREMainThread : public QThread {
00033     Q_OBJECT
00034     public:
00035     AZUREMainThread(RunTab *tab, const Config& configure) :
00036         stream_(&buffer_), configure_(stream_), worker_(configure_) {
00037         configure_.configfile = configure.configfile;
00038         configure_.paramMask = configure.paramMask;
00039         configure_.screenCheckMask = configure.screenCheckMask;
00040         configure_.fileCheckMask = configure.fileCheckMask;
00041         configure_.chiVariance = configure.chiVariance;
00042         configure_.outputdir = configure.outputdir;
00043         configure_.checkdir = configure.checkdir;
00044         configure_.paramfile = configure.paramfile;
00045         configure_.integralsfile = configure.integralsfile;
00046         configure_.rateParams = configure.rateParams;
00047         connect(&buffer_, SIGNAL(updateLog(QString)), tab->runtimeText, SLOT(write(QString)));
00048         connect(tab->stopAzureButton, SIGNAL(clicked()), this, SLOT(stopAZURE()));
00049         connect(this, SIGNAL(readyToRun()), &worker_, SLOT(run()));
00050         connect(&worker_, SIGNAL(done()), this, SLOT(quit()));
00051         worker_.moveToThread(this);
00052     };
00053     const Config& configure() const {return configure_;};
00054     signals:
00055     void readyToRun();
00056     public slots:
00057     void stopAZURE() {
00058         configure_.stopFlag=true;
00059     };
00060     protected:
00061     void run() {
00062         emit readyToRun();
00063         exec();
00064     };
00065     private:
00066     TextEditBuffer buffer_;
00067     std::ostream stream_;
00068     Config configure_;
00069     AZUREMainThreadWorker worker_;
00070 };
00071
00072 #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.](#)

```

00001 #ifndef AZUREPLOT_H
00002 #define AZUREPLOT_H
00003
00004 #include <qwt_plot.h>
00005 #include <qwt_symbol.h>
00006 #include <qwt_plot_zoomer.h>
00007
00008 class QwtPlotCurve;
00009 class QwtPlotIntervalCurve;
00010 class PlotTab;
00011
00012 struct PlotPoint {
00013     double energy;
00014     double excitationEnergy;
00015     double angle;
00016     double fitCrossSection;
00017     double fitSFactor;
00018     double dataCrossSection;
00019     double dataErrorCrossSection;
00020     double dataSFactor;
00021     double dataErrorSFactor;
00022 };
00023
00024 class AZUREZoomer : public QwtPlotZoomer {
00025 public:
00026     AZUREZoomer(QWidget *canvas) : QwtPlotZoomer(canvas) {};
00027 protected:
00028     QwtText trackerTextF( const QPointF &pos ) const;
00029 };
00030
00031
00032 class PlotEntry {
00033 public:
00034     PlotEntry(int type, int entranceKey, int exitKey, int index, QString filename);
00035     ~PlotEntry();
00036
00037     int type() const {return type_};
00038
00039     bool readData();
00040     void attach(QwtPlot*,int,int,QwtSymbol::Style);
00041     void detach();
00042
00043 public:
00044     friend class AZUREPlot;
00045
00046 private:
00047     bool hasNegative_;
00048     int type_;
00049     int entranceKey_;
00050     int exitKey_;
00051     int index_;
00052     QString filename_;
00053     QwtPlotCurve* dataCurve_;
00054     QwtPlotIntervalCurve* dataErrorCurve_;
00055     QwtPlotCurve* fitCurve_;
00056     QVector<PlotPoint> points_;
00057 };
00058
00059 class AZUREPlot : public QwtPlot {
00060
00061     Q_OBJECT
00062
00063 public:
00064     AZUREPlot(PlotTab* plotTab, QWidget* parent = 0);
00065     void setXAxisLog(bool set);
00066     void setYAxisLog(bool set);
00067     void setXAxisType(unsigned int type);
00068     void setYAxisType(unsigned int type);

```

```

00069     void clearEntries();
00070
00071     public slots:
00072     void draw(QList<PlotEntry*> newEntries);
00073     void update();
00074     void exportPlot();
00075     void print();
00076
00077     private:
00078     unsigned int xAxisType;
00079     unsigned int yAxisType;
00080     QList<PlotEntry*> entries;
00081     AZUREZoomer* zoomer;
00082     PlotTab* containingTab;
00083 };
00084
00085 #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.](#)

```

00001 #ifndef AZURESETUP_H
00002 #define AZURESETUP_H
00003
00004 #include <QMainWindow>
00005
00006 #include "PairsTab.h"
00007 #include "LevelsTab.h"
00008 #include "SegmentsTab.h"
00009 #include "TargetIntTab.h"
00010 #include "Config.h"
00011
00012 class RunTab;
00013 #ifdef USE_QWT
00014 class PlotTab;
00015 #endif
00016 class AZUREMainThread;
00017
00018 QT_BEGIN_NAMESPACE
00019
00020 class QTabWidget;
00021 class QMenu;
00022 class QAction;
00023 class QActionGroup;
00024 class QTextEdit;
00025
00026 QT_END_NAMESPACE
00027
00028 class Directories {
00029     public:

```

```

00030 Directories() : outputDir(QString(""), checksDir(QString("")) {}
00031     QString outputDir;
00032     QString checksDir;
00033 };
00034
00035 class AZURESetup : public QMainWindow {
00036     Q_OBJECT
00037
00038 public:
00039     AZURESetup();
00040     Config& GetConfig();
00041     void open(QString filename);
00042
00043 public slots:
00044     void SaveAndRun();
00045     void DeleteThread();
00046
00047 private slots:
00048     void reset();
00049     void open();
00050     void openRecent();
00051     void clearRecent();
00052     void save();
00053     void saveAs();
00054     void matrixChanged(QAction* action);
00055     void editChecks();
00056     void editDirs();
00057     void editOptions();
00058     void showAbout();
00059     void showTabInfo();
00060     void openWebsite();
00061
00062 private:
00063     bool readFile(QString filename);
00064     bool readConfig(QTextStream& inStream);
00065     bool writeFile(QString filename);
00066     bool writeConfig(QTextStream& outStream, QString directory);
00067     bool readLastRun(QTextStream& inStream);
00068     bool writeLastRun(QTextStream& outStream);
00069     void createActions();
00070     void createMenus();
00071     void updateRecent();
00072
00073     Config config;
00074
00075     QAction* aboutAction;
00076     QAction* resetAction;
00077     QAction* quitAction;
00078     QAction* openAction;
00079     QAction* saveAction;
00080     QAction* saveAsAction;
00081     QAction* editChecksAction;
00082     QAction* editDirsAction;
00083     QAction* copyAction;
00084     QAction* aMatrixAction;
00085     QAction* rMatrixAction;
00086     QAction* editOptionsAction;
00087     QAction* recentSeparator;
00088     QAction* clearRecentAction;
00089     enum { numRecent = 5 };
00090     QAction* recentFileActions[numRecent];
00091     QAction* showTabInfoAction;
00092     QAction* openAZURESiteAction;
00093
00094     QActionGroup* matrixActionGroup;
00095
00096     QMenu *fileMenu;
00097     QMenu *editMenu;
00098     QMenu *configMenu;
00099     QMenu *formalismMenu;
00100     QMenu *recentFileMenu;
00101     QMenu *helpMenu;
00102     QTabWidget *tabWidget;
00103     PairsTab *pairsTab;
00104     LevelsTab *levelsTab;
00105     SegmentsTab *segmentsTab;
00106     TargetIntTab *targetIntTab;
00107     RunTab *runTab;
00108     AZUREMainThread *azureMain;
00109
00110 #ifdef USE_QWT
00111     PlotTab* plotTab;
00112 #endif
00113 };
00114
00115 #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.](#)

```
00001 #ifndef CHANNELDETAILS_H
00002 #define CHANNELDETAILS_H
00003
00004 #include <QWidget>
00005
00006 QT_BEGIN_NAMESPACE
00007
00008 class QLineEdit;
00009 class QLabel;
00010
00011 QT_END_NAMESPACE
00012
00013 class ChannelDetails : public QWidget {
00014     Q_OBJECT
00015
00016 public:
00017     ChannelDetails(QWidget *parent = 0);
00018     void setNormParam(int which);
00019     QLineEdit *reducedWidthText;
00020     QLabel *details;
00021 private:
00022     QLabel *normParam;
00023     QLabel *normUnits;
00024 };
00025
00026 #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.](#)

```

00001 #ifndef CHANNELSMODEL_H
00002 #define CHANNELSMODEL_H
00003
00004 #include <QAbstractTableModel>
00005 #include <QList>
00006
00007 struct ChannelsData {
00008     static const int SIZE = 7;
00009     int isFixed;
00010     int levelIndex;
00011     int pairIndex;
00012     double sValue;
00013     int lValue;
00014     QChar radType;
00015     double reducedWidth;
00016 };
00017
00018 class PairsModel;
00019
00020 class ChannelsModel : public QAbstractTableModel {
00021     Q_OBJECT
00022
00023 public:
00024     ChannelsModel(QObject *parent = 0);
00025
00026     int rowCount(const QModelIndex &parent) const;
00027     int columnCount(const QModelIndex &parent) const;
00028     QVariant data(const QModelIndex &index, int role) const;
00029     QVariant headerData(int section, Qt::Orientation orientation, int role) const;
00030     QList<ChannelsData> getChannels() const {return channelsList;};
00031     bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
00032     bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
00033     bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
00034     Qt::ItemFlags flags(const QModelIndex &index) const;
00035     bool isChannel(const ChannelsData &channel) const;
00036     QString getSpinLabel(const ChannelsData &channel) const;
00037     void setPairsModel(PairsModel *model);
00038
00039 private:
00040     QList<ChannelsData> channelsList;
00041     PairsModel *pairsModel;
00042 };
00043
00044 #endif

```

8.35 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChooseFileButton.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.](#)

```

00001 #ifndef CHOOSEFILEBUTTON_H
00002 #define CHOOSEFILEBUTTON_H

```

```

00003
00004 #include <QPushButton>
00005 #include <QWidget>
00006 #include <QLineEdit>
00007 #include <QString>
00008
00009 class ChooseFileButton : public QPushButton {
00010
00011     Q_OBJECT;
00012
00013 public:
00014     ChooseFileButton(const QString& text, QWidget *parent = 0);
00015     void setLineEdit(QLineEdit* lineEdit);
00016
00017 public slots:
00018     void click();
00019
00020 signals:
00021     void clicked(QLineEdit *lineEdit);
00022
00023 private:
00024     QLineEdit *thisLineEdit;
00025
00026 };
00027
00028 #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.](#)

```

00001 #ifndef EDITCHECKSDIALOG_H
00002 #define EDITCHECKSDIALOG_H
00003
00004 #include <QDialog>
00005
00006 QT_BEGIN_NAMESPACE
00007
00008 class QPushButton;
00009 class QGroupBox;
00010 class QComboBox;
00011
00012 QT_END_NAMESPACE
00013
00014 class EditChecksDialog : public QDialog {
00015
00016     Q_OBJECT
00017
00018 public:
00019     EditChecksDialog(QWidget *parent = 0);
00020     QComboBox *compoundCheckCombo;
00021     QComboBox *boundaryCheckCombo;
00022     QComboBox *dataCheckCombo;
00023     QComboBox *lMatrixCheckCombo;
00024     QComboBox *legendreCheckCombo;
00025     QComboBox *coulAmpCheckCombo;
00026     QComboBox *pathwaysCheckCombo;
00027     QComboBox *angDistsCheckCombo;
00028
00029 private:
00030     QPushButton *okButton;

```

```

00031     QPushButton *cancelButton;
00032     QGroupBox *stoppingPowerBox;
00033
00034 };
00035
00036 #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.](#)

```

00001 #ifndef EDITDIRSDIALOG_H
00002 #define EDITDIRSDIALOG_H
00003
00004 #include <QDialog>
00005
00006 QT_BEGIN_NAMESPACE
00007
00008 class QPushButton;
00009 class QGroupBox;
00010 class QComboBox;
00011 class QLineEdit;
00012
00013 QT_END_NAMESPACE
00014
00015 class EditDirsDialog : public QDialog {
00016
00017     Q_OBJECT
00018
00019     public:
00020         EditDirsDialog(QWidget *parent = 0);
00021         QLineEdit *outputDirectoryText;
00022         QLineEdit *checksDirectoryText;
00023
00024     private slots:
00025         void setChooseDirectory(QLineEdit*);
00026
00027     private:
00028         QPushButton *okButton;
00029         QPushButton *cancelButton;
00030 };
00031
00032 #endif

```

8.41 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditOptionsDialog.h File Reference

```
#include <QDialog>
```

Classes

- class [EditOptionsDialog](#)

8.42 EditOptionsDialog.h

[Go to the documentation of this file.](#)

```

00001 #ifndef EDITOPTIONSIALOG_H
00002 #define EDITOPTIONSIALOG_H
00003
00004
00005 #include <QDialog>
00006
00007 QT_BEGIN_NAMESPACE
00008
00009 class QPushButton;
00010 class QGroupBox;
00011 class QCheckBox;
00012
00013 QT_END_NAMESPACE
00014
00015 QT_END_NAMESPACE
00016
00017 class EditOptionsDialog : public QDialog {
00018
00019     Q_OBJECT
00020
00021 public:
00022     EditOptionsDialog(QWidget *parent =0);
00023     QCheckBox* useBruneCheck;
00024     QCheckBox* useGSLCoulCheck;
00025     QCheckBox* ignoreExternalsCheck;
00026     QCheckBox* useRMCCheck;
00027     QCheckBox* noTransformCheck;
00028     // QCheckBox* noLongWavelengthCheck;
00029
00030 private slots:
00031     void useBruneCheckChanged(int);
00032     void useRMCCheckChanged(int);
00033
00034 private:
00035     QPushButton *okButton;
00036     QPushButton *cancelButton;
00037
00038 };
00039
00040 #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.](#)

```

00001 #ifndef ELEMENTMAP_H
00002 #define ELEMENTMAP_H
00003
00004 #include <QString>
00005 #include <map>
00006
00007 static const std::pair<int, QString> elements[] = {
00008     std::pair<int, QString>(0,"n"),

```

```
00009     std::pair<int, QString>(1, "H"),
00010     std::pair<int, QString>(2, "He"),
00011     std::pair<int, QString>(3, "Li"),
00012     std::pair<int, QString>(4, "Be"),
00013     std::pair<int, QString>(5, "B"),
00014     std::pair<int, QString>(6, "C"),
00015     std::pair<int, QString>(7, "N"),
00016     std::pair<int, QString>(8, "O"),
00017     std::pair<int, QString>(9, "F"),
00018     std::pair<int, QString>(10, "Ne"),
00019     std::pair<int, QString>(11, "Na"),
00020     std::pair<int, QString>(12, "Mg"),
00021     std::pair<int, QString>(13, "Al"),
00022     std::pair<int, QString>(14, "Si"),
00023     std::pair<int, QString>(15, "P"),
00024     std::pair<int, QString>(16, "S"),
00025     std::pair<int, QString>(17, "Cl"),
00026     std::pair<int, QString>(18, "Ar"),
00027     std::pair<int, QString>(19, "K"),
00028     std::pair<int, QString>(20, "Ca"),
00029     std::pair<int, QString>(21, "Sc"),
00030     std::pair<int, QString>(22, "Ti"),
00031     std::pair<int, QString>(23, "V"),
00032     std::pair<int, QString>(24, "Cr"),
00033     std::pair<int, QString>(25, "Mn"),
00034     std::pair<int, QString>(26, "Fe"),
00035     std::pair<int, QString>(27, "Co"),
00036     std::pair<int, QString>(28, "Ni"),
00037     std::pair<int, QString>(29, "Cu"),
00038     std::pair<int, QString>(30, "Zn"),
00039     std::pair<int, QString>(31, "Ga"),
00040     std::pair<int, QString>(32, "Ge"),
00041     std::pair<int, QString>(33, "As"),
00042     std::pair<int, QString>(34, "Se"),
00043     std::pair<int, QString>(35, "Br"),
00044     std::pair<int, QString>(36, "Kr"),
00045     std::pair<int, QString>(37, "Rb"),
00046     std::pair<int, QString>(38, "Sr"),
00047     std::pair<int, QString>(39, "Y"),
00048     std::pair<int, QString>(40, "Zr"),
00049     std::pair<int, QString>(41, "Nb"),
00050     std::pair<int, QString>(42, "Mo"),
00051     std::pair<int, QString>(43, "Tc"),
00052     std::pair<int, QString>(44, "Ru"),
00053     std::pair<int, QString>(45, "Rh"),
00054     std::pair<int, QString>(46, "Pd"),
00055     std::pair<int, QString>(47, "Ag"),
00056     std::pair<int, QString>(48, "Cd"),
00057     std::pair<int, QString>(49, "In"),
00058     std::pair<int, QString>(50, "Sn"),
00059     std::pair<int, QString>(51, "Sb"),
00060     std::pair<int, QString>(52, "Te"),
00061     std::pair<int, QString>(53, "I"),
00062     std::pair<int, QString>(54, "Xe"),
00063     std::pair<int, QString>(55, "Cs"),
00064     std::pair<int, QString>(56, "Ba"),
00065     std::pair<int, QString>(57, "La"),
00066     std::pair<int, QString>(58, "Ce"),
00067     std::pair<int, QString>(59, "Pr"),
00068     std::pair<int, QString>(60, "Nd"),
00069     std::pair<int, QString>(61, "Pm"),
00070     std::pair<int, QString>(62, "Sm"),
00071     std::pair<int, QString>(63, "Eu"),
00072     std::pair<int, QString>(64, "Gd"),
00073     std::pair<int, QString>(65, "Tb"),
00074     std::pair<int, QString>(66, "Dy"),
00075     std::pair<int, QString>(67, "Ho"),
00076     std::pair<int, QString>(68, "Er"),
00077     std::pair<int, QString>(69, "Tm"),
00078     std::pair<int, QString>(70, "Yb"),
00079     std::pair<int, QString>(71, "Lu"),
00080     std::pair<int, QString>(72, "Hf"),
00081     std::pair<int, QString>(73, "Ta"),
00082     std::pair<int, QString>(74, "W"),
00083     std::pair<int, QString>(75, "Re"),
00084     std::pair<int, QString>(76, "Os"),
00085     std::pair<int, QString>(77, "Ir"),
00086     std::pair<int, QString>(78, "Pt"),
00087     std::pair<int, QString>(79, "Au"),
00088     std::pair<int, QString>(80, "Hg"),
00089     std::pair<int, QString>(81, "Tl"),
00090     std::pair<int, QString>(82, "Pb"),
00091     std::pair<int, QString>(83, "Bi"),
00092     std::pair<int, QString>(84, "Po"),
00093     std::pair<int, QString>(85, "At"),
00094     std::pair<int, QString>(86, "Rn"),
00095     std::pair<int, QString>(87, "Fr"),
```

```

00096     std::pair<int, QString>(88,"Ra"),
00097     std::pair<int, QString>(89,"Ac"),
00098     std::pair<int, QString>(90,"Th"),
00099     std::pair<int, QString>(91,"Pa"),
00100     std::pair<int, QString>(92,"U"),
00101     std::pair<int, QString>(93,"Np"),
00102     std::pair<int, QString>(94,"Pu"),
00103     std::pair<int, QString>(95,"Am"),
00104     std::pair<int, QString>(96,"Cm"),
00105     std::pair<int, QString>(97,"Bk"),
00106     std::pair<int, QString>(98,"Cf"),
00107     std::pair<int, QString>(99,"Es"),
00108     std::pair<int, QString>(100,"Fm"),
00109     std::pair<int, QString>(101,"Md"),
00110     std::pair<int, QString>(102,"No"),
00111     std::pair<int, QString>(103,"Lr"),
00112     std::pair<int, QString>(104,"Rf"),
00113     std::pair<int, QString>(105,"Db"),
00114     std::pair<int, QString>(106,"Sg"),
00115     std::pair<int, QString>(107,"Bh"),
00116     std::pair<int, QString>(108,"Hs"),
00117     std::pair<int, QString>(109,"Mt"),
00118     std::pair<int, QString>(110,"Ds"),
00119     std::pair<int, QString>(111,"Rg"),
00120     std::pair<int, QString>(112,"Cn")
00121 };
00122
00123 static const std::map<int, QString>
00124     elementMap(elements,elements+sizeof(elements)/sizeof(std::pair<int,QString>));
00125 #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.](#)

```

00001 #ifndef FILTEREDTEXTEDIT_H
00002 #define FILTEREDTEXTEDIT_H
00003
00004 #include <QTextEdit>
00005
00006 #include <iostream>
00007
00008 class FilteredTextEdit : public QTextEdit {
00009     Q_OBJECT
00010
00011 public:
00012     FilteredTextEdit(QWidget *parent = 0) :
00013         QTextEdit(parent), filtered_(false) {
00014         QFont font("Courier");
00015         font.setStyleHint(QFont::TypeWriter);
00016         setCurrentFont(font);
00017     };
00018     void SetMouseFiltered(bool filtered) { filtered_=filtered;};
00019     bool IsMouseFiltered() const {return filtered_;};
00020 public slots:
00021     void write(QString string) {
00022         if(string[0]!='\r') {
00023             QTextCursor cursor = textCursor();

```

```

00024         cursor.select(QTextCursor::LineUnderCursor);
00025         setTextCursor(cursor);
00026         string.remove(0,1);
00027     }
00028     insertPlainText(string);
00029 };
00030 void mousePressEvent(QMouseEvent *event) {
00031     if(!filtered_) QTextEdit::mousePressEvent(event);
00032 };
00033 void mouseDoubleClickEvent(QMouseEvent *event) {
00034     if(!filtered_) QTextEdit::mouseDoubleClickEvent(event);
00035 }
00036 private:
00037     bool filtered_;
00038 };
00039
00040 #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.](#)

```

00001 #ifndef INFODIALOG_H
00002 #define INFODIALOG_H
00003
00004 #include <QDialog>
00005
00006 class InfoDialog : public QDialog {
00007     Q_OBJECT
00008
00009     public:
00010         InfoDialog(const QString&, QWidget* parent=0, QString title="");
00011 };
00012
00013
00014 #endif

```

8.49 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsHeaderView.h File Reference ↩

```
#include <QHeaderView>
#include <QMouseEvent>
```

Classes

- class [LevelsHeaderView](#)

8.50 LevelsHeaderView.h

[Go to the documentation of this file.](#)

```
00001 #ifndef LEVELSHEADERVIEW_H
00002 #define LEVELSHEADERVIEW_H
00003
00004 #include <QHeaderView>
00005 #include <QMouseEvent>
00006
00007 class LevelsHeaderView : public QHeaderView {
00008 public:
00009     LevelsHeaderView(Qt::Orientation orientation, QWidget *parent) :
00010         QHeaderView(orientation, parent) {
00011         setSectionsClickable(true);
00012     };
00013 protected:
00014     virtual void mouseMoveEvent ( QMouseEvent * e ) {
00015         int pos = orientation() == Qt::Horizontal ? e->x() : e->y();
00016         int section = logicalIndexAt(pos);
00017         if(section>1) QHeaderView::mouseMoveEvent(e);
00018     };
00019     virtual void mousePressEvent ( QMouseEvent * e ) {
00020         int pos = orientation() == Qt::Horizontal ? e->x() : e->y();
00021         int section = logicalIndexAt(pos);
00022         if(section>1) QHeaderView::mousePressEvent(e);
00023     };
00024     virtual void mouseReleaseEvent ( QMouseEvent * e ) {
00025         int pos = orientation() == Qt::Horizontal ? e->x() : e->y();
00026         int section = logicalIndexAt(pos);
00027         if(section>1) QHeaderView::mouseReleaseEvent(e);
00028     };
00029 };
00030
00031 #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.](#)

```
00001 #ifndef LEVELSMODEL_H
00002 #define LEVELSMODEL_H
00003
00004 #include <QAbstractTableModel>
00005 #include <QList>
00006
00007 struct LevelsData {
00008     static const int SIZE = 5;
00009     int isActive;
00010     int isFixed;
00011     double jValue;
00012     int piValue;
00013     double energy;
00014 };
00015
00016 class LevelsModel : public QAbstractTableModel {
```



```

00017     Q_OBJECT
00018
00019 public:
00020     LevelsModel(QObject *parent = 0);
00021
00022     int rowCount(const QModelIndex &parent) const;
00023     int columnCount(const QModelIndex &parent) const;
00024     QVariant data(const QModelIndex &index, int role) const;
00025     QVariant headerData(int section, Qt::Orientation orientation, int role) const;
00026     bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
00027     bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
00028     bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
00029     Qt::ItemFlags flags(const QModelIndex &index) const;
00030     int isLevel(const LevelsData &level) const;
00031     QList<LevelsData> getLevels() const {return levelsList;};
00032     QString getSpinLabel(const LevelsData &level) const;
00033
00034 private:
00035     QList<LevelsData> levelsList;
00036 };
00037
00038 #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.](#)

```

00001 #ifndef LEVELSTAB_H
00002 #define LEVELSTAB_H
00003
00004 #include <QWidget>
00005 #include <QItemSelection>
00006 #include <QLineEdit>
00007 #include <QSpinBox>
00008 #include <QPushButton>
00009 #include <QTableView>
00010 #include <QSortFilterProxyModel>
00011 #include <QSignalMapper>
00012 #include <QPointer>
00013 #include "PairsModel.h"
00014 #include "LevelsModel.h"
00015 #include "ChannelsModel.h"
00016 #include "ChannelDetails.h"

```

```

00017
00018 class InfoDialog;
00019
00020 class LevelsTab : public QWidget {
00021     Q_OBJECT
00022
00023 public:
00024     LevelsTab(QWidget *parent = 0);
00025     void setPairsModel(PairsModel*);
00026     void updateChannelsLevelAdded(int levelIndex);
00027     void updateChannelsLevelDeleted(int levelIndex);
00028     void updateChannelsLevelEdited(int levelIndex);
00029     QList<ChannelsData> calculateChannels(int levelIndex);
00030     bool writeNuclearFile(QTextStream& outStream);
00031     bool readNuclearFile(QTextStream& inStream);
00032     void reset();
00033
00034 public slots:
00035     void addLevel();
00036     void addLevel(LevelsData level, bool fromFile);
00037     void removeLevel();
00038     void editLevel();
00039     void updateButtons(const QItemSelection &selection);
00040     void updateFilter(const QItemSelection &selection);
00041     void updateChannelsPairAddedEdited();
00042     void updateChannelsPairRemoved(int pairIndex);
00043     void updateDetails(const QItemSelection &selection);
00044     void updateReducedWidth(const QString &string);
00045     void showInfo(int which=0,QString title="");
00046
00047 signals:
00048     void readNewPair(PairsData,int,bool);
00049     void readExistingPair(PairsData,int,bool);
00050
00051 private:
00052     QSpinBox *maxLSpin;
00053     QSpinBox *maxMultSpin;
00054     QSpinBox *maxNumMultSpin;
00055     QPushButton *addLevelButton;
00056     QPushButton *removeLevelButton;
00057     PairsModel *pairsModel;
00058     LevelsModel *levelsModel;
00059     ChannelsModel *channelsModel;
00060     QTableView *levelsView;
00061     QTableView *channelsView;
00062     QSortFilterProxyModel *levelsModelProxy;
00063     QSortFilterProxyModel *proxyModel;
00064     ChannelDetails *channelDetails;
00065     QSignalMapper* mapper;
00066     QPushButton *infoButton[5];
00067     static const std::vector<QString> infoText;
00068     QPointer<InfoDialog> infoDialog[5];
00069 };
00070
00071
00072 #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.](#)

```
00001 #ifndef PAIRSMODEL_H
00002 #define PAIRSMODEL_H
00003
00004 #include <QAbstractTableModel>
00005 #include <QList>
00006
00007 struct PairsData {
00008     static const int SIZE = 15;
00009     double lightJ;
00010     int lightPi;
00011     int lightZ;
00012     double lightM;
00013     double lightG;
00014     double heavyJ;
00015     int heavyPi;
00016     int heavyZ;
00017     double heavyM;
00018     double heavyG;
00019     double excitationEnergy;
00020     double seperationEnergy;
00021     double channelRadius;
00022     int pairType;
00023     int ecMultMask;
00024 };
00025
00026 class PairsModel : public QAbstractTableModel {
00027     Q_OBJECT
00028
00029 public:
00030     PairsModel(QObject *parent = 0);
00031
00032     int rowCount(const QModelIndex &parent) const;
00033     int columnCount(const QModelIndex &parent) const;
00034     QVariant data(const QModelIndex &index, int role) const;
00035     QVariant headerData(int section, Qt::Orientation orientation, int role) const;
00036     bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
00037     bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
00038     bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
00039
00040     int isPair(const PairsData &pair) const;
00041     int numPairs() const {return pairsList.size();};
00042     QList<PairsData> getPairs() const {return pairsList;};
00043     QString getParticleLabel(const PairsData &pair, int which=-1) const;
00044     QString getReactionLabel(const PairsData &firstPair, const PairsData &secondPair);
00045     QString getReactionLabelTotalCapture(const PairsData &firstPair);
00046     QString getSpinLabel(const PairsData &pair, int which) const;
00047
00048 private:
00049     QList<PairsData> pairsList;
00050 };
00051
00052 #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.](#)

```

00001 #ifndef PAIRSTAB_H
00002 #define PAIRSTAB_H
00003
00004 #include <QWidget>
00005 #include <QItemSelection>
00006 #include <QSignalMapper>
00007 #include <QTableView>
00008 #include <QPushButton>
00009 #include <QSignalMapper>
00010 #include <QPointer>
00011
00012 #include "PairsModel.h"
00013 #include "AddPairDialog.h"
00014
00015 QT_BEGIN_NAMESPACE
00016
00017 class QPushButton;
00018
00019 QT_END_NAMESPACE
00020
00021 class InfoDialog;
00022
00023 class PairsTab : public QWidget {
00024     Q_OBJECT
00025
00026 public:
00027     PairsTab(QWidget *parent = 0);
00028     PairsModel *getPairsModel();
00029     bool parseOldECSection(QTextStream&);
00030
00031 public slots:
00032     void addPair();
00033     void addPair(PairsData pair,int pairIndex,bool fromFile);
00034     void editPair();
00035     void editPair(PairsData pair,int pairIndex,bool fromFile);
00036     void removePair();
00037     void updateButtons(const QItemSelection &selection);
00038     void showInfo(int which=0,QString title="");
00039
00040 signals:
00041     void pairAdded(int);
00042     void pairRemoved(int);
00043     void pairEdited(int);
00044
00045 private:
00046     PairsModel *pairsModel;
00047     QTableView *pairsView;
00048     QPushButton *addButton;
00049     QPushButton *deleteButton;
00050     QSignalMapper* mapper;
00051     QPushButton *infoButton[5];
00052     static const std::vector<QString> infoText;
00053     QPointer<InfoDialog> infoDialog[5];
00054 };
00055
00056 #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.](#)

```

00001 #ifndef PLOTTAB_H
00002 #define PLOTTAB_H
00003
00004 #include <QWidget>
00005 #include <QSortFilterProxyModel>
00006 #include <QSignalMapper>
00007 #include <QPointer>
00008
00009 class Config;
00010 class AZUREPlot;
00011 class PlotEntry;
00012 class SegmentsDataModel;
00013 class SegmentsTestModel;
00014 class InfoDialog;
00015
00016 QT_BEGIN_NAMESPACE
00017
00018 class QRadioButton;
00019 class QListView;
00020 class QCheckBox;
00021 class QPushButton;
00022 class QComboBox;
00023
00024 QT_END_NAMESPACE
00025
00026 class SegTestProxyModel : public QSortFilterProxyModel {
00027 public:
00028     SegTestProxyModel(QWidget* parent = 0) : QSortFilterProxyModel(parent) {};
00029     QVariant data(const QModelIndex& index, int role = Qt::DisplayRole) const;
00030     bool filterAcceptsRow(int source_row, const QModelIndex &source_parent) const;
00031 };
00032
00033 class SegDataProxyModel : public QSortFilterProxyModel {
00034 public:
00035     SegDataProxyModel(QWidget* parent = 0) : QSortFilterProxyModel(parent) {};
00036     QVariant data(const QModelIndex& index, int role = Qt::DisplayRole) const;
00037 };
00038
00039 class PlotTab : public QWidget {
00040
00041     Q_OBJECT
00042
00043 public:
00044     PlotTab(Config& config, SegmentsDataModel* dataModel, SegmentsTestModel* testModel, QWidget* parent
= 0);
00045     QList<PlotEntry*> getDataSegments();
00046     QList<PlotEntry*> getTestSegments();
00047     void reset();
00048
00049 public slots:
00050     void draw();
00051     void xAxisTypeChanged();
00052     void yAxisTypeChanged();
00053     void xAxisLogScaleChanged(bool);
00054     void yAxisLogScaleChanged(bool);
00055     void showInfo(int which=0, QString title="");
00056
00057 public:
00058     friend class AZUREPlot;
00059
00060 private:
00061     Config& configure;
00062     AZUREPlot* azurePlot;
00063     QListView* dataSegmentSelectorList;
00064     QListView* testSegmentSelectorList;
00065     QRadioButton* yAxisXSButton;
00066     QRadioButton* yAxisSFButton;
00067     QComboBox* xAxisTypeCombo;
00068     QCheckBox* xAxisIsLogCheck;
00069     QCheckBox* yAxisIsLogCheck;
00070     SegTestProxyModel* segTestProxyModel;
00071     SegDataProxyModel* segDataProxyModel;
00072     QPushButton* refreshButton;
00073     QPushButton* exportButton;
00074     QPushButton* printButton;
00075     QSignalMapper* mapper;
00076     QPushButton* infoButton[5];
00077     static const std::vector<QString> infoText;
00078     QPointer<InfoDialog> infoDialog[5];
00079 };
00080
00081 #endif

```

8.61 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RichTextDelegate.h File Reference

```
#include <QStyledItemDelegate>
```

Classes

- class [RichTextDelegate](#)

8.62 RichTextDelegate.h

[Go to the documentation of this file.](#)

```
00001 #ifndef RICHTEXTDELEGATE_H
00002 #define RICHTEXTDELEGATE_H
00003
00004 #include <QStyledItemDelegate>
00005
00006 class QPainter;
00007
00008 class RichTextDelegate : public QStyledItemDelegate {
00009
00010 protected:
00011     void paint(QPainter *painter, const QStyleOptionViewItem &option, const QModelIndex &index) const;
00012     QSize sizeHint ( const QStyleOptionViewItem & option, const QModelIndex & index ) const;
00013
00014 };
00015
00016 #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.](#)

```
00001 #ifndef RUNTAB_H
00002 #define RUNTAB_H
00003
00004 #include <QWidget>
00005 #include <QSignalMapper>
00006 #include <QPointer>
00007
00008 class ChooseFileButton;
00009 class FilteredTextEdit;
00010 class InfoDialog;
```

```

00011
00012 QT_BEGIN_NAMESPACE
00013
00014 class QComboBox;
00015 class QPushButton;
00016 class QRadioButton;
00017 class QLineEdit;
00018 class QTextEdit;
00019 class QGroupBox;
00020
00021 QT_END_NAMESPACE
00022
00023 class RunTab : public QWidget {
00024     Q_OBJECT
00025
00026 public:
00027     RunTab(QWidget* parent=0);
00028     friend class AZURESetup;
00029     friend class AZUREMainThread;
00030     void reset();
00031
00032 public slots:
00033     void showInfo(int which=0, QString title="");
00034
00035 private slots:
00036     void calculationTypeChanged(int index);
00037     void paramFileButtonChanged(bool checked);
00038     void integralsFileButtonChanged(bool checked);
00039     void fileTempButtonChanged(bool checked);
00040     void setChooseFile(QLineEdit* lineEdit);
00041
00042 private:
00043     QComboBox* calcType;
00044     QPushButton* calcButton;
00045     QPushButton* stopAZUREButton;
00046     QLineEdit* paramFileText;
00047     QLineEdit* integralsFileText;
00048     QRadioButton* newParamFileButton;
00049     QRadioButton* oldParamFileButton;
00050     QGroupBox* integralsFileGroup;
00051     QRadioButton* newIntegralsFileButton;
00052     QRadioButton* oldIntegralsFileButton;
00053     FilteredTextEdit* runtimeText;
00054     QLineEdit* chiVarianceText;
00055     QGroupBox* rateParamsGroup;
00056     QRadioButton* gridTempButton;
00057     QRadioButton* fileTempButton;
00058     QLineEdit* rateEntranceKey;
00059     QLineEdit* rateExitKey;
00060     QLineEdit* minTempText;
00061     QLineEdit* maxTempText;
00062     QLineEdit* tempStepText;
00063     QLineEdit* fileTempText;
00064     ChooseFileButton* rateParamsChoose;
00065     ChooseFileButton* paramFileChoose;
00066     ChooseFileButton* integralsFileChoose;
00067     QSignalMapper* mapper;
00068     QPushButton *infoButton[5];
00069     static const std::vector<QString> infoText;
00070     QPointer<InfoDialog> infoDialog[5];
00071 };
00072
00073
00074
00075 #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.](#)

```

00001 #ifndef SEGMENTS DATAMODEL_H
00002 #define SEGMENTS DATAMODEL_H
00003
00004 #include <QAbstractTableModel>
00005 #include <QList>
00006
00007 class PairsModel;
00008
00009 struct SegmentsDataData {
00010     static const int SIZE = 14;
00011     int isActive;
00012     int entrancePairIndex;
00013     int exitPairIndex;
00014     double lowEnergy;
00015     double highEnergy;
00016     double lowAngle;
00017     double highAngle;
00018     int dataType;
00019     QString dataFile;
00020     double dataNorm;
00021     double dataNormError;
00022     int varyNorm;
00023     double phaseJ;
00024     int phaseL;
00025 };
00026
00027 class SegmentsDataModel : public QAbstractTableModel {
00028     Q_OBJECT
00029
00030 public:
00031     SegmentsDataModel(QObject *parent = 0);
00032
00033     int rowCount(const QModelIndex &parent) const;
00034     int columnCount(const QModelIndex &parent) const;
00035     QVariant data(const QModelIndex &index, int role) const;
00036     QVariant headerData(int section, Qt::Orientation orientation, int role) const;
00037     bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
00038     bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
00039     bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
00040     Qt::ItemFlags flags(const QModelIndex &index) const;
00041     int isSegDataLine(const SegmentsDataData &line) const;
00042     QList<SegmentsDataData> getLines() const {return segDataLineList;};
00043     void setPairsModel(PairsModel* model);
00044     QString getReactionLabel(const QModelIndex &index);
00045
00046 private:
00047     QList<SegmentsDataData> segDataLineList;
00048     PairsModel* pairsModel;
00049 };
00050
00051
00052 #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.](#)

```

00001 #ifndef SEGMENTSTAB_H
00002 #define SEGMENTSTAB_H
00003
00004 #include <QWidget>
00005 #include <QLineEdit>
00006 #include <QItemSelection>
00007 #include <QTableView>
00008 #include <QPushButton>
00009 #include <QSignalMapper>
00010 #include <QPointer>
00011 #include "SegmentsDataModel.h"
00012 #include "SegmentsTestModel.h"
00013 #include "AddSegDataDialog.h"
00014 #include "AddSegTestDialog.h"
00015
00016 class InfoDialog;
00017
00018 class SegmentsTab : public QWidget {
00019     Q_OBJECT
00020
00021 public:
00022     SegmentsTab(QWidget *parent = 0);
00023     SegmentsTestModel* getSegmentsTestModel();
00024     SegmentsDataModel* getSegmentsDataModel();
00025     void reset();
00026     /*QLineEdit *getSegDataFileText() const {return segDataFileText;};
00027     QLineEdit *getSegTestFileText() const {return segTestFileText;};*/
00028
00029 public slots:
00030     void addSegDataLine();
00031     void addSegDataLine(SegmentsDataData line);
00032     void addSegTestLine();
00033     void addSegTestLine(SegmentsTestData line);
00034     void editSegDataLine();
00035     void editSegTestLine();
00036     void deleteSegDataLine();
00037     void deleteSegTestLine();
00038     void moveSegDataLineUp();
00039     void moveSegDataLineDown();
00040     void moveSegTestLineUp();
00041     void moveSegTestLineDown();
00042     void updateSegDataButtons(const QItemSelection &selection);
00043     void updateSegTestButtons(const QItemSelection &selection);
00044     /*void openSegDataFile();
00045     void openSegDataFile(QString filename);
00046     void saveSegDataFile();
00047     void saveAsSegDataFile();*/
00048     /*bool readSegDataFile(QString filename);*/
00049     bool readSegDataFile(QTextStream& inStream);
00050     /*bool writeSegDataFile(QString filename);*/
00051     bool writeSegDataFile(QTextStream& outStream);
00052     /*void openSegTestFile();
00053     void openSegTestFile(QString filename);
00054     void saveSegTestFile();
00055     void saveAsSegTestFile();*/
00056     /*bool readSegTestFile(QString filename);*/
00057     bool readSegTestFile(QTextStream& inStream);
00058     /*bool writeSegTestFile(QString filename);*/
00059     bool writeSegTestFile(QTextStream& outStream);
00060     void setPairsModel(PairsModel* model) {
00061         segmentsDataModel->setPairsModel(model);
00062         segmentsTestModel->setPairsModel(model);
00063     }
00064     void showInfo(int which=0,QString title="");
00065
00066 private:
00067     void moveSegDataLine(unsigned int upDown);
00068     void moveSegTestLine(unsigned int upDown);
00069
00070     /*QLineEdit *segDataFileText;*/
00071     SegmentsDataModel *segmentsDataModel;
00072     QTableView *segmentsDataView;
00073     QPushButton *segDataAddButton;

```

```

00074 //QPushButton *segDataEditButton;
00075 QPushButton *segDataDeleteButton;
00076 QPushButton *segDataUpButton;
00077 QPushButton *segDataDownButton;
00078 /*QLineEdit *segTestFileText;*/
00079 SegmentsTestModel *segmentsTestModel;
00080 QTableView *segmentsTestView;
00081 QPushButton *segTestAddButton;
00082 //QPushButton *segTestEditButton;
00083 QPushButton *segTestDeleteButton;
00084 QPushButton *segTestUpButton;
00085 QPushButton *segTestDownButton;
00086 QSignalMapper* mapper;
00087 QPushButton *infoButton[5];
00088 static const std::vector<QString> infoText;
00089 QPointer<InfoDialog> infoDialog[5];
00090 };
00091
00092 #endif

```

8.69 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTestModel.h File Reference

```

#include <QAbstractTableModel>
#include <QList>

```

Classes

- struct [SegmentsTestData](#)
- class [SegmentsTestModel](#)

8.70 SegmentsTestModel.h

[Go to the documentation of this file.](#)

```

00001 #ifndef SEGMENTSTESTMODEL_H
00002 #define SEGMENTSTESTMODEL_H
00003
00004 #include <QAbstractTableModel>
00005 #include <QList>
00006
00007 class PairsModel;
00008
00009 struct SegmentsTestData {
00010     static const int SIZE = 13;
00011     int isActive;
00012     int entrancePairIndex;
00013     int exitPairIndex;
00014     double lowEnergy;
00015     double highEnergy;
00016     double energyStep;
00017     double lowAngle;
00018     double highAngle;
00019     double angleStep;
00020     int dataType;
00021     double phaseJ;
00022     int phaseL;
00023     int maxAngDistOrder;
00024 };
00025
00026 class SegmentsTestModel : public QAbstractTableModel {
00027     Q_OBJECT
00028
00029     public:
00030         SegmentsTestModel(QObject *parent = 0);
00031
00032         int rowCount(const QModelIndex &parent) const;
00033         int columnCount(const QModelIndex &parent) const;

```

```

00034     QVariant data(const QModelIndex &index, int role) const;
00035     QVariant headerData(int section, Qt::Orientation orientation, int role) const;
00036     bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
00037     bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
00038     bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
00039     Qt::ItemFlags flags(const QModelIndex &index) const;
00040     int isSegTestLine(const SegmentsTestData &line) const;
00041     QList<SegmentsTestData> getLines() const {return segTestLineList;};
00042     void setPairsModel(PairsModel* model);
00043     QString getReactionLabel(const QModelIndex &index);
00044 private:
00045     QList<SegmentsTestData> segTestLineList;
00046     PairsModel* pairsModel;
00047 };
00048
00049
00050 #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.](#)

```

00001 #ifndef TARGETINTMODEL_H
00002 #define TARGETINTMODEL_H
00003
00004 #include <QAbstractTableModel>
00005 #include <QList>
00006
00007 Q_DECLARE_METATYPE(QList<double>);
00008
00009 struct TargetIntData {
00010     static const int SIZE = 12;
00011     int isActive;
00012     QString segmentsList;
00013     int numPoints;
00014     bool isConvolution;

```

```

00015     double sigma;
00016     bool isTargetIntegration;
00017     double density;
00018     QString stoppingPowerEq;
00019     int numParameters;
00020     QList<double> parameters;
00021     bool isQCoefficients;
00022     QList<double> qCoefficients;
00023 };
00024
00025 class TargetIntModel : public QAbstractTableModel {
00026     Q_OBJECT
00027
00028     public:
00029     TargetIntModel(QObject *parent = 0);
00030     int rowCount(const QModelIndex &parent) const;
00031     int columnCount(const QModelIndex &parent) const;
00032     QVariant data(const QModelIndex &index, int role) const;
00033     QVariant headerData(int section, Qt::Orientation orientation, int role) const;
00034     bool setData(const QModelIndex &index, const QVariant &value, int role=Qt::EditRole);
00035     bool insertRows(int position, int rows, const QModelIndex &index=QModelIndex());
00036     bool removeRows(int position, int rows, const QModelIndex &index=QModelIndex());
00037     Qt::ItemFlags flags(const QModelIndex &index) const;
00038     QList<TargetIntData> getLines() const {return targetIntList;};
00039     private:
00040     QList<TargetIntData> targetIntList;
00041 };
00042
00043 #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.](#)

```

00001 #ifndef TARGETINTTAB_H
00002 #define TARGETINTTAB_H
00003
00004 #include <QWidget>
00005 #include <QItemSelection>
00006 #include <QTableView>
00007 #include <QSignalMapper>
00008 #include <QPointer>
00009 #include "TargetIntModel.h"
00010 #include "AddTargetIntDialog.h"
00011
00012 QT_BEGIN_NAMESPACE
00013
00014 class QPushButton;
00015 class QLineEdit;
00016
00017 QT_END_NAMESPACE
00018

```

```

00019 class InfoDialog;
00020
00021 class TargetIntTab : public QWidget {
00022     Q_OBJECT
00023
00024 public:
00025     TargetIntTab(QWidget *parent = 0);
00026     TargetIntModel* getTargetIntModel();
00027     bool writeFile(QTextStream& outStream);
00028     bool readFile(QTextStream& inStream);
00029     void reset();
00030
00031 public slots:
00032     void addLine();
00033     void addLine(TargetIntData line);
00034     void editLine();
00035     void deleteLine();
00036     void updateButtons(const QItemSelection &selection);
00037     void showInfo(int which=0,QString title="");
00038
00039 private:
00040     TargetIntModel *targetIntModel;
00041     QTableView *targetIntView;
00042     QPushButton *addButton;
00043     QPushButton *deleteButton;
00044     QSignalMapper* mapper;
00045     QPushButton *infoButton[5];
00046     static const std::vector<QString> infoText;
00047     QPointer<InfoDialog> infoDialog[5];
00048 };
00049
00050 #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.](#)

```

00001 #ifndef TEXTEDITBUFFER_H
00002 #define TEXTEDITBUFFER_H
00003
00004 #include <QWidget>
00005 #include <QString>
00006 #include <string>
00007 #include <streambuf>
00008 #include <assert.h>
00009 #include <iostream>
00010
00011 class QTextEdit;
00012
00013 class TextEditBuffer : public QWidget, public std::streambuf {
00014
00015     Q_OBJECT
00016

```

```

00017 public:
00018   QTextEditBuffer(std::size_t buff_size = 256, QWidget* parent=0) : QWidget(parent),
    buffer_(buff_size+1) {
00019     char* base = &buffer_.front();
00020     setp(base, base + buffer_.size() -1);
00021   };
00022   signals:
00023     void updateLog(QString);
00024
00025   protected:
00026     virtual int_type overflow(int_type ch) {
00027       if(ch !=traits_type::eof()){
00028         assert(std::less_equal<char*>() (pptr(),epptr()));
00029         *pptr()=ch;
00030         pbump(1);
00031         if(writeToTextEdit()) return traits_type::to_int_type(ch);
00032       }
00033       return traits_type::eof();
00034     };
00035     virtual int sync() {
00036       return writeToTextEdit() ? 0 : -1;
00037     };
00038   private:
00039     bool writeToTextEdit() {
00040       std::string tempString(pbase(),pptr());
00041       pbump(pbase()-pptr());
00042       if(!tempString.empty()) emit updateLog(QString::fromStdString(tempString));
00043       return true;
00044     };
00045   private:
00046     std::vector<char> buffer_;
00047   };
00048
00049 #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.](#)

```

00001 #include <QLabel>
00002 #include <QPushButton>
00003 #include <QVBoxLayout>
00004
00005 #include "AboutAZURE2Dialog.h"
00006
00007 AboutAZURE2Dialog::AboutAZURE2Dialog(QWidget *parent) : QDialog(parent) {
00008     setWindowTitle(tr("About AZURE2"));
00009
00010     QLabel *label = new QLabel("<center><img src=\":/azure-icon.png\" width=\"128\" height=\"128\"
/><br/>"
00011                                "<b>AZURE2</b><br/>v1.0.0<br/><br/>"
00012                                "E. Uberseder, R.J. deBoer, R.E. Azuma<br/>"
00013                                "Joint Institute For Nuclear Astrophysics (JINA)</center>",
00014                                this);
00015     QPushButton* okButton = new QPushButton(tr("OK"),this);
00016     okButton->setMaximumSize(80,30);
00017
00018     QVBoxLayout* layout = new QVBoxLayout(this);
00019
00020     layout->addWidget(label);
00021     layout->addWidget(okButton);
00022     layout->setAlignment(okButton,Qt::AlignHCenter);
00023
00024     connect(okButton,SIGNAL(clicked()),this,SLOT(close()));
00025
00026     setLayout(layout);
00027 }

```

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.](#)

```
00001 #include "AddLevelDialog.h"
00002
00003 AddLevelDialog::AddLevelDialog(QWidget *parent) : QDialog(parent) {
00004
00005     this->setMaximumSize(250,150);
00006     //this->setMinimumSize(220,110);
00007
00008     QRegExp rx("^\\d{0,2}(\\.\\.[05]{0,1})?$");
00009     QValidator *validator = new QRegExpValidator(rx, this);
00010     jValueLabel = new QLabel(tr("Level Spin:"));
00011     jValueText = new QLineEdit;
00012     jValueText->setValidator(validator);
00013     piValueCombo = new QComboBox;
00014     piValueCombo->addItem("-");
00015     piValueCombo->addItem("+");
00016
00017     energyLabel = new QLabel(tr("Excitation Energy [MeV]:"));
00018     energyText = new QLineEdit;
00019
00020     cancelButton = new QPushButton(tr("Cancel"));
00021     okButton = new QPushButton(tr("Accept"));
00022     okButton->setDefault(true);
00023
00024     QHBoxLayout *energyLayout = new QHBoxLayout;
00025     energyLayout->addWidget(energyLabel);
00026     energyLayout->addWidget(energyText);
00027
00028     QGridLayout *spinLayout = new QGridLayout;
00029     spinLayout->addWidget(jValueLabel,0,0);
00030     spinLayout->addWidget(jValueText,0,1);
00031     spinLayout->addWidget(piValueCombo,0,2);
00032     spinLayout->setColumnStretch(1,1);
00033
00034     QHBoxLayout *buttonBox = new QHBoxLayout;
00035     buttonBox->addWidget(cancelButton);
00036     buttonBox->addWidget(okButton);
00037
00038     QVBoxLayout *mainLayout = new QVBoxLayout;
00039     mainLayout->addLayout(energyLayout);
00040     mainLayout->addLayout(spinLayout);
00041     mainLayout->addLayout(buttonBox);
00042
00043     setLayout(mainLayout);
00044
00045     connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
00046     connect(cancelButton,SIGNAL(clicked()),this,SLOT(reject()));
00047
00048     setWindowTitle(tr("Add a Level"));
00049 }
```

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.](#)

```
00001 #include <QComboBox>
00002 #include <QGridLayout>
00003 #include <QSpacerItem>
00004 #include <QLabel>
00005 #include <QLineEdit>
00006 #include <QPushButton>
00007 #include <QGroupBox>
00008 #include <QCheckBox>
00009
00010
00011 #include "AddPairDialog.h"
00012
00013 AddPairDialog::AddPairDialog(QWidget *parent) : QDialog(parent) {
00014
00015     //this->setMaximumSize(370,440);
00016     //this->setMinimumSize(370,440);
00017     this->setMaximumWidth(370);
00018     this->setMinimumWidth(370);
00019
00020     excitationEnergyLabel = new QLabel(tr("Excitation Energy [MeV]:"));
00021     excitationEnergyText = new QLineEdit;
00022     seperationEnergyLabel = new QLabel(tr("Separation Energy [MeV]:"));
00023     seperationEnergyText = new QLineEdit;
00024     channelRadiusLabel = new QLabel(tr("Channel Radius [fm]:"));
00025     channelRadiusText = new QLineEdit;
00026     pairTypeLabel = new QLabel(tr("Particle Pair Type:"));
00027     pairTypeCombo = new QComboBox;
00028     pairTypeCombo->addItem(tr("Particle, Particle"));
00029     pairTypeCombo->addItem(tr("Particle, Gamma"));
00030     pairTypeCombo->addItem(tr("Beta Decay"));
00031     connect(pairTypeCombo, SIGNAL(currentIndexChanged(int)), this, SLOT(updateLightParticle(int)));
00032
00033     QRegExp rx("^\\d{0,2}(\\.\\[05]{0,1})?$");
00034     QValidator *validator = new QRegExpValidator(rx, this);
00035     lightJLabel = new QLabel(tr("J:"));
00036     lightJText = new QLineEdit;
00037     lightJText->setValidator(validator);
00038     lightPiLabel = new QLabel(tr("Pi:"));
00039     lightPiCombo = new QComboBox;
00040     lightPiCombo->addItem(tr("-"));
00041     lightPiCombo->addItem(tr("+"));
00042     lightZLabel = new QLabel(tr("Z:"));
00043     lightZText = new QLineEdit;
00044     lightMLabel = new QLabel(tr("M:"));
00045     lightMText = new QLineEdit;
00046     //lightGLabel = new QLabel(tr("g:"));
00047     //lightGText = new QLineEdit;
00048
00049     heavyJLabel = new QLabel(tr("J:"));
00050     heavyJText = new QLineEdit;
00051     heavyJText->setValidator(validator);
00052     heavyPiLabel = new QLabel(tr("Pi:"));
00053     heavyPiCombo = new QComboBox;
00054     heavyPiCombo->addItem(tr("-"));
00055     heavyPiCombo->addItem(tr("+"));
00056     heavyZLabel = new QLabel(tr("Z:"));
00057     heavyZText = new QLineEdit;
00058     heavyMLabel = new QLabel(tr("M:"));
00059     heavyMText = new QLineEdit;
00060     //heavyGLabel = new QLabel(tr("g:"));
00061     //heavyGText = new QLineEdit;
00062
00063     e1Check = new QCheckBox(tr("E1"));
00064     e1Check->setChecked(false);
00065     //m1Check = new QCheckBox(tr("M1"));
00066     //m1Check->setChecked(false);
00067     e2Check = new QCheckBox(tr("E2"));
00068     e2Check->setChecked(false);
00069
00070     cancelButton = new QPushButton(tr("Cancel"));
```



```

00071     okButton = new QPushButton(tr("Accept"));
00072     okButton->setDefault(true);
00073
00074
00075     QGridLayout *pairTypeLayout = new QGridLayout;
00076     pairTypeLayout->addWidget(pairTypeLabel,0,0);
00077     pairTypeLayout->addWidget(pairTypeCombo,0,1);
00078     pairTypeLayout->setColumnStretch(1,1);
00079
00080     QGroupBox *channelGroup = new QGroupBox(tr("Channel Properties"));
00081     QGridLayout *channelLayout = new QGridLayout;
00082     channelLayout->addWidget(excitationEnergyLabel,0,0,Qt::AlignRight);
00083     channelLayout->addWidget(excitationEnergyText,0,1);
00084     channelLayout->addWidget(seperationEnergyLabel,1,0,Qt::AlignRight);
00085     channelLayout->addWidget(seperationEnergyText,1,1);
00086     channelLayout->addWidget(channelRadiusLabel,2,0,Qt::AlignRight);
00087     channelLayout->addWidget(channelRadiusText,2,1);
00088     channelGroup->setLayout(channelLayout);
00089
00090     QGroupBox *lightGroup = new QGroupBox(tr("Light Particle"));
00091     QGridLayout *lightLayout = new QGridLayout;
00092     lightLayout->addWidget(lightJLabel,0,0,Qt::AlignRight);
00093     QHBoxLayout *lightSpinLayout = new QHBoxLayout;
00094     lightSpinLayout->addWidget(lightJText);
00095     lightSpinLayout->addWidget(lightPiCombo);
00096     lightLayout->addLayout(lightSpinLayout,0,1);
00097     lightLayout->addWidget(lightZLabel,2,0,Qt::AlignRight);
00098     lightLayout->addWidget(lightZText,2,1);
00099     lightLayout->addWidget(lightMLabel,3,0,Qt::AlignRight);
00100     lightLayout->addWidget(lightMText,3,1);
00101     //lightLayout->addWidget(lightGLabel,4,0,Qt::AlignRight);
00102     //lightLayout->addWidget(lightGText,4,1);
00103     lightGroup->setLayout(lightLayout);
00104
00105     QGroupBox *heavyGroup = new QGroupBox(tr("Heavy Particle"));
00106     QGridLayout *heavyLayout = new QGridLayout;
00107     heavyLayout->addWidget(heavyJLabel,0,0,Qt::AlignRight);
00108     QHBoxLayout *heavySpinLayout = new QHBoxLayout;
00109     heavySpinLayout->addWidget(heavyJText);
00110     heavySpinLayout->addWidget(heavyPiCombo);
00111     heavyLayout->addLayout(heavySpinLayout,0,1);
00112     heavyLayout->addWidget(heavyZLabel,2,0,Qt::AlignRight);
00113     heavyLayout->addWidget(heavyZText,2,1);
00114     heavyLayout->addWidget(heavyMLabel,3,0,Qt::AlignRight);
00115     heavyLayout->addWidget(heavyMText,3,1);
00116     //heavyLayout->addWidget(heavyGLabel,4,0,Qt::AlignRight);
00117     //heavyLayout->addWidget(heavyGText,4,1);
00118     heavyGroup->setLayout(heavyLayout);
00119
00120     QHBoxLayout *entryLayout = new QHBoxLayout;
00121     entryLayout->addWidget(lightGroup);
00122     entryLayout->addWidget(heavyGroup);
00123
00124     QHBoxLayout *buttonBox = new QHBoxLayout;
00125     buttonBox->addWidget(cancelButton);
00126     buttonBox->addWidget(okButton);
00127
00128     multBox= new QGroupBox(tr("External Capture Multipolarities"));
00129     multBox->hide();
00130     QHBoxLayout *multLayout = new QHBoxLayout;
00131     multLayout->addWidget(e1Check);
00132     //multLayout->addWidget(m1Check);
00133     multLayout->addWidget(e2Check);
00134     multBox->setLayout(multLayout);
00135
00136     QVBoxLayout *mainLayout = new QVBoxLayout;
00137     mainLayout->addLayout(pairTypeLayout);
00138     mainLayout->addLayout(entryLayout);
00139     mainLayout->addWidget(channelGroup);
00140     mainLayout->addWidget(multBox);
00141     mainLayout->addLayout(buttonBox);
00142
00143     setLayout(mainLayout);
00144
00145     connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
00146     connect(cancelButton, SIGNAL(clicked()),this,SLOT(reject()));
00147
00148     setWindowTitle(tr("Add a Particle Pair"));
00149 }
00150
00151 void AddPairDialog::updateLightParticle(int index) {
00152     if(index==1) {
00153         lightJText->setText("1.0");
00154         lightJText->setEnabled(false);
00155         lightPiCombo->setCurrentIndex(1);
00156         lightPiCombo->setEnabled(false);
00157         lightZText->setText("0");

```

```

00158     lightZText->setEnabled(false);
00159     lightMText->setText("0.0");
00160     lightMText->setEnabled(false);
00161     //lightGText->setText("0.0");
00162     //lightGText->setEnabled(false);
00163     seperationEnergyText->setText("0.0");
00164     seperationEnergyText->setEnabled(false);
00165     excitationEnergyText->setEnabled(true);
00166     channelRadiusText->setText("0");
00167     channelRadiusText->setEnabled(false);
00168     multBox->show();
00169 } else if(index==2) {
00170     lightJText->setEnabled(false);
00171     lightJText->setText("0.5");
00172     lightPiCombo->setEnabled(false);
00173     lightPiCombo->setCurrentIndex(1);
00174     lightZText->setEnabled(true);
00175     lightMText->setEnabled(false);
00176     lightMText->setText("0.0005");
00177     //lightGText->setEnabled(false);
00178     //lightGText->setText("2.0023");
00179     seperationEnergyText->setEnabled(true);
00180     excitationEnergyText->setEnabled(false);
00181     excitationEnergyText->setText("0.000");
00182     channelRadiusText->setEnabled(true);
00183     multBox->hide();
00184     this->adjustSize();
00185 } else {
00186     lightJText->setEnabled(true);
00187     lightPiCombo->setEnabled(true);
00188     lightZText->setEnabled(true);
00189     lightMText->setEnabled(true);
00190     //lightGText->setEnabled(true);
00191     seperationEnergyText->setEnabled(true);
00192     excitationEnergyText->setEnabled(true);
00193     channelRadiusText->setEnabled(true);
00194     multBox->hide();
00195     this->adjustSize();
00196 }
00197 }

```

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.](#)

```

00001 #include "AddSegDataDialog.h"
00002
00003 #include <QPushButton>
00004 #include <QGroupBox>
00005 #include <QGridLayout>
00006 #include <QFileDialog>
00007
00008 AddSegDataDialog::AddSegDataDialog(QWidget *parent) : QDialog(parent) {
00009
00010     // this->setMaximumSize(370,420);
00011     // this->setMinimumSize(370,420);
00012
00013     entrancePairIndexSpin = new QSpinBox;
00014     entrancePairIndexSpin->setMinimum(1);
00015     entrancePairIndexSpin->setMaximum(100);
00016     entrancePairIndexSpin->setSingleStep(1);
00017     exitPairIndexSpin = new QSpinBox;

```

```

00018     exitPairIndexSpin->setMinimum(1);
00019     exitPairIndexSpin->setMaximum(100);
00020     exitPairIndexSpin->setSingleStep(1);
00021     lowEnergyText = new QLineEdit;
00022     highEnergyText = new QLineEdit;
00023     lowAngleText = new QLineEdit;
00024     lowAngleText->setText("0");
00025     lowAngleText->setEnabled(false);
00026     highAngleText = new QLineEdit;
00027     highAngleText->setText("180");
00028     highAngleText->setEnabled(false);
00029     dataTypeCombo = new QComboBox;
00030     dataTypeCombo->addItem(tr("Angle Integrated"));
00031     dataTypeCombo->addItem(tr("Differential"));
00032     dataTypeCombo->addItem(tr("Phase Shift"));
00033     dataTypeCombo->addItem(tr("Angle Integrated Total Capture"));
00034     connect(dataTypeCombo, SIGNAL(currentIndexChanged(int)), this, SLOT(dataTypeChanged(int)));
00035     QRegExp spinRX("^\\d{0,2}(\\.\\{05\\}{0,1})?$");
00036     QValidator *spinValidator = new QRegExpValidator(spinRX, this);
00037     phaseJValueText = new QLineEdit;
00038     phaseJValueText->setValidator(spinValidator);
00039     phaseJValueText->setVisible(false);
00040     phaseJValueText->setMaximumWidth(50);
00041     QRegExp intRX("^\\{0-6\\}$");
00042     QValidator *intValidator = new QRegExpValidator(intRX, this);
00043     phaseLValueText = new QLineEdit;
00044     phaseLValueText->setValidator(intValidator);
00045     phaseLValueText->setVisible(false);
00046     phaseLValueText->setMaximumWidth(50);
00047     dataFileText = new QLineEdit;
00048     QPushButton *chooseFileButton = new QPushButton(tr("Choose..."));
00049     connect(chooseFileButton, SIGNAL(clicked()), this, SLOT(setChooseFile()));
00050     dataNormText = new QLineEdit;
00051     dataNormText->setText("1.0");
00052     dataNormErrorLabel = new QLabel(tr("Norm. Error [%]:"));
00053     // dataNormErrorLabel->setVisible(false);
00054     dataNormErrorText = new QLineEdit(this);
00055     // dataNormErrorText->setVisible(false);
00056     dataNormErrorText->setText("0.0");
00057     dataNormErrorText->setMaximumWidth(50);
00058     varyNormCheck = new QCheckBox(tr("Vary Norm?"));
00059     //connect(varyNormCheck, SIGNAL(stateChanged(int)), this, SLOT(varyNormChanged(int)));
00060
00061     cancelButton = new QPushButton(tr("Cancel"));
00062     okButton = new QPushButton(tr("Accept"));
00063     okButton->setDefault(true);
00064
00065     QGroupBox *valueBox = new QGroupBox;
00066     QGridLayout *valueLayout = new QGridLayout;
00067     QGridLayout *pairLayout = new QGridLayout;
00068     pairLayout->addWidget(new QLabel(tr("Entrance Pair Key:")), 0, 0, Qt::AlignRight);
00069     pairLayout->addWidget(entrancePairIndexSpin, 0, 1);
00070     pairLayout->addWidget(new QLabel(tr("Exit Pair Key:")), 0, 2, Qt::AlignRight);
00071     pairLayout->addWidget(exitPairIndexSpin, 0, 3);
00072     totalCaptureLabel = new QLabel(tr("Total Capture"));
00073     totalCaptureLabel->setVisible(false);
00074     pairLayout->addWidget(totalCaptureLabel, 0, 4);
00075     valueLayout->addLayout(pairLayout, 0, 0, 1, 2);
00076     QGroupBox* energyBox = new QGroupBox(tr("Lab Energy [MeV]"));
00077     QGridLayout *energyLayout = new QGridLayout;
00078     energyLayout->addWidget(new QLabel(tr("Low Energy:")), 0, 0, Qt::AlignRight);
00079     energyLayout->addWidget(lowEnergyText, 0, 1);
00080     energyLayout->addWidget(new QLabel(tr("High Energy:")), 1, 0, Qt::AlignRight);
00081     energyLayout->addWidget(highEnergyText, 1, 1);
00082     energyBox->setLayout(energyLayout);
00083     valueLayout->addWidget(energyBox, 1, 0);
00084     QGroupBox* angleBox = new QGroupBox(tr("Lab Angle [degrees]"));
00085     QGridLayout *angleLayout = new QGridLayout;
00086     angleLayout->addWidget(new QLabel(tr("Low Angle:")), 0, 0, Qt::AlignRight);
00087     angleLayout->addWidget(lowAngleText, 0, 1);
00088     angleLayout->addWidget(new QLabel(tr("High Angle:")), 1, 0, Qt::AlignRight);
00089     angleLayout->addWidget(highAngleText, 1, 1);
00090     angleBox->setLayout(angleLayout);
00091     valueLayout->addWidget(angleBox, 1, 1);
00092
00093     QGridLayout* lowerLayout = new QGridLayout;
00094     lowerLayout->addWidget(new QLabel(tr("Data Type:")), 0, 0, Qt::AlignRight);
00095     lowerLayout->addWidget(dataTypeCombo, 0, 1);
00096
00097     QGridLayout* phaseLayout = new QGridLayout;
00098     phaseLayout->addItem(new QSpacerItem(1, 25), 0, 0);
00099     phaseLayout->setColumnStretch(0, 1);
00100     phaseJValueLabel = new QLabel(tr("J:"));
00101     phaseJValueLabel->setVisible(false);
00102     phaseLayout->addWidget(phaseJValueLabel, 0, 1);
00103     phaseLayout->addWidget(phaseJValueText, 0, 2);
00104     phaseLValueLabel = new QLabel(tr("l:"));

```

```

00105     phaseLValueLabel->setVisible(false);
00106     phaseLayout->addWidget(phaseLValueLabel,0,3);
00107     phaseLayout->addWidget(phaseLValueText,0,4);
00108     lowerLayout->addLayout(phaseLayout,0,2);
00109
00110     lowerLayout->addWidget(new QLabel(tr("Data Norm:")),1,0,Qt::AlignRight);
00111     QHBoxLayout *normLayout = new QHBoxLayout;
00112     normLayout->addWidget(dataNormText);
00113     normLayout->addWidget(varyNormCheck);
00114     lowerLayout->addLayout(normLayout,1,1);
00115     QGridLayout *normErrorLayout = new QGridLayout;
00116     normErrorLayout->addItem(new QSpacerItem(1,25),0,0);
00117     normErrorLayout->setColumnStretch(0,1);
00118     normErrorLayout->addWidget(dataNormErrorLabel,0,1,Qt::AlignRight);
00119     normErrorLayout->addWidget(dataNormErrorText,0,2);
00120     lowerLayout->addLayout(normErrorLayout,1,2);
00121
00122     lowerLayout->addWidget(new QLabel(tr("Data File:")),2,0,Qt::AlignRight);
00123     QGridLayout *fileLayout = new QGridLayout;
00124     fileLayout->addWidget(dataFileText,0,0);
00125     fileLayout->addWidget(chooseFileButton,0,1);
00126     fileLayout->setColumnStretch(0,1);
00127     lowerLayout->addLayout(fileLayout,2,1,1,2);
00128     valueLayout->addLayout(lowerLayout,2,0,1,2);
00129     valueBox->setLayout(valueLayout);
00130
00131     QHBoxLayout *buttonBox = new QHBoxLayout;
00132     buttonBox->addWidget(cancelButton);
00133     buttonBox->addWidget(okButton);
00134
00135     QVBoxLayout *mainLayout = new QVBoxLayout;
00136     mainLayout->addWidget(valueBox);
00137     mainLayout->addLayout(buttonBox);
00138
00139     setLayout(mainLayout);
00140
00141     connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
00142     connect(cancelButton, SIGNAL(clicked()),this,SLOT(reject()));
00143
00144     setWindowTitle(tr("Add a Segment From Data"));
00145 }
00146
00147 void AddSegDataDialog::setChooseFile() {
00148     QString filename = QFileDialog::getOpenFileName(this);
00149     if(!filename.isEmpty()) {
00150         dataFileText->setText(QDir::fromNativeSeparators(filename));
00151     }
00152 }
00153
00154 void AddSegDataDialog::dataTypeChanged(int index) {
00155     if(index==2) {
00156         phaseJValueLabel->setVisible(true);
00157         phaseLValueLabel->setVisible(true);
00158         phaseJValueText->setVisible(true);
00159         phaseLValueText->setVisible(true);
00160     } else {
00161         phaseJValueLabel->setVisible(false);
00162         phaseLValueLabel->setVisible(false);
00163         phaseJValueText->setVisible(false);
00164         phaseLValueText->setVisible(false);
00165     }
00166     if(index==1) {
00167         lowAngleText->setEnabled(true);
00168         highAngleText->setEnabled(true);
00169     } else {
00170         lowAngleText->setEnabled(false);
00171         highAngleText->setEnabled(false);
00172         lowAngleText->setText("0");
00173         highAngleText->setText("180");
00174     }
00175     if(index==3) {
00176         exitPairIndexSpin->setVisible(false);
00177         totalCaptureLabel->setVisible(true);
00178     } else {
00179         totalCaptureLabel->setVisible(false);
00180         exitPairIndexSpin->setVisible(true);
00181     }
00182 }
00183
00184 void AddSegDataDialog::varyNormChanged(int state) {
00185     if(state==Qt::Checked) {
00186         dataNormErrorLabel->setVisible(true);
00187         dataNormErrorText->setVisible(true);
00188     } else {
00189         dataNormErrorLabel->setVisible(false);
00190         dataNormErrorText->setVisible(false);
00191     }

```

```
00192 }
```

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.](#)

```
00001 #include "AddSegTestDialog.h"
00002
00003 #include <QPushButton>
00004 #include <QGroupBox>
00005 #include <QGridLayout>
00006 #include <QHBoxLayout>
00007
00008
00009 AddSegTestDialog::AddSegTestDialog(QWidget *parent) : QDialog(parent) {
00010
00011     // this->setMaximumSize(370,420);
00012     // this->setMinimumSize(370,420);
00013
00014     entrancePairIndexSpin = new QSpinBox;
00015     entrancePairIndexSpin->setMinimum(1);
00016     entrancePairIndexSpin->setMaximum(100);
00017     entrancePairIndexSpin->setSingleStep(1);
00018     exitPairIndexSpin = new QSpinBox;
00019     exitPairIndexSpin->setMinimum(1);
00020     exitPairIndexSpin->setMaximum(100);
00021     exitPairIndexSpin->setSingleStep(1);
00022     lowEnergyText = new QLineEdit;
00023     highEnergyText = new QLineEdit;
00024     energyStepText = new QLineEdit;
00025     lowAngleText = new QLineEdit;
00026     lowAngleText->setText("0");
00027     lowAngleText->setEnabled(false);
00028     highAngleText = new QLineEdit;
00029     highAngleText->setText("0");
00030     highAngleText->setEnabled(false);
00031     angleStepText = new QLineEdit;
00032     angleStepText->setText("0");
00033     angleStepText->setEnabled(false);
00034     dataTypeCombo = new QComboBox;
00035     dataTypeCombo->addItem(tr("Angle Integrated"));
00036     dataTypeCombo->addItem(tr("Differential"));
00037     dataTypeCombo->addItem(tr("Phase Shift"));
00038     dataTypeCombo->addItem(tr("Angular Distribution Coefficients"));
00039     dataTypeCombo->addItem(tr("Angle Integrated Total Capture"));
00040     connect(dataTypeCombo, SIGNAL(currentIndexChanged(int)), this, SLOT(dataTypeChanged(int)));
00041     QRegExp spinRX("^\\d{0,2}(\\.\\d{0,1})?$");
00042     QValidator *spinValidator = new QRegExpValidator(spinRX, this);
00043     phaseJValueText = new QLineEdit;
00044     phaseJValueText->setValidator(spinValidator);
00045     phaseJValueText->setVisible(false);
00046     phaseJValueText->setMaximumWidth(50);
00047     QRegExp intrRX("[0-6]$");
00048     QValidator *intValidator = new QRegExpValidator(intrRX, this);
00049     phaseLValueText = new QLineEdit;
00050     phaseLValueText->setValidator(intValidator);
00051     phaseLValueText->setVisible(false);
00052     phaseLValueText->setMaximumWidth(50);
00053
00054     cancelButton = new QPushButton(tr("Cancel"));
00055     okButton = new QPushButton(tr("Accept"));
00056     okButton->setDefault(true);
```

```

00057
00058     QGroupBox *valueBox = new QGroupBox;
00059     QGridLayout *valueLayout = new QGridLayout;
00060     QGridLayout *pairLayout = new QGridLayout;
00061     pairLayout->addWidget(new QLabel(tr("Entrance Pair Key:")),0,0,Qt::AlignRight);
00062     pairLayout->addWidget(entrancePairIndexSpin,0,1);
00063     pairLayout->addWidget(new QLabel(tr("Exit Pair Key:")),0,2,Qt::AlignRight);
00064     pairLayout->addWidget(exitPairIndexSpin,0,3);
00065     totalCaptureLabel = new QLabel(tr("Total Capture"));
00066     totalCaptureLabel->setVisible(false);
00067     pairLayout->addWidget(totalCaptureLabel,0,4);
00068     valueLayout->addLayout(pairLayout,0,0,1,2);
00069     QGroupBox *energyBox = new QGroupBox(tr("Lab Energy [MeV]"));
00070     QGridLayout *energyLayout = new QGridLayout;
00071     energyLayout->addWidget(new QLabel(tr("Low Energy:")),0,0,Qt::AlignRight);
00072     energyLayout->addWidget(lowEnergyText,0,1);
00073     energyLayout->addWidget(new QLabel(tr("High Energy:")),1,0,Qt::AlignRight);
00074     energyLayout->addWidget(highEnergyText,1,1);
00075     energyLayout->addWidget(new QLabel(tr("Energy Step:")),2,0,Qt::AlignRight);
00076     energyLayout->addWidget(energyStepText,2,1);
00077     energyBox->setLayout(energyLayout);
00078     valueLayout->addWidget(energyBox,1,0);
00079     QGroupBox *angleBox = new QGroupBox(tr("Lab Angle [degrees]"));
00080     QGridLayout *angleLayout = new QGridLayout;
00081     angleLayout->addWidget(new QLabel(tr("Low Angle:")),0,0,Qt::AlignRight);
00082     angleLayout->addWidget(lowAngleText,0,1);
00083     angleLayout->addWidget(new QLabel(tr("High Angle:")),1,0,Qt::AlignRight);
00084     angleLayout->addWidget(highAngleText,1,1);
00085     angleLayout->addWidget(new QLabel(tr("Angle Step:")),2,0,Qt::AlignRight);
00086     angleLayout->addWidget(angleStepText,2,1);
00087     angleBox->setLayout(angleLayout);
00088     valueLayout->addWidget(angleBox,1,1);
00089
00090     QGridLayout* lowerLayout = new QGridLayout;
00091     lowerLayout->addWidget(new QLabel(tr("Data Type:")),0,0,Qt::AlignRight);
00092     lowerLayout->addWidget(dataTypeCombo,0,1);
00093     lowerLayout->addItem(new QSpacerItem(1,25),0,2);
00094     lowerLayout->setColumnStretch(2,1);
00095
00096     QHBoxLayout* phaseLayout = new QHBoxLayout;
00097     phaseJValueLabel = new QLabel(tr("J:"));
00098     phaseJValueLabel->setVisible(false);
00099     phaseLayout->addWidget(phaseJValueLabel);
00100     phaseLayout->addWidget(phaseJValueText);
00101     phaseLValueLabel = new QLabel(tr("l:"));
00102     phaseLValueLabel->setVisible(false);
00103     phaseLayout->addWidget(phaseLValueLabel);
00104     phaseLayout->addWidget(phaseLValueText);
00105     angDistLabel = new QLabel(tr("Maximum Order"));
00106     angDistSpin = new QSpinBox;
00107     angDistSpin->setMinimum(0);
00108     angDistSpin->setMaximum(10);
00109     angDistSpin->setSingleStep(1);
00110     angDistLabel->setVisible(false);
00111     angDistSpin->setVisible(false);
00112     phaseLayout->addWidget(angDistLabel);
00113     phaseLayout->addWidget(angDistSpin);
00114     lowerLayout->addLayout(phaseLayout,0,3);
00115
00116     valueLayout->addLayout(lowerLayout,2,0,1,2);
00117     valueBox->setLayout(valueLayout);
00118
00119     QHBoxLayout *buttonBox = new QHBoxLayout;
00120     buttonBox->addWidget(cancelButton);
00121     buttonBox->addWidget(okButton);
00122
00123     QVBoxLayout *mainLayout = new QVBoxLayout;
00124     mainLayout->addWidget(valueBox);
00125     mainLayout->addLayout(buttonBox);
00126
00127     setLayout(mainLayout);
00128
00129     connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
00130     connect(cancelButton, SIGNAL(clicked()),this,SLOT(reject()));
00131
00132     setWindowTitle(tr("Add a Segment Without Data"));
00133 }
00134
00135 void AddSegTestDialog::dataTypeChanged(int index) {
00136     if(index==2) {
00137         angDistLabel->setVisible(false);
00138         angDistSpin->setVisible(false);
00139         phaseJValueLabel->setVisible(true);
00140         phaseLValueLabel->setVisible(true);
00141         phaseJValueText->setVisible(true);
00142         phaseLValueText->setVisible(true);
00143     } else if(index==3) {

```

```

00144     phaseJValueLabel->setVisible(false);
00145     phaseLValueLabel->setVisible(false);
00146     phaseJValueText->setVisible(false);
00147     phaseLValueText->setVisible(false);
00148     angDistLabel->setVisible(true);
00149     angDistSpin->setVisible(true);
00150 } else {
00151     phaseJValueLabel->setVisible(false);
00152     phaseLValueLabel->setVisible(false);
00153     phaseJValueText->setVisible(false);
00154     phaseLValueText->setVisible(false);
00155     angDistLabel->setVisible(false);
00156     angDistSpin->setVisible(false);
00157 }
00158 if(index==1) {
00159     lowAngleText->setEnabled(true);
00160     highAngleText->setEnabled(true);
00161     angleStepText->setEnabled(true);
00162 } else {
00163     lowAngleText->setEnabled(false);
00164     highAngleText->setEnabled(false);
00165     angleStepText->setEnabled(false);
00166     lowAngleText->setText("0");
00167     highAngleText->setText("0");
00168     angleStepText->setText("0");
00169 }
00170 if(index==4) {
00171     exitPairIndexSpin->setVisible(false);
00172     totalCaptureLabel->setVisible(true);
00173 } else {
00174     totalCaptureLabel->setVisible(false);
00175     exitPairIndexSpin->setVisible(true);
00176 }
00177 }

```

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.](#)

```

00001 #include <QLineEdit>
00002 #include <QSpinBox>
00003 #include <QCheckBox>
00004 #include <QTableWidget>
00005 #include <QHBoxLayout>
00006 #include <QLabel>
00007 #include <QGroupBox>
00008 #include <QHeaderView>
00009 #include <QPushButton>
00010
00011
00012 #include "AddTargetIntDialog.h"
00013
00014 AddTargetIntDialog::AddTargetIntDialog(QWidget *parent) : QDialog(parent) {
00015     segmentsListText = new QLineEdit;
00016     numPointsSpin = new QSpinBox;

```



```

00017 numPointsSpin->setEnabled(false);
00018 numPointsSpin -> setMinimum(1);
00019 numPointsSpin -> setMaximum(10000);
00020 numPointsSpin -> setSingleStep(10);
00021 numPointsSpin -> setValue(10);
00022 isConvolutionCheck = new QCheckBox(tr("Include Gaussian Convolution"));
00023 isConvolutionCheck->setChecked(false);
00024 connect(isConvolutionCheck, SIGNAL(toggled(bool)), this, SLOT(convolutionCheckChanged(bool)));
00025 sigmaText = new QLineEdit;
00026 sigmaText->setEnabled(false);
00027 isTargetIntegrationCheck = new QCheckBox(tr("Include Target Integration"));
00028 isTargetIntegrationCheck->setChecked(false);
00029 connect(isTargetIntegrationCheck, SIGNAL(toggled(bool)), this, SLOT(targetIntCheckChanged(bool)));
00030 densityText = new QLineEdit;
00031 densityText->setEnabled(false);
00032 stoppingPowerEqText = new QLineEdit;
00033 numParametersSpin = new QSpinBox;
00034 numParametersSpin -> setMinimum(0);
00035 numParametersSpin -> setMaximum(10);
00036 numParametersSpin -> setSingleStep(1);
00037 numParametersSpin -> setValue(0);
00038 connect(numParametersSpin, SIGNAL(valueChanged(int)), this, SLOT(parameterSpinChanged(int)));
00039 parametersTable = new QTableWidgetItem(this);
00040 parametersTable->setColumnCount(2);
00041 parametersTable->setRowCount(0);
00042 parametersTable->verticalHeader()->hide();
00043 parametersTable->verticalHeader()->setHighlightSections(false);
00044 parametersTable->horizontalHeader()->setHighlightSections(false);
00045 parametersTable->horizontalHeader()->setSectionResizeMode(0, QHeaderView::Stretch);
00046 parametersTable->horizontalHeader()->setSectionResizeMode(1, QHeaderView::Stretch);
00047 parametersTable->setShowGrid(false);
00048 connect(parametersTable, SIGNAL(cellChanged(int, int)), this, SLOT(parameterChanged(int, int)));
00049 QStringList labelList;
00050 labelList.append(QString(tr("Parameter")));
00051 labelList.append(QString(tr("Value")));
00052 parametersTable->setHorizontalHeaderLabels(labelList);
00053 isQCoefficientCheck = new QCheckBox(tr("Include Attenuation Coefficients"));
00054 isQCoefficientCheck->setChecked(false);
00055 connect(isQCoefficientCheck, SIGNAL(toggled(bool)), this, SLOT(qCoefficientCheckChanged(bool)));
00056 numQCoefficientSpin = new QSpinBox;
00057 numQCoefficientSpin->setMinimum(0);
00058 numQCoefficientSpin->setMaximum(6);
00059 numQCoefficientSpin->setSingleStep(1);
00060 numQCoefficientSpin->setValue(0);
00061 numQCoefficientSpin->setEnabled(false);
00062 connect(numQCoefficientSpin, SIGNAL(valueChanged(int)), this, SLOT(qCoefficientSpinChanged(int)));
00063 qCoefficientTable = new QTableWidgetItem(this);
00064 qCoefficientTable->setColumnCount(2);
00065 qCoefficientTable->setRowCount(0);
00066 qCoefficientTable->verticalHeader()->hide();
00067 qCoefficientTable->verticalHeader()->setHighlightSections(false);
00068 qCoefficientTable->horizontalHeader()->setHighlightSections(false);
00069 qCoefficientTable->horizontalHeader()->setSectionResizeMode(0, QHeaderView::Stretch);
00070 qCoefficientTable->horizontalHeader()->setSectionResizeMode(1, QHeaderView::Stretch);
00071 qCoefficientTable->setShowGrid(false);
00072 connect(qCoefficientTable, SIGNAL(cellChanged(int, int)), this, SLOT(qCoefficientChanged(int, int)));
00073 QStringList qlabelList;
00074 qlabelList.append(QString(tr("Coefficient")));
00075 qlabelList.append(QString(tr("Value")));
00076 qCoefficientTable->setHorizontalHeaderLabels(qlabelList);
00077
00078
00079 cancelButton = new QPushButton(tr("Cancel"));
00080 okButton = new QPushButton(tr("Accept"));
00081 okButton->setDefault(true);
00082
00083 QHBoxLayout *segListLayout = new QHBoxLayout;
00084 segListLayout->addWidget(new QLabel(tr("Segments List:")));
00085 segListLayout->addWidget(segmentsListText);
00086
00087 QHBoxLayout *numPointsLayout = new QHBoxLayout;
00088 numPointsLayout->addWidget(new QLabel(tr("Number of Integration Points:")));
00089 numPointsLayout->addWidget(numPointsSpin);
00090
00091 QHBoxLayout *topLayout = new QHBoxLayout;
00092 topLayout->addLayout(segListLayout);
00093 topLayout->addLayout(numPointsLayout);
00094
00095 QGridLayout *checkBoxLayout = new QGridLayout;
00096 checkBoxLayout->addWidget(isConvolutionCheck, 0, 0);
00097 checkBoxLayout->addWidget(new QLabel(tr("Sigma [MeV]:")), 0, 1, Qt::AlignRight);
00098 checkBoxLayout->addWidget(sigmaText, 0, 2);
00099
00100 checkBoxLayout->addWidget(isTargetIntegrationCheck, 1, 0);
00101 checkBoxLayout->addWidget(new QLabel(tr("Active Density [atoms/cm^2]:")), 1, 1, Qt::AlignRight);
00102 checkBoxLayout->addWidget(densityText, 1, 2);
00103

```



```

00104     stoppingPowerBox = new QGroupBox(tr("Effective Stopping Cross Section [MeV cm^2/atoms]"));
00105     QVBoxLayout *stoppingPowerLayout = new QVBoxLayout;
00106     QHBoxLayout *stoppingPowerTopLayout = new QHBoxLayout;
00107     QHBoxLayout *equationLayout = new QHBoxLayout;
00108     equationLayout->addWidget(new QLabel(tr("y=")));
00109     equationLayout->addWidget(stoppingPowerEqText);
00110     QHBoxLayout *numParamLayout = new QHBoxLayout;
00111     numParamLayout->addWidget(new QLabel(tr("Number of Parameters:")));
00112     numParamLayout->addWidget(numParametersSpin);
00113     stoppingPowerTopLayout->addLayout(equationLayout);
00114     stoppingPowerTopLayout->addLayout(numParamLayout);
00115     stoppingPowerLayout->addLayout(stoppingPowerTopLayout);
00116     stoppingPowerLayout->addWidget(parametersTable);
00117     stoppingPowerBox->setLayout(stoppingPowerLayout);
00118     stoppingPowerBox->hide();
00119
00120     QGridLayout *qCoefficientCheckBoxLayout = new QGridLayout;
00121     qCoefficientCheckBoxLayout->addWidget(isQCoefficientCheck, 0, 0);
00122     qCoefficientCheckBoxLayout->addItem(new QSpacerItem(1, 20), 0, 1);
00123     qCoefficientCheckBoxLayout->addWidget(new QLabel(tr("Number of Coefficients:")), 0, 2, Qt::AlignRight);
00124     qCoefficientCheckBoxLayout->addWidget(numQCoefficientSpin, 0, 3);
00125     qCoefficientCheckBoxLayout->setColumnStretch(1, 1);
00126
00127     qCoefficientBox = new QGroupBox(tr("Attenuation Coefficients"));
00128     QVBoxLayout *qCoefficientLayout = new QVBoxLayout;
00129     qCoefficientLayout->addWidget(qCoefficientTable);
00130     qCoefficientBox->setLayout(qCoefficientLayout);
00131     qCoefficientBox->hide();
00132
00133     QHBoxLayout *buttonBox = new QHBoxLayout;
00134     buttonBox->addWidget(cancelButton);
00135     buttonBox->addWidget(okButton);
00136
00137     QVBoxLayout *mainLayout = new QVBoxLayout;
00138     mainLayout->addLayout(topLayout);
00139     mainLayout->addLayout(checkBoxLayout);
00140     mainLayout->addWidget(stoppingPowerBox);
00141     mainLayout->addLayout(qCoefficientCheckBoxLayout);
00142     mainLayout->addWidget(qCoefficientBox);
00143     mainLayout->addLayout(buttonBox);
00144
00145     setLayout(mainLayout);
00146
00147     connect(okButton, SIGNAL(clicked()), this, SLOT(accept()));
00148     connect(cancelButton, SIGNAL(clicked()), this, SLOT(reject()));
00149
00150     setWindowTitle(tr("Add an Experimental Effect Line"));
00151 }
00152
00153 void AddTargetIntDialog::createParameterItem(int row, double value) {
00154     QTableWidgetItem *labelItem = new QTableWidgetItem(QString("a%1").arg(row));
00155     QTableWidgetItem *valueItem = new QTableWidgetItem(QString("%1").arg(value));
00156     labelItem->setTextAlignment(Qt::AlignCenter);
00157     labelItem->setFlags(Qt::ItemIsEditable);
00158     valueItem->setTextAlignment(Qt::AlignCenter);
00159     parametersTable->setItem(row, 0, labelItem);
00160     parametersTable->setItem(row, 1, valueItem);
00161     parametersTable->resizeRowsToContents();
00162 }
00163
00164 void AddTargetIntDialog::createQCoefficientItem(int row, double value) {
00165     QTableWidgetItem *labelItem = new QTableWidgetItem(QString("q%1").arg(row));
00166     QTableWidgetItem *valueItem = new QTableWidgetItem(QString("%1").arg(value));
00167     labelItem->setTextAlignment(Qt::AlignCenter);
00168     labelItem->setFlags(Qt::ItemIsEditable);
00169     valueItem->setTextAlignment(Qt::AlignCenter);
00170     qCoefficientTable->setItem(row, 0, labelItem);
00171     qCoefficientTable->setItem(row, 1, valueItem);
00172     qCoefficientTable->resizeRowsToContents();
00173 }
00174
00175 void AddTargetIntDialog::convolutionCheckChanged(bool checked) {
00176     if(checked) {
00177         sigmaText->setEnabled(true);
00178         numPointsSpin->setEnabled(true);
00179     } else {
00180         sigmaText->setEnabled(false);
00181         if(!isTargetIntegrationCheck->isChecked()) numPointsSpin->setEnabled(false);
00182     }
00183 }
00184
00185 void AddTargetIntDialog::targetIntCheckChanged(bool checked) {
00186     if(checked) {
00187         stoppingPowerBox->show();
00188         densityText->setEnabled(true);
00189         numPointsSpin->setEnabled(true);
00190     } else {

```

```

00191     stoppingPowerBox->hide();
00192     densityText->setEnabled(false);
00193     if(!isConvolutionCheck->isChecked()) numPointsSpin->setEnabled(false);
00194     this->adjustSize();
00195 }
00196 }
00197
00198 void AddTargetIntDialog::parameterSpinChanged(int newNumber) {
00199     parametersTable->clearContents();
00200     parametersTable->setRowCount(newNumber);
00201     for(int i=0;i<newNumber;i++) {
00202         if(i<tempParameters.size())
00203             createParameterItem(i,tempParameters.at(i));
00204         else
00205             createParameterItem(i);
00206     }
00207 }
00208
00209 void AddTargetIntDialog::parameterChanged(int row, int column) {
00210     if(column==1) {
00211         while(row>=tempParameters.size()) {
00212             double tempDouble=0.0;
00213             tempParameters.append(tempDouble);
00214         }
00215         tempParameters[row]=parametersTable->item(row,column)->text().toDouble();
00216     }
00217 }
00218
00219 void AddTargetIntDialog::qCoefficientCheckChanged(bool checked) {
00220     if(checked) {
00221         qCoefficientBox->show();
00222         numQCoefficientSpin->setEnabled(true);
00223     } else {
00224         qCoefficientBox->hide();
00225         numQCoefficientSpin->setEnabled(false);
00226         this->adjustSize();
00227     }
00228 }
00229
00230 void AddTargetIntDialog::qCoefficientSpinChanged(int newNumber){
00231     qCoefficientTable->clearContents();
00232     qCoefficientTable->setRowCount(newNumber);
00233     for(int i=0;i<newNumber;i++) {
00234         if(i<tempQCoefficients.size())
00235             createQCoefficientItem(i,tempQCoefficients.at(i));
00236         else
00237             createQCoefficientItem(i);
00238     }
00239 }
00240
00241 void AddTargetIntDialog::qCoefficientChanged(int row, int column) {
00242     if(column==1) {
00243         while(row>=tempQCoefficients.size()) {
00244             double tempDouble=0.0;
00245             tempQCoefficients.append(tempDouble);
00246         }
00247         tempQCoefficients[row]=qCoefficientTable->item(row,column)->text().toDouble();
00248     }
00249 }

```

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.](#)

```
00001 #include <QCheckBox>
00002 #include <QDialog>
00003 #include <QFileDialog>
00004 #include <QCheckBox>
00005 #include <QListView>
00006 #include <QPushButton>
00007 #include <QMessageBox>
00008 #include <QCheckBox>
00009 #include <QImageWriter>
00010 #include <QtPrintSupport/QPrinter>
00011 #include <QtPrintSupport/QPrintDialog>
00012
00013 #include "AZUREPlot.h"
00014 #include "PlotTab.h"
00015 #include "qwt_plot_curve.h"
00016 #include "qwt_plot_intervalcurve.h"
00017 #include "qwt_interval_symbol.h"
00018 #include "qwt_scale_engine.h"
00019 #include "qwt_plot_panner.h"
00020 #include "qwt_plot_renderer.h"
00021 #include <iostream>
00022
00023 QwtText AZUREZoomer::trackerTextF( const QPointF &pos ) const
00024 {
00025     QString text;
00026     switch (rubberBand()) {
00027     case HLineRubberBand:
00028         text.sprintf( "%.4g", pos.y() );
00029         break;
00030     case VLineRubberBand:
00031         text.sprintf( "%.4g", pos.x() );
00032         break;
00033     default:
00034         text.sprintf( "%.4g, %.4g", pos.x(), pos.y() );
00035     }
00036     return QwtText( text );
00037 }
00038
00039
00040 PlotEntry::PlotEntry(int type, int entranceKey, int exitKey, int index, QString filename) :
00041     type_(type), entranceKey_(entranceKey), exitKey_(exitKey), index_(index), filename_(filename),
00042     dataCurve_(NULL), dataErrorCurve_(NULL), fitCurve_(NULL) {
00043 }
00044
00045 PlotEntry::~PlotEntry() {
00046     if(dataCurve_) delete dataCurve_;
00047     if(dataErrorCurve_) delete dataErrorCurve_;
00048     if(fitCurve_) delete fitCurve_;
00049 }
00050
00051 bool PlotEntry::readData() {
00052     QFile file(filename_);
00053     if(!file.open(QIODevice::ReadOnly)) {
00054         return false;
00055     }
00056     QTextStream inStream(&file);
00057     QString line("");
00058     bool previousLineBreak=false;
00059     bool foundBlock=false;
00060     int blockNumber=0;
00061     while(!inStream.atEnd()&&!foundBlock) {
00062         line=inStream.readLine();
00063         if(line.trimmed().isEmpty()) {
00064             if(!previousLineBreak) {
00065                 previousLineBreak=true;
00066                 continue;
00067             }

```

```

00068         if(blockNumber==index_) {
00069             foundBlock=true;
00070             break;
00071         } else {
00072             blockNumber++;
00073             previousLineBreak=false;
00074             points_.clear();
00075             continue;
00076         }
00077     }
00078     if(!inStream.atEnd() && !foundBlock) {
00079         QTextStream in(&line);
00080         PlotPoint newPoint = {0.,0.,0.,0.,0.,0.,0.,0.,0.};
00081         if(type_==0) {
00082             in » newPoint.energy » newPoint.excitationEnergy » newPoint.angle » newPoint.fitCrossSection »
newPoint.fitSFactor
00083             » newPoint.dataCrossSection » newPoint.dataErrorCrossSection » newPoint.dataSFactor
00084             » newPoint.dataErrorSFactor;
00085         } else {
00086             in » newPoint.energy » newPoint.excitationEnergy » newPoint.angle » newPoint.fitCrossSection »
newPoint.fitSFactor;
00087         }
00088         points_.push_back(newPoint);
00089     }
00090 }
00091 inStream.flush();
00092 file.close();
00093 if(!foundBlock) {
00094     return false;
00095 }
00096 hasNegative_=false;
00097 for(QVector<PlotPoint>::const_iterator it=points_.begin();
00098     it<points_.end();it++) {
00099     if(it->fitCrossSection<=0.||
00100        (type_==0&&
00101         (it->dataCrossSection<=0.||
00102          (fabs(it->dataCrossSection)-
00103           fabs(it->dataErrorCrossSection))<=0.))) {
00104         hasNegative_=true;
00105         break;
00106     }
00107 }
00108 return true;
00109 }
00110
00111 void PlotEntry::attach(QwtPlot* plot, int xAxisType, int yAxisType, QwtSymbol::Style style) {
00112     QVector<QPointF> fit(points_.size());
00113     if(type_==0) {
00114         QVector<QPointF> data(points_.size());
00115         QVector<QwtIntervalSample> error(points_.size());
00116
00117         if(xAxisType==0&&yAxisType==0) {
00118             for(int i=0;i<points_.size();i++) {
00119                 data[i]=QPointF(points_[i].energy,points_[i].dataCrossSection);
00120                 fit[i]=QPointF(points_[i].energy,points_[i].fitCrossSection);
00121                 error[i]=QwtIntervalSample(points_[i].energy,
00122                                             QwtInterval(points_[i].dataCrossSection-points_[i].dataErrorCrossSection,
00123                                                         points_[i].dataCrossSection+points_[i].dataErrorCrossSection));
00124             }
00125         } else if(xAxisType==0&&yAxisType==1) {
00126             for(int i=0;i<points_.size();i++) {
00127                 data[i]=QPointF(points_[i].energy,points_[i].dataSFactor);
00128                 fit[i]=QPointF(points_[i].energy,points_[i].fitSFactor);
00129                 error[i]=QwtIntervalSample(points_[i].energy,
00130                                             QwtInterval(points_[i].dataSFactor-points_[i].dataErrorSFactor,
00131                                                         points_[i].dataSFactor+points_[i].dataErrorSFactor));
00132             }
00133         } else if(xAxisType==1&&yAxisType==0) {
00134             for(int i=0;i<points_.size();i++) {
00135                 data[i]=QPointF(points_[i].excitationEnergy,points_[i].dataCrossSection);
00136                 fit[i]=QPointF(points_[i].excitationEnergy,points_[i].fitCrossSection);
00137                 error[i]=QwtIntervalSample(points_[i].excitationEnergy,
00138                                             QwtInterval(points_[i].dataCrossSection-points_[i].dataErrorCrossSection,
00139                                                         points_[i].dataCrossSection+points_[i].dataErrorCrossSection));
00140             }
00141         } else if(xAxisType==1&&yAxisType==1) {
00142             for(int i=0;i<points_.size();i++) {
00143                 data[i]=QPointF(points_[i].excitationEnergy,points_[i].dataSFactor);
00144                 fit[i]=QPointF(points_[i].excitationEnergy,points_[i].fitSFactor);
00145                 error[i]=QwtIntervalSample(points_[i].excitationEnergy,
00146                                             QwtInterval(points_[i].dataSFactor-points_[i].dataErrorSFactor,
00147                                                         points_[i].dataSFactor+points_[i].dataErrorSFactor));
00148             }
00149         } else if(xAxisType==2&&yAxisType==0) {
00150             for(int i=0;i<points_.size();i++) {
00151                 data[i]=QPointF(points_[i].angle,points_[i].dataCrossSection);
00152                 fit[i]=QPointF(points_[i].angle,points_[i].fitCrossSection);

```

```

00153     error[i]=QwtIntervalSample(points_[i].angle,
00154         QwtInterval(points_[i].dataCrossSection-points_[i].dataErrorCrossSection,
00155             points_[i].dataCrossSection+points_[i].dataErrorCrossSection));
00156     }
00157 } else if(xAxisType==2&&yAxisType==1) {
00158     for(int i=0;i<points_.size();i++) {
00159         data[i]=QPointF(points_[i].angle,points_[i].dataSFactor);
00160         fit[i]=QPointF(points_[i].angle,points_[i].fitSFactor);
00161         error[i]=QwtIntervalSample(points_[i].angle,
00162             QwtInterval(points_[i].dataSFactor-points_[i].dataErrorSFactor,
00163                 points_[i].dataSFactor+points_[i].dataErrorSFactor));
00164     }
00165 }
00166
00167 dataCurve_ = new QwtPlotCurve;
00168 dataCurve_>setRenderHint( QwtPlotItem::RenderAntialiased );
00169 dataCurve_>setStyle( QwtPlotCurve::NoCurve );
00170 QwtSymbol *symbol = new QwtSymbol(style);
00171 symbol->setSize( 6 );
00172 symbol->setPen( QPen( Qt::black ) );
00173 symbol->setColor( QColor( Qt::black ) );
00174
00175 dataErrorCurve_ = new QwtPlotIntervalCurve;
00176 dataErrorCurve_>setRenderHint( QwtPlotItem::RenderAntialiased );
00177 dataErrorCurve_>setStyle( QwtPlotIntervalCurve::NoCurve );
00178 QwtIntervalSymbol *errorBar =
00179     new QwtIntervalSymbol( QwtIntervalSymbol::Bar );
00180 errorBar->setWidth( 8 );
00181 errorBar->setPen( QPen( Qt::black ) );
00182
00183 dataCurve_>setSymbol( symbol );
00184 dataCurve_>setSamples(data);
00185
00186 dataErrorCurve_>setSymbol(errorBar);
00187 dataErrorCurve_>setSamples(error);
00188 } else {
00189     if(xAxisType==0&&yAxisType==0) {
00190         for(int i=0;i<points_.size();i++)
00191             fit[i]=QPointF(points_[i].energy,points_[i].fitCrossSection);
00192     } else if(xAxisType==0&&yAxisType==1) {
00193         for(int i=0;i<points_.size();i++)
00194             fit[i]=QPointF(points_[i].energy,points_[i].fitSFactor);
00195     } else if(xAxisType==1&&yAxisType==0) {
00196         for(int i=0;i<points_.size();i++)
00197             fit[i]=QPointF(points_[i].excitationEnergy,points_[i].fitCrossSection);
00198     } else if(xAxisType==1&&yAxisType==1) {
00199         for(int i=0;i<points_.size();i++)
00200             fit[i]=QPointF(points_[i].excitationEnergy,points_[i].fitSFactor);
00201     } else if(xAxisType==2&&yAxisType==0) {
00202         for(int i=0;i<points_.size();i++)
00203             fit[i]=QPointF(points_[i].angle,points_[i].fitCrossSection);
00204     } else if(xAxisType==2&&yAxisType==1) {
00205         for(int i=0;i<points_.size();i++)
00206             fit[i]=QPointF(points_[i].angle,points_[i].fitSFactor);
00207     }
00208 }
00209
00210 fitCurve_ = new QwtPlotCurve;
00211 fitCurve_>setRenderHint( QwtPlotItem::RenderAntialiased );
00212 fitCurve_>setStyle( QwtPlotCurve::Lines );
00213 fitCurve_>setPen( QPen( Qt::red , 2 ) );
00214
00215 fitCurve_>setSymbol(new QwtSymbol(QwtSymbol::NoSymbol));
00216 fitCurve_>setSamples(fit);
00217
00218 if(dataCurve_) dataCurve_>attach(plot);
00219 if(dataErrorCurve_) dataErrorCurve_>attach(plot);
00220 if(fitCurve_) fitCurve_>attach(plot);
00221 }
00222
00223 void PlotEntry::detach() {
00224     if(dataCurve_) dataCurve_>detach();
00225     if(dataErrorCurve_) dataErrorCurve_>detach();
00226     if(fitCurve_) fitCurve_>detach();
00227 }
00228
00229 AZUREPlot::AZUREPlot(PlotTab* plotTab,QWidget* parent) :
00230     containingTab(plotTab), QwtPlot(parent) {
00231     setCanvasBackground(QColor(Qt::white));
00232     setAutoReplot(true);
00233
00234     zoomer = new AZUREZoomer( canvas() );
00235     zoomer->setRubberBandPen( QColor( Qt::black ) );
00236     zoomer->setTrackerPen( QColor( Qt::black ) );
00237     zoomer->setMousePattern( QwtEventPattern::MouseSelect2,
00238         Qt::RightButton, Qt::ControlModifier );
00239     zoomer->setMousePattern( QwtEventPattern::MouseSelect3,

```

```

00240         Qt::RightButton );
00241
00242     QwtPlotPanner *panner = new QwtPlotPanner( canvas() );
00243     panner->setMouseButton( Qt::MidButton );
00244
00245 }
00246
00247 void AZUREPlot::setXAxisLog(bool set) {
00248     setAxisAutoScale(QwtPlot::xBottom,true);
00249     setAxisAutoScale(QwtPlot::yLeft,true);
00250     if(set) setAxisScaleEngine(QwtPlot::xBottom,new QwtLogScaleEngine);
00251     else setAxisScaleEngine(QwtPlot::xBottom,new QwtLinearScaleEngine);
00252     zoomer->setZoomBase(false);
00253 };
00254
00255 void AZUREPlot::setYAxisLog(bool set) {
00256     setAxisAutoScale(QwtPlot::xBottom,true);
00257     setAxisAutoScale(QwtPlot::yLeft,true);
00258     if(set) {
00259         QwtLogScaleEngine* scaleEngine = new QwtLogScaleEngine;
00260         scaleEngine->setMargins(0.5,0.5);
00261         setAxisScaleEngine(QwtPlot::yLeft,scaleEngine);
00262     } else setAxisScaleEngine(QwtPlot::yLeft,new QwtLinearScaleEngine);
00263     zoomer->setZoomBase(false);
00264 };
00265
00266 void AZUREPlot::setXAxisType(unsigned int type) {
00267     QwtText text;
00268     if(type==0) text=QwtText(QString("Center of Mass Energy [MeV]"));
00269     else if(type==1) text=QwtText(QString("Excitation Energy [MeV]"));
00270     else text=QwtText(QString("Center of Mass Angle [degrees]"));
00271     setAxisTitle(QwtPlot::xBottom,text);
00272
00273     xAxisType=type;
00274     update();
00275 }
00276
00277 void AZUREPlot::setYAxisType(unsigned int type) {
00278     QwtText text = (type==0) ? QwtText(QString("Cross Section [b]")) : QwtText(QString("S-Factor [MeV
00279     b]"));
00280     setAxisTitle(QwtPlot::yLeft,text);
00281     yAxisType=type;
00282     update();
00283 }
00284
00285 void AZUREPlot::draw(QList<PlotEntry*> newEntries) {
00286     clearEntries();
00287
00288     int numDataEntries=0;
00289     bool hasNegative=false;
00290     for(int i = 0; i<newEntries.size(); i++) {
00291         if(newEntries[i]->readData()) {
00292             QwtSymbol::Style style = (newEntries[i]->type()==0) ? (QwtSymbol::Style)
00293             numDataEntries++ : QwtSymbol::NoSymbol;
00294             newEntries[i]->attach(this,xAxisType,yAxisType,style);
00295             entries.push_back(newEntries[i]);
00296             if(newEntries[i]->hasNegative_) hasNegative=true;
00297         } else delete newEntries[i];
00298     }
00299     setAxisAutoScale(QwtPlot::xBottom,true);
00300     setAxisAutoScale(QwtPlot::yLeft,true);
00301     replot();
00302     zoomer->setZoomBase(false);
00303     if(hasNegative) {
00304         containingTab->yAxisIsLogCheck->setChecked(false);
00305         containingTab->yAxisIsLogCheck->setEnabled(false);
00306         QMessageBox::information(this,
00307             tr("Negative or Zero Values"),
00308             tr("Negative or zero values were detected in a dataset. "
00309             "Log plotting is not available."));
00310     }
00311 }
00312
00313 void AZUREPlot::update() {
00314     setAxisAutoScale(QwtPlot::xBottom,true);
00315     setAxisAutoScale(QwtPlot::yLeft,true);
00316     int numDataEntries=0;
00317     for(int i = 0; i<entries.size(); i++) {
00318         entries[i]->detach();
00319         QwtSymbol::Style style = (entries[i]->type()==0) ? (QwtSymbol::Style) numDataEntries++ :
00320         QwtSymbol::NoSymbol;
00321         entries[i]->attach(this,xAxisType,yAxisType,style);
00322     }
00323     replot();
00324     zoomer->setZoomBase(false);
00325 }

```

```

00325
00326 void AZUREPlot::exportPlot()
00327 {
00328     #ifndef QT_NO_PRINTER
00329         QString fileName = "AZUREPlot.pdf";
00330     #else
00331         QString fileName = "AZUREPlot.png";
00332     #endif
00333
00334     #ifndef QT_NO_FILEDIALOG
00335         const QList<QByteArray> imageFormats =
00336             QImageWriter::supportedImageFormats();
00337
00338         QStringList filter;
00339         filter += "PDF Documents (*.pdf)";
00340     #ifndef QWT_NO_SVG
00341         filter += "SVG Documents (*.svg)";
00342     #endif
00343         filter += "Postscript Documents (*.ps)";
00344
00345         if (imageFormats.size() > 0) {
00346             QString imageFilter("Images (");
00347             for (int i=0; i<imageFormats.size(); i++) {
00348                 if (i>0) imageFilter += " ";
00349                 imageFilter += "*.";
00350                 imageFilter += imageFormats[i];
00351             }
00352             imageFilter += ")";
00353
00354             filter += imageFilter;
00355         }
00356         fileName = QFileDialog::getSaveFileName(
00357             this, "Export File Name", fileName,
00358             filter.join(";"), NULL, QFileDialog::DontConfirmOverwrite);
00359     #endif
00360         if (!fileName.isEmpty()) {
00361             QwtPlotRenderer renderer;
00362             renderer.setDiscardFlag(QwtPlotRenderer::DiscardBackground, true);
00363             renderer.renderDocument(this, fileName, QSizeF(300, 200), 85);
00364         }
00365     }
00366
00367 void AZUREPlot::print()
00368 {
00369     QPrinter printer(QPrinter::HighResolution);
00370     QString docName("AZUREPlot");
00371     printer.setDocName(docName);
00372
00373     printer.setCreator("AZURE2");
00374     printer.setOrientation(QPrinter::Landscape);
00375
00376     QPrintDialog dialog(&printer);
00377     if (dialog.exec()) {
00378         QwtPlotRenderer renderer;
00379         renderer.renderTo(this, printer);
00380     }
00381 }
00382
00383 void AZUREPlot::clearEntries() {
00384     for (int i = 0; i < entries.size(); i++) {
00385         entries[i] -> detach();
00386         delete entries[i];
00387     }
00388     if (entries.size() > 0) entries.clear();
00389     containingTab->yAxisIsLogCheck->setEnabled(true);
00390     update();
00391 }

```

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.](#)

```
00001 #include <QTextEdit>
00002 #include <QMessageBox>
00003 #include <QScrollBar>
00004 #include <QFileDialog>
00005 #include <QMenu>
00006 #include <QMenuBar>
00007 #include <QRadioButton>
00008 #include <QAction>
00009 #include <QActionGroup>
00010 #include <QSettings>
00011 #include <QTextStream>
00012 #include <QDesktopServices>
00013
00014 #include "AZURESetup.h"
00015 #include "EditChecksDialog.h"
00016 #include "EditDirsDialog.h"
00017 #include "RunTab.h"
00018 #include "Config.h"
00019 #include "EditOptionsDialog.h"
00020 #include "AZUREMainThread.h"
00021 #include "AboutAZURE2Dialog.h"
00022
00023 #ifdef USE_QWT
00024 #include "PlotTab.h"
00025 #endif
00026 #include <iostream>
00027
00028 struct SegPairs {int firstPair; int secondPair;};
00029 extern bool readSegmentFile(const Config& configure, std::vector<SegPairs>& segPairs);
00030 extern bool checkExternalCapture(Config& configure, const std::vector<SegPairs>& segPairs);
00031 extern void startMessage(const Config& configure);
00032 extern void exitMessage(const Config& configure);
00033
00034
00035 AZURESetup::AZURESetup() : config(std::cout) {
00036     setMinimumSize(1000,640);
00037
00038     tabWidget=new QTabWidget();
00039
00040     pairsTab=new PairsTab;
00041
00042     levelsTab = new LevelsTab;
00043     levelsTab->setPairsModel(pairsTab->getPairsModel());
00044     connect (pairsTab,SIGNAL(pairAdded(int)),levelsTab,SLOT(updateChannelsPairAddedEdited()));
```

```

00045     connect (pairsTab, SIGNAL (pairEdited (int)), levelsTab, SLOT (updateChannelsPairAddedEdited ()));
00046     connect (pairsTab, SIGNAL (pairRemoved (int)), levelsTab, SLOT (updateChannelsPairRemoved (int)));
00047
00048     connect (levelsTab, SIGNAL (readNewPair (PairsData, int, bool)), pairsTab, SLOT (addPair (PairsData, int, bool)));
00049
00050     connect (levelsTab, SIGNAL (readExistingPair (PairsData, int, bool)), pairsTab, SLOT (editPair (PairsData, int, bool)));
00051
00052     segmentsTab = new SegmentsTab;
00053     segmentsTab->setPairsModel (pairsTab->getPairsModel ());
00054
00055     targetIntTab=new TargetIntTab;
00056
00057     runTab = new RunTab ();
00058     connect (runTab->calcButton, SIGNAL (clicked ()), this, SLOT (SaveAndRun ()));
00059
00060 #ifdef USE_QWT
00061     plotTab = new
00062     PlotTab (config, segmentsTab->getSegmentsDataModel (), segmentsTab->getSegmentsTestModel ());
00063 #endif
00064
00065     tabWidget->addTab (pairsTab, tr ("%Particle Pairs"));
00066     tabWidget->addTab (levelsTab, tr ("%Levels and Channels"));
00067     tabWidget->addTab (segmentsTab, tr ("%Segments"));
00068     tabWidget->addTab (targetIntTab, tr ("%Experimental Effects"));
00069     tabWidget->addTab (runTab, tr ("%Calculate"));
00070
00071 #ifdef USE_QWT
00072     tabWidget->addTab (plotTab, tr ("Pl&ot"));
00073 #endif
00074
00075     setCentralWidget (tabWidget);
00076
00077     createActions ();
00078     createMenus ();
00079
00080     setWindowTitle (tr ("AZURE2 -- untitled"));
00081 }
00082
00083 Config& AZURESetup::GetConfig () {
00084     return config;
00085 }
00086
00087 void AZURESetup::createActions () {
00088     aboutAction = new QAction (tr ("%About AZURE2..."), this);
00089     connect (aboutAction, SIGNAL (triggered ()), this, SLOT (showAbout ()));
00090
00091     resetAction = new QAction (tr ("%New Project"), this);
00092     resetAction->setShortcuts (QKeySequence::New);
00093     connect (resetAction, SIGNAL (triggered ()), this, SLOT (reset ()));
00094
00095     quitAction = new QAction (tr ("%Quit"), this);
00096     quitAction->setShortcuts (QKeySequence::Quit);
00097     connect (quitAction, SIGNAL (triggered ()), this, SLOT (close ()));
00098
00099     openAction = new QAction (tr ("%Open..."), this);
00100     openAction->setShortcuts (QKeySequence::Open);
00101     connect (openAction, SIGNAL (triggered ()), this, SLOT (open ()));
00102
00103     saveAction = new QAction (tr ("%Save"), this);
00104     saveAction->setShortcuts (QKeySequence::Save);
00105     connect (saveAction, SIGNAL (triggered ()), this, SLOT (save ()));
00106
00107     saveAsAction = new QAction (tr ("%Save &As..."), this);
00108     saveAsAction->setShortcuts (QKeySequence::SaveAs);
00109     connect (saveAsAction, SIGNAL (triggered ()), this, SLOT (saveAs ()));
00110
00111     for (int i=0; i<numRecent; i++) {
00112         recentFileActions[i] = new QAction (this);
00113         recentFileActions[i] -> setVisible (false);
00114         connect (recentFileActions[i], SIGNAL (triggered ()), this, SLOT (openRecent ()));
00115     }
00116
00117     clearRecentAction = new QAction (tr ("%Clear"), this);
00118     clearRecentAction->setVisible (false);
00119     connect (clearRecentAction, SIGNAL (triggered ()), this, SLOT (clearRecent ()));
00120
00121     copyAction = new QAction (tr ("%Copy"), this);
00122     copyAction->setShortcuts (QKeySequence::Copy);
00123
00124     matrixActionGroup = new QActionGroup (this);
00125     aMatrixAction = new QAction (tr ("%A-Matrix"), this);
00126     aMatrixAction->setCheckable (true);
00127     matrixActionGroup->addAction (aMatrixAction);
00128     rMatrixAction = new QAction (tr ("%R-Matrix"), this);
00129     rMatrixAction->setCheckable (true);
00130     matrixActionGroup->addAction (rMatrixAction);
00131     aMatrixAction->setChecked (true);
00132     connect (matrixActionGroup, SIGNAL (triggered (QAction*)), this, SLOT (matrixChanged (QAction*)));

```

```

00129
00130 editChecksAction = new QAction(tr("&Checks..."),this);
00131 connect(editChecksAction,SIGNAL(triggered()),this,SLOT(editChecks()));
00132
00133 editDirsAction = new QAction(tr("&Directories..."),this);
00134 connect(editDirsAction,SIGNAL(triggered()),this,SLOT(editDirs()));
00135
00136 editOptionsAction = new QAction(tr("&Runtime Options..."),this);
00137 connect(editOptionsAction,SIGNAL(triggered()),this,SLOT(editOptions()));
00138
00139 showTabInfoAction = new QAction(tr("Show Documentation For Current Tab"),this);
00140 showTabInfoAction->setShortcut(QKeySequence(Qt::CTRL + Qt::Key_D));
00141 connect(showTabInfoAction,SIGNAL(triggered()),this,SLOT(showTabInfo()));
00142
00143 openAZURESiteAction = new QAction(tr("Open AZURE Website..."),this);
00144 connect(openAZURESiteAction,SIGNAL(triggered()),this,SLOT(openWebsite()));
00145 }
00146
00147 void AZURESetup::createMenus() {
00148     fileMenu = menuBar()->addMenu(tr("&File"));
00149     fileMenu->addAction(aboutAction);
00150     fileMenu->addSeparator();
00151     fileMenu->addAction(resetAction);
00152     fileMenu->addAction(openAction);
00153     recentFileMenu = fileMenu->addMenu(tr("Open &Recent..."));
00154     for(int i = 0; i < numRecent; i++) recentFileMenu->addAction(recentFileActions[i]);
00155     recentSeparator = recentFileMenu->addSeparator();
00156     recentFileMenu->addAction(clearRecentAction);
00157     updateRecent();
00158     fileMenu->addAction(saveAction);
00159     fileMenu->addAction(saveAsAction);
00160     fileMenu->addSeparator();
00161     fileMenu->addAction(quitAction);
00162
00163     configMenu = menuBar()->addMenu(tr("Co&nfigure"));
00164     formalismMenu = configMenu->addMenu(tr("&Formalism"));
00165     formalismMenu->addAction(aMatrixAction);
00166     formalismMenu->addAction(rMatrixAction);
00167     configMenu->addAction(editChecksAction);
00168     configMenu->addAction(editDirsAction);
00169     configMenu->addAction(editOptionsAction);
00170
00171     helpMenu = menuBar()->addMenu(tr("&Documentation"));
00172     helpMenu->addAction(showTabInfoAction);
00173     helpMenu->addAction(openAZURESiteAction);
00174 }
00175
00176 void AZURESetup::updateRecent() {
00177     QSettings settings;
00178     QStringList files = settings.value("recentFileList").toStringList();
00179
00180     int numFiles = qMin(files.size(),(int)numRecent);
00181
00182     for(int i = 0; i<numFiles; i++) {
00183         recentFileActions[i]->setText(tr("%1 %2").arg(i+1).arg(QFileInfo(files[i]).fileName()));
00184         recentFileActions[i]->setData(files[i]);
00185         recentFileActions[i]->setVisible(true);
00186     }
00187
00188     for(int i = numFiles; i<numRecent; i++) recentFileActions[i]->setVisible(false);
00189     recentSeparator->setVisible(numFiles>0);
00190     clearRecentAction->setVisible(numFiles>0);
00191 }
00192
00193 void AZURESetup::open() {
00194     QString filename = QFileDialog::getOpenFileName(this);
00195     if(!filename.isEmpty()) {
00196         if(!this->readFile(filename)) {
00197             reset();
00198             QMessageBox::information(this,
00199                                     tr("Can't Access File"),
00200                                     tr("An error was encountered while reading the file.));
00201         }
00202     }
00203 }
00204
00205 void AZURESetup::open(QString filename) {
00206     if(!filename.isEmpty()) {
00207         if(!this->readFile(filename)) {
00208             reset();
00209             QMessageBox::information(this,
00210                                     tr("Can't Access File"),
00211                                     tr("An error was encountered while reading the file.));
00212         }
00213     }
00214 }
00215

```

```

00216 void AZURESetup::openRecent() {
00217     QString filename = qobject_cast<QAction*>(sender())->data().toString();
00218     open(filename);
00219 }
00220
00221 void AZURESetup::clearRecent() {
00222     QSettings settings;
00223     QStringList files = settings.value("recentFileList").toStringList();
00224     files.clear();
00225     settings.setValue("recentFileList", files);
00226     updateRecent();
00227 }
00228
00229 bool AZURESetup::readFile(QString filename) {
00230     QFile file(filename);
00231     if(!file.open(QIODevice::ReadOnly)) return false;
00232     QFileInfo info(file);
00233     QString directory=info.absolutePath();
00234
00235     reset();
00236
00237     QTextStream in(&file);
00238     QString line("");
00239
00240     while(line.trimmed()!=QString("<config>")&&!in.atEnd()) line = in.readLine();
00241     if(in.atEnd()) return false;
00242     if(!this->readConfig(in)) return false;
00243
00244     line=QString("");
00245     while(line.trimmed()!=QString("<levels>")&&!in.atEnd()) line = in.readLine();
00246     if(in.atEnd()) return false;
00247     if(!levelsTab->readNuclearFile(in)) return false;
00248
00249     line=QString("");
00250     while(line.trimmed()!=QString("<segmentsData>")&&!in.atEnd()) line = in.readLine();
00251     if(in.atEnd()) return false;
00252     if(!segmentsTab->readSegDataFile(in)) return false;
00253
00254     line=QString("");
00255     while(line.trimmed()!=QString("<segmentsTest>")&&!in.atEnd()) line = in.readLine();
00256     if(in.atEnd()) return false;
00257     if(!segmentsTab->readSegTestFile(in)) return false;
00258
00259     line=QString("");
00260     while(line.trimmed()!=QString("<targetInt>")&&!in.atEnd()) line = in.readLine();
00261     if(in.atEnd()) return false;
00262     if(!targetIntTab->readFile(in)) return false;
00263
00264     line=QString("");
00265     while(line.trimmed()!=QString("<lastRun>")&&!in.atEnd()) line = in.readLine();
00266     if(in.atEnd())
00267         if(!this->readLastRun(in)) return false;
00268
00269     file.close();
00270
00271     QFile file2(filename);
00272     if(!file2.open(QIODevice::ReadOnly)) return false;
00273     QTextStream in2(&file2);
00274     line=QString("");
00275     while(line.trimmed()!=QString("<externalCapture>")&&!in2.atEnd()) line = in2.readLine();
00276     if(!in2.atEnd()) {
00277         if(!pairsTab->parseOldECSection(in2)) return false;
00278     }
00279     file2.close();
00280
00281     GetConfig().configfile=QDir::fromNativeSeparators(info.absoluteFilePath()).toStdString();
00282     setWindowTitle(QString("AZURE2 -- %1").arg(QString::fromStdString(GetConfig().configfile)));
00283     QDir::setCurrent(directory);
00284
00285     QSettings settings;
00286     QStringList files = settings.value("recentFileList").toStringList();
00287     QString fullFileName = QDir::fromNativeSeparators(info.absoluteFilePath());
00288     files.removeAll(fullFileName);
00289     files.prepend(fullFileName);
00290     while(files.size()>numRecent) files.removeLast();
00291
00292     settings.setValue("recentFileList", files);
00293     updateRecent();
00294
00295     return true;
00296 }
00297
00298 bool AZURESetup::readLastRun(QTextStream& inStream) {
00299     unsigned int paramMask;
00300     unsigned int useTempFile;
00301     unsigned int rateEntrancePair;
00302     unsigned int rateExitPair;

```

```

00303   QString paramFile;
00304   QString integralsFile;
00305   QString temperatureFile;
00306   QString dummyString;
00307   double minTemp;
00308   double maxTemp;
00309   double tempStep;
00310
00311   inStream » paramMask; dummyString=inStream.readLine();
00312   dummyString=inStream.readLine(); paramFile=dummyString.trimmed();
00313   dummyString=inStream.readLine(); integralsFile=dummyString.trimmed();
00314   inStream » rateEntrancePair » rateExitPair; dummyString=inStream.readLine();
00315   inStream » useTempFile; dummyString=inStream.readLine(); temperatureFile=dummyString.trimmed();
00316   inStream » minTemp » maxTemp » tempStep;
00317
00318   QString line("");
00319   while (line.trimmed() != QString("</lastRun>") && !inStream.atEnd())
00320       line=inStream.readLine();
00321   if (line.trimmed() != QString("</lastRun>")) return false;
00322
00323   if (paramMask & Config::CALCULATE_WITH_DATA) {
00324       if (paramMask & Config::PERFORM_FIT) runTab->calcType->setCurrentIndex(1);
00325       else if (paramMask & Config::PERFORM_ERROR_ANALYSIS) runTab->calcType->setCurrentIndex(3);
00326       else runTab->calcType->setCurrentIndex(0);
00327   } else {
00328       if (paramMask & Config::CALCULATE_REACTION_RATE) runTab->calcType->setCurrentIndex(4);
00329       else runTab->calcType->setCurrentIndex(2);
00330   }
00331
00332   if (paramMask & Config::USE_GSL_COULOMB_FUNC) GetConfig().paramMask |= Config::USE_GSL_COULOMB_FUNC;
00333   else GetConfig().paramMask &= ~Config::USE_GSL_COULOMB_FUNC;
00334
00335   if (paramMask & Config::USE_BRUNE_FORMALISM) GetConfig().paramMask |= Config::USE_BRUNE_FORMALISM;
00336   else GetConfig().paramMask &= ~Config::USE_BRUNE_FORMALISM;
00337
00338   if (paramMask & Config::IGNORE_ZERO_WIDTHS) GetConfig().paramMask |= Config::IGNORE_ZERO_WIDTHS;
00339   else GetConfig().paramMask &= ~Config::IGNORE_ZERO_WIDTHS;
00340
00341   if (paramMask & Config::USE_RMC_FORMALISM) GetConfig().paramMask |= Config::USE_RMC_FORMALISM;
00342   else GetConfig().paramMask &= ~Config::USE_RMC_FORMALISM;
00343
00344   if (paramMask & Config::TRANSFORM_PARAMETERS) GetConfig().paramMask |= Config::TRANSFORM_PARAMETERS;
00345   else GetConfig().paramMask &= ~Config::TRANSFORM_PARAMETERS;
00346
00347   if (paramMask & Config::USE_LONGWAVELENGTH_APPROX) GetConfig().paramMask |=
Config::USE_LONGWAVELENGTH_APPROX;
00348   else GetConfig().paramMask &= ~Config::USE_LONGWAVELENGTH_APPROX;
00349
00350   if (rateEntrancePair != 0) runTab->rateEntranceKey->setText (QString("%1").arg (rateEntrancePair));
00351   if (rateExitPair != 0) runTab->rateExitKey->setText (QString("%1").arg (rateExitPair));
00352
00353   if (minTemp != -1.) runTab->minTempText->setText (QString("%1").arg (minTemp));
00354   if (maxTemp != -1.) runTab->maxTempText->setText (QString("%1").arg (maxTemp));
00355   if (tempStep != -1.) runTab->tempStepText->setText (QString("%1").arg (tempStep));
00356
00357   if (useTempFile == 1) runTab->fileTempButton->setChecked (true);
00358   else runTab->gridTempButton->setChecked (true);
00359   if (temperatureFile[0] == QChar('\"')) temperatureFile.remove (0, 1);
00360   if (temperatureFile[temperatureFile.length() - 1] == QChar('\"'))
temperatureFile.remove (temperatureFile.length() - 1, 1);
00361   if (!temperatureFile.trimmed().isEmpty()) runTab->fileTempText->setText (temperatureFile.trimmed());
00362
00363   if (paramMask & Config::USE_PREVIOUS_PARAMETERS) runTab->oldParamFileButton->setChecked (true);
00364   else runTab->newParamFileButton->setChecked (true);
00365   if (paramFile[0] == QChar('\"')) paramFile.remove (0, 1);
00366   if (paramFile[paramFile.length() - 1] == QChar('\"')) paramFile.remove (paramFile.length() - 1, 1);
00367   if (!paramFile.trimmed().isEmpty()) runTab->paramFileText->setText (paramFile.trimmed());
00368
00369   if (paramMask & Config::USE_PREVIOUS_INTEGRALS) runTab->oldIntegralsFileButton->setChecked (true);
00370   else runTab->newIntegralsFileButton->setChecked (true);
00371   if (integralsFile[0] == QChar('\"')) integralsFile.remove (0, 1);
00372   if (integralsFile[integralsFile.length() - 1] == QChar('\"'))
integralsFile.remove (integralsFile.length() - 1, 1);
00373   if (!integralsFile.trimmed().isEmpty()) runTab->integralsFileText->setText (integralsFile.trimmed());
00374
00375   return true;
00376 }
00377
00378 bool AZURESetup::readConfig (QTextStream& inStream) {
00379
00380   QString isAMatrix;
00381   QString outputDirectory;
00382   QString checksDirectory;
00383   QString compoundCheck;
00384   QString boundaryCheck;
00385   QString dataCheck;
00386   QString lMatrixCheck;

```

```

00387   QString legendreCheck;
00388   QString coulAmpCheck;
00389   QString pathwaysCheck;
00390   QString angDistsCheck;
00391   QString dummyString;
00392
00393   inStream » isAMatrix; dummyString=inStream.readLine();
00394   dummyString=inStream.readLine();
00395   int poundSignPos = dummyString.lastIndexOf('#');
00396   if(poundSignPos==-1) outputDirectory=dummyString.trimmed();
00397   else outputDirectory=dummyString.left(poundSignPos).trimmed();
00398   dummyString=inStream.readLine();
00399   poundSignPos = dummyString.lastIndexOf('#');
00400   if(poundSignPos==-1) checksDirectory=dummyString.trimmed();
00401   else checksDirectory=dummyString.left(poundSignPos).trimmed();
00402   inStream » compoundCheck; dummyString=inStream.readLine();
00403   inStream » boundaryCheck; dummyString=inStream.readLine();
00404   inStream » dataCheck; dummyString=inStream.readLine();
00405   inStream » lMatrixCheck; dummyString=inStream.readLine();
00406   inStream » legendreCheck; dummyString=inStream.readLine();
00407   inStream » coulAmpCheck; dummyString=inStream.readLine();
00408   inStream » pathwaysCheck; dummyString=inStream.readLine();
00409   inStream » angDistsCheck; dummyString=inStream.readLine();
00410
00411   QString line("");
00412   while(line.trimmed() != QString("</config>") && !inStream.atEnd())
00413       line=inStream.readLine();
00414   if(line.trimmed() != QString("</config>")) return false;
00415
00416   if(isAMatrix=="false") rMatrixAction->activate(QAction::Trigger);
00417   else aMatrixAction->activate(QAction::Trigger);
00418   GetConfig().outputDir=outputDirectory.toString();
00419   GetConfig().checkDir=checksDirectory.toString();
00420   if(compoundCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_COMPOUND_NUCLEUS;
00421   else if(compoundCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_COMPOUND_NUCLEUS;
00422   if(boundaryCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_BOUNDARY_CONDITIONS;
00423   else if(boundaryCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_BOUNDARY_CONDITIONS;
00424   if(dataCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_DATA;
00425   else if(dataCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_DATA;
00426   if(lMatrixCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_ENERGY_DEP;
00427   else if(lMatrixCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_ENERGY_DEP;
00428   if(legendreCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_LEGENDRE;
00429   else if(legendreCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_LEGENDRE;
00430   if(coulAmpCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_COUL_AMPLITUDES;
00431   else if(coulAmpCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_COUL_AMPLITUDES;
00432   if(pathwaysCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_PATHWAYS;
00433   else if(pathwaysCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_PATHWAYS;
00434   if(angDistsCheck=="file") GetConfig().fileCheckMask |= Config::CHECK_ANGULAR_DIST;
00435   else if(angDistsCheck=="screen") GetConfig().screenCheckMask |= Config::CHECK_ANGULAR_DIST;
00436
00437   return true;
00438 }
00439
00440 void AZURESetup::save() {
00441     if(!GetConfig().configfile.empty()) {
00442         if(!this->writeFile(QString::fromStdString(GetConfig().configfile))
00443             QMessageBox::information(this,
00444                 tr("Can't Access File"),
00445                 tr("An error occurred while writing the file."));
00446     } else saveAs();
00447 }
00448
00449 void AZURESetup::saveAs() {
00450     QString filename = QFileDialog::getSaveFileName(this);
00451     if(!filename.isEmpty()) {
00452         if(!this->writeFile(filename))
00453             QMessageBox::information(this,
00454                 tr("Can't Access File"),
00455                 tr("An error occurred while writing the file."));
00456         else {
00457             QSettings settings;
00458             QStringList files = settings.value("recentFileList").toStringList();
00459             QFile file(filename);
00460             QFileInfo info(file);
00461             QString fullFileName = QDir::fromNativeSeparators(info.absoluteFilePath());
00462             files.removeAll(fullFileName);
00463             files.prepend(fullFileName);
00464             while(files.size()>numRecent) files.removeLast();
00465             settings.setValue("recentFileList", files);
00466             updateRecent();
00467         }
00468     }
00469 }
00470
00471 bool AZURESetup::writeFile(QString filename) {
00472     QFile file(filename);
00473     if(!file.open(QIODevice::WriteOnly)) return false;

```

```

00474     QFileInfo info(file);
00475     QString directory=info.absolutePath();
00476
00477     QTextStream out(&file);
00478     out << "<config>" << endl;
00479     if(!this->writeConfig(out,directory)) return false;
00480     out << "</config>" << endl;
00481
00482     out << "<levels>" << endl;
00483     if(!levelsTab->writeNuclearFile(out)) return false;
00484     out << "</levels>" << endl;
00485
00486     out << "<segmentsData>" << endl;
00487     if(!segmentsTab->writeSegDataFile(out)) return false;
00488     out << "</segmentsData>" << endl;
00489
00490     out << "<segmentsTest>" << endl;
00491     if(!segmentsTab->writeSegTestFile(out)) return false;
00492     out << "</segmentsTest>" << endl;
00493
00494     out << "<targetInt>" << endl;
00495     if(!targetIntTab->writeFile(out)) return false;
00496     out << "</targetInt>" << endl;
00497
00498     out << "<lastRun>" << endl;
00499     if(!writeLastRun(out)) return false;
00500     out << "</lastRun>" << endl;
00501
00502     GetConfig().configfile=QDir::fromNativeSeparators(info.absoluteFilePath()).toStdString();
00503     setWindowTitle(QString("AZURE2 -- %1").arg(QString::fromStdString(GetConfig().configfile)));
00504     QDir::setCurrent(directory);
00505
00506     out.flush();
00507     file.close();
00508     return true;
00509 }
00510
00511 bool AZURESetup::writeConfig(QTextStream& outStream, QString directory) {
00512     QString isAMatrix;
00513     QString outputDirectory;
00514     QString checksDirectory;
00515     QString compoundCheck;
00516     QString boundaryCheck;
00517     QString dataCheck;
00518     QString lMatrixCheck;
00519     QString legendreCheck;
00520     QString coulAmpCheck;
00521     QString pathwaysCheck;
00522     QString angDistsCheck;
00523
00524     if(GetConfig().paramMask & Config::USE_AMATRIX) isAMatrix="true";
00525     else isAMatrix="false";
00526     bool emptyCheckDir=false;
00527     bool emptyOutputDir=false;
00528     if(!GetConfig().outputdir.empty())
00529         outputDirectory=QString::fromStdString(GetConfig().outputdir);
00530     else {
00531         outputDirectory=QDir::fromNativeSeparators(directory)+'//';
00532         GetConfig().outputdir=outputDirectory.toStdString();
00533         emptyOutputDir=true;
00534     }
00535     if(!GetConfig().checkdir.empty())
00536         checksDirectory=QString::fromStdString(GetConfig().checkdir);
00537     else {
00538         checksDirectory=QDir::fromNativeSeparators(directory)+'//';
00539         GetConfig().checkdir=checksDirectory.toStdString();
00540         emptyCheckDir=true;
00541     }
00542     if(emptyCheckDir&&emptyOutputDir) {
00543         QMessageBox::information(this,tr("Unspecified Directories"),
00544             QString("The output and checks directories are unspecified. "
00545                 "They will be set to %1.").arg(outputDirectory.trimmed()));
00546     } else if(emptyCheckDir) {
00547         QMessageBox::information(this,tr("Unspecified Directory"),
00548             QString("The checks directory is unspecified. "
00549                 "It will be set to %1.").arg(checksDirectory.trimmed()));
00550     } else if(emptyOutputDir) {
00551         QMessageBox::information(this,tr("Unspecified Directory"),
00552             QString("The output directory is unspecified. "
00553                 "It will be set to %1.").arg(outputDirectory.trimmed()));
00554     }
00555     if(GetConfig().fileCheckMask & Config::CHECK_COMPOUND_NUCLEUS) compoundCheck="file";
00556     else if(GetConfig().screenCheckMask & Config::CHECK_COMPOUND_NUCLEUS) compoundCheck="screen";
00557     else compoundCheck="none";
00558     if(GetConfig().fileCheckMask & Config::CHECK_BOUNDARY_CONDITIONS) boundaryCheck="file";
00559     else if(GetConfig().screenCheckMask & Config::CHECK_BOUNDARY_CONDITIONS) boundaryCheck="screen";
00560     else boundaryCheck="none";

```



```
00561     if(GetConfig().fileCheckMask & Config::CHECK_DATA) dataCheck="file";
00562     else if(GetConfig().screenCheckMask & Config::CHECK_DATA) dataCheck="screen";
00563     else dataCheck="none";
00564     if(GetConfig().fileCheckMask & Config::CHECK_ENERGY_DEP) lMatrixCheck="file";
00565     else if(GetConfig().screenCheckMask & Config::CHECK_ENERGY_DEP) lMatrixCheck="screen";
00566     else lMatrixCheck="none";
00567     if(GetConfig().fileCheckMask & Config::CHECK_LEGENDRE) legendreCheck="file";
00568     else if(GetConfig().screenCheckMask & Config::CHECK_LEGENDRE) legendreCheck="screen";
00569     else legendreCheck="none";
00570     if(GetConfig().fileCheckMask & Config::CHECK_COUL_AMPLITUDES) coulAmpCheck="file";
00571     else if(GetConfig().screenCheckMask & Config::CHECK_COUL_AMPLITUDES) coulAmpCheck="screen";
00572     else coulAmpCheck="none";
00573     if(GetConfig().fileCheckMask & Config::CHECK_PATHWAYS) pathwaysCheck="file";
00574     else if(GetConfig().screenCheckMask & Config::CHECK_PATHWAYS) pathwaysCheck="screen";
00575     else pathwaysCheck="none";
00576     if(GetConfig().fileCheckMask & Config::CHECK_ANGULAR_DIST) angDistsCheck="file";
00577     else if(GetConfig().screenCheckMask & Config::CHECK_ANGULAR_DIST) angDistsCheck="screen";
00578     else angDistsCheck="none";
00579
00580     outStream.setFieldAlignment(QTextStream::AlignLeft);
00581     outStream << qSetFieldWidth(100) << isAMatrix << qSetFieldWidth(0) << "#Perform A-Matrix Calculation" <<
endl;
00582     outStream << qSetFieldWidth(100) << outputDirectory << qSetFieldWidth(0) << "#Full Path to Output
Directory" << endl;
00583     outStream << qSetFieldWidth(100) << checksDirectory << qSetFieldWidth(0) << "#Full Path to Checks
Directory" << endl;
00584     outStream << qSetFieldWidth(100) << compoundCheck << qSetFieldWidth(0) << "#Compound Nucleus Check" <<
endl;
00585     outStream << qSetFieldWidth(100) << boundaryCheck << qSetFieldWidth(0) << "#Boundary Condition Check" <<
endl;
00586     outStream << qSetFieldWidth(100) << dataCheck << qSetFieldWidth(0) << "#Data Check" << endl;
00587     outStream << qSetFieldWidth(100) << lMatrixCheck << qSetFieldWidth(0) << "#Lo-Matrix and Penetrability
Check" << endl;
00588     outStream << qSetFieldWidth(100) << legendreCheck << qSetFieldWidth(0) << "#Legendre Polynomial Check" <<
endl;
00589     outStream << qSetFieldWidth(100) << coulAmpCheck << qSetFieldWidth(0) << "#Coulomb Amplitudes Check" <<
endl;
00590     outStream << qSetFieldWidth(100) << pathwaysCheck << qSetFieldWidth(0) << "#Reaction Pathway Check" <<
endl;
00591     outStream << qSetFieldWidth(100) << angDistsCheck << qSetFieldWidth(0) << "#Angular Distributions Check"
<< endl;
00592
00593     return true;
00594 }
00595
00596 bool AZURESetup::writeLastRun(QTextStream& outStream) {
00597     unsigned int paramMask = GetConfig().paramMask;
00598
00599     if(runTab->calcType->currentIndex()==1 ||
00600        runTab->calcType->currentIndex()==3) paramMask |= Config::PERFORM_FIT;
00601     else paramMask &= ~Config::PERFORM_FIT;
00602     if(runTab->calcType->currentIndex()==2 ||
00603        runTab->calcType->currentIndex()==4) paramMask &= ~Config::CALCULATE_WITH_DATA;
00604     else paramMask |= Config::CALCULATE_WITH_DATA;
00605     if(runTab->calcType->currentIndex()==3) paramMask |= Config::PERFORM_ERROR_ANALYSIS;
00606     else paramMask &= ~Config::PERFORM_ERROR_ANALYSIS;
00607     if(runTab->calcType->currentIndex()==4) paramMask |= Config::CALCULATE_REACTION_RATE;
00608     else paramMask &= ~Config::CALCULATE_REACTION_RATE;
00609
00610     if(runTab->oldParamFileButton->isChecked())
00611         paramMask |= Config::USE_PREVIOUS_PARAMETERS;
00612     else paramMask &= ~Config::USE_PREVIOUS_PARAMETERS;
00613     if(runTab->oldIntegralsFileButton->isChecked())
00614         paramMask |= Config::USE_PREVIOUS_INTEGRALS;
00615     else paramMask &= ~Config::USE_PREVIOUS_INTEGRALS;
00616
00617     outStream << paramMask << endl;
00618     outStream << "' ' << runTab->paramFileText->text() << "' ' << endl;
00619     outStream << "' ' << runTab->integralsFileText->text() << "' ' << endl;
00620     if(!runTab->rateEntranceKey->text().isEmpty()) outStream << runTab->rateEntranceKey->text() << ' ';
00621     else outStream << "0 ";
00622     if(!runTab->rateExitKey->text().isEmpty()) outStream << runTab->rateExitKey->text();
00623     else outStream << "0";
00624     outStream << endl;
00625     if(runTab->fileTempButton->isChecked()) outStream << "1 ";
00626     else outStream << "0 ";
00627     outStream << "' ' << runTab->fileTempText->text() << "' ' << endl;
00628     if(!runTab->minTempText->text().isEmpty()) outStream << runTab->minTempText->text() << ' ';
00629     else outStream << "-1. ";
00630     if(!runTab->maxTempText->text().isEmpty()) outStream << runTab->maxTempText->text() << ' ';
00631     else outStream << "-1. ";
00632     if(!runTab->tempStepText->text().isEmpty()) outStream << runTab->tempStepText->text();
00633     else outStream << "-1.";
00634     outStream << endl;
00635
00636     return true;
00637 }
```



```

00638
00639 void AZURESetup::matrixChanged(QAction *action) {
00640     if(action==aMatrixAction) GetConfig().paramMask |= Config::USE_AMATRIX;
00641     else {
00642         if(GetConfig().paramMask & Config::IGNORE_ZERO_WIDTHS) {
00643             QMessageBox::information(this, tr("Incompatible Option"),
00644                 tr("The option to ignore external widths is not possible for R-Matrix formalism.
Remove option to use R-Matrix formalism."));
00645             aMatrixAction->activate(QAction::Trigger);
00646         } else GetConfig().paramMask &= ~Config::USE_AMATRIX;
00647     }
00648 }
00649
00650 void AZURESetup::editChecks() {
00651     EditChecksDialog aDialog;
00652     if(GetConfig().fileCheckMask & Config::CHECK_COMPOUND_NUCLEUS)
00653         aDialog.compoundCheckCombo->setCurrentIndex(2);
00654     else if(GetConfig().screenCheckMask & Config::CHECK_COMPOUND_NUCLEUS)
00655         aDialog.compoundCheckCombo->setCurrentIndex(1);
00656     else aDialog.compoundCheckCombo->setCurrentIndex(0);
00657
00658     if(GetConfig().fileCheckMask & Config::CHECK_BOUNDARY_CONDITIONS)
00659         aDialog.boundaryCheckCombo->setCurrentIndex(2);
00660     else if(GetConfig().screenCheckMask & Config::CHECK_BOUNDARY_CONDITIONS)
00661         aDialog.boundaryCheckCombo->setCurrentIndex(1);
00662     else aDialog.boundaryCheckCombo->setCurrentIndex(0);
00663
00664     if(GetConfig().fileCheckMask & Config::CHECK_DATA)
00665         aDialog.dataCheckCombo->setCurrentIndex(2);
00666     else if(GetConfig().screenCheckMask & Config::CHECK_DATA)
00667         aDialog.dataCheckCombo->setCurrentIndex(1);
00668     else aDialog.dataCheckCombo->setCurrentIndex(0);
00669
00670     if(GetConfig().fileCheckMask & Config::CHECK_ENERGY_DEP)
00671         aDialog.lMatrixCheckCombo->setCurrentIndex(2);
00672     else if(GetConfig().screenCheckMask & Config::CHECK_ENERGY_DEP)
00673         aDialog.lMatrixCheckCombo->setCurrentIndex(1);
00674     else aDialog.lMatrixCheckCombo->setCurrentIndex(0);
00675
00676     if(GetConfig().fileCheckMask & Config::CHECK_LEGENDRE)
00677         aDialog.legendreCheckCombo->setCurrentIndex(2);
00678     else if(GetConfig().screenCheckMask & Config::CHECK_LEGENDRE)
00679         aDialog.legendreCheckCombo->setCurrentIndex(1);
00680     else aDialog.legendreCheckCombo->setCurrentIndex(0);
00681
00682     if(GetConfig().fileCheckMask & Config::CHECK_COUL_AMPLITUDES)
00683         aDialog.coulAmpCheckCombo->setCurrentIndex(2);
00684     else if(GetConfig().screenCheckMask & Config::CHECK_COUL_AMPLITUDES)
00685         aDialog.coulAmpCheckCombo->setCurrentIndex(1);
00686     else aDialog.coulAmpCheckCombo->setCurrentIndex(0);
00687
00688     if(GetConfig().fileCheckMask & Config::CHECK_PATHWAYS)
00689         aDialog.pathwaysCheckCombo->setCurrentIndex(2);
00690     else if(GetConfig().screenCheckMask & Config::CHECK_PATHWAYS)
00691         aDialog.pathwaysCheckCombo->setCurrentIndex(1);
00692     else aDialog.pathwaysCheckCombo->setCurrentIndex(0);
00693
00694     if(GetConfig().fileCheckMask & Config::CHECK_ANGULAR_DIST)
00695         aDialog.angDistsCheckCombo->setCurrentIndex(2);
00696     else if(GetConfig().screenCheckMask & Config::CHECK_ANGULAR_DIST)
00697         aDialog.angDistsCheckCombo->setCurrentIndex(1);
00698     else aDialog.angDistsCheckCombo->setCurrentIndex(0);
00699
00700     if(aDialog.exec()) {
00701         GetConfig().fileCheckMask=0;
00702         GetConfig().screenCheckMask=0;
00703         if(aDialog.compoundCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |=
Config::CHECK_COMPOUND_NUCLEUS;
00704         else if(aDialog.compoundCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |=
Config::CHECK_COMPOUND_NUCLEUS;
00705         if(aDialog.boundaryCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |=
Config::CHECK_BOUNDARY_CONDITIONS;
00706         else if(aDialog.boundaryCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |=
Config::CHECK_BOUNDARY_CONDITIONS;
00707         if(aDialog.dataCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |= Config::CHECK_DATA;
00708         else if(aDialog.dataCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |=
Config::CHECK_DATA;
00709         if(aDialog.lMatrixCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |=
Config::CHECK_ENERGY_DEP;
00710         else if(aDialog.lMatrixCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |=
Config::CHECK_ENERGY_DEP;
00711         if(aDialog.legendreCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |=
Config::CHECK_LEGENDRE;
00712         else if(aDialog.legendreCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |=
Config::CHECK_LEGENDRE;
00713         if(aDialog.coulAmpCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |=
Config::CHECK_COUL_AMPLITUDES;

```

```

00714     else if(aDialog.coulAmpCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |=
Config::CHECK_COUL_AMPLITUDES;
00715     if(aDialog.pathwaysCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |=
Config::CHECK_PATHWAYS;
00716     else if(aDialog.pathwaysCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |=
Config::CHECK_PATHWAYS;
00717     if(aDialog.angDistsCheckCombo->currentIndex()==1) GetConfig().screenCheckMask |=
Config::CHECK_ANGULAR_DIST;
00718     else if(aDialog.angDistsCheckCombo->currentIndex()==2) GetConfig().fileCheckMask |=
Config::CHECK_ANGULAR_DIST;
00719 }
00720 }
00721
00722 void AZURESetup::editDirs() {
00723     EditDirsDialog aDialog;
00724
00725     aDialog.outputDirectoryText->setText(QString::fromStdString(GetConfig().outputdir));
00726     aDialog.checksDirectoryText->setText(QString::fromStdString(GetConfig().checkdir));
00727
00728     if(aDialog.exec()) {
00729         GetConfig().outputdir=aDialog.outputDirectoryText->text().toStdString();
00730         GetConfig().checkdir=aDialog.checksDirectoryText->text().toStdString();
00731     }
00732 }
00733
00734 void AZURESetup::editOptions() {
00735     EditOptionsDialog aDialog;
00736
00737     if(GetConfig().paramMask & Config::USE_GSL_COULOMB_FUNC) aDialog.useGSLCoulCheck->setChecked(true);
00738     else aDialog.useGSLCoulCheck->setChecked(false);
00739
00740     if(GetConfig().paramMask & Config::USE_BRUNE_FORMALISM) aDialog.useBruneCheck->setChecked(true);
00741     else aDialog.useBruneCheck->setChecked(false);
00742
00743     if(GetConfig().paramMask & Config::IGNORE_ZERO_WIDTHS)
aDialog.ignoreExternalsCheck->setChecked(true);
00744     else aDialog.ignoreExternalsCheck->setChecked(false);
00745
00746     if(GetConfig().paramMask & Config::USE_RMC_FORMALISM) aDialog.useRMCCheck->setChecked(true);
00747     else aDialog.useRMCCheck->setChecked(false);
00748
00749     if(!(GetConfig().paramMask & Config::TRANSFORM_PARAMETERS))
aDialog.noTransformCheck->setChecked(true);
00750     else aDialog.noTransformCheck->setChecked(false);
00751
00752     //if(!(GetConfig().paramMask & Config::USE_LONGWAVELENGTH_APPROX))
aDialog.noLongWavelengthCheck->setChecked(true);
00753     //else aDialog.noLongWavelengthCheck->setChecked(false);
00754
00755     if(aDialog.exec()) {
00756         if(aDialog.useGSLCoulCheck->isChecked()) GetConfig().paramMask |= Config::USE_GSL_COULOMB_FUNC;
00757         else GetConfig().paramMask &= ~Config::USE_GSL_COULOMB_FUNC;
00758
00759         if(aDialog.useBruneCheck->isChecked())
GetConfig().paramMask |= Config::USE_BRUNE_FORMALISM;
00760         else GetConfig().paramMask &= ~Config::USE_BRUNE_FORMALISM;
00761
00762         if(aDialog.ignoreExternalsCheck->isChecked()) {
GetConfig().paramMask |= Config::IGNORE_ZERO_WIDTHS;
00763         if(!(GetConfig().paramMask & Config::USE_AMATRIX)) {
00764             QMessageBox::information(this,tr("Incompatible Option"),
00765                                     tr("The option to ignore external widths is not possible for R-Matrix formalism. The
formalism will be changed to A-Matrix. "));
00766             aMatrixAction->activate(QAction::Trigger);
00767         }
00768     } else GetConfig().paramMask &= ~Config::IGNORE_ZERO_WIDTHS;
00769
00770     if(aDialog.useRMCCheck->isChecked()) GetConfig().paramMask |= Config::USE_RMC_FORMALISM;
00771     else GetConfig().paramMask &= ~Config::USE_RMC_FORMALISM;
00772
00773     if(aDialog.noTransformCheck->isChecked()) GetConfig().paramMask &= ~Config::TRANSFORM_PARAMETERS;
00774     else GetConfig().paramMask |= Config::TRANSFORM_PARAMETERS;
00775
00776     //if(aDialog.noLongWavelengthCheck->isChecked()) GetConfig().paramMask &=
~Config::USE_LONGWAVELENGTH_APPROX;
00777     //else GetConfig().paramMask |= Config::USE_LONGWAVELENGTH_APPROX;
00778 }
00779 }
00780 }
00781 }
00782
00783 void AZURESetup::SaveAndRun() {
00784     save();
00785     if(GetConfig().configfile.empty()) return;
00786     runTab->runtimeText->clear();
00787     QFile file(QString::fromStdString(GetConfig().configfile));
00788     QFile::Info info(file);
00789     QString directory=info.absolutePath();
00790     if(GetConfig().outputdir.empty())

```

```

    GetConfig().outputdir=QDir::fromNativeSeparators(directory).toStdString()+"/";
00791     if(GetConfig().checkdir.empty())
    GetConfig().checkdir=QDir::fromNativeSeparators(directory).toStdString()+"/";
00792
00793     GetConfig().chiVariance=runTab->chiVarianceText->text().toDouble();
00794
00795     if(runTab->calcType->currentIndex()==1 ||
00796         runTab->calcType->currentIndex()==3) GetConfig().paramMask |= Config::PERFORM_FIT;
00797     else GetConfig().paramMask &= ~Config::PERFORM_FIT;
00798     if(runTab->calcType->currentIndex()==2 ||
00799         runTab->calcType->currentIndex()==4) GetConfig().paramMask &= ~Config::CALCULATE_WITH_DATA;
00800     else GetConfig().paramMask |= Config::CALCULATE_WITH_DATA;
00801     if(runTab->calcType->currentIndex()==3) GetConfig().paramMask |= Config::PERFORM_ERROR_ANALYSIS;
00802     else GetConfig().paramMask &= ~Config::PERFORM_ERROR_ANALYSIS;
00803     if(runTab->calcType->currentIndex()==4) GetConfig().paramMask |= Config::CALCULATE_REACTION_RATE;
00804     else GetConfig().paramMask &= ~Config::CALCULATE_REACTION_RATE;
00805
00806     if(runTab->oldParamFileButton->isChecked()) {
00807         GetConfig().paramMask |= Config::USE_PREVIOUS_PARAMETERS;
00808         GetConfig().paramfile=runTab->paramFileText->text().toStdString();
00809     } else GetConfig().paramMask &= ~Config::USE_PREVIOUS_PARAMETERS;
00810
00811     std::vector<SegPairs> segPairs;
00812     if(!(GetConfig().paramMask & Config::CALCULATE_REACTION_RATE)) {
00813         if(!readSegmentFile(GetConfig(),segPairs)) return;
00814     } else {
00815         GetConfig().rateParams.entrancePair=runTab->rateEntranceKey->text().toInt();
00816         GetConfig().rateParams.exitPair=runTab->rateExitKey->text().toInt();
00817         if(GetConfig().rateParams.entrancePair==GetConfig().rateParams.exitPair) {
00818             QMessageBox::information(this,tr("No Scattering Rates"),
00819                                     tr("Reaction rates cannot be calculated for elastic scattering."));
00820             return;
00821         }
00822         if(!runTab->fileTempButton->isChecked()) {
00823             GetConfig().rateParams.useFile=false;
00824             GetConfig().rateParams.minTemp = runTab->minTempText->text().toDouble();
00825             GetConfig().rateParams.maxTemp = runTab->maxTempText->text().toDouble();
00826             GetConfig().rateParams.tempStep = runTab->tempStepText->text().toDouble();
00827         } else {
00828             GetConfig().rateParams.useFile=true;
00829             GetConfig().rateParams.temperatureFile = runTab->fileTempText->text().toStdString();
00830         }
00831         SegPairs tempPair = {runTab->rateEntranceKey->text().toInt(),
00832                             runTab->rateExitKey->text().toInt()};
00833         segPairs.push_back(tempPair);
00834     }
00835     if(segPairs.size()==0) {
00836         QMessageBox::information(this,tr("Empty Segments"),tr("No active segments have been found."));
00837         return;
00838     }
00839     int maxPairs=pairsTab->getPairsModel()->getPairs().size();
00840     for(std::vector<SegPairs>::const_iterator it = segPairs.begin();it<segPairs.end();it++) {
00841         if(it->secondPair==-1) {
00842             QList<PairsData> pairsList = pairsTab->getPairsModel()->getPairs();
00843             int i;
00844             for(i = 0; i<pairsList.size();i++)
00845                 if(pairsList[i].pairType==10) break;
00846             if(i==pairsList.size()) {
00847                 QMessageBox::information(this,tr("No Capture Pairs"),
00848                                         tr("Total capture is specified, but no capture pairs exist."));
00849                 return;
00850             }
00851         } else if(it->firstPair>maxPairs||it->secondPair>maxPairs||it->firstPair<1||it->secondPair<1) {
00852             QMessageBox::information(this,tr("Undefined Key"),tr("An undefined pair key is specified."));
00853             return;
00854         }
00855     }
00856
00857     GetConfig().paramMask &= ~Config::USE_EXTERNAL_CAPTURE;
00858     if(!checkExternalCapture(GetConfig(),segPairs)) return;
00859     if(GetConfig().paramMask & Config::USE_EXTERNAL_CAPTURE) {
00860         if(runTab->oldIntegralsFileButton->isChecked() &&
00861             !(GetConfig().paramMask & Config::CALCULATE_REACTION_RATE)) {
00862             GetConfig().paramMask |= Config::USE_PREVIOUS_INTEGRALS;
00863             GetConfig().integralsfile=runTab->integralsFileText->text().toStdString();
00864         } else GetConfig().paramMask &= ~Config::USE_PREVIOUS_INTEGRALS;
00865     }
00866
00867     if(!QDir(QString::fromStdString(GetConfig().outputdir)).exists()) {
00868         QMessageBox::information(this,tr("Directory Doesn't Exist"),
00869                                 tr("The specified output directory doesn't exist."));
00870         return;
00871     }
00872     if(!QDir(QString::fromStdString(GetConfig().checkdir)).exists()) {
00873         QMessageBox::information(this,tr("Directory Doesn't Exist"),
00874                                 tr("The specified checks directory doesn't exist."));
00875         return;

```

```

00876     }
00877     if((GetConfig().paramMask & Config::USE_PREVIOUS_PARAMETERS) &&
00878        !QFile(QString::fromStdString(GetConfig().paramfile)).exists()) {
00879         QMessageBox::information(this, tr("File Doesn't Exist"),
00880                                tr("The specified parameter file doesn't exist."));
00881         return;
00882     }
00883     if(((GetConfig().paramMask & Config::USE_PREVIOUS_INTEGRALS) &&
00884        (GetConfig().paramMask & Config::USE_EXTERNAL_CAPTURE)) &&
00885        !QFile(QString::fromStdString(GetConfig().integralsfile)).exists()) {
00886         QMessageBox::information(this, tr("File Doesn't Exist"),
00887                                tr("The specified integrals file doesn't exist."));
00888         return;
00889     }
00890     if((GetConfig().paramMask & Config::CALCULATE_REACTION_RATE &&
00891        GetConfig().rateParams.useFile) &&
00892        !QFile(QString::fromStdString(GetConfig().rateParams.temperatureFile)).exists()) {
00893         QMessageBox::information(this, tr("File Doesn't Exist"),
00894                                tr("The specified rate temperature file doesn't exist."));
00895         return;
00896     }
00897
00898     if(!(GetConfig().paramMask & Config::CALCULATE_WITH_DATA) &&
00899        !(GetConfig().paramMask & Config::CALCULATE_REACTION_RATE)) {
00900         QList<TargetIntData> targetIntData = targetIntTab->getTargetIntModel()->getLines();
00901         QList<SegmentsTestData> segmentsTestData=segmentsTab->getSegmentsTestModel()->getLines();
00902         for(unsigned int i=0; i<targetIntData.size(); i++) {
00903             if(targetIntData.at(i).isActive==1&&
00904                (targetIntData.at(i).isTargetIntegration||targetIntData.at(i).isConvolution)) {
00905                 unsigned int j=0;
00906                 unsigned int lastSegNum=0;
00907                 bool inclusive=false;
00908                 QList<unsigned int> tempList;
00909                 QString segmentsList = targetIntData.at(i).segmentsList;
00910                 while(j<segmentsList.length()) {
00911                     if(segmentsList[j]>='0' && segmentsList[j]<='9') {
00912                         QString tempString;
00913                         while (segmentsList[j]!=',' && segmentsList[j]!='-' &&
00914                            j<segmentsList.length()) {
00915                             tempString+=segmentsList[j];
00916                             j++;
00917                         }
00918                         QTextStream stm(&tempString);
00919                         unsigned int tempSegNum; stm>tempSegNum;
00920                         if(inclusive==true) for(int k=lastSegNum+1; k<=tempSegNum; k++)
00921                             tempList.push_back(k);
00922                         else tempList.push_back(tempSegNum);
00923                         lastSegNum=tempSegNum;
00924                     }
00925                     if(segmentsList[j]=='-') inclusive=true;
00926                     else inclusive =false;
00927                     j++;
00928                 }
00929                 bool isAngularDistribution=false;
00930                 for(j=0; j<tempList.size(); j++) {
00931                     if(tempList.at(j)<=segmentsTestData.size()) {
00932                         for(int k = 0; k<segmentsTestData.size(); k++) {
00933                             if(segmentsTestData.at(k).isActive==1&&
00934                                tempList.at(j)-1==k&&
00935                                segmentsTestData.at(k).dataType==3) {
00936                                 isAngularDistribution=true;
00937                                 break;
00938                             }
00939                         }
00940                     }
00941                     if(isAngularDistribution) break;
00942                 }
00943                 if(isAngularDistribution) {
00944                     QMessageBox::information(this, tr("Incomptable Options"),
00945                                            tr("Angular distribution coefficients cannot be used with convolution or target
00946 integration."));
00947                     return;
00948                 }
00949             }
00950         }
00951
00952         azureMain = new AZUREMainThread(runTab, GetConfig());
00953         connect(azureMain, SIGNAL(finished()), this, SLOT(DeleteThread()));
00954         setWindowTitle(QString("AZURE2 -- %1 --
00955 Running").arg(QString::fromStdString(GetConfig().configfile)));
00956         runTab->calcButton->setEnabled(false);
00957         runTab->stopAZUREButton->setEnabled(true);
00958         runTab->runtimeText->SetMouseFiltered(true);
00959         startMessage(azureMain->configure());
00960         azureMain->start();
00961     }

```

```

00961
00962 void AZURESetup::DeleteThread() {
00963     exitMessage(azureMain->configure());
00964     QScrollBar *sb = runTab->runtimeText->verticalScrollBar();
00965     sb->setValue(sb->maximum());
00966
00967     setWindowTitle(QString("AZURE2 -- %1").arg(QString::fromStdString(GetConfig().configfile)));
00968     runTab->calcButton->setEnabled(true);
00969     runTab->stopAZUREButton->setEnabled(false);
00970     runTab->runtimeText->SetMouseFiltered(false);
00971     delete azureMain;
00972 }
00973
00974 void AZURESetup::showAbout() {
00975     AboutAZURE2Dialog aboutDialog;
00976     aboutDialog.exec();
00977 }
00978
00979 void AZURESetup::reset() {
00980     GetConfig().Reset();
00981     aMatrixAction->activate(QAction::Trigger);
00982     levelsTab->reset();
00983     segmentsTab->reset();
00984     targetIntTab->reset();
00985     runTab->reset();
00986 #ifdef USE_QWT
00987     plotTab->reset();
00988 #endif
00989     setWindowTitle(tr("AZURE2 -- untitled"));
00990     GetConfig().configfile="";
00991 }
00992
00993 void AZURESetup::showTabInfo() {
00994     QString tabTitle = tabWidget->tabText(tabWidget->currentIndex()).remove(QChar('&'));
00995     if(tabWidget->currentIndex()==0) pairsTab->showInfo(0,tabTitle);
00996     if(tabWidget->currentIndex()==1) levelsTab->showInfo(0,tabTitle);
00997     if(tabWidget->currentIndex()==2) segmentsTab->showInfo(0,tabTitle);
00998     if(tabWidget->currentIndex()==3) targetIntTab->showInfo(0,tabTitle);
00999     if(tabWidget->currentIndex()==4) runTab->showInfo(0,tabTitle);
01000 #ifdef USE_QWT
01001     if(tabWidget->currentIndex()==5) plotTab->showInfo(0,tabTitle);
01002 #endif
01003 }
01004
01005 void AZURESetup::openWebsite() {
01006     if(!QDesktopServices::openUrl(QUrl("https://azure.nd.edu")))
01007         QMessageBox::information(this,
01008             tr("Can't Open Browser"),
01009             tr("AZURE2 could not access your web browser. "
01010                 "Please navitgate to https://azure.nd.edu/ "
01011                 "to visit the website.));
01012 }

```

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.](#)

```

00001 #include <QLabel>
00002 #include <QLineEdit>
00003 #include <QGridLayout>
00004 #include <QVBoxLayout>
00005

```

```

00006 #include "ChannelDetails.h"
00007
00008 ChannelDetails::ChannelDetails(QWidget *parent) : QWidget(parent) {
00009     details=new QLabel;
00010     QFont font("Monospace");
00011     font.setStyleHint(QFont::TypeWriter);
00012     details->setFont(font);
00013     reducedWidthText = new QLineEdit;
00014     reducedWidthText->setMaximumWidth(100);
00015     normParam=new QLabel;
00016     normUnits=new QLabel;
00017
00018     QGridLayout *reducedWidthLayout=new QGridLayout;
00019     reducedWidthLayout->addWidget(normParam,0,0);
00020     reducedWidthLayout->addWidget(reducedWidthText,0,1);
00021     reducedWidthLayout->addWidget(normUnits,0,2);
00022     reducedWidthLayout->addItem(new QSpacerItem(20,20),0,3);
00023     reducedWidthLayout->setColumnStretch(3,1);
00024
00025     QVBoxLayout *mainLayout = new QVBoxLayout;
00026     mainLayout->addWidget(details);
00027     mainLayout->addLayout(reducedWidthLayout);
00028     setLayout(mainLayout);
00029 }
00030
00031 void ChannelDetails::setNormParam(int which) {
00032     if(which==1) {
00033         normParam->setText("ANC:");
00034         normUnits->setText("fm-1/2");
00035     } else if(which==2) {
00036         normParam->setText("Mu:");
00037         normUnits->setText("nm");
00038     } else if(which==3) {
00039         normParam->setText("Q:");
00040         normUnits->setText("b");
00041     } else if(which==4) {
00042         normParam->setText("B:");
00043         normUnits->setText("");
00044     } else {
00045         normParam->setText("Partial Width:");
00046         normUnits->setText("eV");
00047     }
00048 }

```

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.](#)

```

00001 #include "ChannelsModel.h"
00002 #include "PairsModel.h"
00003
00004 ChannelsModel::ChannelsModel(QObject *parent) : QAbstractTableModel(parent) {
00005 }
00006
00007 int ChannelsModel::rowCount(const QModelIndex &parent) const {
00008     Q_UNUSED(parent);
00009     return channelsList.size();
00010 }
00011
00012 int ChannelsModel::columnCount(const QModelIndex &parent) const {
00013     Q_UNUSED(parent);
00014     return ChannelsData::SIZE;
00015 }
00016
00017 QVariant ChannelsModel::data(const QModelIndex &index, int role) const {
00018     if(!index.isValid()) return QVariant();
00019

```

```

00020     if(index.row() >= channelsList.size() || index.row() < 0) return QVariant();
00021
00022     if (role == Qt::DisplayRole) {
00023         ChannelsData channel = channelsList.at(index.row());
00024         if(index.column() == 1) return channel.levelIndex;
00025         else if(index.column() == 2) {
00026             PairsData pair=pairsModel->getPairs().at(channel.pairIndex);
00027             if(channel.reducedWidth!=0.)
00028                 return QString("<center><font
style=' font-weight:bold;'>%1</font></center>").arg(pairsModel->getParticleLabel(pair));
00029             else return QString("<center>%1</center>").arg(pairsModel->getParticleLabel(pair));
00030         } else if(index.column() == 3) {
00031             if(channel.reducedWidth!=0.)
00032                 return QString("<center><font
style=' font-weight:bold;'>%1</font></center>").arg(getSpinLabel(channel));
00033             else return QString("<center>%1</center>").arg(getSpinLabel(channel));
00034         } else if(index.column() == 4) {
00035             if(channel.radType=='P') {
00036                 if(channel.reducedWidth!=0.)
00037                     return QString("<center><font
style=' font-weight:bold;'>%1</font></center>").arg(channel.lValue);
00038                 else return QString("<center>%1</center>").arg(channel.lValue);
00039             } else if(channel.radType=='F') {
00040                 if(channel.reducedWidth!=0.)
00041                     return QString("<center><font style=' font-weight:bold;'>F</font></center>");
00042                 else return QString("<center>F</center>");
00043             } else if(channel.radType=='G') {
00044                 if(channel.reducedWidth!=0.)
00045                     return QString("<center><font style=' font-weight:bold;'>GT</font></center>");
00046                 else return QString("<center>GT</center>");
00047             } else {
00048                 if(channel.reducedWidth!=0.)
00049                     return QString("<center><font
style=' font-weight:bold;'>%1%2</font></center>").arg(channel.radType).arg(channel.lValue);
00050                 else return QString("<center>%1%2</center>").arg(channel.radType).arg(channel.lValue);
00051             }
00052         } else if(index.column() == 5) return channel.radType;
00053         else if(index.column() == 6) return channel.reducedWidth;
00054     } else if(role==Qt::EditRole) {
00055         ChannelsData channel = channelsList.at(index.row());
00056         if(index.column() == 1) return channel.levelIndex;
00057         else if(index.column() == 2) return channel.pairIndex;
00058         else if(index.column() == 3) return channel.sValue;
00059         else if(index.column() == 4) return channel.lValue;
00060         else if(index.column() == 5) return channel.radType;
00061         else if(index.column() == 6) return channel.reducedWidth;
00062     } else if(role==Qt::TextAlignmentRole) return Qt::AlignCenter;
00063     else if (role==Qt::CheckStateRole && index.column()==0) {
00064         ChannelsData channel = channelsList.at(index.row());
00065         if(channel.isFixed==1) return Qt::Checked;
00066         else return Qt::Unchecked;
00067     }
00068     return QVariant();
00069 }
00070
00071 QVariant ChannelsModel::headerData(int section, Qt::Orientation orientation, int role) const {
00072     if(role!= Qt::DisplayRole) return QVariant();
00073     if(orientation == Qt::Horizontal) {
00074         switch(section) {
00075             case 0:
00076                 return tr("Fix?");
00077             case 1:
00078                 return tr("level");
00079             case 2:
00080                 return tr("Channel\nPair");
00081             case 3:
00082                 return tr("s");
00083             case 4:
00084                 return tr("l");
00085             case 5:
00086                 return tr("radiation type");
00087             case 6:
00088                 return tr("reduced width");
00089             default:
00090                 return QVariant();
00091         }
00092     } else if(orientation == Qt::Vertical) {
00093         return section+1;
00094     }
00095     return QVariant();
00096 }
00097
00098 bool ChannelsModel::setData(const QModelIndex &index, const QVariant &value, int role) {
00099     if (index.isValid()){
00100         if (role == Qt::EditRole ) {
00101             int row = index.row();
00102             ChannelsData tempData = channelsList.value(row);

```



```

00103         if (index.column() == 0) tempData.isFixed=value.toInt();
00104     else if (index.column() == 1) tempData.levelIndex=value.toInt();
00105     else if (index.column() == 2) tempData.pairIndex=value.toInt();
00106     else if (index.column() == 3) tempData.sValue=value.toDouble();
00107     else if (index.column() == 4) tempData.lValue=value.toInt();
00108     else if (index.column() == 5) tempData.radType=value.toChar();
00109     else if (index.column() == 6) tempData.reducedWidth=value.toDouble();
00110     else return false;
00111
00112     channelsList.replace(row,tempData);
00113     if(index.column()!=6) emit (dataChanged(index,index));
00114     return true;
00115 } else if(role== Qt::CheckStateRole) {
00116     int row = index.row();
00117     ChannelsData tempData = channelsList.value(row);
00118     if(index.column()==0) {
00119         if(value==Qt::Checked) tempData.isFixed=1;
00120         else tempData.isFixed=0;
00121     } else return false;
00122
00123     channelsList.replace(row,tempData);
00124     emit (dataChanged(index,index));
00125     return true;
00126 }
00127 }
00128 return false;
00129 }
00130
00131 bool ChannelsModel::insertRows(int position, int rows, const QModelIndex &index) {
00132     Q_UNUSED(index);
00133     if(rows>0) {
00134         beginInsertRows(QModelIndex(),position,position+rows-1);
00135         for(int row=0; row<rows; row++) {
00136             ChannelsData tempData={0,-1,-1,0.0,0,'P',0.0};
00137             channelsList.insert(position,tempData);
00138         }
00139         endInsertRows();
00140     }
00141     return true;
00142 }
00143
00144 bool ChannelsModel::removeRows(int position, int rows, const QModelIndex &index) {
00145     Q_UNUSED(index);
00146     if(rows>0) {
00147         beginRemoveRows(QModelIndex(),position,position+rows-1);
00148         for(int row=0; row<rows; ++row) {
00149             channelsList.removeAt(position);
00150         }
00151         endRemoveRows();
00152     }
00153     return true;
00154 }
00155
00156 Qt::ItemFlags ChannelsModel::flags(const QModelIndex &index) const {
00157     if (!index.isValid()) return Qt::ItemIsEnabled;
00158     if (index.column()==0) return QAbstractTableModel::flags(index) | Qt::ItemIsUserCheckable;
00159     return QAbstractTableModel::flags(index);
00160 }
00161
00162 bool ChannelsModel::isChannel(const ChannelsData &channel) const {
00163     bool foundChannel=false;
00164     for(int i=0;i<channelsList.size();i++) {
00165         ChannelsData tempChannel=channelsList.value(i);
00166         if(tempChannel.levelIndex==channel.levelIndex&&
00167            tempChannel.pairIndex==channel.pairIndex&&
00168            tempChannel.sValue==channel.sValue&&
00169            tempChannel.lValue==channel.lValue&&
00170            tempChannel.radType==channel.radType) {
00171             foundChannel=true;
00172             break;
00173         }
00174     }
00175     return foundChannel;
00176 }
00177 }
00178
00179 QString ChannelsModel::getSpinLabel(const ChannelsData &channel) const {
00180     if(((int)(channel.sValue*2))%2!=0&&channel.sValue!=0.) return
00181         QString("%1/2").arg((int)(channel.sValue*2));
00182     else return QString("%1").arg(channel.sValue);
00183 }
00184
00184 void ChannelsModel::setPairsModel(PairsModel* model) {
00185     pairsModel=model;
00186 }

```


8.97 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChooseFileButton.cpp File Reference

```
#include "ChooseFileButton.h"
```

8.98 ChooseFileButton.cpp

[Go to the documentation of this file.](#)

```
00001 #include "ChooseFileButton.h"
00002
00003 ChooseFileButton::ChooseFileButton(const QString& text, QWidget *parent) :
00004     QPushButton(text, parent) {
00005     connect(this, SIGNAL(clicked()), this, SLOT(click()));
00006 };
00007
00008 void ChooseFileButton::setLineEdit(QLineEdit* lineEdit) {
00009     thisLineEdit=lineEdit;
00010 };
00011
00012 void ChooseFileButton::click() {
00013     emit(clicked(thisLineEdit));
00014 };
00015
```

8.99 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditChecksDialog.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.](#)

```
00001 #include <QComboBox>
00002 #include <QGridLayout>
00003 #include <QLabel>
00004 #include <QPushButton>
00005 #include <QGroupBox>
00006 #include "EditChecksDialog.h"
00007
00008 EditChecksDialog::EditChecksDialog(QWidget *parent) : QDialog(parent) {
00009     compoundCheckCombo=new QComboBox;
00010     compoundCheckCombo->addItem(tr("None"));
00011     compoundCheckCombo->addItem(tr("Screen"));
00012     compoundCheckCombo->addItem(tr("File"));
00013     boundaryCheckCombo=new QComboBox;
00014     boundaryCheckCombo->addItem(tr("None"));
00015     boundaryCheckCombo->addItem(tr("Screen"));
00016     boundaryCheckCombo->addItem(tr("File"));
00017     dataCheckCombo=new QComboBox;
00018     dataCheckCombo->addItem(tr("None"));
00019     dataCheckCombo->addItem(tr("Screen"));
00020     dataCheckCombo->addItem(tr("File"));
00021     lMatrixCheckCombo=new QComboBox;
00022     lMatrixCheckCombo->addItem(tr("None"));
```

```

00023  lMatrixCheckCombo->addItem(tr("Screen"));
00024  lMatrixCheckCombo->addItem(tr("File"));
00025  legendreCheckCombo=new QComboBox;
00026  legendreCheckCombo->addItem(tr("None"));
00027  legendreCheckCombo->addItem(tr("Screen"));
00028  legendreCheckCombo->addItem(tr("File"));
00029  coulAmpCheckCombo=new QComboBox;
00030  coulAmpCheckCombo->addItem(tr("None"));
00031  coulAmpCheckCombo->addItem(tr("Screen"));
00032  coulAmpCheckCombo->addItem(tr("File"));
00033  pathwaysCheckCombo=new QComboBox;
00034  pathwaysCheckCombo->addItem(tr("None"));
00035  pathwaysCheckCombo->addItem(tr("Screen"));
00036  pathwaysCheckCombo->addItem(tr("File"));
00037  angDistsCheckCombo=new QComboBox;
00038  angDistsCheckCombo->addItem(tr("None"));
00039  angDistsCheckCombo->addItem(tr("Screen"));
00040  angDistsCheckCombo->addItem(tr("File"));
00041
00042  QGroupBox *checkFilesBox=new QGroupBox(tr("Check Configuration"));
00043  QGridLayout *checkFilesLayout = new QGridLayout;
00044  checkFilesLayout->addWidget(new QLabel(tr("Compound Nucleus:")),0,0,Qt::AlignRight);
00045  checkFilesLayout->addWidget(compoundCheckCombo,0,1);
00046  checkFilesLayout->addWidget(new QLabel(tr("Boundary Conditions:")),1,0,Qt::AlignRight);
00047  checkFilesLayout->addWidget(boundaryCheckCombo,1,1);
00048  checkFilesLayout->addWidget(new QLabel(tr("Data:")),2,0,Qt::AlignRight);
00049  checkFilesLayout->addWidget(dataCheckCombo,2,1);
00050  checkFilesLayout->addWidget(new QLabel(tr("L-Matrix, Phases, Penetrabilities:")),3,0,Qt::AlignRight);
00051  checkFilesLayout->addWidget(lMatrixCheckCombo,3,1);
00052  checkFilesLayout->addWidget(new QLabel(tr("Legendre Polynomials:")),4,0,Qt::AlignRight);
00053  checkFilesLayout->addWidget(legendreCheckCombo,4,1);
00054  checkFilesLayout->addWidget(new QLabel(tr("Coulomb Amplitudes:")),5,0,Qt::AlignRight);
00055  checkFilesLayout->addWidget(coulAmpCheckCombo,5,1);
00056  checkFilesLayout->addWidget(new QLabel(tr("Reaction Pathways:")),6,0,Qt::AlignRight);
00057  checkFilesLayout->addWidget(pathwaysCheckCombo,6,1);
00058  checkFilesLayout->addWidget(new QLabel(tr("Angular Distributions:")),7,0,Qt::AlignRight);
00059  checkFilesLayout->addWidget(angDistsCheckCombo,7,1);
00060  checkFilesLayout->setColumnStretch(0,0);
00061  checkFilesLayout->setColumnStretch(1,1);
00062  checkFilesBox->setLayout(checkFilesLayout);
00063
00064  cancelButton = new QPushButton(tr("Cancel"));
00065  okButton = new QPushButton(tr("Accept"));
00066  okButton->setDefault(true);
00067  connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
00068  connect(cancelButton, SIGNAL(clicked()),this,SLOT(reject()));
00069
00070  QHBoxLayout *buttonBox = new QHBoxLayout;
00071  buttonBox->addWidget(cancelButton);
00072  buttonBox->addWidget(okButton);
00073
00074  QVBoxLayout *mainLayout = new QVBoxLayout;
00075  mainLayout->addWidget(checkFilesBox);
00076  mainLayout->addLayout(buttonBox);
00077
00078  setWindowTitle(tr("Edit Check Configuration"));
00079
00080  setLayout(mainLayout);
00081 }

```

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.](#)

```
00001 #include <QGroupBox>
00002 #include <QGridLayout>
00003 #include <QLabel>
00004 #include <QHBoxLayout>
00005 #include <QFileDialog>
00006
00007 #include "EditDirsDialog.h"
00008 #include "ChooseFileButton.h"
00009
00010 EditDirsDialog::EditDirsDialog(QWidget *parent) : QDialog(parent) {
00011     this->setMinimumWidth(500);
00012     QGroupBox *directoryBox=new QGroupBox(tr("Directory Configuration"));
00013     QGridLayout *directoryLayout = new QGridLayout;
00014     directoryLayout->addWidget(new QLabel(tr("Output Directory:")),0,0,Qt::AlignRight);
00015     outputDirectoryText = new QLineEdit;
00016     directoryLayout->addWidget(outputDirectoryText,0,1);
00017     ChooseFileButton *chooseButton=new ChooseFileButton(tr("Choose..."));
00018     chooseButton->setLineEdit(outputDirectoryText);
00019     connect(chooseButton,SIGNAL(clicked(QLineEdit*)),this,SLOT(setChooseDirectory(QLineEdit*)));
00020     directoryLayout->addWidget(chooseButton,0,2);
00021     directoryLayout->addWidget(new QLabel(tr("Checks Directory:")),1,0,Qt::AlignRight);
00022     checksDirectoryText = new QLineEdit;
00023     directoryLayout->addWidget(checksDirectoryText,1,1);
00024     chooseButton=new ChooseFileButton(tr("Choose..."));
00025     chooseButton->setLineEdit(checksDirectoryText);
00026     connect(chooseButton,SIGNAL(clicked(QLineEdit*)),this,SLOT(setChooseDirectory(QLineEdit*)));
00027     directoryLayout->addWidget(chooseButton,1,2);
00028     directoryBox->setLayout(directoryLayout);
00029
00030     cancelButton = new QPushButton(tr("Cancel"));
00031     okButton = new QPushButton(tr("Accept"));
00032     okButton->setDefault(true);
00033     connect(okButton, SIGNAL(clicked()),this,SLOT(accept()));
00034     connect(cancelButton,SIGNAL(clicked()),this,SLOT(reject()));
00035
00036     QHBoxLayout *buttonBox = new QHBoxLayout;
00037     buttonBox->addWidget(cancelButton);
00038     buttonBox->addWidget(okButton);
00039
00040     QVBoxLayout *mainLayout = new QVBoxLayout;
00041     mainLayout->addWidget(directoryBox);
00042     mainLayout->addLayout(buttonBox);
00043
00044     setWindowTitle(tr("Edit Directory Configuration"));
00045
00046     setLayout(mainLayout);
00047 }
00048
00049 void EditDirsDialog::setChooseDirectory(QLineEdit *lineEdit) {
00050     QString filename = QFileDialog::getExistingDirectory(this);
00051     if(!filename.isEmpty()) {
00052         lineEdit->setText(QDir::fromNativeSeparators(filename)+' /');
00053     }
00054 }
```

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.](#)

```

00001 #include "EditOptionsDialog.h"
00002
00003 #include <QCheckBox>
00004 #include <QVBoxLayout>
00005 #include <QPushButton>
00006 #include <QGroupBox>
00007
00008 EditOptionsDialog::EditOptionsDialog(QWidget* parent) : QDialog(parent) {
00009
00010     useGSLCoulCheck = new QCheckBox(tr("Use GSL Coulomb functions"));
00011     useBruneCheck = new QCheckBox(tr("Use Brune formalism"));
00012     ignoreExternalsCheck = new QCheckBox(tr("Ignore external width\nif internal width is zeroed"));
00013     useRMCCheck = new QCheckBox(tr("Use RMC capture formalism\n(neutron capture only)"));
00014     noTransformCheck = new QCheckBox(tr("Do not perform parameter\ntransformations"));
00015     // noLongWavelengthCheck = new QCheckBox(tr("Do not use long wavelength\n"
00016     //                                     "approximation for EL external capture"));
00017
00018     connect(useBruneCheck, SIGNAL(stateChanged(int)), this, SLOT(useBruneCheckChanged(int)));
00019     connect(useRMCCheck, SIGNAL(stateChanged(int)), this, SLOT(useRMCCheckChanged(int)));
00020
00021     QGroupBox* optionsBox = new QGroupBox(tr("AZURE2 Options"));
00022     QVBoxLayout* optionsLayout = new QVBoxLayout;
00023     optionsLayout->addWidget(useGSLCoulCheck);
00024     optionsLayout->addWidget(useBruneCheck);
00025     optionsLayout->addWidget(ignoreExternalsCheck);
00026     optionsLayout->addWidget(useRMCCheck);
00027     optionsLayout->addWidget(noTransformCheck);
00028     //optionsLayout->addWidget(noLongWavelengthCheck);
00029     optionsBox->setLayout(optionsLayout);
00030
00031     cancelButton = new QPushButton(tr("Cancel"));
00032     okButton = new QPushButton(tr("Accept"));
00033     okButton->setDefault(true);
00034     connect(okButton, SIGNAL(clicked()), this, SLOT(accept()));
00035     connect(cancelButton, SIGNAL(clicked()), this, SLOT(reject()));
00036
00037     QHBoxLayout *buttonBox = new QHBoxLayout;
00038     buttonBox->addWidget(cancelButton);
00039     buttonBox->addWidget(okButton);
00040     QVBoxLayout *mainLayout = new QVBoxLayout;
00041     mainLayout->addWidget(optionsBox);
00042     mainLayout->addLayout(buttonBox);
00043     setWindowTitle(tr("Edit Options"));
00044     setLayout(mainLayout);
00045 }
00046
00047 void EditOptionsDialog::useBruneCheckChanged(int state) {
00048     if(state==Qt::Checked) {
00049         useRMCCheck->setChecked(false);
00050         useRMCCheck->setEnabled(false);
00051     } else useRMCCheck->setEnabled(true);
00052 }
00053
00054 void EditOptionsDialog::useRMCCheckChanged(int state) {
00055     if(state==Qt::Checked) {
00056         useBruneCheck->setChecked(false);
00057         useBruneCheck->setEnabled(false);
00058     } else useBruneCheck->setEnabled(true);
00059 }

```

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.](#)

```

00001 #include <QTextEdit>
00002 #include <QPushButton>
00003 #include <QVBoxLayout>
00004
00005 #include "InfoDialog.h"
00006
00007 InfoDialog::InfoDialog(const QString& string,
00008                        QWidget* parent,
00009                        QString title) : QDialog(parent) {
00010     setWindowTitle(QString("Documentation for %1 Tab").arg(title));
00011     setMinimumSize(600,400);
00012
00013     QTextEdit* textEdit = new QTextEdit;
00014     textEdit->setReadOnly(true);
00015     textEdit->setAcceptRichText(false);
00016     textEdit->setHtml(string);
00017
00018     QPushButton* okButton = new QPushButton(tr("OK"),this);
00019     okButton->setMaximumSize(80,30);
00020
00021     QVBoxLayout* layout = new QVBoxLayout(this);
00022     layout->addWidget(textEdit);
00023     layout->addWidget(okButton);
00024     layout->setAlignment(okButton,Qt::AlignHCenter);
00025
00026     connect(okButton,SIGNAL(clicked()),this,SLOT(close()));
00027
00028     setLayout(layout);
00029 }

```

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.](#)

```
00001 #include "PairsTab.h"
00002 #include "LevelsTab.h"
00003 #include "SegmentsTab.h"
00004 #include "TargetIntTab.h"
00005 #include "RunTab.h"
00006 #include "PlotTab.h"
00007
00008 std::vector<QString> setInfoStrings(QString infoString1 = "",
00009                                   QString infoString2 = "",
00010                                   QString infoString3 = "",
00011                                   QString infoString4 = "",
00012                                   QString infoString5 = "") {
00013     std::vector<QString> stringVector;
00014     if(infoString1!="") stringVector.push_back(infoString1);
00015     if(infoString2!="") stringVector.push_back(infoString2);
00016     if(infoString3!="") stringVector.push_back(infoString3);
00017     if(infoString4!="") stringVector.push_back(infoString4);
00018     if(infoString5!="") stringVector.push_back(infoString5);
00019     return stringVector;
00020 };
00021
00022 const std::vector<QString> PairsTab::infoText =
00023     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>"
00024 "<ul><li><p>Use the &#43; or &#45; buttons on the lower left corner to add or delete a decay pair
respectively.</p></ul>"
00025 "<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>"
00026 "<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>");
00027
00028 const std::vector<QString> LevelsTab::infoText =
00029     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>"
00030 "<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-parity 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>"
00031 "<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>"
00032 "<ul><li><p>&gamma;-ray decays are limited to bound states.</p></ul>"
00033 "<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>"
00034 "<ul><li><p>Relative interferences for channels can be specified by changing the sign of the
corresponding partial width or ANC.</p></ul>"
00035 "<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>"
00036 "<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>");
00037
00038 const std::vector<QString> SegmentsTab::infoText =
00039     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>"
00040 "<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>"
00041 "<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 coefficients.</p></ul>"
00042 "<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>"
00043 "<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>"
00044 "<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>";
00045
00046 const std::vector<QString> TargetIntTab::infoText =
00047     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>"
00048 "<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>"
00049 "<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>"
00050 "<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>"
00051 "<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>"
00052 "<ul><li><p>Examples: ``5, 8, 13" and ``4-12" and ``3,6,7-14" are all valid. </p></ul>");
00053
00054 const std::vector<QString> RunTab::infoText =
00055     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>"
00056 "<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>"
00057 "<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>"
00058 "<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>"
00059 "<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>"
00060 "<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>"
00061 "<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>");

```
00062
00063 const std::vector<QString> PlotTab::infoText =
00064     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.](#)

```
00001 #include "LevelsModel.h"
00002
00003 LevelsModel::LevelsModel(QObject *parent) : QAbstractTableModel(parent) {
00004 }
00005
00006 int LevelsModel::rowCount(const QModelIndex &parent) const {
00007     Q_UNUSED(parent);
00008     return levelsList.size();
00009 }
00010
00011 int LevelsModel::columnCount(const QModelIndex &parent) const {
00012     Q_UNUSED(parent);
00013     return LevelsData::SIZE;
00014 }
00015
00016 QVariant LevelsModel::data(const QModelIndex &index, int role) const {
00017     if(!index.isValid()) return QVariant();
00018
00019     if(index.row() >= levelsList.size() || index.row() < 0) return QVariant();
00020
00021     if (role == Qt::DisplayRole) {
00022         LevelsData level = levelsList.at(index.row());
00023         if(index.column() == 2) return getSpinLabel(level);
00024         else if(index.column() == 3) return level.piValue;
00025         else if(index.column() == 4) return level.energy;
00026     } else if (role == Qt::EditRole) {
00027         LevelsData level = levelsList.at(index.row());
00028         if(index.column() == 2) return level.jValue;
00029         else if(index.column() == 3) return level.piValue;
00030         else if(index.column() == 4) return level.energy;
00031     } else if(role==Qt::TextAlignmentRole) return Qt::AlignCenter;
00032     else if (role==Qt::CheckStateRole && (index.column()==0||index.column()==1)) {
00033         LevelsData level = levelsList.at(index.row());
00034         if(index.column()==0) {
00035             if(level.isActive==1) return Qt::Checked;
00036             else return Qt::Unchecked;
00037         } else {
00038             if(level.isFixed==1) return Qt::Checked;
00039             else return Qt::Unchecked;
00040         }
00041     }
00042     return QVariant();
00043 }
00044
00045 QVariant LevelsModel::headerData(int section, Qt::Orientation orientation, int role) const {
00046     if(role!= Qt::DisplayRole) return QVariant();
00047     if(orientation == Qt::Horizontal) {
00048         switch(section) {
00049             case 0:
00050                 return tr("Include?");
```



```

00051     case 1:
00052         return tr("Fix?");
00053     case 2:
00054         return tr("Level\nSpin");
00055     case 3:
00056         return tr("Parity");
00057     case 4:
00058         return tr("Energy\n[MeV]");
00059     default:
00060         return QVariant();
00061     }
00062 } else if(orientation == Qt::Vertical) {
00063     return section+1;
00064 }
00065 return QVariant();
00066 }
00067
00068 bool LevelsModel::setData(const QModelIndex &index, const QVariant &value, int role) {
00069     if (index.isValid()) {
00070         if (role == Qt::EditRole) {
00071             int row = index.row();
00072             LevelsData tempData = levelsList.value(row);
00073             if (index.column() == 0) tempData.isActive=value.toInt();
00074             else if (index.column() == 1) tempData.isFixed=value.toInt();
00075             else if (index.column() == 2) tempData.jValue=value.toDouble();
00076             else if (index.column() == 3) tempData.piValue=value.toInt();
00077             else if (index.column() == 4) tempData.energy=value.toDouble();
00078             else return false;
00079
00080             levelsList.replace(row,tempData);
00081             emit (dataChanged(index,index));
00082             return true;
00083         } else if (role== Qt::CheckStateRole) {
00084             int row = index.row();
00085             LevelsData tempData = levelsList.value(row);
00086             if (index.column()==0) {
00087                 if (value==Qt::Checked) tempData.isActive=1;
00088                 else tempData.isActive=0;
00089             } else if (index.column()==1) {
00090                 if (value==Qt::Checked) tempData.isFixed=1;
00091                 else tempData.isFixed=0;
00092             } else return false;
00093
00094             levelsList.replace(row,tempData);
00095             emit (dataChanged(index,index));
00096             return true;
00097         }
00098     }
00099     return false;
00100 }
00101
00102 bool LevelsModel::insertRows(int position, int rows, const QModelIndex &index) {
00103     Q_UNUSED(index);
00104     if (rows>0) {
00105         beginInsertRows(QModelIndex(),position,position+rows-1);
00106         for (int row=0; row<rows; row++) {
00107             LevelsData tempData={1,0,0.0,-1,0.0};
00108             levelsList.insert (position,tempData);
00109         }
00110         endInsertRows();
00111     }
00112     return true;
00113 }
00114
00115 bool LevelsModel::removeRows(int position, int rows, const QModelIndex &index) {
00116     Q_UNUSED(index);
00117     if (rows>0) {
00118         beginRemoveRows(QModelIndex(),position,position+rows-1);
00119         for (int row=0; row<rows;++row) {
00120             levelsList.removeAt (position);
00121         }
00122         endRemoveRows();
00123     }
00124     return true;
00125 }
00126
00127 Qt::ItemFlags LevelsModel::flags(const QModelIndex &index) const {
00128     if (!index.isValid()) return Qt::ItemIsEnabled;
00129     if (index.column()==0 || index.column() ==1) return QAbstractTableModel::flags(index) |
Qt::ItemIsUserCheckable;
00130     return QAbstractTableModel::flags (index);
00131 }
00132
00133 int LevelsModel::isLevel(const LevelsData &level) const {
00134     int foundLevel=-1;
00135     for (int i=0; i<levelsList.size(); i++) {
00136         LevelsData tempLevel=levelsList.value(i);

```

```

00137         if (tempLevel.jValue==level.jValue&&
00138             tempLevel.piValue==level.piValue&&
00139             tempLevel.energy==level.energy) {
00140             foundLevel=i;
00141             break;
00142         }
00143     }
00144     return foundLevel;
00145 }
00146
00147 QString LevelsModel::getSpinLabel(const LevelsData &level) const {
00148     QString tempSpin;
00149     if(((int)(level.jValue*2))%2!=0&&level.jValue!=0.)
00150         tempSpin=QString("%1/2").arg((int)(level.jValue*2));
00151     else tempSpin=QString("%1").arg(level.jValue);
00152     if(level.piValue===-1) return QString("%1-").arg(tempSpin);
00153     else return QString("%1+").arg(tempSpin);
00154 }

```

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.](#)

```

00001 #include <QHeaderView>
00002
00003 #include "LevelsTab.h"
00004 #include "LevelsHeaderView.h"
00005 #include "AddLevelDialog.h"
00006 #include "RichTextDelegate.h"
00007 #include "InfoDialog.h"
00008 #include <iostream>
00009
00010 LevelsTab::LevelsTab(QWidget *parent) : QWidget(parent) {
00011     levelsModel=new LevelsModel(this);
00012     levelsModelProxy = new QSortFilterProxyModel(this);
00013     levelsModelProxy->setSourceModel(levelsModel);
00014     levelsModelProxy->setDynamicSortFilter(true);
00015     levelsView=new QTableView;
00016     levelsView->setHorizontalHeader(new LevelsHeaderView(Qt::Horizontal, levelsView));
00017     levelsView->setModel(levelsModelProxy);
00018     levelsView->horizontalHeader()->setSortIndicator(4,Qt::AscendingOrder);
00019     levelsView->setSortingEnabled(true);
00020     levelsView->verticalHeader()->hide();
00021     levelsView->horizontalHeader()->setHighlightSections(false);
00022     levelsView->setColumnWidth(0,60);
00023     levelsView->setColumnWidth(1,40);
00024     levelsView->horizontalHeader()->setSectionResizeMode(0,QHeaderView::Fixed);
00025     levelsView->horizontalHeader()->setSectionResizeMode(1,QHeaderView::Fixed);
00026     levelsView->horizontalHeader()->setSectionResizeMode(2,QHeaderView::Stretch);
00027     levelsView->horizontalHeader()->setSectionResizeMode(4,QHeaderView::Stretch);
00028     levelsView->setColumnHidden(3,true);
00029     levelsView->setSelectionBehavior(QAbstractItemView::SelectRows);
00030     levelsView->setSelectionMode(QAbstractItemView::SingleSelection);
00031     levelsView->setEditTriggers(QAbstractItemView::NoEditTriggers);
00032     levelsView->setShowGrid(false);
00033
00034     connect (levelsView->selectionModel(), SIGNAL(selectionChanged(QItemSelection,QItemSelection)), this, SLOT(updateButtons(QItemSelection,QItemSelection)));
00035     connect (levelsView->selectionModel(), SIGNAL(selectionChanged(QItemSelection,QItemSelection)), this, SLOT(updateFilter(QItemSelection,QItemSelection)));

```

```

00035     connect (levelsView,SIGNAL(doubleClicked(QModelIndex)),this,SLOT(editLevel()));
00036
00037     maxLSpin = new QSpinBox;
00038     maxLSpin->setMinimum(0);
00039     maxLSpin->setMaximum(10);
00040     maxLSpin->setSingleStep(1);
00041     maxLSpin->setValue(2);
00042     QLabel *maxLLabel = new QLabel(tr("Maximum Orbital Momentum"));
00043     maxMultSpin = new QSpinBox;
00044     maxMultSpin->setMinimum(1);
00045     maxMultSpin->setMaximum(10);
00046     maxMultSpin->setSingleStep(1);
00047     maxMultSpin->setValue(2);
00048     QLabel *maxMultLabel = new QLabel(tr("Maximum Gamma Multipolarity"));
00049     maxNumMultSpin = new QSpinBox;
00050     maxNumMultSpin->setMinimum(1);
00051     maxNumMultSpin->setMaximum(10);
00052     maxNumMultSpin->setSingleStep(1);
00053     maxNumMultSpin->setValue(2);
00054     QLabel *maxNumMultLabel = new QLabel(tr("Maximum Gamma Multipolarities\nPer Decay"));
00055     connect (maxLSpin,SIGNAL(valueChanged(int)),this,SLOT(updateChannelsPairAddedEdited()));
00056     connect (maxMultSpin,SIGNAL(valueChanged(int)),this,SLOT(updateChannelsPairAddedEdited()));
00057     connect (maxNumMultSpin,SIGNAL(valueChanged(int)),this,SLOT(updateChannelsPairAddedEdited()));
00058
00059     channelsModel = new ChannelsModel(this);
00060     proxyModel = new QSortFilterProxyModel(this);
00061     proxyModel->setSourceModel(channelsModel);
00062     proxyModel->setDynamicSortFilter(true);
00063     proxyModel->setFilterKeyColumn(1);
00064     proxyModel->setFilterRegExp("-1");
00065     proxyModel->sort(1,Qt::AscendingOrder);
00066     channelsView = new QTableView;
00067     channelsView->setModel(proxyModel);
00068     channelsView->verticalHeader()->hide();
00069     channelsView->horizontalHeader()->setHighlightSections(false);
00070     channelsView->setColumnWidth(0,40);
00071     channelsView->setColumnWidth(2,160);
00072     channelsView->horizontalHeader()->setSectionResizeMode(0,QHeaderView::Fixed);
00073     channelsView->horizontalHeader()->setSectionResizeMode(2,QHeaderView::Fixed);
00074     channelsView->horizontalHeader()->setSectionResizeMode(3,QHeaderView::Stretch);
00075     channelsView->horizontalHeader()->setSectionResizeMode(4,QHeaderView::Stretch);
00076     channelsView->setColumnHidden(1,true);
00077     channelsView->setColumnHidden(5,true);
00078     channelsView->setColumnHidden(6,true);
00079     channelsView->setItemDelegateForColumn(2,new RichTextDelegate());
00080     channelsView->setItemDelegateForColumn(3,new RichTextDelegate());
00081     channelsView->setItemDelegateForColumn(4,new RichTextDelegate());
00082     channelsView->setSelectionBehavior(QAbstractItemView::SelectRows);
00083     channelsView->setSelectionMode(QAbstractItemView::SingleSelection);
00084     channelsView->setEditTriggers(QAbstractItemView::NoEditTriggers);
00085     channelsView->setShowGrid(false);
00086
00087     connect (channelsView->selectionModel(),SIGNAL(selectionChanged(QItemSelection,QItemSelection)),this,SLOT(updateDetails()));
00088
00089     channelDetails=new ChannelDetails(this);
00090     channelDetails->hide();
00091     connect (channelDetails->reducedWidthText,SIGNAL(textEdited(const
00092     QString&)),this,SLOT(updateReducedWidth(const QString&)));
00093
00094     addLevelButton = new QPushButton(tr("+"));
00095     addLevelButton->setMaximumSize(28,28);
00096     connect (addLevelButton,SIGNAL(clicked()),this,SLOT(addLevel()));
00097     removeLevelButton = new QPushButton(tr("-"));
00098     removeLevelButton->setMaximumSize(28,28);
00099     removeLevelButton->setEnabled(false);
00100     connect (removeLevelButton,SIGNAL(clicked()),this,SLOT(removeLevel()));
00101
00102     /*
00103     mapper = new QSignalMapper(this);
00104     connect (mapper,SIGNAL(mapped(int)),this,SLOT(showInfo(int)));
00105
00106     infoButton[0] = new QPushButton(this);
00107     infoButton[0]->setMaximumSize(28,28);
00108     infoButton[0]->setIcon(style()->standardIcon(QStyle::SP_MessageBoxInformation));
00109     mapper->setMapping (infoButton[0],1);
00110     connect (infoButton[0],SIGNAL(clicked()),mapper,SLOT(map()));
00111     */
00112
00113     QGridLayout *buttonBox = new QGridLayout;
00114     buttonBox->addWidget (addLevelButton,0,0);
00115     buttonBox->addWidget (removeLevelButton,0,1);
00116     buttonBox->addItem (new QSpacerItem(28,28),0,2);
00117     //buttonBox->addWidget (infoButton[0],0,3);
00118     buttonBox->setColumnStretch(0,0);
00119     buttonBox->setColumnStretch(1,0);
00120     buttonBox->setColumnStretch(2,1);
00121     buttonBox->setColumnStretch(3,0);

```

```

00120 #ifdef MACX_SPACING
00121     buttonBox->setHorizontalSpacing(11);
00122 #else
00123     buttonBox->setHorizontalSpacing(0);
00124 #endif
00125
00126     QGroupBox *levelsBox=new QGroupBox(tr("Compound Nucleus Levels"));
00127     QGridLayout *levelsLayout=new QGridLayout;
00128     levelsLayout->setContentsMargins(6,6,6,6);
00129     levelsLayout->addWidget(levelsView,0,0);
00130     levelsLayout->addLayout(buttonBox,1,0);
00131 #ifdef MACX_SPACING
00132     levelsLayout->setVerticalSpacing(0);
00133 #endif
00134     levelsBox->setLayout(levelsLayout);
00135
00136     QGroupBox *configBox = new QGroupBox(tr("Channel Configuration"));
00137     QGridLayout *configLayout = new QGridLayout;
00138     configLayout->addWidget(maxLSpin,0,0);
00139     configLayout->addWidget(maxLLabel,0,1);
00140     configLayout->addWidget(maxMultSpin,1,0);
00141     configLayout->addWidget(maxMultLabel,1,1);
00142     configLayout->addWidget(maxNumMultSpin,2,0);
00143     configLayout->addWidget(maxNumMultLabel,2,1);
00144     configLayout->setColumnStretch(0,0);
00145     configLayout->setColumnStretch(1,1);
00146     configBox->setLayout(configLayout);
00147
00148     QGroupBox *channelsBox=new QGroupBox(tr("Channels In Selected Level"));
00149     QGridLayout *channelsLayout=new QGridLayout;
00150     channelsLayout->setContentsMargins(6,6,6,6);
00151     channelsLayout->addWidget(channelsView,0,0);
00152 #ifdef MACX_SPACING
00153     channelsLayout->addItem(new QSpacerItem(40,40),1,0);
00154     channelsLayout->setVerticalSpacing(0);
00155 #else
00156     channelsLayout->addItem(new QSpacerItem(34,34),1,0);
00157 #endif
00158     channelsBox->setLayout(channelsLayout);
00159
00160     QGridLayout *detailsBox = new QGridLayout;
00161     QLabel *detailsLabel = new QLabel(tr("Channel Details (select from list to view):"));
00162     detailsLabel->setAlignment(Qt::AlignHCenter);
00163     detailsBox->addWidget(detailsLabel,0,0);
00164     detailsBox->addWidget(channelDetails,1,0);
00165     detailsBox->setRowStretch(0,0);
00166     detailsBox->setRowStretch(1,1);
00167
00168     QGridLayout *mainLayout = new QGridLayout;
00169     mainLayout->addWidget(levelsBox,0,0,2,1);
00170     mainLayout->addWidget(channelsBox,0,1,2,1);
00171     mainLayout->addWidget(configBox,0,2,1,1);
00172     mainLayout->addWidget(detailsBox,1,2,1,1);
00173     mainLayout->setColumnStretch(0,1);
00174     mainLayout->setColumnStretch(1,1);
00175     mainLayout->setColumnStretch(2,1);
00176
00177     setLayout(mainLayout);
00178 }
00179
00180 void LevelsTab::setPairsModel(PairsModel *model) {
00181     pairsModel=model;
00182     channelsModel->setPairsModel(model);
00183 }
00184
00185 void LevelsTab::addLevel() {
00186     AddLevelDialog aDialog;
00187     if(aDialog.exec()) {
00188         LevelsData newLevel;
00189         newLevel.isActive=1;
00190         newLevel.isFixed=0;
00191         newLevel.jValue=(aDialog.jValueText->text()).toDouble();
00192         if(aDialog.piValueCombo->currentIndex()==0) newLevel.piValue=-1;
00193         else newLevel.piValue=1;
00194         newLevel.energy=(aDialog.energyText->text()).toDouble();
00195         addLevel(newLevel,false);
00196     }
00197 }
00198
00199 void LevelsTab::addLevel(LevelsData level, bool fromFile) {
00200     QList<LevelsData> levels = levelsModel->getLevels();
00201     if(levelsModel->isLevel(level)==-1) {
00202         levelsModel->insertRows(levels.size(),1,QModelIndex());
00203         QModelIndex index = levelsModel->index(levels.size(),0,QModelIndex());
00204         levelsModel->setData(index,level.isActive,Qt::EditRole);
00205         index = levelsModel->index(levels.size(),1,QModelIndex());
00206         levelsModel->setData(index,level.isFixed,Qt::EditRole);

```

```

00207     index = levelsModel->index(levels.size(), 2, QModelIndex());
00208     levelsModel->setData(index, level.jValue, Qt::EditRole);
00209     index = levelsModel->index(levels.size(), 3, QModelIndex());
00210     levelsModel->setData(index, level.piValue, Qt::EditRole);
00211     index = levelsModel->index(levels.size(), 4, QModelIndex());
00212     levelsModel->setData(index, level.energy, Qt::EditRole);
00213     levelsView->resizeRowsToContents();
00214     if(!fromFile) updateChannelsLevelAdded(levels.size());
00215 } else {
00216     QMessageBox::information(this, tr("Duplicate Level"), tr("This level already exists."));
00217 }
00218 }
00219
00220 void LevelsTab::removeLevel() {
00221     QItemSelectionModel *selectionModel = levelsView->selectionModel();
00222     QModelIndexList indexes = selectionModel->selectedRows();
00223     QModelIndex index=levelsModelProxy->mapToSource(indexes[0]);
00224
00225     levelsModel->removeRows(index.row(), 1, QModelIndex());
00226     updateChannelsLevelDeleted(index.row());
00227     selectionModel->clearSelection();
00228 }
00229
00230 void LevelsTab::editLevel() {
00231     QItemSelectionModel *selectionModel = levelsView->selectionModel();
00232     QModelIndexList indexes = selectionModel->selectedRows();
00233     QModelIndex index=levelsModelProxy->mapToSource(indexes[0]);
00234
00235     QModelIndex i=levelsModel->index(index.row(), 2, QModelIndex());
00236     QVariant var=levelsModel->data(i, Qt::EditRole);
00237     QString jValue=var.toString();
00238     i=levelsModel->index(index.row(), 3, QModelIndex());
00239     var=levelsModel->data(i, Qt::EditRole);
00240     int piValue=var.toInt();
00241     i=levelsModel->index(index.row(), 4, QModelIndex());
00242     var=levelsModel->data(i, Qt::EditRole);
00243     QString energy=var.toString();
00244
00245     AddLevelDialog aDialog;
00246     aDialog.setWindowTitle(tr("Edit a Level"));
00247     aDialog.jValueText->setText(jValue);
00248     if(piValue==1) aDialog.piValueCombo->setCurrentIndex(0);
00249     else aDialog.piValueCombo->setCurrentIndex(1);
00250     aDialog.energyText->setText(energy);
00251
00252     if(aDialog.exec()) {
00253         QString newJValue=aDialog.jValueText->text();
00254         if(newJValue!=jValue) {
00255             i=levelsModel->index(index.row(), 2, QModelIndex());
00256             levelsModel->setData(i, newJValue, Qt::EditRole);
00257         }
00258         int newPiValue;
00259         if(aDialog.piValueCombo->currentIndex()==0) newPiValue=-1;
00260         else newPiValue =1;
00261         if(newPiValue!=piValue) {
00262             i=levelsModel->index(index.row(), 3, QModelIndex());
00263             levelsModel->setData(i, newPiValue, Qt::EditRole);
00264         }
00265         QString newEnergy=aDialog.energyText->text();
00266         if(newEnergy!=energy) {
00267             i=levelsModel->index(index.row(), 4, QModelIndex());
00268             levelsModel->setData(i, newEnergy, Qt::EditRole);
00269         }
00270         updateChannelsLevelEdited(index.row());
00271     }
00272 }
00273
00274 void LevelsTab::updateButtons(const QItemSelection &selection) {
00275     QModelIndexList indexes=selection.indexes();
00276
00277     if(indexes.isEmpty()) {
00278         removeLevelButton->setEnabled(false);
00279     } else {
00280         removeLevelButton->setEnabled(true);
00281     }
00282 }
00283
00284 void LevelsTab::updateChannelsLevelAdded(int levelIndex) {
00285     QList<ChannelsData> newChannels = calculateChannels(levelIndex);
00286     channelsModel->insertRows(0, newChannels.size(), QModelIndex());
00287     for(int i=0; i<newChannels.size(); i++) {
00288         QModelIndex index = channelsModel->index(i, 0, QModelIndex());
00289         channelsModel->setData(index, newChannels.at(i).isFixed, Qt::EditRole);
00290         index = channelsModel->index(i, 1, QModelIndex());
00291         channelsModel->setData(index, newChannels.at(i).levelIndex, Qt::EditRole);
00292         index = channelsModel->index(i, 2, QModelIndex());
00293         channelsModel->setData(index, newChannels.at(i).pairIndex, Qt::EditRole);

```

```

00294     index = channelsModel->index(i,3,QModelIndex());
00295     channelsModel->setData(index,newChannels.at(i).sValue,Qt::EditRole);
00296     index = channelsModel->index(i,4,QModelIndex());
00297     channelsModel->setData(index,newChannels.at(i).lValue,Qt::EditRole);
00298     index = channelsModel->index(i,5,QModelIndex());
00299     channelsModel->setData(index,newChannels.at(i).radType,Qt::EditRole);
00300     index = channelsModel->index(i,6,QModelIndex());
00301     channelsModel->setData(index,newChannels.at(i).reducedWidth,Qt::EditRole);
00302 }
00303 channelsView->resizeRowsToContents();
00304 }
00305
00306 void LevelsTab::updateChannelsLevelDeleted(int levelIndex) {
00307     QList<ChannelsData> channels=channelsModel->getChannels();
00308     int deleted=0;
00309     for(int i=0;i<channels.size();i++) {
00310         if(channels.at(i).levelIndex==levelIndex) {
00311             channelsModel->removeRows(i-deleted,1,QModelIndex());
00312             deleted++;
00313         }
00314     }
00315     channels=channelsModel->getChannels();
00316     for(int i=0;i<channels.size();i++) {
00317         if(channels.at(i).levelIndex>levelIndex) {
00318             QModelIndex index=channelsModel->index(i,1,QModelIndex());
00319             channelsModel->setData(index,channels.at(i).levelIndex-1,Qt::EditRole);
00320         }
00321     }
00322 }
00323
00324 void LevelsTab::updateChannelsLevelEdited(int levelIndex) {
00325     QList<ChannelsData> channels=channelsModel->getChannels();
00326     int deleted=0;
00327     for(int i=0;i<channels.size();i++) {
00328         if(channels.at(i).levelIndex==levelIndex) {
00329             channelsModel->removeRows(i-deleted,1,QModelIndex());
00330             deleted++;
00331         }
00332     }
00333     QList<ChannelsData> newChannels = calculateChannels(levelIndex);
00334     channelsModel->insertRows(0,newChannels.size(),QModelIndex());
00335     for(int i=0;i<newChannels.size();i++) {
00336         QModelIndex index = channelsModel->index(i,1,QModelIndex());
00337         channelsModel->setData(index,newChannels.at(i).levelIndex,Qt::EditRole);
00338         index = channelsModel->index(i,2,QModelIndex());
00339         channelsModel->setData(index,newChannels.at(i).pairIndex,Qt::EditRole);
00340         index = channelsModel->index(i,3,QModelIndex());
00341         channelsModel->setData(index,newChannels.at(i).sValue,Qt::EditRole);
00342         index = channelsModel->index(i,4,QModelIndex());
00343         channelsModel->setData(index,newChannels.at(i).lValue,Qt::EditRole);
00344         index = channelsModel->index(i,5,QModelIndex());
00345         channelsModel->setData(index,newChannels.at(i).radType,Qt::EditRole);
00346         for(int ii=0;ii<channels.size();ii++) {
00347             if(channels.at(ii).levelIndex==newChannels.at(i).levelIndex&&
00348                channels.at(ii).pairIndex==newChannels.at(i).pairIndex&&
00349                channels.at(ii).sValue==newChannels.at(i).sValue&&
00350                channels.at(ii).lValue==newChannels.at(i).lValue&&
00351                channels.at(ii).radType==newChannels.at(i).radType) {
00352                 newChannels[i].isFixed=channels.at(ii).isFixed;
00353                 newChannels[i].reducedWidth=channels.at(ii).reducedWidth;
00354                 break;
00355             }
00356         }
00357         index = channelsModel->index(i,6,QModelIndex());
00358         channelsModel->setData(index,newChannels.at(i).reducedWidth,Qt::EditRole);
00359         index = channelsModel->index(i,0,QModelIndex());
00360         channelsModel->setData(index,newChannels.at(i).isFixed,Qt::EditRole);
00361     }
00362     channelsView->resizeRowsToContents();
00363 }
00364
00365 QList<ChannelsData> LevelsTab::calculateChannels(int levelIndex) {
00366     QList<ChannelsData> channels;
00367     QList<PairsData> pairs = pairsModel->getPairs();
00368     LevelsData level = (levelsModel->getLevels()).at(levelIndex);
00369
00370     int maxL=maxLSpin->value();
00371     int maxMult=maxMultSpin->value();
00372     int maxNumMult=maxNumMultSpin->value();
00373
00374     for(int i=0;i<pairs.size();i++) {
00375         PairsData pair=pairs.at(i);
00376         if(pair.pairType==0) {
00377             for(double s=fabs(pair.heavyJ-pair.lightJ);s<=pair.heavyJ+pair.lightJ;s+=1.0) {
00378                 for(double l=fabs(s-level.jValue);l<=s+level.jValue;l+=1.0) {
00379                     if(int(l*2.0)%2==0&&pair.lightPi*pair.heavyPi*pow(-1,int(l))==level.piValue&&int(l)<=maxL) {
00380                         ChannelsData channel={0,levelIndex,i,s,int(l),'P',0.0};

```

```

00381         channels.push_back(channel);
00382     }
00383 }
00384 }
00385 } else if (pair.pairType==20) {
00386     if (fabs(pair.heavyJ-level.jValue)==0.&&pair.heavyPi==level.piValue) {
00387         ChannelsData gtChannel = {0,levelIndex,i,0.,1,'G',0.0};
00388         ChannelsData fChannel = {0,levelIndex,i,0.,0,'F',0.0};
00389         channels.push_back(gtChannel);
00390         channels.push_back(fChannel);
00391     } else if (fabs(pair.heavyJ-level.jValue)==1.&&pair.heavyPi==level.piValue) {
00392         ChannelsData gtChannel = {0,levelIndex,i,1.,1,'G',0.0};
00393         channels.push_back(gtChannel);
00394     }
00395 } else {
00396     int numMult=1;
00397     for (int l=1;l<=maxMult;l++) {
00398         if (fabs(l-pair.heavyJ)<=level.jValue&&level.jValue<=l+pair.heavyJ&&numMult<=maxNumMult) {
00399             QChar radType;
00400             int parityChange=pair.heavyPi*level.piValue;
00401             if (l%2!=0) {
00402                 if (parityChange==1) radType='E';
00403                 else radType='M';
00404             } else {
00405                 if (parityChange==1) radType='M';
00406                 else radType='E';
00407             }
00408             ChannelsData channel={0,levelIndex,i,pair.heavyJ,l,radType,0.0};
00409             channels.push_back(channel);
00410             numMult++;
00411         }
00412     }
00413 }
00414 }
00415 return channels;
00416 }
00417
00418 void LevelsTab::updateFilter(const QItemSelection &selection) {
00419     QModelIndexList indexes = selection.indexes();
00420     if (indexes.isEmpty())
00421         proxyModel->setFilterRegExp("-1");
00422     else {
00423         QModelIndex index = levelsModelProxy->mapToSource(indexes.at(0));
00424         int row=index.row();
00425         proxyModel->setFilterRegExp(QString("\\b%1\\b").arg(row));
00426     }
00427     channelsView->resizeRowsToContents();
00428 }
00429
00430 void LevelsTab::updateChannelsPairAddedEdited() {
00431     QList<LevelsData> levels=levelsModel->getLevels();
00432     QList<ChannelsData> channels=channelsModel->getChannels();
00433     channelsModel->removeRows(0,channels.size(),QModelIndex());
00434     for (int levelIndex=0;levelIndex<levels.size();levelIndex++) {
00435         QList<ChannelsData> newChannels=calculateChannels(levelIndex);
00436         channelsModel->insertRows(0,newChannels.size(),QModelIndex());
00437         for (int i=0;i<newChannels.size();i++) {
00438             QModelIndex index = channelsModel->index(i,1,QModelIndex());
00439             channelsModel->setData(index,newChannels.at(i).levelIndex,Qt::EditRole);
00440             index = channelsModel->index(i,2,QModelIndex());
00441             channelsModel->setData(index,newChannels.at(i).pairIndex,Qt::EditRole);
00442             index = channelsModel->index(i,3,QModelIndex());
00443             channelsModel->setData(index,newChannels.at(i).sValue,Qt::EditRole);
00444             index = channelsModel->index(i,4,QModelIndex());
00445             channelsModel->setData(index,newChannels.at(i).lValue,Qt::EditRole);
00446             index = channelsModel->index(i,5,QModelIndex());
00447             channelsModel->setData(index,newChannels.at(i).radType,Qt::EditRole);
00448             for (int ii=0;ii<channels.size();ii++) {
00449                 if (channels.at(ii).levelIndex==newChannels.at(i).levelIndex&&
00450                     channels.at(ii).pairIndex==newChannels.at(i).pairIndex&&
00451                     channels.at(ii).sValue==newChannels.at(i).sValue&&
00452                     channels.at(ii).lValue==newChannels.at(i).lValue&&
00453                     channels.at(ii).radType==newChannels.at(i).radType) {
00454                     newChannels[i].reducedWidth=channels.at(ii).reducedWidth;
00455                     newChannels[i].isFixed=channels.at(ii).isFixed;
00456                     break;
00457                 }
00458             }
00459             index = channelsModel->index(i,6,QModelIndex());
00460             channelsModel->setData(index,newChannels.at(i).reducedWidth,Qt::EditRole);
00461             index = channelsModel->index(i,0,QModelIndex());
00462             channelsModel->setData(index,newChannels.at(i).isFixed,Qt::EditRole);
00463         }
00464     }
00465     channelsView->resizeRowsToContents();
00466 }
00467

```



```

00468 void LevelsTab::updateChannelsPairRemoved(int pairIndex) {
00469     QList<ChannelsData> channels=channelsModel->getChannels();
00470     int deleted=0;
00471     for(int i=0;i<channels.size();i++) {
00472         if(channels.at(i).pairIndex==pairIndex) {
00473             channelsModel->removeRows(i-deleted,1,QModelIndex());
00474             deleted++;
00475         }
00476     }
00477     channels=channelsModel->getChannels();
00478     for(int i=0;i<channels.size();i++) {
00479         if(channels.at(i).pairIndex>pairIndex) {
00480             QModelIndex index=channelsModel->index(i,2,QModelIndex());
00481             channelsModel->setData(index,channels.at(i).pairIndex-1,Qt::EditRole);
00482         }
00483     }
00484     channelsView->resizeRowsToContents();
00485 }
00486
00487 void LevelsTab::updateDetails(const QItemSelection &selection) {
00488     QModelIndexList indexes=selection.indexes();
00489
00490     if(indexes.isEmpty()) {
00491         channelDetails->hide();
00492     } else {
00493         QModelIndex index=proxyModel->mapToSource(indexes.at(0));
00494
00495         QModelIndex i=channelsModel->index(index.row(),1,QModelIndex());
00496         QVariant var=channelsModel->data(i,Qt::EditRole);
00497         int levelIndex = var.toInt();
00498         i=channelsModel->index(index.row(),2,QModelIndex());
00499         var=channelsModel->data(i,Qt::EditRole);
00500         int pairIndex = var.toInt();
00501
00502         ChannelsData channel=channelsModel->getChannels().at(index.row());
00503         LevelsData level=levelsModel->getLevels().at(levelIndex);
00504         PairsData pair=pairsModel->getPairs().at(pairIndex);
00505
00506         QString details="";
00507         QTextStream stm(&details,QIODevice::Append);
00508         stm << QString("%1 MeV level with spin %2\n" transitioning via pair key
00509             #3").arg(level.energy).arg(levelsModel->getSpinLabel(level)).arg(pairIndex+1)
00510         << endl;
00511         if(channel.radType=='P') {
00512             stm << QString("Channel configuration is\n" s = %1, l =
00513             %2").arg(channelsModel->getSpinLabel(channel)).arg(channel.lValue)
00514             << endl << endl;
00515             stm << qSetFieldWidth(21) << right << "Light Particle Spin: "
00516             << qSetFieldWidth(0) << left << QString("%1").arg(pairsModel->getSpinLabel(pair,0)) << endl;
00517             stm << qSetFieldWidth(21) << right << "Light Particle Z: "
00518             << qSetFieldWidth(0) << left << QString("%1").arg(pair.lightZ) << endl;
00519             stm << qSetFieldWidth(21) << right << "Light Particle M: "
00520             << qSetFieldWidth(0) << left << QString("%1").arg(pair.lightM) << endl;
00521             stm << qSetFieldWidth(21) << right << "Light Particle G: "
00522             << qSetFieldWidth(0) << left << QString("%1").arg(pair.lightG) << endl;
00523         } else if(channel.radType=='G' || channel.radType=='F') {
00524             if(channel.radType=='G')
00525                 stm << QString("Channel is Gamow-Teller beta decay") << endl << endl;
00526             else
00527                 stm << QString("Channel is Fermi beta decay") << endl << endl;
00528             stm << qSetFieldWidth(21) << right << "Fermion Charge: "
00529             << qSetFieldWidth(0) << left << QString("%1").arg(pair.lightZ) << endl;
00530         } else {
00531             stm << QString("Capture gamma is %1%2 radiation").arg(channel.radType).arg(channel.lValue) <<
00532             endl;
00533             if(((channel.radType=='E' && channel.lValue==1) &&
00534                 (pair.ecMultMask & (1<<0))) ||
00535                 ((channel.radType=='M' && channel.lValue==1) &&
00536                 (pair.ecMultMask & (1<<1))) ||
00537                 ((channel.radType=='E' && channel.lValue==2) &&
00538                 (pair.ecMultMask & (1<<2))))
00539                 stm << "Capture is internal and external" << endl;
00540             else stm << "Capture is internal only" << endl;
00541             stm << endl;
00542             stm << qSetFieldWidth(21) << right << "Heavy Particle Spin: "
00543             << qSetFieldWidth(0) << left << QString("%1").arg(pairsModel->getSpinLabel(pair,1)) << endl;
00544             stm << qSetFieldWidth(21) << right << "Heavy Particle Z: "
00545             << qSetFieldWidth(0) << left << QString("%1").arg(pair.heavyZ) << endl;
00546             stm << qSetFieldWidth(21) << right << "Heavy Particle M: "
00547             << qSetFieldWidth(0) << left << QString("%1").arg(pair.heavyM) << endl;
00548             stm << qSetFieldWidth(21) << right << "Heavy Particle G: "
00549             << qSetFieldWidth(0) << left << QString("%1").arg(pair.heavyG) << endl;
00550             if(channel.radType!='G' && channel.radType!='F')
00551                 stm << qSetFieldWidth(21) << right << "Excitation Energy: "
00552                 << qSetFieldWidth(0) << left << QString("%1").arg(pair.excitationEnergy) << endl;
00553             if(channel.radType=='M' && channel.radType!='E') {

```



```

00552         stm << qSetFieldWidth(21) << right << "Separation Energy: "
00553         << qSetFieldWidth(0) << left << QString("%1").arg(pair.seperationEnergy) << endl;
00554         stm << qSetFieldWidth(21) << right << "Channel Radius: "
00555         << qSetFieldWidth(0) << left << QString("%1").arg(pair.channelRadius) << endl;
00556     }
00557     stm.flush();
00558     channelDetails->details->setText(details);
00559     if (level.energy < (pair.seperationEnergy + pair.excitationEnergy) && pair.pairType == 0)
        channelDetails->setNormParam(1);
00560     else if (pair.pairType == 10 && level.energy == pair.excitationEnergy && level.jValue == pair.heavyJ &&
00561             level.piValue == pair.heavyPi && channel.radType == 'M' && channel.lValue == 1)
        channelDetails->setNormParam(2);
00562     else if (pair.pairType == 10 && level.energy == pair.excitationEnergy && level.jValue == pair.heavyJ &&
00563             level.piValue == pair.heavyPi && channel.radType == 'E' && channel.lValue == 2)
        channelDetails->setNormParam(3);
00564     else if (pair.pairType == 20) channelDetails->setNormParam(4);
00565     else channelDetails->setNormParam(0);
00566     channelDetails->reducedWidthText->setText(QString("%1").arg(channel.reducedWidth));
00567     channelDetails->show();
00570 }
00571 }
00572
00573 void LevelsTab::updateReducedWidth(const QString &string) {
00574     QItemSelectionModel *selectionModel = channelsView->selectionModel();
00575     QModelIndex index = proxyModel->mapToSource(selectionModel->selectedRows().at(0));
00576
00577     if (index.isValid()) {
00578         double reducedWidth = string.toDouble();
00579         QModelIndex i = channelsModel->index(index.row(), 6, QModelIndex());
00580         channelsModel->setData(i, reducedWidth, Qt::EditRole);
00581     }
00582 }
00583
00584 bool LevelsTab::writeNuclearFile(QTextStream& outStream) {
00585     QList<PairsData> pairs = pairsModel->getPairs();
00586     QList<LevelsData> levels = levelsModel->getLevels();
00587     QList<ChannelsData> channels = channelsModel->getChannels();
00588
00589     outStream.setFieldAlignment(QTextStream::AlignRight);
00590
00591     double lowJ = 0;
00592     double highJ = 0;
00593     for (int la = 0; la < levels.size(); la++) {
00594         double tempJ = levels.at(la).jValue;
00595         if (la == 0) {
00596             lowJ = tempJ;
00597             highJ = tempJ;
00598         } else if (tempJ <= lowJ) lowJ = tempJ;
00599         else if (tempJ >= highJ) highJ = tempJ;
00600     }
00601     QList<QList<LevelsData> > sortedLevels;
00602     QList<QList<int> > levelsMap;
00603     for (double j = lowJ; j <= highJ; j += 0.5) {
00604         for (int pi = -1; pi <= 1; pi += 2) {
00605             for (int la = 0; la < levels.size(); la++) {
00606                 if (levels.at(la).jValue == j && levels.at(la).piValue == pi) {
00607                     if (sortedLevels.size() == 0 || (sortedLevels.at(sortedLevels.size() - 1).at(0).jValue != j ||
00608                         sortedLevels.at(sortedLevels.size() - 1).at(0).piValue != pi)) {
00609                         QList<LevelsData> tempLevelList;
00610                         sortedLevels.append(tempLevelList);
00611                         QList<int> tempKeyList;
00612                         levelsMap.append(tempKeyList);
00613                     }
00614                     if (sortedLevels.at(sortedLevels.size() - 1).size() == 0) {
00615                         sortedLevels[sortedLevels.size() - 1].append(levels.at(la));
00616                         levelsMap[levelsMap.size() - 1].append(la);
00617                     } else for (int mu = 0; mu < sortedLevels.at(sortedLevels.size() - 1).size(); mu++) {
00618                         if (levels.at(la).energy <= sortedLevels.at(sortedLevels.size() - 1).at(mu).energy) {
00619                             sortedLevels[sortedLevels.size() - 1].insert(mu, levels.at(la));
00620                             levelsMap[levelsMap.size() - 1].insert(mu, la);
00621                             break;
00622                         }
00623                     } else if (mu == sortedLevels.at(sortedLevels.size() - 1).size() - 1) {
00624                         sortedLevels[sortedLevels.size() - 1].append(levels.at(la));
00625                         levelsMap[levelsMap.size() - 1].append(la);
00626                         break;
00627                     }
00628                 }
00629             }
00630         }
00631     }
00632 }
00633 int levelId = 1;
00634 for (int i = 0; i < sortedLevels.size(); i++) {
00635     for (int ii = 0; ii < sortedLevels.at(i).size(); ii++) {
00636         for (int ch = 0; ch < channels.size(); ch++) {
00637             if (channels.at(ch).levelIndex == levelsMap.at(i).at(ii)) {

```

```

00638         outStream « qSetFieldWidth(4) « sortedLevels.at(i).at(ii).jValue
00639         « qSetFieldWidth(5) « sortedLevels.at(i).at(ii).piValue
00640         « qSetFieldWidth(13) « sortedLevels.at(i).at(ii).energy
00641         « qSetFieldWidth(5) « sortedLevels.at(i).at(ii).isFixed
00642         « qSetFieldWidth(5) « "1"
00643         « qSetFieldWidth(5) « channels.at(ch).pairIndex+1
00644         « qSetFieldWidth(5) « int(channels.at(ch).sValue*2)
00645         « qSetFieldWidth(5) « int(channels.at(ch).lValue*2)
00646         « qSetFieldWidth(5) « levelId
00647         « qSetFieldWidth(5) « sortedLevels.at(i).at(ii).isActive
00648         « qSetFieldWidth(5) « channels.at(ch).isFixed
00649         « qSetFieldWidth(20) « channels.at(ch).reducedWidth
00650         « qSetFieldWidth(5) « pairs.at(channels.at(ch).pairIndex).lightJ
00651         « qSetFieldWidth(5) « pairs.at(channels.at(ch).pairIndex).lightPi
00652         « qSetFieldWidth(5) « pairs.at(channels.at(ch).pairIndex).heavyJ
00653         « qSetFieldWidth(5) « pairs.at(channels.at(ch).pairIndex).heavyPi
00654         « qSetFieldWidth(13) « pairs.at(channels.at(ch).pairIndex).excitationEnergy
00655         « qSetFieldWidth(8) « pairs.at(channels.at(ch).pairIndex).lightM
00656         « qSetFieldWidth(8) « pairs.at(channels.at(ch).pairIndex).heavyM
00657         « qSetFieldWidth(5) « pairs.at(channels.at(ch).pairIndex).lightZ
00658         « qSetFieldWidth(5) « pairs.at(channels.at(ch).pairIndex).heavyZ
00659         « qSetFieldWidth(13) « pairs.at(0).seperationEnergy
00660         « qSetFieldWidth(13) « pairs.at(channels.at(ch).pairIndex).seperationEnergy
00661         « " 0 0 0.0"
00662         « qSetFieldWidth(6) « pairs.at(channels.at(ch).pairIndex).pairType
00663         « qSetFieldWidth(8) « pairs.at(channels.at(ch).pairIndex).channelRadius
00664         « qSetFieldWidth(13) « pairs.at(channels.at(ch).pairIndex).lightG
00665         « qSetFieldWidth(13) « pairs.at(channels.at(ch).pairIndex).heavyG
00666         « qSetFieldWidth(8) « pairs.at(channels.at(ch).pairIndex).ecMultMask
00667         « qSetFieldWidth(0) « endl;
00668     }
00669 }
00670     outStream « endl;
00671     levelId++;
00672 }
00673 }
00674
00675     outStream.setFieldAlignment(QTextStream::AlignLeft);
00676
00677     return true;
00678 }
00679
00680 bool LevelsTab::readNuclearFile(QTextStream &inStream) {
00681     double levelJ;
00682     int levelPi;
00683     double levelEnergy;
00684     int levelFix;
00685     int aa;
00686     int ir;
00687     double channelsS;
00688     int channelL;
00689     int levelId;
00690     int levelYN;
00691     int channelFix;
00692     double channelReducedWidth;
00693     double lightJ;
00694     int lightPi;
00695     double heavyJ;
00696     int heavyPi;
00697     double excitationEnergy;
00698     double lightM;
00699     double heavyM;
00700     int lightZ;
00701     int heavyZ;
00702     double seperationEnergyIn;
00703     double seperationEnergyOut;
00704     int pairType;
00705     double channelRadius;
00706     double lightG;
00707     double heavyG;
00708     double dummyDouble;
00709     int dummyInt;
00710     int ecMultMask;
00711
00712     maxLSpin->blockSignals(true);
00713     maxMultSpin->blockSignals(true);
00714     maxNumMultSpin->blockSignals(true);
00715     maxLSpin->setValue(0);
00716     maxMultSpin->setValue(0);
00717     maxNumMultSpin->setValue(0);
00718
00719     int maxLValue=0;
00720     int maxMultValue=0;
00721     int maxNumMultValue=0;
00722     bool firstLine=true;
00723     int lastPair;

```

```

00725     int currentPair=0;
00726     int thisNumMult=0;
00727
00728     QString line("");
00729     while(!inStream.atEnd() && line.trimmed() != QString("</levels>")) {
00730         line=inStream.readLine();
00731         if(line.trimmed().isEmpty()) continue;
00732         if(!inStream.atEnd() && line.trimmed() != QString("</levels>")) {
00733             QTextStream in(&line);
00734             in » levelJ » levelPi » levelEnergy » levelFix » aa » ir » channels » channelL » levelId »
levelYN » channelFix
00735             » channelReducedWidth » lightJ » lightPi » heavyJ » heavyPi » excitationEnergy
00736             » lightM » heavyM » lightZ » heavyZ » seperationEnergyIn » seperationEnergyOut
00737             » dummyInt » dummyInt » dummyDouble » pairType » channelRadius » lightG » heavyG;
00738             if(in.status() != QTextStream::Ok) return false;
00739             in » ecMultMask;
00740             if(in.status() != QTextStream::Ok) ecMultMask=0;
00741             if(firstLine) {
00742                 lastPair=ir;
00743                 firstLine=false;
00744             } else lastPair=currentPair;
00745             currentPair=ir;
00746             if(lastPair==currentPair) thisNumMult++;
00747             else thisNumMult=1;
00748             if(pairType==0) {
00749                 if(channelL/2>maxLValue) maxLValue=channelL/2;
00750             } else if (pairType==10) {
00751                 if(channelL/2>maxMultValue) maxMultValue=channelL/2;
00752                 if(thisNumMult>maxNumMultValue) maxNumMultValue=thisNumMult;
00753             }
00754
00755             PairsData newPair={lightJ,lightPi,lightZ,lightM,lightG,heavyJ,heavyPi,heavyZ,heavyM,
00756                             heavyG,excitationEnergy,seperationEnergyOut,channelRadius,pairType,ecMultMask};
00757             int pairIndex=ir-1;
00758             if(pairsModel->numPairs()<ir) {
00759                 emit(readNewPair(newPair,pairIndex,true));
00760             } else if(pairsModel->isPair(newPair) == -1) {
00761                 emit(readExistingPair(newPair,pairIndex,true));
00762             }
00763             LevelsData newLevel = {levelYN,levelFix,levelJ,levelPi,levelEnergy};
00764             int levelIndex = levelsModel->isLevel(newLevel);
00765             if(levelIndex == -1) {
00766                 addLevel(newLevel,true);
00767                 levelIndex = levelsModel->isLevel(newLevel);
00768             }
00769             QChar radType;
00770             if(pairType==0) radType='P';
00771             else if(pairType==20) {
00772                 if(channelL==0) radType = 'F';
00773                 else radType = 'G';
00774             }
00775             else {
00776                 int parityChange=heavyPi*levelPi;
00777                 if((channelL/2)%2!=0) {
00778                     if(parityChange==-1) radType='E';
00779                     else radType='M';
00780                 } else {
00781                     if(parityChange==-1) radType='M';
00782                     else radType='E';
00783                 }
00784             }
00785
00786             int channelIndex=channelsModel->getChannels().size();
00787             channelsModel->insertRows(channelIndex,1,QModelIndex());
00788             QModelIndex index = channelsModel->index(channelIndex,0,QModelIndex());
00789             channelsModel->setData(index,channelFix,Qt::EditRole);
00790             index = channelsModel->index(channelIndex,1,QModelIndex());
00791             channelsModel->setData(index,levelIndex,Qt::EditRole);
00792             index = channelsModel->index(channelIndex,2,QModelIndex());
00793             channelsModel->setData(index,pairIndex,Qt::EditRole);
00794             index = channelsModel->index(channelIndex,3,QModelIndex());
00795             channelsModel->setData(index,channels/2.0,Qt::EditRole);
00796             index = channelsModel->index(channelIndex,4,QModelIndex());
00797             channelsModel->setData(index,channelL/2,Qt::EditRole);
00798             index = channelsModel->index(channelIndex,5,QModelIndex());
00799             channelsModel->setData(index,radType,Qt::EditRole);
00800             index = channelsModel->index(channelIndex,6,QModelIndex());
00801             channelsModel->setData(index,channelReducedWidth,Qt::EditRole);
00802             channelsView->resizeRowsToContents();
00803         }
00804     }
00805     if(line.trimmed() != QString("</levels>")) return false;
00806
00807     if(maxLValue>maxLSpin->value()) maxLSpin->setValue(maxLValue);
00808     if(maxMultValue>maxMultSpin->value()) maxMultSpin->setValue(maxMultValue);
00809     if(maxNumMultValue>maxNumMultSpin->value()) maxNumMultSpin->setValue(maxNumMultValue);
00810     maxLSpin->blockSignals(false);

```

```

00811     maxMultSpin->blockSignals(false);
00812     maxNumMultSpin->blockSignals(false);
00813     levelsView->resizeRowsToContents();
00814
00815     return true;
00816 }
00817
00818 void LevelsTab::reset() {
00819     pairsModel->removeRows(0,pairsModel->getPairs().size(),QModelIndex());
00820     levelsModel->removeRows(0,levelsModel->getLevels().size(),QModelIndex());
00821     channelsModel->removeRows(0,channelsModel->getChannels().size(),QModelIndex());
00822     maxLSpin->blockSignals(true);
00823     maxMultSpin->blockSignals(true);
00824     maxNumMultSpin->blockSignals(true);
00825     maxLSpin->setValue(2);
00826     maxMultSpin->setValue(2);
00827     maxNumMultSpin->setValue(2);
00828     maxLSpin->blockSignals(false);
00829     maxMultSpin->blockSignals(false);
00830     maxNumMultSpin->blockSignals(false);
00831 }
00832
00833 void LevelsTab::showInfo(int which,QString title) {
00834     if(which<infoText.size()) {
00835         if(!infoDialog[which]) {
00836             infoDialog[which] = new InfoDialog(infoText[which],this,title);
00837             infoDialog[which]->setAttribute(Qt::WA_DeleteOnClose);
00838             infoDialog[which]->show();
00839         } else infoDialog[which]->raise();
00840     }
00841 }

```

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.](#)

```
00001 #include <QMainWindow>
00002 #include <QApplication>
00003 #include <QResource>
00004 #include <iostream>
00005 #include "AZURESetup.h"
00006
00007 inline void initResource() { Q_INIT_RESOURCE(AZURESetup); }
00008
00009 int start_gui(int argc, char *argv[]) {
00010
00011     #ifdef Q_OS_MACX
00012         if ( QSysInfo::MacintoshVersion > QSysInfo::MV_10_8 )
00013         {
00014             // fix Mac OS X 10.9 (mavericks) font issue
00015             // https://bugreports.qt-project.org/browse/QTBUG-32789
00016             QFont::insertSubstitution(".Lucida Grande UI", "Lucida Grande");
00017         }
00018     #endif
00019
00020     QApplication app(argc, argv);
00021
00022     #ifdef WIN_SPACING
00023     QFont font = app.font();
00024     font.setPointSizeF(10.5);
00025     app.setFont(font);
00026     #endif
00027
00028     initResource();
00029     QCoreApplication::setOrganizationName("jina");
00030     QCoreApplication::setApplicationName("azure2");
00031
00032     AZURESetup azureSetup;
00033     azureSetup.show();
00034
00035     QString filename="";
00036     for(int i=1;i<argc;i++)
00037         if(strncmp(argv[i],"--",2)!=0) {
00038             filename=argv[i];
00039             break;
00040         }
00041     if(!filename.trimmed().isEmpty()) azureSetup.open(filename);
00042
00043     return app.exec();
00044 }
```

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.](#)

```
00001 #include "PairsModel.h"
00002 #include "ElementMap.h"
00003 #include <math.h>
00004
00005 PairsModel::PairsModel(QObject *parent) : QAbstractTableModel(parent) {
00006 }
00007
00008 int PairsModel::rowCount(const QModelIndex &parent) const {
00009     Q_UNUSED(parent);
00010     return pairsList.size();
00011 }
```

```

00012
00013 int PairsModel::columnCount(const QModelIndex &parent) const {
00014     Q_UNUSED(parent);
00015     return PairsData::SIZE;
00016 }
00017
00018 QVariant PairsModel::data(const QModelIndex &index, int role) const {
00019     if(!index.isValid()) return QVariant();
00020
00021     if(index.row() >= pairsList.size() || index.row() < 0) return QVariant();
00022
00023     if (role == Qt::DisplayRole) {
00024         PairsData pair = pairsList.at(index.row());
00025         if(index.column() == 2) return getSpinLabel(pair,0);
00026         else if(index.column() == 3) return pair.lightPi;
00027         else if(index.column() == 0) return getParticleLabel(pair,0);
00028         else if(index.column() == 1) return pair.lightM;
00029         else if(index.column() == 4) return pair.lightG;
00030         else if(index.column() == 7) return getSpinLabel(pair,1);
00031         else if(index.column() == 8) return pair.heavyPi;
00032         else if(index.column() == 5) return getParticleLabel(pair,1);
00033         else if(index.column() == 6) return pair.heavyM;
00034         else if(index.column() == 9) return pair.heavyG;
00035         else if(index.column() == 10) return pair.excitationEnergy;
00036         else if(index.column() == 11) return pair.seperationEnergy;
00037         else if(index.column() == 12) return pair.channelRadius;
00038         else if(index.column() == 13) {
00039             if(pair.pairType == 10) return "Particle, Gamma";
00040             else if(pair.pairType == 20) return "Beta Decay";
00041             else return "Particle, Particle";
00042         } else if(index.column() == 14) return pair.ecMultMask;
00043     } else if (role == Qt::EditRole) {
00044         PairsData pair = pairsList.at(index.row());
00045         if(index.column() == 0) return pair.lightJ;
00046         else if(index.column() == 1) return pair.lightPi;
00047         else if(index.column() == 2) return pair.lightZ;
00048         else if(index.column() == 3) return pair.lightM;
00049         else if(index.column() == 4) return pair.lightG;
00050         else if(index.column() == 5) return pair.heavyJ;
00051         else if(index.column() == 6) return pair.heavyPi;
00052         else if(index.column() == 7) return pair.heavyZ;
00053         else if(index.column() == 8) return pair.heavyM;
00054         else if(index.column() == 9) return pair.heavyG;
00055         else if(index.column() == 10) return pair.excitationEnergy;
00056         else if(index.column() == 11) return pair.seperationEnergy;
00057         else if(index.column() == 12) return pair.channelRadius;
00058         else if(index.column() == 13) return pair.pairType;
00059         else if(index.column() == 14) return pair.ecMultMask;
00060     } else if(role == Qt::TextAlignmentRole) return Qt::AlignCenter;
00061
00062     return QVariant();
00063 }
00064
00065 QVariant PairsModel::headerData(int section, Qt::Orientation orientation, int role) const {
00066     if(role!= Qt::DisplayRole) return QVariant();
00067     if(orientation == Qt::Horizontal) {
00068         switch(section) {
00069             case 2:
00070                 return tr("Light\nSpin");
00071             case 3:
00072                 return tr("Light\nParity");
00073             case 0:
00074                 return tr("Light\nParticle");
00075             case 1:
00076                 return tr("Light\nM");
00077             case 4:
00078                 return tr("Light\nG-Factor");
00079             case 7:
00080                 return tr("Heavy\nSpin");
00081             case 8:
00082                 return tr("Heavy\nParity");
00083             case 5:
00084                 return tr("Heavy\nParticle");
00085             case 6:
00086                 return tr("Heavy\nM");
00087             case 9:
00088                 return tr("Heavy\nG-Factor");
00089             case 10:
00090                 return tr("Excitation\nEnergy");
00091             case 11:
00092                 return tr("Separation\nEnergy");
00093             case 12:
00094                 return tr("Channel\nRadius");
00095             case 13:
00096                 return tr("Pair\nType");
00097             case 14:
00098                 return tr("EC\nMultipolarities");

```

```

00099     default:
00100         return QVariant();
00101     }
00102 } else if(orientation == Qt::Vertical) {
00103     return section+1;
00104 }
00105 return QVariant();
00106 }
00107
00108 bool PairsModel::setData(const QModelIndex &index, const QVariant &value, int role) {
00109     if (index.isValid() && role == Qt::EditRole ) {
00110         int row = index.row();
00111         PairsData tempData = pairsList.value(row);
00112         if (index.column() == 0) tempData.lightJ=value.toDouble();
00113         else if(index.column() == 1) tempData.lightPi=value.toInt();
00114         else if(index.column() == 2) tempData.lightZ=value.toInt();
00115         else if(index.column() == 3) tempData.lightM=value.toDouble();
00116         else if(index.column() == 4) tempData.lightG=value.toDouble();
00117         else if(index.column() == 5) tempData.heavyJ=value.toDouble();
00118         else if(index.column() == 6) tempData.heavyPi=value.toInt();
00119         else if(index.column() == 7) tempData.heavyZ=value.toInt();
00120         else if(index.column() == 8) tempData.heavyM=value.toDouble();
00121         else if(index.column() == 9) tempData.heavyG=value.toDouble();
00122         else if(index.column() == 10) tempData.excitationEnergy=value.toDouble();
00123         else if(index.column() == 11) tempData.seperationEnergy=value.toDouble();
00124         else if(index.column() == 12) tempData.channelRadius=value.toDouble();
00125         else if(index.column() == 13) tempData.pairType=value.toInt();
00126         else if(index.column() == 14) tempData.ecMultMask=value.toInt();
00127         else return false;
00128
00129         pairsList.replace(row,tempData);
00130         emit(dataChanged(index,index));
00131         return true;
00132     }
00133     return false;
00134 }
00135
00136 bool PairsModel::insertRows(int position, int rows, const QModelIndex &index) {
00137     Q_UNUSED(index);
00138     if(rows>0) {
00139         beginInsertRows(QModelIndex(),position,position+rows-1);
00140         for(int row=0; row<rows; row++) {
00141             PairsData tempData;
00142             pairsList.insert(position,tempData);
00143         }
00144         endInsertRows();
00145     }
00146     return true;
00147 }
00148
00149 bool PairsModel::removeRows(int position, int rows, const QModelIndex &index) {
00150     Q_UNUSED(index);
00151     if(rows>0) {
00152         beginRemoveRows(QModelIndex(),position,position+rows-1);
00153         for(int row=0; row<rows; ++row) {
00154             pairsList.removeAt(position);
00155         }
00156         endRemoveRows();
00157     }
00158     return true;
00159 }
00160
00161 int PairsModel::isPair(const PairsData &pair) const {
00162     int foundPair=-1;
00163     for(int i=0;i<pairsList.size();i++) {
00164         PairsData tempPair=pairsList.value(i);
00165         if(tempPair.lightJ==pair.lightJ&&
00166            tempPair.lightPi==pair.lightPi&&
00167            tempPair.lightZ==pair.lightZ&&
00168            tempPair.lightM==pair.lightM&&
00169            tempPair.lightG==pair.lightG&&
00170            tempPair.heavyJ==pair.heavyJ&&
00171            tempPair.heavyPi==pair.heavyPi&&
00172            tempPair.heavyZ==pair.heavyZ&&
00173            tempPair.heavyM==pair.heavyM&&
00174            tempPair.heavyG==pair.heavyG&&
00175            tempPair.seperationEnergy==pair.seperationEnergy&&
00176            tempPair.excitationEnergy==pair.excitationEnergy&&
00177            tempPair.channelRadius==pair.channelRadius&&
00178            tempPair.pairType==pair.pairType&&
00179            tempPair.ecMultMask==pair.ecMultMask) {
00180             foundPair=i;
00181             break;
00182         }
00183     }
00184     return foundPair;
00185 }

```

```

00186
00187 QString PairsModel::getParticleLabel(const PairsData &pair, int which) const {
00188     if(which!=-1) {
00189         if(pair.pairType==10&&which==0) return "<center>&gamma;</center>";
00190         else if(pair.pairType==20&&which==0) {
00191             if(pair.lightZ<0) return "<center>&beta;<sup>-</sup></center>";
00192             else return "<center>&beta;<sup>+</sup></center>";
00193         } else {
00194             int tempZ;
00195             int tempM;
00196             if(which==0) {
00197                 tempZ=pair.lightZ;
00198                 tempM=round(pair.lightM);
00199             } else {
00200                 tempZ=pair.heavyZ;
00201                 tempM=round(pair.heavyM);
00202             }
00203             std::map<int, QString>::const_iterator it=elementMap.find(tempZ);
00204             if(it!=elementMap.end()) {
00205                 if(tempM==1) {
00206                     if(tempZ==1) return "<center><i>p</i></center>";
00207                     else return QString("<center><i>%1</i></center>").arg(it->second);
00208                 } else if(tempZ==2&&tempM==4) return "<center>&alpha;</center>";
00209                 else return QString("<center><sup>%1</sup>%2</center>").arg(tempM).arg(it->second);
00210                 } else return "?";
00211             }
00212         } else {
00213             QString lightLabel;
00214             std::map<int, QString>::const_iterator it=elementMap.find(pair.lightZ);
00215             if(pair.pairType==10) lightLabel="&gamma;";
00216             else if(pair.pairType==20) {
00217                 if(pair.lightZ<0) lightLabel="&beta;<sup>-</sup>";
00218                 else lightLabel="&beta;<sup>+</sup>";
00219             } else if(it!=elementMap.end()) {
00220                 if(round(pair.lightM)==1) {
00221                     if(pair.lightZ==1) lightLabel="<i>p</i>";
00222                     else lightLabel=QString("<i>%1</i>").arg(it->second);
00223                 } else if(pair.lightZ==2&&round(pair.lightM)==4) lightLabel="&alpha;";
00224                 else lightLabel=QString("<sup>%1</sup>%2").arg(round(pair.lightM)).arg(it->second);
00225                 } else lightLabel="?";
00226             QString heavyLabel;
00227             it=elementMap.find(pair.heavyZ);
00228             if(it!=elementMap.end()) {
00229                 if(round(pair.heavyM)==1) {
00230                     if(pair.heavyZ==1) heavyLabel="<i>p</i>";
00231                     else heavyLabel=QString("<i>%1</i>").arg(it->second);
00232                 } else if(pair.heavyZ==2&&round(pair.heavyM)==4) heavyLabel="&alpha;";
00233                 else heavyLabel=QString("<sup>%1</sup>%2").arg(round(pair.heavyM)).arg(it->second);
00234                 } else heavyLabel="?";
00235             if(pair.pairType==20) return QString("<center>%1%2) [%3
MeV]</center>").arg(heavyLabel).arg(lightLabel).arg(pair.excitationEnergy,0,'f',3);
00236             else return QString("<center>%1+%2 [%3
MeV]</center>").arg(heavyLabel).arg(lightLabel).arg(pair.excitationEnergy,0,'f',3);
00237         }
00238     }
00239
00240 QString PairsModel::getReactionLabel(const PairsData &firstPair, const PairsData &secondPair) {
00241     QString lightLabel[2];
00242     std::map<int, QString>::const_iterator it=elementMap.find(firstPair.lightZ);
00243     if(firstPair.pairType==10) lightLabel[0]="&gamma;";
00244     else if(firstPair.pairType==20) {
00245         if(firstPair.lightZ<0) lightLabel[0]="&beta;<sup>-</sup>";
00246         else lightLabel[0]="&beta;<sup>+</sup>";
00247     } else if(it!=elementMap.end()) {
00248         if(round(firstPair.lightM)==1) {
00249             if(firstPair.lightZ==1) lightLabel[0]="<i>p</i>";
00250             else lightLabel[0]=QString("<i>%1</i>").arg(it->second);
00251         } else if(firstPair.lightZ==2&&round(firstPair.lightM)==4) lightLabel[0]="&alpha;";
00252         else lightLabel[0]=QString("<sup>%1</sup>%2").arg(round(firstPair.lightM)).arg(it->second);
00253         } else lightLabel[0]="?";
00254     it=elementMap.find(secondPair.lightZ);
00255     if(secondPair.pairType==10) lightLabel[1]="&gamma;";
00256     else if(secondPair.pairType==20) {
00257         if(secondPair.lightZ<0) lightLabel[1]="&beta;<sup>-</sup>";
00258         else lightLabel[1]="&beta;<sup>+</sup>";
00259     } else if(it!=elementMap.end()) {
00260         if(round(secondPair.lightM)==1) {
00261             if(secondPair.lightZ==1) lightLabel[1]="<i>p</i>";
00262             else lightLabel[1]=QString("<i>%1</i>").arg(it->second);
00263         } else if(secondPair.lightZ==2&&round(secondPair.lightM)==4) lightLabel[1]="&alpha;";
00264         else lightLabel[1]=QString("<sup>%1</sup>%2").arg(round(secondPair.lightM)).arg(it->second);
00265         } else lightLabel[1]="?";
00266     QString heavyLabel[2];
00267     it=elementMap.find(firstPair.heavyZ);
00268     if(it!=elementMap.end()) {
00269         if(round(firstPair.heavyM)==1) {
00270             if(firstPair.heavyZ==1) heavyLabel[0]="<i>p</i>";

```



```

00271         else heavyLabel[0]=QString("<i>%1</i>").arg(it->second);
00272     } else if (firstPair.heavyZ==2&&round(firstPair.heavyM)==4) heavyLabel[0]="&alpha;";
00273     else heavyLabel[0]=QString("<sup>%1</sup>%2").arg(round(firstPair.heavyM)).arg(it->second);
00274 } else heavyLabel[0]="?";
00275 it=elementMap.find(secondPair.heavyZ);
00276 if(it!=elementMap.end()) {
00277     if(round(secondPair.heavyM)==1) {
00278         if(secondPair.heavyZ==1) heavyLabel[1]="<i>p</i>";
00279         else heavyLabel[1]=QString("<i>%1</i>").arg(it->second);
00280     } else if (secondPair.heavyZ==2&&round(secondPair.heavyM)==4) heavyLabel[1]="&alpha;";
00281     else heavyLabel[1]=QString("<sup>%1</sup>%2").arg(round(secondPair.heavyM)).arg(it->second);
00282 } else heavyLabel[1]="?";
00283 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
00284 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
00285 }
00286
00287 QString PairsModel::getReactionLabelTotalCapture(const PairsData &firstPair) {
00288     QString lightLabel[2];
00289     std::map<int, QString>::const_iterator it=elementMap.find(firstPair.lightZ);
00290     if(firstPair.pairType==10) lightLabel[0]="&gamma;";
00291     else if(firstPair.pairType==20) {
00292         if(firstPair.lightZ<0) lightLabel[0]="&beta;<sup>-</sup>";
00293         else lightLabel[0]="&beta;<sup>+</sup>";
00294     } else if(it!=elementMap.end()) {
00295         if(round(firstPair.lightM)==1) {
00296             if(firstPair.lightZ==1) lightLabel[0]="<i>p</i>";
00297             else lightLabel[0]=QString("<i>%1</i>").arg(it->second);
00298         } else if (firstPair.lightZ==2&&round(firstPair.lightM)==4) lightLabel[0]="&alpha;";
00299         else lightLabel[0]=QString("<sup>%1</sup>%2").arg(round(firstPair.lightM)).arg(it->second);
00300     } else lightLabel[0]="?";
00301     lightLabel[1]="&gamma;";
00302     QString heavyLabel[2];
00303     it=elementMap.find(firstPair.heavyZ);
00304     if(it!=elementMap.end()) {
00305         if(round(firstPair.heavyM)==1) {
00306             if(firstPair.heavyZ==1) heavyLabel[0]="<i>p</i>";
00307             else heavyLabel[0]=QString("<i>%1</i>").arg(it->second);
00308         } else if (firstPair.heavyZ==2&&round(firstPair.heavyM)==4) heavyLabel[0]="&alpha;";
00309         else heavyLabel[0]=QString("<sup>%1</sup>%2").arg(round(firstPair.heavyM)).arg(it->second);
00310     } else heavyLabel[0]="?";
00311     int i;
00312     for(i=0;i<pairsList.size();i++)
00313         if(pairsList[i].pairType==10) break;
00314     if(i==pairsList.size()) heavyLabel[1]="?";
00315     else {
00316         PairsData secondPair = pairsList[i];
00317         it=elementMap.find(secondPair.heavyZ);
00318         if(it!=elementMap.end()) {
00319             if(round(secondPair.heavyM)==1) {
00320                 if(secondPair.heavyZ==1) heavyLabel[1]="<i>p</i>";
00321                 else heavyLabel[1]=QString("<i>%1</i>").arg(it->second);
00322             } else if (secondPair.heavyZ==2&&round(secondPair.heavyM)==4) heavyLabel[1]="&alpha;";
00323             else heavyLabel[1]=QString("<sup>%1</sup>%2").arg(round(secondPair.heavyM)).arg(it->second);
00324         } else heavyLabel[1]="?";
00325     }
00326     if(firstPair.pairType==20) return QString("%1(%2%3)%4
[TOTAL]").arg(heavyLabel[0]).arg(lightLabel[0]).arg(lightLabel[1]).arg(heavyLabel[1]);
00327     else return QString("%1(%2,%3)%4
[TOTAL]").arg(heavyLabel[0]).arg(lightLabel[0]).arg(lightLabel[1]).arg(heavyLabel[1]);
00328 }
00329
00330 QString PairsModel::getSpinLabel(const PairsData &pair, int which) const {
00331     double tempJ;
00332     int tempPi;
00333     if(which==0) {
00334         tempJ=pair.lightJ;
00335         tempPi=pair.lightPi;
00336     } else {
00337         tempJ=pair.heavyJ;
00338         tempPi=pair.heavyPi;
00339     }
00340     QString tempSpin;
00341     if(((int)(tempJ*2))%2!=0&&tempJ!=0.) tempSpin=QString("%1/2").arg((int)(tempJ*2));
00342     else tempSpin=QString("%1").arg(tempJ);
00343     if(tempPi==-1) return QString("%1-").arg(tempSpin);
00344     else return QString("%1+").arg(tempSpin);
00345 }

```

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.](#)

```
00001 #include <QHeaderView>
00002 #include <QGridLayout>
00003 #include <QComboBox>
00004 #include <QLineEdit>
00005 #include <QCheckBox>
00006 #include <QTextStream>
00007 #include <QMessageBox>
00008
00009 #include "PairsTab.h"
00010 #include "RichTextDelegate.h"
00011 #include "InfoDialog.h"
00012 #include <iostream>
00013
00014 PairsTab::PairsTab(QWidget *parent) : QWidget(parent) {
00015     pairsModel = new PairsModel(this);
00016
00017     pairsView = new QTableView;
00018     pairsView->setModel(pairsModel);
00019     pairsView->verticalHeader()->setHighlightSections(false);
00020     pairsView->horizontalHeader()->setHighlightSections(false);
00021     pairsView->setColumnHidden(1,true);
00022     pairsView->setColumnHidden(3,true);
00023     pairsView->setColumnHidden(4,true);
00024     pairsView->setColumnHidden(6,true);
00025     pairsView->setColumnHidden(8,true);
00026     pairsView->setColumnHidden(9,true);
00027     pairsView->setColumnHidden(13,true);
00028     pairsView->setColumnHidden(14,true);
00029     RichTextDelegate *rt = new RichTextDelegate();
00030     pairsView->setItemDelegateForColumn(0,rt);
00031     pairsView->setItemDelegateForColumn(5,rt);
00032     pairsView->setSelectionBehavior(QAbstractItemView::SelectRows);
00033     pairsView->setSelectionMode(QAbstractItemView::SingleSelection);
00034     pairsView->setEditTriggers(QAbstractItemView::NoEditTriggers);
00035     pairsView->setShowGrid(false);
00036
00037     connect(pairsView->selectionModel(), SIGNAL(selectionChanged(QItemSelection,QItemSelection)), this, SLOT(updateButtons(QItemSelection,QItemSelection)));
00038     connect(pairsView, SIGNAL(doubleClicked(QModelIndex)), this, SLOT(editPair(QModelIndex)));
00039
00040     for(int i = 0; i<PairsData::SIZE;i++)
00041         pairsView->horizontalHeader()->setSectionResizeMode(i,QHeaderView::Stretch);
00042
00043     addButton=new QPushButton(tr("+"));
00044     addButton->setMaximumSize(28,28);
00045     connect(addButton, SIGNAL(clicked()), this, SLOT(addPair()));
00046     deleteButton = new QPushButton(tr("-"));
00047     deleteButton->setMaximumSize(28,28);
00048     deleteButton->setEnabled(false);
00049     connect(deleteButton, SIGNAL(clicked()), this, SLOT(removePair()));
00050
00051     /*
00052     mapper = new QSignalMapper(this);
```

```

00052     connect (mapper, SIGNAL (mapped (int)), this, SLOT (showInfo (int)));
00053
00054     infoButton[0] = new QPushButton (this);
00055     infoButton[0]->setMaximumSize (28, 28);
00056     infoButton[0]->setIcon (style()->standardIcon (QStyle::SP_MessageBoxInformation));
00057     mapper->setMapping (infoButton[0], 0);
00058     connect (infoButton[0], SIGNAL (clicked()), mapper, SLOT (map()));
00059     */
00060
00061     QGridLayout *buttonBox = new QGridLayout;
00062     buttonBox->addWidget (addButton, 0, 0);
00063     buttonBox->addWidget (deleteButton, 0, 1);
00064     buttonBox->addItem (new QSpacerItem (28, 28), 0, 2);
00065     // buttonBox->addWidget (infoButton[0], 0, 3);
00066     buttonBox->setColumnStretch (0, 0);
00067     buttonBox->setColumnStretch (1, 0);
00068     buttonBox->setColumnStretch (2, 1);
00069     buttonBox->setColumnStretch (3, 0);
00070 #ifdef MACX_SPACING
00071     buttonBox->setHorizontalSpacing (11);
00072 #else
00073     buttonBox->setHorizontalSpacing (0);
00074 #endif
00075
00076     QVBoxLayout *layout = new QVBoxLayout ();
00077     layout->addWidget (pairsView);
00078     layout->addLayout (buttonBox);
00079     setLayout (layout);
00080 }
00081
00082 PairsModel *PairsTab::getPairsModel () {
00083     return pairsModel;
00084 }
00085
00086 void PairsTab::addPair () {
00087     AddPairDialog aDialog;
00088     if (aDialog.exec ()) {
00089         if (! (pairsModel->getPairs ().size ()==0 && aDialog.pairTypeCombo->currentIndex ()!=0)) {
00090             PairsData newPair;
00091             newPair.lightJ = (aDialog.lightJText->text ().toDouble ());
00092             if (aDialog.lightPiCombo->currentIndex () == 0) newPair.lightPi=-1;
00093             else newPair.lightPi=1;
00094             newPair.lightZ = (aDialog.lightZText->text ().toInt ());
00095             newPair.lightM = (aDialog.lightMText->text ().toDouble ());
00096             newPair.lightG=0; // (aDialog.lightGText->text ().toDouble ());
00097             newPair.heavyJ = (aDialog.heavyJText->text ().toDouble ());
00098             if (aDialog.heavyPiCombo->currentIndex () == 0) newPair.heavyPi=-1;
00099             else newPair.heavyPi=1;
00100             newPair.heavyZ = (aDialog.heavyZText->text ().toInt ());
00101             newPair.heavyM = (aDialog.heavyMText->text ().toDouble ());
00102             newPair.heavyG=0; // (aDialog.heavyGText->text ().toDouble ());
00103             newPair.seperationEnergy = (aDialog.seperationEnergyText->text ().toDouble ());
00104             newPair.excitationEnergy = (aDialog.excitationEnergyText->text ().toDouble ());
00105             newPair.channelRadius = (aDialog.channelRadiusText->text ().toDouble ());
00106             if (aDialog.pairTypeCombo->currentIndex () == 1) newPair.pairType=10;
00107             else if (aDialog.pairTypeCombo->currentIndex () == 2) newPair.pairType=20;
00108             else newPair.pairType=0;
00109             unsigned int newMask=0;
00110             if (aDialog.elCheck->isChecked ()) newMask |= (1<0);
00111             //if (aDialog.m1Check->isChecked ()) newMask |= (1<1);
00112             if (aDialog.e2Check->isChecked ()) newMask |= (1<2);
00113             newPair.ecMultMask=newMask;
00114             addPair (newPair, pairsModel->numPairs (), false);
00115         } else {
00116             QMessageBox::information (this, tr ("Pair Type Error"),
00117                                     tr ("The first pair must be a particle,particle pair."));
00118         }
00119     }
00120 }
00121
00122 void PairsTab::addPair (PairsData pair, int pairIndex, bool fromFile) {
00123     if (pairsModel->isPair (pair)==-1) {
00124         pairsModel->insertRows (pairsModel->numPairs (), pairIndex+1-pairsModel->numPairs (), QModelIndex ());
00125
00126         QModelIndex index = pairsModel->index (pairIndex, 0, QModelIndex ());
00127         pairsModel->setData (index, pair.lightJ, Qt::EditRole);
00128         index = pairsModel->index (pairIndex, 1, QModelIndex ());
00129         pairsModel->setData (index, pair.lightPi, Qt::EditRole);
00130         index = pairsModel->index (pairIndex, 2, QModelIndex ());
00131         pairsModel->setData (index, pair.lightZ, Qt::EditRole);
00132         index = pairsModel->index (pairIndex, 3, QModelIndex ());
00133         pairsModel->setData (index, pair.lightM, Qt::EditRole);
00134         index = pairsModel->index (pairIndex, 4, QModelIndex ());
00135         pairsModel->setData (index, pair.lightG, Qt::EditRole);
00136         index = pairsModel->index (pairIndex, 5, QModelIndex ());
00137         pairsModel->setData (index, pair.heavyJ, Qt::EditRole);
00138         index = pairsModel->index (pairIndex, 6, QModelIndex ());

```

```

00139     pairsModel->setData(index, pair.heavyPi, Qt::EditRole);
00140     index = pairsModel->index(pairIndex, 7, QModelIndex());
00141     pairsModel->setData(index, pair.heavyZ, Qt::EditRole);
00142     index = pairsModel->index(pairIndex, 8, QModelIndex());
00143     pairsModel->setData(index, pair.heavyM, Qt::EditRole);
00144     index = pairsModel->index(pairIndex, 9, QModelIndex());
00145     pairsModel->setData(index, pair.heavyG, Qt::EditRole);
00146     index = pairsModel->index(pairIndex, 10, QModelIndex());
00147     pairsModel->setData(index, pair.excitationEnergy, Qt::EditRole);
00148     index = pairsModel->index(pairIndex, 11, QModelIndex());
00149     pairsModel->setData(index, pair.seperationEnergy, Qt::EditRole);
00150     index = pairsModel->index(pairIndex, 12, QModelIndex());
00151     pairsModel->setData(index, pair.channelRadius, Qt::EditRole);
00152     index = pairsModel->index(pairIndex, 13, QModelIndex());
00153     pairsModel->setData(index, pair.pairType, Qt::EditRole);
00154     index = pairsModel->index(pairIndex, 14, QModelIndex());
00155     pairsModel->setData(index, pair.ecMultMask, Qt::EditRole);
00156
00157     pairsView->resizeRowsToContents();
00158     if(!fromFile) emit(pairAdded(pairIndex));
00159 } else {
00160     QMessageBox::information(this, tr("Duplicate Pair"),
00161         tr("This pair already exists.));
00162 }
00163 }
00164
00165 void PairsTab::removePair() {
00166     QItemSelectionModel *selectionModel = pairsView->selectionModel();
00167     QModelIndexList indexes = selectionModel->selectedRows();
00168     QModelIndex index=indexes[0];
00169     bool previousIsGamma=false;
00170     if(index.row()==0&&(pairsModel->getPairs()).size()!=1) {
00171         QModelIndex previousIndex=pairsModel->index(index.row()+1, 13, QModelIndex());
00172         QVariant previousPairType=pairsModel->data(previousIndex, Qt::EditRole);
00173         if (previousPairType.toInt()==10) previousIsGamma=true;
00174     }
00175     if(!previousIsGamma) {
00176         pairsModel->removeRows(index.row(), 1, QModelIndex());
00177         emit(pairRemoved(index.row()));
00178     } else {
00179         QMessageBox::information(this, tr("Entrance Channel Error"),
00180             tr("This delete will result in a forbidden particle,gamma entrance pair.));
00181     }
00182 }
00183
00184 void PairsTab::editPair() {
00185     QItemSelectionModel *selectionModel = pairsView->selectionModel();
00186     QModelIndexList indexes = selectionModel->selectedRows();
00187
00188     QModelIndex index=indexes[0];
00189     QModelIndex i = pairsModel->index(index.row(), 0, QModelIndex());
00190     QVariant var = pairsModel->data(i, Qt::EditRole);
00191     QString lightJ = var.toString();
00192     i = pairsModel->index(index.row(), 1, QModelIndex());
00193     var =pairsModel->data(i, Qt::EditRole);
00194     int lightPi = var.toInt();
00195     i = pairsModel->index(index.row(), 2, QModelIndex());
00196     var = pairsModel->data(i, Qt::EditRole);
00197     QString lightZ = var.toString();
00198     i = pairsModel->index(index.row(), 3, QModelIndex());
00199     var = pairsModel->data(i, Qt::EditRole);
00200     QString lightM = var.toString();
00201     i = pairsModel->index(index.row(), 4, QModelIndex());
00202     var = pairsModel->data(i, Qt::EditRole);
00203     QString lightG = var.toString();
00204     i = pairsModel->index(index.row(), 5, QModelIndex());
00205     var = pairsModel->data(i, Qt::EditRole);
00206     QString heavyJ = var.toString();
00207     i = pairsModel->index(index.row(), 6, QModelIndex());
00208     var = pairsModel->data(i, Qt::EditRole);
00209     int heavyPi = var.toInt();
00210     i = pairsModel->index(index.row(), 7, QModelIndex());
00211     var = pairsModel->data(i, Qt::EditRole);
00212     QString heavyZ = var.toString();
00213     i = pairsModel->index(index.row(), 8, QModelIndex());
00214     var = pairsModel->data(i, Qt::EditRole);
00215     QString heavyM = var.toString();
00216     i = pairsModel->index(index.row(), 9, QModelIndex());
00217     var = pairsModel->data(i, Qt::EditRole);
00218     QString heavyG = var.toString();
00219     i = pairsModel->index(index.row(), 10, QModelIndex());
00220     var = pairsModel->data(i, Qt::EditRole);
00221     QString excitationEnergy = var.toString();
00222     i = pairsModel->index(index.row(), 11, QModelIndex());
00223     var = pairsModel->data(i, Qt::EditRole);
00224     QString seperationEnergy = var.toString();
00225     i = pairsModel->index(index.row(), 12, QModelIndex());

```

```

00226     var = pairsModel->data(i, Qt::EditRole);
00227     QString channelRadius = var.toString();
00228     i = pairsModel->index(index.row(), 13, QModelIndex());
00229     var = pairsModel->data(i, Qt::EditRole);
00230     int pairType = var.toInt();
00231     i = pairsModel->index(index.row(), 14, QModelIndex());
00232     var = pairsModel->data(i, Qt::EditRole);
00233     int ecMultMask = var.toInt();
00234
00235
00236     AddPairDialog aDialog;
00237     aDialog.setWindowTitle(tr("Edit a Particle Pair"));
00238     aDialog.lightJText->setText(lightJ);
00239     if(lightPi==1) aDialog.lightPiCombo->setCurrentIndex(0);
00240     else aDialog.lightPiCombo->setCurrentIndex(1);
00241     aDialog.lightZText->setText(lightZ);
00242     aDialog.lightMText->setText(lightM);
00243     //aDialog.lightGText->setText(lightG);
00244     aDialog.heavyJText->setText(heavyJ);
00245     if(heavyPi==1) aDialog.heavyPiCombo->setCurrentIndex(0);
00246     else aDialog.heavyPiCombo->setCurrentIndex(1);
00247     aDialog.heavyZText->setText(heavyZ);
00248     aDialog.heavyMText->setText(heavyM);
00249     //aDialog.heavyGText->setText(heavyG);
00250     aDialog.excitationEnergyText->setText(excitationEnergy);
00251     aDialog.seperationEnergyText->setText(seperationEnergy);
00252     aDialog.channelRadiusText->setText(channelRadius);
00253     if(pairType == 10) aDialog.pairTypeCombo->setCurrentIndex(1);
00254     else if(pairType == 20) aDialog.pairTypeCombo->setCurrentIndex(2);
00255     else aDialog.pairTypeCombo->setCurrentIndex(0);
00256     if(ecMultMask&(1<<0)) aDialog.e1Check->setChecked(true);
00257     else aDialog.e1Check->setChecked(false);
00258     //if(ecMultMask&(1<<1)) aDialog.m1Check->setChecked(true);
00259     //else aDialog.m1Check->setChecked(false);
00260     if(ecMultMask&(1<<2)) aDialog.e2Check->setChecked(true);
00261     else aDialog.e2Check->setChecked(false);
00262
00263     if (aDialog.exec()) {
00264         if(!(index.row()==0&&aDialog.pairTypeCombo->currentIndex()!=0)) {
00265             PairsData pair;
00266             pair.lightJ = aDialog.lightJText->text().toDouble();
00267             if(aDialog.lightPiCombo->currentIndex()==0) pair.lightPi=-1;
00268             else pair.lightPi=1;
00269             pair.lightZ = aDialog.lightZText->text().toInt();
00270             pair.lightM = aDialog.lightMText->text().toDouble();
00271             pair.lightG = 0.; //aDialog.lightGText->text().toDouble();
00272             pair.heavyJ = aDialog.heavyJText->text().toDouble();
00273             if(aDialog.heavyPiCombo->currentIndex()==0) pair.heavyPi=-1;
00274             else pair.heavyPi=1;
00275             pair.heavyZ = aDialog.heavyZText->text().toInt();
00276             pair.heavyM = aDialog.heavyMText->text().toDouble();
00277             pair.heavyG = 0.; //aDialog.heavyGText->text().toDouble();
00278             pair.excitationEnergy = aDialog.excitationEnergyText->text().toDouble();
00279             pair.seperationEnergy = aDialog.seperationEnergyText->text().toDouble();
00280             pair.channelRadius = aDialog.channelRadiusText->text().toDouble();
00281             if(aDialog.pairTypeCombo->currentIndex()==1) pair.pairType=10;
00282             else if(aDialog.pairTypeCombo->currentIndex()==2) pair.pairType=20;
00283             else pair.pairType=0;
00284             unsigned char newECMultMask=0;
00285             if(aDialog.e1Check->isChecked()) newECMultMask |= (1<<0);
00286             //if(aDialog.m1Check->isChecked()) newECMultMask |= (1<<1);
00287             if(aDialog.e2Check->isChecked()) newECMultMask |= (1<<2);
00288             pair.ecMultMask=newECMultMask;
00289             editPair(pair,index.row(),false);
00290         } else {
00291             QMessageBox::information(this,tr("Pair Type Error"),
00292                                     tr("The first pair must be a particle,particle pair."));
00293         }
00294     }
00295 }
00296
00297 void PairsTab::editPair(PairsData pair,int pairIndex,bool fromFile) {
00298     QModelIndex i = pairsModel->index(pairIndex,0,QModelIndex());
00299     QVariant var = pairsModel->data(i, Qt::EditRole);
00300     if (pair.lightJ != var.toDouble()) pairsModel->setData(i,pair.lightJ, Qt::EditRole);
00301     i = pairsModel->index(pairIndex,1,QModelIndex());
00302     var = pairsModel->data(i, Qt::EditRole);
00303     if (pair.lightPi != var.toInt()) pairsModel->setData(i,pair.lightPi, Qt::EditRole);
00304     i = pairsModel->index(pairIndex,2,QModelIndex());
00305     var = pairsModel->data(i, Qt::EditRole);
00306     if (pair.lightZ != var.toInt()) pairsModel->setData(i,pair.lightZ, Qt::EditRole);
00307     i = pairsModel->index(pairIndex,3,QModelIndex());
00308     var = pairsModel->data(i, Qt::EditRole);
00309     if (pair.lightM != var.toDouble()) pairsModel->setData(i,pair.lightM, Qt::EditRole);
00310     i = pairsModel->index(pairIndex,4,QModelIndex());
00311     var = pairsModel->data(i, Qt::EditRole);
00312     if (pair.lightG != var.toDouble()) pairsModel->setData(i,pair.lightG, Qt::EditRole);

```

```

00313     i = pairsModel->index(pairIndex,5,QModelIndex());
00314     var = pairsModel->data(i, Qt::EditRole);
00315     if (pair.heavyJ != var.toDouble()) pairsModel->setData(i,pair.heavyJ, Qt::EditRole);
00316     i = pairsModel->index(pairIndex,6,QModelIndex());
00317     var = pairsModel->data(i, Qt::EditRole);
00318     if (pair.heavyPi != var.toInt()) pairsModel->setData(i,pair.heavyPi, Qt::EditRole);
00319     i = pairsModel->index(pairIndex,7,QModelIndex());
00320     var = pairsModel->data(i, Qt::EditRole);
00321     if (pair.heavyZ != var.toInt()) pairsModel->setData(i,pair.heavyZ, Qt::EditRole);
00322     i = pairsModel->index(pairIndex,8,QModelIndex());
00323     var = pairsModel->data(i, Qt::EditRole);
00324     if (pair.heavyM != var.toDouble()) pairsModel->setData(i,pair.heavyM, Qt::EditRole);
00325     i = pairsModel->index(pairIndex,9,QModelIndex());
00326     var = pairsModel->data(i, Qt::EditRole);
00327     if (pair.heavyG != var.toDouble()) pairsModel->setData(i,pair.heavyG, Qt::EditRole);
00328     i = pairsModel->index(pairIndex,10,QModelIndex());
00329     var = pairsModel->data(i, Qt::EditRole);
00330     if (pair.excitationEnergy != var.toDouble()) pairsModel->setData(i,pair.excitationEnergy,
Qt::EditRole);
00331     i = pairsModel->index(pairIndex,11,QModelIndex());
00332     var = pairsModel->data(i, Qt::EditRole);
00333     if (pair.seperationEnergy != var.toDouble()) pairsModel->setData(i,pair.seperationEnergy,
Qt::EditRole);
00334     i = pairsModel->index(pairIndex,12,QModelIndex());
00335     var = pairsModel->data(i, Qt::EditRole);
00336     if (pair.channelRadius != var.toDouble()) pairsModel->setData(i,pair.channelRadius, Qt::EditRole);
00337     i = pairsModel->index(pairIndex,13,QModelIndex());
00338     var = pairsModel->data(i, Qt::EditRole);
00339     if (pair.pairType != var.toInt()) pairsModel->setData(i,pair.pairType, Qt::EditRole);
00340     i = pairsModel->index(pairIndex,14,QModelIndex());
00341     var = pairsModel->data(i, Qt::EditRole);
00342     if (pair.pairType != var.toInt()) pairsModel->setData(i,pair.ecMultMask, Qt::EditRole);
00343
00344     if(!fromFile) emit (pairEdited(pairIndex));
00345 }
00346
00347 void PairsTab::updateButtons(const QModelIndexList &selection) {
00348     QModelIndexList indexes = selection.indexes();
00349
00350     if (!indexes.isEmpty()) {
00351         deleteButton->setEnabled(true);
00352     } else {
00353         deleteButton->setEnabled(false);
00354     }
00355 }
00356
00357 bool PairsTab::parseOldECSection(QTextStream& inStream) {
00358     int isActive;
00359     int exitPairIndex;
00360     double minJ;
00361     double maxJ;
00362     int multMask;
00363     QString line("");
00364     while (line.trimmed() != QString("</externalCapture>") && !inStream.atEnd()) {
00365         line = inStream.readLine();
00366         if (line.trimmed().isEmpty()) continue;
00367         if (line.trimmed() != QString("</externalCapture>") && !inStream.atEnd()) {
00368             QTextStream in(&line);
00369             in >> isActive >> exitPairIndex >> minJ >> maxJ >> multMask;
00370             if (in.status() != QTextStream::Ok) return false;
00371             if (exitPairIndex-1 < pairsModel->getPairs().size()) {
00372                 QModelIndex i = pairsModel->index(exitPairIndex-1,14,QModelIndex());
00373                 QVariant var = pairsModel->data(i, Qt::EditRole);
00374                 pairsModel->setData(i,multMask, Qt::EditRole);
00375             }
00376         }
00377     }
00378     if (line.trimmed() != QString("</externalCapture>")) return false;
00379     return true;
00380 }
00381
00382 void PairsTab::showInfo(int which, QString title) {
00383     if (which < infoText.size()) {
00384         if (!infoDialog[which]) {
00385             infoDialog[which] = new InfoDialog(infoText[which],this,title);
00386             infoDialog[which]->setAttribute(Qt::WA_DeleteOnClose);
00387             infoDialog[which]->show();
00388         } else infoDialog[which]->raise();
00389     }
00390 }

```

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.](#)

```
00001 #include <QVBoxLayout>
00002 #include <QGridLayout>
00003 #include <QGroupBox>
00004 #include <QComboBox>
00005 #include <QRadioButton>
00006 #include <QCheckBox>
00007 #include <QPushButton>
00008 #include <QCheckBox>
00009 #include <QListView>
00010
00011 #include "PlotTab.h"
00012 #include "Config.h"
00013 #include "AZUREPlot.h"
00014 #include "SegmentsDataModel.h"
00015 #include "SegmentsTestModel.h"
00016 #include "RichTextDelegate.h"
00017 #include "InfoDialog.h"
00018 #include <iostream>
00019
00020 QVariant SegTestProxyModel::data(const QModelIndex& index, int role) const {
00021     if (index.isValid() && role == Qt::DisplayRole) {
00022         QModelIndex sourceIndex = mapToSource(index);
00023         return QString("##%1:
%2").arg(sourceIndex.row()+1).arg(static_cast<SegmentsTestModel*>(sourceModel())->getReactionLabel(sourceIndex));
00024     }
00025     return QVariant();
00026 }
00027
00028
00029 bool SegTestProxyModel::filterAcceptsRow(int source_row, const QModelIndex &source_parent) const {
00030     if (QSortFilterProxyModel::filterAcceptsRow(source_row, source_parent)) {
00031         SegmentsTestModel* model = static_cast<SegmentsTestModel*>(sourceModel());
00032         QModelIndex source_index = model->index(source_row, 9, source_parent);
00033         int dataType = model->data(source_index, Qt::EditRole).toInt();
00034         if (dataType!=3) return true;
00035         return false;
00036     }
00037     return false;
00038 }
00039
00040 QVariant SegDataProxyModel::data(const QModelIndex& index, int role) const {
00041     if (index.isValid() && role == Qt::DisplayRole) {
00042         QModelIndex sourceIndex = mapToSource(index);
00043         return QString("##%1:
%2").arg(sourceIndex.row()+1).arg(static_cast<SegmentsDataModel*>(sourceModel())->getReactionLabel(sourceIndex));
00044     }
00045 }
```



```

00045     return QVariant();
00046 }
00047
00048
00049
00050 PlotTab::PlotTab(Config& config, SegmentsDataModel* dataModel, SegmentsTestModel* testModel, QWidget*
parent) :
00051     configure(config), QWidget(parent) {
00052     azurePlot = new AZUREPlot(this, this);
00053
00054     segDataProxyModel = new SegDataProxyModel(this);
00055     segDataProxyModel->setSourceModel(dataModel);
00056     segDataProxyModel->setDynamicSortFilter(true);
00057     segDataProxyModel->setFilterKeyColumn(0);
00058     segDataProxyModel->setFilterRole(Qt::CheckStateRole);
00059     segDataProxyModel->setFilterRegExp(QString("%1").arg(Qt::Checked));
00060     segTestProxyModel = new SegTestProxyModel(this);
00061     segTestProxyModel->setSourceModel(testModel);
00062     segTestProxyModel->setDynamicSortFilter(true);
00063     segTestProxyModel->setFilterKeyColumn(0);
00064     segTestProxyModel->setFilterRole(Qt::CheckStateRole);
00065     segTestProxyModel->setFilterRegExp(QString("%1").arg(Qt::Checked));
00066
00067     QVBoxLayout* rightLayout = new QVBoxLayout;
00068
00069     QGridLayout *topLayout = new QGridLayout;
00070
00071     QGroupBox *xAxisBox = new QGroupBox(tr("X-Axis Configuration"));
00072     xAxisTypeCombo = new QComboBox;
00073     xAxisTypeCombo->addItem(tr("CoM Energy"));
00074     xAxisTypeCombo->addItem(tr("Excitation Energy"));
00075     xAxisTypeCombo->addItem(tr("CoM Angle"));
00076     connect(xAxisTypeCombo, SIGNAL(activated(int)), this, SLOT(xAxisTypeChanged()));
00077     xAxisTypeCombo->setCurrentIndex(0);
00078     azurePlot->setXAxisType(0);
00079 #ifndef MACX_SPACING
00080     xAxisIsLogCheck = new QCheckBox(tr("Use Log Scale"));
00081 #else
00082     xAxisIsLogCheck = new QCheckBox(tr("Use Logarithmic Scale"));
00083 #endif
00084     connect(xAxisIsLogCheck, SIGNAL(toggled(bool)), this, SLOT(xAxisLogScaleChanged(bool)));
00085     QHBoxLayout* xAxisLayout = new QHBoxLayout;
00086     xAxisLayout->setContentsMargins(5, 5, 5, 5);
00087     xAxisLayout->addWidget(xAxisTypeCombo);
00088     xAxisLayout->addWidget(xAxisIsLogCheck);
00089     xAxisBox->setLayout(xAxisLayout);
00090     QGroupBox *yAxisBox = new QGroupBox(tr("Y-Axis Configuration"));
00091     yAxisXSButton = new QRadioButton(tr("Cross Section"));
00092     connect(yAxisXSButton, SIGNAL(toggled(bool)), this, SLOT(yAxisTypeChanged()));
00093     yAxisXSButton->setChecked(true);
00094     yAxisSFButton = new QRadioButton(tr("S-Factor"));
00095     connect(yAxisSFButton, SIGNAL(toggled(bool)), this, SLOT(yAxisTypeChanged()));
00096 #ifndef MACX_SPACING
00097     yAxisIsLogCheck = new QCheckBox(tr("Use Log Scale"));
00098 #else
00099     yAxisIsLogCheck = new QCheckBox(tr("Use Logarithmic Scale"));
00100 #endif
00101     connect(yAxisIsLogCheck, SIGNAL(toggled(bool)), this, SLOT(yAxisLogScaleChanged(bool)));
00102     yAxisIsLogCheck->setChecked(true);
00103     QHBoxLayout* yAxisLayout = new QHBoxLayout;
00104     yAxisLayout->setContentsMargins(5, 5, 5, 5);
00105     yAxisLayout->addWidget(yAxisXSButton);
00106     yAxisLayout->addWidget(yAxisSFButton);
00107     yAxisLayout->addWidget(yAxisIsLogCheck);
00108     yAxisBox->setLayout(yAxisLayout);
00109
00110     topLayout->addWidget(xAxisBox, 0, 0);
00111     topLayout->addWidget(yAxisBox, 0, 1);
00112
00113     rightLayout->addLayout(topLayout);
00114     rightLayout->addWidget(azurePlot);
00115
00116     dataSegmentSelectorList = new QListView;
00117     testSegmentSelectorList = new QListView;
00118     dataSegmentSelectorList->setAttribute(Qt::WA_MacShowFocusRect, 0);
00119     testSegmentSelectorList->setAttribute(Qt::WA_MacShowFocusRect, 0);
00120     dataSegmentSelectorList->setModel(segDataProxyModel);
00121     testSegmentSelectorList->setModel(segTestProxyModel);
00122     dataSegmentSelectorList->setItemDelegate(new RichTextDelegate());
00123     testSegmentSelectorList->setItemDelegate(new RichTextDelegate());
00124     dataSegmentSelectorList->setSelectionMode(QAbstractItemView::MultiSelection);
00125     testSegmentSelectorList->setSelectionMode(QAbstractItemView::MultiSelection);
00126     dataSegmentSelectorList->setResizeMode(QListView::Adjust);
00127     testSegmentSelectorList->setResizeMode(QListView::Adjust);
00128
00129     QGroupBox *dataSegmentSelectorBox = new QGroupBox(tr("Segments From Data"));
00130     QGridLayout *dataSegmentSelectorLayout = new QGridLayout;

```



```

00131     dataSegmentSelectorLayout->setContentsMargins(5,5,5,5);
00132     QGroupBox *testSegmentSelectorBox = new QGroupBox(tr("Segments Without Data"));
00133     QGridLayout *testSegmentSelectorLayout = new QGridLayout;
00134     testSegmentSelectorLayout->setContentsMargins(5,5,5,5);
00135
00136     dataSegmentSelectorLayout->addWidget(dataSegmentSelectorList,0,0);
00137     testSegmentSelectorLayout->addWidget(testSegmentSelectorList,0,0);
00138     dataSegmentSelectorBox->setLayout(dataSegmentSelectorLayout);
00139     testSegmentSelectorBox->setLayout(testSegmentSelectorLayout);
00140
00141     refreshButton = new QPushButton(tr("&Draw"));
00142     connect(refreshButton,SIGNAL(clicked()),this,SLOT(draw()));
00143     exportButton = new QPushButton(tr("Export..."));
00144     connect(exportButton,SIGNAL(clicked()),azurePlot,SLOT(exportPlot()));
00145     printButton = new QPushButton(tr("Print..."));
00146     connect(printButton,SIGNAL(clicked()),azurePlot,SLOT(print()));
00147     QHBoxLayout *buttonLayout = new QHBoxLayout;
00148     buttonLayout->addWidget(refreshButton);
00149     buttonLayout->addWidget(exportButton);
00150     buttonLayout->addWidget(printButton);
00151
00152     QVBoxLayout* leftLayout = new QVBoxLayout;
00153     leftLayout->addWidget(dataSegmentSelectorBox);
00154     leftLayout->addWidget(testSegmentSelectorBox);
00155     leftLayout->addLayout(buttonLayout);
00156
00157     QGridLayout* mainLayout = new QGridLayout;
00158     mainLayout->addLayout(leftLayout,0,0);
00159     mainLayout->addLayout(rightLayout,0,1);
00160     mainLayout->setColumnStretch(1,1);
00161
00162     setLayout(mainLayout);
00163 }
00164
00165 QList<PlotEntry*> PlotTab::getDataSegments() {
00166     QList<PlotEntry*> dataSegmentPlotEntries;
00167     QModelIndexList indexes = dataSegmentSelectorList->selectionModel()->selectedIndexes();
00168     for(int i = 0; i< indexes.size(); i++) {
00169         int sourceRow = segDataProxyModel->mapToSource(indexes[i]).row();
00170         QModelIndex sourceIndex =
00171             segDataProxyModel->mapToSource(segDataProxyModel->index(indexes[i].row(),1,QModelIndex()));
00172         int entranceKey = segDataProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
00173         sourceIndex =
00174             segDataProxyModel->mapToSource(segDataProxyModel->index(indexes[i].row(),2,QModelIndex()));
00175         int exitKey = segDataProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
00176         sourceIndex =
00177             segDataProxyModel->mapToSource(segDataProxyModel->index(indexes[i].row(),7,QModelIndex()));
00178         int dataType = segDataProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
00179         QString filename = (dataType==3) ?
00180             QString::fromStdString(configure.outputdir)+QString("AZUREOut_aa=%1_TOTAL_CAPTURE.out").arg(entranceKey)
00181             :
00182             QString::fromStdString(configure.outputdir)+QString("AZUREOut_aa=%1_R=%2.out").arg(entranceKey).arg(exitKey);
00183         int numPreviousInBlock = 0;
00184         for(int j =0; j<indexes[i].row(); j++) {
00185             sourceIndex = segDataProxyModel->mapToSource(segDataProxyModel->index(j,1,QModelIndex()));
00186             int previousEntranceKey =
00187                 segDataProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
00188             sourceIndex = segDataProxyModel->mapToSource(segDataProxyModel->index(j,2,QModelIndex()));
00189             int previousExitKey = segDataProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
00190             if(previousEntranceKey==entranceKey&&previousExitKey==exitKey) numPreviousInBlock++;
00191         }
00192         PlotEntry* newPlotEntry = new PlotEntry(0,entranceKey,exitKey,numPreviousInBlock,filename);
00193         dataSegmentPlotEntries.push_back(newPlotEntry);
00194     }
00195     return dataSegmentPlotEntries;
00196 }
00197
00198 QList<PlotEntry*> PlotTab::getTestSegments() {
00199     QList<PlotEntry*> testSegmentPlotEntries;
00200     QModelIndexList indexes = testSegmentSelectorList->selectionModel()->selectedIndexes();
00201     for(int i = 0; i< indexes.size(); i++) {
00202         int sourceRow = segTestProxyModel->mapToSource(indexes[i]).row();
00203         QModelIndex sourceIndex =
00204             segTestProxyModel->mapToSource(segTestProxyModel->index(indexes[i].row(),1,QModelIndex()));
00205         int entranceKey = segTestProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
00206         sourceIndex =
00207             segTestProxyModel->mapToSource(segTestProxyModel->index(indexes[i].row(),2,QModelIndex()));
00208         int exitKey = segTestProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
00209         sourceIndex =
00210             segTestProxyModel->mapToSource(segTestProxyModel->index(indexes[i].row(),9,QModelIndex()));
00211         int dataType = segTestProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
00212         QString filename = (dataType==4) ?
00213             QString::fromStdString(configure.outputdir)+QString("AZUREOut_aa=%1_TOTAL_CAPTURE.extrap").arg(entranceKey)
00214             :

```

```

00208     QString::fromStdString(configure.outputdir)+QString("AZUREOut_aa=%1_R=%2.extrap").arg(entranceKey).arg(exitKey);
00209     int numPreviousInBlock = 0;
00210     for(int j =0; j<indexes[i].row(); j++) {
00211         sourceIndex = segTestProxyModel->mapToSource(segTestProxyModel->index(j,1,QModelIndex()));
00212         int previousEntranceKey =
segTestProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
00213         sourceIndex = segTestProxyModel->mapToSource(segTestProxyModel->index(j,2,QModelIndex()));
00214         int previousExitKey = segTestProxyModel->sourceModel()->data(sourceIndex,Qt::EditRole).toInt();
00215         if(previousEntranceKey==entranceKey&&previousExitKey==exitKey) numPreviousInBlock++;
00216     }
00217     PlotEntry* newPlotEntry = new PlotEntry(1,entranceKey,exitKey,numPreviousInBlock,filename);
00218     testSegmentPlotEntries.push_back(newPlotEntry);
00219 }
00220 return testSegmentPlotEntries;
00221 }
00222
00223 void PlotTab::draw() {
00224     QList<PlotEntry*> entries = getDataSegments();
00225     entries.append(getTestSegments());
00226     azurePlot->draw(entries);
00227 }
00228
00229 void PlotTab::xAxisTypeChanged() {
00230     azurePlot->setXAxisType(xAxisTypeCombo->currentIndex());
00231 }
00232
00233 void PlotTab::yAxisTypeChanged() {
00234     if(yAxisXSButton->isChecked())
00235         azurePlot->setYAxisType(0);
00236     else if(yAxisSFButton->isChecked())
00237         azurePlot->setYAxisType(1);
00238 }
00239
00240 void PlotTab::xAxisLogScaleChanged(bool checked) {
00241     azurePlot->setXAxisLog(checked);
00242 }
00243
00244 void PlotTab::yAxisLogScaleChanged(bool checked) {
00245     azurePlot->setYAxisLog(checked);
00246 }
00247
00248 void PlotTab::reset() {
00249     azurePlot->clearEntries();
00250     xAxisTypeCombo->setCurrentIndex(0);
00251     xAxisIsLogCheck->setChecked(false);
00252     yAxisXSButton->setChecked(true);
00253     yAxisIsLogCheck->setChecked(true);
00254 }
00255
00256 void PlotTab::showInfo(int which,QString title) {
00257     if(which<infoText.size()) {
00258         if(!infoDialog[which]) {
00259             infoDialog[which] = new InfoDialog(infoText[which],this,title);
00260             infoDialog[which]->setAttribute(Qt::WA_DeleteOnClose);
00261             infoDialog[which]->show();
00262         } else infoDialog[which]->raise();
00263     }
00264 }

```

8.121 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RichTextDelegate.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.](#)

```

00001 #include <QTextDocument>
00002 #include <QPainter>
00003 #include <QAbstractTextDocumentLayout>
00004 #include <QApplication>
00005 #include <QPushButton>
00006
00007 #include "RichTextDelegate.h"
00008
00009 void RichTextDelegate::paint(QPainter *painter, const QStyleOptionViewItem &option, const QModelIndex
&index) const {
00010     QTextDocument document;
00011     QString value = index.data(Qt::DisplayRole).toString();
00012
00013     document.setHtml(value);
00014     QStyleOptionViewItemV4 opt(option);
00015     initStyleOption(&opt, index);
00016
00017     document.setTextWidth(opt.rect.width());
00018
00019     opt.text=QString();
00020     QStyle *style = opt.widget ? opt.widget->style() : QApplication::style();
00021     style->drawControl(QStyle::CE_ItemViewItem, &opt, painter);
00022
00023     QAbstractTextDocumentLayout::PaintContext ctx;
00024     if (opt.state & QStyle::State_Selected) {
00025         if (opt.state & QStyle::State_Active)
00026             ctx.palette.setColor(QPalette::Text, opt.palette.color(QPalette::Active,
QPalette::HighlightedText));
00027         else ctx.palette.setColor(QPalette::Text, opt.palette.color(QPalette::Inactive,
QPalette::HighlightedText));
00028     }
00029
00030     painter->translate(opt.rect.topLeft());
00031     document.documentLayout()->draw(painter,ctx);
00032     painter->translate(-opt.rect.topLeft());
00033 }
00034
00035 QSize RichTextDelegate::sizeHint(const QStyleOptionViewItem & option, const QModelIndex & index )
const {
00036     QTextDocument document;
00037     QString value = index.data(Qt::DisplayRole).toString();
00038     document.setHtml(value);
00039     document.setTextWidth(option.rect.width());
00040     return QSize(option.rect.width(),document.size().height());
00041 }
00042

```

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.](#)

```
00001 #include <QTextEdit>
```

```

00002 #include <QComboBox>
00003 #include <QGridLayout>
00004 #include <QLabel>
00005 #include <QSpacerItem>
00006 #include <QRadioButton>
00007 #include <QGroupBox>
00008 #include <QFileDialog>
00009
00010 #include "RunTab.h"
00011 #include "ChooseFileButton.h"
00012 #include "FilteredTextEdit.h"
00013 #include "InfoDialog.h"
00014
00015 RunTab::RunTab(QWidget* parent) : QWidget(parent) {
00016     calcType = new QComboBox;
00017     calcType->addItem(tr("Calculate Segments From Data"));
00018     calcType->addItem(tr("Fit Segments From Data"));
00019     calcType->addItem(tr("Calculate Segments Without Data"));
00020     calcType->addItem(tr("Perform MINOS Error Analysis"));
00021     calcType->addItem(tr("Calculate Reaction Rate"));
00022     connect(calcType, SIGNAL(currentIndexChanged(int)), this, SLOT(calculationTypeChanged(int)));
00023
00024     chiVarianceText = new QLineEdit;
00025     chiVarianceText->setText("1.0");
00026     chiVarianceText->setEnabled(false);
00027     chiVarianceText->setMinimumWidth(50);
00028     chiVarianceText->setMaximumWidth(50);
00029
00030     calcButton = new QPushButton(tr("Save and &Run"));
00031     stopAZUREButton = new QPushButton(tr("Stop AZURE2"));
00032     stopAZUREButton->setEnabled(false);
00033
00034     QGridLayout* calcLayout = new QGridLayout;
00035     calcLayout->addWidget(new QLabel(tr("Calculation Type:")), 0, 0, Qt::AlignRight);
00036     calcLayout->addWidget(calcType, 0, 1);
00037     calcLayout->setColumnStretch(1, 0);
00038     calcLayout->addItem(new QSpacerItem(28, 28), 0, 2);
00039     calcLayout->setColumnStretch(2, 1);
00040     calcLayout->addWidget(new QLabel(tr("Chi-Squared Variance:")), 0, 3, Qt::AlignRight);
00041     calcLayout->addWidget(chiVarianceText, 0, 4);
00042     calcLayout->setColumnStretch(4, 0);
00043     calcLayout->addWidget(calcButton, 0, 5);
00044     calcLayout->setColumnStretch(5, 0);
00045     calcLayout->addWidget(stopAZUREButton, 0, 6);
00046     calcLayout->setColumnStretch(6, 0);
00047
00048     paramFileText = new QLineEdit;
00049     paramFileText->setEnabled(false);
00050     newParamFileButton = new QRadioButton(tr("Create New Parameters File"));
00051     newParamFileButton->setMinimumWidth(200);
00052     newParamFileButton->setMaximumWidth(200);
00053     oldParamFileButton = new QRadioButton(tr("Use: "));
00054     connect(oldParamFileButton, SIGNAL(toggled(bool)), this, SLOT(paramFileButtonChanged(bool)));
00055     newParamFileButton->setChecked(true);
00056
00057     QGroupBox* paramFileGroup = new QGroupBox(tr("Parameters File"));
00058     QHBoxLayout* paramFileLayout = new QHBoxLayout;
00059     paramFileLayout->setContentsMargins(5, 5, 5, 5);
00060     paramFileLayout->addWidget(newParamFileButton);
00061     paramFileLayout->addWidget(oldParamFileButton);
00062     paramFileLayout->addWidget(paramFileText);
00063     paramFileChoose=new ChooseFileButton(tr("Choose..."));
00064     paramFileChoose->setEnabled(false);
00065     paramFileChoose->setLineEdit(paramFileText);
00066     connect(paramFileChoose, SIGNAL(clicked(QLineEdit*)), this, SLOT(setChooseFile(QLineEdit*)));
00067     paramFileLayout->addWidget(paramFileChoose);
00068     paramFileGroup->setLayout(paramFileLayout);
00069
00070     integralsFileText = new QLineEdit;
00071     integralsFileText->setEnabled(false);
00072     newIntegralsFileButton = new QRadioButton(tr("Create New Integrals File"));
00073     newIntegralsFileButton->setMinimumWidth(200);
00074     newIntegralsFileButton->setMaximumWidth(200);
00075     oldIntegralsFileButton = new QRadioButton(tr("Use: "));
00076     connect(oldIntegralsFileButton, SIGNAL(toggled(bool)), this, SLOT(integralsFileButtonChanged(bool)));
00077     newIntegralsFileButton->setChecked(true);
00078
00079     integralsFileGroup = new QGroupBox(tr("External Capture Integrals File"));
00080     QHBoxLayout* integralsFileLayout = new QHBoxLayout;
00081     integralsFileLayout->setContentsMargins(5, 5, 5, 5);
00082     integralsFileLayout->addWidget(newIntegralsFileButton);
00083     integralsFileLayout->addWidget(oldIntegralsFileButton);
00084     integralsFileLayout->addWidget(integralsFileText);
00085     integralsFileChoose=new ChooseFileButton(tr("Choose..."));
00086     integralsFileChoose->setEnabled(false);
00087     integralsFileChoose->setLineEdit(integralsFileText);
00088     connect(integralsFileChoose, SIGNAL(clicked(QLineEdit*)), this, SLOT(setChooseFile(QLineEdit*)));

```

```

00089     integralsFileLayout->addWidget(integralsFileChoose);
00090     integralsFileGroup->setLayout(integralsFileLayout);
00091
00092     gridTempButton = new QRadioButton(tr("Create Temperatures"));
00093     rateEntranceKey = new QLineEdit;
00094     rateEntranceKey->setMinimumWidth(50);
00095     rateEntranceKey->setMaximumWidth(50);
00096     rateExitKey = new QLineEdit;
00097     rateExitKey->setMinimumWidth(50);
00098     rateExitKey->setMaximumWidth(50);
00099     minTempText = new QLineEdit;
00100     minTempText->setMinimumWidth(50);
00101     minTempText->setMaximumWidth(50);
00102     maxTempText = new QLineEdit;
00103     maxTempText->setMinimumWidth(50);
00104     maxTempText->setMaximumWidth(50);
00105     tempStepText = new QLineEdit;
00106     tempStepText->setMinimumWidth(50);
00107     tempStepText->setMaximumWidth(50);
00108     fileTempButton = new QRadioButton(tr("Use Temperature File: "));
00109     connect(fileTempButton, SIGNAL(toggled(bool)), this, SLOT(fileTempButtonChanged(bool)));
00110     fileTempText = new QLineEdit;
00111     fileTempText->setEnabled(false);
00112     gridTempButton->setChecked(true);
00113
00114     rateParamsGroup = new QGroupBox(tr("Reaction Rate Parameters"));
00115     rateParamsGroup->hide();
00116     QGridLayout* keyLayout = new QGridLayout;
00117     keyLayout->addWidget(new QLabel(tr("Entrance Key:")), 0, 0, Qt::AlignRight);
00118     keyLayout->addWidget(rateEntranceKey, 0, 1);
00119     keyLayout->addWidget(new QLabel(tr("Exit Key:")), 0, 2, Qt::AlignRight);
00120     keyLayout->addWidget(rateExitKey, 0, 3);
00121     keyLayout->addItem(new QSpacerItem(28, 28), 0, 4);
00122     keyLayout->setColumnStretch(4, 1);
00123     QGridLayout* gridTempLayout = new QGridLayout;
00124     gridTempLayout->addWidget(gridTempButton, 0, 0);
00125     gridTempLayout->setColumnStretch(0, 0);
00126     gridTempLayout->addItem(new QSpacerItem(28, 28), 0, 1);
00127     gridTempLayout->setColumnStretch(1, 1);
00128     gridTempLayout->addWidget(new QLabel(tr("Minimum Temperature (GK):")), 0, 2, Qt::AlignRight);
00129     gridTempLayout->addWidget(minTempText, 0, 3);
00130     gridTempLayout->setColumnStretch(3, 0);
00131     gridTempLayout->addWidget(new QLabel(tr("Maximum Temperature (GK):")), 0, 4, Qt::AlignRight);
00132     gridTempLayout->addWidget(maxTempText, 0, 5);
00133     gridTempLayout->setColumnStretch(5, 0);
00134     gridTempLayout->addWidget(new QLabel(tr("Temperature Step (GK):")), 0, 6, Qt::AlignRight);
00135     gridTempLayout->addWidget(tempStepText, 0, 7);
00136     gridTempLayout->setColumnStretch(7, 0);
00137     QHBoxLayout* fileTempLayout = new QHBoxLayout;
00138     fileTempLayout->addWidget(fileTempButton);
00139     fileTempLayout->addWidget(fileTempText);
00140     rateParamsChoose = new ChooseFileButton(tr("Choose..."));
00141     rateParamsChoose->setEnabled(false);
00142     rateParamsChoose->setLineEdit(fileTempText);
00143     connect(rateParamsChoose, SIGNAL(clicked(QLineEdit*)), this, SLOT(setChooseFile(QLineEdit*)));
00144     fileTempLayout->addWidget(rateParamsChoose);
00145
00146     QVBoxLayout* rateParamsLayout = new QVBoxLayout;
00147     rateParamsLayout->setContentsMargins(5, 5, 5, 5);
00148     rateParamsLayout->addLayout(keyLayout);
00149     rateParamsLayout->addLayout(gridTempLayout);
00150     rateParamsLayout->addLayout(fileTempLayout);
00151     rateParamsGroup->setLayout(rateParamsLayout);
00152
00153     runtimeText = new FilteredTextEdit;
00154     runtimeText->setReadOnly(true);
00155     runtimeText->setAcceptRichText(false);
00156
00157     QVBoxLayout* mainLayout = new QVBoxLayout;
00158     mainLayout->addLayout(calcLayout);
00159     mainLayout->addWidget(paramFileGroup);
00160     mainLayout->addWidget(integralsFileGroup);
00161     mainLayout->addWidget(rateParamsGroup);
00162     mainLayout->addWidget(runtimeText);
00163     setLayout(mainLayout);
00164 }
00165
00166
00167 void RunTab::calculationTypeChanged(int index) {
00168     if(index==4) {
00169         integralsFileGroup->hide();
00170         rateParamsGroup->show();
00171     } else {
00172         rateParamsGroup->hide();
00173         integralsFileGroup->show();
00174     }
00175     if(index==3) chiVarianceText->setEnabled(true);

```

```

00176     else {
00177         chiVarianceText->setText("1.0");
00178         chiVarianceText->setEnabled(false);
00179     }
00180 }
00181
00182 void RunTab::paramFileButtonChanged(bool checked) {
00183     if(checked) {
00184         paramFileText->setEnabled(true);
00185         paramFileChoose->setEnabled(true);
00186     } else {
00187         paramFileText->setEnabled(false);
00188         paramFileChoose->setEnabled(false);
00189     }
00190 }
00191
00192 void RunTab::integralsFileButtonChanged(bool checked) {
00193     if(checked) {
00194         integralsFileText->setEnabled(true);
00195         integralsFileChoose->setEnabled(true);
00196     } else {
00197         integralsFileText->setEnabled(false);
00198         integralsFileChoose->setEnabled(false);
00199     }
00200 }
00201
00202 void RunTab::fileTempButtonChanged(bool checked) {
00203     if(checked) {
00204         fileTempText->setEnabled(true);
00205         rateParamsChoose->setEnabled(true);
00206         minTempText->setEnabled(false);
00207         maxTempText->setEnabled(false);
00208         tempStepText->setEnabled(false);
00209     } else {
00210         fileTempText->setEnabled(false);
00211         rateParamsChoose->setEnabled(false);
00212         minTempText->setEnabled(true);
00213         maxTempText->setEnabled(true);
00214         tempStepText->setEnabled(true);
00215     }
00216 }
00217
00218 void RunTab::setChooseFile(QLineEdit *lineEdit) {
00219     QString filename = QFileDialog::getOpenFileName(this);
00220     if(!filename.isEmpty()) {
00221         lineEdit->setText(QDir::fromNativeSeparators(filename));
00222     }
00223 }
00224
00225 void RunTab::reset() {
00226     calcType->setCurrentIndex(0);
00227     newParamFileButton->setChecked(true);
00228     paramFileText->setText("");
00229     newIntegralsFileButton->setChecked(true);
00230     integralsFileText->setText("");
00231     rateEntranceKey->setText("");
00232     rateExitKey->setText("");
00233     gridTempButton->setChecked(true);
00234     minTempText->setText("");
00235     maxTempText->setText("");
00236     tempStepText->setText("");
00237     fileTempText->setText("");
00238     runtimeText->clear();
00239 }
00240
00241 void RunTab::showInfo(int which, QString title) {
00242     if(which<infoText.size()) {
00243         if(!infoDialog[which]) {
00244             infoDialog[which] = new InfoDialog(infoText[which],this,title);
00245             infoDialog[which]->setAttribute(Qt::WA_DeleteOnClose);
00246             infoDialog[which]->show();
00247         } else infoDialog[which]->raise();
00248     }
00249 }

```

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.](#)

```
00001 #include "SegmentsDataModel.h"
00002 #include "PairsModel.h"
00003 #include <QColor>
00004
00005 SegmentsDataModel::SegmentsDataModel(QObject *parent) : QAbstractTableModel(parent) {
00006 }
00007
00008 int SegmentsDataModel::rowCount(const QModelIndex &parent) const {
00009     Q_UNUSED(parent);
00010     return segDataLineList.size();
00011 }
00012
00013 int SegmentsDataModel::columnCount(const QModelIndex &parent) const {
00014     Q_UNUSED(parent);
00015     return SegmentsDataData::SIZE;
00016 }
00017
00018 QVariant SegmentsDataModel::data(const QModelIndex &index, int role) const {
00019     if(!index.isValid()) return QVariant();
00020
00021     if(index.row() >= segDataLineList.size() || index.row() < 0) return QVariant();
00022
00023     if (role == Qt::DisplayRole) {
00024         SegmentsDataData line = segDataLineList.at(index.row());
00025         if(index.column() == 1) {
00026             if(line.dataType==3) {
00027                 int i = 0;
00028                 QList<PairsData> pairsList = pairsModel->getPairs();
00029                 for(i=0;i<pairsList.size();i++)
00030                     if(pairsList[i].pairType==10) break;
00031                 if(pairsList.size()>=line.entrancePairIndex&&i<pairsList.size()) {
00032                     PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
00033                     return QString("<center>%1</center>").arg(pairsModel->getReactionLabelTotalCapture(firstPair));
00034                 } else return QString("<center><font
style=' color:red;font-weight:bold;'>UNDEFINED</font></center>");
00035             } else {
00036                 if(pairsModel->getPairs().size()>=line.entrancePairIndex&&pairsModel->getPairs().size()>=line.exitPairIndex)
00037                 {
00038                     PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
00039                     PairsData secondPair=pairsModel->getPairs().at(line.exitPairIndex-1);
00040                     return QString("<center>%1</center>").arg(pairsModel->getReactionLabel(firstPair,secondPair));
00041                 } else return QString("<center><font
style=' color:red;font-weight:bold;'>UNDEFINED</font></center>");
00042             } else if(index.column() == 2) return QVariant();
00043             else if(index.column() == 3) {
00044                 if(line.lowEnergy==line.highEnergy) return line.lowEnergy;
00045                 else return QString("%1-%2").arg(line.lowEnergy).arg(line.highEnergy);
00046             } else if(index.column() == 4) return QVariant();
00047             else if(index.column() == 5) {
00048                 if(line.lowAngle==line.highAngle) return line.lowAngle;
00049                 else return QString("%1-%2").arg(line.lowAngle).arg(line.highAngle);
00050             } else if(index.column() == 6) return QVariant();
00051             else if(index.column() == 7) {
00052                 if(line.dataType==2) {
00053                     QChar orbital;
00054                     switch (line.phaseL) {
00055                         case 0:
00056                             orbital='s';
00057                             break;
00058                         case 1:
00059                             orbital='p';
00060                             break;
00061                         case 2:
00062                             orbital='d';
00063                             break;
00064                         case 3:
00065                             orbital='f';
00066                             break;
00067                         case 4:
00068                             orbital='g';
00069                             break;
00070                         case 5:
00071                             orbital='h';
```

```

00072         break;
00073     case 6:
00074         orbital='i';
00075         break;
00076     default:
00077         orbital='?';
00078     }
00079     QString tempSpin;
00080     if(((int)(line.phaseJ*2))%2!=0&&line.phaseJ!=0.)
tempSpin=QString("%1/2").arg((int)(line.phaseJ*2));
00081     else tempSpin=QString("%1").arg(line.phaseJ);
00082     return QString("<center>Phase Shift [%1<sub>%2</sub>]</center>").arg(orbital).arg(tempSpin);
00083     } else if(line.dataType==1) return QString(tr("<center>Differential</center>"));
00084     else if(line.dataType==3) return QString(tr("<center>Total Capture</center>"));
00085     else return QString(tr("<center>Angle Integrated</center>"));
00086     } else if(index.column() == 8) return line.dataFile;
00087     else if(index.column() == 9) {
00088         if(line.varyNorm==1) return QString("<center><font
style='color:red;font-weight:bold;'>%1</font></center>").arg(line.dataNorm,0,'g',2);
00089         else return QString("<center>%1</center>").arg(line.dataNorm,0,'g',2);
00090     } else if(index.column() == 10) return QVariant();
00091     else if(index.column() == 11) {
00092         if(line.varyNorm==1) return QString(tr("YES"));
00093         else return QString(tr("NO"));
00094     } else if(index.column() == 12) {
00095         return QVariant();
00096     } else if(index.column() == 13) {
00097         return QVariant();
00098     }
00099     } else if (role == Qt::EditRole) {
00100         SegmentsDataData line = segDataLineList.at(index.row());
00101         if(index.column() == 1) return line.entrancePairIndex;
00102         else if(index.column() == 2) return line.exitPairIndex;
00103         else if(index.column() == 3) return line.lowEnergy;
00104         else if(index.column() == 4) return line.highEnergy;
00105         else if(index.column() == 5) return line.lowAngle;
00106         else if(index.column() == 6) return line.highAngle;
00107         else if(index.column() == 7) return line.dataType;
00108         else if(index.column() == 8) return line.dataFile;
00109         else if(index.column() == 9) return line.dataNorm;
00110         else if(index.column() == 10) return line.dataNormError;
00111         else if(index.column() == 11) return line.varyNorm;
00112         else if(index.column() == 12) return line.phaseJ;
00113         else if(index.column() == 13) return line.phaseL;
00114     } else if (role==Qt::CheckStateRole && index.column()==0) {
00115         SegmentsDataData line = segDataLineList.at(index.row());
00116         if(line.isActive==1) return Qt::Checked;
00117         else return Qt::Unchecked;
00118     } else if(role == Qt::TextAlignmentRole) return Qt::AlignCenter;
00119     return QVariant();
00120 }
00121 }
00122
00123 QVariant SegmentsDataModel::headerData(int section, Qt::Orientation orientation, int role) const {
00124     if(role!= Qt::DisplayRole) return QVariant();
00125     if(orientation == Qt::Horizontal) {
00126         switch(section) {
00127             case 0:
00128                 return tr("");
00129             case 1:
00130                 return tr("Reaction");
00131             case 2:
00132                 return QVariant();
00133             case 3:
00134                 return tr("Energy\nRange");
00135             case 4:
00136                 return QVariant();
00137             case 5:
00138                 return tr("Angle\nRange");
00139             case 6:
00140                 return QVariant();
00141             case 7:
00142                 return tr("Data Type");
00143             case 8:
00144                 return tr("Data File");
00145             case 9:
00146                 return tr("Data\nNorm.");
00147             case 10:
00148                 return tr("Data\nNorm. Err.");
00149             case 11:
00150                 return tr("Vary\nNorm.?");
00151             case 12:
00152                 return QVariant();
00153             case 13:
00154                 return QVariant();
00155             default:
00156                 return QVariant();

```



```

00157     }
00158 } else if(orientation == Qt::Vertical) {
00159     return section+1;
00160 }
00161 return QVariant();
00162 }
00163
00164 bool SegmentsDataModel::setData(const QModelIndex &index, const QVariant &value, int role) {
00165     if (index.isValid() && role == Qt::EditRole) {
00166         int row = index.row();
00167         SegmentsDataData tempData = segDataLineList.value(row);
00168         if(index.column() == 0) tempData.isActive=value.toInt();
00169         else if(index.column() == 1) tempData.entrancePairIndex=value.toInt();
00170         else if(index.column() == 2) tempData.exitPairIndex=value.toInt();
00171         else if(index.column() == 3) tempData.lowEnergy=value.toDouble();
00172         else if(index.column() == 4) tempData.highEnergy=value.toDouble();
00173         else if(index.column() == 5) tempData.lowAngle=value.toDouble();
00174         else if(index.column() == 6) tempData.highAngle=value.toDouble();
00175         else if(index.column() == 7) tempData.dataType=value.toInt();
00176         else if(index.column() == 8) tempData.dataFile=value.toString();
00177         else if(index.column() == 9) tempData.dataNorm=value.toDouble();
00178         else if(index.column() == 10) tempData.dataNormError=value.toDouble();
00179         else if(index.column() == 11) tempData.varyNorm=value.toInt();
00180         else if(index.column() == 12) tempData.phaseJ=value.toDouble();
00181         else if(index.column() == 13) tempData.phaseL=value.toInt();
00182         else return false;
00183
00184         segDataLineList.replace(row,tempData);
00185         emit(dataChanged(index,index));
00186         return true;
00187     } else if(role== Qt::CheckStateRole) {
00188         int row = index.row();
00189         SegmentsDataData tempData = segDataLineList.value(row);
00190         if(index.column()==0) {
00191             if(value==Qt::Checked) tempData.isActive=1;
00192             else tempData.isActive=0;
00193         } else return false;
00194         segDataLineList.replace(row,tempData);
00195         emit(dataChanged(index,index));
00196         return true;
00197     }
00198     return false;
00199 }
00200
00201 bool SegmentsDataModel::insertRows(int position, int rows, const QModelIndex &index) {
00202     Q_UNUSED(index);
00203     if(rows>0) {
00204         beginInsertRows(QModelIndex(),position,position+rows-1);
00205         for(int row=0; row<rows; row++) {
00206             SegmentsDataData tempData;
00207             segDataLineList.insert(position,tempData);
00208         }
00209         endInsertRows();
00210     }
00211     return true;
00212 }
00213
00214 bool SegmentsDataModel::removeRows(int position, int rows, const QModelIndex &index) {
00215     Q_UNUSED(index);
00216     if(rows>0) {
00217         beginRemoveRows(QModelIndex(),position,position+rows-1);
00218         for(int row=0; row<rows;++row) {
00219             segDataLineList.removeAt(position);
00220         }
00221         endRemoveRows();
00222     }
00223     return true;
00224 }
00225
00226 Qt::ItemFlags SegmentsDataModel::flags(const QModelIndex &index) const {
00227     if (!index.isValid()) return Qt::ItemIsEnabled;
00228     if(index.column()==0) return QAbstractTableModel::flags(index) | Qt::ItemIsUserCheckable;
00229     return QAbstractTableModel::flags(index);
00230 }
00231
00232 int SegmentsDataModel::isSegDataLine(const SegmentsDataData &line) const {
00233     int foundLine=-1;
00234     for(int i=0;i<segDataLineList.size();i++) {
00235         SegmentsDataData tempLine=segDataLineList.value(i);
00236         if(tempLine.entrancePairIndex==line.entrancePairIndex&&
00237            tempLine.exitPairIndex==line.exitPairIndex&&
00238            tempLine.lowEnergy==line.lowEnergy&&
00239            tempLine.highEnergy==line.highEnergy&&
00240            tempLine.lowAngle==line.lowAngle&&
00241            tempLine.highAngle==line.highAngle&&
00242            tempLine.dataType==line.dataType&&
00243            tempLine.dataFile==line.dataFile&&

```

```

00244         tempLine.dataNorm==line.dataNorm&&
00245         tempLine.dataNormError==line.dataNormError&&
00246         tempLine.varyNorm==line.varyNorm&&
00247         tempLine.phaseJ==line.phaseJ&&
00248         tempLine.phaseL==line.phaseL) {
00249     foundLine=i;
00250     break;
00251 }
00252 }
00253 return foundLine;
00254 }
00255
00256 void SegmentsDataModel::setPairsModel(PairsModel* model) {
00257     pairsModel=model;
00258 }
00259
00260 QString SegmentsDataModel::getReactionLabel(const QModelIndex &index) {
00261     SegmentsDataData line = segDataLineList.at(index.row());
00262     if(line.dataType==3) {
00263         int i = 0;
00264         QList<PairsData> pairsList = pairsModel->getPairs();
00265         for(i=0;i<pairsList.size();i++)
00266             if(pairsList[i].pairType==10) break;
00267         if(pairsList.size()>=line.entrancePairIndex&&i<pairsList.size()) {
00268             PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
00269             return pairsModel->getReactionLabelTotalCapture(firstPair);
00270         }
00271         return QString("<font style='color:red;font-weight:bold;'>UNDEFINED</font>");
00272     } else {
00273         int numPairs = pairsModel->getPairs().size();
00274         if(line.entrancePairIndex-1>=numPairs ||
00275            line.exitPairIndex-1>=numPairs) return QString("<font
style='color:red;font-weight:bold;'>UNDEFINED</font>");
00276         PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
00277         PairsData secondPair=pairsModel->getPairs().at(line.exitPairIndex-1);
00278         return pairsModel->getReactionLabel(firstPair,secondPair);
00279     }
00280 }

```

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.](#)

```

00001 #include <QGridLayout>
00002 #include <QMessageBox>
00003 #include <QGridLayout>
00004 #include <QGroupBox>
00005 #include <QHeaderView>
00006 #include <QTextStream>
00007
00008 #include "SegmentsTab.h"
00009 #include "RichTextDelegate.h"
00010 #include "InfoDialog.h"
00011
00012 SegmentsTab::SegmentsTab(QWidget *parent) : QWidget(parent) {
00013     segmentsDataModel = new SegmentsDataModel;
00014     segmentsDataView = new QTableView;
00015     segmentsDataView->setModel(segmentsDataModel);

```

```

00016   RichTextDelegate *rt = new RichTextDelegate();
00017   segmentsDataView->setItemDelegateForColumn(1, rt);
00018   segmentsDataView->setColumnHidden(2, true);
00019   segmentsDataView->setColumnHidden(4, true);
00020   segmentsDataView->setColumnHidden(6, true);
00021   segmentsDataView->horizontalHeader()->setStretchLastSection(true);
00022   segmentsDataView->horizontalHeader()->setHighlightSections(false);
00023   segmentsDataView->setSelectionBehavior(QAbstractItemView::SelectRows);
00024   segmentsDataView->setSelectionMode(QAbstractItemView::SingleSelection);
00025   segmentsDataView->setEditTriggers(QAbstractItemView::NoEditTriggers);
00026   segmentsDataView->setShowGrid(false);
00027   segmentsDataView->setColumnWidth(0, 27);
00028   segmentsDataView->horizontalHeader()->setSectionResizeMode(0, QHeaderView::Fixed);
00029   segmentsDataView->horizontalHeader()->setSectionResizeMode(1, QHeaderView::Fixed);
00030   segmentsDataView->horizontalHeader()->setSectionResizeMode(2, QHeaderView::Fixed);
00031   for(int i = 1; i<3; i++) segmentsDataView->setColumnWidth(i, 200);
00032   for(int i = 3; i<7; i++) segmentsDataView->setColumnWidth(i, 120);
00033   segmentsDataView->setColumnWidth(7, 140);
00034   segmentsDataView->setItemDelegateForColumn(7, rt);
00035   segmentsDataView->setColumnWidth(8, 220);
00036   segmentsDataView->setItemDelegateForColumn(9, rt);
00037   segmentsDataView->setColumnHidden(10, true);
00038   segmentsDataView->setColumnHidden(11, true);
00039   segmentsDataView->setColumnHidden(12, true);
00040   segmentsDataView->setColumnHidden(13, true);
00041
00042   connect(segmentsDataView->selectionModel(), SIGNAL(selectionChanged(QItemSelection,QItemSelection)), this, SLOT(updateSegData));
00043   connect(segmentsDataView, SIGNAL(doubleClicked(QModelIndex)), this, SLOT(editSegDataLine()));
00044
00045   segmentsTestModel = new SegmentsTestModel;
00046   segmentsTestView = new QTableView;
00047   segmentsTestView->setModel(segmentsTestModel);
00048   segmentsTestView->setItemDelegateForColumn(1, rt);
00049   segmentsTestView->setColumnHidden(2, true);
00050   segmentsTestView->setColumnHidden(4, true);
00051   segmentsTestView->setColumnHidden(7, true);
00052   segmentsTestView->horizontalHeader()->setStretchLastSection(true);
00053   segmentsTestView->horizontalHeader()->setHighlightSections(false);
00054   segmentsTestView->setSelectionBehavior(QAbstractItemView::SelectRows);
00055   segmentsTestView->setSelectionMode(QAbstractItemView::SingleSelection);
00056   segmentsTestView->setEditTriggers(QAbstractItemView::NoEditTriggers);
00057   segmentsTestView->setShowGrid(false);
00058   segmentsTestView->setColumnWidth(0, 27);
00059   segmentsTestView->horizontalHeader()->setSectionResizeMode(0, QHeaderView::Fixed);
00060   segmentsTestView->horizontalHeader()->setSectionResizeMode(1, QHeaderView::Fixed);
00061   segmentsTestView->horizontalHeader()->setSectionResizeMode(2, QHeaderView::Fixed);
00062   for(int i = 1; i<3; i++) segmentsTestView->setColumnWidth(i, 200);
00063   segmentsTestView->setColumnWidth(3, 120);
00064   segmentsTestView->setColumnWidth(5, 90);
00065   segmentsTestView->setColumnWidth(6, 120);
00066   for(int i = 8; i<SegmentsTestData::SIZE; i++) segmentsTestView->setColumnWidth(i, 90);
00067   segmentsTestView->setItemDelegateForColumn(9, rt);
00068   segmentsTestView->setColumnHidden(10, true);
00069   segmentsTestView->setColumnHidden(11, true);
00070   segmentsTestView->setColumnHidden(12, true);
00071
00072   connect(segmentsTestView->selectionModel(), SIGNAL(selectionChanged(QItemSelection,QItemSelection)), this, SLOT(updateSegData));
00073   connect(segmentsTestView, SIGNAL(doubleClicked(QModelIndex)), this, SLOT(editSegTestLine()));
00074
00075   //segDataAddButton=new QPushButton(tr("Add Line"));
00076   //segDataDeleteButton = new QPushButton(tr("Delete Line"));
00077   segDataAddButton=new QPushButton(tr("+"));
00078   segDataAddButton->setMaximumSize(28, 28);
00079   segDataDeleteButton = new QPushButton(tr("-"));
00080   segDataDeleteButton->setEnabled(false);
00081   segDataDeleteButton->setMaximumSize(28, 28);
00082   segDataUpButton = new QPushButton;
00083   segDataUpButton->setIcon(style()->standardIcon(QStyle::SP_ArrowUp));
00084   segDataUpButton->setEnabled(false);
00085   segDataUpButton->setMaximumSize(28, 28);
00086   segDataDownButton = new QPushButton;
00087   segDataDownButton->setIcon(style()->standardIcon(QStyle::SP_ArrowDown));
00088   segDataDownButton->setEnabled(false);
00089   segDataDownButton->setMaximumSize(28, 28);
00090   connect(segDataAddButton, SIGNAL(clicked()), this, SLOT(addSegDataLine()));
00091   connect(segDataDeleteButton, SIGNAL(clicked()), this, SLOT(deleteSegDataLine()));
00092   connect(segDataUpButton, SIGNAL(clicked()), this, SLOT(moveSegDataLineUp()));
00093   connect(segDataDownButton, SIGNAL(clicked()), this, SLOT(moveSegDataLineDown()));
00094
00095   segTestAddButton=new QPushButton(tr("+"));
00096   segTestAddButton->setMaximumSize(28, 28);
00097   segTestDeleteButton = new QPushButton(tr("-"));
00098   segTestDeleteButton->setMaximumSize(28, 28);
00099   segTestDeleteButton->setEnabled(false);
00100   segTestUpButton = new QPushButton;
00101   segTestUpButton->setIcon(style()->standardIcon(QStyle::SP_ArrowUp));
00102   segTestUpButton->setEnabled(false);

```

```

00101     segTestUpButton->setMaximumSize(28,28);
00102     segTestDownButton = new QPushButton;
00103     segTestDownButton->setIcon(style()->standardIcon(QStyle::SP_ArrowDown));
00104     segTestDownButton->setEnabled(false);
00105     segTestDownButton->setMaximumSize(28,28);
00106     connect(segTestAddButton,SIGNAL(clicked()),this,SLOT(addSegTestLine()));
00107     connect(segTestDeleteButton,SIGNAL(clicked()),this,SLOT(deleteSegTestLine()));
00108     connect(segTestUpButton,SIGNAL(clicked()),this,SLOT(moveSegTestLineUp()));
00109     connect(segTestDownButton,SIGNAL(clicked()),this,SLOT(moveSegTestLineDown()));
00110
00111     QGroupBox *segDataBox = new QGroupBox(tr("Segments From Data"));
00112     QGridLayout *segDataLayout = new QGridLayout;
00113     segDataLayout->addWidget(segmentsDataView,0,0);
00114     QGridLayout *segDataButtonBox = new QGridLayout;
00115     segDataButtonBox->addWidget(segDataAddButton,0,0);
00116     segDataButtonBox->addWidget(segDataDeleteButton,0,1);
00117     segDataButtonBox->addItem(new QSpacerItem(28,28),0,2);
00118     segDataButtonBox->addWidget(segDataUpButton,0,3);
00119     segDataButtonBox->addWidget(segDataDownButton,0,4);
00120     segDataButtonBox->setColumnStretch(0,0);
00121     segDataButtonBox->setColumnStretch(1,0);
00122     segDataButtonBox->setColumnStretch(2,1);
00123     segDataButtonBox->setColumnStretch(3,0);
00124     segDataButtonBox->setColumnStretch(4,0);
00125 #ifdef MACX_SPACING
00126     segDataButtonBox->setHorizontalSpacing(11);
00127 #else
00128     segDataButtonBox->setHorizontalSpacing(0);
00129 #endif
00130     segDataLayout->addLayout(segDataButtonBox,1,0);
00131     segDataBox->setLayout(segDataLayout);
00132
00133     QGroupBox *segTestBox = new QGroupBox(tr("Segments Without Data"));
00134     QGridLayout *segTestLayout = new QGridLayout;
00135     segTestLayout->addWidget(segmentsTestView,0,0);
00136     QGridLayout *segTestButtonBox = new QGridLayout;
00137     segTestButtonBox->addWidget(segTestAddButton,0,0);
00138     segTestButtonBox->addWidget(segTestDeleteButton,0,1);
00139     segTestButtonBox->addItem(new QSpacerItem(28,28),0,2);
00140     segTestButtonBox->addWidget(segTestUpButton,0,3);
00141     segTestButtonBox->addWidget(segTestDownButton,0,4);
00142     segTestButtonBox->setColumnStretch(0,0);
00143     segTestButtonBox->setColumnStretch(1,0);
00144     segTestButtonBox->setColumnStretch(2,1);
00145     segTestButtonBox->setColumnStretch(3,0);
00146     segTestButtonBox->setColumnStretch(4,0);
00147 #ifdef MACX_SPACING
00148     segTestButtonBox->setHorizontalSpacing(11);
00149 #else
00150     segTestButtonBox->setHorizontalSpacing(0);
00151 #endif
00152     segTestLayout->addLayout(segTestButtonBox,1,0);
00153     segTestBox->setLayout(segTestLayout);
00154
00155     QGridLayout *mainLayout= new QGridLayout;
00156     mainLayout->addWidget(segDataBox,0,0);
00157     mainLayout->addWidget(segTestBox,1,0);
00158
00159     setLayout(mainLayout);
00160 }
00161
00162 SegmentsTestModel* SegmentsTab::getSegmentsTestModel() {
00163     return segmentsTestModel;
00164 }
00165
00166 SegmentsDataModel* SegmentsTab::getSegmentsDataModel() {
00167     return segmentsDataModel;
00168 }
00169
00170 void SegmentsTab::deleteSegDataLine() {
00171     QItemSelectionModel *selectionModel = segmentsDataView->selectionModel();
00172     QModelIndexList indexes = selectionModel->selectedRows();
00173     QModelIndex index=indexes.at(0);
00174
00175     segmentsDataModel->removeRows(index.row(),1,QModelIndex());
00176     updateSegDataButtons(selectionModel->selection());
00177 }
00178
00179 void SegmentsTab::deleteSegTestLine() {
00180     QItemSelectionModel *selectionModel = segmentsTestView->selectionModel();
00181     QModelIndexList indexes = selectionModel->selectedRows();
00182     QModelIndex index=indexes.at(0);
00183
00184     segmentsTestModel->removeRows(index.row(),1,QModelIndex());
00185     updateSegTestButtons(selectionModel->selection());
00186 }
00187

```

```

00188 void SegmentsTab::addSegDataLine() {
00189     AddSegDataDialog aDialog;
00190     if(aDialog.exec()) {
00191         SegmentsDataData newLine;
00192         newLine.isActive=1;
00193         newLine.entrancePairIndex=aDialog.entrancePairIndexSpin->value();
00194         if(aDialog.dataTypeCombo->currentIndex()==3)
00195             newLine.exitPairIndex=-1;
00196         else newLine.exitPairIndex=aDialog.exitPairIndexSpin->value();
00197         newLine.lowEnergy=aDialog.lowEnergyText->text().toDouble();
00198         newLine.highEnergy=aDialog.highEnergyText->text().toDouble();
00199         newLine.lowAngle=aDialog.lowAngleText->text().toDouble();
00200         newLine.highAngle=aDialog.highAngleText->text().toDouble();
00201         newLine.dataType=aDialog.dataTypeCombo->currentIndex();
00202         newLine.dataFile=aDialog.dataFileText->text();
00203         newLine.dataNorm=aDialog.dataNormText->text().toDouble();
00204         newLine.dataNormError=aDialog.dataNormErrorText->text().toDouble();
00205         if(aDialog.varyNormCheck->isChecked()) newLine.varyNorm=1;
00206         else newLine.varyNorm=0;
00207         newLine.phaseJ=aDialog.phaseJValueText->text().toDouble();
00208         newLine.phaseL=aDialog.phaseLValueText->text().toInt();
00209         addSegDataLine(newLine);
00210     }
00211 }
00212
00213 void SegmentsTab::addSegDataLine(SegmentsDataData line) {
00214     QList<SegmentsDataData> lines = segmentsDataModel->getLines();
00215     if(segmentsDataModel->isSegDataLine(line)==-1) {
00216         segmentsDataModel->insertRows(lines.size(),1,QModelIndex());
00217         QModelIndex index = segmentsDataModel->index(lines.size(),0,QModelIndex());
00218         segmentsDataModel->setData(index,line.isActive,Qt::EditRole);
00219         index = segmentsDataModel->index(lines.size(),1,QModelIndex());
00220         segmentsDataModel->setData(index,line.entrancePairIndex,Qt::EditRole);
00221         index = segmentsDataModel->index(lines.size(),2,QModelIndex());
00222         segmentsDataModel->setData(index,line.exitPairIndex,Qt::EditRole);
00223         index = segmentsDataModel->index(lines.size(),3,QModelIndex());
00224         segmentsDataModel->setData(index,line.lowEnergy,Qt::EditRole);
00225         index = segmentsDataModel->index(lines.size(),4,QModelIndex());
00226         segmentsDataModel->setData(index,line.highEnergy,Qt::EditRole);
00227         index = segmentsDataModel->index(lines.size(),5,QModelIndex());
00228         segmentsDataModel->setData(index,line.lowAngle,Qt::EditRole);
00229         index = segmentsDataModel->index(lines.size(),6,QModelIndex());
00230         segmentsDataModel->setData(index,line.highAngle,Qt::EditRole);
00231         index = segmentsDataModel->index(lines.size(),7,QModelIndex());
00232         segmentsDataModel->setData(index,line.dataType,Qt::EditRole);
00233         index = segmentsDataModel->index(lines.size(),8,QModelIndex());
00234         segmentsDataModel->setData(index,line.dataFile,Qt::EditRole);
00235         index = segmentsDataModel->index(lines.size(),9,QModelIndex());
00236         segmentsDataModel->setData(index,line.dataNorm,Qt::EditRole);
00237         index = segmentsDataModel->index(lines.size(),10,QModelIndex());
00238         segmentsDataModel->setData(index,line.dataNormError,Qt::EditRole);
00239         index = segmentsDataModel->index(lines.size(),11,QModelIndex());
00240         segmentsDataModel->setData(index,line.varyNorm,Qt::EditRole);
00241         index = segmentsDataModel->index(lines.size(),12,QModelIndex());
00242         segmentsDataModel->setData(index,line.phaseJ,Qt::EditRole);
00243         index = segmentsDataModel->index(lines.size(),13,QModelIndex());
00244         segmentsDataModel->setData(index,line.phaseL,Qt::EditRole);
00245         segmentsDataView->resizeRowToContents(lines.size());
00246         updateSegDataButtons(segmentsDataView->selectionModel()->selection());
00247     } else {
00248         QMessageBox::information(this,tr("Duplicate Line"),tr("This line already exists."));
00249     }
00250 }
00251
00252 void SegmentsTab::addSegTestLine() {
00253     AddSegTestDialog aDialog;
00254     if(aDialog.exec()) {
00255         SegmentsTestData newLine;
00256         newLine.isActive=1;
00257         newLine.entrancePairIndex=aDialog.entrancePairIndexSpin->value();
00258         if(aDialog.dataTypeCombo->currentIndex()==4)
00259             newLine.exitPairIndex=-1;
00260         else newLine.exitPairIndex=aDialog.exitPairIndexSpin->value();
00261         newLine.lowEnergy=aDialog.lowEnergyText->text().toDouble();
00262         newLine.highEnergy=aDialog.highEnergyText->text().toDouble();
00263         newLine.energyStep=aDialog.energyStepText->text().toDouble();
00264         newLine.lowAngle=aDialog.lowAngleText->text().toDouble();
00265         newLine.highAngle=aDialog.highAngleText->text().toDouble();
00266         newLine.angleStep=aDialog.angleStepText->text().toDouble();
00267         newLine.dataType=aDialog.dataTypeCombo->currentIndex();
00268         newLine.phaseJ=aDialog.phaseJValueText->text().toDouble();
00269         newLine.phaseL=aDialog.phaseLValueText->text().toInt();
00270         newLine.maxAngDistOrder=aDialog.angDistSpin->value();
00271         addSegTestLine(newLine);
00272     }
00273 }
00274

```

```

00275 void SegmentsTab::addSegTestLine(SegmentsTestData line) {
00276     QList<SegmentsTestData> lines = segmentsTestModel->getLines();
00277     if (segmentsTestModel->isSegTestLine(line)==-1) {
00278         segmentsTestModel->insertRows(lines.size(),1,QModelIndex());
00279         QModelIndex index = segmentsTestModel->index(lines.size(),0,QModelIndex());
00280         segmentsTestModel->setData(index,line.isActive,Qt::EditRole);
00281         index = segmentsTestModel->index(lines.size(),1,QModelIndex());
00282         segmentsTestModel->setData(index,line.entrancePairIndex,Qt::EditRole);
00283         index = segmentsTestModel->index(lines.size(),2,QModelIndex());
00284         segmentsTestModel->setData(index,line.exitPairIndex,Qt::EditRole);
00285         index = segmentsTestModel->index(lines.size(),3,QModelIndex());
00286         segmentsTestModel->setData(index,line.lowEnergy,Qt::EditRole);
00287         index = segmentsTestModel->index(lines.size(),4,QModelIndex());
00288         segmentsTestModel->setData(index,line.highEnergy,Qt::EditRole);
00289         index = segmentsTestModel->index(lines.size(),5,QModelIndex());
00290         segmentsTestModel->setData(index,line.energyStep,Qt::EditRole);
00291         index = segmentsTestModel->index(lines.size(),6,QModelIndex());
00292         segmentsTestModel->setData(index,line.lowAngle,Qt::EditRole);
00293         index = segmentsTestModel->index(lines.size(),7,QModelIndex());
00294         segmentsTestModel->setData(index,line.highAngle,Qt::EditRole);
00295         index = segmentsTestModel->index(lines.size(),8,QModelIndex());
00296         segmentsTestModel->setData(index,line.angleStep,Qt::EditRole);
00297         index = segmentsTestModel->index(lines.size(),9,QModelIndex());
00298         segmentsTestModel->setData(index,line.dataType,Qt::EditRole);
00299         index = segmentsTestModel->index(lines.size(),10,QModelIndex());
00300         segmentsTestModel->setData(index,line.phaseJ,Qt::EditRole);
00301         index = segmentsTestModel->index(lines.size(),11,QModelIndex());
00302         segmentsTestModel->setData(index,line.phaseL,Qt::EditRole);
00303         index = segmentsTestModel->index(lines.size(),12,QModelIndex());
00304         segmentsTestModel->setData(index,line.maxAngDistOrder,Qt::EditRole);
00305         segmentsTestView->resizeRowToContents(lines.size());
00306         updateSegTestButtons(segmentsTestView->selectionModel()->selection());
00307     } else {
00308         QMessageBox::information(this,tr("Duplicate Line"),tr("This line already exists."));
00309     }
00310 }
00311
00312 void SegmentsTab::editSegDataLine() {
00313     QItemSelectionModel *selectionModel = segmentsDataView->selectionModel();
00314     QModelIndexList indexes = selectionModel->selectedRows();
00315     QModelIndex index=indexes[0];
00316
00317     QModelIndex i=segmentsDataModel->index(index.row(),1,QModelIndex());
00318     QVariant var=segmentsDataModel->data(i,Qt::EditRole);
00319     int entrancePairIndex=var.toInt();
00320     i=segmentsDataModel->index(index.row(),2,QModelIndex());
00321     var=segmentsDataModel->data(i,Qt::EditRole);
00322     int exitPairIndex=var.toInt();
00323     i=segmentsDataModel->index(index.row(),3,QModelIndex());
00324     var=segmentsDataModel->data(i,Qt::EditRole);
00325     QString lowEnergy=var.toString();
00326     i=segmentsDataModel->index(index.row(),4,QModelIndex());
00327     var=segmentsDataModel->data(i,Qt::EditRole);
00328     QString highEnergy=var.toString();
00329     i=segmentsDataModel->index(index.row(),5,QModelIndex());
00330     var=segmentsDataModel->data(i,Qt::EditRole);
00331     QString lowAngle=var.toString();
00332     i=segmentsDataModel->index(index.row(),6,QModelIndex());
00333     var=segmentsDataModel->data(i,Qt::EditRole);
00334     QString highAngle=var.toString();
00335     i=segmentsDataModel->index(index.row(),7,QModelIndex());
00336     var=segmentsDataModel->data(i,Qt::EditRole);
00337     int dataType=var.toInt();
00338     i=segmentsDataModel->index(index.row(),8,QModelIndex());
00339     var=segmentsDataModel->data(i,Qt::EditRole);
00340     QString dataFile=var.toString();
00341     i=segmentsDataModel->index(index.row(),9,QModelIndex());
00342     var=segmentsDataModel->data(i,Qt::EditRole);
00343     QString dataNorm=var.toString();
00344     i=segmentsDataModel->index(index.row(),10,QModelIndex());
00345     var=segmentsDataModel->data(i,Qt::EditRole);
00346     QString dataNormError=var.toString();
00347     i=segmentsDataModel->index(index.row(),11,QModelIndex());
00348     var=segmentsDataModel->data(i,Qt::EditRole);
00349     int varyNorm=var.toInt();
00350     i=segmentsDataModel->index(index.row(),12,QModelIndex());
00351     var=segmentsDataModel->data(i,Qt::EditRole);
00352     QString phaseJ=var.toString();
00353     i=segmentsDataModel->index(index.row(),13,QModelIndex());
00354     var=segmentsDataModel->data(i,Qt::EditRole);
00355     QString phaseL=var.toString();
00356
00357     AddSegDataDialog aDialog;
00358     aDialog.setWindowTitle(tr("Edit a Segment From Data"));
00359     aDialog.entrancePairIndexSpin->setValue(entrancePairIndex);
00360     aDialog.exitPairIndexSpin->setValue(exitPairIndex);

```



```

00362 aDialog.lowEnergyText->setText(lowEnergy);
00363 aDialog.highEnergyText->setText(highEnergy);
00364 aDialog.lowAngleText->setText(lowAngle);
00365 aDialog.highAngleText->setText(highAngle);
00366 aDialog.dataTypeCombo->setCurrentIndex(dataType);
00367 aDialog.dataFileText->setText(dataFile);
00368 aDialog.dataNormText->setText(dataNorm);
00369 aDialog.dataNormErrorText->setText(dataNormError);
00370 if (varyNorm==1) aDialog.varyNormCheck->setChecked(true);
00371 else aDialog.varyNormCheck->setChecked(false);
00372 aDialog.phaseJValueText->setText(phaseJ);
00373 aDialog.phaseLValueText->setText(phaseL);
00374
00375 if (aDialog.exec()) {
00376     int newEntrancePairIndex=aDialog.entrancePairIndexSpin->value();
00377     if (newEntrancePairIndex!=entrancePairIndex) {
00378         i=segmentsDataModel->index(index.row(),1,QModelIndex());
00379         segmentsDataModel->setData(i,newEntrancePairIndex,Qt::EditRole);
00380     }
00381     int newExitPairIndex= (aDialog.dataTypeCombo->currentIndex()==3) ? -1 :
00382     aDialog.exitPairIndexSpin->value();
00383     if (newExitPairIndex!=exitPairIndex) {
00384         i=segmentsDataModel->index(index.row(),2,QModelIndex());
00385         segmentsDataModel->setData(i,newExitPairIndex,Qt::EditRole);
00386     }
00387     QString newLowEnergy=aDialog.lowEnergyText->text();
00388     if (newLowEnergy!=lowEnergy) {
00389         i=segmentsDataModel->index(index.row(),3,QModelIndex());
00390         segmentsDataModel->setData(i,newLowEnergy,Qt::EditRole);
00391     }
00392     QString newHighEnergy=aDialog.highEnergyText->text();
00393     if (newHighEnergy!=highEnergy) {
00394         i=segmentsDataModel->index(index.row(),4,QModelIndex());
00395         segmentsDataModel->setData(i,newHighEnergy,Qt::EditRole);
00396     }
00397     QString newLowAngle=aDialog.lowAngleText->text();
00398     if (newLowAngle!=lowAngle) {
00399         i=segmentsDataModel->index(index.row(),5,QModelIndex());
00400         segmentsDataModel->setData(i,newLowAngle,Qt::EditRole);
00401     }
00402     QString newHighAngle=aDialog.highAngleText->text();
00403     if (newHighAngle!=highAngle) {
00404         i=segmentsDataModel->index(index.row(),6,QModelIndex());
00405         segmentsDataModel->setData(i,newHighAngle,Qt::EditRole);
00406     }
00407     int newDataType=aDialog.dataTypeCombo->currentIndex();
00408     if (newDataType!=dataType) {
00409         i=segmentsDataModel->index(index.row(),7,QModelIndex());
00410         segmentsDataModel->setData(i,newDataType,Qt::EditRole);
00411     }
00412     QString newDataFile=aDialog.dataFileText->text();
00413     if (newDataFile!=dataFile) {
00414         i=segmentsDataModel->index(index.row(),8,QModelIndex());
00415         segmentsDataModel->setData(i,newDataFile,Qt::EditRole);
00416     }
00417     QString newDataNorm=aDialog.dataNormText->text();
00418     if (newDataNorm!=dataNorm) {
00419         i=segmentsDataModel->index(index.row(),9,QModelIndex());
00420         segmentsDataModel->setData(i,newDataNorm,Qt::EditRole);
00421     }
00422     QString newDataNormError=aDialog.dataNormErrorText->text();
00423     if (newDataNormError!=dataNormError) {
00424         i=segmentsDataModel->index(index.row(),10,QModelIndex());
00425         segmentsDataModel->setData(i,newDataNormError,Qt::EditRole);
00426     }
00427     int newVaryNorm=0;
00428     if (aDialog.varyNormCheck->isChecked()) newVaryNorm=1;
00429     if (newVaryNorm!=varyNorm) {
00430         i=segmentsDataModel->index(index.row(),11,QModelIndex());
00431         segmentsDataModel->setData(i,newVaryNorm,Qt::EditRole);
00432     }
00433     QString newPhaseJ=aDialog.phaseJValueText->text();
00434     if (newPhaseJ!=phaseJ) {
00435         i=segmentsDataModel->index(index.row(),12,QModelIndex());
00436         segmentsDataModel->setData(i,newPhaseJ,Qt::EditRole);
00437     }
00438     QString newPhaseL=aDialog.phaseLValueText->text();
00439     if (newPhaseL!=phaseL) {
00440         i=segmentsDataModel->index(index.row(),13,QModelIndex());
00441         segmentsDataModel->setData(i,newPhaseL,Qt::EditRole);
00442     }
00443 }
00444 }
00445
00446 void SegmentsTab::editSegTestLine() {
00447     QTableWidgetItem *selectionModel = segmentsTestView->selectionModel();
00448     QModelIndexList indexes = selectionModel->selectedRows();

```

```

00449     QModelIndex index=indexes[0];
00450
00451     QModelIndex i=segmentsTestModel->index(index.row(),1,QModelIndex());
00452     QVariant var=segmentsTestModel->data(i,Qt::EditRole);
00453     int entrancePairIndex=var.toInt();
00454     i=segmentsTestModel->index(index.row(),2,QModelIndex());
00455     var=segmentsTestModel->data(i,Qt::EditRole);
00456     int exitPairIndex=var.toInt();
00457     i=segmentsTestModel->index(index.row(),3,QModelIndex());
00458     var=segmentsTestModel->data(i,Qt::EditRole);
00459     QString lowEnergy=var.toString();
00460     i=segmentsTestModel->index(index.row(),4,QModelIndex());
00461     var=segmentsTestModel->data(i,Qt::EditRole);
00462     QString highEnergy=var.toString();
00463     i=segmentsTestModel->index(index.row(),5,QModelIndex());
00464     var=segmentsTestModel->data(i,Qt::EditRole);
00465     QString energyStep=var.toString();
00466     i=segmentsTestModel->index(index.row(),6,QModelIndex());
00467     var=segmentsTestModel->data(i,Qt::EditRole);
00468     QString lowAngle=var.toString();
00469     i=segmentsTestModel->index(index.row(),7,QModelIndex());
00470     var=segmentsTestModel->data(i,Qt::EditRole);
00471     QString highAngle=var.toString();
00472     i=segmentsTestModel->index(index.row(),8,QModelIndex());
00473     var=segmentsTestModel->data(i,Qt::EditRole);
00474     QString angleStep=var.toString();
00475     i=segmentsTestModel->index(index.row(),9,QModelIndex());
00476     var=segmentsTestModel->data(i,Qt::EditRole);
00477     int dataType=var.toInt();
00478     i=segmentsTestModel->index(index.row(),10,QModelIndex());
00479     var=segmentsTestModel->data(i,Qt::EditRole);
00480     QString phaseJ=var.toString();
00481     i=segmentsTestModel->index(index.row(),11,QModelIndex());
00482     var=segmentsTestModel->data(i,Qt::EditRole);
00483     QString phaseL=var.toString();
00484     i=segmentsTestModel->index(index.row(),12,QModelIndex());
00485     var=segmentsTestModel->data(i,Qt::EditRole);
00486     int maxAngDistOrder=var.toInt();
00487
00488     AddSegTestDialog aDialog;
00489     aDialog.setWindowTitle(tr("Edit a Segment Without Data"));
00490     aDialog.entrancePairIndexSpin->setValue(entrancePairIndex);
00491     aDialog.exitPairIndexSpin->setValue(exitPairIndex);
00492     aDialog.lowEnergyText->setText(lowEnergy);
00493     aDialog.highEnergyText->setText(highEnergy);
00494     aDialog.energyStepText->setText(energyStep);
00495     aDialog.lowAngleText->setText(lowAngle);
00496     aDialog.highAngleText->setText(highAngle);
00497     aDialog.angleStepText->setText(angleStep);
00498     aDialog.dataTypeCombo->setCurrentIndex(dataType);
00499     aDialog.phaseJValueText->setText(phaseJ);
00500     aDialog.phaseLValueText->setText(phaseL);
00501     aDialog.angDistSpin->setValue(maxAngDistOrder);
00502
00503
00504     if(aDialog.exec()) {
00505         int newEntrancePairIndex=aDialog.entrancePairIndexSpin->value();
00506         if(newEntrancePairIndex!=entrancePairIndex) {
00507             i=segmentsTestModel->index(index.row(),1,QModelIndex());
00508             segmentsTestModel->setData(i,newEntrancePairIndex,Qt::EditRole);
00509         }
00510         int newExitPairIndex= (aDialog.dataTypeCombo->currentIndex()==4) ? -1 :
00511             aDialog.exitPairIndexSpin->value();
00512         if(newExitPairIndex!=exitPairIndex) {
00513             i=segmentsTestModel->index(index.row(),2,QModelIndex());
00514             segmentsTestModel->setData(i,newExitPairIndex,Qt::EditRole);
00515         }
00516         QString newLowEnergy=aDialog.lowEnergyText->text();
00517         if(newLowEnergy!=lowEnergy) {
00518             i=segmentsTestModel->index(index.row(),3,QModelIndex());
00519             segmentsTestModel->setData(i,newLowEnergy,Qt::EditRole);
00520         }
00521         QString newHighEnergy=aDialog.highEnergyText->text();
00522         if(newHighEnergy!=highEnergy) {
00523             i=segmentsTestModel->index(index.row(),4,QModelIndex());
00524             segmentsTestModel->setData(i,newHighEnergy,Qt::EditRole);
00525         }
00526         QString newEnergyStep=aDialog.energyStepText->text();
00527         if(newEnergyStep!=energyStep) {
00528             i=segmentsTestModel->index(index.row(),5,QModelIndex());
00529             segmentsTestModel->setData(i,newEnergyStep,Qt::EditRole);
00530         }
00531         QString newLowAngle=aDialog.lowAngleText->text();
00532         if(newLowAngle!=lowAngle) {
00533             i=segmentsTestModel->index(index.row(),6,QModelIndex());
00534             segmentsTestModel->setData(i,newLowAngle,Qt::EditRole);
00535         }
00536     }

```



```

00536     QString newHighAngle=aDialog.highAngleText->text();
00537     if(newHighAngle!=highAngle) {
00538         i=segmentsTestModel->index(index.row(),7,QModelIndex());
00539         segmentsTestModel->setData(i,newHighAngle,Qt::EditRole);
00540     }
00541     QString newAngleStep=aDialog.angleStepText->text();
00542     if(newAngleStep!=angleStep) {
00543         i=segmentsTestModel->index(index.row(),8,QModelIndex());
00544         segmentsTestModel->setData(i,newAngleStep,Qt::EditRole);
00545     }
00546     int newDataType = aDialog.dataTypeCombo->currentIndex();
00547     if(newDataType!=dataType) {
00548         i=segmentsTestModel->index(index.row(),9,QModelIndex());
00549         segmentsTestModel->setData(i,newDataType,Qt::EditRole);
00550     }
00551     QString newPhaseJ=aDialog.phaseJValueText->text();
00552     if(newPhaseJ!=phaseJ) {
00553         i=segmentsTestModel->index(index.row(),10,QModelIndex());
00554         segmentsTestModel->setData(i,newPhaseJ,Qt::EditRole);
00555     }
00556     QString newPhaseL=aDialog.phaseLValueText->text();
00557     if(newPhaseL!=phaseL) {
00558         i=segmentsTestModel->index(index.row(),11,QModelIndex());
00559         segmentsTestModel->setData(i,newPhaseL,Qt::EditRole);
00560     }
00561     int newMaxAngDistOrder=aDialog.angDistSpin->value();
00562     if(newMaxAngDistOrder!=maxAngDistOrder) {
00563         i=segmentsTestModel->index(index.row(),12,QModelIndex());
00564         segmentsTestModel->setData(i,newMaxAngDistOrder,Qt::EditRole);
00565     }
00566 }
00567 }
00568
00569 void SegmentsTab::moveSegDataLineUp() {
00570     moveSegDataLine(1);
00571 }
00572
00573 void SegmentsTab::moveSegDataLineDown() {
00574     moveSegDataLine(0);
00575 }
00576
00577 void SegmentsTab::moveSegDataLine(unsigned int upDown) {
00578     QItemSelectionModel *selectionModel = segmentsDataView->selectionModel();
00579     QModelIndexList selectionList = selectionModel->selectedRows();
00580     QModelIndex selectionIndex=selectionList.at(0);
00581
00582     int previous = selectionIndex.row();
00583     int future;
00584     if(upDown==0) future = previous+1;
00585     else future = previous-1;
00586     SegmentsDataData line = segmentsDataModel->getLines().at(previous);
00587     segmentsDataModel->removeRows(previous,1,QModelIndex());
00588     segmentsDataModel->insertRows(future,1,QModelIndex());
00589     QModelIndex index = segmentsDataModel->index(future,0,QModelIndex());
00590     segmentsDataModel->setData(index,line.isActive,Qt::EditRole);
00591     index = segmentsDataModel->index(future,1,QModelIndex());
00592     segmentsDataModel->setData(index,line.entrancePairIndex,Qt::EditRole);
00593     index = segmentsDataModel->index(future,2,QModelIndex());
00594     segmentsDataModel->setData(index,line.exitPairIndex,Qt::EditRole);
00595     index = segmentsDataModel->index(future,3,QModelIndex());
00596     segmentsDataModel->setData(index,line.lowEnergy,Qt::EditRole);
00597     index = segmentsDataModel->index(future,4,QModelIndex());
00598     segmentsDataModel->setData(index,line.highEnergy,Qt::EditRole);
00599     index = segmentsDataModel->index(future,5,QModelIndex());
00600     segmentsDataModel->setData(index,line.lowAngle,Qt::EditRole);
00601     index = segmentsDataModel->index(future,6,QModelIndex());
00602     segmentsDataModel->setData(index,line.highAngle,Qt::EditRole);
00603     index = segmentsDataModel->index(future,7,QModelIndex());
00604     segmentsDataModel->setData(index,line.dataType,Qt::EditRole);
00605     index = segmentsDataModel->index(future,8,QModelIndex());
00606     segmentsDataModel->setData(index,line.dataFile,Qt::EditRole);
00607     index = segmentsDataModel->index(future,9,QModelIndex());
00608     segmentsDataModel->setData(index,line.dataNorm,Qt::EditRole);
00609     index = segmentsDataModel->index(future,10,QModelIndex());
00610     segmentsDataModel->setData(index,line.dataNormError,Qt::EditRole);
00611     index = segmentsDataModel->index(future,11,QModelIndex());
00612     segmentsDataModel->setData(index,line.varyNorm,Qt::EditRole);
00613     index = segmentsDataModel->index(future,12,QModelIndex());
00614     segmentsDataModel->setData(index,line.phaseJ,Qt::EditRole);
00615     index = segmentsDataModel->index(future,13,QModelIndex());
00616     segmentsDataModel->setData(index,line.phaseL,Qt::EditRole);
00617     segmentsDataView->resizeRowToContents(future);
00618
00619     selectionModel->select(segmentsDataModel->index(future,0,QModelIndex()),
00620         QItemSelectionModel::ClearAndSelect | QItemSelectionModel::Rows);
00621 }
00622

```

```

00623 void SegmentsTab::moveSegTestLineUp() {
00624     moveSegTestLine(1);
00625 }
00626
00627 void SegmentsTab::moveSegTestLineDown() {
00628     moveSegTestLine(0);
00629 }
00630
00631 void SegmentsTab::moveSegTestLine(unsigned int upDown) {
00632     QItemSelectionModel *selectionModel = segmentsTestView->selectionModel();
00633     QModelIndexList selectionList = selectionModel->selectedRows();
00634     QModelIndex selectionIndex=selectionList.at(0);
00635
00636     int previous = selectionIndex.row();
00637     int future;
00638     if(upDown==0) future = previous+1;
00639     else future = previous-1;
00640
00641     SegmentsTestData line = segmentsTestModel->getLines().at(previous);
00642     segmentsTestModel->removeRows(previous,1,QModelIndex());
00643     segmentsTestModel->insertRows(future,1,QModelIndex());
00644     QModelIndex index = segmentsTestModel->index(future,0,QModelIndex());
00645     segmentsTestModel->setData(index,line.isActive,Qt::EditRole);
00646     index = segmentsTestModel->index(future,1,QModelIndex());
00647     segmentsTestModel->setData(index,line.entrancePairIndex,Qt::EditRole);
00648     index = segmentsTestModel->index(future,2,QModelIndex());
00649     segmentsTestModel->setData(index,line.exitPairIndex,Qt::EditRole);
00650     index = segmentsTestModel->index(future,3,QModelIndex());
00651     segmentsTestModel->setData(index,line.lowEnergy,Qt::EditRole);
00652     index = segmentsTestModel->index(future,4,QModelIndex());
00653     segmentsTestModel->setData(index,line.highEnergy,Qt::EditRole);
00654     index = segmentsTestModel->index(future,5,QModelIndex());
00655     segmentsTestModel->setData(index,line.energyStep,Qt::EditRole);
00656     index = segmentsTestModel->index(future,6,QModelIndex());
00657     segmentsTestModel->setData(index,line.lowAngle,Qt::EditRole);
00658     index = segmentsTestModel->index(future,7,QModelIndex());
00659     segmentsTestModel->setData(index,line.highAngle,Qt::EditRole);
00660     index = segmentsTestModel->index(future,8,QModelIndex());
00661     segmentsTestModel->setData(index,line.angleStep,Qt::EditRole);
00662     index = segmentsTestModel->index(future,9,QModelIndex());
00663     segmentsTestModel->setData(index,line.dataType,Qt::EditRole);
00664     index = segmentsTestModel->index(future,10,QModelIndex());
00665     segmentsTestModel->setData(index,line.phaseJ,Qt::EditRole);
00666     index = segmentsTestModel->index(future,11,QModelIndex());
00667     segmentsTestModel->setData(index,line.phaseL,Qt::EditRole);
00668     index = segmentsTestModel->index(future,12,QModelIndex());
00669     segmentsTestModel->setData(index,line.maxAngDistOrder,Qt::EditRole);
00670     segmentsTestView->resizeRowToContents(future);
00671
00672     selectionModel->select(segmentsTestModel->index(future,0,QModelIndex()),
00673         QItemSelectionModel::ClearAndSelect | QItemSelectionModel::Rows);
00674 }
00675
00676 void SegmentsTab::updateSegDataButtons(const QItemSelection &selection) {
00677     QModelIndexList indexes=selection.indexes();
00678
00679     if(indexes.isEmpty()) {
00680         segDataDeleteButton->setEnabled(false);
00681         segDataUpButton->setEnabled(false);
00682         segDataDownButton->setEnabled(false);
00683     } else {
00684         segDataDeleteButton->setEnabled(true);
00685         if(indexes.at(0).row()==0) segDataUpButton->setEnabled(false);
00686         else segDataUpButton->setEnabled(true);
00687         if(indexes.at(0).row()==segmentsDataModel->rowCount(QModelIndex())-1)
00688             segDataDownButton->setEnabled(false);
00689         else segDataDownButton->setEnabled(true);
00690     }
00691 }
00692
00693 void SegmentsTab::updateSegTestButtons(const QItemSelection &selection) {
00694     QModelIndexList indexes=selection.indexes();
00695
00696     if(indexes.isEmpty()) {
00697         segTestDeleteButton->setEnabled(false);
00698         segTestUpButton->setEnabled(false);
00699         segTestDownButton->setEnabled(false);
00700     } else {
00701         segTestDeleteButton->setEnabled(true);
00702         if(indexes.at(0).row()==0) segTestUpButton->setEnabled(false);
00703         else segTestUpButton->setEnabled(true);
00704         if(indexes.at(0).row()==segmentsTestModel->rowCount(QModelIndex())-1)
00705             segTestDownButton->setEnabled(false);
00706         else segTestDownButton->setEnabled(true);
00707     }
00708 }

```

```

00708
00709 bool SegmentsTab::readSegDataFile(QTextStream& inStream) {
00710
00711     int isActive;
00712     int entrancePairIndex;
00713     int exitPairIndex;
00714     double lowEnergy;
00715     double highEnergy;
00716     double lowAngle;
00717     double highAngle;
00718     int dataType;
00719     QString dataFile;
00720     double dataNorm;
00721     double dataNormError;
00722     int varyNorm;
00723     double phaseJ;
00724     int phaseL;
00725
00726     QString line("");
00727     while(!inStream.atEnd() && line.trimmed() != QString("</segmentsData>")) {
00728         line=inStream.readLine();
00729         if(line.trimmed().isEmpty()) continue;
00730         if(!inStream.atEnd() && line.trimmed() != QString("</segmentsData>")) {
00731             QTextStream in(&line);
00732             in » isActive » entrancePairIndex » exitPairIndex » lowEnergy » highEnergy » lowAngle »
highAngle
00733             » dataType;
00734             if(dataType==2) in»phaseJ»phaseL;
00735             else {
00736                 phaseJ=0.;
00737                 phaseL=0;
00738             }
00739             in » dataNorm » varyNorm;
00740             QString restOfLine=in.readLine();
00741             QTextStream stm(&restOfLine);
00742             stm»dataNormError;
00743             if(stm.status() !=QTextStream::Ok) {
00744                 dataNormError=0.;
00745                 dataFile=restOfLine.trimmed();
00746             } else dataFile=stm.readLine().trimmed();
00747             if(in.status() !=QTextStream::Ok) return false;
00748             SegmentsDataData newLine =
{isActive,entrancePairIndex,exitPairIndex,lowEnergy,highEnergy,lowAngle,
highAngle,dataType,dataFile,dataNorm,dataNormError,varyNorm,phaseJ,phaseL};
00749             addSegDataLine(newLine);
00750         }
00751     }
00752     if(line.trimmed() !=QString("</segmentsData>")) return false;
00753     return true;
00754 }
00755
00756
00757 bool SegmentsTab::writeSegDataFile(QTextStream& outStream) {
00758
00759     QList<SegmentsDataData> lines = segmentsDataModel->getLines();
00760
00761     for(int i = 0; i<lines.size(); i++) {
00762         outStream « qSetFieldWidth(15) « lines.at(i).isActive
00763             « qSetFieldWidth(15) « lines.at(i).entrancePairIndex
00764             « qSetFieldWidth(15) « lines.at(i).exitPairIndex
00765             « qSetFieldWidth(15) « lines.at(i).lowEnergy
00766             « qSetFieldWidth(15) « lines.at(i).highEnergy
00767             « qSetFieldWidth(15) « lines.at(i).lowAngle
00768             « qSetFieldWidth(15) « lines.at(i).highAngle
00769             « qSetFieldWidth(15) « lines.at(i).dataType;
00770         if(lines.at(i).dataType == 2) outStream « qSetFieldWidth(15) « lines.at(i).phaseJ
00771             « qSetFieldWidth(15) « lines.at(i).phaseL;
00772         outStream « qSetFieldWidth(15) « lines.at(i).dataNorm
00773             « qSetFieldWidth(15) « lines.at(i).varyNorm
00774             « qSetFieldWidth(15) « lines.at(i).dataNormError
00775             « qSetFieldWidth(0) « lines.at(i).dataFile « endl;
00776     }
00777     return true;
00778 }
00779
00780
00781 bool SegmentsTab::readSegTestFile(QTextStream& inStream) {
00782
00783     int isActive;
00784     int entrancePairIndex;
00785     int exitPairIndex;
00786     double lowEnergy;
00787     double highEnergy;
00788     double energyStep;
00789     double lowAngle;
00790     double highAngle;
00791     double angleStep;
00792     int dataType;

```

```

00793     double phaseJ;
00794     int phaseL;
00795     int maxAngDistOrder;
00796
00797     QString line("");
00798     while(!inStream.atEnd() && line.trimmed() != QString("</segmentsTest>")) {
00799         line = inStream.readLine();
00800         if(line.trimmed().isEmpty()) continue;
00801         if(!inStream.atEnd() && line.trimmed() != QString("</segmentsTest>")) {
00802             QTextStream in(&line);
00803             in » isActive » entrancePairIndex » exitPairIndex » lowEnergy » highEnergy » energyStep »
lowAngle » highAngle » angleStep
00804             » dataType;
00805             if(dataType==2) {
00806                 in » phaseJ » phaseL;
00807             } else {
00808                 phaseJ=0.;
00809                 phaseL = 0;
00810             }
00811             if(dataType==3) {
00812                 in » maxAngDistOrder;
00813             } else {
00814                 maxAngDistOrder=0;
00815             }
00816             if(in.status() != QTextStream::Ok) return false;
00817             SegmentsTestData newLine =
{isActive,entrancePairIndex,exitPairIndex,lowEnergy,highEnergy,energyStep,lowAngle,
highAngle,angleStep,dataType,phaseJ,phaseL,maxAngDistOrder};
00818             addSegTestLine(newLine);
00819         }
00820     }
00821     if(line.trimmed() != QString("</segmentsTest>")) return false;
00822     return true;
00823 }
00824 }
00825
00826 bool SegmentsTab::writeSegTestFile(QTextStream& outStream) {
00827
00828     QList<SegmentsTestData> lines = segmentsTestModel->getLines();
00829
00830     for(int i = 0; i<lines.size(); i++) {
00831         outStream « qSetFieldWidth(15) « lines.at(i).isActive
00832             « qSetFieldWidth(15) « lines.at(i).entrancePairIndex
00833             « qSetFieldWidth(15) « lines.at(i).exitPairIndex
00834             « qSetFieldWidth(15) « lines.at(i).lowEnergy
00835             « qSetFieldWidth(15) « lines.at(i).highEnergy
00836             « qSetFieldWidth(15) « lines.at(i).energyStep
00837             « qSetFieldWidth(15) « lines.at(i).lowAngle
00838             « qSetFieldWidth(15) « lines.at(i).highAngle
00839             « qSetFieldWidth(15) « lines.at(i).angleStep;
00840         if(lines.at(i).dataType==2) {
00841             outStream « qSetFieldWidth(15) « lines.at(i).dataType
00842             « qSetFieldWidth(15) « lines.at(i).phaseJ
00843             « qSetFieldWidth(0) « lines.at(i).phaseL
00844             « endl;
00845         } else if(lines.at(i).dataType==3) {
00846             outStream « qSetFieldWidth(15) « lines.at(i).dataType
00847             « qSetFieldWidth(0) « lines.at(i).maxAngDistOrder
00848             « endl;
00849         } else outStream « qSetFieldWidth(0) « lines.at(i).dataType
00850             « endl;
00851     }
00852
00853     return true;
00854 }
00855
00856 void SegmentsTab::reset() {
00857     segmentsDataModel->removeRows(0,segmentsDataModel->getLines().size(),QModelIndex());
00858     segmentsTestModel->removeRows(0,segmentsTestModel->getLines().size(),QModelIndex());
00859 }
00860
00861 void SegmentsTab::showInfo(int which,QString title) {
00862     if(which<infoText.size()) {
00863         if(!infoDialog[which]) {
00864             infoDialog[which] = new InfoDialog(infoText[which],this,title);
00865             infoDialog[which]->setAttribute(Qt::WA_DeleteOnClose);
00866             infoDialog[which]->show();
00867         } else infoDialog[which]->raise();
00868     }
00869 }

```

8.129 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTestModel.cpp File Reference

```
#include "SegmentsTestModel.h"
#include "PairsModel.h"
```

8.130 SegmentsTestModel.cpp

[Go to the documentation of this file.](#)

```
00001 #include "SegmentsTestModel.h"
00002 #include "PairsModel.h"
00003
00004 SegmentsTestModel::SegmentsTestModel(QObject *parent) : QAbstractTableModel(parent) {
00005 }
00006
00007 int SegmentsTestModel::rowCount(const QModelIndex &parent) const {
00008     Q_UNUSED(parent);
00009     return segTestLineList.size();
00010 }
00011
00012 int SegmentsTestModel::columnCount(const QModelIndex &parent) const {
00013     Q_UNUSED(parent);
00014     return SegmentsTestData::SIZE;
00015 }
00016
00017 QVariant SegmentsTestModel::data(const QModelIndex &index, int role) const {
00018     if(!index.isValid()) return QVariant();
00019
00020     if(index.row() >= segTestLineList.size() || index.row() < 0) return QVariant();
00021
00022     if (role == Qt::DisplayRole) {
00023         SegmentsTestData line = segTestLineList.at(index.row());
00024         if(index.column() == 1) {
00025             if(line.dataType==4) {
00026                 int i = 0;
00027                 QList<PairsData> pairsList = pairsModel->getPairs();
00028                 for(i=0;i<pairsList.size();i++)
00029                     if(pairsList[i].pairType==10) break;
00030                 if(pairsList.size()>=line.entrancePairIndex&&i<pairsList.size()) {
00031                     PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
00032                     return QString("<center>%1</center>").arg(pairsModel->getReactionLabelTotalCapture(firstPair));
00033                 } else return QString("<center><font
style='color:red;font-weight:bold;'>UNDEFINED</font></center>");
00034             } else {
00035                 if(pairsModel->getPairs().size()>=line.entrancePairIndex&&pairsModel->getPairs().size()>=line.exitPairIndex)
00036                 {
00037                     PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
00038                     PairsData secondPair=pairsModel->getPairs().at(line.exitPairIndex-1);
00039                     return QString("<center>%1</center>").arg(pairsModel->getReactionLabel(firstPair,secondPair));
00040                 } else return QString("<center><font
style='color:red;font-weight:bold;'>UNDEFINED</font></center>");
00041             }
00042         } else if(index.column() == 2) return QVariant();
00043         else if(index.column() == 3) {
00044             if(line.lowEnergy==line.highEnergy) return line.lowEnergy;
00045             else return QString("%1-%2").arg(line.lowEnergy).arg(line.highEnergy);
00046         } else if(index.column() == 4) return QVariant();
00047         else if(index.column() == 5) return line.energyStep;
00048         else if(index.column() == 6) {
00049             if(line.lowAngle==line.highAngle) return line.lowAngle;
00050             else return QString("%1-%2").arg(line.lowAngle).arg(line.highAngle);
00051         } else if(index.column() == 7) return QVariant();
00052         else if(index.column() == 8) return line.angleStep;
00053         else if(index.column() == 9) {
00054             if(line.dataType==3) return QString(tr("<center>Angular Distribution</center>"));
00055             else if(line.dataType==2) {
00056                 QChar orbital;
00057                 switch (line.phaseL) {
00058                     case 0:
00059                         orbital='s';
00060                         break;
00061                     case 1:
00062                         orbital='p';
00063                         break;
00064                     case 2:
00065                         break;
```

```

00064         orbital='d';
00065         break;
00066     case 3:
00067         orbital='f';
00068         break;
00069     case 4:
00070         orbital='g';
00071         break;
00072     case 5:
00073         orbital='h';
00074         break;
00075     case 6:
00076         orbital='i';
00077         break;
00078     default:
00079         orbital='?';
00080     }
00081     QString tempSpin;
00082     if(((int)(line.phaseJ*2))%2!=0&&line.phaseJ!=0.)
tempSpin=QString("%1/2").arg((int)(line.phaseJ*2));
00083     else tempSpin=QString("%1").arg(line.phaseJ);
00084     return QString("<center>Phase Shift [%1<sub>%2</sub>]</center>").arg(orbital).arg(tempSpin);
00085     } else if(line.dataType==1) return QString(tr("<center>Differential</center>"));
00086     else if(line.dataType==4) return QString(tr("<center>Total Capture</center>"));
00087     else return QString(tr("<center>Angle Integrated</center>"));
00088     }
00089     else if(index.column() == 10) return QVariant();
00090     else if(index.column() == 11) return QVariant();
00091     else if(index.column() == 12) return QVariant();
00092     } else if (role == Qt::EditRole) {
00093         SegmentsTestData line = segTestLineList.at(index.row());
00094         if(index.column() == 1) return line.entrancePairIndex;
00095         else if(index.column() == 2) return line.exitPairIndex;
00096         else if(index.column() == 3) return line.lowEnergy;
00097         else if(index.column() == 4) return line.highEnergy;
00098         else if(index.column() == 5) return line.energyStep;
00099         else if(index.column() == 6) return line.lowAngle;
00100         else if(index.column() == 7) return line.highAngle;
00101         else if(index.column() == 8) return line.angleStep;
00102         else if(index.column() == 9) return line.dataType;
00103         else if(index.column() == 10) return line.phaseJ;
00104         else if(index.column() == 11) return line.phaseL;
00105         else if(index.column() == 12) return line.maxAngDistOrder;
00106     } else if (role==Qt::CheckStateRole && index.column()==0) {
00107         SegmentsTestData line = segTestLineList.at(index.row());
00108         if(line.isActive==1) return Qt::Checked;
00109         else return Qt::Unchecked;
00110     } else if(role == Qt::TextAlignmentRole) return Qt::AlignCenter;
00111
00112     return QVariant();
00113 }
00114
00115 QVariant SegmentsTestModel::headerData(int section, Qt::Orientation orientation, int role) const {
00116     if(role!= Qt::DisplayRole) return QVariant();
00117     if(orientation == Qt::Horizontal) {
00118         switch(section) {
00119             case 0:
00120                 return tr("");
00121             case 1:
00122                 return tr("Reaction");
00123             case 2:
00124                 return QVariant();
00125             case 3:
00126                 return tr("Energy\nRange");
00127             case 4:
00128                 return QVariant();
00129             case 5:
00130                 return tr("Energy\nStep");
00131             case 6:
00132                 return tr("Angle\nRange");
00133             case 7:
00134                 return QVariant();
00135             case 8:
00136                 return tr("Angle\nStep");
00137             case 9:
00138                 return tr("Data Type");
00139             case 10:
00140                 return QVariant();
00141             case 11:
00142                 return QVariant();
00143             case 12:
00144                 return QVariant();
00145             default:
00146                 return QVariant();
00147         }
00148     } else if(orientation == Qt::Vertical) {
00149         return section+1;

```

```

00150     }
00151     return QVariant();
00152 }
00153
00154 bool SegmentsTestModel::setData(const QModelIndex &index, const QVariant &value, int role) {
00155     if (index.isValid() && role == Qt::EditRole) {
00156         int row = index.row();
00157         SegmentsTestData tempData = segTestLineList.value(row);
00158         if (index.column() == 0) tempData.isActive=value.toInt();
00159         else if (index.column() == 1) tempData.entrancePairIndex=value.toInt();
00160         else if (index.column() == 2) tempData.exitPairIndex=value.toInt();
00161         else if (index.column() == 3) tempData.lowEnergy=value.toDouble();
00162         else if (index.column() == 4) tempData.highEnergy=value.toDouble();
00163         else if (index.column() == 5) tempData.energyStep=value.toDouble();
00164         else if (index.column() == 6) tempData.lowAngle=value.toDouble();
00165         else if (index.column() == 7) tempData.highAngle=value.toDouble();
00166         else if (index.column() == 8) tempData.angleStep=value.toDouble();
00167         else if (index.column() == 9) tempData.dataType=value.toInt();
00168         else if (index.column() == 10) tempData.phaseJ=value.toDouble();
00169         else if (index.column() == 11) tempData.phaseL=value.toInt();
00170         else if (index.column() == 12) tempData.maxAngDistOrder=value.toInt();
00171         else return false;
00172
00173         segTestLineList.replace(row,tempData);
00174         emit(dataChanged(index,index));
00175         return true;
00176     } else if (role== Qt::CheckStateRole) {
00177         int row = index.row();
00178         SegmentsTestData tempData = segTestLineList.value(row);
00179         if (index.column()==0) {
00180             if (value==Qt::Checked) tempData.isActive=1;
00181             else tempData.isActive=0;
00182         } else return false;
00183         segTestLineList.replace(row,tempData);
00184         emit(dataChanged(index,index));
00185         return true;
00186     }
00187     return false;
00188 }
00189
00190 bool SegmentsTestModel::insertRows(int position, int rows, const QModelIndex &index) {
00191     Q_UNUSED(index);
00192     if (rows>0) {
00193         beginInsertRows(QModelIndex(),position,position+rows-1);
00194         for (int row=0; row<rows; row++) {
00195             SegmentsTestData tempData;
00196             segTestLineList.insert(position,tempData);
00197         }
00198         endInsertRows();
00199     }
00200     return true;
00201 }
00202
00203 bool SegmentsTestModel::removeRows(int position, int rows, const QModelIndex &index) {
00204     Q_UNUSED(index);
00205     if (rows>0) {
00206         beginRemoveRows(QModelIndex(),position,position+rows-1);
00207         for (int row=0; row<rows; ++row) {
00208             segTestLineList.removeAt(position);
00209         }
00210         endRemoveRows();
00211     }
00212     return true;
00213 }
00214
00215 Qt::ItemFlags SegmentsTestModel::flags(const QModelIndex &index) const {
00216     if (!index.isValid()) return Qt::ItemIsEnabled;
00217     if (index.column()==0) return QAbstractTableModel::flags(index) | Qt::ItemIsUserCheckable;
00218     return QAbstractTableModel::flags(index);
00219 }
00220
00221 int SegmentsTestModel::isSegTestLine(const SegmentsTestData &line) const {
00222     int foundLine=-1;
00223     for (int i=0; i<segTestLineList.size(); i++) {
00224         SegmentsTestData tempLine=segTestLineList.value(i);
00225         if (tempLine.entrancePairIndex==line.entrancePairIndex&&
00226             tempLine.exitPairIndex==line.exitPairIndex&&
00227             tempLine.lowEnergy==line.lowEnergy&&
00228             tempLine.highEnergy==line.highEnergy&&
00229             tempLine.energyStep==line.energyStep&&
00230             tempLine.lowAngle==line.lowAngle&&
00231             tempLine.highAngle==line.highAngle&&
00232             tempLine.angleStep==line.angleStep&&
00233             tempLine.dataType==line.dataType&&
00234             tempLine.phaseJ==line.phaseJ&&
00235             tempLine.phaseL==line.phaseL&&
00236             tempLine.maxAngDistOrder==line.maxAngDistOrder) {

```

```

00237         foundLine=i;
00238         break;
00239     }
00240 }
00241 return foundLine;
00242 }
00243
00244 void SegmentsTestModel::setPairsModel(PairsModel* model) {
00245     pairsModel=model;
00246 }
00247
00248 QString SegmentsTestModel::getReactionLabel(const QModelIndex &index) {
00249     SegmentsTestData line = segTestLineList.at(index.row());
00250     if(line.dataType==4) {
00251         int i = 0;
00252         QList<PairsData> pairsList = pairsModel->getPairs();
00253         for(i=0;i<pairsList.size();i++)
00254             if(pairsList[i].pairType==10) break;
00255         if(pairsList.size()>=line.entrancePairIndex&&i<pairsList.size()) {
00256             PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
00257             return pairsModel->getReactionLabelTotalCapture(firstPair);
00258         }
00259         return QString("<font style=' color:red;font-weight:bold;'>UNDEFINED</font>");
00260     } else {
00261         int numPairs = pairsModel->getPairs().size();
00262         if(line.entrancePairIndex-1>=numPairs ||
00263            line.exitPairIndex-1>=numPairs) return QString("<font
style=' color:red;font-weight:bold;'>UNDEFINED</font>");
00264         PairsData firstPair=pairsModel->getPairs().at(line.entrancePairIndex-1);
00265         PairsData secondPair=pairsModel->getPairs().at(line.exitPairIndex-1);
00266         return pairsModel->getReactionLabel(firstPair,secondPair);
00267     }
00268 }

```

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.](#)

```

00001 #include <QVariant>
00002
00003 #include "TargetIntModel.h"
00004
00005 TargetIntModel::TargetIntModel(QObject *parent) : QAbstractTableModel(parent) {
00006 }
00007
00008 int TargetIntModel::rowCount(const QModelIndex &parent) const {
00009     Q_UNUSED(parent);
00010     return targetIntList.size();
00011 }
00012
00013 int TargetIntModel::columnCount(const QModelIndex &parent) const {
00014     Q_UNUSED(parent);
00015     return TargetIntData::SIZE;
00016 }
00017
00018 QVariant TargetIntModel::data(const QModelIndex &index, int role) const {
00019     if(!index.isValid()) return QVariant();
00020     if(index.row()>=targetIntList.size() || index.row() < 0) return QVariant();
00021     if(role == Qt::DisplayRole) {
00022         TargetIntData targetInt=targetIntList.at(index.row());
00023         if(index.column() == 1) return targetInt.segmentsList;
00024         else if(index.column() == 2) {
00025             if(targetInt.isTargetIntegration||targetInt.isConvolution) return targetInt.numPoints;
00026             else return QString(tr("N/A"));
00027         }
00028         else if(index.column() == 3) {
00029             if(targetInt.isConvolution) return QString(tr("YES"));

```



```

00030         else return QString(tr("NO"));
00031     } else if (index.column() == 4) return targetInt.sigma;
00032     else if (index.column() == 5) {
00033         if (targetInt.isTargetIntegration) return QString(tr("YES"));
00034         else return QString(tr("NO"));
00035     } else if (index.column() == 6) return targetInt.density;
00036     else if (index.column() == 7) return targetInt.stoppingPowerEq;
00037     else if (index.column() == 8) return targetInt.numParameters;
00038     else if (index.column() == 9) return QVariant();
00039     else if (index.column() == 10) {
00040         if (targetInt.isQCoefficients) return QString(tr("YES"));
00041         else return QString(tr("NO"));
00042     } else if (index.column() == 11) return QVariant();
00043 } else if (role == Qt::EditRole) {
00044     TargetIntData targetInt=targetIntList.at(index.row());
00045     if (index.column()==1) return targetInt.segmentsList;
00046     if (index.column()==2) return targetInt.numPoints;
00047     if (index.column()==3) return targetInt.isConvolution;
00048     if (index.column()==4) return targetInt.sigma;
00049     if (index.column()==5) return targetInt.isTargetIntegration;
00050     if (index.column()==6) return targetInt.density;
00051     if (index.column()==7) return targetInt.stoppingPowerEq;
00052     if (index.column()==8) return targetInt.numParameters;
00053     if (index.column()==9) return QVariant::fromValue<QList<double> >(targetInt.parameters);
00054     if (index.column()==10) return targetInt.isQCoefficients;
00055     if (index.column()==11) return QVariant::fromValue<QList<double> >(targetInt.qCoefficients);
00056 } else if (role==Qt::CheckStateRole && index.column()==0) {
00057     TargetIntData targetInt=targetIntList.at(index.row());
00058     if (targetInt.isActive==1) return Qt::Checked;
00059     else return Qt::Unchecked;
00060 } else if (role == Qt::TextAlignmentRole) return Qt::AlignCenter;
00061 return QVariant();
00062 }
00063
00064 QVariant TargetIntModel::headerData(int section, Qt::Orientation orientation, int role) const {
00065     if (role!=Qt::DisplayRole) return QVariant();
00066     if (orientation == Qt::Horizontal) {
00067         switch(section) {
00068             case 0:
00069                 return tr("");
00070             case 1:
00071                 return tr("Segment List");
00072             case 2:
00073                 return tr("Number of Integration Points");
00074             case 3:
00075                 return tr("Convolution Active?");
00076             case 4:
00077                 return tr("Gaussian Sigma");
00078             case 5:
00079                 return tr("Target Integration Active?");
00080             case 6:
00081                 return tr("Target Density");
00082             case 7:
00083                 return tr("Stopping Power Equation");
00084             case 8:
00085                 return tr("Number of Parameters");
00086             case 9:
00087                 return tr("Parameters List");
00088             case 10:
00089                 return tr("Use Q-Coefficients?");
00090             case 11:
00091                 return tr("Q-Coefficient List");
00092             default:
00093                 return QVariant();
00094         }
00095     } else if (orientation == Qt::Vertical) return section+1;
00096     return QVariant();
00097 }
00098
00099 bool TargetIntModel::setData(const QModelIndex &index, const QVariant &value, int role) {
00100     if (index.isValid() && role == Qt::EditRole) {
00101         int row = index.row();
00102         TargetIntData tempData = targetIntList.value(row);
00103         if (index.column() == 0) tempData.isActive = value.toInt();
00104         else if (index.column() == 1) tempData.segmentsList = value.toString();
00105         else if (index.column() == 2) tempData.numPoints = value.toInt();
00106         else if (index.column() == 3) tempData.isConvolution = value.toBool();
00107         else if (index.column() == 4) tempData.sigma = value.toDouble();
00108         else if (index.column() == 5) tempData.isTargetIntegration = value.toBool();
00109         else if (index.column() == 6) tempData.density = value.toDouble();
00110         else if (index.column() == 7) tempData.stoppingPowerEq = value.toString();
00111         else if (index.column() == 8) tempData.numParameters = value.toInt();
00112         else if (index.column() == 9) tempData.parameters = value.value<QList<double> >();
00113         else if (index.column() == 10) tempData.isQCoefficients = value.toBool();
00114         else if (index.column() == 11) tempData.qCoefficients = value.value<QList<double> >();
00115         else return false;
00116         targetIntList.replace(row, tempData);

```

```

00117     emit(dataChanged(index, index));
00118     return true;
00119 } else if(role == Qt::CheckStateRole) {
00120     int row = index.row();
00121     TargetIntData tempData = targetIntList.value(row);
00122     if(index.column() == 0) {
00123         if(value==Qt::Checked) tempData.isActive=1;
00124         else tempData.isActive=0;
00125     } else return false;
00126     targetIntList.replace(row, tempData);
00127     emit(dataChanged(index, index));
00128     return true;
00129 }
00130 return false;
00131 }
00132
00133 bool TargetIntModel::insertRows(int position, int rows, const QModelIndex &index) {
00134     Q_UNUSED(index);
00135     if(rows>0) {
00136         beginInsertRows(QModelIndex(), position, position+rows-1);
00137         for(int row=0; row<rows; row++) {
00138             TargetIntData tempData;
00139             targetIntList.insert(position, tempData);
00140         }
00141         endInsertRows();
00142     }
00143     return true;
00144 }
00145
00146 bool TargetIntModel::removeRows(int position, int rows, const QModelIndex &index) {
00147     Q_UNUSED(index);
00148     if(rows>0) {
00149         beginRemoveRows(QModelIndex(), position, position+rows-1);
00150         for(int row=0; row<rows; ++row) {
00151             targetIntList.removeAt(position);
00152         }
00153         endRemoveRows();
00154     }
00155     return true;
00156 }
00157
00158 Qt::ItemFlags TargetIntModel::flags(const QModelIndex &index) const {
00159     if (!index.isValid()) return Qt::ItemIsEnabled;
00160     if(index.column()==0) return QAbstractTableModel::flags(index) | Qt::ItemIsUserCheckable;
00161     return QAbstractTableModel::flags(index);
00162 }

```

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.](#)

```

00001 #include <QGridLayout>
00002 #include <QCheckBox>
00003 #include <QSpacerItem>
00004 #include <QLineEdit>

```

```

00005 #include <QSpinBox>
00006 #include <QPushButton>
00007 #include <QTextStream>
00008 #include <QHeaderView>
00009
00010 #include "TargetIntTab.h"
00011 #include "InfoDialog.h"
00012
00013
00014 TargetIntTab::TargetIntTab(QWidget *parent) : QWidget(parent) {
00015     targetIntModel = new TargetIntModel(this);
00016     targetIntView = new QTableView;
00017     targetIntView->setModel(targetIntModel);
00018     targetIntView->verticalHeader()->setHighlightSections(false);
00019     targetIntView->horizontalHeader()->setHighlightSections(false);
00020     targetIntView->setColumnHidden(4,true);
00021     targetIntView->setColumnHidden(6,true);
00022     targetIntView->setColumnHidden(7,true);
00023     targetIntView->setColumnHidden(8,true);
00024     targetIntView->setColumnHidden(9,true);
00025     targetIntView->setColumnHidden(11,true);
00026     targetIntView->setColumnWidth(0,27);
00027     targetIntView->horizontalHeader()->setSectionResizeMode(0,QHeaderView::Fixed);
00028     targetIntView->horizontalHeader()->setSectionResizeMode(1,QHeaderView::Stretch);
00029     targetIntView->horizontalHeader()->setSectionResizeMode(2,QHeaderView::Stretch);
00030     targetIntView->horizontalHeader()->setSectionResizeMode(3,QHeaderView::Stretch);
00031     targetIntView->horizontalHeader()->setSectionResizeMode(5,QHeaderView::Stretch);
00032     targetIntView->horizontalHeader()->setSectionResizeMode(10,QHeaderView::Stretch);
00033     targetIntView->setSelectionBehavior(QAbstractItemView::SelectRows);
00034     targetIntView->setSelectionMode(QAbstractItemView::SingleSelection);
00035     targetIntView->setEditTriggers(QAbstractItemView::NoEditTriggers);
00036     targetIntView->setShowGrid(false);
00037
00038     connect(targetIntView->selectionModel(), SIGNAL(selectionChanged(QItemSelection,QItemSelection)), this, SLOT(updateButtons));
00039     connect(targetIntView, SIGNAL(doubleClicked(QModelIndex)), this, SLOT(editLine()));
00040
00041     addButton = new QPushButton(tr("+"));
00042     addButton->setMaximumSize(28,28);
00043     connect(addButton, SIGNAL(clicked()), this, SLOT(addLine()));
00044     deleteButton = new QPushButton(tr("-"));
00045     deleteButton->setMaximumSize(28,28);
00046     deleteButton->setEnabled(false);
00047     connect(deleteButton, SIGNAL(clicked()), this, SLOT(deleteLine()));
00048
00049     QGridLayout *buttonBox = new QGridLayout;
00050     buttonBox->addWidget(addButton,0,0);
00051     buttonBox->addWidget(deleteButton,0,1);
00052     buttonBox->addItem(new QSpacerItem(28,28),0,2);
00053     buttonBox->setColumnStretch(0,0);
00054     buttonBox->setColumnStretch(1,0);
00055     buttonBox->setColumnStretch(2,1);
00056 #ifdef MACX_SPACING
00057     buttonBox->setHorizontalSpacing(11);
00058 #else
00059     buttonBox->setHorizontalSpacing(0);
00060 #endif
00061
00062     QGridLayout *mainLayout = new QGridLayout;
00063     mainLayout->addWidget(targetIntView,0,0);
00064     mainLayout->addLayout(buttonBox,1,0);
00065     mainLayout->setRowStretch(0,1);
00066     mainLayout->setRowStretch(1,0);
00067
00068     setLayout(mainLayout);
00069 }
00070
00071 TargetIntModel* TargetIntTab::getTargetIntModel() {
00072     return targetIntModel;
00073 }
00074
00075 void TargetIntTab::addLine() {
00076     AddTargetIntDialog aDialog;
00077     if(aDialog.exec()) {
00078         TargetIntData newLine;
00079         newLine.isActive=1;
00080         newLine.segmentsList=aDialog.segmentsListText->text();
00081         newLine.numPoints=aDialog.numPointsSpin->value();
00082         if(aDialog.isConvolutionCheck->isChecked())
00083             newLine.isConvolution=true;
00084         else newLine.isConvolution=false;
00085         newLine.sigma=aDialog.sigmaText->text().toDouble();
00086         if(aDialog.isTargetIntegrationCheck->isChecked())
00087             newLine.isTargetIntegration=true;
00088         else newLine.isTargetIntegration=false;
00089         newLine.density=aDialog.densityText->text().toDouble();
00090         newLine.stoppingPowerEq=aDialog.stoppingPowerEqText->text();
00091         newLine.numParameters=aDialog.numParametersSpin->value();
00092     }
00093 }

```

```

00091     for(int i=0;i<newLine.numParameters;i++)
00092         newLine.parameters.append(aDialog.tempParameters.at(i));
00093     newLine.isQCoefficients = (aDialog.isQCoefficientCheck->isChecked()) ? (true) : (false);
00094     for(int i=0;i<aDialog.numQCoefficientSpin->value();i++)
00095         newLine.qCoefficients.append(aDialog.tempQCoefficients.at(i));
00096     addLine(newLine);
00097 }
00098 }
00099
00100 void TargetIntTab::addLine(TargetIntData line) {
00101     QList<TargetIntData> lines = targetIntModel->getLines();
00102     targetIntModel->insertRows(lines.size(),1,QModelIndex());
00103     QModelIndex index = targetIntModel->index(lines.size(),0,QModelIndex());
00104     targetIntModel->setData(index,line.isActive,Qt::EditRole);
00105     index = targetIntModel->index(lines.size(),1,QModelIndex());
00106     targetIntModel->setData(index,line.segmentsList,Qt::EditRole);
00107     index = targetIntModel->index(lines.size(),2,QModelIndex());
00108     targetIntModel->setData(index,line.numPoints,Qt::EditRole);
00109     index = targetIntModel->index(lines.size(),3,QModelIndex());
00110     targetIntModel->setData(index,line.isConvolution,Qt::EditRole);
00111     index = targetIntModel->index(lines.size(),4,QModelIndex());
00112     targetIntModel->setData(index,line.sigma,Qt::EditRole);
00113     index = targetIntModel->index(lines.size(),5,QModelIndex());
00114     targetIntModel->setData(index,line.isTargetIntegration,Qt::EditRole);
00115     index = targetIntModel->index(lines.size(),6,QModelIndex());
00116     targetIntModel->setData(index,line.density,Qt::EditRole);
00117     index = targetIntModel->index(lines.size(),7,QModelIndex());
00118     targetIntModel->setData(index,line.stoppingPowerEq,Qt::EditRole);
00119     index = targetIntModel->index(lines.size(),8,QModelIndex());
00120     targetIntModel->setData(index,line.numParameters,Qt::EditRole);
00121     index = targetIntModel->index(lines.size(),9,QModelIndex());
00122     targetIntModel->setData(index,QVariant::fromValue<QList<double>>(line.parameters),Qt::EditRole);
00123     index = targetIntModel->index(lines.size(),10,QModelIndex());
00124     targetIntModel->setData(index,line.isQCoefficients,Qt::EditRole);
00125     index = targetIntModel->index(lines.size(),11,QModelIndex());
00126     targetIntModel->setData(index,QVariant::fromValue<QList<double>>(line.qCoefficients),Qt::EditRole);
00127
00128     targetIntView->resizeRowsToContents();
00129 }
00130
00131 void TargetIntTab::editLine() {
00132     QItemSelectionModel *selectionModel = targetIntView->selectionModel();
00133     QModelIndexList indexes = selectionModel->selectedRows();
00134     QModelIndex index = indexes[0];
00135
00136     QModelIndex i=targetIntModel->index(index.row(),1,QModelIndex());
00137     QVariant var=targetIntModel->data(i,Qt::EditRole);
00138     QString segmentsList=var.toString();
00139     i=targetIntModel->index(index.row(),2,QModelIndex());
00140     var=targetIntModel->data(i,Qt::EditRole);
00141     int numPoints=var.toInt();
00142     i=targetIntModel->index(index.row(),3,QModelIndex());
00143     var=targetIntModel->data(i,Qt::EditRole);
00144     bool isConvolution = var.toBool();
00145     i=targetIntModel->index(index.row(),4,QModelIndex());
00146     var=targetIntModel->data(i,Qt::EditRole);
00147     QString sigma = var.toString();
00148     i=targetIntModel->index(index.row(),5,QModelIndex());
00149     var=targetIntModel->data(i,Qt::EditRole);
00150     bool isTargetIntegration = var.toBool();
00151     i=targetIntModel->index(index.row(),6,QModelIndex());
00152     var=targetIntModel->data(i,Qt::EditRole);
00153     QString density = var.toString();
00154     i=targetIntModel->index(index.row(),7,QModelIndex());
00155     var=targetIntModel->data(i,Qt::EditRole);
00156     QString stoppingPowerEq = var.toString();
00157     i=targetIntModel->index(index.row(),8,QModelIndex());
00158     var=targetIntModel->data(i,Qt::EditRole);
00159     int numParameters = var.toInt();
00160     i=targetIntModel->index(index.row(),9,QModelIndex());
00161     var=targetIntModel->data(i,Qt::EditRole);
00162     QList<double> parameters = var.value<QList<double>>();
00163     i=targetIntModel->index(index.row(),10,QModelIndex());
00164     var=targetIntModel->data(i,Qt::EditRole);
00165     bool isQCoefficient = var.toBool();
00166     i=targetIntModel->index(index.row(),11,QModelIndex());
00167     var=targetIntModel->data(i,Qt::EditRole);
00168     QList<double> qCoefficients = var.value<QList<double>>();
00169
00170     AddTargetIntDialog aDialog;
00171     aDialog.setWindowTitle(tr("Edit an Experimental Effect Line"));
00172     aDialog.segmentsListText->setText(segmentsList);
00173     aDialog.numPointsSpin->setValue(numPoints);
00174     if(isConvolution)
00175         aDialog.isConvolutionCheck->setChecked(true);
00176     else aDialog.isConvolutionCheck->setChecked(false);
00177     aDialog.sigmaText->setText(sigma);

```

```

00178     if (isTargetIntegration) aDialog.isTargetIntegrationCheck->setChecked(true);
00179     else aDialog.isTargetIntegrationCheck->setChecked(false);
00180     aDialog.densityText->setText(density);
00181     aDialog.stoppingPowerEqText->setText(stoppingPowerEq);
00182     aDialog.tempParameters=parameters;
00183     aDialog.numParametersSpin->setValue(numParameters);
00184     if (isQCoefficient) aDialog.isQCoefficientCheck->setChecked(true);
00185     else aDialog.isQCoefficientCheck->setChecked(false);
00186     aDialog.tempQCoefficients=qCoefficients;
00187     aDialog.numQCoefficientSpin->setValue(qCoefficients.size());
00188
00189     if (aDialog.exec()) {
00190         QString newSegmentsList = aDialog.segmentsListText->text();
00191         if (segmentsList!=newSegmentsList) {
00192             i=targetIntModel->index(index.row(),1,QModelIndex());
00193             targetIntModel->setData(i,newSegmentsList,Qt::EditRole);
00194         }
00195         int newNumPoints = aDialog.numPointsSpin->value();
00196         if (numPoints!=newNumPoints) {
00197             i=targetIntModel->index(index.row(),2,QModelIndex());
00198             targetIntModel->setData(i,newNumPoints,Qt::EditRole);
00199         }
00200         bool newIsConvolution=false;
00201         if (aDialog.isConvolutionCheck->isChecked()) newIsConvolution=true;
00202         if (isConvolution!=newIsConvolution) {
00203             i=targetIntModel->index(index.row(),3,QModelIndex());
00204             targetIntModel->setData(i,newIsConvolution,Qt::EditRole);
00205         }
00206         QString newSigma = aDialog.sigmaText->text();
00207         if (sigma!=newSigma) {
00208             i=targetIntModel->index(index.row(),4,QModelIndex());
00209             targetIntModel->setData(i,newSigma,Qt::EditRole);
00210         }
00211         bool newIsTargetIntegration=false;
00212         if (aDialog.isTargetIntegrationCheck->isChecked()) newIsTargetIntegration=true;
00213         if (isTargetIntegration!=newIsTargetIntegration) {
00214             i=targetIntModel->index(index.row(),5,QModelIndex());
00215             targetIntModel->setData(i,newIsTargetIntegration,Qt::EditRole);
00216         }
00217         QString newDensity = aDialog.densityText->text();
00218         if (density!=newDensity) {
00219             i=targetIntModel->index(index.row(),6,QModelIndex());
00220             targetIntModel->setData(i,newDensity,Qt::EditRole);
00221         }
00222         QString newStoppingPowerEq=aDialog.stoppingPowerEqText->text();
00223         if (stoppingPowerEq!=newStoppingPowerEq) {
00224             i=targetIntModel->index(index.row(),7,QModelIndex());
00225             targetIntModel->setData(i,newStoppingPowerEq,Qt::EditRole);
00226         }
00227         int newNumParameters=aDialog.numParametersSpin->value();
00228         if (numParameters!=newNumParameters) {
00229             i=targetIntModel->index(index.row(),8,QModelIndex());
00230             targetIntModel->setData(i,newNumParameters,Qt::EditRole);
00231         }
00232         QList<double> newParameters = aDialog.tempParameters;
00233         if (parameters!=newParameters||numParameters!=newNumParameters) {
00234             parameters.clear();
00235             for (int j=0;j<newNumParameters;j++)
00236                 parameters.append(newParameters.at(j));
00237             i=targetIntModel->index(index.row(),9,QModelIndex());
00238             targetIntModel->setData(i,QVariant::fromValue<QList<double>>(parameters),Qt::EditRole);
00239         }
00240         bool newIsQCoefficient=false;
00241         if (aDialog.isQCoefficientCheck->isChecked()) newIsQCoefficient=true;
00242         if (isQCoefficient!=newIsQCoefficient) {
00243             i=targetIntModel->index(index.row(),10,QModelIndex());
00244             targetIntModel->setData(i,newIsQCoefficient,Qt::EditRole);
00245         }
00246         QList<double> newQCoefficients = aDialog.tempQCoefficients;
00247         if (qCoefficients!=newQCoefficients||qCoefficients.size()!=aDialog.numQCoefficientSpin->value()) {
00248             qCoefficients.clear();
00249             for (int j=0;j<aDialog.numQCoefficientSpin->value();j++)
00250                 qCoefficients.append(newQCoefficients.at(j));
00251             i=targetIntModel->index(index.row(),11,QModelIndex());
00252             targetIntModel->setData(i,QVariant::fromValue<QList<double>>(qCoefficients),Qt::EditRole);
00253         }
00254     }
00255 }
00256
00257 void TargetIntTab::deleteLine() {
00258     QItemSelectionModel *selectionModel = targetIntView->selectionModel();
00259     QModelIndexList indexes = selectionModel->selectedRows();
00260     QModelIndex index=indexes.at(0);
00261
00262     targetIntModel->removeRows(index.row(),1,QModelIndex());
00263 }
00264

```

```

00265 void TargetIntTab::updateButtons(const QItemSelection &selection) {
00266     QModelIndexList indexes=selection.indexes();
00267
00268     if(indexes.isEmpty()) {
00269         deleteButton->setEnabled(false);
00270     } else {
00271         deleteButton->setEnabled(true);
00272     }
00273 }
00274
00275 bool TargetIntTab::writeFile(QTextStream& outStream) {
00276
00277     QList<TargetIntData> lines = targetIntModel->getLines();
00278
00279     for(int i=0;i<lines.size();i++) {
00280         outStream << qSetFieldWidth(15) << lines.at(i).isActive
00281             << qSetFieldWidth(15) << "\"+lines[i].segmentsList.remove(' ')+"\"
00282             << qSetFieldWidth(15) << lines.at(i).numPoints;
00283         if(lines.at(i).isConvolution) outStream << qSetFieldWidth(15) << '1';
00284         else outStream << qSetFieldWidth(15) << '0';
00285         outStream << qSetFieldWidth(15) << lines.at(i).sigma;
00286         if(lines.at(i).isTargetIntegration) outStream << qSetFieldWidth(15) << '1';
00287         else outStream << qSetFieldWidth(15) << '0';
00288         outStream << qSetFieldWidth(15) << lines.at(i).density
00289             << qSetFieldWidth(0) << " \"+lines[i].stoppingPowerEq.remove(' ')+"\"
00290             << qSetFieldWidth(0) << lines.at(i).numParameters << qSetFieldWidth(0) << ' ';
00291         for(int j=0;j<lines.at(i).numParameters;j++)
00292             outStream << lines.at(i).parameters.at(j) << qSetFieldWidth(0) << ' ';
00293         if(lines.at(i).isQCoefficients) outStream << qSetFieldWidth(0) << "          1";
00294         else outStream << qSetFieldWidth(0) << "          0";
00295         outStream << qSetFieldWidth(0) << "          " << lines.at(i).qCoefficients.size() << ' ';
00296         for(int j=0;j<lines.at(i).qCoefficients.size();j++)
00297             outStream << qSetFieldWidth(0) << lines.at(i).qCoefficients.at(j) << ' ';
00298         outStream<<endl;
00299     }
00300
00301     return true;
00302 }
00303
00304 bool TargetIntTab::readFile(QTextStream& inStream) {
00305
00306     int isActive;
00307     QString segmentsList;
00308     int numPoints;
00309     int isConvolution;
00310     double sigma;
00311     int isTargetIntegration;
00312     double density;
00313     QString stoppingPowerEq;
00314     int numParameters;
00315     QList<double> parameters;
00316     int numQCoefficients;
00317     QList<double> qCoefficients;
00318     int isQCoefficient;
00319
00320     QString line("");
00321     while(!inStream.atEnd() && line.trimmed() != QString("</targetInt>")) {
00322         line=inStream.readLine();
00323         if(line.trimmed().isEmpty()) continue;
00324         if(!inStream.atEnd() && line.trimmed() != QString("</targetInt>")) {
00325             parameters.clear();
00326             qCoefficients.clear();
00327             QTextStream in(&line);
00328             in >> isActive >> segmentsList >> numPoints >> isConvolution >> sigma >> isTargetIntegration
00329             >> density >> stoppingPowerEq >> numParameters;
00330             if(in.status() != QTextStream::Ok) return false;
00331             int i=0;
00332             while(i<numParameters) {
00333                 double tempParameter;
00334                 in >> tempParameter;
00335                 parameters.append(tempParameter);
00336                 i++;
00337             }
00338             if(in.status() != QTextStream::Ok) return false;
00339             in >> isQCoefficient;
00340             if(in.status() == QTextStream::Ok) {
00341                 in >> numQCoefficients;
00342                 i=0;
00343                 while(i<numQCoefficients) {
00344                     double tempQCoefficient;
00345                     in >> tempQCoefficient;
00346                     qCoefficients.append(tempQCoefficient);
00347                     i++;
00348                 }
00349                 if(in.status() != QTextStream::Ok) return false;
00350             } else {
00351                 isQCoefficient=0;

```

```

00352     }
00353     bool tempIsQCoefficient=false;
00354     if(isQCoefficient==1) tempIsQCoefficient=true;
00355     bool tempIsConvolution=false;
00356     if(isConvolution==1) tempIsConvolution=true;
00357     bool tempIsTargetIntegration=false;
00358     if(isTargetIntegration==1) tempIsTargetIntegration=true;
00359     TargetIntData newLine =
{isActive,segmentsList.remove('\\"),numPoints,tempIsConvolution,sigma,tempIsTargetIntegration,
00360 density,stoppingPowerEq.remove('\\"),numParameters,parameters,tempIsQCoefficient,qCoefficients};
00361     addLine(newLine);
00362     }
00363     }
00364     targetIntView->resizeRowsToContents();
00365     if(line.trimmed()!=QString("</targetInt>")) return false;
00366     return true;
00367 }
00368
00369 void TargetIntTab::reset() {
00370     targetIntModel->removeRows(0,targetIntModel->getLines().size(),QModelIndex());
00371 }
00372
00373 void TargetIntTab::showInfo(int which,QString title) {
00374     if(which<infoText.size()) {
00375         if(!infoDialog[which]) {
00376             infoDialog[which] = new InfoDialog(infoText[which],this,title);
00377             infoDialog[which]->setAttribute(Qt::WA_DeleteOnClose);
00378             infoDialog[which]->show();
00379         } else infoDialog[which]->raise();
00380     }
00381 }

```

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.](#)

```

00001 #ifndef ACHANNEL_H
00002 #define ACHANNEL_H
00003
00004 class NucLine;
00005
00006
00007
00012 class AChannel {
00013 public:
00014     AChannel(NucLine, int);
00015     AChannel(int, double, int, char);
00016     int GetPairNum() const;
00017     int GetL() const;
00018     double GetS() const;
00019     double GetBoundaryCondition() const;
00020     char GetRadType() const;
00021     void SetBoundaryCondition(double);
00022 private:
00023     int l_;
00024     int pair_;
00025     double s_;
00026     char radtype_;
00027     double boundary_condition_;
00028 };
00029
00030 #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.](#)

```
00001 #ifndef ALEVEL_H
00002 #define ALEVEL_H
00003
00004 #include "Constants.h"
00005
00006 class NucLine;
00007
00008
00009
00014 class ALevel {
00015 public:
00016     ALevel(NucLine);
00017     ALevel(double);
00018     bool IsInRMatrix() const;
00019     bool EnergyFixed() const;
00020     bool ChannelFixed(int) const;
00021     bool IsECLevel() const;
00022     int NumNFIntegrals() const;
00023     int GetTransformIterations() const;
00024     int GetECPairNum() const;
00025     unsigned char GetECMultMask() const;
00026     double GetE() const;
00027     double GetGamma(int) const;
00028     double GetFitGamma(int) const;
00029     double GetFitE() const;
00030     double GetNFIntegral(int) const;
00031     double GetSqrtNFFactor() const;
00032     double GetECConversionFactor(int) const;
00033     double GetTransformGamma(int) const;
00034     double GetTransformE() const;
00035     double GetBigGamma(int) const;
00036     double GetShiftFunction(int) const;
00037     complex GetExternalGamma(int) const;
00038     void AddGamma(NucLine);
00039     void AddGamma(double);
00040     void SetGamma(int, double);
00041     void SetE(double);
00042     void SetFitGamma(int, double);
00043     void SetFitE(double);
00044     void AddNFIntegral(double);
00045     void SetSqrtNFFactor(double);
00046     void AddECConversionFactor(double);
00047     void SetTransformGamma(int, double);
00048     void SetTransformE(double);
00049     void SetBigGamma(int, double);
00050     void SetTransformIterations(int);
00051     void SetExternalGamma(int, complex);
00052     void SetShiftFunction(int, double);
00053     void SetECParams(int, unsigned char);
00054 private:
00055     bool isinrmatrix_;
00056     bool energyfixed_;
00057     bool isECLevel_;
00058     int transform_iter_;
00059     int ecPairNum_;
00060     unsigned char ecMultMask_;
00061     double level_e_;
00062     double fitlevel_e_;
00063     double sqrt_nf_factor_;
```



```

00064     double transform_e_;
00065     std::vector<bool> channelfixed_;
00066     vector_r gammas_;
00067     vector_r fitgammas_;
00068     vector_r nf_integrals_;
00069     vector_r ec_conv_factors_;
00070     vector_r transform_gammas_;
00071     vector_r big_gammas_;
00072     vector_r shifts_;
00073     vector_c external_gammas_;
00074 };
00075
00076 #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.](#)

```

00001 #ifndef AMATRIXFUNC_H
00002 #define AMATRIXFUNC_H
00003
00004 #include "GenMatrixFunc.h"
00005
00007
00014 class AMatrixFunc : public GenMatrixFunc {
00015 public:
00016     AMatrixFunc(CNuc*, const Config &configure);
00020     CNuc *compound() const {return compound_};
00024     const Config &configure() const {return configure_};
00025
00026     void ClearMatrices();
00027     void FillMatrices(EPoint*);
00028     void InvertMatrices();
00029     void CalculateTMatrix(EPoint*);
00033     void CalculateCrossSection();
00034
00035     complex GetAMatrixElement(int,int,int) const;
00036     matrix_c *GetJSpecAInvMatrix(int);
00037     void AddAInvMatrixElement(int,int,int,complex);
00038     void AddAMatrix(matrix_c);
00039 private:
00040     const Config &configure_;
00041     CNuc *compound_;
00042     vector_matrix_c a_inv_matrices_;
00043     vector_matrix_c a_matrices_;
00044 };
00045
00046 #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.](#)

```
00001 #ifndef ANGCOEFF_H
00002 #define ANGCOEFF_H
00003
00005
00011 class AngCoeff {
00012 public:
00016     static double ClebGord(double,double,double,double,double,double);
00020     static double Racah(double,double,double,double,double,double);
00021 };
00022
00023 #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.](#)

```
00001 #ifndef AZURECALC_H
00002 #define AZURECALC_H
00003
00004 #include "Minuit2/FCNBase.h"
00005 #include "Constants.h"
00006 #include <vector>
00007
00008 class Config;
00009 class EData;
00010 class CNuc;
00011
00013
00021 class AZURECalc : public ROOT::Minuit2::FCNBase {
00022 public:
00027     AZURECalc(EData* data,CNuc* compound, const Config& configure) : configure_(configure) {
00028         data_=data;
00029         compound_=compound;
00030     };
00031
00032     ~AZURECalc() {};
00036     virtual double Up() const {return theErrorDef;};
00042     virtual double operator()(const vector_r&) const;
00043
00047     const Config &configure() const {return configure_};
00051     EData *data() const {return data_};
00055     CNuc *compound() const {return compound_};
00056
00060     void SetErrorDef(double def) {theErrorDef=def;};
00061 private:
00062     const Config &configure_;
00063     EData *data_;
00064     CNuc *compound_;
00065     double theErrorDef;
00066 };
00067
00068 #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.](#)

```
00001 #ifndef AZUREFBUFFER_H
00002 #define AZUREFBUFFER_H
00003
00004 #include <fstream>
00005 #include <string>
00006 #include <assert.h>
00007
00008
00009
00015 class AZUREFBuffer {
00016 public:
00021 AZUREFBuffer(int entranceKey,int exitKey,std::string outputdir,bool isExtrap,bool isAngDist) :
00022     isAngDist_(isAngDist) {
00023         char filename[256];
00024         entrancekey_=entranceKey;
00025         exitkey_=exitKey;
00026         if(exitkey_== -1) {
00027             if(!isExtrap)
00028                 sprintf(filename,"%sAZUREOut_aa=%d_TOTAL_CAPTURE.out",outputdir.c_str(),entranceKey);
00029             else sprintf(filename,"%sAZUREOut_aa=%d_TOTAL_CAPTURE.extrap",outputdir.c_str(),entranceKey);
00030             if(!isExtrap)
00031                 sprintf(filename,"%sAZUREOut_aa=%d_R=%d.out",outputdir.c_str(),entranceKey,exitKey);
00032             else {
00033                 if(!isAngDist)
00034                     sprintf(filename,"%sAZUREOut_aa=%d_R=%d.extrap",outputdir.c_str(),entranceKey,exitKey);
00035                 else sprintf(filename,"%sAZUREOut_aa=%d_R=%d.acoeff",outputdir.c_str(),entranceKey,exitKey);
00036             }
00037             fbuffer_=new std::filebuf;
00038             fbuffer_->open(filename,std::ios::out);
00039             assert(fbuffer_->is_open());
00040         };
00043 ~AZUREFBuffer() {
00044     fbuffer_->close();
00045     delete fbuffer_;
00046 };
00050 bool IsAngDist() const {return isAngDist_};
00054 int GetEntranceKey() const {return entrancekey_};
00058 int GetExitKey() const {return exitkey_};
00062 std::filebuf *GetFBuffer() {return fbuffer_};
00063 private:
00064     bool isAngDist_;
00065     int entrancekey_;
00066     int exitkey_;
00067     std::filebuf *fbuffer_;
00068 };
00069
00070 #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.](#)

```
00001 #ifndef AZUREMAIN_H
00002 #define AZUREMAIN_H
00003
00004 #include "CNuc.h"
00005 #include "EData.h"
00006
00008
00014 class AZUREMain {
00015 public:
00020   AZUREMain(const Config &configure) : configure_(configure) {
00021     compound_ = new CNuc;
00022     data_ = new EData;
00023   };
00027   ~AZUREMain() {
00028     delete compound_;
00029     delete data_;
00030   };
00035   int operator()();
00039   const Config &configure() const {return configure_};
00043   CNuc *compound() const {return compound_};
00047   EData *data() const {return data_};
00048 private:
00049   const Config &configure_;
00050   CNuc *compound_;
00051   EData *data_;
00052 };
00053
00054 #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.](#)

```
00001 #ifndef AZUREOUTPUT_H
00002 #define AZUREOUTPUT_H
00003
00004 #include "AZUREFBuffer.h"
00005 #include <vector>
00006
00008
00016 class AZUREOutput {
00017 public:
00018     AZUREOutput(std::string);
00019     ~AZUREOutput();
00020     bool IsExtrap() const;
00021     std::filebuf *operator()(int entranceKey, int exitKey, bool isAngDist=false);
00022     int NumAZUREFBuffers() const;
00023     int IsAZUREFBuffer(int, int, bool);
00024     std::string GetOutputDir() const;
00025     void AddAZUREFBuffer(AZUREFBuffer*);
00026     void SetExtrap();
00027     AZUREFBuffer *GetAZUREFBuffer(int);
00028 private:
00029     bool is_extrap_;
00030     std::string outputdir_;
00031     std::vector<AZUREFBuffer*> azurefbuffers_;
00032 };
00033
00034 #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.](#)

```
00001 #ifndef AZUREPARAMS_H
00002 #define AZUREPARAMS_H
00003
00004 #include <Minuit2/MnUserParameters.h>
00005 #include <iostream>
00006 #include <fstream>
00007 #include <iomanip>
00008 #include "Constants.h"
00009
00010 class Config;
00011
00013
00020 class AZUREParams {
00021 public:
00022     ROOT::Minuit2::MnUserParameters &GetMinuitParams();
00023     void ReadUserParameters(const Config&);
00024     void WriteUserParameters(const Config&, bool);
00025     void WriteParameterErrors(const std::vector<std::pair<double, double> >&, const Config&);
00026 private:
00027     ROOT::Minuit2::MnUserParameters params_;
00028 };
00029
00030 #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.](#)

```
00001 #ifndef CNUC_H
00002 #define CNUC_H
00003
00004 #include <string>
00005 #include <map>
00006 #include "JGroup.h"
00007 #include "PPair.h"
00008
00009 namespace ROOT {
00010     namespace Minuit2 {
00011         class MnUserParameters;
00012     }
00013 }
00014 class Config;
00015
00016
00025 class CNuc {
00026 public:
00027     bool IsPairKey(int);
00028     int NumPairs() const;
```

```

00029 int NumJGroups() const;
00030 int IsPair(PPair);
00031 int IsJGroup(JGroup);
00032 int GetPairNumFromKey(int);
00033 int Fill(const Config&);
00034 void ParseExternalCapture(const Config&, std::map<int, int>&);
00035 int GetMaxLValue() const;
00036 void Initialize(const Config&);
00037 void AddPair(PPair);
00038 void AddJGroup(JGroup);
00039 void PrintNuc(const Config&);
00040 void TransformIn(const Config&);
00041 void SortPathways(const Config&);
00042 void PrintPathways(const Config&);
00043 void CalcBoundaryConditions(const Config&);
00044 void PrintBoundaryConditions(const Config&);
00045 void CalcAngularDists(int);
00046 void PrintAngularDists(const Config&);
00047 void FillMnParams(ROOT::Minuit2::MnUserParameters&);
00048 void FillCompoundFromParams(const vector_r &);
00049 void TransformOut(const Config&);
00050 void PrintTransformParams(const Config&);
00051 void SetMaxLValue(int);
00052 void CalcShiftFunctions(const Config&);
00053 complex CalcExternalWidth(JGroup*, ALevel*, AChannel*, bool, const Config&);
00054 PPair *GetPair(int);
00055 JGroup *GetJGroup(int);
00056 CNuc *Clone() const;
00057 private:
00058     std::vector<PPair> pairs_;
00059     std::vector<JGroup> jgroups_;
00060     int maxLValue_;
00061 };
00062
00063 extern double DoubleFactorial(int);
00064
00065 #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.](#)

```

00001 #ifndef CONFIG_H
00002 #define CONFIG_H
00003
00004 #include <string>
00005 #include <fstream>
00006
00008
00013 struct RateParams {
00015     bool useFile;
00017     std::string temperatureFile;
00019     int entrancePair;
00021     int exitPair;

```

```

00023 double minTemp;
00025 double maxTemp;
00027 double tempStep;
00028 };
00029
00031
00037 class Config {
00038 public:
00039     Config(std::ostream& stream);
00040     void Reset();
00044     enum ParameterFlags {
00045         USE_AMATRIX = (1<<0),
00046         PERFORM_ERROR_ANALYSIS = (1<<1),
00047         PERFORM_FIT = (1<<2),
00048         CALCULATE_WITH_DATA = (1<<3),
00049         USE_PREVIOUS_PARAMETERS = (1<<4),
00050         USE_EXTERNAL_CAPTURE = (1<<5),
00051         USE_PREVIOUS_INTEGRALS = (1<<6),
00052         CALCULATE_REACTION_RATE = (1<<7),
00053         TRANSFORM_PARAMETERS = (1<<8),
00054         USE_BRUNE_FORMALISM = (1<<9),
00055         IGNORE_ZERO_WIDTHS = (1<<10),
00056         USE_RMC_FORMALISM = (1<<11),
00057         USE_GSL_COULOMB_FUNC = (1<<12),
00058         USE_LONGWAVELENGTH_APPROX = (1<<13),
00059     };
00063     enum CheckFileFlags {
00064         CHECK_COMPOUND_NUCLEUS = (1<<0),
00065         CHECK_PATHWAYS = (1<<1),
00066         CHECK_DATA = (1<<2),
00067         CHECK_ENERGY_DEP = (1<<3),
00068         CHECK_LEGENDRE = (1<<4),
00069         CHECK_BOUNDARY_CONDITIONS = (1<<5),
00070         CHECK_ANGULAR_DIST = (1<<6),
00071         CHECK_COUL_AMPLITUDES = (1<<7),
00072     };
00074     std::ostream &outStream;
00076     std::string configfile;
00078     bool stopFlag;
00080     unsigned int paramMask;
00082     unsigned int screenCheckMask;
00084     unsigned int fileCheckMask;
00086     double chiVariance;
00088     std::string outputdir;
00090     std::string checkdir;
00092     std::string paramfile;
00094     std::string integralsfile;
00096     RateParams rateParams;
00098     static const int maxLOrder=20;
00099     int ReadConfigFile();
00100     #ifndef NO_STAT
00101     int CheckForInputFiles();
00102     #endif
00103 };
00104
00105 #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.](#)

```
00001 #ifndef CONSTANTS_H
00002 #define CONSTANTS_H
00003
00004 #include <complex>
00005 #include <vector>
00006
00007 const double pi=3.141592650;
00008 const double hbarc=197.32696310;
00009 const double uconv=931.4940880;
00010 const double fstruc=1.00/137.0359996790;
00011 const double boltzConst=8.6171e-2;
00012 const double lightSpeedInCmPerS=29979245800.;
00013 const double avagadroNum=6.02214179e23;
00014 const double nuclearMagneton=0.105155;
00015 const unsigned char isE1 = 1 << 0;
00016 const unsigned char isM1 = 1 << 1;
00017 const unsigned char isE2 = 1 << 2;
00018 const int maxECMult=2;
00019
00020 typedef std::complex<double> complex;
00021 typedef std::vector<double> vector_r;
00022 typedef std::vector<std::complex<double> > vector_c;
00023 typedef std::vector<std::vector<double> > matrix_r;
00024 typedef std::vector<std::vector<std::complex<double> > > matrix_c;
00025 typedef std::vector<std::vector<std::vector<double> > > vector_matrix_r;
00026 typedef std::vector<std::vector<std::vector<std::complex<double> > > > vector_matrix_c;
00027
00028 #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.](#)

```
00001 #ifndef COULFUNC_H
00002 #define COULFUNC_H
00003
00004 class PPair;
00005
00007
00013 struct CoulWaves {
00015     double F;
00017     double dF;
00019     double G;
00021     double dG;
00022 };
00023
00025
00032 class CoulFunc {
00033 public:
00034     CoulFunc(PPair *pPair, bool useGSLFunctions);
00035     int z1() const;
00036     int z2() const;
00037     double redmass() const;
00038     int lLast() const;
00039     double radiusLast() const;
00040     double energyLast() const;
```

```

00041     struct CoulWaves coulLast() const;
00042     void setLast(int, double, double, CoulWaves);
00043     CoulWaves operator()(int, double, double);
00044     double Penetrability(int, double, double);
00045     double PEShift(int, double, double);
00046     double PEShift_dE(int, double, double);;
00047 private:
00048     static double thisPEShift(double, void*);
00049     typedef struct DEShiftParams {
00050         CoulFunc *coulFunc;
00051         int lValue;
00052         double radius;
00053     } DEShiftParams;
00054     DEShiftParams dEShiftParams_;
00055     bool useGSLFunctions_;
00056     int z1_;
00057     int z2_;
00058     int lLast_;
00059     double redmass_;
00060     double radiusLast_;
00061     double energyLast_;
00062     struct CoulWaves coulLast_;
00063 };
00064
00065 #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.](#)

```

00001 #ifndef DATALINE_H
00002 #define DATALINE_H
00003
00004 #include <fstream>
00005
00007
00012 class DataLine {
00013 public:
00017     DataLine(std::ifstream &stream) {
00018         stream » energy_ » angle_ » crossSection_ » error_;
00019     };
00023     double angle() const {return angle_;};
00027     double energy() const {return energy_;};
00031     double crossSection() const {return crossSection_;};
00035     double error() const {return error_;};
00036 private:
00037     double angle_;
00038     double energy_;
00039     double crossSection_;
00040     double error_;
00041 };
00042
00043 #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.](#)

```
00001 #ifndef DECAY_H
00002 #define DECAY_H
00003
00004 #include "KGroup.h"
00005 #include "KLGroup.h"
00006
00007
00014 class Decay {
00015 public:
00016     Decay(int);
00017     int GetPairNum() const;
00018     int NumKGroups() const;
00019     int NumKLGroups() const;
00020     int IsKGroup(KGroup);
00021     int IsKLGroup(KLGroup);
00022     void AddKGroup(KGroup);
00023     void AddKLGroup(KLGroup);
00024     KGroup *GetKGroup(int);
00025     KLGroup *GetKLGroup(int);
00026 private:
00027     int pair_;
00028     std::vector<KGroup> kgroups_;
00029     std::vector<KLGroup> klgroups_;
00030 };
00031
00032
00033 #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.](#)

```

00001 #ifndef ECINTEGRAL_H
00002 #define ECINTEGRAL_H
00003
00004 #include "CoulFunc.h"
00005 #include "WhitFunc.h"
00006 #include "Config.h"
00007
00008 class EffectiveCharge;
00009
00010
00011
00018 class ECIntegral {
00019 public:
00025 ECIntegral(PPair *pPair, const Config& configure) {
00026     params_.coulFunc = new CoulFunc(pPair,!!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
00027     params_.whitFunc = new WhitFunc(pPair);
00028     params_.useLongWavelengthApprox = !(configure.paramMask&Config::USE_LONGWAVELENGTH_APPROX);
00029     pair_ = pPair;
00030 };
00034 ~ECIntegral() {
00035     delete params_.coulFunc;
00036     delete params_.whitFunc;
00037 };
00038 complex operator() (int,int,double,double,double,double,int,char,double,double,bool);
00039 private:
00040 void ResetIntegrals() {FW_=0.;GW_=0.;};
00041 void Integrate(double);
00042 static double FWIntegrand(double,void*);
00043 static double GWIntegrand(double,void*);
00044 static double WWIntegrand(double,void*);
00045 CoulFunc *coulfunction() const {return params_.coulFunc;};
00046 WhitFunc *whitfunction() const {return params_.whitFunc;};
00047 PPair *pair() const {return pair_;};
00048 double FW() const {return FW_;};
00049 double GW() const {return GW_;};
00050 typedef struct Params {
00051     EffectiveCharge* effectiveCharge;
00052     CoulFunc *coulFunc;
00053     WhitFunc *whitFunc;
00054     int liValue;
00055     int lfValue;
00056     int multLValue;
00057     double pairEnergy;
00058     double bindingEnergy;
00059     bool useLongWavelengthApprox;
00060 } Params;
00061 Params params_;
00062 PPair *pair_;
00063 double FW_;
00064 double GW_;
00065 };
00066
00067
00068
00069
00070 #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.](#)

```
00001 #ifndef ECMGROUP_H
00002 #define ECMGROUP_H
00003
00004 #include "Constants.h"
00005
00007
00017 class ECMGroup {
00018 public:
00019     ECMGroup(char, int, int, double, int, int, int);
00020     ECMGroup(char, int, int, double, int, int, int, int, int, int);
00021     bool IsChannelCapture() const;
00022     char GetRadType() const;
00023     int GetMult() const;
00024     int GetL() const;
00025     int GetFinalChannel() const;
00026     int GetJGroupNum() const;
00027     int GetLevelNum() const;
00028     int GetChanCapDecay() const;
00029     int GetChanCapKGroup() const;
00030     int GetChanCapMGroup() const;
00031     int GetIntChannelNum() const;
00032     double GetJ() const;
00033     double GetStatSpinFactor() const;
00034     void SetStatSpinFactor(double);
00035 private:
00036     char radtype_;
00037     int mult_;
00038     int li_;
00039     int chf_;
00040     int jGroupNum_;
00041     int levelNum_;
00042     bool ischancap_;
00043     int chdecay_;
00044     int chkgroup_;
00045     int chmgroup_;
00046     int internalChannel_;
00047     double ji_;
00048     double statspinfactor_;
00049     complex tmatrix_;
00050 };
00051
00052 #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.](#)

```

00001 #ifndef EDATA_H
00002 #define EDATA_H
00003
00004 #include "ESegment.h"
00005 #include "TargetEffect.h"
00006 #include "EDataIterator.h"
00007
00008 class CNuc;
00009 namespace ROOT {
00010     namespace Minuit2 {
00011         class MnUserParameters;
00012     }
00013 }
00014
00016
00021 class EData {
00022 public:
00023     EData();
00024     int NumSegments() const;
00025     int Fill(const Config&, CNuc*);
00026     int MakePoints(const Config&, CNuc*);
00027     int Iterations() const;
00028     int NumTargetEffects() const;
00029     int GetNormParamOffset() const;
00030     int ReadTargetEffectsFile(const Config&, CNuc*);
00031     bool IsFit() const;
00032     bool IsErrorAnalysis() const;
00033     bool IsSegmentKey(int);
00034     void SetFit(bool);
00035     void SetErrorAnalysis(bool);
00036     void Iterate();
00037     void ResetIterations();
00038     int Initialize(CNuc*, const Config&);
00039     void AddSegment(ESegment);
00040     void PrintData(const Config&);
00041     void CalcLegendreP(int);
00042     void PrintLegendreP(const Config&);
00043     int CalcEDependentValues(CNuc*, const Config&);
00044     void PrintEDependentValues(const Config&, CNuc*);
00045     void CalcCoulombAmplitude(CNuc*);
00046     void PrintCoulombAmplitude(const Config&, CNuc*);
00047     void WriteOutputFiles(const Config&, bool=false);
00048     int CalculateECAmplitudes(CNuc*, const Config&);
00049     void MapData();
00050     void AddTargetEffect(TargetEffect);
00051     void SetNormParamOffset(int);
00052     void FillMnParams(ROOT::Minuit2::MnUserParameters&);
00053     void FillNormsFromParams(const vector_r &);
00054     void DeleteLastSegment();
00055     ESegment *GetSegment(int);
00056     ESegment *GetSegmentFromKey(int);
00057     EData *Clone() const;
00058     TargetEffect *GetTargetEffect(int);
00059     EDataIterator begin();
00060     EDataIterator end();
00061     std::vector<ESegment>& GetSegments();
00062 private:
00063     std::vector<TargetEffect> targetEffects_;
00064     std::vector<ESegment> segments_;
00065     int iterations_;
00066     int normParamOffset_;
00067     bool isFit_;
00068     bool isErrorAnalysis_;
00069 };
00070
00071 #endif

```

8.171 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EDatalterator.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
00013
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.](#)

```
00001 #ifndef EFFECTIVECHARGE_H
00002 #define EFFECTIVECHARGE_H
00003
00004 class PPair;
00005
00007
00014 class EffectiveCharge {
00015 public:
00016     EffectiveCharge(PPair*, double, int);
00017     double operator()(double);
00018 private:
00019     static double Integrand(double, void*);
00020     int z1_;
00021     int z2_;
00022     int L_;
00023     double m1_;
00024     double m2_;
00025     double energy_;
00026 };
00027
00028 #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.](#)

```
00001 #ifndef EIGENFUNC_H
00002 #define EIGENFUNC_H
00003
00004 #include "Constants.h"
00005
00007
00013 class EigenFunc {
00014 public:
00015     EigenFunc(const matrix_r&);
00016     EigenFunc(const matrix_r&, const std::vector<vector_r> &);
00020     const vector_r& eigenvalues() const {return eigenvalues_;};
00024     const matrix_r& eigenvectors() const {return eigenvectors_;};
00025 private:
00026     vector_r eigenvalues_;
00027     matrix_r eigenvectors_;
00028 };
00029
00030 #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.](#)

```
00001 #ifndef EPOINT_H
00002 #define EPOINT_H
00003
00004 #include "Constants.h"
00005
00006
00007
00013 struct EnergyMap {
00015     int segment;
00017     int point;
00018 };
00019
00020 class ESegment;
00021 class EData;
00022 class CNuc;
00023 class PPair;
00024 class TargetEffect;
00025 class DataLine;
00026 class Config;
00027
00028
00029
00036 class EPoint {
00037     public:
00038         EPoint(DataLine, ESegment*);
00039         EPoint(double, double, ESegment*);
00040         EPoint(double, double, int, int, bool, bool, bool, double, int, int);
00041         bool IsDifferential() const;
00042         bool IsPhase() const;
00043         bool IsMapped() const;
00044         bool IsTargetEffect() const;
00045         bool IsAngularDist() const;
00046         int GetEntranceKey() const;
00047         int GetExitKey() const;
00048         int GetMaxLOrder() const;
00049         int GetL() const;
00050         int NumLocalMappedPoints() const;
00051         int NumSubPoints() const;
00052         int GetTargetEffectNum() const;
00053         int GetMaxAngDistOrder() const;
00054         int GetNumAngularDists() const;
00055         double GetLabAngle() const;
00056         double GetCMAngle() const;
00057         double GetLabEnergy() const;
00058         double GetCMEnergy() const;
00059         double GetExcitationEnergy() const;
00060         double GetLegendreP(int) const;
00061         double GetLabCrossSection() const;
00062         double GetCMCrossSection() const;
00063         double GetLabCrossSectionError() const;
00064         double GetCMCrossSectionError() const;
00065         double GetGeometricalFactor() const;
00066         double GetFitCrossSection() const;
00067         double GetSFactorConversion() const;
00068         double GetSqrtPenetrability(int,int) const;
00069         double GetJ() const;
00070         double GetStoppingPower() const;
```

```

00071 double GetTargetThickness() const;
00072 double GetAngularDist(int) const;
00073 complex GetLoElement(int,int) const;
00074 complex GetExpCoulombPhase(int,int) const;
00075 complex GetExpHardSpherePhase(int,int) const;
00076 complex GetCoulombAmplitude() const;
00077 complex GetECAmplitude(int,int) const;
00078 EnergyMap GetMap() const;
00079 void Initialize(CNuc*,const Config&);
00080 void ConvertLabEnergy(PPair*);
00081 void ConvertDecayEnergy(PPair*);
00082 void ConvertLabAngle(PPair*);
00083 void ConvertLabAngle(PPair*,PPair*,const Config&);
00084 void ConvertCrossSection(PPair*,PPair*);
00085 void AddLegendreP(double);
00086 void SetGeometricalFactor(double);
00087 void SetFitCrossSection(double);
00088 void SetSFactorConversion(double);
00089 void SetExitKey(int);
00090 void CalcLegendreP(int,TargetEffect*);
00091 void CalcEDependentValues(CNuc*,const Config&);
00092 void AddLoElement(int,int,complex);
00093 void AddSqrtPenetrability(int,int,double);
00094 void AddExpCoulombPhase(int,int,complex);
00095 void AddExpHardSpherePhase(int,int,complex);
00096 void CalcCoulombAmplitude(CNuc*);
00097 void SetCoulombAmplitude(complex);
00098 void CalculateECAmplitudes(CNuc*,const Config&);
00099 void AddECAmplitude(int,int,complex);
00100 void Calculate(CNuc*,const Config &configure,EPoint* parent=NULL, int subPointNum=0);
00101 void SetMap(int,int);
00102 void AddLocalMappedPoint(EPoint*);
00103 void ClearLocalMappedPoints();
00104 void SetTargetEffectNum(int);
00105 void AddSubPoint(EPoint);
00106 void IntegrateTargetEffect();
00107 void SetParentData(EData*);
00108 void SetStoppingPower(double);
00109 void SetTargetThickness(double);
00110 void SetAngularDists(vector_r);
00111 EData *GetParentData() const;
00112 EPoint* GetLocalMappedPoint(int) const;
00113 EPoint* GetSubPoint(int);
00114 std::vector<EPoint>& GetSubPoints();
00115 std::vector<EPoint*>& GetMappedPoints();
00116 private:
00117 bool is_differential_;
00118 bool is_phase_;
00119 bool is_mapped_;
00120 bool is_ang_dist_;
00121 int entrance_key_;
00122 int exit_key_;
00123 int l_value_;
00124 int targetEffectNum_;
00125 int max_ang_dist_order_;
00126 double cm_angle_;
00127 double lab_angle_;
00128 double cm_energy_;
00129 double lab_energy_;
00130 double excitation_energy_;
00131 double cm_crosssection_;
00132 double cm_dcrosssection_;
00133 double lab_crosssection_;
00134 double lab_dcrosssection_;
00135 double geofactor_;
00136 double fitcrosssection_;
00137 double sfactorconv_;
00138 double j_value_;
00139 double stoppingPower_;
00140 double targetThickness_;
00141 struct EnergyMap energy_map_;
00142 complex coulombamplitude_;
00143 vector_r legendreP_;
00144 vector_r angularDists_;
00145 matrix_c lo_elements_;
00146 matrix_r penetrabilities_;
00147 matrix_c coulombphase_;
00148 matrix_c hardspherephase_;
00149 matrix_c ec_amplitudes_;
00150 std::vector<EPoint*> local_mapped_points_;
00151 std::vector<EPoint> integrationPoints_;
00152 EData* parentData_;
00153 };
00154
00155 #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.](#)

```
00001 #ifndef EQUATION_H
00002 #define EQUATION_H
00003
00004 #include <string>
00005 #include <vector>
00006 #include <exception>
00007 #include <sstream>
00008 #include <map>
00009
00010 class Config;
00011
00013
00019 class GenericFunction {
00020 public:
00024 GenericFunction() {};
00028 GenericFunction(double (*function)(double)) :
00029     function_(function) {};
00033 double Evaluate(double value) const {
00034     return (*function_)(value);
00035 };
00036 private:
00037     double (*function_)(double);
00038 };
00039
00041
00049 class Equation {
00050 public:
00051     Equation();
00052     Equation(std::string equation, int numParams, const Config&);
00053     Equation(std::string equation, std::vector<double> parameters, const Config&);
00054     Equation(std::string equation, double parameters[], size_t arraySize, const Config&);
00055     void Initialize(std::string equation, int numParams, const Config &);
00056     void SetParameter(unsigned int index, double value, const Config&);
00057     std::vector<double> GetParameters() const;
00058     double Evaluate(const Config&, double x=0.0) const;
00059 private:
00060     enum Associativity {LEFT, RIGHT};
00061     enum TokenType {NUMBER=1, OPERATOR=2, VARIABLE=4, PARAMETER=8, LEFTPAR=16, RIGHTPAR=32, FUNCTION=64};
00062     enum OperatorType {ADD=0, SUBTRACT=0, MULT=1, DIVIDE=1, POWER=2, BADTYPE=10};
00063     typedef std::pair<TokenType, std::string> TokenPair;
00064     void BuildFunctionList();
00065     void Parse(const Config&);
00066     bool IsOperator(char) const;
00067     bool IsDigit(char) const;
00068     unsigned int FindFunction(unsigned int &position);
00069     TokenPair GetToken(unsigned int &position, const Config&);
```

```

00070 OperatorType GetOperatorType(char) const;
00071 Associativity GetOperatorAssociativity(char) const;
00072 std::string BinaryOperation(double left, double right, char op, const Config&) const;
00073 double FunctionOperation(TokenPair token, double x, const Config&) const;
00074 double GetTokenValue(TokenPair token, double x, const Config&) const;
00075 std::string infixEquation_;
00076 std::vector<TokenPair> output_;
00077 std::vector<double> parameters_;
00078 std::vector<Equation> subEquations_;
00079 std::map<std::string, GenericFunction> functionList_;
00080 };
00081
00083
00089 class SyntaxError : public std::exception {
00090 public:
00095 SyntaxError(std::string equation, int type, int position=-1) {
00096     std::string typeMessage = GetTypeMessage(type);
00097     std::ostringstream stm;
00098     stm << "Syntax Error in " << equation << ':' << typeMessage << std::endl;
00099     if(position!=-1) {
00100         stm << "                "; for(int i=0;i<position;i++) stm << ' '; stm << '^';
00101     }
00102     messageString_=stm.str();
00103     message_=messageString_.c_str();
00104 };
00105 ~SyntaxError() throw() {
00106     delete[] message_;
00107 };
00111 virtual const char* what() const throw() {
00112     return message_;
00113 };
00114 private:
00115     std::string GetTypeMessage(int type) {
00116         switch(type) {
00117             case 0: return " Unknown token type.";
00118             case 1: return " Parameter out of range.";
00119             case 2: return " Mismatched Parentheses.";
00120             case 3: return " Unexpected token.";
00121             default: return " Unknown error.";
00122         }
00123     };
00124     std::string messageString_;
00125     const char* message_;
00126 };
00127
00128
00129 #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.](#)

```

00001 #ifndef ESEGMENT_H
00002 #define ESEGMENT_H
00003
00004 #include "EPoint.h"
00005
00006 class EData;
00007 class ExtrapolLine;

```

```

00008 class SegLine;
00009
00011
00017 class ESegment {
00018 public:
00019     ESegment(SegLine);
00020     ESegment(ExtrapLine);
00021     bool IsInSegment(EPoint);
00022     bool IsDifferential() const;
00023     bool IsPhase() const;
00024     bool IsTargetEffect() const;
00025     bool IsVaryNorm() const;
00026     bool IsAngularDist() const;
00027     int IsTotalCapture() const;
00028     int NumPoints() const;
00029     int GetEntranceKey() const;
00030     int GetExitKey() const;
00031     int Fill(CNuc*, EData*, const Config&);
00032     int GetL() const;
00033     int GetTargetEffectNum() const;
00034     int GetSegmentKey() const;
00035     int GetMaxAngDistOrder() const;
00036     double GetMinEnergy() const;
00037     double GetMaxEnergy() const;
00038     double GetMinAngle() const;
00039     double GetMaxAngle() const;
00040     double GetSegmentChiSquared() const;
00041     double GetEStep() const;
00042     double GetAStep() const;
00043     double GetJ() const;
00044     double GetNorm() const;
00045     double GetNominalNorm() const;
00046     double GetNormError() const;
00047     std::string GetDataFile() const;
00048     void AddPoint(EPoint);
00049     void SetSegmentChiSquared(double);
00050     void SetTargetEffectNum(int);
00051     void SetSegmentKey(int);
00052     void SetNorm(double);
00053     void SetExitKey(int);
00054     void SetIsTotalCapture(int);
00055     void SetVaryNorm(bool);
00056     EPoint *GetPoint(int);
00057     std::vector<EPoint>& GetPoints();
00058 private:
00059     bool isdifferential_;
00060     bool isphase_;
00061     bool isTargetEffect_;
00062     bool varyNorm_;
00063     bool isAngDist_;
00064     int isTotalCapture_;
00065     int entrancekey_;
00066     int exitkey_;
00067     int l_;
00068     int targetEffectNum_;
00069     int segmentKey_;
00070     int maxAngDistOrder_;
00071     double min_e_;
00072     double max_e_;
00073     double min_a_;
00074     double max_a_;
00075     double e_step_;
00076     double a_step_;
00077     double segment_chi_squared_;
00078     double j_;
00079     double dataNorm_;
00080     double dataNormNominal_;
00081     double dataNormError_;
00082     std::string datafile_;
00083     std::vector<EPoint> points_;
00084 };
00085
00086 #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.](#)

```
00001 #ifndef EXTRAPLINE_H
00002 #define EXTRAPLINE_H
00003
00004 #include <iostream>
00005 #include <string>
00006
00008
00013 class ExtrapLine {
00014 public:
00018     ExtrapLine(std::istream &stream) {
00019         stream » isActive_ » entranceKey_ » exitKey_ » minE_
00020             » maxE_ » eStep_ » minA_ » maxA_ » aStep_ » isDiff_;
00021         if(isDiff_==2) stream » phaseJ_ » phaseL_;
00022         else if(isDiff_==3) stream » maxAngDistOrder_;
00023     };
00024
00028     int isActive() const {return isActive_};
00033     int entranceKey() const {return entranceKey_};
00038     int exitKey() const {return exitKey_};
00042     double minE() const {return minE_};
00046     double maxE() const {return maxE_};
00050     double minA() const {return minA_};
00054     double maxA() const {return maxA_};
00058     double eStep() const {return eStep_};
00062     double aStep() const {return aStep_};
00067     int isDiff() const {return isDiff_};
00072     double phaseJ() const {return phaseJ_};
00077     int phaseL() const {return phaseL_};
00082     int maxAngDistOrder() const {return maxAngDistOrder_};
00083 private:
00084     int isActive_;
00085     int entranceKey_;
00086     int exitKey_;
00087     double minE_;
00088     double maxE_;
00089     double minA_;
00090     double maxA_;
00091     double eStep_;
00092     double aStep_;
00093     int isDiff_;
00094     double phaseJ_;
00095     int phaseL_;
00096     int maxAngDistOrder_;
00097 };
00098
00099 #endif
```

8.185 /Users/kuba/Desktop/R-Matrix/AZURE2/include/GenMatrixFunc.h

File Reference

```
#include "Constants.h"
#include "Config.h"
```

Classes

- struct [TempTMatrix](#)

A temporary T-Matrix structure.

- class [GenMatrixFunc](#)

A generalized function class to calculate cross sections.

8.186 GenMatrixFunc.h

[Go to the documentation of this file.](#)

```

00001 #ifndef GENMATRIXFUNC_H
00002 #define GENMATRIXFUNC_H
00003
00004 #include "Constants.h"
00005 #include "Config.h"
00006
00007 class EPoint;
00008 class CNuc;
00009
00010
00011
00018 struct TempTMatrix {
00020     double jValue;
00022     int lValue;
00024     int lpValue;
00026     complex TMatrix;
00027 };
00028
00029
00031
00038 class GenMatrixFunc {
00039 public:
00040     GenMatrixFunc() {};
00041     virtual ~GenMatrixFunc() {};
00045     virtual void ClearMatrices()=0;
00049     virtual void FillMatrices(EPoint*)=0;
00053     virtual void InvertMatrices()=0;
00057     virtual void CalculateTMatrix(EPoint*)=0;
00058     void CalculateCrossSection(EPoint*);
00059     void NewTempTMatrix(TempTMatrix);
00060     void AddToTempTMatrix(int,complex);
00061     void ClearTempTMatrices();
00062     void AddTMatrixElement(int,int,complex,int decayNum=1);
00063     void AddECTMatrixElement(int,int,complex);
00064     int IsTempTMatrix(double,int,int);
00065     int NumTempTMatrices() const;
00066     TempTMatrix *GetTempTMatrix(int);
00067     complex GetTMatrixElement(int,int,int decayNum=1) const;
00068     complex GetECTMatrixElement(int,int) const;
00069
00073     virtual CNuc *compound() const = 0;
00077     virtual const Config& configure() const = 0;
00078 protected:
00080     std::vector<matrix_c> tmatrix_;
00082     matrix_c ec_tmatrix_;
00083 private:
00084     std::vector<TempTMatrix> temp_t_matrices_;
00085 };
00086
00087 #endif

```

8.187 /Users/kuba/Desktop/R-Matrix/AZURE2/include/GSLEException.h

File Reference

```

#include <iostream>
#include <exception>
#include <sstream>

```

Classes

- class [GSLEException](#)

8.188 GSLEException.h

[Go to the documentation of this file.](#)

```

00001 #ifndef GSLEXCEPTION_H
00002 #define GSLEXCEPTION_H
00003
00004 #include <iostream>
00005 #include <exception>
00006 #include <sstream>
00007
00013 class GSLEException : public std::exception {
00014 public:
00015     GSLEException(std::string message, std::string line = "", std::string file = "") {
00016         std::ostringstream stm;
00017         if(line != "" && file != "") {
00018             stm << "Exception thrown from line " << line << " of file " << file << " with message: " << std::endl
00019             << message;
00020         } else {
00021             stm << "Exception thrown with message: " << std::endl
00022             << message;
00023         }
00024         messageString_=stm.str();
00025         message_=messageString_.c_str();
00026     };
00027     ~GSLEException() throw() {
00028     };
00029     virtual const char* what() const throw() {
00030         return message_;
00031     };
00032     static void GSLErrorHandler(const char*, const char*, int, int);
00033 private:
00034     std::string messageString_;
00035     const char* message_;
00036 };
00037
00038 #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.](#)

```

00001 #ifndef INTEGRATEDFERMIFUNC_H
00002 #define INTEGRATEDFERMIFUNC_H
00003
00005
00013 class IntegratedFermiFunc {
00014 public:
00015     IntegratedFermiFunc(int,double V0 = 0.);
00016     double operator () (double,double,double);
00017 private:
00018     static const double alpha_;
00019     static const double pi_;
00020     static const double electronMass_;
00021     static const double hbarc_;
00022     static double Integrand(double, void*);
00023     typedef struct Params_ {
00024         int charge;
00025         double gamma0;
00026         double Z;
00027         double radius;
00028         double W0;
00029         double GammaDenom2;
00030         double V0;

```

```

00031     } Params_;
00032     int charge_;
00033     double V0_;
00034 };
00035
00036 #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.](#)

```

00001 #ifndef INTERFERENCE_H
00002 #define INTERFERENCE_H
00003
00004 #include <string>
00005
00006
00016 class Interference {
00017 public:
00018     Interference(int, int, double, std::string);
00019     std::string GetInterferenceType() const;
00020     int GetM1() const;
00021     int GetM2() const;
00022     double GetZ1Z2() const;
00023 private:
00024     int m1_;
00025     int m2_;
00026     double z1z2_;
00027     std::string intertype_;
00028 };
00029
00030 #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.](#)

```
00001 #ifndef JGROUP_H
00002 #define JGROUP_H
00003
00004 #include "Constants.h"
00005 #include "Alevel.h"
00006 #include "AChannel.h"
00007
00008 class NucLine;
00009
00010
00011
00017 class JGroup {
00018 public:
00019     JGroup(NucLine);
00020     JGroup(double,int);
00021     bool IsInRMatrix() const;
00022     int IsLevel(ALevel);
00023     int GetPi() const;
00024     int NumLevels() const;
00025     int NumChannels();
00026     int IsChannel(AChannel);
00027     double GetJ() const;
00028     void AddLevel(ALevel);
00029     void AddChannel(AChannel);
00030     AChannel *GetChannel(int);
00031     ALevel *GetLevel(int);
00032 private:
00033     bool isinrmatrix_;
00034     int pi_;
00035     double j_;
00036     std::vector<ALevel> levels_;
00037     std::vector<AChannel> channels_;
00038 };
00039
00040 #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.](#)

```
00001 #ifndef KGROUP_H
00002 #define KGROUP_H
00003
00004 #include "MGroup.h"
00005 #include "ECMGroup.h"
00006
00007
00008
00016 class KGroup {
00017 public:
00018     KGroup(double, double);
00019     int NumMGroups() const;
00020     int NumECMGroups() const;
00021     int IsMGroup(MGroup);
```

```

00022     double GetS() const;
00023     double GetSp() const;
00024     void AddMGroup(MGroup);
00025     void AddECMGroup(ECMGroup);
00026     MGroup *GetMGroup(int);
00027     ECMGroup *GetECMGroup(int);
00028 private:
00029     double s_;
00030     double sp_;
00031     std::vector<MGroup> mgroups_;
00032     std::vector<ECMGroup> ec_mgroups_;
00033 };
00034
00035 #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.](#)

```

00001 #ifndef KLGROUP_H
00002 #define KLGROUP_H
00003
00004 #include <vector>
00005 #include "Interference.h"
00006
00008
00016 class KLGroup {
00017 public:
00018     KLGroup(int,int);
00019     int GetK() const;
00020     int GetLOrder() const;
00021     int NumInterferences() const;
00022     int IsInterference(Interference);
00023     void AddInterference(Interference);
00024     Interference *GetInterference(int);
00025 private:
00026     int k_;
00027     int lorder_;
00028     std::vector<Interference> interferences_;
00029 };
00030
00031 #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.](#)

```
00001 #ifndef MATRIXINV_H
00002 #define MATRIXINV_H
00003
00004 #include "Constants.h"
00005
00007
00012 class MatrixInv {
00013 public:
00014     MatrixInv(const matrix_c&);
00018     const matrix_c& inverse() const {return inverse_};
00019 private:
00020     matrix_c inverse_;
00021 };
00022
00023 #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.](#)

```
00001 #ifndef MGROUP_H
00002 #define MGROUP_H
00003
00004 #include "Constants.h"
00005
00007
00013 class MGroup {
00014 public:
00015     MGroup(int, int, int);
00016     int GetChNum() const;
00017     int GetChpNum() const;
00018     int GetJNum() const;
00019     double GetStatSpinFactor() const;
00020     void SetStatSpinFactor(double);
00021 private:
00022     int jnum_;
00023     int ch_;
00024     int chp_;
00025     double statspinfactor_;
00026     complex tmatrix_;
00027 };
00028
00029 #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.](#)

```
00001 #ifndef NFIntegral_H
00002 #define NFIntegral_H
00003
00004 #include "PPair.h"
00005 #include "WhitFunc.h"
00006 #include <math.h>
00007
00008
00015 class NFIntegral {
00016 public:
00020   NFIntegral(PPair* pPair) {
00021     params_.whitFunc = new WhitFunc(pPair);
00022     chanrad_ = pPair->GetChRad();
00023     total_sep_e_ = pPair->GetSepE()+pPair->GetExE();
00024   };
00028   ~NFIntegral() {
00029     delete params_.whitFunc;
00030   };
00036   double operator()(int lFinal, double levelEnergy);
00040   double chanRad() const {return chanrad_;};
00044   double totalSepE() const {return total_sep_e_;};
00045 private:
00046   static double Integrand(double, void*);
00047   typedef struct Params {
00048     WhitFunc *whitFunc;
00049     int lfValue;
00050     double bindingEnergy;
00051     double whitChRadSquaredValue;
00052   } Params;
00053   Params params_;
00054   double chanrad_;
00055   double total_sep_e_;
00056 };
00057
00058 #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.](#)

```

00001 #ifndef NUCLINE_H
00002 #define NUCLINE_H
00003
00004 #include <iostream>
00005
00006
00007
00013 class NucLine {
00014 public:
00018   NucLine(std::istream &stream) {
00019     stream >> levelJ_ >> levelPi_ >> levelE_ >> levelFix_ >> aa_ >> ir_
00020       >> s_ >> l_ >> levelID_ >> isActive_ >> channelFix_ >> gamma_ >> j1_ >> pi1_
00021       >> j2_ >> pi2_ >> e2_ >> m1_ >> m2_ >> z1_ >> z2_
00022       >> entranceSepE_ >> sepE_ >> j3_ >> pi3_ >> e3_
00023       >> pType_ >> chRad_ >> g1_ >> g2_ >> ecMultMask_;
00024     s_/=2.;
00025     l_/=2;
00026   };
00030   double levelJ() const {return levelJ_};
00034   int levelPi() const {return levelPi_};
00039   double levelE() const {return levelE_};
00044   int levelFix() const {return levelFix_};
00049   int aa() const {return aa_};
00054   int ir() const {return ir_};
00058   double s() const {return s_};
00062   int l() const {return l_};
00066   int levelID() const {return levelID_};
00071   int isActive() const {return isActive_};
00076   int channelFix() const {return channelFix_};
00080   double gamma() const {return gamma_};
00084   double j1() const {return j1_};
00088   int pi1() const {return pi1_};
00092   double j2() const {return j2_};
00096   int pi2() const {return pi2_};
00101   double e2() const {return e2_};
00105   double m1() const {return m1_};
00109   double m2() const {return m2_};
00113   int z1() const {return z1_};
00117   int z2() const {return z2_};
00121   double entranceSepE() const {return entranceSepE_};
00125   double sepE() const {return sepE_};
00129   int j3() const {return j3_};
00133   int pi3() const {return pi3_};
00137   double e3() const {return e3_};
00142   int pType() const {return pType_};
00146   double chRad() const {return chRad_};
00150   double g1() const {return g1_};
00154   double g2() const {return g2_};
00158   double ecMultMask() const {return ecMultMask_};
00159 private:
00160   double levelJ_;
00161   int levelPi_;
00162   double levelE_;
00163   int levelFix_;
00164   int aa_;
00165   int ir_;
00166   double s_;
00167   int l_;
00168   int levelID_;
00169   int isActive_;
00170   int channelFix_;
00171   double gamma_;
00172   double j1_;
00173   int pi1_;
00174   double j2_;
00175   int pi2_;
00176   double e2_;
00177   double m1_;
00178   double m2_;
00179   int z1_;
00180   int z2_;
00181   double entranceSepE_;
00182   double sepE_;
00183   int j3_;
00184   int pi3_;
00185   double e3_;
00186   int pType_;
00187   double chRad_;
00188   double g1_;
00189   double g2_;
00190   unsigned int ecMultMask_;
00191 };
00192
00193 #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.](#)

```
00001 #ifndef PPAIR_H
00002 #define PPAIR_H
00003
00004 #include "Decay.h"
00005
00006 class NucLine;
00007
00008
00009
00016 class PPair {
00017 public:
00018     PPair(NucLine);
00019     bool IsEntrance() const;
00020     int GetZ(int) const;
00021     int GetPi(int) const;
00022     int GetPType() const;
00023     int NumDecays() const;
00024     int IsDecay(Decay);
00025     int IsDecay(int);
00026     int GetPairKey() const;
00027     double GetM(int) const;
00028     double GetG(int) const;
00029     double GetJ(int) const;
00030     double GetExE() const;
00031     double GetSepE() const;
00032     double GetChRad() const;
00033     double GetRedMass() const;
00034     double GetIli2Factor() const;
00035     void AddDecay(Decay);
00036     void SetEntrance();
00037     Decay *GetDecay(int);
00038 private:
00039     bool entrance_;
00040     bool ec_entrance_;
00041     int pair_z_[2];
00042     int pair_pi_[2];
00043     int pair_ptype_;
00044     int pair_key_;
00045     double pair_m_[2];
00046     double pair_g_[2];
00047     double pair_j_[2];
00048     double pair_ex_e_;
00049     double pair_sep_e_;
00050     double pair_ch_rad_;
00051     double red_mass_;
00052     double ili2factor_;
00053     std::vector<Decay> decays_;
00054 };
00055
00056 #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.](#)

```
00001 #ifndef REACTIONRATE_H
00002 #define REACTIONRATE_H
00003
00004 #include "Constants.h"
00005
00006 class CNuc;
00007
00008 extern double gsl_reactionrate_integration(double, CNuc*, const Config&, int, int);
00009
00010
00011
00016 class RateData {
00017 public:
00019 RateData(double t, double r) :
00020     temperature(t), rate(r) {};
00022 bool operator<(const RateData& right) const {
00023     return temperature < right.temperature;
00024 };
00026 double temperature;
00028 double rate;
00029 };
00030
00031
00032
00038 class ReactionRate {
00039 public:
00044 ReactionRate(CNuc*, const vector_r&, const Config &, int, int);
00048 CNuc *compound() const {return compound_};
00052 const Config &configure() const {return configure_};
00056 int entranceKey() const {return entrance_key_};
00060 int exitKey() const {return exit_key_};
00061 void CalculateRates();
00062 void CalculateFileRates();
00066 void WriteRates();
00067 private:
00068 int entrance_key_;
00069 int exit_key_;
00070 CNuc *compound_;
00071 const Config &configure_;
00072 std::vector<RateData> rates_;
00073 };
00074
00075
00076
00077 #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.](#)

```
00001 #ifndef RMATRIXFUNC_H
00002 #define RMATRIXFUNC_H
00003
00004 #include "GenMatrixFunc.h"
00005
00007
00014 class RMatrixFunc : public GenMatrixFunc {
00015 public:
00016     RMatrixFunc(CNuc*, const Config&);
00020     CNuc *compound() const {return compound_;;}
00021     const Config &configure() const {return configure_;;}
00022
00023     void ClearMatrices();
00024     void FillMatrices(EPoint*);
00025     void InvertMatrices();
00026     void CalculateTMatrix(EPoint*);
00030     void CalculateCrossSection();
00031
00032     complex GetRMatrixElement(int, int, int) const;
00033     complex GetRLMatrixElement(int, int, int) const;
00034     complex GetRLInvMatrixElement(int, int, int) const;
00035     complex GetRLInvRMatrixElement(int, int, int) const;
00036     matrix_c *GetJSpecRLMatrix(int);
00037     void AddRMatrixElement(int, int, int, complex);
00038     void AddRLMatrixElement(int, int, int, complex);
00039     void AddRLInvMatrix(matrix_c);
00040     void AddRLInvRMatrixElement(int, int, int, complex);
00041 private:
00042     CNuc *compound_;
00043     const Config& configure_;
00044     vector_matrix_c r_matrices_;
00045     vector_matrix_c rl_matrices_;
00046     vector_matrix_c rl_inv_matrices_;
00047     vector_matrix_c rl_inv_r_matrices_;
00048 };
00049
00050 #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.](#)

```

00001 #ifndef SEGLINE_H
00002 #define SEGLINE_H
00003
00004 #include <iostream>
00005 #include <string>
00006
00007
00008
00013 class SegLine {
00014 public:
00018     SegLine(std::istream &stream) {
00019         stream » isActive_ » entranceKey_ » exitKey_ » minE_ » maxE_ » minA_ » maxA_ » isDiff_;
00020         if(isDiff_==2) stream » phaseJ_ » phaseL_;
00021         stream » dataNorm_ » varyNorm_ » dataNormError_;
00022         std::string dummyString;
00023         getline(stream,dummyString);
00024         int p2 = dummyString.find_last_not_of(" \n\t\r");
00025         if (p2 != std::string::npos) {
00026             int p1 = dummyString.find_first_not_of(" \n\t\r");
00027             if (p1 == std::string::npos) p1 = 0;
00028             dataFile_=dummyString.substr(p1, (p2-p1)+1);
00029         } else dataFile_=std::string();
00030     };
00034     int isActive() const {return isActive_};
00039     int entranceKey() const {return entranceKey_};
00044     int exitKey() const {return exitKey_};
00048     double minE() const {return minE_};
00052     double maxE() const {return maxE_};
00056     double minA() const {return minA_};
00060     double maxA() const {return maxA_};
00065     int isDiff() const {return isDiff_};
00069     std::string dataFile() const {return dataFile_};
00073     double dataNorm() const {return dataNorm_};
00077     double dataNormError() const {return dataNormError_};
00081     int varyNorm() const {return varyNorm_};
00085     double phaseJ() const {return phaseJ_};
00090     int phaseL() const {return phaseL_};
00091 private:
00092     int isActive_;
00093     int entranceKey_;
00094     int exitKey_;
00095     double minE_;
00096     double maxE_;
00097     double minA_;
00098     double maxA_;
00099     int isDiff_;
00100     std::string dataFile_;
00101     double dataNorm_;
00102     double dataNormError_;
00103     int varyNorm_;
00104     double phaseJ_;
00105     int phaseL_;
00106 };
00107
00108 #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.](#)

```

00001 #ifndef SHIFTFUNC_H
00002 #define SHIFTFUNC_H
00003
00004 #include "PPair.h"
00005 #include "WhitFunc.h"
00006
00007 class ShftFunc;
00008
00010
00018 class ShftFunc {
00019 public:
00023     ShftFunc(PPair* pPair) {
00024         totalSepE_=pPair->GetSepE()+pPair->GetExE();
00025         radius_=(double) pPair->GetChRad();
00026         params_.whitFunc= new WhitFunc(pPair);
00027     };
00031     ~ShftFunc() {
00032         delete params_.whitFunc;
00033     };
00039     double operator()(int l,double energy);
00044     double EnergyDerivative(int l,double energy);
00045 private:
00046     double totalSepE() const {return totalSepE_;};
00047     double radius() const {return radius_;};
00048     static double thisShftFunc(double,void*);
00049     static double theWhitFunc(double,void*);
00050     typedef struct Params {
00051         int lValue;
00052         double bindingEnergy;
00053         WhitFunc *whitFunc;
00054     } Params;
00055     Params params_;
00056     double totalSepE_;
00057     double radius_;
00058 };
00059
00060 #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.](#)

```

00001 #ifndef TARGETEFFECT_H
00002 #define TARGETEFFECT_H
00003
00004 #include <string>
00005 #include <fstream>

```

```

00006 #include <vector>
00007 #include "Constants.h"
00008 #include "Equation.h"
00009
00011
00019 class TargetEffect {
00020 public:
00021     TargetEffect(std::istream &, const Config&);
00022     bool IsActive() const;
00023     bool IsConvolution() const;
00024     bool IsTargetIntegration() const;
00025     bool IsQCoefficients() const;
00026     int NumSubPoints() const;
00027     int NumQCoefficients() const;
00028     double GetSigma() const;
00029     double GetDensity() const;
00030     double TargetThickness(double, const Config&);
00031     double GetConvolutionFactor(double, double) const;
00032     double GetQCoefficient(int) const;
00033     void SetSigma(double);
00034     void SetNumSubPoints(int);
00035     std::vector<int> GetSegmentsList() const;
00036     Equation *GetStoppingPowerEq();
00038     static constexpr double convolutionRange=3.;
00039 private:
00040     bool isConvolution_;
00041     bool isTargetIntegration_;
00042     bool isActive_;
00043     bool isQCoefficients_;
00044     int numIntegrationPoints_;
00045     double sigma_;
00046     double density_;
00047     Equation stoppingPowerEq_;
00048     std::string segmentsList_;
00049     vector_r qCoefficients_;
00050 };
00051
00052 #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.](#)

```
00001 #ifndef WHITFUNC_H
00002 #define WHITFUNC_H
00003
00004 #include "PPair.h"
00005 #include <gsl/gsl_sf_hyperg.h>
00006 #include "Constants.h"
00007
00008 extern double gsl_whit_function(int,double,double,double,int,int);
00009
00010
00011
00016 class WhitFunc {
00017 public:
00021 WhitFunc(PPair *pPair) {
00022     z1_=pPair->GetZ(1);
00023     z2_=pPair->GetZ(2);
00024     redmass_=(double)pPair->GetRedMass();
00025 };
00029 int z1() const {
00030     return z1_;
00031 };
00035 int z2() const {
00036     return z2_;
00037 };
00041 double redmass() const {
00042     return redmass_;
00043 };
00049 double operator()(int l, double radius, double energy) const {
00050     const double k=-sqrt(uconv/2.)*fstruc*z1()*z2()*sqrt(redmass()/energy);
00051     const double m=1+0.5;
00052     const double z=2.0*sqrt(2.0*uconv)/hbarc*radius*sqrt(redmass()*energy);
00053
00054     const double a=m-k+0.5;
00055     const double b=1.0+2.0*m;
00056
00057     return exp(-z/2.0)*pow(z,m+0.50)*gsl_sf_hyperg_U(a,b,z);
00058 };
00059 private:
00060     int z1_;
00061     int z2_;
00062     double redmass_;
00063 };
00064
00065 #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.](#)

```
00001 #include "AChannel.h"
00002 #include "NucLine.h"
00003 #include <math.h>
```



```

00004 #include <assert.h>
00005
00006
00011 AChannel::AChannel(NucLine nucLine, int pairNum) {
00012     l_=nucLine.l();
00013     s_=nucLine.s();
00014     pair_=pairNum;
00015     if(nucLine.pType()==0) {
00016         radtype_='P';
00017     } else if(nucLine.pType()==10) {
00018         if(nucLine.levelPi()*nucLine.pi2()==pow(-1,nucLine.l())) radtype_='E';
00019         else radtype_='M';
00020     } else if(nucLine.pType()==20) {
00021         if(nucLine.l()==0) radtype_='F';
00022         else radtype_='G';
00023     }
00024 }
00025
00030 AChannel::AChannel(int lValue, double sValue, int pairNum, char radType) {
00031     l_=lValue;
00032     s_=sValue;
00033     pair_=pairNum;
00034     radtype_=radType;
00035 };
00036
00041 int AChannel::GetPairNum() const {
00042     return pair_;
00043 }
00044
00049 int AChannel::GetL() const {
00050     return l_;
00051 }
00052
00057 double AChannel::GetS() const {
00058     return s_;
00059 }
00060
00065 double AChannel::GetBoundaryCondition() const {
00066     return boundary_condition_;
00067 }
00068
00076 char AChannel::GetRadType() const {
00077     return radtype_;
00078 }
00079
00083 void AChannel::SetBoundaryCondition(double boundaryCondition) {
00084     boundary_condition_=boundaryCondition;
00085 }
00086

```

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.](#)

```

00001 #include "ALevel.h"
00002 #include "NucLine.h"
00003
00008 ALevel::ALevel(NucLine nucLine) :
00009     level_e_(nucLine.levelE()), fitlevel_e_(0.0), isinrmatrix_(true), sqrt_nfactor_(1.0),
00010     isECLLevel_(false),
00011     ecMultMask_(0), ecPairNum_(0) {
00012     if(nucLine.levelFix()==1) energyfixed_=true;
00013     else energyfixed_=false;
00014 }
00019 ALevel::ALevel(double energy) :

```

```

00020    energyfixed_(true), level_e_(energy), fitlevel_e_(0.0), isinrmatrix_(false), sqrt_nf_factor_(1.0),
        isECLevel_(false),
00021    ecMultMask_(0), ecPairNum_(0) {};;
00022
00027 bool ALevel::EnergyFixed() const {
00028     return energyfixed_;
00029 }
00030
00036 bool ALevel::IsInRMatrix() const {
00037     return isinrmatrix_;
00038 }
00039
00045 bool ALevel::ChannelFixed(int channelNum) const {
00046     return channelfixed_[channelNum-1];
00047 }
00048
00053 bool ALevel::IsECLevel() const {
00054     return isECLevel_;
00055 }
00056
00061 int ALevel::NumNFIntegrals() const {
00062     return nf_integrals_.size();
00063 }
00064
00069 int ALevel::GetTransformIterations() const {
00070     return transform_iter_;
00071 }
00072
00077 int ALevel::GetECPairNum() const {
00078     return ecPairNum_;
00079 }
00080
00085 unsigned char ALevel::GetECMultMask() const {
00086     return ecMultMask_;
00087 }
00088
00093 double ALevel::GetE() const {
00094     return level_e_;
00095 }
00096
00101 double ALevel::GetGamma(int channelNum) const {
00102     return gammas_[channelNum-1];
00103 }
00104
00109 double ALevel::GetFitGamma(int channelNum) const {
00110     return fitgammas_[channelNum-1];
00111 }
00112
00117 double ALevel::GetFitE() const {
00118     return fitlevel_e_;
00119 }
00120
00125 double ALevel::GetNFIntegral(int channelNum) const {
00126     return nf_integrals_[channelNum-1];
00127 }
00128
00133 double ALevel::GetSqrtNFFactor() const {
00134     return sqrt_nf_factor_;
00135 }
00136
00142 double ALevel::GetECConversionFactor(int channelNum) const {
00143     return ec_conv_factors_[channelNum-1];
00144 }
00145
00150 double ALevel::GetTransformGamma(int channelNum) const {
00151     return transform_gammas_[channelNum-1];
00152 }
00153
00158 double ALevel::GetTransformE() const {
00159     return transform_e_;
00160 }
00161
00166 double ALevel::GetBigGamma(int channelNum) const {
00167     return big_gammas_[channelNum-1];
00168 }
00169
00174 double ALevel::GetShiftFunction(int channelNum) const {
00175     return shifts_[channelNum-1];
00176 }
00177
00182 complex ALevel::GetExternalGamma(int channelNum) const {
00183     return external_gammas_[channelNum-1];
00184 }
00185
00191 void ALevel::AddGamma(NucLine nucLine) {
00192     double b=nucLine.gamma();
00193     gammas_.push_back(b);

```

```

00194     fitgammas_.push_back(0.0);
00195     transform_gammas_.push_back(0.0);
00196     big_gammas_.push_back(0.0);
00197     external_gammas_.push_back(complex(0.0,0.0));
00198     if(nucLine.channelFix()==1) channelfixed_.push_back(true);
00199     else channelfixed_.push_back(false);
00200     shifts_.push_back(0.0);
00201 }
00202
00208 void ALevel::AddGamma(double reducedWidth) {
00209     gammas_.push_back(reducedWidth);
00210     fitgammas_.push_back(0.0);
00211     transform_gammas_.push_back(0.0);
00212     big_gammas_.push_back(0.0);
00213     external_gammas_.push_back(complex(0.0,0.0));
00214     channelfixed_.push_back(false);
00215 }
00216
00221 void ALevel::SetGamma(int channelNum, double reducedWidth) {
00222     gammas_[channelNum-1]=reducedWidth;
00223 }
00224
00229 void ALevel::SetE(double energy) {
00230     level_e_=energy;
00231 }
00232
00237 void ALevel::SetFitGamma(int channelNum,double reducedWidth) {
00238     fitgammas_[channelNum-1]=reducedWidth;
00239 }
00240
00245 void ALevel::SetFitE(double energy) {
00246     fitlevel_e_=energy;
00247 }
00248
00255 void ALevel::AddNFIintegral(double integral) {
00256     nf_integrals_.push_back(integral);
00257 }
00258
00263 void ALevel::SetSqrtNFFactor(double term) {
00264     sqrt_nf_factor_=term;
00265 }
00266
00271 void ALevel::AddECConversionFactor(double conversionFactor) {
00272     ec_conv_factors_.push_back(conversionFactor);
00273 }
00274
00279 void ALevel::SetTransformGamma(int channelNum,double reducedWidth) {
00280     transform_gammas_[channelNum-1]=reducedWidth;
00281 }
00282
00287 void ALevel::SetTransformE(double energy) {
00288     transform_e_=energy;
00289 }
00290
00295 void ALevel::SetBigGamma(int channelNum, double partialWidth) {
00296     big_gammas_[channelNum-1]=partialWidth;
00297 }
00298
00303 void ALevel::SetTransformIterations(int iterations) {
00304     transform_iter_=iterations;
00305 }
00306
00311 void ALevel::SetExternalGamma(int channelNum, complex reducedWidth) {
00312     external_gammas_[channelNum-1]=reducedWidth;
00313 }
00314
00319 void ALevel::SetShiftFunction(int channelNum, double shiftFunction) {
00320     shifts_[channelNum-1]=shiftFunction;
00321 }
00322
00327 void ALevel::SetECParams(int pairNum, unsigned char multMask) {
00328     isEClevel_=true;
00329     ecPairNum_=pairNum;
00330     ecMultMask_=multMask;
00331 }

```

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.](#)

```
00001 #include "AMatrixFunc.h"
00002 #include "CNuc.h"
00003 #include "Config.h"
00004 #include "EPoint.h"
00005 #include "MatrixInv.h"
00006 #include <assert.h>
00007
00012 AMatrixFunc::AMatrixFunc(CNuc* compound, const Config &configure) :
00013     compound_(compound), configure_(configure) {}
00014
00019 complex AMatrixFunc::GetAMatrixElement(int jGroupNum, int lambdaNum, int muNum) const {
00020     return a_matrices_[jGroupNum-1][lambdaNum-1][muNum-1];
00021 }
00022
00027 matrix_c *AMatrixFunc::GetJSpecAInvMatrix(int jGroupNum) {
00028     matrix_c *b=&a_inv_matrices_[jGroupNum-1];
00029     return b;
00030 }
00031
00036 void AMatrixFunc::ClearMatrices() {
00037     a_inv_matrices_.clear();
00038     a_matrices_.clear();
00039     tmatrix_.clear();
00040     ec_tmatrix_.clear();
00041 }
00042
00047 void AMatrixFunc::FillMatrices(EPoint *point) {
00048     double inEnergy;
00049     if(compound()->
00050         GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->
00051         GetPType()==20)
00052         inEnergy=point->GetCMEnergy()+
00053         compound()->
00054         GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->
00055         GetSepE()+
00056         compound()->
00057         GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->
00058         GetExE();
00059     else inEnergy = point->GetCMEnergy()+
00060         compound()->GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->GetSepE()+
00061         compound()->GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->GetExE();
00062     for(int j=1;j<=compound()->NumJGroups();j++) {
00063         if(compound()->GetJGroup(j)->IsInRMatrix()) {
00064             for(int la=1;la<=compound()->GetJGroup(j)->NumLevels();la++) {
00065                 if(compound()->GetJGroup(j)->GetLevel(la)->IsInRMatrix()) {
00066                     ALevel *level=compound()->GetJGroup(j)->GetLevel(la);
00067                     for(int lap=1;lap<=compound()->GetJGroup(j)->NumLevels();lap++) {
00068                         if(compound()->GetJGroup(j)->GetLevel(lap)->IsInRMatrix()) {
00069                             ALevel *levelp=compound()->GetJGroup(j)->GetLevel(lap);
00070                             complex sum(0.0,0.0);
00071                             for(int ch=1;ch<=compound()->GetJGroup(j)->NumChannels();ch++) {
00072                                 double gammaCh=level->GetFitGamma(ch);
00073                                 double gammaChp=levelp->GetFitGamma(ch);
00074                                 complex loElement=point->GetLoElement(j,ch);
00075                                 sum+=gammaCh*gammaChp*loElement;
00076                                 if((compound()->GetJGroup(j)->GetChannel(ch)->GetRadType() == 'M' ||
00077                                     compound()->GetJGroup(j)->GetChannel(ch)->GetRadType() == 'E') &&
00078                                     la==lap &&
00079                                     (configure().paramMask & Config::USE_RMC_FORMALISM))
00080                                     sum+=complex(0.0,1.0)*gammaCh*gammaChp;
00081                                 if((configure().paramMask & Config::USE_BRUNE_FORMALISM) &&
00082                                     compound()->GetJGroup(j)->GetChannel(ch)->GetRadType()=='P') {
00083                                     sum+=gammaCh*gammaChp*compound()->GetJGroup(j)->GetChannel(ch)->GetBoundaryCondition();
00084                                     if(la==lap) sum-=gammaCh*gammaChp*level->GetShiftFunction(ch);
00085                                     else sum-=gammaCh*gammaChp*
00086                                     (level->GetShiftFunction(ch)*(inEnergy-levelp->GetFitE())-levelp->GetShiftFunction(ch)*(inEnergy-level->GetFitE()))/
00087                                     (level->GetFitE()-levelp->GetFitE());
00088                                 }
00089                             }
00090                         }
00091                     }
00092                 }
00093             }
00094         }
00095     }
00096 }
```

```

00089         if(la==lap) {
00090             double resenergy=level->GetFitE();
00091             this->AddInvMatrixElement(j, la, lap, resenergy-inEnergy-sum);
00092         } else this->AddInvMatrixElement(j, la, lap, -sum);
00093     }
00094 }
00095 }
00096 }
00097 }
00098 }
00099 }
00100
00105 void AMatrixFunc::InvertMatrices() {
00106     for(int j=1; j<=compound()->NumJGroups(); j++) {
00107         if(compound()->GetJGroup(j)->IsInRMatrix()) {
00108             matrix_c *theAInvMatrix = this->GetJSpecAInvMatrix(j);
00109             MatrixInv matrixInv(*theAInvMatrix);
00110             this->AddAMatrix(matrixInv.inverse());
00111         }
00112     }
00113 }
00114
00119 void AMatrixFunc::CalculateTMatrix(EPoint *point) {
00120     int aa=compound()->GetPairNumFromKey(point->GetEntranceKey());
00121     int irEnd;
00122     int irStart;
00123     bool isRMC=false;
00124     if((configure().paramMask & Config::USE_RMC_FORMALISM) &&
00125        compound()->GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->GetPType()==10) {
00126         irStart=1;
00127         irEnd=compound()->GetPair(aa)->NumDecays();
00128         isRMC=true;
00129     } else {
00130         irStart=0;
00131         while(irStart<compound()->GetPair(aa)->NumDecays()) {
00132             irStart++;
00133             if(compound()->GetPair(aa)->GetDecay(irStart)->GetPairNum()==
00134                compound()->GetPairNumFromKey(point->GetExitKey())) break;
00135         }
00136         irEnd=irStart;
00137     }
00138     for(int ir=irStart; ir<=irEnd; ir++) {
00139         Decay *theDecay=compound()->GetPair(aa)->GetDecay(ir);
00140         for(int k=1; k<=theDecay->NumKGroups(); k++) {
00141             for(int m=1; m<=theDecay->GetKGroup(k)->NumMGroups(); m++) {
00142                 MGroup *theMGroup=theDecay->GetKGroup(k)->GetMGroup(m);
00143                 JGroup *theJGroup=compound()->GetJGroup(theMGroup->GetJNum());
00144                 AChannel *entranceChannel=theJGroup->GetChannel(theMGroup->GetChNum());
00145                 AChannel *exitChannel=theJGroup->GetChannel(theMGroup->GetChpNum());
00146                 complex uphase=point->GetExpCoulombPhase(theMGroup->GetJNum(), theMGroup->GetChNum()) *
00147                    point->GetExpHardSpherePhase(theMGroup->GetJNum(), theMGroup->GetChNum()) *
00148                    point->GetExpCoulombPhase(theMGroup->GetJNum(), theMGroup->GetChpNum()) *
00149                    point->GetExpHardSpherePhase(theMGroup->GetJNum(), theMGroup->GetChpNum());
00150                 complex umatrix(0., 0.);
00151                 for(int la=1; la<=compound()->GetJGroup(theMGroup->GetJNum())->NumLevels(); la++) {
00152                     if(compound()->GetJGroup(theMGroup->GetJNum())->GetLevel(la)->IsInRMatrix()) {
00153                         ALevel *level=compound()->GetJGroup(theMGroup->GetJNum())->GetLevel(la);
00154                         for(int lap=1; lap<=compound()->GetJGroup(theMGroup->GetJNum())->NumLevels(); lap++) {
00155                             if(compound()->GetJGroup(theMGroup->GetJNum())->GetLevel(lap)->IsInRMatrix()) {
00156                                 ALevel *levelp=compound()->GetJGroup(theMGroup->GetJNum())->GetLevel(lap);
00157                                 umatrix+=2.0*complex(0.0, 1.0) *
00158                                    point->GetSqrtPenetrability(theMGroup->GetJNum(), theMGroup->GetChNum()) *
00159                                    point->GetSqrtPenetrability(theMGroup->GetJNum(), theMGroup->GetChpNum()) *
00160                                    level->GetFitGamma(theMGroup->GetChNum()) *
00161                                    levelp->GetFitGamma(theMGroup->GetChpNum()) *
00162                                    this->GetAMatrixElement(theMGroup->GetJNum(), la, lap);
00163                             }
00164                         }
00165                     }
00166                 }
00167                 complex tphase=point->GetExpCoulombPhase(theMGroup->GetJNum(), theMGroup->GetChNum()) *
00168                    point->GetExpCoulombPhase(theMGroup->GetJNum(), theMGroup->GetChpNum());
00169                 complex tmatrix;
00170                 if(isRMC) this->AddTMatrixElement(k, m, complex(0.0, -1.0)*umatrix, ir);
00171                 else {
00172                     if(theMGroup->GetChNum()==theMGroup->GetChpNum()) {
00173                         tmatrix=tphase-uphase*(1.0+umatrix);
00174                     } else tmatrix=-uphase*umatrix;
00175                     this->AddTMatrixElement(k, m, tmatrix);
00176                 }
00177             }
00178             for(int m=1; m<=theDecay->GetKGroup(k)->NumECMGroups(); m++) {
00179                 ECMGroup *theECMGroup=theDecay->GetKGroup(k)->GetECMGroup(m);
00180                 ALevel *finalLevel=compound()->GetJGroup(theECMGroup->GetJGroupNum())
00181                    ->GetLevel(theECMGroup->GetLevelNum());
00182                 double ecNormParam=finalLevel->GetFitGamma(theECMGroup->GetFinalChannel()) *
00183                    finalLevel->GetSqrtNFFactor()*finalLevel->GetECConversionFactor(theECMGroup->GetFinalChannel());

```

```

00184     complex tmatrix=ecNormParam*point->GetECAmplitude(k,m);
00185     if(theECMGroup->IsChannelCapture()) {
00186         int internalChannel=theECMGroup->GetIntChannelNum();
00187         MGroup *chanMGroup=compound()->GetPair(aa)->GetDecay(theECMGroup->GetChanCapDecay())
00188             ->GetKGroup(theECMGroup->GetChanCapKGroup())->GetMGroup(theECMGroup->GetChanCapMGroup());
00189         AChannel *chanEntranceChannel=compound()->GetJGroup(chanMGroup->GetJNum())
00190             ->GetChannel(chanMGroup->GetChNum());
00191         AChannel *chanExitChannel=compound()->GetJGroup(chanMGroup->GetJNum())
00192             ->GetChannel(chanMGroup->GetChpNum());
00193         complex umatrix(0.,0.);
00194         for(int la=1;la<=compound()->GetJGroup(chanMGroup->GetJNum())->NumLevels();la++) {
00195             if(compound()->GetJGroup(chanMGroup->GetJNum())->GetLevel(la)->IsInRMatrix()) {
00196                 ALevel *level=compound()->GetJGroup(chanMGroup->GetJNum())->GetLevel(la);
00197                 if(internalChannel && (configure().paramMask & Config::IGNORE_ZERO_WIDTHS))
00198                     if(fabs(level->GetFitGamma(internalChannel))<1.0e-8) continue;
00199                 for(int lap=1;lap<=compound()->GetJGroup(chanMGroup->GetJNum())->NumLevels();lap++) {
00200                     if(compound()->GetJGroup(chanMGroup->GetJNum())->GetLevel(lap)->IsInRMatrix()) {
00201                         ALevel *levelp=compound()->GetJGroup(chanMGroup->GetJNum())->GetLevel(lap);
00202                         if(internalChannel && (configure().paramMask & Config::IGNORE_ZERO_WIDTHS))
00203                             if(fabs(levelp->GetFitGamma(internalChannel))<1.0e-8) continue;
00204                         umatrix+=2.0*complex(0.0,1.0)*
00205                             point->GetSqrtPenetrability(chanMGroup->GetJNum(),chanMGroup->GetChNum())*
00206                             level->GetFitGamma(chanMGroup->GetChNum())*
00207                             levelp->GetFitGamma(chanMGroup->GetChpNum())*
00208                             this->GetAMatrixElement(chanMGroup->GetJNum(),la,lap);
00209                     }
00210                 }
00211             }
00212         }
00213         tmatrix=tmatrix*umatrix;
00214     }
00215     this->AddECTMatrixElement(k,m,tmatrix);
00216 }
00217 }
00218 }
00219 }
00220
00225 void AMatrixFunc::AddAInvMatrixElement(int jGroupNum, int lambdaNum, int muNum, complex
    aMatrixElement) {
00226     matrix_c e;
00227     vector_c f;
00228     while(jGroupNum>a_inv_matrices_.size()) a_inv_matrices_.push_back(e);
00229     while(lambdaNum>a_inv_matrices_[jGroupNum-1].size()) a_inv_matrices_[jGroupNum-1].push_back(f);
00230     a_inv_matrices_[jGroupNum-1][lambdaNum-1].push_back(aMatrixElement);
00231     assert(muNum==a_inv_matrices_[jGroupNum-1][lambdaNum-1].size());
00232 }
00233
00238 void AMatrixFunc::AddAMatrix(matrix_c aMatrix) {
00239     a_matrices_.push_back(aMatrix);
00240 }

```

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.

```

00001 #include "AngCoeff.h"
00002 #include <gsl/gsl_sf_coupling.h>
00003 #include <math.h>
00004
00005 double AngCoeff::ClebGord(double j1, double j2, double j3, double m1, double m2, double m3) {
00006     m3=-m3;
00007     int j1x2=(int)(2*j1);
00008     int j2x2=(int)(2*j2);
00009     int j3x2=(int)(2*j3);
00010     int m1x2=(int)(2*m1);

```

```

00011     int m2x2=(int) (2*m2);
00012     int m3x2=(int) (2*m3);
00013
00014     double w3j=gsl_sf_coupling_3j(j1x2, j2x2, j3x2, m1x2, m2x2, m3x2);
00015
00016     return pow(-1.0, j1-j2-m3)*sqrt(2.0*j3+1.)*w3j;
00017 }
00018
00019 double AngCoeff::Racah(double j1, double j2, double l2, double l1, double j3, double l3) {
00020
00021     int j1x2=(int) (2*j1);
00022     int j2x2=(int) (2*j2);
00023     int j3x2=(int) (2*j3);
00024     int l1x2=(int) (2*l1);
00025     int l2x2=(int) (2*l2);
00026     int l3x2=(int) (2*l3);
00027
00028     double w6j=gsl_sf_coupling_6j(j1x2, j2x2, j3x2, l1x2, l2x2, l3x2);
00029
00030     return pow(-1.0, j1+j2+l2+l1)*w6j;
00031 }

```

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 "GSLError.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 prompts 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](#) object 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.](#)

```

00001 /*
00002  * AZURE2.cpp
00003  * Ethan Uberseder, University of Notre Dame, 2011
00004  *
00005  * This file contains the main() function, as well as other C functions used to
00006  * initialize the program. The functions defined here are (mostly) procedural
00007  * C, and used to initialize the AZUREMain object from command line input. All
00008  * actual calculations are managed by the AZUREMain object, and (almost) all
00009  * subsequent routines/classes follow object-oriented C++ programming conventions.
00010  */
00011
00012
00013 #include "AZUREMain.h"
00014 #include "Config.h"
00015 #include "NucLine.h"
00016 #include "SegLine.h"
00017 #include "ExtrapLine.h"
00018 #include "GSLException.h"
00019 #include <stdlib.h>
00020 #include <iostream>
00021 #include <iomanip>
00022 #include <fstream>
00023 #include <vector>
00024 #include <gsl/gsl_errno.h>
00025 #ifndef NO_READLINE
00026 #include <readline/readline.h>
00027 #include <readline/history.h>
00028 #endif
00029 #include <string.h>
00030
00031 #ifdef GUI_BUILD
00032 extern int start_gui(int argc, char *argv[]);
00033 #endif
00034 struct SegPairs {int firstPair; int secondPair;};
00035
00040 void welcomeMessage(const Config& configure) {
00041     configure.outStream << std::endl
00042         << "O-----O" << std::endl
00043         << "| ### ## | Version 1.0 |" << std::endl
00044         << "| # # # # # |" << std::endl
00045         << "| ### ## |" << std::endl
00046         << "| # # # # # | Joint Institute for |" << std::endl
00047         << "| # # ### ## # # ### ## | Nuclear Astrophysics |" << std::endl
00048         << "O-----O" << std::endl
00049         << std::endl;
00050 }
00051
00056 void exitMessage(const Config& configure) {
00057     configure.outStream << std::endl
00058         << "Thanks for using AZURE2." << std::endl;
00059 }
00060
00066 void printHelp() {
00067     std::cout << "Syntax: AZURE2 <options> configfile" << std::endl << std::endl
00068         << "Options:" << std::endl
00069         << std::setw(25) << std::left << "\t--no-gui:" << std::setw(0) << "Do not use graphical setup
utility (if built)." << std::endl
00070         << std::setw(25) << std::left << "\t" << std::setw(0) << "If this flag is not set all other
options are ignored," << std::endl
00071         << std::setw(25) << std::left << "\t" << std::setw(0) << "and configuration occurs within the
setup utility." << std::endl
00072 #ifndef NO_READLINE
00073         << std::setw(25) << std::left << "\t--no-readline:" << std::setw(0) << "Do not use readline
package." << std::endl
00074 #endif
00075         << std::setw(25) << std::left << "\t--no-transform:" << std::setw(0) << "Do not perform initial
parameter transformations." << std::endl
00076         << std::setw(25) << std::left << "\t--no-long-wavelength:" << std::setw(0) << "Do not use long
wavelength approximation for EL capture." << std::endl
00077         << std::setw(25) << std::left << "\t--use-brune:" << std::setw(0) << "Use the alternative level
matrix of C.R. Brune." << std::endl
00078         << std::setw(25) << std::left << "\t--ignore-externals:" << std::setw(0) << "Ignore external
resonant capture amplitude if internal width is zero." << std::endl
00079         << std::setw(25) << std::left << "\t--use-rmc:" << std::setw(0) << "Use Reich-Moore approximation
for capture (neutron capture only)." << std::endl
00080         << std::setw(25) << std::left << "\t--gsl-coul:" << std::setw(0) << "Use GSL Coulomb functions
(faster, but less accurate)." << std::endl;
00081 }
00082
00089 bool parseOptions(int argc, char *argv[], Config& configure) {
00090 #ifndef NO_READLINE
00091     bool useReadline=true;

```

```

00092 #else
00093     bool useReadline = false;
00094 #endif
00095     configure.configfile="";
00096     std::vector<std::string> options;
00097     for(int i=1;i<argc;i++) {
00098         std::string arg=argv[i];
00099         if(!arg.empty() && arg.substr(0,2)=="--") options.push_back(arg);
00100         else configure.configfile=arg;
00101     }
00102     char* optionsFile = getenv("AZURE_OPTIONS_FILE");
00103     if(optionsFile) {
00104         std::ifstream in(optionsFile);
00105         if(in) {
00106             while(!in.eof()) {
00107                 std::string newOption;
00108                 getline(in,newOption);
00109                 if(!in.eof()) {
00110                     std::string trimmedOption="";
00111                     for(std::string::iterator it = newOption.begin(); it<newOption.end(); it++)
00112                         if(*it!=' ' && *it!='\t' && *it!='\n')
00113                             trimmedOption+=*it;
00114                     if(!trimmedOption.empty() && trimmedOption.substr(0,2)=="--") options.push_back(trimmedOption);
00115                 }
00116             }
00117             in.close();
00118         } else configure.outStream << "AZURE_OPTIONS_FILE variable set, but file not readable." <<
std::endl;
00119     }
00120     for(std::vector<std::string>::iterator it = options.begin(); it<options.end(); it++) {
00121 #ifndef NO_READLINE
00122         if(*it=="--no-readline") useReadline=false;
00123         else if(*it=="--no-transform") configure.paramMask &= ~Config::TRANSFORM_PARAMETERS;
00124 #else
00125         if (*it == "--no-transform") configure.paramMask &= ~Config::TRANSFORM_PARAMETERS;
00126 #endif
00127         else if(*it=="--no-long-wavelength") configure.paramMask &= ~Config::USE_LONGWAVELENGTH_APPROX;
00128         else if(*it=="--use-brune") configure.paramMask |= Config::USE_BRUNE_FORMALISM;
00129         else if(*it=="--gsl-coul") configure.paramMask |= Config::USE_GSL_COULOMB_FUNC;
00130         else if(*it=="--ignore-externals") configure.paramMask |= Config::IGNORE_ZERO_WIDTHS;
00131         else if(*it=="--use-rcm") configure.paramMask |= Config::USE_RMC_FORMALISM;
00132         else if(*it=="--no-gui") continue;
00133         else configure.outStream << "WARNING: Unknown option " << *it << ' ' << std::endl;
00134     }
00135     return useReadline;
00136 }
00137
00143 int commandShell(const Config& configure) {
00144     int command=0;
00145
00146     configure.outStream << "Please select from the following options: " << std::endl
00147         << "\t1. Calculate Segments From Data" << std::endl
00148         << "\t2. Fit Segments From Data" << std::endl
00149         << "\t3. Calculate Segments Without Data" << std::endl
00150         << "\t4. Perform MINOS Error Analysis" << std::endl
00151         << "\t5. Calculate Reaction Rate" << std::endl
00152         << "\t6. Exit" << std::endl;
00153
00154     while(command<1||command>6) {
00155         configure.outStream << "azure2: ";
00156         std::string inString;
00157         getline(std::cin,inString);
00158         if(inString.empty()) continue;
00159         std::istringstream in;
00160         in.str(inString);
00161         if(!(in>command))
00162             configure.outStream << "Please enter an integer." << std::endl;
00163         else if(command<1||command>6)
00164             configure.outStream << "Invalid option. Please try again."
00165                 << std::endl;
00166     }
00167     return command;
00168 }
00169
00175 void processCommand(int command, Config& configure) {
00176     if(command==2) configure.paramMask |= Config::PERFORM_FIT;
00177     else if(command==3) configure.paramMask &= ~Config::CALCULATE_WITH_DATA;
00178     else if(command==4) {
00179         bool goodAnswer=false;
00180         while (!goodAnswer) {
00181             configure.outStream << std::setw(30) << "Allowed Chi-Squared Variance: ";
00182             std::string inString;
00183             getline(std::cin,inString);
00184             std::istringstream stm;
00185             stm.str(inString);
00186             if(!(stm>configure.chiVariance) || configure.chiVariance<0.)
00187                 configure.outStream << "Please enter a positive number." << std::endl;

```

```

00188         else goodAnswer=true;
00189     }
00190     configure.paramMask |= Config::PERFORM_FIT;
00191     configure.paramMask |= Config::PERFORM_ERROR_ANALYSIS;
00192 } else if(command==5) {
00193     configure.paramMask &= ~Config::CALCULATE_WITH_DATA;
00194     configure.paramMask |= Config::CALCULATE_REACTION_RATE;
00195 } else if(command==6) {
00196     exitMessage(configure);
00197     exit(0);
00198 }
00199 }
00200
00206 void getParameterFile(bool useReadline, Config& configure) {
00207     bool validInfile=false;
00208     configure.outStream << std::endl;
00209     if(!useReadline) configure.outStream << "External Parameter File (leave blank for new file): ";
00210     while(!validInfile) {
00211         std::string inFile;
00212         if(!useReadline) getline(std::cin,inFile);
00213 #ifndef NO_READLINE
00214         else {
00215             char *line = readline("External Parameter File (leave blank for new file): ");
00216             inFile=line;
00217             size_t endpos = inFile.find_last_not_of(" \t");
00218             if( std::string::npos != endpos ) inFile = inFile.substr( 0, endpos+1 );
00219             if(line && *line) add_history(line);
00220             free(line);
00221         }
00222 #endif
00223         if(!inFile.empty()) {
00224             std::ifstream in;
00225             in.open(inFile.c_str());
00226             if(in) {
00227                 validInfile=true;
00228                 configure.paramMask |= Config::USE_PREVIOUS_PARAMETERS;
00229                 configure.paramfile=inFile;
00230                 in.close();
00231             }
00232             in.clear();
00233         } else validInfile=true;
00234         if(!validInfile) {
00235             configure.outStream << "Cannot Read From " << inFile << ". Please reenter file." << std::endl;
00236             if(!useReadline) configure.outStream << "External Parameter File (leave blank for new file): ";
00237         }
00238     }
00239 }
00240
00247 bool readSegmentFile(const Config& configure, std::vector<SegPairs>& segPairs) {
00248     std::ifstream in;
00249     std::string startTag, stopTag;
00250     if(configure.paramMask & Config::CALCULATE_WITH_DATA) {
00251         startTag="<segmentsData>";
00252         stopTag="</segmentsData>";
00253     } else {
00254         startTag="<segmentsTest>";
00255         stopTag="</segmentsTest>";
00256     }
00257     in.open(configure.configfile.c_str());
00258     if(in) {
00259         std::string line="";
00260         while(line!=startTag&&!in.eof()) getline(in,line);
00261         if(line==startTag) {
00262             line="";
00263             while(line!=stopTag&&!in.eof()) {
00264                 getline(in,line);
00265                 bool empty=true;
00266                 for(unsigned int i=0;i<line.size();++i)
00267                     if(line[i]!=' ' && line[i]!='\t') {
00268                         empty=false;
00269                         break;
00270                     }
00271                 if(empty==true) continue;
00272                 if(line!=stopTag&&!in.eof()) {
00273                     std::stringstream stm;
00274                     stm.str(line);
00275                     if(configure.paramMask & Config::CALCULATE_WITH_DATA) {
00276                         SegLine segment(stm);
00277                         if(!(stm.rdstate() & (std::stringstream::failbit |
00278                             std::stringstream::badbit)) && segment.isActive()==1) {
00279                             SegPairs tempSet={segment.entranceKey(), segment.exitKey()};
00280                             segPairs.push_back(tempSet);
00281                         } else {
00282                             ExtrapolLine segment(stm);
00283                             if(!(stm.rdstate() & (std::stringstream::failbit |

```

```

00284         SegPairs tempSet={segment.entranceKey(),segment.exitKey()};
00285         segPairs.push_back(tempSet);
00286     }
00287 }
00288 }
00289 }
00290 if(line!=stopTag) {
00291     configure.outStream << "Problem reading segments. Check configuration file." << std::endl;
00292     return false;
00293 }
00294 } else {
00295     configure.outStream << "Problem reading segments. Check configuration file." << std::endl;
00296     return false;
00297 }
00298 in.close();
00299 } else {
00300     configure.outStream << "Cannot read segments. Check configuration file." << std::endl;
00301     return false;
00302 }
00303 in.clear();
00304 return true;
00305 }
00306
00312 void getTemperatureFile(bool useReadline, Config& configure) {
00313     bool validInfile=false;
00314     if(!useReadline) configure.outStream << std::setw(38) << "Temperature File Name: ";
00315     while(!validInfile) {
00316         std::string inFile;
00317         if(!useReadline) getline(std::cin,inFile);
00318 #ifndef NO_READLINE
00319         else {
00320             char *line = readline("Temperature File Name: ");
00321             inFile=line;
00322             size_t endpos = inFile.find_last_not_of(" \t");
00323             if( std::string::npos != endpos ) inFile = inFile.substr( 0, endpos+1 );
00324             if(line && *line) add_history(line);
00325             free(line);
00326         }
00327 #endif
00328         if(!inFile.empty()) {
00329             std::ifstream in;
00330             in.open(inFile.c_str());
00331             if(in) {
00332                 validInfile=true;
00333                 configure.rateParams.temperatureFile=inFile;
00334                 in.close();
00335             }
00336             in.clear();
00337         }
00338         if(!validInfile) {
00339             if(inFile.empty()) configure.outStream << "Please enter a file name." << std::endl;
00340             else configure.outStream << "Cannot Read From " << inFile << ". Please reenter file." << std::endl;
00341             if(!useReadline) configure.outStream << "Temperature File Name: ";
00342         }
00343     }
00344 }
00345
00351 void getRateParams(Config& configure, std::vector<SegPairs>& segPairs,bool useReadline) {
00352     configure.rateParams.entrancePair=0;
00353     configure.rateParams.exitPair=0;
00354     configure.rateParams.minTemp=-1.;
00355     configure.rateParams.maxTemp=-1.;
00356     configure.rateParams.tempStep=-1.;
00357     while(configure.rateParams.entrancePair==configure.rateParams.exitPair){
00358         while(!configure.rateParams.entrancePair) {
00359             configure.outStream << std::setw(38) << "Reaction Rate Entrance Pair: ";
00360             std::string inString;
00361             getline(std::cin,inString);
00362             std::istringstream stm;
00363             stm.str(inString);
00364             if(!(stm > configure.rateParams.entrancePair) || configure.rateParams.entrancePair==0)
00365                 configure.outStream << "Please enter an integer greater than zero." << std::endl ;
00366         }
00367         while(!configure.rateParams.exitPair) {
00368             configure.outStream << std::setw(38) << "Reaction Rate Exit Pair: ";
00369             std::string inString;
00370             getline(std::cin,inString);
00371             std::istringstream stm;
00372             stm.str(inString);
00373             if(!(stm > configure.rateParams.exitPair) || configure.rateParams.exitPair==0)
00374                 configure.outStream << "Please enter an integer greater than zero." << std::endl;
00375         }
00376         if(configure.rateParams.entrancePair==configure.rateParams.exitPair) {
00377             configure.outStream << "Cannot calculate rate for elastic scattering." << std::endl;
00378             configure.rateParams.entrancePair=0;configure.rateParams.exitPair=0;
00379         }
00380     }

```

```

00381 SegPairs tempSet={configure.rateParams.entrancePair,configure.rateParams.exitPair};
00382 segPairs.push_back(tempSet);
00383 bool goodAnswer = false;
00384 std::string fileAnswer="";
00385 while(!goodAnswer) {
00386     configure.outStream << std::setw(38) << "Use temperatures from file (yes/no): ";
00387     getline(std::cin,fileAnswer);
00388     std::string trimmedAnswer;
00389     for(int i =0;i<fileAnswer.length();i++)
00390         if(fileAnswer[i]!=' ' && fileAnswer[i]!='\t' && fileAnswer[i]!='\n')
00391             trimmedAnswer+=fileAnswer[i];
00392     if(trimmedAnswer!="yes" && trimmedAnswer!="no")
00393         configure.outStream << "Please type 'yes' or 'no'." << std::endl;
00394     else {
00395         goodAnswer = true;
00396         fileAnswer=trimmedAnswer;
00397     }
00398 }
00399 configure.rateParams.useFile = (fileAnswer=="yes") ? (true) : (false);
00400 if(configure.rateParams.useFile) getTemperatureFile(useReadline,configure);
00401 else {
00402     while(configure.rateParams.minTemp<0.) {
00403         configure.outStream << std::setw(38) << "Reaction Rate Min Temp [GK]: ";
00404         std::string inString;
00405         getline(std::cin,inString);
00406         std::istringstream stm;
00407         stm.str(inString);
00408         if(!(stm > configure.rateParams.minTemp) || configure.rateParams.minTemp<0.)
00409             configure.outStream << "Please enter a positive number." << std::endl;
00410     }
00411     while(configure.rateParams.maxTemp<0.) {
00412         configure.outStream << std::setw(38) << "Reaction Rate Max Temp [GK]: ";
00413         std::string inString;
00414         getline(std::cin,inString);
00415         std::istringstream stm;
00416         stm.str(inString);
00417         if(!(stm > configure.rateParams.maxTemp) || configure.rateParams.maxTemp<0.)
00418             configure.outStream << "Please enter a positive number." << std::endl;
00419     }
00420     while(configure.rateParams.tempStep<0.) {
00421         configure.outStream << std::setw(38) << "Reaction Rate Temp Step [GK]: ";
00422         std::string inString;
00423         getline(std::cin,inString);
00424         std::istringstream stm;
00425         stm.str(inString);
00426         if(!(stm > configure.rateParams.tempStep) || configure.rateParams.tempStep<0.)
00427             configure.outStream << "Please enter a positive number." << std::endl;
00428     }
00429 }
00430 }
00431
00438 bool checkExternalCapture(Config& configure, const std::vector<SegPairs>& segPairs) {
00439     std::ifstream in;
00440     in.open(configure.configfile.c_str());
00441     if(!in) {
00442         configure.outStream << "Cannot read nuclear information. Check configuration file." << std::endl;
00443         return false;
00444     }
00445     std::string line="";
00446     while(line!="<levels>" && !in.eof()) getline(in,line);
00447     if(line!="<levels>") {
00448         configure.outStream << "Cannot read nuclear information. Check configuration file." << std::endl;
00449         return false;
00450     }
00451     line="";
00452     while(line!="</levels>" && !in.eof() &&
00453           !(configure.paramMask & Config::USE_EXTERNAL_CAPTURE)) {
00454         getline(in,line);
00455         bool empty=true;
00456         for(unsigned int i=0;i<line.size();++i)
00457             if(line[i]!=' ' && line[i]!='\t') {
00458                 empty=false;
00459                 break;
00460             }
00461         if(empty==true) continue;
00462         if(line!="</levels>" && !in.eof()) {
00463             std::istringstream stm;
00464             stm.str(line);
00465             NucLine tempNucLine(stm);
00466             if((stm.rdstate() & (std::stringstream::failbit | std::stringstream::badbit))) {
00467                 configure.outStream << "Problem reading nuclear information. Check configuration file." <<
std::endl;
00468                 return false;
00469             }
00470             if(tempNucLine.ecMultMask()!=0) {
00471                 for(int i=0;i<segPairs.size();i++) {
00472                     if(tempNucLine.ir()==segPairs[i].secondPair||

```

```

00473         segPairs[i].secondPair==1) {
00474             configure.paramMask |= Config::USE_EXTERNAL_CAPTURE;
00475             break;
00476         }
00477     }
00478 }
00479 }
00480 }
00481 in.close();
00482 in.clear();
00483 if((configure.paramMask & Config::USE_EXTERNAL_CAPTURE)&&
00484     (configure.paramMask & Config::USE_RMC_FORMALISM)) {
00485     configure.outStream << "WARNING: External capture is not compatible with Reich-Moore. Ignoring
external capture."
00486         << std::endl;
00487     configure.paramMask &= ~Config::USE_EXTERNAL_CAPTURE;
00488 }
00489 return true;
00490 }
00491 }
00492 void getExternalCaptureFile(bool useReadline, Config& configure) {
00493     if((configure.paramMask & Config::USE_EXTERNAL_CAPTURE)&&
00494         !(configure.paramMask & Config::CALCULATE_REACTION_RATE)) {
00495         configure.outStream << std::endl;
00496         if(!useReadline) configure.outStream << "External Capture Amplitude File (leave blank for new
file): ";
00497         bool validInfile=false;
00498         while(!validInfile) {
00499             std::string inFile;
00500             if(!useReadline) getline(std::cin,inFile);
00501             #ifndef NO_READLINE
00502             else {
00503                 char *line = readline("External Capture Amplitude File (leave blank for new file): ");
00504                 inFile=line;
00505                 size_t endpos = inFile.find_last_not_of(" \t");
00506                 if( std::string::npos != endpos ) inFile = inFile.substr( 0, endpos+1 );
00507                 if(line && *line) add_history(line);
00508                 free(line);
00509             }
00510             #endif
00511             if(!inFile.empty()) {
00512                 std::ifstream in;
00513                 in.open(inFile.c_str());
00514                 if(in) {
00515                     validInfile=true;
00516                     configure.paramMask |= Config::USE_PREVIOUS_INTEGRALS;
00517                     configure.integralsfile=inFile;
00518                     in.close();
00519                 }
00520                 in.clear();
00521             } else validInfile=true;
00522             if(!validInfile) {
00523                 configure.outStream << "Cannot Read From " << inFile << ". Please reenter file." << std::endl;
00524                 if(!useReadline) configure.outStream << "External Capture Amplitude File (leave blank for new
file): ";
00525             }
00526         }
00527     }
00528 }
00529 }
00530 }
00531 }
00532 }
00533 }
00534 }
00535 void startMessage(const Config& configure) {
00536     if(configure.paramMask & Config::PERFORM_ERROR_ANALYSIS)
00537         configure.outStream << "Calling AZURE2 for MINOS Error Analysis..." << std::endl;
00538     else if(configure.paramMask & Config::CALCULATE_REACTION_RATE)
00539         configure.outStream << "Calling AZURE2 for reaction rate calculation..." << std::endl;
00540     else if(!configure.paramMask & Config::CALCULATE_WITH_DATA)
00541         configure.outStream << "Calling AZURE2 for calculation of segments without data..." << std::endl;
00542     else if(configure.paramMask & Config::PERFORM_FIT)
00543         configure.outStream << "Calling AZURE2 for fitting of segments from data..." << std::endl;
00544     else configure.outStream << "Calling AZURE2 for calculation of segments from data..." << std::endl;
00545 }
00546 }
00547 int main(int argc,char *argv[]){
00548     //Check for --help option first. If set, print help and exit.
00549     for(int i=1;i<argc;i++)
00550         if(strcmp(argv[i],"--help")==0) {
00551             printHelp();
00552             return 0;
00553         }
00554     //Set GSL Error Handler
00555     gsl_set_error_handler (&GSLException::GSLErrorHandler);
00556     //If GUI is built, look for --no-gui option. If not set, hand control to GUI.
00557     #ifdef GUI_BUILD
00558     bool useGUI=true;

```



```

00571     for(int i=1;i<argc;i++)
00572         if(strcmp(argv[i],"--no-gui")==0) useGUI=false;
00573     if(useGUI) return start_gui(argc,argv);
00574 #endif
00575
00576     //Create new configuration structure, and parse the command line parameters
00577     Config configure(std::cout);
00578     bool useReadline = parseOptions(argc,argv,configure);
00579
00580     //Read the parameters from the runtime configuration file
00581     if(configure.configfile.empty()) {
00582         configure.outStream << "A valid configuration file must be specified." << std::endl
00583             << "\tSyntax: AZURE2 <options> configfile" << std::endl;
00584         return -1;
00585     }
00586     if(configure.ReadConfigFile()==-1) {
00587         configure.outStream << "Could not open " << configure.configfile << ". Check that file exists."
00588             << std::endl;
00589         return -1;
00590     }
00591 #ifndef NO_STAT
00592     else if(configure.CheckForInputFiles() == -1) return -1;
00593 #endif
00594     if((configure.paramMask & Config::USE_RMC_FORMALISM) && (configure.paramMask &
00595         Config::USE_BRUNE_FORMALISM)) {
00596         configure.outStream << "WARNING: --use-brune is incompatible with --use-rmc. Ignoring --use-brune."
00597             << std::endl;
00598         configure.paramMask &= ~Config::USE_BRUNE_FORMALISM;
00599     }
00600     if((configure.paramMask & Config::USE_BRUNE_FORMALISM) || (configure.paramMask &
00601         Config::IGNORE_ZERO_WIDTHS)) {
00602         if(!(configure.paramMask & Config::USE_AMATRIX))
00603             configure.outStream << "WARNING: R-Matrix specified but --ignore-externals and --use-brune
00604             options require A-Matrix. A-Matrix will be used."
00605                 << std::endl;
00606         configure.paramMask |= Config::USE_AMATRIX;
00607     }
00608     //Print welcome message
00609     welcomeMessage(configure);
00610
00611     //Read and process command, setting appropriate configuration flags
00612     processCommand(commandShell(configure),configure);
00613
00614     //Open history file for readline
00615 #ifndef NO_READLINE
00616     if(useReadline) read_history("./.azure_history");
00617 #endif
00618
00619     //Read the external parameter file to be used, if any
00620     getParameterFile(useReadline,configure);
00621
00622     //Parse the segment files for entrance,exit pairs
00623     std::vector<SegPairs> segPairs;
00624     if(!(configure.paramMask & Config::CALCULATE_REACTION_RATE)) {
00625         if(!readSegmentFile(configure,segPairs)) exit(1);
00626     } else getRateParams(configure,segPairs,useReadline);
00627
00628     //Check if the entrance,exit pairs are in the external capture file
00629     // If so, external capture will be needed
00630     if(!checkExternalCapture(configure,segPairs)) exit(1);
00631
00632     //Read the external capture file name to be used, if any
00633     getExternalCaptureFile(useReadline,configure);
00634
00635     //Create instance of main AZURE function, print start message,
00636     // and execute
00637     AZUREMain azureMain(configure);
00638     configure.outStream << std::endl; startMessage(configure);
00639     int returnValue = azureMain();
00640
00641     //Print exit message
00642     exitMessage(configure);
00643
00644     //Write readline history file
00645 #ifndef NO_READLINE
00646     if(useReadline) write_history("./.azure_history");
00647 #endif
00648     return returnValue;
00649 }

```

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.](#)

```
00001 #include "AZURECalc.h"
00002 #include "Config.h"
00003 #include "CNuc.h"
00004 #include "EData.h"
00005 #include <iostream>
00006 #include <iomanip>
00007
00008 double AZURECalc::operator()(const vector_r& p) const {
00009
00010     int thisIteration=data()->Iterations();
00011     data()->Iterate();
00012     bool isFit=data()->IsFit();
00013
00014     CNuc * localCompound = NULL;
00015     EData *localData = NULL;
00016     if(isFit) {
00017         localCompound = compound()->Clone();
00018         localData = data()->Clone();
00019     } else {
00020         localCompound = compound();
00021         localData = data();
00022     }
00023
00024     //Fill Compound Nucleus From Minuit Parameters
00025     localCompound->FillCompoundFromParams(p);
00026     localData->FillNormsFromParams(p);
00027     if(configure().paramMask & Config::USE_BRUNE_FORMALISM)
00028         localCompound->CalcShiftFunctions(configure());
00029
00029     //loop over segments and points
00030     double chiSquared=0.0;
00031     double segmentChiSquared=0.0;
00032     ESegmentIterator firstSumIterator = localData->GetSegments().end();
00033     ESegmentIterator lastSumIterator = localData->GetSegments().end();
00034     for(EDataIterator data=localData->begin();data!=localData->end();data++) {
00035         if(data.segment()->GetPoints().begin()==data.point()) {
00036             segmentChiSquared=0.0;
00037             if(data.segment()->IsTotalCapture() {
00038                 firstSumIterator=data.segment();
00039                 lastSumIterator=data.segment()->IsTotalCapture()-1;
00040             }
00041         }
00042         if(!data.point()->IsMapped()) data.point()->Calculate(localCompound,configure());
00043         if(firstSumIterator!=localData->GetSegments().end() &&
00044             data.segment()!=lastSumIterator) continue;
00045         double fitCrossSection=data.point()->GetFitCrossSection();
00046         ESegmentIterator thisSegment = data.segment();
00047         if(data.segment()==lastSumIterator) {
00048             int pointIndex=data.point()-data.segment()->GetPoints().begin()+1;
00049             for(ESegmentIterator it=firstSumIterator;it<data.segment();it++)
00050                 fitCrossSection+=it->GetPoint(pointIndex)->GetFitCrossSection();
00051             thisSegment = firstSumIterator;
00052         }
00053         double dataNorm=thisSegment->GetNorm();
00054         double CrossSection=data.point()->GetCMCrossSection()*dataNorm;
00055         double CrossSectionError=data.point()->GetCMCrossSectionError()*dataNorm;
00056         double chi=(fitCrossSection-CrossSection)/CrossSectionError;
00057         double pointChiSquared=pow(chi,2.0);
00058         segmentChiSquared+=pointChiSquared;
00059         if(data.segment()->GetPoints().end()-1==data.point()) {
00060             if(!isFit) thisSegment->SetSegmentChiSquared(segmentChiSquared);
```

```

00061     if(data.segment()==lastSumIterator) {
00062         firstSumIterator=localData->GetSegments().end();
00063         lastSumIterator=localData->GetSegments().end();
00064     }
00065     double dataNormNominal=thisSegment->GetNominalNorm();
00066     double dataNormError=dataNormNominal/100.*thisSegment->GetNormError();
00067     if(dataNormError!=0.)
00068         segmentChiSquared += pow((dataNorm-dataNormNominal)/dataNormError,2.0);
00069     chiSquared+=segmentChiSquared;
00070 }
00071 }
00072
00073 if(!localData->IsErrorAnalysis()&&thisIteration!=0) {
00074     if(thisIteration%100==0) configure().outStream
00075         << "\r\tIteration: " << std::setw(6) << thisIteration
00076         << " Chi-Squared: " << chiSquared; configure().outStream.flush();
00077
00078     if(thisIteration%1000==0) {
00079         localData->WriteOutputFiles(configure(),isFit);
00080         localCompound->TransformOut(configure());
00081         localCompound->PrintTransformParams(configure());
00082     }
00083 }
00084 if(isFit) {
00085     delete localCompound;
00086     delete localData;
00087 }
00088 if(configure().stopFlag&&isFit) return 0.;
00089 else return chiSquared;
00090 }

```

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 "GSLEException.h"
#include <Minuit2/FunctionMinimum.h>
#include <Minuit2/MnMigrad.h>
#include <Minuit2/MnMinos.h>

```

8.235 AZUREMain.cpp

[Go to the documentation of this file.](#)

```

00001 #include "AZURECalc.h"
00002 #include "AZUREMain.h"
00003 #include "AZUREParams.h"
00004 #include "Config.h"
00005 #include "ReactionRate.h"
00006 #include "Minuit2/MnPrint.h"
00007 #include "GSLEException.h"
00008 #include <Minuit2/FunctionMinimum.h>
00009 #include <Minuit2/MnMigrad.h>
00010 #include <Minuit2/MnMinos.h>
00011
00012 int AZUREMain::operator() () {
00013     //Fill compound nucleus from nucfile
00014     configure().outStream << "Filling Compound Nucleus..." << std::endl;
00015     if(compound()->Fill(configure())==-1) {
00016         configure().outStream << "Could not fill compound nucleus from file."
00017         << std::endl;
00018         return -1;
00019     } else if(compound()->NumPairs()==0 || compound()->NumJGroups()==0) {
00020         configure().outStream << "No nuclear data exists. Calculation not possible." << std::endl;

```

```

00021     return -1;
00022 }
00023 if((configure().screenCheckMask|configure().fileCheckMask) &
00024     Config::CHECK_COMPOUND_NUCLEUS) compound()->PrintNuc(configure());
00025
00026 if(!(configure().paramMask & Config::CALCULATE_REACTION_RATE)) {
00027     //Fill the data object from the segments and data file
00028     // Compound object is passed to the function for pair key verification and
00029     // center of mass conversions, s-factor conversions, etc.
00030     configure().outStream << "Filling Data Structures..." << std::endl;
00031     if(configure().paramMask & Config::CALCULATE_WITH_DATA) {
00032         if(data()->Fill(configure(),compound())==1) {
00033             configure().outStream << "Could not fill data object from file." << std::endl;
00034             return -1;
00035         } else if(data()->NumSegments()==0) {
00036             configure().outStream << "There is no data provided." << std::endl;
00037             return -1;
00038         }
00039     } else {
00040         if(data()->MakePoints(configure(),compound())==1) {
00041             configure().outStream << "Could not fill data object from file." << std::endl;
00042             return -1;
00043         } else if(data()->NumSegments()==0) {
00044             configure().outStream << "Extrapolation segments produce no data." << std::endl;
00045             return -1;
00046         }
00047     }
00048     if((configure().fileCheckMask|configure().screenCheckMask) & Config::CHECK_DATA)
00049         data()->PrintData(configure());
00050 } else {
00051     if(!compound()->IsPairKey(configure().rateParams.entrancePair)||!compound()->IsPairKey(configure().rateParams.exitPair))
00052     {
00053         configure().outStream << "Reaction rate pairs do not exist in compound nucleus." << std::endl;
00054         return -1;
00055     } else {
00056         compound()->GetPair(compound()->GetPairNumFromKey(configure().rateParams.entrancePair))->SetEntrance();
00057     }
00058 }
00059 //Initialize compound nucleus object
00060 try {
00061     compound()->Initialize(configure());
00062 } catch (GSLException e) {
00063     configure().outStream << e.what() << std::endl;
00064     configure().outStream << std::endl
00065         << "Calculation was aborted." << std::endl;
00066     return -1;
00067 }
00068
00069 //Create new parameters for minuit, fill them from compound nucleus object and data file.
00070 AZUREParams params;
00071 compound()->FillMnParams(params.GetMinuitParams());
00072 data()->FillMnParams(params.GetMinuitParams());
00073 if(!(configure().paramMask & Config::USE_PREVIOUS_PARAMETERS)) {
00074     configure().outStream << "Creating New param.par File..." << std::endl;
00075     params.WriteUserParameters(configure(),false);
00076 } else {
00077     configure().outStream << "Reading User Parameter File..." << std::endl;
00078     params.ReadUserParameters(configure());
00079 }
00080
00081 if(!(configure().paramMask & Config::CALCULATE_REACTION_RATE)) {
00082     //Initialize data object
00083     if(data()->Initialize(compound(),configure())==1) {
00084         configure().outStream << std::endl
00085             << "Calculation was aborted." << std::endl;
00086         return -1;
00087     }
00088 }
00089 //Declare a new instance of FCNBase
00090 AZURECalc theFunc(data(),compound(),configure());
00091 theFunc.SetErrorDef(1.0);
00092
00093 if(configure().paramMask & Config::PERFORM_FIT) {
00094     //Call Minuit for function minimization, write minimized parameters to params
00095     if(configure().paramMask & Config::USE_AMATRIX) configure().outStream << "Performing A-Matrix
Fit..." << std::endl;
00096     else configure().outStream << "Performing R-Matrix Fit..." << std::endl;
00097     data()->SetFit(true);
00098     ROOT::Minuit2::MnMigrad migrad(theFunc,params.GetMinuitParams());
00099     ROOT::Minuit2::FunctionMinimum min=migrad(50000);
00100     if(configure().paramMask & Config::PERFORM_ERROR_ANALYSIS) {
00101         configure().outStream << std::endl
00102             << "Performing parameter error analysis with Up=" << configure().chiVariance << "." <<
std::endl;

```

```

00103     data()->SetErrorAnalysis(true);
00104     theFunc.SetErrorDef(configure().chiVariance);
00105     ROOT::Minuit2::MnMinos minos(theFunc,min);
00106     std::vector<std::pair<double,double> > errors;
00107     for(int i = 0; i<params.GetMinuitParams().Params().size(); i++) {
00108         configure().outStream << "\tParameter " << i+1 << "..." << std::endl;
00109         if(!params.GetMinuitParams().Parameter(i).IsFixed()) {
00110             std::pair< double, double > error=minos(i);
00111             errors.push_back(error);
00112         } else errors.push_back(std::pair< double, double > (0.,0.));
00113     }
00114
00115     // New output of the covariance matrix
00116     char filename[256];
00117     sprintf(filename,"%scovariance_matrix.out",configure().outputdir.c_str());
00118     std::ofstream out;
00119     out.open(filename);
00120     if(out) {
00121
00122         // Write header information
00123         params.GetMinuitParams()=min.UserParameters();
00124         out << "Parameter List" << std::endl << std::endl;
00125         for(int i = 0; i<params.GetMinuitParams().Params().size(); i++) {
00126             out << std::setw(20) << params.GetMinuitParams().GetName(i)
00127                 << " (" << std::setw(3) << i << ")"
00128                 << std::scientific << std::setw(20) << params.GetMinuitParams().Value(i);
00129             if(params.GetMinuitParams().Parameter(i).IsFixed()) {
00130                 out << " Fixed" << std::endl;
00131             }
00132             else {
00133                 out << " Fitted by Minuit" << std::endl;
00134             }
00135         }
00136
00137         std::cout << min.UserCovariance();
00138
00139         std::vector<double> CovarianceData;
00140         CovarianceData=min.UserCovariance().Data();
00141         int parameterTable[100];
00142         int size=0;
00143
00144         // Header for Covariance Matrix
00145         out << std::endl << "Covariance Matrix" << std::endl << std::endl;
00146         out << std::setw(5) << " ";
00147         for(int i = 0; i<params.GetMinuitParams().Params().size(); i++) {
00148             if(!params.GetMinuitParams().Parameter(i).IsFixed()) {
00149                 out << std::setw(15) << i ;
00150                 parameterTable[size]=i;
00151                 size=size+1;
00152             }
00153         }
00154         out << std::endl;
00155
00156         std::cout << "covariance length " << CovarianceData.size() << std::endl;
00157
00158         // Covariance matrix
00159         for(int i = 0; i<size; i++) {
00160             out << std::setw(5) << parameterTable[i];
00161             for(int j = 0; j<size; j++) {
00162                 int k;
00163                 if(i<=j) { k=((j+1)*j)/2+i; }
00164                 if(i>j) { k=((i+1)*i)/2+j; }
00165                 std::cout << std::setw(5) << i << std::setw(5) << j << std::setw(5) << k << std::endl;
00166                 out << std::setw(15) << CovarianceData[k];
00167             }
00168             out << std::endl;
00169         }
00170
00171         // Header for Correlation Matrix
00172         out << std::endl << "Correlation Matrix" << std::endl << std::endl;
00173         out << std::setw(5) << " ";
00174         for(int i = 0; i<params.GetMinuitParams().Params().size(); i++) {
00175             if(!params.GetMinuitParams().Parameter(i).IsFixed()) {
00176                 out << std::setw(15) << i ;
00177             }
00178         }
00179         out << std::endl;
00180
00181         // Correlation matrix
00182         for(int i = 0; i<size; i++) {
00183             out << std::setw(5) << parameterTable[i];
00184             for(int j = 0; j<size; j++) {
00185                 int k;
00186                 if(i<=j) { k=((j+1)*j)/2+i; }
00187                 if(i>j) { k=((i+1)*i)/2+j; }
00188                 int jdiag=((j+2)*(j+1))/2-1;
00189                 int iddiag=((i+2)*(i+1))/2-1;

```

```

00190         std::cout << k << " " << CovarianceData[k] << std::endl;
00191         out << std::setw(15) << std::fixed <<
CovarianceData[k]/sqrt(fabs(CovarianceData[jdiag]*CovarianceData[idiag]));
00192     }
00193     out << std::endl;
00194 }
00195
00196     out.flush();
00197     out.close();
00198 } else std::cout << "Could not save " << configure().outputdir.c_str()
00199     << "covariance_matrix.out file." << std::endl;
00200
00201     // ECS this line added to set the azure variables to the minimised fit parameters
00202     params.GetMinuitParams()=min.UserParameters();
00203     params.WriteParameterErrors(errors,configure());
00204 }
00205
00206     params.GetMinuitParams()=min.UserParameters();
00207     params.WriteUserParameters(configure(),true);
00208 } else {
00209     if(configure().paramMask & Config::USE_AMATRIX) configure().outStream << "Performing A-Matrix
Calculation..." << std::endl;
00210     else configure().outStream << "Performing R-Matrix Calculation..." << std::endl;
00211 }
00212 data()->SetFit(false);
00213 data()->SetErrorAnalysis(false);
00214 double chiSquared=theFunc(params.GetMinuitParams().Params());
00215 if(configure().paramMask & Config::CALCULATE_WITH_DATA) {
00216     configure().outStream << std::endl << std::endl;
00217     for(ESegmentIterator segment=data()->GetSegments().begin();
00218         segment<data()->GetSegments().end();segment++) {
00219         configure().outStream << "Segment #"
00220             << segment->GetSegmentKey()
00221             << " Chi-Squared/N: "
00222             << segment->GetSegmentChiSquared()/segment->NumPoints()
00223             << std::endl;
00224         if(segment->IsTotalCapture()) segment+=segment->IsTotalCapture()-1;
00225     }
00226     configure().outStream << "Total Chi-Squared: "
00227         << chiSquared << std::endl << std::endl;
00228 }
00229
00230 //Write Output Files
00231 configure().outStream << "Writing output files..." << std::endl;
00232 data()->WriteOutputFiles(configure());
00233 } else {
00234     //Calculate Reaction Rate
00235     // This uses the adaptive integration routines of GSL. As the energy stepsize is
00236     // unknown until integration, AZURE is called not in segment control mode but as
00237     // an energy dependent function. As every data point must be reinitialized (new energy
00238     // dependent terms calculated) the routine is slow. This should be a tradeoff
00239     // for good accuracy.
00240     configure().outStream << "Performing reaction rate calculation..." << std::endl;
00241     ReactionRate reactionRate(compound(),params.GetMinuitParams().Params(),configure(),
00242         configure().rateParams.entrancePair,configure().rateParams.exitPair);
00243     if(configure().paramMask & Config::USE_BRUNE_FORMALISM)
00244         compound()->CalcShiftFunctions(configure());
00245     if(configure().rateParams.useFile)
00246         reactionRate.CalculateFileRates();
00247     else
00248         reactionRate.CalculateRates();
00249     reactionRate.WriteRates();
00250 }
00251
00252 configure().outStream << "Performing final parameter transformation..." << std::endl;
00253 try {
00254     compound()->TransformOut(configure());
00255 } catch (GSLException e) {
00256     configure().outStream << e.what() << std::endl;
00257     configure().outStream << "Problem with output transformation. Aborting."
00258         << std::endl;
00259     return -1;
00260 }
00261 compound()->PrintTransformParams(configure());
00262
00263 return 0;
00264 }

```

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.](#)

```
00001 #include "AZUREOutput.h"
00002
00007 AZUREOutput::AZUREOutput(std::string outputdir) {
00008     outputdir_=outputdir;
00009     is_extrap_=false;
00010 }
00011
00016 AZUREOutput::~AZUREOutput() {
00017     for(int i=0;i<azurefbuffers_.size();i++) delete azurefbuffers_[i];
00018 }
00019
00024 bool AZUREOutput::IsExtrap() const {
00025     return is_extrap_;
00026 }
00027
00037 std::filebuf *AZUREOutput::operator()(int entranceKey, int exitKey, bool isAngDist) {
00038     int c=this->IsAZUREFBuffer(entranceKey,exitKey,isAngDist);
00039     if(!c) {
00040         AZUREFBuffer *d = new
00041         AZUREFBuffer(entranceKey,exitKey,this->GetOutputDir(),this->IsExtrap(),isAngDist);
00042         this->AddAZUREFBuffer(d);
00043         c=this->IsAZUREFBuffer(entranceKey,exitKey,isAngDist);
00044     }
00045     return this->GetAZUREFBuffer(c)->GetFBuffer();
00046 }
00051 int AZUREOutput::NumAZUREFBuffers() const {
00052     return azurefbuffers_.size();
00053 }
00054
00061 int AZUREOutput::IsAZUREFBuffer(int entranceKey, int exitKey, bool isAngDist) {
00062     bool c=false;
00063     int d=0;
00064     while(!c&&d<this->NumAZUREFBuffers()) {
00065         if(this->GetAZUREFBuffer(d+1)->IsAngDist()==isAngDist&&
00066         this->GetAZUREFBuffer(d+1)->GetEntranceKey()==entranceKey&&
00067         this->GetAZUREFBuffer(d+1)->GetExitKey()==exitKey) c=true;
00068         d++;
00069     }
00070     if(c) return d;
00071     else return 0;
00072 }
00073
00078 std::string AZUREOutput::GetOutputDir() const {
00079     return outputdir_;
00080 };
00081
00086 void AZUREOutput::AddAZUREFBuffer(AZUREFBuffer *azureFBuffer) {
00087     azurefbuffers_.push_back(azureFBuffer);
00088 }
00089
00094 void AZUREOutput::SetExtrap() {
00095     is_extrap_=true;
00096 }
00097
00102 AZUREFBuffer *AZUREOutput::GetAZUREFBuffer(int fBufferNum) {
00103     AZUREFBuffer *b=azurefbuffers_[fBufferNum-1];
00104     return b;
00105 }
00106
00107
```

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.](#)

```
00001 #include "AZUREParams.h"
00002 #include "Config.h"
00003
00009 ROOT::Minuit2::MnUserParameters &AZUREParams::GetMinuitParams() {
00010     return params_;
00011 }
00012
00019 void AZUREParams::ReadUserParameters(const Config& configure) {
00020     std::vector<std::string> names;
00021     vector_r values;
00022     vector_r errors;
00023     std::vector<bool> fixed;
00024     std::string tempname,tempfixed,tempfixed_nows;
00025     double tempvalue,temperror;
00026
00027     std::ifstream in;
00028     in.open(configure.paramfile.c_str());
00029     if(in) {
00030         while(!in.eof()) {
00031             in >> tempname >> tempvalue >> temperror; getline(in,tempfixed);
00032             if(!in.eof()) {
00033                 names.push_back(tempname);
00034                 values.push_back(tempvalue);
00035                 errors.push_back(temperror);
00036                 tempfixed_nows.clear();
00037                 for(int i=0;i<tempfixed.length();i++)
00038                     if(tempfixed[i]!=' ' && tempfixed[i]!='\t')
00039                         tempfixed_nows.push_back(tempfixed[i]);
00040                 if(tempfixed_nows=="fixed") fixed.push_back(true);
00041                 else fixed.push_back(false);
00042             }
00043         }
00044         in.close();
00045     } else configure.outStream << "Could not read user parameter file." << std::endl;
00046
00047     for(int i=0;i<GetMinuitParams().Params().size();i++) {
00048         for(int ii=0;ii<names.size();ii++) {
00049             if(GetMinuitParams().GetName(i)==names[ii]) {
00050                 if(GetMinuitParams().Value(i)==0.0&&values[ii]!=0.0&&
00051                    GetMinuitParams().Parameter(i).IsFixed()) GetMinuitParams().Release(i);
00052                 if(GetMinuitParams().Value(i)!=0.0&&values[ii]==0.0&&
00053                    !GetMinuitParams().Parameter(i).IsFixed()) GetMinuitParams().Fix(i);
00054                 GetMinuitParams().SetValue(i,values[ii]);
00055                 GetMinuitParams().SetError(i,errors[ii]);
00056                 if(fixed[ii]&&!GetMinuitParams().Parameter(i).IsFixed()) GetMinuitParams().Fix(i);
00057             }
00058         }
00059     }
00060 }
00061
00066 void AZUREParams::WriteUserParameters(const Config& configure, bool fitParameters) {
00067     char filename[256];
00068     if(fitParameters) sprintf(filename,"%sparam.sav",configure.outputdir.c_str());
00069     else sprintf(filename,"%sparam.par",configure.outputdir.c_str());
00070     std::ofstream out;
00071     out.open(filename);
00072     if(out) {
00073         out.precision(7);
00074         for(int i=0;i<GetMinuitParams().Params().size();i++) {
00075             out << std::setw(20) << GetMinuitParams().GetName(i)
00076                 << std::scientific << std::setw(20) << GetMinuitParams().Value(i)
00077                 << std::scientific << std::setw(20) << GetMinuitParams().Error(i) << std::endl;
00078         }
00079         out.flush();
00080         out.close();
00081     } else configure.outStream << "Could not save param.par file." << std::endl;
00082 }
```



```

00083
00088 void AZUREParams::WriteParameterErrors(const std::vector<std::pair<double,double> > &errors,const
    Config& configure) {
00089     char filename[256];
00090     sprintf(filename,"%sparam.errors",configure.outputdir.c_str());
00091     std::ofstream out;
00092     out.open(filename);
00093     if(out) {
00094         out.precision(7);
00095         for(int i=0;i<GetMinuitParams().Params().size();i++) {
00096             out << std::setw(20) << GetMinuitParams().GetName(i)
00097                 << std::scientific << std::setw(20) << GetMinuitParams().Value(i)
00098                 << std::scientific << std::setw(20) << fabs(errors[i].first)
00099                 << std::scientific << std::setw(20) << fabs(errors[i].second)
00100                 << std::endl;
00101         }
00102         out.flush();
00103         out.close();
00104     } else configure.outStream << "Could not save param.errors file." << std::endl;
00105 }

```

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.](#)

```

00001 #include <iostream>
00002 #include <iomanip>
00003 #include <sstream>
00004 #include "AngCoeff.h"
00005 #include "CNuc.h"
00006 #include "Config.h"
00007 #include "CoulFunc.h"
00008 #include "EigenFunc.h"
00009 #include "ECIntegral.h"
00010 #include "NucLine.h"
00011 #include "Minuit2/MnUserParameters.h"
00012 #include "NFIntegral.h"
00013 #include "ShftFunc.h"
00014
00019 bool CNuc::IsPairKey(int key) {
00020     bool b=false;
00021     int c=0;
00022     while(!b&&c<this->NumPairs()) {
00023         if(key==this->GetPair(c+1)->GetPairKey()) b=true;
00024         c++;
00025     }
00026     return b;
00027 }
00028
00033 int CNuc::NumPairs() const {

```

```

00034     return pairs_.size();
00035 }
00036
00041 int CNuc::NumJGroups() const {
00042     return jgroups_.size();
00043 }
00044
00050 int CNuc::IsPair(PPair pair) {
00051     bool b=false;
00052     int c=0;
00053     while(!b&& c<this->NumPairs())
00054     {
00055         if(pair.GetPairKey()==this->GetPair(c+1)->GetPairKey()) b=true;
00056         c++;
00057     }
00058     if(b) return c;
00059     else return 0;
00060 }
00061
00067 int CNuc::IsJGroup(JGroup jGroup) {
00068     bool b=false;
00069     int c=0;
00070     while(!b&& c<this->NumJGroups())
00071     {
00072         if(jGroup.GetJ()==this->GetJGroup(c+1)->GetJ() &&
00073            jGroup.GetPi()==this->GetJGroup(c+1)->GetPi()) b=true;
00074         c++;
00075     }
00076     if(b) return c;
00077     else return 0;
00078 }
00079
00087 int CNuc::GetPairNumFromKey(int key) {
00088     bool b=false;
00089     int c=0;
00090     while(!b&& c<this->NumPairs()) {
00091         if(key==this->GetPair(c+1)->GetPairKey()) b=true;
00092         c++;
00093     }
00094     if(b) return c;
00095     else return 0;
00096 }
00097
00103 int CNuc::Fill(const Config &configure) {
00104     int PairNum,LevelNum,ChannelNum,JGroupNum;
00105     int maxLValue=0;
00106     std::ifstream in(configure.configfile.c_str());
00107     if(!in) return -1;
00108     std::string line = "";
00109     while(line!="<levels>"&&!in.eof()) getline(in,line);
00110     if(line!="<levels>") return -1;
00111     std::map<int,int> ecPairs;
00112     line="";
00113     while(!in.eof() && line!="</levels>") {
00114         getline(in,line);
00115         bool empty=true;
00116         for(unsigned int i=0;i<line.size();++i)
00117             if(line[i]!=' ' && line[i]!='\t') {
00118                 empty=false;
00119                 break;
00120             }
00121         if(empty==true) continue;
00122         if(!in.eof() && line!="</levels>") {
00123             std::istringstream stm;
00124             stm.str(line);
00125             NucLine Line(stm);
00126             if(stm.rdstate() & (std::stringstream::failbit | std::stringstream::badbit)) return -1;
00127             if(Line.l()>maxLValue&&Line.pType()==0) maxLValue=Line.l();
00128             if(Line.isActive()==1) {
00129                 PPair NewPair(Line);
00130                 PairNum=this->IsPair(NewPair);
00131                 if(!PairNum) {
00132                     this->AddPair(NewPair);
00133                     PairNum=this->IsPair(NewPair);
00134                 }
00135                 if(Line.ecMultMask()!=0) {
00136                     std::map<int,int>::iterator it = ecPairs.find(PairNum);
00137                     if(it==ecPairs.end()) ecPairs[PairNum]=Line.ecMultMask();
00138                 }
00139                 JGroup NewJGroup(Line);
00140                 JGroupNum=this->IsJGroup(NewJGroup);
00141                 if(!JGroupNum) {
00142                     this->AddJGroup(NewJGroup);
00143                     JGroupNum=this->IsJGroup(NewJGroup);
00144                 }
00145                 AChannel NewChannel(Line,PairNum);
00146                 ChannelNum=this->GetJGroup(JGroupNum)->IsChannel(NewChannel);

```

```

00147     if(!ChannelNum) {
00148         this->GetJGroup(JGroupNum)->AddChannel(NewChannel);
00149         ChannelNum=this->GetJGroup(JGroupNum)->IsChannel(NewChannel);
00150         if(this->GetJGroup(JGroupNum)->GetChannel(ChannelNum)->GetL()>maxLValue&&
00151             this->GetJGroup(JGroupNum)->GetChannel(ChannelNum)->GetRadType()=='P')
00152             maxLValue=this->GetJGroup(JGroupNum)->GetChannel(ChannelNum)->GetL();
00153     }
00154     ALevel NewLevel(Line);
00155     LevelNum=this->GetJGroup(JGroupNum)->IsLevel(NewLevel);
00156     if(!LevelNum) {
00157         this->GetJGroup(JGroupNum)->AddLevel(NewLevel);
00158         LevelNum=this->GetJGroup(JGroupNum)->IsLevel(NewLevel);
00159     }
00160     this->GetJGroup(JGroupNum)->GetLevel(LevelNum)->AddGamma(Line);
00161     }
00162 }
00163 }
00164
00165 if(line!="</levels>") return -1;
00166
00167 in.close();
00168
00169 this->SetMaxLValue(maxLValue);
00170 if((configure.paramMask & Config::USE_EXTERNAL_CAPTURE) &&
00171     this->NumJGroups()>0 && this->NumPairs()>0)
00172     this->ParseExternalCapture(configure,ecPairs);
00173
00174 return 0;
00175 }
00176
00182 void CNuc::ParseExternalCapture(const Config& configure,std::map<int,int>& ecPairs) {
00183     for(std::map<int,int>::iterator ec=ecPairs.begin();ec!=ecPairs.end();ec++) {
00184         PPair *exitPair=this->GetPair(ec->first);
00185         if(exitPair->GetPType()!=10) {
00186             configure.outStream << "Final state is not a capture pair." << std::endl;
00187             continue;
00188         }
00189         //create new level in compound nucleus for EC state, if it doesn't exist
00190         double jValue=exitPair->GetJ(2);
00191         int parity=exitPair->GetPi(2);
00192         JGroup newJGroup(jValue,parity);
00193         int jGroupNum=this->IsJGroup(newJGroup);
00194         int levelNum=0;
00195         if(jGroupNum) {
00196             ALevel newLevel(exitPair->GetExE());
00197             levelNum=this->GetJGroup(jGroupNum)->IsLevel(newLevel);
00198             if(!levelNum) {
00199                 this->GetJGroup(jGroupNum)->AddLevel(newLevel);
00200                 levelNum=this->GetJGroup(jGroupNum)->IsLevel(newLevel);
00201                 for(int ch=1;ch<=this->GetJGroup(jGroupNum)->NumChannels();ch++) {
00202                     if(this->GetJGroup(jGroupNum)->GetChannel(ch)->GetRadType()=='P')
00203                         this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddGamma(0.1);
00204                     else this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddGamma(0.0);
00205                 }
00206             }
00207             this->GetJGroup(jGroupNum)->GetLevel(levelNum)->SetECParams(ec->first,ec->second);
00208             for(int ch=1;ch<=this->GetJGroup(jGroupNum)->NumChannels();ch++) {
00209                 PPair *theFinalPair=this->GetPair(this->GetJGroup(jGroupNum)->GetChannel(ch)->GetPairNum());
00210                 double nfIntegralValue=0.;
00211                 double ecConvert=0.;
00212                 if(theFinalPair->GetPType()==0) {
00213                     NFIntegral newNFIntegral(theFinalPair);
00214
00215                     nfIntegralValue=newNFIntegral(this->GetJGroup(jGroupNum)->GetChannel(ch)->GetL(),exitPair->GetExE());
00216                     WhitFunc newWhitFunc(theFinalPair);
00217                     double whitConv=newWhitFunc(this->GetJGroup(jGroupNum)->GetChannel(ch)->GetL(),
00218                         theFinalPair->GetChRad(),
00219                         fabs(exitPair->GetExE()-theFinalPair->GetSepE()-theFinalPair->GetExE()));
00220                     ecConvert=sqrt(2.0*theFinalPair->GetRedMass()*theFinalPair->GetChRad()*uconv/pow(hbarc,2.0))/whitConv;
00221                 }
00222                 this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddNFIntegral(nfIntegralValue);
00223                 this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddECConversionFactor(ecConvert);
00224             }
00225             else {
00226                 this->AddJGroup(newJGroup);
00227                 jGroupNum=this->IsJGroup(newJGroup);
00228                 ALevel newLevel(exitPair->GetExE());
00229                 this->GetJGroup(jGroupNum)->AddLevel(newLevel);
00230                 levelNum=this->GetJGroup(jGroupNum)->IsLevel(newLevel);
00231                 this->GetJGroup(jGroupNum)->GetLevel(levelNum)->SetECParams(ec->first,ec->second);
00232                 for(int ir=1;ir<=this->NumPairs();ir++) {
00233                     if(this->GetPair(ir)->GetPType()==0) {
00234                         double s1=this->GetPair(ir)->GetJ(1);
00235                         double s2=this->GetPair(ir)->GetJ(2);
00236                         int sPi=this->GetPair(ir)->GetPi(1)*this->GetPair(ir)->GetPi(2);
00237                         for(double chS=fabs(s1-s2);chS<=s1+s2;chS+=1.) {

```

```

00237         for(int chL=0;chL<=this->GetMaxLValue();chL++) {
00238             int chPi=sPi*(int)pow(-1,chL);
00239             if(fabs(chS-chL)<=jValue&&jValue<=chS+chL&&chPi==parity) {
00240                 AChannel newChannel(chL,chS,ir,'P');
00241                 this->GetJGroup(jGroupNum)->AddChannel(newChannel);
00242                 this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddGamma(0.1);
00243                 NFIntegral newNFIntegral(this->GetPair(ir));
00244                 double nfIntegralValue=newNFIntegral(chL,exitPair->GetExE());
00245                 WhitFunc newWhitFunc(this->GetPair(ir));
00246                 double whitConv=newWhitFunc(chL,this->GetPair(ir)->GetChRad(),
00247
fabs(exitPair->GetExE()-this->GetPair(ir)->GetSepE()-this->GetPair(ir)->GetExE()));
00248                 double
ecConvert=sqrt(2.0*this->GetPair(ir)->GetRedMass()*this->GetPair(ir)->GetChRad()*uconv/pow(hbarc,2.0))/whitConv;
00249                 this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddNFIntegral(nfIntegralValue);
00250                 this->GetJGroup(jGroupNum)->GetLevel(levelNum)->AddECConversionFactor(ecConvert);
00251             }
00252         }
00253     }
00254 }
00255 }
00256 }
00257 }
00258 }
00259 }
00260
00265 int CNuc::GetMaxLValue() const {
00266     return maxLValue_;
00267 }
00268
00275 void CNuc::Initialize(const Config &configure) {
00276     //Calculate Boundary Conditions
00277     configure.outStream << "Calculating Boundary Conditions..." << std::endl;
00278     this->CalcBoundaryConditions(configure);
00279     if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_BOUNDARY_CONDITIONS)
00280         this->PrintBoundaryConditions(configure);
00281
00282     //Transform Input Parameters
00283     if(configure.paramMask & Config::TRANSFORM_PARAMETERS) {
00284         configure.outStream << "Performing Input Parameter Transformation..." << std::endl;
00285         this->TransformIn(configure);
00286     }
00287
00288     //Sort reaction pathways
00289     configure.outStream << "Sorting Reaction Pathways..." << std::endl;
00290     this->SortPathways(configure);
00291     if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_PATHWAYS)
00292         this->PrintPathways(configure);
00293
00294     //Calculate Angular Distribution Coefficients
00295     configure.outStream << "Calculating Angular Distribution Coefficients..." << std::endl;
00296     this->CalcAngularDists(configure.maxLOrder);
00297     if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_ANGULAR_DISTS)
00298         this->PrintAngularDists(configure);
00299
00300 }
00301
00306 void CNuc::AddPair(PPair pPair) {
00307     pairs_.push_back(pPair);
00308 }
00309
00314 void CNuc::AddJGroup(JGroup jGroup) {
00315     jgroups_.push_back(jGroup);
00316 }
00317
00323 void CNuc::PrintNuc(const Config &configure) {
00324     std::streambuf *sbuffer;
00325     std::filebuf fbuffer;
00326     if(configure.fileCheckMask & Config::CHECK_COMPOUND_NUCLEUS) {
00327         std::string outfile=configure.checkdir+"compoundnucleus.chk";
00328         fbuffer.open(outfile.c_str(),std::ios::out);
00329         sbuffer = &fbuffer;
00330     } else if(configure.screenCheckMask & Config::CHECK_COMPOUND_NUCLEUS)
00331         sbuffer = configure.outStream.rdbuf();
00332     std::ostream out(sbuffer);
00333     if(((configure.fileCheckMask & Config::CHECK_COMPOUND_NUCLEUS)&&
00334         fbuffer.is_open())||
00335         (configure.screenCheckMask & Config::CHECK_COMPOUND_NUCLEUS)) {
00336         out << std::endl
00337         << "*****" << std::endl
00338         << " *           Particle Pairs           * " << std::endl
00339         << "*****" << std::endl;
00340         for(int i=1;i<=this->NumPairs();i++) {
00341             out << "Pair Number: " << i << " Pair Key: " << this->GetPair(i)->GetPairKey() << std::endl;
00342             out << std::setw(30) << "Light Particle J: " << this->GetPair(i)->GetJ(1) << std::endl
00343             << std::setw(30) << "Light Particle Parity: " << this->GetPair(i)->GetPi(1) << std::endl
00344             << std::setw(30) << "Light Particle Z: " << this->GetPair(i)->GetZ(1) << std::endl

```

```

00345     « std::setw(30) « "Light Particle M: " « this->GetPair(i)->GetM(1) « std::endl
00346     « std::setw(30) « "Light Particle g: " « this->GetPair(i)->GetG(1) « std::endl
00347     « std::setw(30) « "Heavy Particle J: " « this->GetPair(i)->GetJ(2) « std::endl
00348     « std::setw(30) « "Heavy Particle Parity: " « this->GetPair(i)->GetPi(2) « std::endl
00349     « std::setw(30) « "Heavy Particle Z: " « this->GetPair(i)->GetZ(2) « std::endl
00350     « std::setw(30) « "Heavy Particle M: " « this->GetPair(i)->GetM(2) « std::endl
00351     « std::setw(30) « "Heavy Particle g: " « this->GetPair(i)->GetG(2) « std::endl
00352     « std::setw(30) « "Seperation Energy [MeV]: " « this->GetPair(i)->GetSepE() « std::endl
00353     « std::setw(30) « "Excitation Energy [MeV]: " « this->GetPair(i)->GetExE() « std::endl
00354     « std::setw(30) « "Channel Radius: " « this->GetPair(i)->GetChRad() « std::endl;
00355     if(this->GetPair(i)->GetPType()==0) out « std::setw(30) « "Pair Type: " « "Particle,Particle" «
std::endl;
00356     else if(this->GetPair(i)->GetPType()==10) out « std::setw(30) « "Pair Type: " « "Particle,Gamma"
« std::endl;
00357     else if(this->GetPair(i)->GetPType()==20) out « std::setw(30) « "Pair Type: " « "Beta Decay" «
std::endl;
00358     else out « std::setw(30) « "Pair Type: Unknown" « std::endl;
00359 }
00360 out « std::endl
00361 « "*****" « std::endl
00362 « " *           Levels           * " « std::endl
00363 « "*****" « std::endl
00364 « std::setw(11) « "J Group #"
00365 « std::setw(5) « "J"
00366 « std::setw(4) « "Pi"
00367 « std::setw(9) « "Level #"
00368 « std::setw(14) « "Energy [MeV]"
00369 « std::setw(11) « "Channel #"
00370 « std::setw(3) « "l"
00371 « std::setw(5) « "s"
00372 « std::setw(8) « "Pair #"
00373 « std::setw(11) « "Width"
00374 « std::setw(11) « "Rad. Type" « std::endl;
00375
00376 for(int i=1;i<=this->NumJGroups();i++) {
00377     for(int ii=1;ii<=this->GetJGroup(i)->NumLevels();ii++) {
00378         for(int iii=1;iii<=this->GetJGroup(i)->NumChannels();iii++) {
00379             out « std::setw(11) « i
00380                 « std::setw(5) « this->GetJGroup(i)->GetJ()
00381                 « std::setw(4) « this->GetJGroup(i)->GetPi()
00382                 « std::setw(9) « ii
00383                 « std::setw(14) « this->GetJGroup(i)->GetLevel(ii)->GetE()
00384                 « std::setw(11) « iii
00385                 « std::setw(3) « this->GetJGroup(i)->GetChannel(iii)->GetL()
00386                 « std::setw(5) « this->GetJGroup(i)->GetChannel(iii)->GetS()
00387                 « std::setw(8) « this->GetJGroup(i)->GetChannel(iii)->GetPairNum()
00388                 « std::setw(11) « this->GetJGroup(i)->GetLevel(ii)->GetGamma(iii)
00389                 « std::setw(11) « this->GetJGroup(i)->GetChannel(iii)->GetRadType() « std::endl;
00390         }
00391     }
00392     out « std::endl;
00393 }
00394 } else configure.outStream « "Could not write compound nucleus check file." « std::endl;
00395 out.flush();
00396 if(fbuffer.is_open()) fbuffer.close();
00397 }
00398
00403 void CNuc::TransformIn(const Config& configure) {
00404     for(int j=1;j<=this->NumJGroups();j++) {
00405         JGroup *theJGroup=this->GetJGroup(j);
00406         if(theJGroup->IsInRMatrix()) {
00407             for(int la=1;la<=theJGroup->NumLevels();la++) {
00408                 ALevel *theLevel=theJGroup->GetLevel(la);
00409                 if(theLevel->IsInRMatrix()) {
00410                     vector_r tempGammas;
00411                     std::vector<bool> isNegative;
00412                     vector_r penes;
00413                     double denom=2.0;
00414                     for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
00415                         AChannel *theChannel=theJGroup->GetChannel(ch);
00416                         double localEnergy=theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE()
00417                             -this->GetPair(theChannel->GetPairNum())->GetSepE();
00418                         double radius=this->GetPair(theChannel->GetPairNum())->GetChRad();
00419                         if(theChannel->GetRadType()=='P') {
00420                             if(localEnergy>0.0) {
00421                                 if(theLevel->GetGamma(ch)<0.0) isNegative.push_back(true);
00422                                 else isNegative.push_back(false);
00423                                 tempGammas.push_back(fabs(theLevel->GetGamma(ch))/1e6);
00424                                 CoulFunc theCoulombFunction(this->GetPair(theChannel->GetPairNum()),
00425                                     !(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
00426                                 double tempPene=theCoulombFunction.Penetrability(theChannel->GetL(),
00427                                     radius,
00428                                     localEnergy);
00429                                 denom-=tempGammas[ch-1]/tempPene*
00430                                     theCoulombFunction.PEShift_dE(theChannel->GetL(),radius,localEnergy);
00431                                 penes.push_back(tempPene);
00432                             } else {

```

```

00433         if(theLevel->GetGamma(ch)<0.0) isNegative.push_back(true);
00434     else isNegative.push_back(false);
00435     tempGammas.push_back(pow(theLevel->GetGamma(ch),2.0));
00436     ShftFunc theShiftFunction(this->GetPair(theChannel->GetPairNum()));
00437     WhitFunc newWhitFunc(this->GetPair(theChannel->GetPairNum()));
00438     double whitConv=newWhitFunc(theChannel->GetL(),radius,fabs(localEnergy));
00439     double tempPene=this->GetPair(theChannel->GetPairNum())->GetRedMass()*radius*uconv/
00440         pow(hbarc,2.0)/pow(whitConv,2.0);
00441     denom-=tempGammas[ch-1]/tempPene*
00442         theShiftFunction.EnergyDerivative(theChannel->GetL(),theLevel->GetE());
00443     penes.push_back(tempPene);
00444 }
00445 } else if(theChannel->GetRadType()=='E' || theChannel->GetRadType()=='M') {
00446     if(fabs(theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE())<1.e-3&&
00447         theJGroup->GetJ()==this->GetPair(theChannel->GetPairNum())->GetJ(2)&&
00448         theJGroup->GetPi()==this->GetPair(theChannel->GetPairNum())->GetPi(2)) {
00449         int tempSign;
00450         if(theLevel->GetGamma(ch)<0) tempSign=-1;
00451         else tempSign=1;
00452         double jValue=theJGroup->GetJ();
00453         if(int(2.*jValue)%2!=0) tempSign=-tempSign;
00454         double tempPene=1e-10;
00455         double tempGamma=theLevel->GetGamma(ch);
00456         if(theChannel->GetRadType()=='M' && theChannel->GetL()==1) {
00457             tempPene=3.0*jValue/4.0/(jValue+1.)/nuclearMagneton/nuclearMagneton;
00458         } else if(theChannel->GetRadType()=='E' && theChannel->GetL()==2) {
00459             tempPene=60.0*jValue*(2.*jValue-1.)/(jValue+1.)/(2.*jValue+3.);
00460             tempGamma=tempGamma*100*sqrt(fstruc+hbarc);
00461         }
00462         tempGammas.push_back(pow(tempGamma,2.0));
00463         penes.push_back(tempPene);
00464         if(tempSign<0) isNegative.push_back(true);
00465         else isNegative.push_back(false);
00466     } else {
00467         if(theLevel->GetGamma(ch)<0.0) isNegative.push_back(true);
00468         else isNegative.push_back(false);
00469         tempGammas.push_back(fabs(theLevel->GetGamma(ch))/1e6);
00470         double tempPene = (configure.paramMask & Config::USE_RMC_FORMALISM) ? 1.0 :
00471             pow(fabs(localEnergy)/hbarc,2.0*theChannel->GetL()+1);
00472         if(tempPene<1e-16) tempPene=1e-16;
00473         penes.push_back(tempPene);
00474     } else if(theChannel->GetRadType()=='F' || theChannel->GetRadType()=='G') {
00475         if(theLevel->GetGamma(ch)<0.0) isNegative.push_back(true);
00476         else isNegative.push_back(false);
00477         tempGammas.push_back(fabs(theLevel->GetGamma(ch)));
00478         penes.push_back(1.0);
00479     }
00480 }
00481 if(denom<0.) configure.outStream << "WARNING: Denominator less than zero in E="
00482     << theLevel->GetE() << " MeV resonance transformation. "
00483     << "Transformation may not have been successful."
00484     << std::endl;
00485 double nFSum=1.0;
00486 for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
00487     AChannel *theChannel=theJGroup->GetChannel(ch);
00488     if(theChannel->GetRadType()!='F' && theChannel->GetRadType()!='G')
00489         tempGammas[ch-1]=sqrt(fabs(tempGammas[ch-1]/penes[ch-1]/denom));
00490     if(isNegative[ch-1]) tempGammas[ch-1]=-tempGammas[ch-1];
00491     theLevel->SetGamma(ch,tempGammas[ch-1]);
00492     if(ch<=theLevel->NumNFIntegrals()) nFSum+=2.0*
00493         this->GetPair(theChannel->GetPairNum())->GetChRad()*
00494         this->GetPair(theChannel->GetPairNum())->GetRedMass()*
00495         uconv/pow(hbarc,2.0)*pow(tempGammas[ch-1],2.0)*theLevel->GetNFIntegral(ch);
00496 }
00497 theLevel->SetSqrtNFFactor(1.0/sqrt(nFSum));
00498 }
00499 }
00500 }
00501 }
00502 for(int j=1;j<=this->NumJGroups();j++) {
00503     JGroup *theJGroup=this->GetJGroup(j);
00504     if(theJGroup->IsInRMatrix()) {
00505         std::vector<int> levelKeys;
00506         vector_r tempEnergies;
00507         matrix_r tempGammas;
00508         matrix_r shifts;
00509         for(int la=1;la<=theJGroup->NumLevels();la++) {
00510             ALevel *theLevel=theJGroup->GetLevel(la);
00511             if(theLevel->IsInRMatrix()) {
00512                 levelKeys.push_back(la);
00513                 tempEnergies.push_back(theLevel->GetE());
00514                 vector_r tempChanVector;
00515                 tempGammas.push_back(tempChanVector);
00516                 shifts.push_back(tempChanVector);
00517             }
00518             for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
00519                 AChannel *theChannel=theJGroup->GetChannel(ch);

```

```

00519     double localEnergy=theLevel->GetE()-this->GetPair(theChannel->GetPairNum()->GetExE()
00520     -this->GetPair(theChannel->GetPairNum()->GetSepE());
00521     double radius=this->GetPair(theChannel->GetPairNum()->GetChRad());
00522     if(theChannel->GetRadType()=='P') {
00523         if(localEnergy>0.0) {
00524             tempGammas[levelKeys.size()-1].push_back(theLevel->GetGamma(ch));
00525             CoulFunc theCoulombFunction(this->GetPair(theChannel->GetPairNum()),
00526             !(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
00527             shifts[levelKeys.size()-1].push_back(theCoulombFunction.PEShift(theChannel->GetL(),
00528             radius,
00529             localEnergy));
00530         } else {
00531             tempGammas[levelKeys.size()-1].push_back(theLevel->GetGamma(ch));
00532             ShiftFunc theShiftFunction(this->GetPair(theChannel->GetPairNum()));
00533             shifts[levelKeys.size()-1].push_back(theShiftFunction(theChannel->GetL(),theLevel->GetE()));
00534         }
00535     } else {
00536         tempGammas[levelKeys.size()-1].push_back(theLevel->GetGamma(ch));
00537         if((theChannel->GetRadType()=='E' || theChannel->GetRadType()=='M') &&
00538         (configure.paramMask & Config::USE_EXTERNAL_CAPTURE) &&
00539         !(fabs(theLevel->GetGamma(ch))<1.0e-8 &&
00540         (configure.paramMask & Config::IGNORE_ZERO_WIDTHS))) {
00541             complex externalWidth =
00542             CalcExternalWidth(theJGroup,theLevel,theChannel,true,configure);
00543             if(pow(tempGammas[levelKeys.size()-1][ch-1],2.0)>=pow(imag(externalWidth),2.0)) {
00544                 if(tempGammas[levelKeys.size()-1][ch-1]<0.0)
00545                     tempGammas[levelKeys.size()-1][ch-1]=-sqrt(pow(tempGammas[levelKeys.size()-1][ch-1],2.0)-
00546                     pow(imag(externalWidth),2.0))-real(externalWidth);
00547             } else
00548                 tempGammas[levelKeys.size()-1][ch-1]=sqrt(pow(tempGammas[levelKeys.size()-1][ch-1],2.0)-
00549                 pow(imag(externalWidth),2.0))-real(externalWidth);
00550         } else {
00551             configure.outStream << "***WARNING: Imaginary portion of external width \n\tfor j=" << j << "
00552             la="
00553                 << la << " ch=" << ch << " is greater than total width." << std::endl;
00554             tempGammas[levelKeys.size()-1][ch-1]=-real(externalWidth);
00555         }
00556         shifts[levelKeys.size()-1].push_back(shifts[levelKeys.size()-1][0]);
00557     }
00558 }
00559 }
00560 if(!(configure.paramMask & Config::USE_BRUNE_FORMALISM)) {
00561     matrix_r nMatrix;
00562     matrix_r mMatrix;
00563     for(int mu=0;mu<tempEnergies.size();mu++) {
00564         vector_r tempLevelVector;
00565         nMatrix.push_back(tempLevelVector);
00566         mMatrix.push_back(tempLevelVector);
00567         for(int la=0;la<tempEnergies.size();la++) {
00568             if(la==mu) {
00569                 mMatrix[mu].push_back(1.0);
00570                 double sum=tempEnergies[la];
00571                 for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
00572                     if(theJGroup->GetChannel(ch)->GetRadType()=='P')
00573                         sum+=(shifts[la][ch-1]-theJGroup->GetChannel(ch)->GetBoundaryCondition()) *
00574                         pow(tempGammas[la][ch-1],2.0);
00575                 }
00576                 nMatrix[mu].push_back(sum);
00577             } else {
00578                 double mSum=0.0;
00579                 double nSum=0.0;
00580                 for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
00581                     if(theJGroup->GetChannel(ch)->GetRadType()=='P') {
00582                         mSum+=(shifts[mu][ch-1]-shifts[la][ch-1]) / (tempEnergies[mu]-tempEnergies[la]) *
00583                         tempGammas[la][ch-1]*tempGammas[mu][ch-1];
00584                         nSum+=(tempEnergies[mu]*shifts[la][ch-1]-tempEnergies[la]*shifts[mu][ch-1]) /
00585                         (tempEnergies[mu]-tempEnergies[la]) -theJGroup->GetChannel(ch)->GetBoundaryCondition()
00586                         *tempGammas[la][ch-1]*tempGammas[mu][ch-1];
00587                     }
00588                 }
00589                 mMatrix[mu].push_back(-mSum);
00590                 nMatrix[mu].push_back(nSum);
00591             }
00592         }
00593     }
00594     //solve eigenvalue problem
00595     EigenFunc eigenFunc(nMatrix,mMatrix);
00596     for(int la=0;la<tempEnergies.size();la++) {
00597         theJGroup->GetLevel(levelKeys[la])->SetE(eigenFunc.eigenvalues()[la]);
00598         for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
00599             double sum=0.0;
00600             for(int mu=0;mu<tempEnergies.size();mu++) {
00601                 sum+=eigenFunc.eigenvectors()[mu][la]*tempGammas[mu][ch-1];
00602             }
00603             theJGroup->GetLevel(levelKeys[la])->SetGamma(ch,sum);

```



```

00604     }
00605     }
00606     } else {
00607     for(int la=0;la<tempEnergies.size();la++)
00608     for(int ch=1;ch<=theJGroup->NumChannels();ch++)
00609     theJGroup->GetLevel(levelKeys[la])>SetGamma(ch,tempGammas[la][ch-1]);
00610     }
00611     }
00612     }
00613 }
00614
00619 void CNuc::SortPathways(const Config& configure) {
00620     int DecayNum, KGroupNum, MGroupNum;
00621     for(int aa=1;aa<=this->NumPairs();aa++) {
00622         if(!this->GetPair(aa)->IsEntrance()) continue;
00623         for(int ir=1;ir<=this->NumPairs();ir++) {
00624             if(this->GetPair(ir)->GetPType()==20) continue;
00625             if(this->GetPair(aa)->GetPType()==20) {
00626                 for(int l = 0; l < 2; l++) {
00627                     for(int j=1;j<=this->NumJGroups();j++) {
00628                         if(!this->GetJGroup(j)->IsInRMatrix()) continue;
00629                         for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
00630                             if(this->GetJGroup(j)->GetChannel(ch)->GetPairNum()!=aa) continue;
00631                             for(int chp=1;chp<=this->GetJGroup(j)->NumChannels();chp++) {
00632                                 if(this->GetJGroup(j)->GetChannel(chp)->GetPairNum()!=ir||
00633                                    this->GetJGroup(j)->GetChannel(ch)->GetL()!=l) continue;
00634                                 Decay NewDecay(ir);
00635                                 DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
00636                                 if(!DecayNum) {
00637                                     this->GetPair(aa)->AddDecay(NewDecay);
00638                                     DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
00639                                 }
00640                                 KGroup NewKGroup(l,0);
00641                                 KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
00642                                 if(!KGroupNum) {
00643                                     this->GetPair(aa)->GetDecay(DecayNum)->AddKGroup(NewKGroup);
00644                                     KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
00645                                 }
00646                                 MGroup NewMGroup(j,ch,chp);
00647                                 MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
00648                                 if(!MGroupNum) {
00649                                     this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->AddMGroup(NewMGroup);
00650                                     MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
00651                                 }
00652                             }
00653                         }
00654                     }
00655                 }
00656             } else if(this->GetPair(ir)->GetPType()==0) {
00657                 for(double s=fabs(this->GetPair(aa)->GetJ(1)-this->GetPair(aa)->GetJ(2));
00658                    s<=(this->GetPair(aa)->GetJ(1)+this->GetPair(aa)->GetJ(2));s+=1.) {
00659                     for(double sp=fabs(this->GetPair(ir)->GetJ(1)-this->GetPair(ir)->GetJ(2));
00660                        sp<=(this->GetPair(ir)->GetJ(1)+this->GetPair(ir)->GetJ(2));sp+=1.) {
00661                         for(int j=1;j<=this->NumJGroups();j++) {
00662                             if(!this->GetJGroup(j)->IsInRMatrix()) continue;
00663                             for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
00664                                 if(this->GetJGroup(j)->GetChannel(ch)->GetPairNum()!=aa) continue;
00665                                 for(int chp=1;chp<=this->GetJGroup(j)->NumChannels();chp++) {
00666                                     if(this->GetJGroup(j)->GetChannel(chp)->GetPairNum()!=ir||
00667                                        this->GetJGroup(j)->GetChannel(ch)->GetS()!=s||
00668                                        this->GetJGroup(j)->GetChannel(chp)->GetS()!=sp) continue;
00669                                     Decay NewDecay(ir);
00670                                     DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
00671                                     if(!DecayNum) {
00672                                         this->GetPair(aa)->AddDecay(NewDecay);
00673                                         DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
00674                                     }
00675                                     KGroup NewKGroup(s,sp);
00676                                     KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
00677                                     if(!KGroupNum) {
00678                                         this->GetPair(aa)->GetDecay(DecayNum)->AddKGroup(NewKGroup);
00679                                         KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
00680                                     }
00681                                     MGroup NewMGroup(j,ch,chp);
00682                                     MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
00683                                     if(!MGroupNum) {
00684                                         this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->AddMGroup(NewMGroup);
00685                                         MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
00686                                     }
00687                                     double statspinfactor=(2.*this->GetJGroup(j)->GetJ()+1.)*
00688                                         this->GetPair(this->GetJGroup(j)->GetChannel(chp)->GetPairNum())->GetI1I2Factor();
00689                                     this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->
00690                                         GetMGroup(MGroupNum)->SetStatSpinFactor(statspinfactor);
00691                                 }
00692                             }
00693                         }

```



```

00694     }
00695     }
00696     } else if(this->GetPair(ir)->GetType()==10 && !(configure.paramMask &
Config::USE_RMC_FORMALISM)) {
00697     for(double s=fabs(this->GetPair(aa)->GetJ(1)-this->GetPair(aa)->GetJ(2));
00698     s<=(this->GetPair(aa)->GetJ(1)+this->GetPair(aa)->GetJ(2));s+=1.) {
00699     for(int j=1;j<=this->NumJGroups();j++) {
00700     if(!this->GetJGroup(j)->IsInRMatrix()) continue;
00701     for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
00702     if(this->GetJGroup(j)->GetChannel(ch)->GetPairNum()!=aa) continue;
00703     for(int chp=1;chp<=this->GetJGroup(j)->NumChannels();chp++) {
00704     if(this->GetJGroup(j)->GetChannel(chp)->GetPairNum()!=ir||
00705     this->GetJGroup(j)->GetChannel(ch)->GetS()!=s) continue;
00706     Decay NewDecay(ir);
00707     DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
00708     if(!DecayNum) {
00709     this->GetPair(aa)->AddDecay(NewDecay);
00710     DecayNum=this->GetPair(aa)->IsDecay(NewDecay);
00711     }
00712     KGroup NewKGroup(s,0);
00713     KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
00714     if(!KGroupNum) {
00715     this->GetPair(aa)->GetDecay(DecayNum)->AddKGroup(NewKGroup);
00716     KGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->IsKGroup(NewKGroup);
00717     }
00718     MGroup NewMGroup(j,ch,chp);
00719     MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
00720     if(!MGroupNum) {
00721     this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->AddMGroup(NewMGroup);
00722     MGroupNum=this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->IsMGroup(NewMGroup);
00723     }
00724     double statspinfactor=(2.*this->GetJGroup(j)->GetJ()+1.)*
00725     this->GetPair(this->GetJGroup(j)->GetChannel(chp)->GetPairNum())->GetI1I2Factor();
00726     this->GetPair(aa)->GetDecay(DecayNum)->GetKGroup(KGroupNum)->
00727     GetMGroup(MGroupNum)->SetStatSpinFactor(statspinfactor);
00728     }
00729     }
00730     }
00731     }
00732     }
00733     }
00734     }
00735     for(int aa=1;aa<=this->NumPairs();aa++) { //loop over all pairs
00736     PPair *entrancePair=this->GetPair(aa);
00737     if(entrancePair->GetType()==20 || !entrancePair->IsEntrance()) continue;
00738     for(int j=1;j<=this->NumJGroups();j++) {
00739     JGroup *theFinalJGroup=this->GetJGroup(j);
00740     for(int la=1;la<=theFinalJGroup->NumLevels();la++) {
00741     ALevel *theFinalLevel=theFinalJGroup->GetLevel(la);
00742     if(!theFinalLevel->IsECLLevel()) continue;
00743     int decayNum=entrancePair->IsDecay(theFinalLevel->GetECPairNum()); //store RESONANCE decay number
to final state
00744     if(!decayNum) continue; //if this is a resonance decay...
00745     for(int k=1;k<=entrancePair->GetDecay(decayNum)->NumKGroups();k++) { //loop over all kgroups for
decays to final state
00746     KGroup *theKGroup=entrancePair->GetDecay(decayNum)->GetKGroup(k);
00747     for(int chp=1;chp<=theFinalJGroup->NumChannels();chp++) { //loop over all final configurations
in the capture state
00748     AChannel *finalChannel=theFinalJGroup->GetChannel(chp);
00749     if(this->GetPair(finalChannel->GetPairNum())->GetType()!=0) continue; //ensure the
configuration is a particle pair
00750     int chDecayNum=entrancePair->IsDecay(finalChannel->GetPairNum());
00751     if(!chDecayNum) continue; //if it is actually a resonance decay...
00752     for(int kp=1;kp<=entrancePair->GetDecay(chDecayNum)->NumKGroups();kp++) {
00753     if(entrancePair->GetDecay(chDecayNum)->GetKGroup(kp)->GetS()!=theKGroup->GetS()) continue;
00754     for(int mp=1;mp<=entrancePair->GetDecay(chDecayNum)->GetKGroup(kp)->NumMGroups();mp++) {
00755     MGroup *chMGroup=entrancePair->GetDecay(chDecayNum)->GetKGroup(kp)->GetMGroup(mp);
00756     AChannel *chChannel=this->GetJGroup(chMGroup->GetJNum())->GetChannel(chMGroup->GetChNum());
00757     AChannel *chChannelp=this->GetJGroup(chMGroup->GetJNum())->GetChannel(chMGroup->GetChpNum());
00758     for(int multL=1;multL<=maxECMult;multL++) { //loop over all allowed gamma parities
00759     char radType;
00760     if(this->GetJGroup(chMGroup->GetJNum())->GetPi()*theFinalJGroup->GetPi()==(int)pow(-1,multL))
radType='E';
00761     else radType='M'; //calculate radiation type
00762     if(!((radType=='E' && multL==1) && (theFinalLevel->GetECMultMask()&isE1)) &&
00763     !((radType=='M' && multL==1) && (theFinalLevel->GetECMultMask()&isM1)) &&
00764     !((radType=='E' && multL==2) && (theFinalLevel->GetECMultMask()&isE2)) ) continue;
//allow only m1,e1,e2
00765     if(fabs(this->GetJGroup(chMGroup->GetJNum())->GetJ()-multL)>theFinalJGroup->GetJ()||
00766     theFinalJGroup->GetJ()>this->GetJGroup(chMGroup->GetJNum())->GetJ()+multL) continue;
00767     if(!(abs(chChannelp->GetL()-multL)<=finalChannel->GetL())&&
00768     finalChannel->GetL()<=chChannelp->GetL()+multL&&
00769     fabs(chChannelp->GetS()-finalChannel->GetL())<=theFinalJGroup->GetJ()&&
00770     theFinalJGroup->GetJ()<=chChannelp->GetS()+finalChannel->GetL()&&
00771     chChannelp->GetS()==finalChannel->GetS())&&
00772     !(fabs(chChannelp->GetS()-multL)<=finalChannel->GetS())&&

```

```

00773         finalChannel->GetS() <= chChannel->GetS() + multL &&
00774         fabs(chChannel->GetL() - finalChannel->GetS()) <= theFinalJGroup->GetJ() &&
00775         theFinalJGroup->GetJ() <= chChannel->GetL() + finalChannel->GetS() &&
00776         chChannel->GetL() == finalChannel->GetL() &&
00777         radType == 'M') continue; //ensure entrance channel for dc can couple to final state

00778     if(chChannel == chChannelp) {
00779         ECMGroup newECMGroup(radType, multL, chChannel->GetL(),
00780             this->GetJGroup(chMGroup->GetJNum())->GetJ(), chp, j, la);
00781         theKGroup->AddECMGroup(newECMGroup);
00782     }
00783     int internalChannel = 0;
00784     for(int intCh = 1; intCh <= this->GetJGroup(chMGroup->GetJNum())->NumChannels(); intCh++) {
00785         if(this->GetJGroup(chMGroup->GetJNum())->GetChannel(intCh)->GetRadType() == radType &&
00786             this->GetJGroup(chMGroup->GetJNum())->GetChannel(intCh)->GetL() == multL &&
00787             this->GetJGroup(chMGroup->GetJNum())->GetChannel(intCh)->GetPairNum() == theFinalLevel->GetECPairNum())
00788     {
00789         internalChannel = intCh;
00790         break;
00791     }
00792     ECMGroup
newECMGroup(radType, multL, chChannel->GetL(), this->GetJGroup(chMGroup->GetJNum())->GetJ(),
00793         chp, j, la, chDecayNum, kp, mp, internalChannel);
00794     theKGroup->AddECMGroup(newECMGroup);
00795 }
00796 }
00797 }
00798 }
00799 }
00800 }
00801 }
00802 }
00803 }
00804
00809 void CNuc::PrintPathways(const Config &configure) {
00810     std::streambuf *sbuffer;
00811     std::filebuf fbuffer;
00812     if(configure.fileCheckMask & Config::CHECK_PATHWAYS) {
00813         std::string outfile = configure.checkdir + "pathways.chk";
00814         fbuffer.open(outfile.c_str(), std::ios::out);
00815         sbuffer = &fbuffer;
00816     } else if(configure.screenCheckMask & Config::CHECK_PATHWAYS) sbuffer = configure.outStream.rdbuf();
00817     std::ostream out(sbuffer);
00818     if(((configure.fileCheckMask & Config::CHECK_PATHWAYS) && fbuffer.is_open())
00819         || (configure.screenCheckMask & Config::CHECK_PATHWAYS)) {
00820         out << std::endl
00821         << "*****" << std::endl
00822         << " * Internal Reaction Pathways * " << std::endl
00823         << "*****" << std::endl
00824         << std::setw(17) << "Entrance Pair #"
00825         << std::setw(9) << "Decay #"
00826         << std::setw(14) << "Decay Pair #"
00827         << std::setw(12) << "K Group #"
00828         << std::setw(16) << "Entrance Ch. s"
00829         << std::setw(13) << "Decay Ch. s"
00830         << std::setw(11) << "M Group #"
00831         << std::setw(11) << "J Group #"
00832         << std::setw(16) << "Entrance Ch. #"
00833         << std::setw(12) << "Exit Ch. #" << std::endl;
00834         for(int i = 1; i <= this->NumPairs(); i++) {
00835             PPair *thePair = this->GetPair(i);
00836             for(int ii = 1; ii <= this->GetPair(i)->NumDecays(); ii++) {
00837                 for(int iii = 1; iii <= this->GetPair(i)->GetDecay(ii)->NumKGroups(); iii++) {
00838                     for(int iiiii = 1; iiiii <= this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->NumMGroups(); iiiii++) {
00839
00840                         out << std::setw(17) << i
00841                         << std::setw(9) << ii
00842                         << std::setw(14) << this->GetPair(i)->GetDecay(ii)->GetPairNum()
00843                         << std::setw(12) << iii
00844                         << std::setw(16) << this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetS()
00845                         << std::setw(13) << this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetSp()
00846                         << std::setw(11) << iiiii
00847                         << std::setw(11) << this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetMGroup(iiii)->GetJNum()
00848                         << std::setw(16) << this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetMGroup(iiii)->GetChNum()
00849                         << std::setw(12) << this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetMGroup(iiii)->GetChpNum()
00850                     << std::endl;
00851                 }
00852             }
00853         }
00854         out << std::endl
00855         << "*****" << std::endl
00856         << " * External Reaction Pathways * " << std::endl
00857         << "*****" << std::endl

```

```

00858     « std::setw(17) « "Entrance Pair #"
00859     « std::setw(9)  « "Decay #"
00860     « std::setw(14) « "Decay Pair #"
00861     « std::setw(12) « "K Group #"
00862     « std::setw(16) « "Entrance Ch. s"
00863     « std::setw(13) « "Decay Ch. s"
00864     « std::setw(11) « "M Group #"
00865     « std::setw(11) « "Mult."
00866     « std::setw(11) « "J_i Value"
00867     « std::setw(11) « "J_f Value"
00868     « std::setw(11) « "l_i Value"
00869     « std::setw(11) « "l_f Value"
00870     « std::setw(13) « "Type"
00871     « std::setw(13) « "Ch. Decay #"
00872     « std::setw(11) « "Ch. K #"
00873     « std::setw(11) « "Ch. M #"
00874     « std::setw(11) « "Int. Ch #"
00875     « std::endl;
00876     for(int i=1;i<=this->NumPairs();i++) {
00877         PPair *thePair=this->GetPair(i);
00878         for(int ii=1;ii<=this->GetPair(i)->NumDecays();ii++){
00879             for(int iii=1;iii<=this->GetPair(i)->GetDecay(ii)->NumKGroups();iii++) {
00880                 for(int iiiii=1;iiiii<=this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->NumECMGroups();iiiii++) {
00881                     ECMGroup *theECMGroup=this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetECMGroup(iiii);
00882                     JGroup *theECJGroup=this->GetJGroup(theECMGroup->GetJGroupNum());
00883                     out « std::setw(17) « i
00884                     « std::setw(9) « ii
00885                     « std::setw(14) « this->GetPair(i)->GetDecay(ii)->GetPairNum()
00886                     « std::setw(12) « iii
00887                     « std::setw(16) « this->GetPair(i)->GetDecay(ii)->GetKGroup(iii)->GetS()
00888                     « std::setw(13) « theECJGroup->GetChannel(theECMGroup->GetFinalChannel())->GetS()
00889                     « std::setw(11) « iiiii
00890                     « std::setw(10) « theECMGroup->GetRadType() « theECMGroup->GetMult()
00891                     « std::setw(11) « theECMGroup->GetJ()
00892                     « std::setw(11) « theECJGroup->GetJ()
00893                     « std::setw(11) « theECMGroup->GetL()
00894                     « std::setw(11) « theECJGroup->GetChannel(theECMGroup->GetFinalChannel())->GetL();
00895                     if(theECMGroup->IsChannelCapture()) out « std::setw(13) « "Channel"
00896                     « std::setw(13) « theECMGroup->GetChanCapDecay()
00897                     « std::setw(11) « theECMGroup->GetChanCapKGroup()
00898                     « std::setw(11) « theECMGroup->GetChanCapMGroup()
00899                     « std::setw(11) « theECMGroup->GetIntChannelNum() « std::endl;
00900                     else out « std::setw(15) « "Hard Sphere" « std::endl;
00901                 }
00902             }
00903             out « std::endl;
00904         }
00905     }
00906 } else configure.outStream « "Could not write pathways check file." « std::endl;
00907 out.flush();
00908 if(fbuffer.is_open()) fbuffer.close();
00909 }
00910
00916 void CNuc::CalcBoundaryConditions(const Config& configure){
00917     for(int j=1;j<=this->NumJGroups();j++) {
00918         if(this->GetJGroup(j)->IsInRMatrix()) {
00919             JGroup *theJGroup=this->GetJGroup(j);
00920             ALevel *firstLevel=theJGroup->GetLevel(1);
00921             if(firstLevel->IsInRMatrix()) {
00922                 for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
00923                     AChannel *theChannel=theJGroup->GetChannel(ch);
00924                     PPair *thePair=this->GetPair(theChannel->GetPairNum());
00925                     if(thePair->GetPType()==0) {
00926                         int lValue=theChannel->GetL();
00927                         double levelEnergy=firstLevel->GetE();
00928                         double resonanceEnergy=levelEnergy- (thePair->GetSepE()+thePair->GetExE());
00929                         if(resonanceEnergy<0.0) {
00930                             ShiftFunc theShiftFunction(thePair);
00931                             theChannel->SetBoundaryCondition(theShiftFunction(lValue,levelEnergy));
00932                         }
00933                     }
00934                     else {
00935                         CoulFunc theCoulombFunction(thePair,
00936                             !(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
00937                         double radius=thePair->GetChRad();
00938                         double boundary=theCoulombFunction.PEShift(lValue,radius,resonanceEnergy);
00939                         theChannel->SetBoundaryCondition(boundary);
00940                     }
00941                 }
00942             }
00943             else {
00944                 double boundary=theJGroup->GetChannel(1)->GetBoundaryCondition();
00945                 theChannel->SetBoundaryCondition(boundary);
00946             }
00947         }
00948     }
00949 }

```

```

00950
00955 void CNuc::PrintBoundaryConditions(const Config &configure) {
00956     std::streambuf *sbuffer;
00957     std::filebuf fbuffer;
00958     if (configure.fileCheckMask & Config::CHECK_BOUNDARY_CONDITIONS) {
00959         std::string outfile=configure.checkdir+"boundaryconditions.chk";
00960         fbuffer.open(outfile.c_str(),std::ios::out);
00961         sbuffer = &fbuffer;
00962     } else if (configure.screenCheckMask & Config::CHECK_BOUNDARY_CONDITIONS) sbuffer =
configure.outStream.rdbuf();
00963     std::ostream out(sbuffer);
00964     if (((configure.fileCheckMask & Config::CHECK_BOUNDARY_CONDITIONS)&&fbuffer.is_open()) ||
00965         (configure.screenCheckMask & Config::CHECK_BOUNDARY_CONDITIONS)) {
00966         out << std::endl
00967             << "*****" << std::endl
00968             << "          Boundary Conditions          " << std::endl
00969             << "*****" << std::endl;
00970         out << std::setw(10) << "J Group #"
00971             << std::setw(10) << "Channel #"
00972             << std::setw(20) << "Boundary Condition"
00973             << std::endl;
00974         for (int j=1;j<=this->NumJGroups();j++) {
00975             if (this->GetJGroup(j)->IsInRMatrix()) {
00976                 JGroup *theJGroup=this->GetJGroup(j);
00977                 for (int ch=1;ch<=theJGroup->NumChannels();ch++) {
00978                     AChannel *theChannel=theJGroup->GetChannel(ch);
00979                     out << std::setw(10) << j
00980                         << std::setw(10) << ch
00981                         << std::setw(20) << theChannel->GetBoundaryCondition()
00982                         << std::endl;
00983                 }
00984             }
00985         }
00986     } else configure.outStream << "Could not write boundary conditions check file." << std::endl;
00987     out.flush();
00988     if (fbuffer.is_open()) fbuffer.close();
00989 }
00990
00995 void CNuc::CalcAngularDists(int maxL) {
00996     for (int aa=1;aa<=this->NumPairs();aa++) {
00997         PPair *entrancePair=this->GetPair(aa);
00998         if (entrancePair->GetType()==20) continue;
00999         for (int ir=1;ir<=this->GetPair(aa)->NumDecays();ir++) {
01000             Decay *theDecay=this->GetPair(aa)->GetDecay(ir);
01001             for (int k=1;k<=theDecay->NumKGroups();k++) {
01002                 for (int lOrder=0;lOrder<=maxL;lOrder++) {
01003                     for (int
m1=1;m1<=theDecay->GetKGroup(k)->NumMGroups()+theDecay->GetKGroup(k)->NumECMGroups();m1++) {
01004                         for (int
m2=1;m2<=theDecay->GetKGroup(k)->NumMGroups()+theDecay->GetKGroup(k)->NumECMGroups();m2++) {
01005                             std::string interferenceType;
01006                             double j1,j2,l1,l1p,l2,l2p;
01007                             int wlp,w2p,path1,path2;
01008                             if (m1>theDecay->GetKGroup(k)->NumMGroups()) {
01009                                 int m1_ec=m1-theDecay->GetKGroup(k)->NumMGroups();
01010                                 ECMGroup *theECMGroup1=theDecay->GetKGroup(k)->GetECMGroup(m1_ec);
01011                                 j1=theECMGroup1->GetJ();
01012                                 l1=(double) theECMGroup1->GetL();
01013                                 l1p=(double) theECMGroup1->GetMult();
01014                                 if (theECMGroup1->GetRadType()=='M') wlp=0;
01015                                 else wlp=1;
01016                                 interferenceType='E';
01017                                 path1=m1_ec;
01018                             } else {
01019                                 JGroup *jgroup1=this->GetJGroup(theDecay->GetKGroup(k)->GetMGroup(m1)->GetJNum());
01020                                 AChannel *channell1=jgroup1->GetChannel(theDecay->GetKGroup(k)->GetMGroup(m1)->GetChNum());
01021                                 AChannel *channellp=jgroup1->GetChannel(theDecay->GetKGroup(k)->GetMGroup(m1)->GetChpNum());
01022                                 j1=jgroup1->GetJ();
01023                                 l1=(double) channell1->GetL();
01024                                 l1p=(double) channellp->GetL();
01025                                 if (channellp->GetRadType()=='M' || channellp->GetRadType()=='P') wlp=0;
01026                                 else wlp=1;
01027                                 interferenceType='R';
01028                                 path1=m1;
01029                             }
01030                             if (m2>theDecay->GetKGroup(k)->NumMGroups()) {
01031                                 int m2_ec=m2-theDecay->GetKGroup(k)->NumMGroups();
01032                                 ECMGroup *theECMGroup2=theDecay->GetKGroup(k)->GetECMGroup(m2_ec);
01033                                 j2=theECMGroup2->GetJ();
01034                                 l2=(double) theECMGroup2->GetL();
01035                                 l2p=(double) theECMGroup2->GetMult();
01036                                 if (theECMGroup2->GetRadType()=='M') w2p=0;
01037                                 else w2p=1;
01038                                 interferenceType+="'E'";
01039                                 path2=m2_ec;
01040                             } else {
01041                                 JGroup *jgroup2=this->GetJGroup(theDecay->GetKGroup(k)->GetMGroup(m2)->GetJNum());

```

```

01042     AChannel *channel12=jgroup2->GetChannel (theDecay->GetKGroup(k)->GetMGroup(m2)->GetChNum());
01043     AChannel *channel12p=jgroup2->GetChannel (theDecay->GetKGroup(k)->GetMGroup(m2)->GetChpNum());
01044     j2=jgroup2->GetJ();
01045     l2=(double) channel12->GetL();
01046     l2p=(double) channel12p->GetL();
01047     if(channel12p->GetRadType()=='M' || channel12p->GetRadType()=='P') w2p=0;
01048     else w2p=1;
01049     interferenceType+='R';
01050     path2=m2;
01051 }
01052     double s=theDecay->GetKGroup(k)->GetS();
01053     double sp=theDecay->GetKGroup(k)->GetSp();
01054     if((int) (l1+l2+lOrder)%2==0&&(int) (l1p+l2p+w1p+w2p+lOrder)%2==0) {
01055     double z1z2=0.0;
01056     double z1=sqrt(2.*l1+1.)*sqrt(2.*l2+1.)*sqrt(2.*j1+1.)*sqrt(2.*j2+1.)
01057         *AngCoeff::ClebGord(l1,l2,lOrder,0.,0.,0.)*AngCoeff::Racah(l1,j1,l2,j2,s,lOrder);
01058     if(this->GetPair(theDecay->GetPairNum())->GetPType()==0) {
01059         double z2=sqrt(2.*l1p+1.)*sqrt(2.*l2p+1.)*sqrt(2.*j1+1.)*sqrt(2.*j2+1.)
01060             *AngCoeff::ClebGord(l1p,l2p,lOrder,0.,0.,0.)*AngCoeff::Racah(l1p,j1,l2p,j2,sp,lOrder);
01061         z1z2=pow(-1.0,sp-s)/4.*z1*z2;
01062     } else if(this->GetPair(theDecay->GetPairNum())->GetPType()==10) {
01063         double jf=this->GetPair(theDecay->GetPairNum())->GetJ(2);
01064         double z2=sqrt(2.*l1p+1.)*sqrt(2.*l2p+1.)*sqrt(2.*j1+1.)*sqrt(2.*j2+1.)
01065             *AngCoeff::ClebGord(l1p,l2p,lOrder,1.,-1.,0)*AngCoeff::Racah(l1p,j1,l2p,j2,jf,lOrder);
01066         z1z2=pow(-1.,1.+s-jf)/4.*z1*z2;
01067     }
01068     if(fabs(z1z2)>1e-10) {
01069         KLGroup NewKLGroup(k,lOrder);
01070         int KLGroupNum=theDecay->IsKLGroup(NewKLGroup);
01071         if(!KLGroupNum) {
01072             theDecay->AddKLGroup(NewKLGroup);
01073             KLGroupNum=theDecay->IsKLGroup(NewKLGroup);
01074         }
01075         Interference NewInterference(path1,path2,z1z2,interferenceType);
01076         int InterNum=theDecay->GetKLGroup(KLGroupNum)->IsInterference(NewInterference);
01077         if(!InterNum) {
01078             theDecay->GetKLGroup(KLGroupNum)->AddInterference(NewInterference);
01079             InterNum=theDecay->GetKLGroup(KLGroupNum)->IsInterference(NewInterference);
01080         }
01081     }
01082 }
01083 }
01084 }
01085 }
01086 }
01087 }
01088 }
01089 }
01090 }
01095 void CNuc::PrintAngularDists(const Config &configure) {
01096     std::streambuf *sbuffer;
01097     std::filebuf fbuffer;
01098     if(configure.fileCheckMask & Config::CHECK_ANGULAR_DISTS) {
01099         std::string outfile=configure.checkdir+"angulardistributions.chk";
01100         fbuffer.open(outfile.c_str(),std::ios::out);
01101         sbuffer = &fbuffer;
01102     } else if(configure.screenCheckMask & Config::CHECK_ANGULAR_DISTS) sbuffer =
configure.outStream.rdbuf();
01103     std::ostream out(sbuffer);
01104     if(((configure.fileCheckMask & Config::CHECK_ANGULAR_DISTS)&&fbuffer.is_open())||
configure.screenCheckMask & Config::CHECK_ANGULAR_DISTS) {
01105         out << std::endl
01106             << "*****" << std::endl
01107             << "      Angular Distributions      " << std::endl
01108             << "*****" << std::endl;
01109         out << std::setw(10) << "ir"
01110             << std::setw(10) << "k"
01111             << std::setw(10) << "L"
01112             << std::setw(10) << "m1"
01113             << std::setw(10) << "m2"
01114             << std::setw(10) << "z1z2"
01115             << std::setw(10) << "type"
01116             << std::endl;
01117         for(int aa=1;aa<=this->NumPairs();aa++) {
01118             for(int ir=1;ir<=this->GetPair(aa)->NumDecays();ir++) {
01119                 Decay *theDecay=this->GetPair(aa)->GetDecay(ir);
01120                 for(int kl=1;kl<=theDecay->NumKLGroups();kl++) {
01121                     KLGroup *theKLGroup=theDecay->GetKLGroup(kl);
01122                     for(int i=1;i<=theKLGroup->NumInterferences();i++){
01123                         Interference *theInter=theKLGroup->GetInterference(i);
01124                         out << std::setw(10) << theDecay->GetPairNum()
01125                             << std::setw(10) << theKLGroup->GetK()
01126                             << std::setw(10) << theKLGroup->GetLOrder()
01127                             << std::setw(10) << theInter->GetM1()
01128                             << std::setw(10) << theInter->GetM2()
01129                             << std::setw(10) << theInter->GetZ1Z2()
01130                             << std::setw(10) << theInter->GetInterferenceType()

```

```

01132         « std::endl;
01133     }
01134     out « std::endl;
01135 }
01136 }
01137 }
01138 } else configure.outStream « "Could not write angular distributions check file." « std::endl;
01139 out.flush();
01140 if(fbuffer.is_open()) fbuffer.close();
01141 }
01142
01147 void CNuc::FillMnParams(ROOT::Minuit2::MnUserParameters &p) {
01148     char varname[50];
01149     for(int j=1; j<=this->NumJGroups(); j++) {
01150         for(int la=1; la<=this->GetJGroup(j)->NumLevels(); la++) {
01151             ALevel *level=this->GetJGroup(j)->GetLevel(la);
01152             sprintf(varname, "j=%d_la=%d_energy", j, la);
01153             p.Add(varname, level->GetE(), 0.1*level->GetE());
01154             bool isUnbound=false;
01155             for(int ir=1; ir<=this->NumPairs(); ir++) {
01156                 PPair *pair=this->GetPair(ir);
01157                 if(pair->GetPType()==0 &&
01158                    level->GetE() > (pair->GetSepE() + pair->GetExE())) isUnbound=true;
01159             }
01160             if(!isUnbound) p.Fix(varname);
01161             if(level->EnergyFixed() && !p.Parameter(p.Index(varname)).IsFixed()) p.Fix(varname);
01162             for(int ch=1; ch<=this->GetJGroup(j)->NumChannels(); ch++) {
01163                 sprintf(varname, "j=%d_la=%d_ch=%d_rwa", j, la, ch);
01164                 p.Add(varname, level->GetGamma(ch), 0.1*level->GetGamma(ch));
01165                 if(level->GetGamma(ch)==0.0) p.Fix(varname);
01166                 if(level->ChannelFixed(ch) && !p.Parameter(p.Index(varname)).IsFixed()) p.Fix(varname);
01167             }
01168         }
01169     }
01170 }
01171
01176 void CNuc::FillCompoundFromParams(const vector_r &p) {
01177     int i=0;
01178     for(int j=1; j<=this->NumJGroups(); j++) {
01179         for(int la=1; la<=this->GetJGroup(j)->NumLevels(); la++) {
01180             ALevel *level=this->GetJGroup(j)->GetLevel(la);
01181             level->SetFitE(p[i]); i++;
01182             double nFSum=1.0;
01183             for(int ch=1; ch<=this->GetJGroup(j)->NumChannels(); ch++) {
01184                 level->SetFitGamma(ch, p[i]);
01185                 if(ch<=level->NumNFIntegrals()) nFSum+=2.0*
01186                     this->GetPair(this->GetJGroup(j)->GetChannel(ch)->GetPairNum())->GetChRad()*
01187                     this->GetPair(this->GetJGroup(j)->GetChannel(ch)->GetPairNum())->GetRedMass()*
01188                     uconv/pow(hbarc, 2.0)*pow(p[i], 2.0)*level->GetNFIntegral(ch);
01189             }
01190             i++;
01191             level->SetSqrtNFFactor(1.0/sqrt(nFSum));
01192         }
01193     }
01194 }
01195
01200 void CNuc::TransformOut(const Config& configure) {
01201     if(!(configure.paramMask & Config::USE_BRUNE_FORMALISM)) {
01202         int maxIterations=1000;
01203         double energyTolerance=1e-6;
01204         for(int j=1; j<=this->NumJGroups(); j++) {
01205             for(int la=1; la<=this->GetJGroup(j)->NumLevels(); la++) {
01206                 ALevel *theLevel=this->GetJGroup(j)->GetLevel(la);
01207                 if(theLevel->IsInRMatrix()) {
01208                     int iteration=1;
01209                     int thisLevel=0;
01210                     bool done=false;
01211                     vector_r tempE;
01212                     vector_r tempBoundary;
01213                     matrix_r tempGamma;
01214                     for(int lap=1; lap<=this->GetJGroup(j)->NumLevels(); lap++) {
01215                         if(this->GetJGroup(j)->GetLevel(lap)->IsInRMatrix()) {
01216                             tempE.push_back(this->GetJGroup(j)->GetLevel(lap)->GetFitE());
01217                             if(this->GetJGroup(j)->GetLevel(lap)==theLevel) thisLevel=tempE.size()-1;
01218                             vector_r tempChanVector;
01219                             tempGamma.push_back(tempChanVector);
01220                             for(int ch=1; ch<=this->GetJGroup(j)->NumChannels(); ch++) {
01221                                 tempGamma[tempE.size()-1].push_back(this->GetJGroup(j)->GetLevel(lap)->GetFitGamma(ch));
01222                                 if(tempE.size()==1)
01223                                     tempBoundary.push_back(this->GetJGroup(j)->GetChannel(ch)->GetBoundaryCondition());
01224                             }
01225                         }
01226                     }
01227                     while(iteration<=maxIterations && !done) {
01228                         vector_r boundaryDiff;
01229                         for(int ch=1; ch<=this->GetJGroup(j)->NumChannels(); ch++) {

```

```

01230         AChannel *theChannel=this->GetJGroup(j)->GetChannel(ch);
01231         PPair *exitPair=this->GetPair(theChannel->GetPairNum());
01232         double localEnergy=tempE[thisLevel]-exitPair->GetSepE()-exitPair->GetExE();
01233         if(theChannel->GetRadType()=='P') {
01234             if(localEnergy<0.0) {
01235                 ShiftFunc theShiftFunction(exitPair);
01236                 newBoundary=theShiftFunction(theChannel->GetL(),tempE[thisLevel]);
01237             }
01238             else {
01239                 CoulFunc theCoulombFunction(exitPair,
01240                     !! (configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
01241                 double radius=exitPair->GetChRad();
01242                 newBoundary=theCoulombFunction.PEShift(theChannel->GetL(),radius,localEnergy);
01243             }
01244             boundaryDiff.push_back(newBoundary-tempBoundary[ch-1]);
01245             tempBoundary[ch-1]=newBoundary;
01246             } else boundaryDiff.push_back(boundaryDiff[0]);
01247         }
01248         matrix_r cMatrix;
01249         for(int mu=0;mu<tempE.size();mu++) {
01250             vector_r tempRow;
01251             cMatrix.push_back(tempRow);
01252             for(int mup=0;mup<tempE.size();mup++) {
01253                 double chanSum=0.0;
01254                 for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
01255                     if(this->GetJGroup(j)->GetChannel(ch)->GetRadType()=='P')
01256                         chanSum+=boundaryDiff[ch-1]*tempGamma[mu][ch-1]*
01257                             tempGamma[mup][ch-1];
01258                 }
01259                 if(mu==mup) cMatrix[mu].push_back(tempE[mu]-chanSum);
01260                 else cMatrix[mu].push_back(-chanSum);
01261             }
01262         }
01263         EigenFunc eigenFunc(cMatrix);
01264         if(fabs(eigenFunc.eigenvalues()[thisLevel]-tempE[thisLevel])<=energyTolerance)
01265             done=true;
01266         matrix_r newGamma;
01267         for(int mu=0;mu<tempE.size();mu++) {
01268             vector_r tempChanVector;
01269             newGamma.push_back(tempChanVector);
01270             for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
01271                 double gammaSum=0.0;
01272                 for(int mup=0;mup<tempE.size();mup++) {
01273                     gammaSum+=eigenFunc.eigenvectors()[mup][mu]*tempGamma[mup][ch-1];
01274                 }
01275                 newGamma[mu].push_back(gammaSum);
01276             }
01277         }
01278         for(int mu=0;mu<tempE.size();mu++) {
01279             tempE[mu]=eigenFunc.eigenvalues()[mu];
01280             for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
01281                 tempGamma[mu][ch-1]=newGamma[mu][ch-1];
01282             }
01283         }
01284         if(!done) {
01285             if(iteration==maxIterations) {
01286                 configure.outStream << "***WARNING: Could Not Transform J = "
01287                     << this->GetJGroup(j)->GetJ();
01288                 if(this->GetJGroup(j)->GetPi()==-1) configure.outStream << "-'";
01289                 else configure.outStream << '+';
01290                 configure.outStream << " E = " << theLevel->GetFitE() << " MeV*" << std::endl;
01291                 tempE[thisLevel]=theLevel->GetFitE();
01292                 for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++)
01293                     tempGamma[thisLevel][ch-1]=theLevel->GetFitGamma(ch);
01294             }
01295             iteration++;
01296         }
01297     }
01298
01299     theLevel->SetTransformE(tempE[thisLevel]);
01300     theLevel->SetTransformIterations(iteration);
01301     double nFSum=1.0;
01302     for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
01303         AChannel *theChannel=this->GetJGroup(j)->GetChannel(ch);
01304         theLevel->SetTransformGamma(ch,tempGamma[thisLevel][ch-1]);
01305         if(ch<=theLevel->NumNFIntegrals()) nFSum+=2.0*
01306             this->GetPair(theChannel->GetPairNum())->GetChRad()*
01307             this->GetPair(theChannel->GetPairNum())->GetRedMass()*
01308             uconv/pow(hbarc,2.0)*pow(tempGamma[thisLevel][ch-1],2.0)*
01309             theLevel->GetNFIntegral(ch);
01310     }
01311     theLevel->SetSqrtNFFactor(1.0/sqrt(nFSum));
01312 } else {
01313     theLevel->SetTransformE(theLevel->GetFitE());
01314     theLevel->SetTransformIterations(0);
01315     for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++)
01316         theLevel->SetTransformGamma(ch,theLevel->GetFitGamma(ch));

```



```

01317     }
01318     }
01319     }
01320 } else {
01321     for(int j=1;j<=this->NumJGroups();j++)
01322         for(int la=1;la<=this->GetJGroup(j)->NumLevels();la++) {
01323             this->GetJGroup(j)->GetLevel(la)->SetTransformIterations(0);
01324             this->GetJGroup(j)->GetLevel(la)->SetTransformE(this->GetJGroup(j)->GetLevel(la)->GetFitE());
01325             for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++)
01326                 this->GetJGroup(j)->GetLevel(la)->
01327                     SetTransformGamma(ch,this->GetJGroup(j)->GetLevel(la)->GetFitGamma(ch));
01328         }
01329     }
01330 }
01331 for(int j=1;j<=this->NumJGroups();j++) {
01332     JGroup *theJGroup=this->GetJGroup(j);
01333     for(int la=1;la<=this->GetJGroup(j)->NumLevels();la++) {
01334         ALevel *theLevel=this->GetJGroup(j)->GetLevel(la);
01335         double normSum=0.0;
01336         vector_r tempPene;
01337         for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
01338             AChannel *theChannel=this->GetJGroup(j)->GetChannel(ch);
01339             PPair *exitPair=this->GetPair(theChannel->GetPairNum());
01340             double localEnergy=theLevel->GetTransformE()-exitPair->GetSepE()-exitPair->GetExE();
01341             if(theChannel->GetRadType()=='P') {
01342                 if(localEnergy<0.0) {
01343                     ShiftFunc theShiftFunction(exitPair);
01344                     normSum+=theShiftFunction.EnergyDerivative(theChannel->GetL(),theLevel->GetTransformE())*
01345                         pow(theLevel->GetTransformGamma(ch),2.0);
01346                     WhitFunc newWhitFunc(exitPair);
01347                     double whitConv=newWhitFunc(theChannel->GetL(),
01348                         exitPair->GetChRad(),
01349                         fabs(localEnergy));
01350                     double pene=exitPair->GetRedMass()*exitPair->GetChRad()*uconv/
01351                         pow(hbarc,2.0)/pow(whitConv,2.0);
01352                     tempPene.push_back(pene);
01353                 }
01354             } else {
01355                 CoulFunc theCoulombFunction(exitPair,
01356                     !(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
01357                 double radius=exitPair->GetChRad();
01358                 normSum+=theCoulombFunction.PEShift_dE(theChannel->GetL(),radius,localEnergy)*
01359                     pow(theLevel->GetTransformGamma(ch),2.0);
01360                 double pene=theCoulombFunction.Penetrability(theChannel->GetL(),radius,localEnergy);
01361                 tempPene.push_back(pene);
01362             }
01363         } else if (theChannel->GetRadType()=='M' || theChannel->GetRadType()=='E') {
01364             if(fabs(theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE())<1.e-3&&
01365                 theJGroup->GetJ()==this->GetPair(theChannel->GetPairNum())->GetJ(2)&&
01366                 theJGroup->GetPi()==this->GetPair(theChannel->GetPairNum())->GetPi(2)) {
01367                 double jValue=theJGroup->GetJ();
01368                 double pene=1e-10;
01369                 if(theChannel->GetRadType()=='M'&&theChannel->GetL()==1)
01370                     pene=3.0*jValue/4.0/(jValue+1.)/nuclearMagneton/nuclearMagneton;
01371                 else if(theChannel->GetRadType()=='E'&&theChannel->GetL()==2)
01372                     pene=60.0*jValue*(2.*jValue-1.)/(jValue+1.)/(2.*jValue+3.);
01373                 if((int)(2*jValue)%2!=0) pene*=-1.;
01374                 tempPene.push_back(pene);
01375             } else {
01376                 double pene = (configure.paramMask & Config::USE_RMC_FORMALISM) ? 1.0 :
01377                     pow(fabs(localEnergy)/hbarc,2.0*theChannel->GetL()+1);
01378                 tempPene.push_back(pene);
01379             }
01380         } else tempPene.push_back(1.0);
01381     }
01382     for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
01383         AChannel* theChannel = this->GetJGroup(j)->GetChannel(ch);
01384         complex externalWidth(0.0,0.0);
01385         if((theChannel->GetRadType()=='M' || theChannel->GetRadType()=='E') &&
01386             theLevel->IsInRMatrix() && (configure.paramMask & Config::USE_EXTERNAL_CAPTURE) &&
01387             !(fabs(theLevel->GetTransformGamma(ch))<1.0e-8 && (configure.paramMask &
01388                 Config::IGNORE_ZERO_WIDTHS)))
01389             externalWidth=CalcExternalWidth(this->GetJGroup(j),theLevel,
01390                 this->GetJGroup(j)->GetChannel(ch),false,configure);
01391             theLevel->SetExternalGamma(ch,externalWidth);
01392             complex totalWidth=theLevel->GetTransformGamma(ch)+externalWidth;
01393             int tempSign = (real(totalWidth)<0.) ? (-1) : (1);
01394             double bigGamma;
01395             if(theChannel->GetRadType()!='F'&&theChannel->GetRadType()!='G')
01396                 bigGamma=tempSign*2.0*real(totalWidth*conj(totalWidth))*tempPene[ch-1]/
01397                     (1.0+normSum);
01398             else bigGamma=real(totalWidth);
01399             theLevel->SetBigGamma(ch,bigGamma);
01400         }
01401     }

```



```

01402
01407 void CNuc::PrintTransformParams(const Config& configure) {
01408     char filename[256];
01409     sprintf(filename,"%sparameters.out",configure.outputdir.c_str());
01410     std::ofstream out;
01411     out.open(filename);
01412     if(out) {
01413         out << "PHYSICAL LEVEL PARAMETERS (BOUNDARY CONDITION SET TO SHIFT AT E_LEVEL)" << std::endl;
01414         out << std::endl;
01415         for(int j=1;j<=this->NumJGroups();j++) {
01416             JGroup *theJGroup=this->GetJGroup(j);
01417             for(int la=1;la<=this->GetJGroup(j)->NumLevels();la++) {
01418                 out.precision(1);
01419                 out << std::fixed;
01420                 ALevel *theLevel=this->GetJGroup(j)->GetLevel(la);
01421                 out << "J = " << std::setw(3) << this->GetJGroup(j)->GetJ();
01422                 if(this->GetJGroup(j)->GetPi()==-1) out << "'-";
01423                 else out << "'+'";
01424                 out.precision(4);
01425                 out << " E_level = " << std::setw(8) << theLevel->GetTransformE() << " MeV"
01426                 << " ITERATIONS = " << std::setw(5) << theLevel->GetTransformIterations() << std::endl;
01427                 for(int ch=1;ch<=this->GetJGroup(j)->NumChannels();ch++) {
01428                     AChannel *theChannel=this->GetJGroup(j)->GetChannel(ch);
01429                     PPair *exitPair=this->GetPair(theChannel->GetPairNum());
01430                     double localEnergy=theLevel->GetTransformE()-exitPair->GetSepE()-exitPair->GetExE();
01431                     out << " R = " << std::setw(2) << exitPair->GetPairKey();
01432                     if(theChannel->GetRadType()=='P') out << " l = " << std::setw(3) << theChannel->GetL();
01433                     else if(theChannel->GetRadType()=='F') out << " Fermi Beta Decay ";
01434                     else if(theChannel->GetRadType()=='G') out << " G-T Beta Decay ";
01435                     else out << " L = " << std::setw(2) << theChannel->GetRadType() << theChannel->GetL();
01436                     out.precision(1);
01437                     if(theChannel->GetRadType()!='G' && theChannel->GetRadType()!='F')
01438                         out << " s = " << std::setw(4) << theChannel->GetS();
01439                     out.precision(6);
01440                     if(localEnergy<0.0 && theChannel->GetRadType()=='P') {
01441                         out << " C = " << std::setw(12) << sqrt(fabs(theLevel->GetBigGamma(ch)))
01442                         << " fm^(-1/2)";
01443                     } else if(fabs(theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE())<1.e-3 &&
01444                         theJGroup->GetJ()==this->GetPair(theChannel->GetPairNum())->GetJ(2) &&
01445                         theJGroup->GetPi()==this->GetPair(theChannel->GetPairNum())->GetPi(2) &&
01446                         theChannel->GetRadType()=='M' && theChannel->GetL()==1) {
01447                         int tempSign = (theLevel->GetBigGamma(ch)<0) ? (-1) : (1);
01448                         out << " mu = " << std::setw(12) << tempSign*sqrt(fabs(theLevel->GetBigGamma(ch)))
01449                         << " nm ";
01450                     } else if(fabs(theLevel->GetE()-this->GetPair(theChannel->GetPairNum())->GetExE())<1.e-3 &&
01451                         theJGroup->GetJ()==this->GetPair(theChannel->GetPairNum())->GetJ(2) &&
01452                         theJGroup->GetPi()==this->GetPair(theChannel->GetPairNum())->GetPi(2) &&
01453                         theChannel->GetRadType()=='E' && theChannel->GetL()==2) {
01454                         int tempSign = (theLevel->GetBigGamma(ch)<0) ? (-1) : (1);
01455                         out << " Q = " << std::setw(12) <<
tempSign*sqrt(fabs(theLevel->GetBigGamma(ch)))/100.0/sqrt(fstrec*hbarc)
01456                         << " b ";
01457                     } else if(theChannel->GetRadType()=='F' || theChannel->GetRadType()=='G') {
01458                         out << " B = " << std::setw(12) << theLevel->GetBigGamma(ch)
01459                         << " ";
01460                     } else {
01461                         if(fabs(theLevel->GetBigGamma(ch))>=1e-3)
01462                             out << " G = " << std::setw(12) << fabs(theLevel->GetBigGamma(ch))*1e3
01463                             << " keV ";
01464                         else if(fabs(theLevel->GetBigGamma(ch))>=1e-6)
01465                             out << " G = " << std::setw(12) << fabs(theLevel->GetBigGamma(ch))*1e6
01466                             << " eV ";
01467                         else
01468                             out << " G = " << std::setw(12) << fabs(theLevel->GetBigGamma(ch))*1e9
01469                             << " meV ";
01470                     }
01471                     out << " g_int = " << std::setw(12) << theLevel->GetTransformGamma(ch);
01472                     if(theChannel->GetRadType()!='G' && theChannel->GetRadType()!='F') out << " MeV^(1/2) ";
01473                     else out << " ";
01474                     out << " g_ext = " << std::setw(20) << theLevel->GetExternalGamma(ch);
01475                     if(theChannel->GetRadType()!='G' && theChannel->GetRadType()!='F') out << " MeV^(1/2) ";
01476                     out << std::endl;
01477                 }
01478                 out << std::endl;
01479             }
01480         }
01481     } else configure.outStream << "Could not save parameters.out file." << std::endl;
01482 }
01483
01488 void CNuc::SetMaxLValue(int maxL) {
01489     maxLValue_=maxL;
01490 }
01491
01497 void CNuc::CalcShiftFunctions(const Config& configure) {
01498     for(int j=1;j<=this->NumJGroups();j++) {
01499         if(this->GetJGroup(j)->IsInRMatrix()) {
01500             JGroup *theJGroup=this->GetJGroup(j);

```

```

01501     for(int la=1;la<=theJGroup->NumLevels();la++) {
01502     ALevel *theLevel=theJGroup->GetLevel(la);
01503     if(theLevel->IsInRMatrix()) {
01504     for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
01505     AChannel *theChannel=theJGroup->GetChannel(ch);
01506     PPair *thePair=this->GetPair(theChannel->GetPairNum());
01507     if(thePair->GetPType()==0) {
01508     int lValue=theChannel->GetL();
01509     double levelEnergy=theLevel->GetFitE();
01510     double resonanceEnergy=levelEnergy-(thePair->GetSepE()+thePair->GetExE());
01511     if(resonanceEnergy<0.0) {
01512     ShiftFunc theShiftFunction(thePair);
01513     theLevel->SetShiftFunction(ch,theShiftFunction(lValue,levelEnergy));
01514     }
01515     else {
01516     CoulFunc theCoulombFunction(thePair,
01517     !! (configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
01518     double radius=thePair->GetChRad();
01519     theLevel->SetShiftFunction(ch,theCoulombFunction.PEShift(lValue,radius,resonanceEnergy));
01520     }
01521     }
01522     else {
01523     theLevel->SetShiftFunction(ch,theJGroup->GetLevel(1)->GetShiftFunction(1));
01524     }
01525     }
01526     }
01527     }
01528     }
01529     }
01530 }
01531
01536 complex CNuc::CalcExternalWidth(JGroup* theJGroup, ALevel* theLevel,
01537     AChannel *theChannel,bool isInitial,const Config& configure) {
01538     complex externalWidth(0.0,0.0);
01539     if(theChannel->GetRadType()=='E' || (theChannel->GetRadType()=='M' && theChannel->GetL()==1)) {
01540     bool isExternal=false;
01541     int j=0;
01542     int la=0;
01543     while(!isExternal&&j<this->NumJGroups()) {
01544     j++;
01545     la=0;
01546     while(!isExternal&&la<this->GetJGroup(j)->NumLevels()) {
01547     la++;
01548     if(this->GetJGroup(j)->GetLevel(la)->IsECLLevel() &&
01549     theChannel->GetPairNum()==this->GetJGroup(j)->GetLevel(la)->GetECPairNum()) {
01550     isExternal=true;
01551     }
01552     }
01553     }
01554     if(isExternal) {
01555     JGroup *theFinalJGroup=this->GetJGroup(j);
01556     ALevel *theFinalLevel=theFinalJGroup->GetLevel(la);
01557     double theLevelEnergy;
01558     if(!isInitial) theLevelEnergy=theLevel->GetTransformE();
01559     else theLevelEnergy=theLevel->GetE();
01560     int multL=theChannel->GetL();
01561     if(((theChannel->GetRadType()=='E' && multL==1) && (theFinalLevel->GetECMultMask()&isE1)) ||
01562     ((theChannel->GetRadType()=='M' && multL==1) && (theFinalLevel->GetECMultMask()&isM1)) ||
01563     ((theChannel->GetRadType()=='E' && multL==2) && (theFinalLevel->GetECMultMask()&isE2))) {
01564     //allow only m1,e1,e2
01565     double theFinalLevelEnergy;
01566     if(!isInitial) theFinalLevelEnergy=theFinalLevel->GetTransformE();
01567     else theFinalLevelEnergy=theFinalLevel->GetE();
01568     for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
01569     double theInitialChannelGamma;
01570     if(!isInitial) theInitialChannelGamma=theLevel->GetTransformGamma(ch);
01571     else theInitialChannelGamma=theLevel->GetGamma(ch);
01572     AChannel *initialChannel=theJGroup->GetChannel(ch);
01573     if(initialChannel->GetRadType()=='P') {
01574     for(int chp=1;chp<=theFinalJGroup->NumChannels();chp++) {
01575     double theFinalChannelGamma;
01576     if(!isInitial) theFinalChannelGamma=theFinalLevel->GetTransformGamma(chp);
01577     else theFinalChannelGamma=theFinalLevel->GetGamma(chp);
01578     AChannel *finalChannel=theFinalJGroup->GetChannel(chp);
01579     if(finalChannel->GetRadType()=='P') {
01580     if(finalChannel->GetPairNum()==initialChannel->GetPairNum()) {
01581     if((abs(initialChannel->GetL()-multL)<=finalChannel->GetL() && finalChannel->GetL()<=initialChannel->GetL()+multL &&
01582     fabs(initialChannel->GetS()-finalChannel->GetL())<=theFinalJGroup->GetJ() &&
01583     theFinalJGroup->GetJ()<=initialChannel->GetS()+finalChannel->GetL() && initialChannel->GetS()==finalChannel->GetS())) ||
01584     (fabs(initialChannel->GetS()-multL)<=finalChannel->GetS() && finalChannel->GetS()<=initialChannel->GetS()+multL &&
01585     fabs(initialChannel->GetL()-finalChannel->GetS())<=theFinalJGroup->GetJ() &&
01586     theFinalJGroup->GetJ()<=initialChannel->GetL()+finalChannel->GetS() && initialChannel->GetL()==finalChannel->GetL() &&
01587     theChannel->GetRadType()=='M')) {

```

```

01587         PPair *theFinalPair=this->GetPair(finalChannel->GetPairNum());
01588
01589         ECIntegral theECIntegral(theFinalPair,configure);
01590         complex integrals = theECIntegral(initialChannel->GetL(),finalChannel->GetL(),
01591             initialChannel->GetS(),finalChannel->GetS(),
01592             theJGroup->GetJ(),theFinalJGroup->GetJ(),
01593             multL,theChannel->GetRadType(),
01594             theLevelEnergy,theFinalLevelEnergy,
01595             true);
01596
01597         double ecNormParam=theFinalChannelGamma*
01598             theFinalLevel->GetSqrtNFFactor()*theFinalLevel->GetECConversionFactor(chp);
01599         externalWidth=ecNormParam*theInitialChannelGamma*integrals;
01600     }
01601 }
01602 }
01603 }
01604 }
01605 }
01606 }
01607 }
01608 }
01609 return externalWidth;
01610 }
01611
01616 PPair *CNuc::GetPair(int pairNum) {
01617     PPair *b = &pairs_[pairNum-1];
01618     return b;
01619 }
01620
01625 JGroup *CNuc::GetJGroup(int jGroupNum) {
01626     JGroup *b = &jgroups_[jGroupNum-1];
01627     return b;
01628 }
01629
01635 CNuc *CNuc::Clone() const {
01636     CNuc *localCompound = new CNuc(*this);
01637     return localCompound;
01638 }

```

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.](#)

```

00001 #include "Config.h"
00002 #ifndef NO_STAT
00003 #include <sys/stat.h>
00004 #endif
00005 #include <iostream>
00006
00012 Config::Config(std::ostream& stream) : outStream(stream) {
00013     Reset();
00014 }
00015
00020 void Config::Reset() {
00021     chiVariance=1.0;
00022     screenCheckMask=0;
00023     fileCheckMask=0;
00024     paramMask=0;
00025     paramMask |=
00026         (USE_AMATRIX|USE_BRUNE_FORMALISM|IGNORE_ZERO_WIDTHS|TRANSFORM_PARAMETERS|CALCULATE_WITH_DATA|USE_LONGWAVELENGTH_APPROX);
00026     stopFlag=false;
00027     outputdir="";
00028     checkdir="";
00029 }

```

```

00030
00035 int Config::ReadConfigFile() {
00036     std::string dummy; std::string temp;
00037     std::ifstream in(configfile.c_str());
00038     if(!in) return -1;
00039     std::string line="";
00040     while(line!="<config>"&&!in.eof()) getline(in,line);
00041     if(line!="<config>") return -1;
00042     in > temp;getline(in,dummy);
00043     if(temp=="true") paramMask |= USE_AMATRIX;
00044     else paramMask &= ~USE_AMATRIX;
00045     getline(in,dummy);
00046     int poundSignPos=dummy.find_last_of('#');
00047     if(poundSignPos==std::string::npos) temp=dummy;
00048     else temp=dummy.substr(0,poundSignPos);
00049     int p2 = temp.find_last_not_of(" \n\t\r");
00050     if (p2 != std::string::npos) {
00051         int p1 = temp.find_first_not_of(" \n\t\r");
00052         if (p1 == std::string::npos) p1 = 0;
00053         outputdir=temp.substr(p1,(p2-p1)+1);
00054     } else outputdir=std::string();
00055     getline(in,dummy);
00056     poundSignPos=dummy.find_last_of('#');
00057     if(poundSignPos==std::string::npos) temp=dummy;
00058     else temp=dummy.substr(0,poundSignPos);
00059     p2 = temp.find_last_not_of(" \n\t\r");
00060     if (p2 != std::string::npos) {
00061         int p1 = temp.find_first_not_of(" \n\t\r");
00062         if (p1 == std::string::npos) p1 = 0;
00063         checkdir=temp.substr(p1,(p2-p1)+1);
00064     } else checkdir=std::string();
00065     in > temp;getline(in,dummy);
00066     if(temp=="screen") screenCheckMask |= CHECK_COMPOUND_NUCLEUS;
00067     else if(temp=="file") fileCheckMask |= CHECK_COMPOUND_NUCLEUS;
00068     in > temp;getline(in,dummy);
00069     if(temp=="screen") screenCheckMask |= CHECK_BOUNDARY_CONDITIONS;
00070     else if(temp=="file") fileCheckMask |= CHECK_BOUNDARY_CONDITIONS;
00071     in > temp;getline(in,dummy);
00072     if(temp=="screen") screenCheckMask |= CHECK_DATA;
00073     else if(temp=="file") fileCheckMask |= CHECK_DATA;
00074     in > temp;getline(in,dummy);
00075     if(temp=="screen") screenCheckMask |= CHECK_ENERGY_DEP;
00076     else if(temp=="file") fileCheckMask |= CHECK_ENERGY_DEP;
00077     in > temp;getline(in,dummy);
00078     if(temp=="screen") screenCheckMask |= CHECK_LEGENDRE;
00079     else if(temp=="file") fileCheckMask |= CHECK_LEGENDRE;
00080     in > temp;getline(in,dummy);
00081     if(temp=="screen") screenCheckMask |= CHECK_COUL_AMPLITUDES;
00082     else if(temp=="file") fileCheckMask |= CHECK_COUL_AMPLITUDES;
00083     in > temp;getline(in,dummy);
00084     if(temp=="screen") screenCheckMask |= CHECK_PATHWAYS;
00085     else if(temp=="file") fileCheckMask |= CHECK_PATHWAYS;
00086     in > temp;getline(in,dummy);
00087     if(temp=="screen") screenCheckMask |= CHECK_ANGULAR_DISTs;
00088     else if(temp=="file") fileCheckMask |= CHECK_ANGULAR_DISTs;
00089     line="";
00090     while(line!="</config>"&&!in.eof()) getline(in,line);
00091     if(line!="</config>") return -1;
00092     in.close();
00093     return 0;
00094 }
00095
00101 #ifndef NO_STAT
00102 int Config::CheckForInputFiles() {
00103     struct stat buffer;
00104     if(stat(outputdir.c_str(),&buffer) != 0) {
00105         ostream << "Could not find output directory: " << outputdir << ". Check that it exists." <<
std::endl;
00106         return -1;
00107     }
00108     if(stat(checkdir.c_str(),&buffer) != 0) {
00109         ostream << "Could not find checks directory: " << checkdir << ". Check that it exists." <<
std::endl;
00110         return -1;
00111     }
00112     return 0;
00113 }
00114 #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.](#)

```
00001 #include "CoulFunc.h"
00002 #include "PPair.h"
00003 #include <iostream>
00004 #include "cwfcomp.H"
00005 #include <gsl/gsl_sf_coulomb.h>
00006 #include <gsl/gsl_deriv.h>
00007
00012 CoulFunc::CoulFunc(PPair *pPair, bool useGSLFunctions) :
00013     useGSLFunctions_(useGSLFunctions) {
00014     z1_=pPair->GetZ(1);
00015     z2_=pPair->GetZ(2);
00016     redmass_=(double)pPair->GetRedMass();
00017     lLast_=0;
00018     radiusLast_=0.0;
00019     energyLast_=0.0;
00020     coulLast_.F=0.0;
00021     coulLast_.dF=0.0;
00022     coulLast_.G=0.0;
00023     coulLast_.dG=0.0;
00024     dEShiftParams_.coulFunc=this;
00025 }
00026
00031 int CoulFunc::z1() const {
00032     return z1_;
00033 }
00034
00039 int CoulFunc::z2() const {
00040     return z2_;
00041 }
00042
00047 double CoulFunc::redmass() const {
00048     return redmass_;
00049 }
00050
00056 int CoulFunc::lLast() const {
00057     return lLast_;
00058 }
00059
00065 double CoulFunc::radiusLast() const {
00066     return radiusLast_;
00067 }
00068
00074 double CoulFunc::energyLast() const {
00075     return energyLast_;
00076 }
00077
00082 struct CoulWaves CoulFunc::coulLast() const {
00083     return coulLast_;
00084 }
00085
00090 void CoulFunc::setLast(int lLast, double rLast, double eLast, CoulWaves coulLast) {
00091     lLast_=lLast;
00092     radiusLast_=rLast;
00093     energyLast_=eLast;
00094     coulLast_.F=coulLast.F;
00095     coulLast_.dF=coulLast.dF;
00096     coulLast_.G=coulLast.G;
00097     coulLast_.dG=coulLast.dG;
00098 }
00099
00106 CoulWaves CoulFunc::operator()(int l,double radius,double energy) {
```

```

00107 struct CoulWaves result={0.0,0.0,0.0,0.0};
00108 if(l==lLast() && radius==radiusLast() && energy==energyLast()) {
00109     result=coulLast();
00110 } else {
00111     struct CoulWaves newResult;
00112     if(!useGSLFunctions_) {
00113         std::complex<double> eta(sqrt(uconv/2.)*fstruc*z1()*z2()*
00114             sqrt(redmass()/energy),0.);
00115         std::complex<double> rho(sqrt(2.*uconv)/hbarc*radius*
00116             sqrt(redmass()*energy),0.);
00117         std::complex<double> lValue( (double) 1, 0.);
00118         Coulomb_wave_functions coul(true,lValue,eta);
00119         std::complex<double> c_F, c_dF, c_G, c_dG;
00120         coul.F_dF(rho,c_F,c_dF);
00121         coul.G_dG(rho,c_G,c_dG);
00122         newResult.F=real(c_F);
00123         newResult.dF=real(c_dF);
00124         newResult.G=real(c_G);
00125         newResult.dG=real(c_dG);
00126     } else {
00127         double eta=sqrt(uconv/2.)*fstruc*z1()*z2()*
00128             sqrt(redmass()/energy);
00129         double rho=sqrt(2.*uconv)/hbarc*radius*
00130             sqrt(redmass()*energy);
00131         double lValue=double(1);
00132         double eF,eG;
00133         gsl_sf_result F,Fp,G,Gp;
00134         gsl_sf_coulomb_wave_FG_e(eta,rho,lValue,0,&F,&Fp,&G,&Gp,&eF,&eG);
00135         newResult.F=F.val*exp(eF);
00136         newResult.dF=Fp.val*exp(eF);
00137         newResult.G=G.val*exp(eG);
00138         newResult.dG=Gp.val*exp(eG);
00139     }
00140     setLast(l,radius,energy,newResult);
00141     result=newResult;
00142 }
00143 return result;
00144 }
00145
00151 double CoulFunc::Penetrability(int l,double radius,double energy) {
00152     struct CoulWaves coul=this->operator()(l,radius,energy);
00153     double rho=sqrt(2.*uconv)/hbarc*radius*sqrt(redmass()*energy);
00154     return rho/(pow(coul.F,2.0)+pow(coul.G,2.0));
00155 }
00156
00162 double CoulFunc::PEShift(int l,double radius,double energy) {
00163     struct CoulWaves coul=this->operator()(l,radius,energy);
00164     double rho=sqrt(2.*uconv)/hbarc*radius*sqrt(redmass()*energy);
00165     if(pow(coul.F,2.0)==0.&&coul.F*coul.dF==0.) return rho*(coul.dG/coul.G);
00166     else return rho/(pow(coul.F,2.0)+pow(coul.G,2.0))*
00167         (coul.F*coul.dF+coul.G*coul.dG);
00168 }
00169
00170 double CoulFunc::thisPEShift(double x, void *p) {
00171     DEShiftParams *params = (DEShiftParams*)p;
00172     CoulFunc *coulFunc=(params->coulFunc);
00173     int lValue=(params->lValue);
00174     double radius=(params->radius);
00175
00176     return coulFunc->PEShift(lValue,radius,x);
00177 }
00178
00184 double CoulFunc::PEShift_dE(int l,double radius,double energy) {
00185     double result;
00186     double error;
00187
00188     dEShiftParams_.radius=radius;
00189     dEShiftParams_.lValue=l;
00190
00191     gsl_function F;
00192     F.function=&thisPEShift;
00193     F.params=&dEShiftParams_;
00194
00195     gsl_deriv_central (&F, energy, 1e-6, &result, &error);
00196
00197     return result;
00198 }
00199

```

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.](#)

```
00001 #include "Decay.h"
00002
00008 Decay::Decay(int pairNum) :
00009     pair_(pairNum) {};
00010
00015 int Decay::GetPairNum() const {
00016     return pair_;
00017 }
00018
00023 int Decay::NumKGroups() const {
00024     return kgroups_.size();
00025 }
00026
00031 int Decay::NumKLGroups() const {
00032     return klgroups_.size();
00033 }
00034
00040 int Decay::IsKGroup(KGroup a) {
00041     bool b=false;
00042     int c=0;
00043     while(!b&& c<this->NumKGroups())
00044     {
00045         if(a.GetS()==this->GetKGroup(c+1)->GetS())&&
00046             a.GetSp()==this->GetKGroup(c+1)->GetSp()) b=true;
00047         c++;
00048     }
00049     if(b) return c;
00050     else return 0;
00051 }
00052
00058 int Decay::IsKLGroup(KLGroup a) {
00059     bool b=false;
00060     int c=0;
00061     while(!b&& c<this->NumKLGroups())
00062     {
00063         if(a.GetK()==this->GetKLGroup(c+1)->GetK())&&
00064             a.GetLOrder()==this->GetKLGroup(c+1)->GetLOrder()) b=true;
00065         c++;
00066     }
00067     if(b) return c;
00068     else return 0;
00069 }
00070
00075 void Decay::AddKGroup(KGroup kGroup) {
00076     kgroups_.push_back(kGroup);
00077 }
00078
00083 void Decay::AddKLGroup(KLGroup klGroup) {
00084     klgroups_.push_back(klGroup);
00085 }
00086
00091 KGroup *Decay::GetKGroup(int kGroupNum) {
00092     KGroup *b=&kgroups_[kGroupNum-1];
00093     return b;
00094 }
00095
00100 KLGroup *Decay::GetKLGroup(int klGroupNum) {
00101     KLGroup *b=&klgroups_[klGroupNum-1];
00102     return b;
00103 }
```

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.](#)

```

00001 #include "ECIntegral.h"
00002 #include "AngCoeff.h"
00003 #include "EffectiveCharge.h"
00004 #include <math.h>
00005 #include <gsl/gsl_integration.h>
00006 #include <assert.h>
00007
00008 extern double DoubleFactorial(int);
00009
00010 double ECIntegral::FWIntegrand(double x, void * p) {
00011     Params *params = (Params*)p;
00012     CoulFunc *theCoulFunc=(params->coulFunc);
00013     WhitFunc *theWhitFunc=(params->whitFunc);
00014     int liValue = (params->liValue);
00015     int lfValue = (params->lfValue);
00016     int multLValue = (params->multLValue);
00017     double pairEnergy = (params->pairEnergy);
00018     double bindingEnergy = (params->bindingEnergy);
00019
00020     struct CoulWaves coul = theCoulFunc->operator()(liValue,x,pairEnergy);
00021     double whit = theWhitFunc->operator()(lfValue,x,bindingEnergy);
00022     double returnValue = coul.F*whit*pow(x,multLValue);
00023     return (!params->useLongWavelengthApprox) ? params->effectiveCharge->operator()(x)*returnValue :
00024         returnValue;
00025 }
00026
00027 double ECIntegral::GWIntegrand(double x, void * p) {
00028     Params *params = (Params*)p;
00029     CoulFunc *theCoulFunc=(params->coulFunc);
00030     WhitFunc *theWhitFunc=(params->whitFunc);
00031     int liValue = (params->liValue);
00032     int lfValue = (params->lfValue);
00033     int multLValue = (params->multLValue);
00034     double pairEnergy = (params->pairEnergy);
00035     double bindingEnergy = (params->bindingEnergy);
00036
00037     struct CoulWaves coul = theCoulFunc->operator()(liValue,x,pairEnergy);
00038     double whit = theWhitFunc->operator()(lfValue,x,bindingEnergy);
00039     double returnValue = coul.G*whit*pow(x,multLValue);
00040     return (!params->useLongWavelengthApprox) ? params->effectiveCharge->operator()(x)*returnValue :
00041         returnValue;
00042 }
00043
00044 double ECIntegral::WWIntegrand(double x, void * p) {
00045     Params *params = (Params*)p;
00046     WhitFunc *theWhitFunc=(params->whitFunc);
00047     int liValue = (params->liValue);
00048     int lfValue = (params->lfValue);
00049     int multLValue = (params->multLValue);
00050     double pairEnergy = (params->pairEnergy);
00051     double bindingEnergy = (params->bindingEnergy);
00052
00053     double whitIn = theWhitFunc->operator()(liValue,x,fabs(pairEnergy));
00054     double whitOut= theWhitFunc->operator()(lfValue,x,fabs(bindingEnergy));
00055     double returnValue = whitIn*whitOut*pow(x,multLValue);
00056     return (!params->useLongWavelengthApprox) ? params->effectiveCharge->operator()(x)*returnValue :
00057         returnValue;
00058 }
00059
00060 void ECIntegral::Integrate(double chanrad) {
00061     gsl_integration_workspace * w
00062         = gsl_integration_workspace_alloc (1000);
00063
00064     if(params_.pairEnergy<0.0) {
00065         gsl_function WW;
00066         WW.function = &WWIntegrand;
00067         WW.params= &params_;
00068
00069         double wwintresult,wwinterror;
00070
00071         gsl_integration_qagiu(&WW,chanrad,0.0,1e-4,1000,w,&wwintresult,&wwinterror);
00072         GW_=wwintresult;
00073         FW_=0.0;
00074     } else {
00075         gsl_function FW;
00076         FW.function = &FWIntegrand;
00077         FW.params= &params_;
00078
00079         gsl_function GW;
00080         GW.function = &GWIntegrand;
00081         GW.params= &params_;
00082     }

```

```

00083
00084     double fwintresult,fwinterror;
00085     double gwintresult,gwinterror;
00086
00087     gsl_integration_qagi(&FW,chanrad,0.0,1e-4,1000,w,&fwintresult,&fwinterror);
00088     gsl_integration_qagi(&GW,chanrad,0.0,1e-4,1000,w,&gwintresult,&gwinterror);
00089     FW_=fwintresult;
00090     GW_=gwintresult;
00091 }
00092
00093 gsl_integration_workspace_free (w);
00094 }
00095
00102 complex ECIntegral::operator()(int theInitialLValue, int theFinalLValue,
00103                                double theInitialSValue, double theFinalSValue,
00104                                double theInitialJValue, double theFinalJValue,
00105                                int theLMult, char radType,
00106                                double inEnergy, double levelEnergy,
00107                                bool isChannelCapture) {
00108     ResetIntegrals();
00109
00110     double sepEnergy = pair()->GetSepE()+pair()->GetExE();
00111     double outEnergy = inEnergy - sepEnergy;
00112     double chanRad = pair()->GetChRad();
00113     double redMass = pair()->GetRedMass();
00114
00115     EffectiveCharge effectiveChargeFunc(pair(),inEnergy-levelEnergy,theLMult);
00116
00117     params_.effectiveCharge=&effectiveChargeFunc;
00118     params_.liValue = theInitialLValue;
00119     params_.lfValue = theFinalLValue;
00120     params_.multLValue = theLMult;
00121     params_.pairEnergy = outEnergy;
00122     params_.bindingEnergy = fabs(levelEnergy-sepEnergy);
00123
00124     if(radType=='E')
00125         params_.multLValue = theLMult;
00126     else
00127         params_.multLValue = 0;
00128     Integrate(chanRad);
00129
00130     complex overlapIntegral(0.,0.);
00131     if(outEnergy>0.0) {
00132         struct CoulWaves
00133             coul=coulfunction()->operator()(theInitialLValue,chanRad,outEnergy);
00134         if(isChannelCapture) {
00135             complex chanExpHSP(coul.G/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)),
00136                                -coul.F/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)));
00137             overlapIntegral=complex(0.0,-0.5)*
00138             sqrt(coulfunction()->Penetrability(theInitialLValue,chanRad,outEnergy))*
00139             pow(redMass*uconv/2./fabs(outEnergy),0.25)/sqrt(hbarc)*
00140             chanExpHSP*(GW()+complex(0.0,1.0)*FW());
00141         } else overlapIntegral=(coul.G/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0))*FW()
00142                                -coul.F/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0))*GW())*
00143             pow(redMass*uconv/2./fabs(outEnergy),0.25)/sqrt(hbarc);
00144     } else {
00145         assert(isChannelCapture);
00146         double whit=whitfunction()->operator()(theInitialLValue,chanRad,fabs(outEnergy));
00147         overlapIntegral=complex(0.0,-0.5)*GW()/whit*
00148             sqrt(redMass*uconv*chanRad)/hbarc;
00149     }
00150
00151
00152     double effectiveCharge;
00153     if(radType=='E') {
00154         if(params_.useLongWavelengthApprox) {
00155             double totalM=pair()->GetM(1)+pair()->GetM(2);
00156             effectiveCharge=sqrt(fstruc*hbarc)*(pair()->GetZ(1)*pow(pair()->GetM(2)/totalM,theLMult)+
00157                                pair()->GetZ(2)*pow(-pair()->GetM(1)/totalM,theLMult));
00158         } else effectiveCharge = 1.;
00159     } else {
00160         effectiveCharge=redMass*1.00727638*
00161             (pair()->GetZ(1)/pow(pair()->GetM(1),2.)+
00162             pair()->GetZ(2)/pow(pair()->GetM(2),2.));
00163     }
00164
00165     complex ecAmplitude(0.0,0.0);
00166     if(radType=='E') {
00167         ecAmplitude=complex(0.0,-1.0)*
00168         effectiveCharge*sqrt((8.*(2.*theLMult+1.)*(theLMult+1.))/theLMult)/DoubleFactorial(2*theLMult+1)*
00169         pow(complex(0.,1.0),theInitialLValue+theLMult-theFinalLValue)*
00170
00171         AngCoeff::ClebGord(theInitialLValue,theLMult,theFinalLValue,0,0,0)*sqrt(2.*theInitialLValue+1.)*sqrt(2.*theFinalJValue+1.)
00172     } else {
00173         AngCoeff::Racah(theLMult,theFinalLValue,theInitialJValue,theInitialSValue,theInitialLValue,theFinalJValue);
00174     }

```

```

00173     complex orbitalTerm=effectiveCharge*
00174         sqrt((2.*theInitialLValue+1.)*(theInitialLValue+1.)*theInitialLValue)*
00175     AngCoeff::Racah(1.,theInitialLValue,theInitialJValue,theInitialSValue,theInitialLValue,theFinalJValue);
00176     complex tau=pow(std::complex<double>(-1.,0.),pair()->GetJ(1)+pair()->GetJ(2))*
00177         (pow(complex(-1.,0.),theFinalSValue)*
00178         sqrt(pair()->GetJ(1)*(pair()->GetJ(1)+1.)*(2.*pair()->GetJ(1)+1.))*
00179     AngCoeff::Racah(theFinalSValue,pair()->GetJ(1),theInitialSValue,pair()->GetJ(1),pair()->GetJ(2),1.)*
00180         pair()->GetG(1)+
00181         pow(complex(-1.,0.),theInitialSValue)*
00182         sqrt(pair()->GetJ(2)*(pair()->GetJ(2)+1.)*(2.*pair()->GetJ(2)+1.))*
00183     AngCoeff::Racah(theFinalSValue,pair()->GetJ(2),theInitialSValue,pair()->GetJ(2),pair()->GetJ(1),1.)*
00184         pair()->GetG(2));
00185     complex spinTerm=-sqrt((2.*theInitialSValue+1.)*(2.*theFinalSValue+1.))*
00186     AngCoeff::Racah(1,theInitialSValue,theFinalJValue,theInitialLValue,theFinalSValue,theInitialJValue)*tau;
00187     ecAmplitude=complex(0.0,1.0)*
00188         sqrt(fstruc)*pow(hbarc,1.5)/(2*1.00727638*ucnv)*sqrt(16/3)*sqrt(2*theFinalJValue+1.)*
00189         (orbitalTerm+spinTerm);
00190 }
00191
00192 return ecAmplitude*overlapIntegral;
00193 };

```

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.](#)

```

00001 #include "ECMGroup.h"
00002 #include <assert.h>
00003
00008 ECMGroup::ECMGroup(char radType, int multipolarity, int lInitial, double jInitial, int
    finalChannelNum, int ecJGroupNum, int ecLevelNum) :
00009     radtype_(radType), mult_(multipolarity), li_(lInitial), ji_(jInitial), chf_(finalChannelNum),
    jGroupNum_(ecJGroupNum), levelNum_(ecLevelNum),
00010     ischancap_(false), chdecay_(0), chkgroup_(0), chmgroup_(0), internalChannel_(0),
    statspinfactor_(0.0){
00011 }
00012
00019 ECMGroup::ECMGroup(char radType, int multipolarity, int lInitial, double jInitial, int
    finalChannelNum, int ecJGroupNum, int ecLevelNum, int decayNum, int kGroupNum, int mGroupNum, int
    internalChannel) :
00020     radtype_(radType), mult_(multipolarity), li_(lInitial), ji_(jInitial), chf_(finalChannelNum),
    jGroupNum_(ecJGroupNum), levelNum_(ecLevelNum),
00021     ischancap_(true), chdecay_(decayNum), chkgroup_(kGroupNum), chmgroup_(mGroupNum),
    internalChannel_(internalChannel), statspinfactor_(0.0){
00022 }
00023
00028 bool ECMGroup::IsChannelCapture() const {
00029     return ischancap_;
00030 }
00031
00036 char ECMGroup::GetRadType() const {
00037     return radtype_;
00038 }
00039
00044 int ECMGroup::GetMult() const {
00045     return mult_;
00046 }
00047
00052 int ECMGroup::GetL() const {
00053     return li_;
00054 }
00055
00060 int ECMGroup::GetFinalChannel() const {

```

```

00061     return chf_;
00062 }
00063
00068 int ECMGroup::GetJGroupNum() const {
00069     return jGroupNum_;
00070 }
00071
00076 int ECMGroup::GetLevelNum() const {
00077     return levelNum_;
00078 }
00079
00084 int ECMGroup::GetChanCapDecay() const {
00085     return chdecay_;
00086 }
00087
00092 int ECMGroup::GetChanCapKGroup() const {
00093     return chkgroup_;
00094 }
00095
00100 int ECMGroup::GetChanCapMGroup() const {
00101     return chmgroup_;
00102 }
00103
00108 int ECMGroup::GetIntChannelNum() const {
00109     return internalChannel_;
00110 }
00111
00116 double ECMGroup::GetJ() const {
00117     return ji_;
00118 }
00119
00124 double ECMGroup::GetStatSpinFactor() const {
00125     return statspinfactor_;
00126 }
00127
00132 void ECMGroup::SetStatSpinFactor(double a) {
00133     statspinfactor_=a;
00134 }

```

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 "GSLErrorException.h"
#include <iostream>
#include <iomanip>
#include <omp.h>
#include <time.h>

```

8.255 EData.cpp

[Go to the documentation of this file.](#)

```

00001 #include "AZUREOutput.h"
00002 #include "CNuc.h"
00003 #include "Config.h"
00004 #include "EData.h"
00005 #include "ExtrapLine.h"
00006 #include "SegLine.h"
00007 #include "Minuit2/MnUserParameters.h"
00008 #include "GSLErrorException.h"
00009 #include <iostream>

```

```

00010 #include <iomanip>
00011 #include <omp.h>
00012 #include <time.h>
00013
00019 EData::EData() {
00020     iterations_=0;
00021     isFit_=true;
00022     isErrorAnalysis_=false;
00023 }
00024
00029 int EData::NumSegments() const {
00030     return segments_.size();
00031 }
00032
00039 int EData::Fill(const Config& configure, CNuc *theCNuc) {
00040     std::ifstream in(configure.configfile.c_str());
00041     if(!in) return -1;
00042     std::string line="";
00043     while(line!="<segmentsData>"&&!in.eof()) getline(in,line);
00044     if(line!="<segmentsData>") return -1;
00045     line="";
00046     int numTotalSegments=0;
00047     while(!in.eof()&&line!="</segmentsData>") {
00048         getline(in,line);
00049         bool empty=true;
00050         for(unsigned int i=0;i<line.size();++i)
00051             if(line[i]!=' ' && line[i]!='\t') {
00052                 empty=false;
00053                 break;
00054             }
00055         if(empty==true) continue;
00056         if(!in.eof()&&line!="</segmentsData>") {
00057             std::stringstream stm;
00058             stm.str(line);
00059             SegLine segment(stm);
00060             if(stm.rdstate() & (std::stringstream::failbit | std::stringstream::badbit)) return -1;
00061             numTotalSegments++;
00062             if(segment.isActive()==1) {
00063                 ESegment NewSegment(segment);
00064                 if(theCNuc->IsPairKey(NewSegment.GetEntranceKey())) {
00065                     theCNuc->GetPair(theCNuc->GetPairNumFromKey(NewSegment.GetEntranceKey()))->SetEntrance();
00066                     bool isValidTotal=false;
00067                     if(NewSegment.GetExitKey()==-1) {
00068                         for(int i = 1;i<=theCNuc->NumPairs();i++) {
00069                             if(theCNuc->GetPair(i)->GetPType()==10) {
00070                                 isValidTotal = true;
00071                                 break;
00072                             }
00073                         }
00074                     }
00075                     if(isValidTotal||theCNuc->IsPairKey(NewSegment.GetExitKey())) {
00076                         NewSegment.SetSegmentKey(numTotalSegments);
00077                         this->AddSegment(NewSegment);
00078                         if(this->GetSegment(this->NumSegments())->Fill(theCNuc,this,configure)==-1) {
00079                             configure.outStream << "WARNING: Could Not Fill Segment #" << this->NumSegments()
00080                                 << " from file." << std::endl;
00081                             this->DeleteLastSegment();
00082                         } else if(this->GetSegment(this->NumSegments())->NumPoints()==0) {
00083                             configure.outStream << "WARNING: Segment #" << numTotalSegments
00084                                 << " is empty and will not be used." << std::endl;
00085                             this->DeleteLastSegment();
00086                         } else {
00087                             int thisSegmentNum = this->NumSegments();
00088                             if(this->GetSegment(thisSegmentNum)->IsTotalCapture()) {
00089                                 int numCapturePairs=0;
00090                                 for(int i = 1;i<=theCNuc->NumPairs();i++) {
00091                                     if(theCNuc->GetPair(i)->GetPType()==10) {
00092                                         numCapturePairs++;
00093                                         if(numCapturePairs==1) {
00094                                             this->GetSegment(thisSegmentNum)->SetExitKey(theCNuc->GetPair(i)->GetPairKey());
00095                                         } else {
00096                                             ESegment newSegment(*this->GetSegment(thisSegmentNum));
00097                                             newSegment.SetExitKey(theCNuc->GetPair(i)->GetPairKey());
00098                                             newSegment.SetIsTotalCapture(0);
00099                                             newSegment.SetVaryNorm(false);
00100                                             this->AddSegment(newSegment);
00101                                         }
00102                                     }
00103                                 }
00104                                 this->GetSegment(thisSegmentNum)->SetIsTotalCapture(numCapturePairs);
00105                             }
00106                         } else {
00107                             if(NewSegment.GetExitKey()==-1) {
00108                                 configure.outStream << "WARNING: Total capture specified but no capture pair exists."
00109                                     << std::endl;
00110                             } else {

```

```

00112         configure.outStream << "WARNING: Pair key " << NewSegment.GetExitKey()
00113         << " not in compound nucleus." << std::endl;
00114     }
00115 }
00116 } else configure.outStream << "WARNING: Pair key " << NewSegment.GetEntranceKey()
00117 << " not in compound nucleus." << std::endl;
00118 }
00119 }
00120 }
00121
00122 if(line!="</segmentsData>") return -1;
00123
00124 in.close();
00125
00126 if(this->NumSegments()>0) {
00127     if(this->ReadTargetEffectsFile(configure,theCNuc)==-1) return -1;
00128     this->MapData();
00129 }
00130
00131 return 0;
00132 }
00133
00140 int EData::MakePoints(const Config& configure, CNuc *theCNuc) {
00141     std::ifstream in(configure.configfile.c_str());
00142     if(!in) return -1;
00143     std::string line = "";
00144     while(line!="<segmentsTest>"&&!in.eof()) getline(in,line);
00145     if(line!="<segmentsTest>") return -1;
00146     line="";
00147     int numTotalSegments=0;
00148     while(!in.eof()&&line!="</segmentsTest>") {
00149         getline(in,line);
00150         bool empty=true;
00151         for(unsigned int i=0;i<line.size();++i)
00152             if(line[i]!=' ' && line[i]!='\t') {
00153                 empty=false;
00154                 break;
00155             }
00156         if(empty==true) continue;
00157         if(!in.eof()&&line!="</segmentsTest>") {
00158             std::stringstream stm;
00159             stm.str(line);
00160             ExtrapolLine segment(stm);
00161             if(stm.rdstate() & (std::stringstream::failbit | std::stringstream::badbit)) return -1;
00162             numTotalSegments++;
00163             if(segment.isActive()==1) {
00164                 ESegment NewSegment(segment);
00165                 if(theCNuc->IsPairKey(NewSegment.GetEntranceKey())) {
00166                     bool isValidTotal=false;
00167                     if(NewSegment.GetExitKey()==-1) {
00168                         for(int i = 1;i<=theCNuc->NumPairs();i++) {
00169                             if(theCNuc->GetPair(i)->GetPType()==10) {
00170                                 isValidTotal = true;
00171                                 break;
00172                             }
00173                         }
00174                     }
00175                     if(isValidTotal||theCNuc->IsPairKey(NewSegment.GetExitKey())) {
00176                         NewSegment.SetSegmentKey(numTotalSegments);
00177                         this->AddSegment(NewSegment);
00178                         ESegment *theSegment=this->GetSegment(this->NumSegments());
00179                         theCNuc->GetPair(theCNuc->GetPairNumFromKey(theSegment->GetEntranceKey()))->SetEntrance();
00180                         PPair
00181                         *entrancePair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(theSegment->GetEntranceKey()));
00182                         PPair *exitPair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(theSegment->GetExitKey()));
00183                         double aStep=theSegment->GetAStep();
00184                         double eStep=theSegment->GetEStep();
00185                         for(double angle=theSegment->GetMinAngle();
00186                             angle<=theSegment->GetMaxAngle();angle+=aStep) {
00187                             for(double energy=theSegment->GetMinEnergy();
00188                                 energy<=theSegment->GetMaxEnergy();energy+=eStep) {
00189                                 EPoint NewPoint(angle,energy,theSegment);
00190                                 theSegment->AddPoint(NewPoint);
00191                                 EPoint *thePoint=theSegment->GetPoint(theSegment->NumPoints());
00192                                 thePoint->SetParentData(this);
00193                                 if(entrancePair->GetPType()==20) thePoint->ConvertDecayEnergy(exitPair);
00194                                 else thePoint->ConvertLabEnergy(entrancePair);
00195                                 if(exitPair->GetPType()==0&&theSegment->IsDifferential()&&
00196                                     !theSegment->IsPhase()&&!theSegment->IsAngularDist()) {
00197                                     if(theSegment->GetEntranceKey()==theSegment->GetExitKey()) {
00198                                         thePoint->ConvertLabAngle(entrancePair);
00199                                     } else {
00200                                         thePoint->ConvertLabAngle(entrancePair,exitPair,configure);
00201                                     }
00202                                     thePoint->ConvertCrossSection(entrancePair,exitPair);
00203                                 }
00204                                 if(eStep==0.0) break;

```

```

00204         }
00205         if(aStep==0.0) break;
00206     }
00207     if(theSegment->NumPoints()==0) {
00208         configure.outStream << "WARNING: Extrapolation segment #" << numTotalSegments
00209         << " is empty and will not be used." << std::endl;
00210         this->DeleteLastSegment();
00211     } else {
00212         int thisSegmentNum = this->NumSegments();
00213         if(this->GetSegment(thisSegmentNum)->IsTotalCapture()) {
00214             int numCapturePairs=0;
00215             for(int i = 1;i<=theCNuc->NumPairs();i++) {
00216                 if(theCNuc->GetPair(i)->GetPType()==10) {
00217                     numCapturePairs++;
00218                     if(numCapturePairs==1) {
00219                         this->GetSegment(thisSegmentNum)->SetExitKey(theCNuc->GetPair(i)->GetPairKey());
00220                     } else {
00221                         ESegment newSegment(*this->GetSegment(thisSegmentNum));
00222                         newSegment.SetExitKey(theCNuc->GetPair(i)->GetPairKey());
00223                         newSegment.SetIsTotalCapture(0);
00224                         newSegment.SetVaryNorm(false);
00225                         this->AddSegment(newSegment);
00226                     }
00227                 }
00228             }
00229             this->GetSegment(thisSegmentNum)->SetIsTotalCapture(numCapturePairs);
00230         }
00231     }
00232 } else {
00233     if(NewSegment.GetExitKey()==-1) {
00234         configure.outStream << "WARNING: Total capture specified but no capture pair exists."
00235         << std::endl;
00236     } else {
00237         configure.outStream << "WARNING: Pair key " << NewSegment.GetExitKey()
00238         << " not in compound nucleus." << std::endl;
00239     }
00240 }
00241 } else configure.outStream << "WARNING: Pair key " << NewSegment.GetEntranceKey()
00242 << " not in compound nucleus." << std::endl;
00243 }
00244 }
00245 }
00246
00247 if(line!="</segmentsTest>") return -1;
00248
00249 in.close();
00250
00251 if(this->NumSegments()>0) {
00252     if(this->ReadTargetEffectsFile(configure,theCNuc)==-1) return -1;
00253     this->MapData();
00254 }
00255
00256 return 0;
00257 }
00258
00263 int EData::Iterations() const {
00264     return iterations_;
00265 }
00266
00271 int EData::NumTargetEffects() const {
00272     return targetEffects_.size();
00273 }
00274
00278 int EData::GetNormParamOffset() const {
00279     return normParamOffset_;
00280 }
00281
00287 int EData::ReadTargetEffectsFile(const Config& configure, CNuc *compound) {
00288     std::ifstream in(configure.configfile.c_str());
00289     if(!in) return -1;
00290     std::string line="";
00291     while(line!="<targetInt>"&&!in.eof()) getline(in,line);
00292     if(line!="<targetInt>") return -1;
00293     line="";
00294     while(line!="</targetInt>"&&!in.eof()) {
00295         getline(in,line);
00296         bool empty=true;
00297         for(unsigned int i=0;i<line.size();++i)
00298             if(line[i]!=' ' && line[i]!='\t') {
00299                 empty=false;
00300                 break;
00301             }
00302         if(empty==true) continue;
00303         if(line!="</targetInt>"&&!in.eof()){
00304             std::istringstream stm;
00305             stm.str(line);
00306             TargetEffect targetEffect(stm,configure);

```

```

00307         if(stm.rdstate() & (std::stringstream::failbit | std::stringstream::badbit)) return -1;
00308         if(targetEffect.IsActive()) {
00309             this->AddTargetEffect(targetEffect);
00310             TargetEffect *thisTargetEffect=this->GetTargetEffect(this->NumTargetEffects());
00311             std::vector<int> segmentsList = thisTargetEffect->GetSegmentsList();
00312             for(int i = 1;i<=segmentsList.size();i++) {
00313                 if(this->IsSegmentKey(segmentsList[i-1]))
00314                     this->GetSegmentFromKey(segmentsList[i-1])>SetTargetEffectNum(this->NumTargetEffects());
00315             }
00316         }
00317     }
00318 }
00319 if(line!="</targetInt>") return -1;
00320 for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
00321     PPair *entrancePair = compound->GetPair(compound->GetPairNumFromKey(segment->GetEntranceKey()));
00322     PPair *exitPair = compound->GetPair(compound->GetPairNumFromKey(segment->GetExitKey()));
00323     double cmConversion;
00324     if(entrancePair->GetPType()==20)
00325         cmConversion = (exitPair->GetM(1)+exitPair->GetM(2))/exitPair->GetM(2);
00326     else
00327         cmConversion = entrancePair->GetM(2)/(entrancePair->GetM(1)+entrancePair->GetM(2));
00328     if(segment->IsTargetEffect()) {
00329         TargetEffect *targetEffect = this->GetTargetEffect(segment->GetTargetEffectNum());
00330         double sigma = targetEffect->GetSigma();
00331         targetEffect->SetSigma(cmConversion*sigma);
00332         for(EPointIterator point=segment->GetPoints().begin();point<segment->GetPoints().end();point++)
00333         {
00334             point->SetTargetEffectNum(segment->GetTargetEffectNum());
00335             if(targetEffect->IsTargetIntegration()||targetEffect->IsConvolution()) {
00336                 double forwardDepth=0.0;
00337                 double backwardDepth=0.0;
00338                 if(targetEffect->IsTargetIntegration()) {
00339                     double totalM=entrancePair->GetM(1)+entrancePair->GetM(2);
00340                     double targetThickness =
00341                         cmConversion*targetEffect->TargetThickness(point->GetLabEnergy(),configure);
00342                     point->SetTargetThickness(targetThickness);
00343                     if(targetEffect->IsConvolution()) {
00344                         backwardDepth=targetThickness+targetEffect->convolutionRange*targetEffect->GetSigma();
00345                         forwardDepth=targetEffect->convolutionRange*targetEffect->GetSigma();
00346                     } else {
00347                         backwardDepth=targetThickness;
00348                         forwardDepth=0.0;
00349                     }
00350                 } else if(targetEffect->IsConvolution()) {
00351                     backwardDepth=targetEffect->convolutionRange*targetEffect->GetSigma();
00352                     forwardDepth=targetEffect->convolutionRange*targetEffect->GetSigma();
00353                 }
00354                 for(int i=0;i<targetEffect->NumSubPoints();i++) {
00355                     double subEnergy=point->GetCMEnergy()+forwardDepth
00356                         -(forwardDepth+backwardDepth)/(targetEffect->NumSubPoints())*i;
00357                     EPoint subPoint(point->GetCMAngle(),subEnergy,&*segment);
00358                     if(targetEffect->IsTargetIntegration()) {
00359                         double stoppingPower=cmConversion*targetEffect->
00360                             GetStoppingPowerEq()->Evaluate(configure,subEnergy/cmConversion);
00361                         subPoint.SetStoppingPower(stoppingPower);
00362                     }
00363                     point->AddSubPoint(subPoint);
00364                 }
00365             }
00366         }
00367     }
00368 }
00369
00375 bool EData::IsFit() const {
00376     return isFit_;
00377 }
00378
00384 bool EData::IsErrorAnalysis() const {
00385     return isErrorAnalysis_;
00386 }
00387
00398 bool EData::IsSegmentKey(int segmentKey) {
00399     bool isKey=false;
00400     for (ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
00401         if(segment->GetSegmentKey()==segmentKey) {
00402             isKey=true;
00403             break;
00404         }
00405     }
00406     return isKey;
00407 }
00408
00414 void EData::SetFit(bool fit) {
00415     isFit_=fit;
00416 }

```



```

00417
00422 void EData::SetErrorAnalysis(bool errorAnalysis) {
00423     isErrorAnalysis_=errorAnalysis;
00424 }
00425
00430 void EData::Iterate(){
00431     iterations_++;
00432 }
00433
00438 void EData::ResetIterations(){
00439     iterations_=0;
00440 }
00441
00447 int EData::Initialize(CNuc *compound,const Config &configure) {
00448     //Calculate channel lo-matrix and channel penetrability for each channel at each local energy
00449     configure.outStream << "Calculating Lo-Matrix, Phases, and Penetrabilities..." << std::endl;
00450     if(this->CalcEDependentValues(compound,configure)==-1) return -1;
00451     if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_ENERGY_DEP)
00452         this->PrintEDependentValues(configure,compound);
00453
00454     //Calculate legendre polynomials for each data point
00455     configure.outStream << "Calculating Legendre Polynomials..." << std::endl;
00456     this->CalcLegendreP(configure.maxLOrder);
00457     if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_LEGENDRE)
00458         this->PrintLegendreP(configure);
00459
00460     //Calculate Coulomb Amplitudes
00461     configure.outStream << "Calculating Coulomb Amplitudes..." << std::endl;
00462     this->CalcCoulombAmplitude(compound);
00463     if((configure.fileCheckMask|configure.screenCheckMask) & Config::CHECK_COUL_AMPLITUDES) {
00464         this->PrintCoulombAmplitude(configure,compound);
00465     }
00466
00467     //Calculate new ec amplitudes
00468     if(configure.paramMask & Config::USE_EXTERNAL_CAPTURE) {
00469         configure.outStream << "Calculating External Capture Amplitudes..." << std::endl;
00470         if(this->CalculateECAmplitudes(compound,configure)==-1) return -1;
00471     }
00472     return 0;
00473 }
00474
00479 void EData::AddSegment(ESegment segment) {
00480     segments_.push_back(segment);
00481 }
00482
00487 void EData::PrintData(const Config &configure) {
00488     std::streambuf *sbuffer;
00489     std::filebuf fbuffer;
00490     if(configure.fileCheckMask & Config::CHECK_DATA) {
00491         std::string outfile=configure.checkdir+"data.chk";
00492         fbuffer.open(outfile.c_str(),std::ios::out);
00493         sbuffer = &fbuffer;
00494     } else if(configure.screenCheckMask & Config::CHECK_DATA) sbuffer = configure.outStream.rdbuf();
00495     std::ostream out(sbuffer);
00496     if(((configure.fileCheckMask & Config::CHECK_DATA)&&fbuffer.is_open())||
00497         (configure.screenCheckMask & Config::CHECK_DATA)) {
00498         out << std::endl
00499         << "*****" << std::endl
00500         << " *           Segments           * " << std::endl
00501         << "*****" << std::endl;
00502         out << std::setw(11) << "Segment #"
00503         << std::setw(17) << "Segment Key #"
00504         << std::setw(17) << "Entrance Key #"
00505         << std::setw(13) << "Exit Key #"
00506         << std::setw(12) << "Min Energy"
00507         << std::setw(12) << "Max Energy"
00508         << std::setw(11) << "Min Angle"
00509         << std::setw(11) << "Max Angle"
00510         << std::setw(25) << "Data File"
00511         << std::endl;
00512         for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
00513             out << std::setw(11) << segment-GetSegments().begin()+1
00514             << std::setw(17) << segment->GetSegmentKey()
00515             << std::setw(17) << segment->GetEntranceKey()
00516             << std::setw(13) << segment->GetExitKey()
00517             << std::setw(12) << segment->GetMinEnergy()
00518             << std::setw(12) << segment->GetMaxEnergy()
00519             << std::setw(11) << segment->GetMinAngle()
00520             << std::setw(11) << segment->GetMaxAngle()
00521             << std::setw(25) << segment->GetDataFile()
00522             << std::endl;
00523         }
00524         out << std::endl
00525         << "*****" << std::endl
00526         << " *           Data           * " << std::endl
00527         << "*****" << std::endl;
00528         out << std::setw(11) << "Segment #"

```

```

00529     « std::setw(14) « "Data Point #"
00530     « std::setw(15) « "Lab Energy"
00531     « std::setw(15) « "CM Energy"
00532     « std::setw(15) « "Angle"
00533     « std::setw(20) « "Cross Section"
00534     « std::setw(22) « "Cross Section Error"
00535     « std::setw(12) « "Map Point"
00536     « std::setw(18) « "# of Subpoints"
00537     « std::setw(18) « "Low Sub Energy"
00538     « std::setw(18) « "High Sub Energy"
00539     « std::endl;
00540     for(EDataIterator data=begin();data!=end();data++) {
00541         out « std::setw(11) « data.segment()->GetSegments().begin()+1
00542         « std::setw(14) « data.point()-(data.segment()->GetPoints()).begin()+1
00543         « std::setw(15) « data.point()->GetLabEnergy()
00544         « std::setw(15) « data.point()->GetCMEnergy()
00545         « std::setw(15) « data.point()->GetCMAngle()
00546         « std::setw(20) « data.point()->GetCMCrossSection()
00547         « std::setw(22) « data.point()->GetCMCrossSectionError();
00548         if(data.point()->IsMapped()){
00549             EnergyMap map=data.point()->GetMap();
00550             char tempMap[25];
00551             sprintf(tempMap,"%d,%d",map.segment,map.point);
00552             out « std::setw(12) « tempMap « std::endl;
00553         } else
00554             out « std::setw(12) « "Not Mapped"
00555             « std::setw(18) « data.point()->NumSubPoints();
00556         if(data.point()->IsTargetEffect() &&
00557            (data.point()->GetParentData()->GetTargetEffect(data.point()->GetTargetEffectNum())->IsConvolution() ||
00558            data.point()->GetParentData()->GetTargetEffect(data.point()->GetTargetEffectNum())->IsTargetIntegration()))
00559         {
00560             out « std::setw(18) « data.point()->GetSubPoint(data.point()->NumSubPoints())->GetCMEnergy()
00561             « std::setw(18) « data.point()->GetSubPoint(1)->GetCMEnergy();
00562             out « std::endl;
00563             if(data.point()==data.segment()->GetPoints().end()-1) out « std::endl;
00564         }
00565     } else configure.outStream « "Could not write data check file." « std::endl;
00566     out.flush();
00567     if(fbbuffer.is_open()) fbbuffer.close();
00568 }
00569
00574 void EData::CalcLegendreP(int maxL) {
00575     for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
00576         TargetEffect *effect = (segment->IsTargetEffect() &&
00577             this->GetTargetEffect(segment->GetTargetEffectNum())->IsQCoefficients()) ?
00578             this->GetTargetEffect(segment->GetTargetEffectNum()) : NULL;
00579     #pragma omp parallel for
00580     for(int i=1;i<=segment->NumPoints();i++) {
00581         EPoint* point=segment->GetPoint(i);
00582         point->CalcLegendreP(maxL, effect);
00583     }
00584 }
00585 }
00586
00591 void EData::PrintLegendreP(const Config &configure) {
00592     std::streambuf *sbuffer;
00593     std::filebuf fbbuffer;
00594     if(configure.fileCheckMask & Config::CHECK_LEGENDRE) {
00595         std::string outfile=configure.checkdir+"legendre.chk";
00596         fbbuffer.open(outfile.c_str(),std::ios::out);
00597         sbuffer = &fbbuffer;
00598     } else if(configure.screenCheckMask & Config::CHECK_LEGENDRE) sbuffer = configure.outStream.rdbuf();
00599     std::ostream out(sbuffer);
00600     if(((configure.fileCheckMask & Config::CHECK_LEGENDRE)&&fbbuffer.is_open()) ||
00601        (configure.screenCheckMask & Config::CHECK_LEGENDRE)) {
00602         out « std::endl
00603         « "*****" « std::endl
00604         « "      Legendre Polynomials      " « std::endl
00605         « "*****" « std::endl;
00606         out « std::setw(10) « "Segment #"
00607         « std::setw(10) « "Point #"
00608         « std::setw(15) « "CM Energy"
00609         « std::setw(15) « "Angle"
00610         « std::setw(5) « "L"
00611         « std::setw(15) « "Leg. Poly." « std::endl;
00612         for(EDataIterator data=begin();data!=end();data++) {
00613             for(int lOrder=0;lOrder<=data.point()->GetMaxLOrder();lOrder++) {
00614                 out « std::setw(10) « data.segment()->GetSegments().begin()+1
00615                 « std::setw(10) « data.point()-(data.segment()->GetPoints()).begin()+1
00616                 « std::setw(15) « data.point()->GetCMEnergy()
00617                 « std::setw(15) « data.point()->GetCMAngle()
00618                 « std::setw(5) « lOrder
00619                 « std::setw(15) « data.point()->GetLegendreP(lOrder) « std::endl;
00620             }

```

```

00621     }
00622 } else configure.outStream << "Could not write legendre polynomials check file." << std::endl;
00623 out.flush();
00624 if(fbuffer.is_open()) fbuffer.close();
00625 }
00626
00631 int EData::CalcEDependentValues(CNuc *theCNuc,const Config& configure) {
00632     for(ESegmentIterator segment=GetSegments().begin(); segment<GetSegments().end(); segment++) {
00633         bool localStop = false;
00634 #pragma omp parallel for shared(localStop,configure)
00635         for(int i=1;i<=segment->NumPoints();i++) {
00636             if(configure.stopFlag||localStop) continue;
00637             EPoint *point = segment->GetPoint(i);
00638             if(!(point->IsMapped())){
00639                 try {
00640                     point->CalcEDependentValues(theCNuc,configure);
00641                 } catch(GSLEException e) {
00642 #pragma omp critical
00643                 {
00644                     configure.outStream << e.what() << std::endl;
00645                     localStop = true;
00646                 }
00647             }
00648         }
00649     }
00650     if(configure.stopFlag||localStop) return -1;
00651 }
00652 return 0;
00653 }
00654
00659 void EData::PrintEDependentValues(const Config &configure,CNuc *theCNuc) {
00660     std::streambuf *sbuffer;
00661     std::filebuf fbuffer;
00662     if(configure.fileCheckMask & Config::CHECK_ENERGY_DEP) {
00663         std::string outfile=configure.checkdir+"lomatrixandpene.chk";
00664         fbuffer.open(outfile.c_str(),std::ios::out);
00665         sbuffer = &fbuffer;
00666     } else if(configure.screenCheckMask & Config::CHECK_ENERGY_DEP) sbuffer =
configure.outStream.rdbuf();
00667     std::ostream out(sbuffer);
00668     if(((configure.fileCheckMask & Config::CHECK_ENERGY_DEP)&&fbuffer.is_open())||
configure.screenCheckMask & Config::CHECK_ENERGY_DEP) {
00669         out << std::endl
00670         << "*****" << std::endl
00671         << " * Lo Matrix and Penetrabilities * " << std::endl
00672         << "*****" << std::endl;
00673         out << std::setw(10) << "Seg #"
00674         << std::setw(10) << "Point #"
00675         << std::setw(5) << "j"
00676         << std::setw(5) << "ch"
00677         << std::setw(5) << "l"
00678         << std::setw(15) << "E chan"
00679         << std::setw(15) << "pene"
00680         << std::setw(25) << "Lo" << std::endl;
00681         for(EDataIterator data=begin();data!=end();data++) {
00682             double inEnergy=data.point()->GetCMEnergy()
00683             +theCNuc->GetPair(theCNuc->GetPairNumFromKey(data.segment()->GetEntranceKey()))->GetSepE();
00684             for(int j=1;j<=theCNuc->NumJGroups();j++) {
00685                 if(theCNuc->GetJGroup(j)->IsInRMatrix()) {
00686                     JGroup *theJGroup=theCNuc->GetJGroup(j);
00687                     for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
00688                         AChannel *theChannel=theJGroup->GetChannel(ch);
00689                         PPair *thePair=theCNuc->GetPair(theChannel->GetPairNum());
00690                         int lValue=theChannel->GetL();
00691                         double localEnergy=inEnergy-thePair->GetSepE()-thePair->GetExE();
00692                         out << std::setw(10) << data.segment()->GetSegments().begin()+1
00693                         << std::setw(10) << data.point()-(data.segment()->GetPoints()).begin()+1
00694                         << std::setw(5) << j
00695                         << std::setw(5) << ch
00696                         << std::setw(5) << lValue
00697                         << std::setw(15) << localEnergy
00698                         << std::setw(15) << data.point()->GetSqrtPenetrability(j,ch)
00699                         << std::setw(25) << data.point()->GetLoElement(j,ch) << std::endl;
00700                     }
00701                 }
00702             }
00703         }
00704     }
00705     } else configure.outStream << "Could not write lo-matrix and penetrabilities check file." <<
std::endl;
00706     out.flush();
00707     if(fbuffer.is_open()) fbuffer.close();
00708 }
00709
00714 void EData::CalcCoulombAmplitude(CNuc *theCNuc) {
00715     for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
00716 #pragma omp parallel for
00717         for(int i=1;i<=segment->NumPoints();i++) {

```

```

00718         EPoint* point = segment->GetPoint(i);
00719         point->CalcCoulombAmplitude(theCNuc);
00720     }
00721 }
00722 }
00723
00728 void EData::PrintCoulombAmplitude(const Config &configure, CNuc *theCNuc) {
00729     std::streambuf *sbuffer;
00730     std::filebuf fbuffer;
00731     if(configure.fileCheckMask & Config::CHECK_COUL_AMPLITUDES) {
00732         std::string outfile=configure.checkdir+"coulombamplitudes.chk";
00733         fbuffer.open(outfile.c_str(),std::ios::out);
00734         sbuffer = &fbuffer;
00735     } else if(configure.screenCheckMask & Config::CHECK_COUL_AMPLITUDES) sbuffer =
configure.outStream.rdbuf();
00736     std::ostream out(sbuffer);
00737     if(((configure.fileCheckMask & Config::CHECK_COUL_AMPLITUDES)&&fbuffer.is_open())||
00738         (configure.screenCheckMask & Config::CHECK_COUL_AMPLITUDES)) {
00739         out << std::endl
00740         << "*****" << std::endl
00741         << "          Coulomb Amplitudes          " << std::endl
00742         << "*****" << std::endl;
00743         out << std::setw(10) << "segment #"
00744         << std::setw(10) << "point #"
00745         << std::setw(10) << "aa"
00746         << std::setw(15) << "cmenergy"
00747         << std::setw(15) << "angle"
00748         << std::setw(25) << "coulomb amplitude"
00749         << std::endl;
00750         for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
00751             if(segment->GetEntranceKey()==segment->GetExitKey()) {
00752                 for(EPointIterator point=segment->GetPoints().begin();point<segment->GetPoints().end();point++) {
00753                     out << std::setw(10) << segment->GetSegments().begin()+1
00754                     << std::setw(10) << point-segment->GetPoints().begin()+1
00755                     << std::setw(10) << theCNuc->GetPairNumFromKey(segment->GetEntranceKey())
00756                     << std::setw(15) << point->GetCMEnergy()
00757                     << std::setw(15) << point->GetCMAngle()
00758                     << std::setw(25) << point->GetCoulombAmplitude()
00759                     << std::endl;
00760                 }
00761             }
00762         }
00763     } else configure.outStream << "Could not write coulomb amplitudes check file." << std::endl;
00764     out.flush();
00765     if(fbuffer.is_open()) fbuffer.close();
00766 }
00767
00774 void EData::WriteOutputFiles(const Config &configure, bool isFit) {
00775     AZUREOutput output(configure.outputdir);
00776     std::ofstream chiOut;
00777     if(!isFit&&(configure.paramMask & Config::CALCULATE_WITH_DATA)) {
00778         std::string chiOutFile = configure.outputdir+"chiSquared.out";
00779         chiOut.open(chiOutFile.c_str());
00780     }
00781     if(!(configure.paramMask & Config::CALCULATE_WITH_DATA)) output.SetExtrap();
00782     bool isVaryNorm=false;
00783     double totalChiSquared=0.;
00784     ESegmentIterator firstSumIterator = GetSegments().end();
00785     for(ESegmentIterator segment=GetSegments().begin();
00786         segment<GetSegments().end();segment++) {
00787         if(segment->IsTotalCapture()) {
00788             firstSumIterator=segment;
00789             segment+=segment->IsTotalCapture()-1;
00790         }
00791         if(segment->IsVaryNorm()) isVaryNorm=true;
00792         int aa=segment->GetEntranceKey();
00793         int ir=segment->GetExitKey();
00794         std::filebuf* buf;
00795         if(firstSumIterator!=GetSegments().end()) buf=output(aa,-1);
00796         else {
00797             if(segment->IsAngularDist()&&
00798                 !(configure.paramMask & Config::CALCULATE_WITH_DATA)) buf=output(aa,ir,true);
00799             else buf=output(aa,ir);
00800         }
00801         std::ostream out(buf);
00802         ESegmentIterator thisSegment = segment;
00803         if(firstSumIterator!=GetSegments().end()) thisSegment = firstSumIterator;
00804         for(EPointIterator point=thisSegment->GetPoints().begin();point<thisSegment->GetPoints().end();point++) {
00805             out.precision(6);
00806             if(segment->IsAngularDist()) {
00807                 out << std::setw(15) << std::scientific << point->GetCMEnergy();
00808                 for(int i = 0;i<point->GetNumAngularDists();i++) out << std::setw(15) << point->GetAngularDist(i);
00809                 out << std::endl;
00810             } else {
00811                 double fitCrossSection=point->GetFitCrossSection();
00812                 if(firstSumIterator!=GetSegments().end()) {
00813                     int pointIndex=point-segment->GetPoints().begin()+1;

```

```

00814         for(ESegmentIterator it=firstSumIterator;it<segment;it++)
00815             fitCrossSection+=it->GetPoint(pointIndex)->GetFitCrossSection();
00816     }
00817     out << std::setw(15) << std::scientific << point->GetCMEnergy()
00818         << std::setw(15) << std::scientific << point->GetExcitationEnergy()
00819         << std::setw(15) << std::scientific << point->GetCMAngle()
00820         << std::setw(15) << std::scientific << fitCrossSection
00821         << std::setw(15) << std::scientific << fitCrossSection*point->GetSFactorConversion();
00822     if(!output.IsExtrap()) {
00823         double dataNorm=thisSegment->GetNorm();
00824         out << std::setw(15) << std::scientific << point->GetCMCrossSection()*dataNorm
00825             << std::setw(15) << std::scientific << point->GetCMCrossSectionError()*dataNorm
00826             << std::setw(15) << std::scientific <<
00827         point->GetCMCrossSection()*dataNorm*point->GetSFactorConversion()
00828             << std::setw(15) << std::scientific <<
00829         point->GetCMCrossSectionError()*dataNorm*point->GetSFactorConversion()
00830             << std::endl;
00831     } else out << std::endl;
00832 }
00833 if(!isFit&&(configure.paramMask & Config::CALCULATE_WITH_DATA)) {
00834     totalChiSquared+=thisSegment->GetSegmentChiSquared();
00835     chiOut << "Segment #"
00836         << thisSegment->GetSegmentKey()
00837         << " Chi-Squared/N: "
00838         << thisSegment->GetSegmentChiSquared()/thisSegment->NumPoints()
00839         << std::endl;
00840 }
00841 out<<std::endl<<std::endl;out.flush();
00842 firstSumIterator=GetSegments().end();
00843 }
00844 if(!isFit&&(configure.paramMask & Config::CALCULATE_WITH_DATA)) {
00845     chiOut << "Total Chi-Squared: "
00846         << totalChiSquared << std::endl << std::endl;
00847     chiOut.flush();chiOut.close();
00848 }
00849 if(isVaryNorm) {
00850     std::string outputfile=configure.outputdir+"normalizations.out";
00851     std::ofstream out(outputfile.c_str());
00852     if(out) {
00853         out.precision(6);
00854         out << std::scientific;
00855         for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
00856             if(segment->IsVaryNorm()) out << std::setw(20) << "Segment Key #" << segment->GetSegmentKey()
00857                 << std::setw(20) << segment->GetNorm() << std::endl;
00858             out.flush();
00859             out.close();
00860         } else configure.outStream << "Could not write normalization file." << std::endl;
00861     }
00862 }
00863
00870 int EData::CalculateECAmplitudes(CNuc *theCNuc,const Config& configure) {
00871     std::ifstream in;
00872     std::ofstream out;
00873     std::string outputfile;
00874     if(configure.paramMask & Config::CALCULATE_WITH_DATA) outputfile=configure.outputdir+"intEC.dat";
00875     else outputfile=configure.outputdir+"intEC.extrap";
00876     if(configure.paramMask & Config::USE_PREVIOUS_INTEGRALS) in.open(configure.integralsfile.c_str());
00877     else {
00878         out.open(outputfile.c_str());
00879         if(!out) configure.outStream << "Could not write to EC Amplitude File." << std::endl;
00880     }
00881     int sumSegmentI=0;
00882     int numSumSegments=0;
00883     for(ESegmentIterator segment=GetSegments().begin();segment<GetSegments().end();segment++) {
00884         if(segment->IsTotalCapture()) {
00885             numSumSegments = segment->IsTotalCapture();
00886             sumSegmentI=0;
00887         }
00888         if(numSumSegments) sumSegmentI++;
00889         char segmentKeyOut[256];
00890         if(numSumSegments) sprintf(segmentKeyOut,"%d
00891 (%d/%d)",segment->GetSegmentKey(),sumSegmentI,numSumSegments);
00892         else sprintf(segmentKeyOut,"%d",segment->GetSegmentKey());
00893         int aa=theCNuc->GetPairNumFromKey(segment->GetEntranceKey());
00894         if(theCNuc->GetPair(aa)->GetPType()==20) continue;
00895         if(theCNuc->GetPair(aa)->IsEntrance()) {
00896             PPair *entrancePair=theCNuc->GetPair(aa);
00897             for(int j=1;j<=theCNuc->NumJGroups();j++) {
00898                 for(int la=1;la<=theCNuc->GetJGroup(j)->NumLevels();la++) {
00899                     if(theCNuc->GetJGroup(j)->GetLevel(la)->IsECLevel()) {
00900                         ALevel *ecLevel = theCNuc->GetJGroup(j)->GetLevel(la);
00901                         int ir=theCNuc->GetPairNumFromKey(segment->GetExitKey());
00902                         if(ecLevel->GetECPairNum()==ir) {
00903                             if(!configure.paramMask & Config::USE_PREVIOUS_INTEGRALS) {
00904                                 configure.outStream << "\tSegment #" << std::setw(12) << segmentKeyOut

```

```

00904         « std::setw(0) « " [                                ] 0%";configure.outStream.flush();
00905     int numPoints=segment->NumPoints();
00906     int pointIndex=0;
00907     time_t startTime = time(NULL);
00908     bool localStop = false;
00909 #pragma omp parallel for shared(configure,localStop)
00910     for(int i=1;i<=numPoints;i++) {
00911         if(configure.stopFlag||localStop) continue;
00912         EPoint *point = segment->GetPoint(i);
00913         if(!(point->IsMapped())) {
00914             try {
00915                 point->CalculateECAmplitudes(theCNuc,configure);
00916             } catch(GSLEException e) {
00917 #pragma omp critical
00918                 {
00919                     configure.outStream « e.what() « std::endl;
00920                     localStop=true;
00921                 }
00922             }
00923         }
00924         ++pointIndex;
00925         if(difftime(time(NULL),startTime)>0.25) {
00926             startTime=time(NULL);
00927             std::string progress=" [";
00928             double percent=0.;
00929             for(int j = 1;j<=25;j++) {
00930                 if(pointIndex>=percent*numPoints&&percent<1.) {
00931                     percent+=0.04;
00932                     progress+='*';
00933                 } else progress+=' ';
00934             } progress+="] ";
00935             configure.outStream « "\r\tSegment #" « std::setw(12) « segmentKeyOut
00936                 « std::setw(0) « progress « percent*100 « '%';configure.outStream.flush();
00937         }
00938     }
00939     if(configure.stopFlag||localStop) {
00940         if(out.is_open()) out.close();
00941         if(in.is_open()) in.close();
00942         return -1;
00943     }
00944     configure.outStream « "\r\tSegment #" « std::setw(12) « segmentKeyOut
00945         « std::setw(0) « " [*****] 100% " « std::endl;
00946     }
00947     for(EPointIterator point=segment->GetPoints().begin();
00948         point<segment->GetPoints().end();point++) {
00949         if(!(point->IsMapped())) {
00950             for(int k=1;k<=entrancePair->GetDecay(ir)->NumKGroups();k++) {
00951                 for(int ecm=1;ecm<=entrancePair->GetDecay(ir)->GetKGroup(k)->NumECMGroups();ecm++) {
00952                     if(!(configure.paramMask & Config::USE_PREVIOUS_INTEGRALS)) {
00953                         if(out.is_open()) out « point->GetECAmplitude(k,ecm) « std::endl;
00954                         for(EPointIterator subPoint=point->GetSubPoints().begin();
00955                             subPoint<point->GetSubPoints().end();subPoint++)
00956                             if(out.is_open()) out « subPoint->GetECAmplitude(k,ecm) « std::endl;
00957                         } else {
00958                             complex ecAmplitude(0.0,0.0);
00959                             in « ecAmplitude;
00960                             point->AddECAmplitude(k,ecm,ecAmplitude);
00961                             for(EPointIterator subPoint=point->GetSubPoints().begin();
00962                                 subPoint<point->GetSubPoints().end();subPoint++) {
00963                                 ecAmplitude=complex(0.0,0.0);
00964                                 in « ecAmplitude;
00965                                 subPoint->AddECAmplitude(k,ecm,ecAmplitude);
00966                             }
00967                         }
00968                         for(EPointMapIterator mappedPoint=point->GetMappedPoints().begin();
00969                             mappedPoint<point->GetMappedPoints().end();mappedPoint++) {
00970                             (*mappedPoint)->AddECAmplitude(k,ecm,point->GetECAmplitude(k,ecm));
00971                             for(int i=1;i<=point->NumSubPoints();i++) {
00972                                 (*mappedPoint)->GetSubPoint(i)->
00973                                     AddECAmplitude(k,ecm,point->GetSubPoint(i)->GetECAmplitude(k,ecm));
00974                             }
00975                         }
00976                     }
00977                 }
00978             }
00979         }
00980     }
00981 }
00982 }
00983 }
00984 }
00985 if(sumSegmentI==numSumSegments) {
00986     sumSegmentI=0;
00987     numSumSegments=0;
00988 }
00989 }
00990 if(out.is_open()) {

```

```

00991     out.flush();
00992     out.close();
00993 }
00994 if(in.is_open()) in.close();
00995 return 0;
00996 }
00997
01003 void EData::MapData() {
01004     for(ESegmentIterator segment=GetSegments().end()-1;
01005         segment>=GetSegments().begin(); segment--) {
01006         for(EPointIterator point=segment->GetPoints().end()-1;
01007             point>=segment->GetPoints().begin(); point--) {
01008             if(point->NumLocalMappedPoints()==0) {
01009                 for(ESegmentIterator testSegment=GetSegments().begin();
01010                     testSegment<GetSegments().end(); testSegment++) {
01011                     if(testSegment->GetEntranceKey()==segment->GetEntranceKey() &&
01012                         testSegment->GetExitKey()==segment->GetExitKey()) {
01013                         for(EPointIterator testPoint=testSegment->GetPoints().begin();
01014                             testPoint<testSegment->GetPoints().end(); testPoint++) {
01015                             if(testPoint->GetCMEnergy()==point->GetCMEnergy()
01016                                 &&!testPoint->IsMapped() &&point!=testPoint
01017                                 &&testPoint->GetTargetEffectNum()==point->GetTargetEffectNum()) {
01018                                 point->SetMap(testSegment->GetSegments().begin()+1,
01019                                     testPoint-testSegment->GetPoints().begin()+1);
01020                                 testPoint->AddLocalMappedPoint(&*point);
01021                                 break;
01022                             }
01023                         }
01024                     if(point->IsMapped()) break;
01025                 }
01026             }
01027         }
01028     }
01029 }
01030 }
01031
01036 void EData::AddTargetEffect(TargetEffect targetEffect) {
01037     targetEffects_.push_back(targetEffect);
01038 }
01039
01044 void EData::SetNormParamOffset(int offset) {
01045     normParamOffset_=offset;
01046 }
01047
01052 void EData::FillMnParams(ROOT::Minuit2::MnUserParameters &p) {
01053     SetNormParamOffset(p.Params().size());
01054     char varname[50];
01055     for(ESegmentIterator segment=GetSegments().begin(); segment<GetSegments().end(); segment++) {
01056         if(segment->IsVaryNorm()) {
01057             sprintf(varname, "segment_%d_norm", segment->GetSegmentKey());
01058             p.Add(varname, segment->GetNorm(), segment->GetNorm()*0.05);
01059         }
01060         if(segment->IsTotalCapture()) segment+=segment->IsTotalCapture()-1;
01061     }
01062 }
01063
01064
01069 void EData::DeleteLastSegment() {
01070     segments_.pop_back();
01071 }
01072
01077 void EData::FillNormsFromParams(const vector_r &p) {
01078     int i=GetNormParamOffset();
01079     for(ESegmentIterator segment=GetSegments().begin(); segment<GetSegments().end(); segment++) {
01080         if(segment->IsVaryNorm()) {
01081             segment->SetNorm(p[i]);
01082             i++;
01083         }
01084         if(segment->IsTotalCapture()) segment+=segment->IsTotalCapture()-1;
01085     }
01086 }
01087
01092 ESegment *EData::GetSegment(int segmentNum) {
01093     ESegment *b=&segments_[segmentNum-1];
01094     return b;
01095 }
01096
01101 ESegment *EData::GetSegmentFromKey(int segmentKey) {
01102     int segmentNumber=1;
01103     while(segmentNumber <= this->NumSegments()) {
01104         if(segmentKey==this->GetSegment(segmentNumber)->GetSegmentKey())
01105             break;
01106         else segmentNumber++;
01107     }
01108     if(segmentNumber<= this->NumSegments()) return this->GetSegment(segmentNumber);
01109     else return NULL;
01110 }

```

```

01111
01117 EData *EData::Clone() const {
01118     EData *dataCopy = new EData(*this);
01119
01120     for(EDataIterator data=dataCopy->begin();data!=dataCopy->end();data++) {
01121         data.point()->SetParentData(dataCopy);
01122         data.point()->ClearLocalMappedPoints();
01123     }
01124     for(EDataIterator data=dataCopy->begin();data!=dataCopy->end();data++) {
01125         if(data.point()->IsMapped()) {
01126             EnergyMap pointMap = data.point()->GetMap();
01127             dataCopy->GetSegment(pointMap.segment)->GetPoint(pointMap.point)->AddLocalMappedPoint(&*data.point());
01128         }
01129     }
01130     return dataCopy;
01131 }
01132
01137 TargetEffect *EData::GetTargetEffect(int effectNumber) {
01138     TargetEffect *temp;
01139     if(effectNumber<=targetEffects_.size())
01140         temp=&targetEffects_[effectNumber-1];
01141     else return temp=NULL;
01142     return temp;
01143 }
01144
01149 EDataIterator EData::begin() {
01150     return EDataIterator(&segments_);
01151 }
01152
01157 EDataIterator EData::end() {
01158     EDataIterator it(&segments_);
01159     return it.SetEnd();
01160 }
01161
01166 std::vector<ESegment>& EData::GetSegments() {
01167     return segments_;
01168 }

```

8.256 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EDatalterator.cpp File Reference

```

#include "EDataIterator.h"
#include "ESegment.h"

```

8.257 EDatalterator.cpp

[Go to the documentation of this file.](#)

```

00001 #include "EDataIterator.h"
00002 #include "ESegment.h"
00003
00004
00009 EDataIterator::EDataIterator(std::vector<ESegment>* segments) :
00010     segments_(segments) {
00011     segmentIterator_=segments->begin();
00012     pointIterator_=(*segmentIterator_).GetPoints().begin();
00013 }
00014
00020 EDataIterator::EDataIterator(const EDataIterator& it) :
00021     segments_(it.segments_), segmentIterator_(it.segmentIterator_), pointIterator_(it.pointIterator_) {
00022 }
00023
00028 EDataIterator& EDataIterator::operator++() {
00029     if(pointIterator_ < ((*segmentIterator_).GetPoints().end()-1) {
00030         pointIterator_++;
00031     } else if(segmentIterator_ < segments->end()-1) {
00032         segmentIterator_++;
00033         pointIterator_=(*segmentIterator_).GetPoints().begin();
00034     } else if ((segmentIterator_ == (segments->end()-1)) &&
00035         (pointIterator_ == ((*segmentIterator_).GetPoints().end()-1)))
00036         pointIterator_++;
00037     return *this;

```



```

00038 }
00039
00044 EDataIterator EDataIterator::operator++(int) {
00045     EDataIterator temp(*this);
00046     operator++();
00047     return temp;
00048 }
00049
00054 bool EDataIterator::operator==(const EDataIterator& rhs) {
00055     return (rhs.segmentIterator_==segmentIterator_ && rhs.pointIterator_==pointIterator_) ? (true) :
        (false);
00056 }
00057
00062 bool EDataIterator::operator!=(const EDataIterator& rhs) {
00063     return (rhs.segmentIterator_==segmentIterator_ && rhs.pointIterator_==pointIterator_) ? (false) :
        (true);
00064 }
00065
00071 EDataIterator& EDataIterator::SetEnd() {
00072     segmentIterator_=segments_>end()-1;
00073     pointIterator_=(*segmentIterator_).GetPoints().end();
00074     return *this;
00075 }
00076
00081 ESegmentIterator& EDataIterator::segment() {
00082     return segmentIterator_;
00083 }
00084
00089 EPointIterator& EDataIterator::point() {
00090     return pointIterator_;
00091 }

```

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.](#)

```
00001 #include "EffectiveCharge.h"
00002 #include "PPair.h"
00003 #include "Constants.h"
00004 #include <gsl/gsl_sf_bessel.h>
00005 #include <gsl/gsl_integration.h>
00006
00007 extern double DoubleFactorial(int);
00008
00013 EffectiveCharge::EffectiveCharge(PPair* pair, double energy, int L) :
00014     z1_(pair->GetZ(1)), z2_(pair->GetZ(2)), L_(L), m1_(pair->GetM(1)), m2_(pair->GetM(2)),
00015     energy_(energy) {
00016 }
00017
00018 double EffectiveCharge::Integrand(double x, void* p) {
00019     int* L = (int*)p;
00020     return gsl_sf_bessel_jl(*L, x)/x;
00021 }
00022
00028 double EffectiveCharge::operator()(double r) {
00029     gsl_integration_workspace * w
00030         = gsl_integration_workspace_alloc (1000);
00031     gsl_function F;
00032     F.function = &Integrand;
00033     F.params= &L_;
00034
00035     double intZ1=0.;
00036     double intZ2=0.;
00037     double intError;
00038     gsl_integration_qags (&F, 0., m2_/(m1_+m2_)*energy_/hbarc*r, 0., 1.e-6, 1000, w, &intZ1, &intError);
00039     gsl_integration_qags (&F, 0., m1_/(m1_+m2_)*energy_/hbarc*r, 0., 1.e-6, 1000, w, &intZ2, &intError);
00040
00041     gsl_integration_workspace_free (w);
00042
00043     return sqrt (fstruc*hbarc) * L_ * DoubleFactorial (2*L_+1) /
00044         pow ((energy_/hbarc*r), (double)L_) * (z1_*intZ1+pow (-1, L_) * z2_*intZ2);
00045 }
```

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.](#)

```
00001 #include "EigenFunc.h"
00002 #include <gsl/gsl_eigen.h>
00003 #include <math.h>
00004
00009 EigenFunc::EigenFunc(const matrix_r &A) {
00010     eigenvalues_.clear();
00011     eigenvectors_.clear();
00012
00013     gsl_matrix * m = gsl_matrix_alloc(A.size(), A.size());
00014     for(int i=0; i<A.size(); i++) {
00015         for(int ii=0; ii<A.size(); ii++) {
00016             gsl_matrix_set (m, i, ii, A[i][ii]);
00017         }
00018     }
00019
00020     gsl_vector * eval = gsl_vector_alloc (A.size());
00021     gsl_matrix * evec = gsl_matrix_alloc (A.size(), A.size());
00022     gsl_eigen_symmv_workspace * w = gsl_eigen_symmv_alloc (A.size());
00023 }
```

```

00024     gsl_eigen_symmv (m,eval,evec,w);
00025     gsl_eigen_symmv_sort (eval,evec,GSL_EIGEN_SORT_VAL_ASC);
00026
00027     for(int i=0;i<A.size();i++) {
00028         eigenvalues_.push_back(gsl_vector_get(eval,i));
00029         vector_r tempRow;
00030         eigenvectors_.push_back(tempRow);
00031         for(int ii=0;ii<A.size();ii++) {
00032             eigenvectors_[i].push_back(gsl_matrix_get(evec,i,ii));
00033         }
00034     }
00035
00036     gsl_vector_free (eval);
00037     gsl_matrix_free (evec);
00038     gsl_matrix_free (m);
00039     gsl_eigen_symmv_free (w);
00040
00041 }
00042
00047 EigenFunc::EigenFunc(const matrix_r &A, const std::vector<vector_r > &B) {
00048     eigenvalues_.clear();
00049     eigenvectors_.clear();
00050
00051     gsl_matrix * n = gsl_matrix_alloc(A.size(),A.size());
00052     gsl_matrix * m = gsl_matrix_alloc(B.size(),B.size());
00053     for(int i=0;i<A.size();i++) {
00054         for(int ii=0;ii<A.size();ii++) {
00055             gsl_matrix_set (n,i,ii,A[i][ii]);
00056             gsl_matrix_set (m,i,ii,B[i][ii]);
00057         }
00058     }
00059
00060     gsl_vector * eval = gsl_vector_alloc (A.size());
00061     gsl_matrix * evec = gsl_matrix_alloc (A.size(), A.size());
00062
00063     gsl_eigen_gensymmv_workspace * w = gsl_eigen_gensymmv_alloc (A.size());
00064
00065     gsl_eigen_gensymmv(n,m,eval,evec,w);
00066     gsl_eigen_gensymmv_sort (eval,evec,GSL_EIGEN_SORT_VAL_ASC);
00067
00068     for(int i=0;i<A.size();i++) {
00069         eigenvalues_.push_back(gsl_vector_get(eval,i));
00070         vector_r tempRow;
00071         eigenvectors_.push_back(tempRow);
00072         for(int ii=0;ii<A.size();ii++) {
00073             eigenvectors_[i].push_back(gsl_matrix_get(evec,i,ii));
00074         }
00075     }
00076
00077     vector_r tempNormVec;
00078     for(int i=0;i<A.size();i++) tempNormVec.push_back(0.0);
00079
00080     for(int i=0;i<A.size();i++) {
00081         for(int ii=0;ii<A.size();ii++) {
00082             double sum=0.0;
00083             for(int iii=0;iii<A.size();iii++) {
00084                 sum+=B[ii][iii]*eigenvectors_[iii][i];
00085             }
00086             tempNormVec[ii]=sum;
00087         }
00088         double sum=0.0;
00089         for(int ii=0;ii<A.size();ii++) {
00090             sum+=tempNormVec[ii]*eigenvectors_[ii][i];
00091         }
00092         for(int ii=0;ii<A.size();ii++) {
00093             eigenvectors_[ii][i]=eigenvectors_[ii][i]/sqrt(sum);
00094         }
00095     }
00096
00097     gsl_vector_free (eval);
00098     gsl_matrix_free (evec);
00099     gsl_matrix_free (m);
00100     gsl_matrix_free (n);
00101     gsl_eigen_gensymmv_free (w);
00102 }

```

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.](#)

```

00001 #include "AMatrixFunc.h"
00002 #include "AngCoeff.h"
00003 #include "CNuc.h"
00004 #include "Config.h"
00005 #include "CoulFunc.h"
00006 #include "DataLine.h"
00007 #include "ECIntegral.h"
00008 #include "EData.h"
00009 #include "ESegment.h"
00010 #include "RMatrixFunc.h"
00011 #include "ShftFunc.h"
00012 #include "TargetEffect.h"
00013 #include "IntegratedFermiFunc.h"
00014 #include <iostream>
00015 #include <assert.h>
00016
00023 EPoint::EPoint(DataLine dataLine, ESegment *parent) {
00024     entrance_key_=parent->GetEntranceKey();
00025     exit_key_=parent->GetExitKey();
00026     cm_angle_=dataLine.angle();
00027     lab_angle_=dataLine.angle();
00028     cm_energy_=dataLine.energy();
00029     lab_energy_=dataLine.energy();
00030     excitation_energy_=dataLine.energy();
00031     cm_crosssection_=dataLine.crossSection();
00032     cm_dcrosssection_=dataLine.error();
00033     lab_crosssection_=dataLine.crossSection();
00034     lab_dcrosssection_=dataLine.error();
00035     geofactor_=0.;
00036     fitcrosssection_=0.;
00037     sfactorconv_=0.;
00038     is_differential_=parent->IsDifferential();
00039     is_phase_=parent->IsPhase();
00040     is_ang_dist_=parent->IsAngularDist();
00041     max_ang_dist_order_=parent->GetMaxAngDistOrder();
00042     j_value_=parent->GetJ();
00043     l_value_=parent->GetL();
00044     is_mapped_=false;
00045     targetEffectNum_=0;
00046     parentData_=NULL;
00047     stoppingPower_=0.0;
00048 }
00049
00056 EPoint::EPoint(double angle, double energy, ESegment* parent) {
00057     entrance_key_=parent->GetEntranceKey();
00058     exit_key_=parent->GetExitKey();
00059     lab_angle_=angle;
00060     cm_angle_=angle;
00061     lab_energy_=energy;
00062     cm_energy_=energy;
00063     excitation_energy_=energy;
00064     cm_crosssection_=0.;
00065     cm_dcrosssection_=0.1;
00066     lab_crosssection_=0.;
00067     lab_dcrosssection_=0.1;
00068     geofactor_=0.;
00069     fitcrosssection_=0.;

```

```

00070     sfactorconv_=0.;
00071     is_differential_=parent->IsDifferential();
00072     is_phase_=parent->IsPhase();
00073     is_ang_dist_=parent->IsAngularDist();
00074     max_ang_dist_order_=parent->GetMaxAngDistOrder();
00075     j_value_=parent->GetJ();
00076     l_value_=parent->GetL();
00077     is_mapped_=false;
00078     targetEffectNum_=0;
00079     parentData_=NULL;
00080     stoppingPower_=0.0;
00081 }
00082
00090 EPoint::EPoint(double angle, double energy, int entranceKey,
00091               int exitKey, bool isDifferential, bool isPhase, bool isAngularDist, double jValue, int
00092               lValue, int maxAngDistOrder) {
00093     entrance_key_=entranceKey;
00094     exit_key_=exitKey;
00095     lab_angle_=angle;
00096     cm_angle_=angle;
00097     lab_energy_=energy;
00098     cm_energy_=energy;
00099     excitation_energy_=energy;
00100     cm_crosssection_=0.;
00101     cm_dcrossection_=0.1;
00102     lab_crosssection_=0.;
00103     lab_dcrossection_=0.1;
00104     geofactor_=0.;
00105     fitcrosssection_=0.;
00106     sfactorconv_=0.;
00107     is_differential_=isDifferential;
00108     is_phase_=isPhase;
00109     is_ang_dist_=isAngularDist;
00110     max_ang_dist_order_=maxAngDistOrder;
00111     j_value_=jValue;
00112     l_value_=lValue;
00113     is_mapped_=false;
00114     targetEffectNum_=0;
00115     parentData_=NULL;
00116     stoppingPower_=0.0;
00117 }
00122 bool EPoint::IsDifferential() const {
00123     return is_differential_;
00124 }
00125
00130 bool EPoint::IsPhase() const {
00131     return is_phase_;
00132 }
00133
00138 bool EPoint::IsAngularDist() const {
00139     return is_ang_dist_;
00140 }
00141
00149 bool EPoint::IsMapped() const {
00150     return is_mapped_;
00151 }
00152
00157 bool EPoint::IsTargetEffect() const {
00158     if(GetTargetEffectNum()!=0) return true;
00159     else return false;
00160 }
00161
00168 int EPoint::GetEntranceKey() const {
00169     return entrance_key_;
00170 }
00171
00178 int EPoint::GetExitKey() const {
00179     return exit_key_;
00180 }
00181
00186 int EPoint::GetMaxLOrder() const {
00187     return legendreP_.size()-1;
00188 }
00189
00195 int EPoint::GetL() const {
00196     return l_value_;
00197 }
00198
00203 int EPoint::NumLocalMappedPoints() const {
00204     return local_mapped_points_.size();
00205 }
00206
00211 int EPoint::NumSubPoints() const {
00212     return integrationPoints_.size();
00213 }
00214

```

```

00219 int EPoint::GetTargetEffectNum() const {
00220     return targetEffectNum_;
00221 }
00222
00227 int EPoint::GetMaxAngDistOrder() const {
00228     return max_ang_dist_order_;
00229 }
00230
00235 int EPoint::GetNumAngularDists() const {
00236     return angularDists_.size();
00237 }
00238
00243 double EPoint::GetLabAngle() const {
00244     return lab_angle_;
00245 }
00246
00251 double EPoint::GetCMAngle() const {
00252     return cm_angle_;
00253 }
00254
00259 double EPoint::GetLabEnergy() const {
00260     return lab_energy_;
00261 }
00262
00267 double EPoint::GetCMEnergy() const {
00268     return cm_energy_;
00269 }
00270
00275 double EPoint::GetExcitationEnergy() const {
00276     return excitation_energy_;
00277 }
00278
00279
00284 double EPoint::GetLegendreP(int lOrder) const {
00285     return legendreP_[lOrder];
00286 }
00287
00292 double EPoint::GetLabCrossSection() const {
00293     return lab_crosssection_;
00294 }
00295
00300 double EPoint::GetCMCrossSection() const {
00301     return cm_crosssection_;
00302 }
00303
00308 double EPoint::GetLabCrossSectionError() const {
00309     return lab_dcrosssection_;
00310 }
00311
00316 double EPoint::GetCMCrossSectionError() const {
00317     return cm_dcrosssection_;
00318 }
00319
00324 double EPoint::GetGeometricalFactor() const {
00325     return geofactor_;
00326 }
00327
00332 double EPoint::GetFitCrossSection() const {
00333     return fitcrosssection_;
00334 }
00335
00340 double EPoint::GetSFactorConversion() const {
00341     return sfactorconv_;
00342 }
00343
00349 double EPoint::GetSqrtPenetrability(int jGroupNum, int channelNum) const {
00350     return penetrabilities_[jGroupNum-1][channelNum-1];
00351 }
00352
00358 double EPoint::GetJ() const {
00359     return j_value_;
00360 }
00361
00366 double EPoint::GetStoppingPower() const {
00367     return stoppingPower_;
00368 }
00369
00374 double EPoint::GetTargetThickness() const {
00375     return targetThickness_;
00376 }
00377
00382 double EPoint::GetAngularDist(int order) const {
00383     return angularDists_[order];
00384 }
00385
00391 complex EPoint::GetLoElement(int jGroupNum, int channelNum) const {
00392     return lo_elements_[jGroupNum-1][channelNum-1];

```

```

00393 }
00394
00401 complex EPoint::GetExpCoulombPhase(int jGroupNum, int channelNum) const {
00402     return coulombphase_[jGroupNum-1][channelNum-1];
00403 }
00404
00411 complex EPoint::GetExpHardSpherePhase(int jGroupNum, int channelNum) const {
00412     return hardspherephase_[jGroupNum-1][channelNum-1];
00413 }
00414
00419 complex EPoint::GetCoulombAmplitude() const {
00420     return coulombamplitude_;
00421 }
00422
00428 complex EPoint::GetECAmplitude(int kGroupNum, int ecMGroupNum) const {
00429     return ec_amplitudes_[kGroupNum-1][ecMGroupNum-1];
00430 }
00431
00436 EnergyMap EPoint::GetMap() const {
00437     EnergyMap thisMap;
00438     if(this->IsMapped()) {
00439         thisMap.segment=energy_map_.segment;
00440         thisMap.point=energy_map_.point;
00441     }
00442     else {
00443         thisMap.segment=0;
00444         thisMap.point=0;
00445     }
00446     return thisMap;
00447 }
00448
00454 void EPoint::Initialize(CNuc *compound, const Config &configure) {
00455     this->CalcEDependentValues(compound, configure);
00456     if(this->IsDifferential())
00457         this->CalcLegendreP(configure.maxLOrder, NULL);
00458     this->CalcCoulombAmplitude(compound);
00459     if(configure.paramMask & Config::USE_EXTERNAL_CAPTURE)
00460         this->CalculateECAmplitudes(compound, configure);
00461 }
00462
00468 void EPoint::ConvertLabEnergy(PPair *pPair) {
00469     cm_energy_=this->GetLabEnergy()*
00470         (pPair->GetM(2))/
00471         (pPair->GetM(1)+pPair->GetM(2));
00472     excitation_energy_=cm_energy_+pPair->GetSepE();
00473 }
00474
00482 void EPoint::ConvertDecayEnergy(PPair *pPair) {
00483     cm_energy_=this->GetLabEnergy()/
00484         (pPair->GetM(2))*
00485         (pPair->GetM(1)+pPair->GetM(2));
00486     excitation_energy_=cm_energy_+pPair->GetSepE();
00487 }
00488
00496 void EPoint::ConvertLabAngle(PPair *pPair) {
00497     cm_angle_=this->GetLabAngle()+180./pi*asin(pPair->GetM(1)/pPair->GetM(2)*sin(pi/180.*this->GetLabAngle()));
00498 }
00499
00507 void EPoint::ConvertLabAngle(PPair *entrancePair, PPair *exitPair, const Config& configure) {
00508     double qValue=entrancePair->GetSepE()+entrancePair->GetExE()-exitPair->GetSepE()-exitPair->GetExE();
00509     double
00510         a13=(entrancePair->GetM(1)*exitPair->GetM(1))*this->GetLabEnergy()/(this->GetLabEnergy()+qValue)/
00511         (entrancePair->GetM(1)+entrancePair->GetM(2))/(exitPair->GetM(1)+exitPair->GetM(2));
00512     double
00513         a24=(entrancePair->GetM(2)*exitPair->GetM(2))*(1+entrancePair->GetM(1)/entrancePair->GetM(2)*qValue/(this->GetLabEnergy()
00514         +entrancePair->GetM(1)+entrancePair->GetM(2))/(exitPair->GetM(1)+exitPair->GetM(2)));
00515     if(a13>a24) {
00516         double thetaMax=asin(sqrt(a24/a13))*180./pi;
00517         if(thetaMax<this->GetLabAngle()) configure.outStream << std::endl << "Lab Angle (" <<
00518             this->GetLabAngle()
00519             << " degrees) is not kinematically possible. Maximum angle is "
00520             << thetaMax << " degrees." << std::endl;
00521         assert(thetaMax>=this->GetLabAngle());
00522     }
00523     double
00524         E3PerEt=a13*pow(cos(this->GetLabAngle()*pi/180.)+sqrt(a24/a13-pow(sin(this->GetLabAngle()*pi/180.),2.0)),2.0);
00525     double tempE3PerEt =
00526         a13*pow(cos((this->GetLabAngle()+0.001)*pi/180.)+sqrt(a24/a13-pow(sin((this->GetLabAngle()+0.001)*pi/180.),2.0)),2.0);
00527     double slope =
00528         sqrt(tempE3PerEt/a24)*sin((this->GetLabAngle()+0.001)*pi/180.)-sqrt(E3PerEt/a24)*sin(this->GetLabAngle()*pi/180.);
00529     bool switchDomain=false;
00530     if(slope<0.) switchDomain=true;
00531 }
00532
00536 cm_angle_=180./pi*asin(sqrt(E3PerEt/a24)*sin(this->GetLabAngle()*pi/180.));

```

```

00529     if(switchDomain) cm_angle_ = 180.- cm_angle_;
00530 }
00531
00538 void EPoint::ConvertCrossSection(PPair *entrancePair, PPair *exitPair) {
00539     double conversionFactor;
00540     if(this->GetLabAngle()==0.0 || this->GetLabAngle()==180.0) {
00541         double m1=entrancePair->GetM(1);
00542         double m2=entrancePair->GetM(2);
00543         double m3=exitPair->GetM(1);
00544         double m4=exitPair->GetM(2);
00545         double e1=this->GetLabEnergy();
00546         double
qValue=entrancePair->GetSepE()+entrancePair->GetExE()-exitPair->GetSepE()-exitPair->GetExE();
00547         double et=e1+qValue;
00548         double a=m1*m4*e1/(m1+m2)/(m3+m4)/et;
00549         double b=m1*m3*e1/(m1+m2)/(m3+m4)/et;
00550         double c=m2*m3/(m1+m2)/(m3+m4) * (1+m1*qValue/m2/et);
00551         double d=m2*m4/(m1+m2)/(m3+m4) * (1+m1*qValue/m2/et);
00552         double e3et=b+d+2*pow(a*c,0.5)*cos(pi/180.*this->GetCMAngle());
00553         conversionFactor=pow(a*c,0.5)*pow(d/b-pow(sin(this->GetLabAngle()*pi/180.),2.0),0.5)/e3et;
00554     }
00555     else {
00556         conversionFactor=pow(sin(pi/180.*this->GetLabAngle())/sin(pi/180.*this->GetCMAngle()),2.0)*cos(pi/180.*(this->GetCMAngle()
00557     )
00558     cm_crosssection_=this->GetLabCrossSection()*conversionFactor;
00559     cm_dcrosssection_=this->GetLabCrossSectionError()*conversionFactor;
00560 }
00561
00566 void EPoint::AddLegendreP(double polynomial) {
00567     legendreP_.push_back(polynomial);
00568 }
00569
00574 void EPoint::SetGeometricalFactor(double geoFactor) {
00575     geofactor_=geoFactor;
00576 }
00577
00582 void EPoint::SetFitCrossSection(double crossSection) {
00583     fitcrosssection_=crossSection;
00584 }
00585
00590 void EPoint::SetSFactorConversion(double conversion) {
00591     sfactorconv_=conversion;
00592 }
00593
00598 void EPoint::SetExitKey(int key) {
00599     exit_key_=key;
00600 }
00601
00606 void EPoint::CalcLegendreP(int maxL,TargetEffect* targetEffect) {
00607     double x=cos(this->GetCMAngle()*pi/180.0);
00608     if(maxL>=0) {
00609         if(targetEffect && targetEffect->NumQCoefficients()>0)
00610             this->AddLegendreP(targetEffect->GetQCoefficient(0));
00611         else this->AddLegendreP(1.0);
00612         double polyMinusTwo=1.0;
00613         if(maxL>=1) {
00614             if(targetEffect && targetEffect->NumQCoefficients()>1)
00615                 this->AddLegendreP(x*targetEffect->GetQCoefficient(1));
00616             else this->AddLegendreP(x);
00617             double polyMinusOne=x;
00618             if(maxL>=2) {
00619                 for(int lOrder=2;lOrder<=maxL;lOrder++) {
00620                     double poly=(2.0*lOrder-1.0)/lOrder*x*polyMinusOne-
(lOrder-1.0)/lOrder*polyMinusTwo;
00621                     if(targetEffect && targetEffect->NumQCoefficients()>lOrder)
00622                         this->AddLegendreP(poly*targetEffect->GetQCoefficient(lOrder));
00623                     else this->AddLegendreP(poly);
00624                     polyMinusTwo=polyMinusOne;
00625                     polyMinusOne=poly;
00626                 }
00627             }
00628         }
00629     }
00630 }
00631 for(int i=1;i<=this->NumSubPoints();i++) {
00632     this->GetSubPoint(i)->CalcLegendreP(maxL, targetEffect);
00633 }
00634 }
00635
00643 void EPoint::CalcEDependentValues(CNuc *theCNuc, const Config& configure) {
00644     PPair *entrancePair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetEntranceKey()));
00645     PPair *exitPair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetExitKey()));
00646
00647     double inEnergy;
00648     double geofactor;
00649     double sfactorconv;
00650     if(theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetEntranceKey()))->GetPType()==20) {

```



```

00651     inEnergy=this->GetCMEnergy()+exitPair->GetSepE()+exitPair->GetExE();
00652     geofactor=1.;
00653     sfactorconv=1.;
00654 } else {
00655     inEnergy=this->GetCMEnergy()+entrancePair->GetSepE()+entrancePair->GetExE();
00656     geofactor=pi*pow(hbarc,2.)/(2*entrancePair->GetRedMass()*uconv*this->GetCMEnergy());
00657     sfactorconv=this->GetCMEnergy()*exp(2*pi*sqrt(uconv/2.)*fstruc*entrancePair->GetZ(1)*
00658         entrancePair->GetZ(2)*sqrt(entrancePair->GetRedMass())
00659         /(this->GetCMEnergy()));
00660 }
00661 this->SetGeometricalFactor(geofactor);
00662 this->SetSFactorConversion(sfactorconv);
00663
00664 for(int j=1;j<=theCNuc->NumJGroups();j++) {
00665     if(theCNuc->GetJGroup(j)->IsInRMatrix()) {
00666         JGroup *theJGroup=theCNuc->GetJGroup(j);
00667         for(int ch=1;ch<=theJGroup->NumChannels();ch++) {
00668             AChannel *theChannel=theJGroup->GetChannel(ch);
00669             PPair *thePair=theCNuc->GetPair(theChannel->GetPairNum());
00670             int lValue=theChannel->GetL();
00671             double localEnergy=inEnergy-thePair->GetSepE()-thePair->GetExE();
00672             if(thePair->GetPType()==0) {
00673                 if(localEnergy<0.0) {
00674                     ShiftFunc theShiftFunction(thePair);
00675                     double localShift=theShiftFunction(lValue,inEnergy);
00676                     double boundary=theChannel->GetBoundaryCondition();
00677                     complex loElement(localShift-boundary,0.0);
00678                     this->AddLoElement(j,ch,loElement);
00679                     this->AddSqrtPenetrability(j,ch,0.0);
00680                     this->AddExpCoulombPhase(j,ch,1.0);
00681                     this->AddExpHardSpherePhase(j,ch,1.0);
00682                 } else {
00683                     CoulFunc theCoulombFunction(thePair,!!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
00684                     double radius=thePair->GetChRad();
00685                     double localPene=theCoulombFunction.Penetrability(lValue,radius,localEnergy);
00686                     double localShift=theCoulombFunction.PEShift(lValue,radius,localEnergy);
00687                     double boundary=theChannel->GetBoundaryCondition();
00688                     complex loElement(localShift-boundary,localPene);
00689                     double redmass=thePair->GetRedMass();
00690                     double eta=sqrt(uconv/2.)*fstruc*thePair->GetZ(1)*thePair->GetZ(2)*
00691                         sqrt(redmass/localEnergy);
00692                     complex expCP(1.0,0.0);
00693                     for(int ll=1;ll<=theChannel->GetL();ll++)
00694                         expCP*=complex((double)ll/sqrt(pow(eta,2.0)+pow((double)ll,2.0)),
00695                             eta/sqrt(pow(eta,2.0)+pow((double)ll,2.0)));
00696                     struct CoulWaves
00697                     {
00698                         coul=theCoulombFunction(lValue,radius,localEnergy);
00699                         complex expHSP(coul.G/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)),
00700                             -coul.F/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)));
00701                     };
00702                     this->AddLoElement(j,ch,loElement);
00703                     this->AddSqrtPenetrability(j,ch,sqrt(localPene));
00704                     this->AddExpCoulombPhase(j,ch,expCP);
00705                     this->AddExpHardSpherePhase(j,ch,expHSP);
00706                 }
00707             } else if(thePair->GetPType()==10) {
00708                 complex loElement = complex(0.0,0.0);
00709                 this->AddLoElement(j,ch,loElement);
00710                 double sqrtPene = (configure.paramMask & Config::USE_RMC_FORMALISM) ? 1. :
00711                     pow(localEnergy/hbarc, (double) lValue+0.5);
00712                 this->AddSqrtPenetrability(j,ch,sqrtPene);
00713                 this->AddExpCoulombPhase(j,ch,1.0);
00714                 this->AddExpHardSpherePhase(j,ch,1.0);
00715             } else if(thePair->GetPType()==20) {
00716                 complex loElement = complex(0.0,0.0);
00717                 this->AddLoElement(j,ch,loElement);
00718                 IntegratedFermiFunc fermiFunc(thePair->GetZ(1));
00719                 double endPointE = thePair->GetSepE()-inEnergy;
00720                 double sqrtPene = (1.+endPointE/0.510998903<=1.) ? 0. :
00721                     sqrt(fermiFunc(1.+endPointE/0.510998903,exitPair->GetZ(1)+exitPair->GetZ(2),thePair->GetChRad()));
00722                 this->AddSqrtPenetrability(j,ch,sqrtPene);
00723                 this->AddExpCoulombPhase(j,ch,1.0);
00724                 this->AddExpHardSpherePhase(j,ch,1.0);
00725             }
00726         }
00727     }
00728 }
00729 for(int i=1;i<=this->NumSubPoints();i++) {
00730     this->GetSubPoint(i)->CalcEDependentValues(theCNuc,configure);
00731 }
00732 for(int i=1;i<=this->NumLocalMappedPoints();i++) {
00733     EPoint *mappedPoint=this->GetLocalMappedPoint(i);
00734     mappedPoint->geofactor_=geofactor_;
00735     mappedPoint->sfactorconv_=sfactorconv_;
00736     mappedPoint->lo_elements_=lo_elements_;
00737     mappedPoint->penetrabilities_=penetrabilities_;
00738     mappedPoint->coulombphase_=coulombphase_;
00739     mappedPoint->hardspherephase_=hardspherephase_;

```

```

00736     for(int ii=1;ii<=this->NumSubPoints();ii++) {
00737         EPoint *subMappedPoint=mappedPoint->GetSubPoint(ii);
00738         subMappedPoint->geofactor_=this->GetSubPoint(ii)->geofactor_;
00739         subMappedPoint->sfactorconv_=this->GetSubPoint(ii)->sfactorconv_;
00740         subMappedPoint->lo_elements_=this->GetSubPoint(ii)->lo_elements_;
00741         subMappedPoint->penetrabilities_=this->GetSubPoint(ii)->penetrabilities_;
00742         subMappedPoint->coulombphase_=this->GetSubPoint(ii)->coulombphase_;
00743         subMappedPoint->hardspherephase_=this->GetSubPoint(ii)->hardspherephase_;
00744     }
00745 }
00746 }
00747
00753 void EPoint::AddLoElement(int jGroupNum, int channelNum, complex loElement) {
00754     vector_c d;
00755     while(jGroupNum>lo_elements_.size()) lo_elements_.push_back(d);
00756     lo_elements_[jGroupNum-1].push_back(loElement);
00757     assert(channelNum==lo_elements_[jGroupNum-1].size());
00758 }
00759
00765 void EPoint::AddSqrtPenetrability(int jGroupNum, int channelNum, double sqrtPene) {
00766     vector_r d;
00767     while(jGroupNum>penetrabilities_.size()) penetrabilities_.push_back(d);
00768     penetrabilities_[jGroupNum-1].push_back(sqrtPene);
00769     assert(channelNum==penetrabilities_[jGroupNum-1].size());
00770 }
00771
00777 void EPoint::AddExpCoulombPhase(int jGroupNum, int channelNum, complex expShift) {
00778     vector_c d;
00779     while(jGroupNum>coulombphase_.size()) coulombphase_.push_back(d);
00780     coulombphase_[jGroupNum-1].push_back(expShift);
00781     assert(channelNum==coulombphase_[jGroupNum-1].size());
00782 }
00783
00789 void EPoint::AddExpHardSpherePhase(int jGroupNum, int channelNum, complex expShift) {
00790     vector_c d;
00791     while(jGroupNum>hardspherephase_.size()) hardspherephase_.push_back(d);
00792     hardspherephase_[jGroupNum-1].push_back(expShift);
00793     assert(channelNum==hardspherephase_[jGroupNum-1].size());
00794 }
00795
00800 void EPoint::CalcCoulombAmplitude(CNuc *theCNuc) {
00801     if(this->GetEntranceKey()==this->GetExitKey()) {
00802         PPair *entrancePair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetEntranceKey()));
00803         int z1=entrancePair->GetZ(1);
00804         int z2=entrancePair->GetZ(2);
00805         double redmass=entrancePair->GetRedMass();
00806         double energy=this->GetCMEnergy();
00807         double angle=this->GetCMAngle();
00808         double eta=sqrt(uconv/2.)*fstruc*z1*z2*
00809             sqrt(redmass/energy);
00810         double cal=(1.0/(2.0*sqrt(pi)))*eta*(1.0/pow(sin(angle*pi/360.0),2.));
00811         double cex=2.0*eta*log(sin(angle*pi/360.0));
00812         complex calpha(cal*cos(cex),-cal*sin(cex));
00813         this->SetCoulombAmplitude(calpha);
00814     } else this->SetCoulombAmplitude(complex(0.,0.));
00815     for(int i=1;i<=this->NumSubPoints();i++) {
00816         this->GetSubPoint(i)->CalcCoulombAmplitude(theCNuc);
00817     }
00818 }
00819
00824 void EPoint::SetCoulombAmplitude(complex amplitude) {
00825     coulombamplitude_=amplitude;
00826 }
00827
00834 void EPoint::CalculateECAmplitudes(CNuc *theCNuc, const Config& configure) {
00835     int aa=theCNuc->GetPairNumFromKey(this->GetEntranceKey());
00836     if(theCNuc->GetPair(aa)->GetPType()==20) return;
00837     if(theCNuc->GetPair(aa)->IsEntrance()) {
00838         PPair *entrancePair=theCNuc->GetPair(aa);
00839         for(int j=1;j<=theCNuc->NumJGroups();j++) {
00840             for(int la=1;la<=theCNuc->GetJGroup(j)->NumLevels();la++) {
00841                 if(theCNuc->GetJGroup(j)->GetLevel(la)->IsECLevel()) {
00842                     ALevel *ecLevel = theCNuc->GetJGroup(j)->GetLevel(la);
00843                     int ir=theCNuc->GetPairNumFromKey(this->GetExitKey());
00844                     if(ecLevel->GetECPairNum()==ir) {
00845                         double inEnergy=this->GetCMEnergy()+entrancePair->GetSepE()+entrancePair->GetExE();
00846                         for(int k=1;k<=entrancePair->GetDecay(ir)->NumKGroups();k++) {
00847                             KGroup *theKGroup=entrancePair->GetDecay(ir)->GetKGroup(k);
00848                             for(int ecm=1;ecm<=theKGroup->NumECMGroups();ecm++) {
00849                                 ECMGroup *theECMGroup=theKGroup->GetECMGroup(ecm);
00850                                 //entrance Phase Calculations;
00851                                 CoulFunc
theCoulombFunction(entrancePair,!!(configure.paramMask&Config::USE_GSL_COULOMB_FUNC));
00852                                 struct CoulWaves
00853                                     coul=theCoulombFunction(theECMGroup->GetL(),entrancePair->GetChRad(),
00854                                         this->GetCMEnergy());
00855                                 double eta=sqrt(uconv/2.)*fstruc*entrancePair->GetZ(1)*entrancePair->GetZ(2)*

```

```

00856         sqrt(entrancePair->GetRedMass()/this->GetCMEnergy());
00857     complex expCP(1.0,0.0);
00858     for(int ll=1;ll<=theECMGroup->GetL();ll++)
00859         expCP*=complex((double)ll/sqrt(pow((double)ll,2.0)+pow(eta,2.0)),
00860             eta/sqrt(pow((double)ll,2.0)+pow(eta,2.0)));
00861     complex expHSP(coul.G/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)),
00862         -coul.F/sqrt(pow(coul.F,2.0)+pow(coul.G,2.0)));
00863
00864     double levelEnergy=ecLevel->GetE();
00865     double sqrtGammaPene=pow((inEnergy-levelEnergy)/hbarc,theECMGroup->GetMult()+0.5);
00866
00867     //Initialize variables
00868     AChannel *theFinalChannel = theCNuc->GetJGroup(j)->GetChannel(theECMGroup->GetFinalChannel());
00869     PPair *theFinalPair=theCNuc->GetPair(theFinalChannel->GetPairNum());
00870     int theInitialLValue;
00871     double theInitialSValue;
00872     if(theECMGroup->IsChannelCapture()) {
00873         MGroup *theChanCapMGroup=entrancePair->GetDecay(theECMGroup->GetChanCapDecay())->
00874             GetKGroup(theECMGroup->GetChanCapKGroup())->GetMGroup(theECMGroup->GetChanCapMGroup());
00875         theInitialLValue=theCNuc->GetJGroup(theChanCapMGroup->GetJNum())->
00876             GetChannel(theChanCapMGroup->GetChpNum())->GetL();
00877         theInitialSValue=theCNuc->GetJGroup(theChanCapMGroup->GetJNum())->
00878             GetChannel(theChanCapMGroup->GetChpNum())->GetS();
00879     } else {
00880         theInitialLValue=theECMGroup->GetL();
00881         theInitialSValue=theKGroup->GetS();
00882     }
00883
00884     ECIntegral theECIntegral(theFinalPair,configure);
00885     complex integrals = theECIntegral(theInitialLValue, theFinalChannel->GetL(),
00886         theInitialSValue, theFinalChannel->GetS(),
00887         theECMGroup->GetJ(), theCNuc->GetJGroup(j)->GetJ(),
00888         theECMGroup->GetMult(), theECMGroup->GetRadType(),
00889         inEnergy, levelEnergy,
00890         theECMGroup->IsChannelCapture());
00891
00892     //calculate the total radial integral
00893     complex ecAmplitude=expCP*expHSP*sqrtGammaPene*integrals;
00894     this->AddECAmplitude(k,ecm,ecAmplitude);
00895     }
00896 }
00897 }
00898 }
00899 }
00900 }
00901 }
00902 for(int i=1;i<=this->NumSubPoints();i++) {
00903     this->GetSubPoint(i)->CalculateECAmplitudes(theCNuc,configure);
00904 }
00905 }
00906
00911 void EPoint::AddECAmplitude(int kGroupNum, int ecMGroupNum, complex ecAmplitude) {
00912     vector_c d;
00913     while(kGroupNum>ec_amplitudes_.size()) ec_amplitudes_.push_back(d);
00914     ec_amplitudes_[kGroupNum-1].push_back(ecAmplitude);
00915     assert(ecMGroupNum==ec_amplitudes_[kGroupNum-1].size());
00916 }
00917
00922 void EPoint::Calculate(CNuc* theCNuc,const Config &configure, EPoint *parent, int subPointNum) {
00923
00924     if(!this->IsTargetEffect()) {
00925         (!this->GetParentData()->GetTargetEffect(this->GetTargetEffectNum())->IsConvolution())&&
00926         !this->GetParentData()->GetTargetEffect(this->GetTargetEffectNum())->IsTargetIntegration()) {
00927             GenMatrixFunc *theMatrixFunc;
00928             if(configure.paramMask & Config::USE_AMATRIX) theMatrixFunc=new AMatrixFunc(theCNuc,configure);
00929             else theMatrixFunc=new RMatrixFunc(theCNuc,configure);
00930             theMatrixFunc->ClearMatrices();
00931             theMatrixFunc->FillMatrices(this);
00932             theMatrixFunc->InvertMatrices();
00933             theMatrixFunc->CalculateTMatrix(this);
00934             theMatrixFunc->CalculateCrossSection(this);
00935             if(subPointNum&&parent) {
00936                 for(int i=1;i<=parent->NumLocalMappedPoints();i++) {
00937                     EPoint *mappedSubPoint = parent->GetLocalMappedPoint(i)->
00938                         GetSubPoint(subPointNum);
00939                     theMatrixFunc->CalculateCrossSection(mappedSubPoint);
00940                 }
00941             } else {
00942                 for(int i=1;i<=this->NumLocalMappedPoints();i++) {
00943                     EPoint *mappedPoint = this->GetLocalMappedPoint(i);
00944                     theMatrixFunc->CalculateCrossSection(mappedPoint);
00945                 }
00946             }
00947             delete theMatrixFunc;
00948         } else {
00949             for(int i = 1; i<=this->NumSubPoints();i++) {
00950                 EPoint *subPoint=this->GetSubPoint(i);

```

```

00951         if(this->NumLocalMappedPoints()>0)
00952             subPoint->Calculate(theCNuc,configure,this,i);
00953         else subPoint->Calculate(theCNuc,configure);
00954     }
00955     this->IntegrateTargetEffect();
00956     for(int i=1;i<=this->NumLocalMappedPoints();i++)
00957         this->GetLocalMappedPoint(i)->IntegrateTargetEffect();
00958 }
00959 }
00960
00965 void EPoint::SetMap(int segmentNum, int pointNum) {
00966     is_mapped_=true;
00967     energy_map_.segment=segmentNum;
00968     energy_map_.point=pointNum;
00969 }
00970
00975 void EPoint::AddLocalMappedPoint(EPoint *point) {
00976     local_mapped_points_.push_back(point);
00977 }
00978
00983 void EPoint::ClearLocalMappedPoints() {
00984     local_mapped_points_.clear();
00985 }
00986
00991 void EPoint::SetTargetEffectNum(int targetEffectNum) {
00992     targetEffectNum_=targetEffectNum;
00993 }
00994
00999 void EPoint::AddSubPoint(EPoint subPoint) {
10000     integrationPoints_.push_back(subPoint);
10001 }
10002
10009 void EPoint::IntegrateTargetEffect() {
10010     double yield=0.0;
10011     TargetEffect *targetEffect=this->GetParentData()->GetTargetEffect(this->GetTargetEffectNum());
10012     double energyStep=this->GetSubPoint(1)->GetCMEnergy()-this->GetSubPoint(2)->GetCMEnergy();
10013     if(targetEffect->IsConvolution()&&targetEffect->IsTargetIntegration()) {
10014         int outerLowerLimit=round((this->GetSubPoint(1)->GetCMEnergy()-this->GetCMEnergy())/energyStep)+1;
10015         int outerUpperLimit=outerLowerLimit-1+round(this->GetTargetThickness()/energyStep);
10016         double outerIntFirst=0.0;
10017         double outerIntEvenSum=0.0;
10018         double outerIntOddSum=0.0;
10019         double outerIntegral=0.0;
10020         int outerCounter=0;
10021         for(int i=outerLowerLimit;i<=outerUpperLimit;i++) {
10022             int innerLowerLimit;
10023             if(i-round(targetEffect->convolutionRange*targetEffect->GetSigma()/energyStep)>1)
10024                 innerLowerLimit=i-round(targetEffect->convolutionRange*targetEffect->GetSigma()/energyStep);
10025             else innerLowerLimit=1;
10026             int innerUpperLimit;
10027             if(i+round(targetEffect->convolutionRange*targetEffect->GetSigma()/energyStep)-1<targetEffect->NumSubPoints())
10028                 innerUpperLimit=i+round(targetEffect->convolutionRange*targetEffect->GetSigma()/energyStep)-1;
10029             else innerUpperLimit=targetEffect->NumSubPoints();
10030             double innerIntFirst=0.0;
10031             double innerIntEvenSum=0.0;
10032             double innerIntOddSum=0.0;
10033             double innerIntegral=0.0;
10034             double centroid=this->GetSubPoint(i)->GetCMEnergy();
10035             int innerCounter=0;
10036             for(int ii=innerLowerLimit;ii<=innerUpperLimit;ii++) {
10037                 double thisEnergy=this->GetSubPoint(ii)->GetCMEnergy();
10038                 double innerIntegrand=this->GetSubPoint(ii)->GetFitCrossSection()/
10039                 this->GetSubPoint(ii)->GetStoppingPower()/1e24*targetEffect->GetConvolutionFactor(thisEnergy,centroid);
10040                 if(innerCounter==0) innerIntFirst=innerIntegrand;
10041                 else if(innerCounter%2==0) {
10042                     innerIntEvenSum+=innerIntegrand;
10043                     if(innerCounter>=2)
10044                         innerIntegral=energyStep/3.0*(innerIntFirst+4.0*innerIntOddSum+2.0*innerIntEvenSum-innerIntegrand);
10045                     } else if(innerCounter%2!=0) {
10046                         innerIntOddSum+=innerIntegrand;
10047                         if(innerCounter>=2)
10048                             innerIntegral=energyStep/3.0*(innerIntFirst+4.0*innerIntOddSum+2.0*innerIntEvenSum-3.0*innerIntegrand);
10049                     }
10050                     innerCounter++;
10051                 }
10052                 if(outerCounter==0) outerIntFirst=innerIntegral;
10053                 else if(outerCounter%2==0) {
10054                     outerIntEvenSum+=innerIntegral;
10055                     if(outerCounter>=2) outerIntegral=energyStep/3.0*
10056                     (outerIntFirst+4.0*outerIntOddSum+2.0*outerIntEvenSum-innerIntegral);
10057                     } else if(outerCounter%2!=0) {
10058                         outerIntOddSum+=innerIntegral;
10059                         if(outerCounter>=2) outerIntegral=energyStep/3.0*
10060                         (outerIntFirst+4.0*outerIntOddSum+2.0*outerIntEvenSum-3.0*innerIntegral);
10061                     }
10062                 }
10063             }
10064         }
10065     }
10066 }

```

```

01060         outerCounter++;
01061     }
01062     yield=outerIntegral;
01063 } else if(targetEffect->IsConvolution()) {
01064     double intFirst=0.0;
01065     double intEvenSum=0.0;
01066     double intOddSum=0.0;
01067     double integral=0.0;
01068     double centroid=this->GetCMEnergy();
01069     for(int i=0;i<this->NumSubPoints();i++) {
01070         double thisEnergy=this->GetSubPoint(i+1)->GetCMEnergy();
01071         double integrand=this->GetSubPoint(i+1)->GetFitCrossSection()
01072 *targetEffect->GetConvolutionFactor(thisEnergy,centroid);
01073         if(i==0) intFirst=integrand;
01074         else if(i%2==0) {
01075             intEvenSum+=integrand;
01076             if(i>=2) integral=energyStep/3.0*(intFirst+4.0*intOddSum+2.0*intEvenSum-integrand);
01077         } else if(i%2!=0) {
01078             intOddSum+=integrand;
01079             if(i>=2) integral=energyStep/3.0*(intFirst+4.0*intOddSum+2.0*intEvenSum-3.0*integrand);
01080         }
01081     }
01082     yield=integral;
01083 } else if(targetEffect->IsTargetIntegration()) {
01084     double intFirst=0.0;
01085     double intEvenSum=0.0;
01086     double intOddSum=0.0;
01087     double integral=0.0;
01088     for(int i=0;i<this->NumSubPoints();i++) {
01089         double thisEnergy=this->GetSubPoint(i+1)->GetCMEnergy();
01090         double integrand=this->GetSubPoint(i+1)->GetFitCrossSection()/
01091 this->GetSubPoint(i+1)->GetStoppingPower()/1e24;
01092         if(i==0) intFirst=integrand;
01093         else if(i%2==0) {
01094             intEvenSum+=integrand;
01095             if(i>=2) integral=energyStep/3.0*(intFirst+4.0*intOddSum+2.0*intEvenSum-integrand);
01096         } else if(i%2!=0) {
01097             intOddSum+=integrand;
01098             if(i>=2) integral=energyStep/3.0*(intFirst+4.0*intOddSum+2.0*intEvenSum-3.0*integrand);
01099         }
01100     }
01101     yield=integral;
01102 }
01103 this->SetFitCrossSection(yield);
01104 }
01105
01110 void EPoint::SetParentData(EData* parentData) {
01111     parentData_=parentData;
01112 }
01113
01119 void EPoint::SetStoppingPower(double stoppingPower) {
01120     stoppingPower_=stoppingPower;
01121 }
01122
01127 void EPoint::SetTargetThickness(double targetThickness) {
01128     targetThickness_=targetThickness;
01129 }
01130
01135 void EPoint::SetAngularDists(vector_r dists) {
01136     angularDists_.clear();
01137     angularDists_=dists;
01138 }
01139
01144 EData *EPoint::GetParentData() const {
01145     return parentData_;
01146 }
01147
01152 EPoint* EPoint::GetLocalMappedPoint(int mappedPointNum) const {
01153     return local_mapped_points_[mappedPointNum-1];
01154 }
01155
01160 EPoint* EPoint::GetSubPoint(int subPoint) {
01161     EPoint *tempPoint;
01162     if(subPoint<=integrationPoints_.size()) tempPoint=&integrationPoints_[subPoint-1];
01163     else tempPoint= NULL;
01164     return tempPoint;
01165 }
01166
01172 std::vector<EPoint>& EPoint::GetSubPoints() {
01173     return integrationPoints_;
01174 }
01175
01180 std::vector<EPoint*>& EPoint::GetMappedPoints() {
01181     return local_mapped_points_;
01182 }

```

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.](#)

```
00001 #include "Equation.h"
00002 #include "Config.h"
00003 #include <iostream>
00004 #include <cmath>
00005 #include <cstdlib>
00006
00011 Equation::Equation() {
00012 }
00013
00020 Equation::Equation(std::string equation, int numParams, const Config &configure) :
    infixEquation_(equation) {
00021     if(equation.size() != 0) {
00022         for(int i=0; i<numParams; i++) {
00023             double tempDouble=0.0;
00024             parameters_.push_back(tempDouble);
00025         }
00026         Parse(configure);
00027     } else {
00028         configure.outStream << "Error: empty equation." << std::endl;
00029         std::exit(-1);
00030     }
00031 }
00032
00040 Equation::Equation(std::string equation, std::vector<double> parameters, const Config &configure) :
    infixEquation_(equation), parameters_(parameters) {
00041     infixEquation_(equation), parameters_(parameters) {
00042         if(equation.size() != 0) {
00043             Parse(configure);
00044         } else {
00045             configure.outStream << "Error: empty equation." << std::endl;
00046             std::exit(-1);
00047         }
00048     }
00049
00056 Equation::Equation(std::string equation, double parameters[], size_t arraySize, const Config
    &configure) :
00057     infixEquation_(equation) {
00058         if(equation.size() != 0) {
00059             parameters_=std::vector<double>(parameters, parameters+arraySize/sizeof(double));
00060             Parse(configure);
00061         } else {
00062             configure.outStream << "Error: empty equation." << std::endl;
00063             std::exit(-1);
00064         }
00065     }
00066
00073 void Equation::Initialize(std::string equation, int numParams, const Config &configure) {
00074     if(equation.size() != 0) {
00075         infixEquation_=equation;
00076         for(int i=0; i<numParams; i++) {
00077             double tempDouble=0.0;
00078             parameters_.push_back(tempDouble);
00079         }
00080         Parse(configure);
00081     } else {
00082         configure.outStream << "Error: empty equation." << std::endl;
00083         std::exit(-1);
00084     }
00085 }
00086
00093 void Equation::BuildFunctionList() {
00094     functionList_["cos"]=GenericFunction(&std::cos);
00095     functionList_["sin"]=GenericFunction(&std::sin);
```

```

00096     functionList_["tan"]=GenericFunction(&std::tan);
00097     functionList_["asin"]=GenericFunction(&std::asin);
00098     functionList_["acos"]=GenericFunction(&std::acos);
00099     functionList_["atan"]=GenericFunction(&std::atan);
00100     functionList_["exp"]=GenericFunction(&std::exp);
00101     functionList_["e^"]=GenericFunction(&std::exp);
00102     functionList_["ln"]=GenericFunction(&std::log);
00103     functionList_["log"]=GenericFunction(&std::log10);
00104     functionList_["sqrt"]=GenericFunction(&std::sqrt);
00105 }
00106
00116 void Equation::Parse(const Config &configure) {
00117     BuildFunctionList();
00118     try{
00119         unsigned int position=0;
00120         std::vector<TokenPair> stack;
00121         //Initial expectation list
00122         unsigned char expecting = FUNCTION|NUMBER|PARAMETER|VARIABLE|LEFTPAR;
00123         while(position<infixEquation_.length()) {
00124             //Retrieve token
00125             TokenPair tempPair=GetToken(position,configure);
00126             //If last token, enforce additional expectations.
00127             if(position>=infixEquation_.length())
00128                 expecting&=FUNCTION|NUMBER|PARAMETER|VARIABLE|RIGHTPAR;
00129             //Check token against expectations
00130             if(!(expecting&tempPair.first)) throw SyntaxError(infixEquation_,3,position-1);
00131             //Set new expectations for next token
00132             if(tempPair.first==NUMBER || tempPair.first==VARIABLE ||
00133                tempPair.first==PARAMETER || tempPair.first==FUNCTION) expecting=OPERATOR|RIGHTPAR;
00134             else if(tempPair.first==OPERATOR || tempPair.first==LEFTPAR)
00135                 expecting=FUNCTION|NUMBER|PARAMETER|VARIABLE|LEFTPAR;
00136             else if(tempPair.first==RIGHTPAR) expecting=OPERATOR|RIGHTPAR;
00137             //Shunting yard algorithm
00138             // Numbers, functions, parameters, and variable go directly to output
00139             if(tempPair.first==NUMBER || tempPair.first==PARAMETER || tempPair.first==VARIABLE || tempPair.first==FUNCTION)
00140             {
00141                 output_.push_back(tempPair);
00142             } else if(tempPair.first==OPERATOR) {
00143                 if(stack.size()==0) stack.push_back(tempPair);
00144                 else while(stack.size()>0) {
00145                     //if top of stack is an operator
00146                     if(stack[stack.size()-1].first==OPERATOR) {
00147                         char lastChar=stack[stack.size()-1].second[0];
00148                         //check precedence and associativity, while conditions are met move from stack to output
00149                         if ((GetOperatorType(tempPair.second[0])<=GetOperatorType(lastChar)&&
00150                             GetOperatorAssociativity(tempPair.second[0])==LEFT) ||
00151                             (GetOperatorType(tempPair.second[0])<GetOperatorType(lastChar)&&
00152                             GetOperatorAssociativity(tempPair.second[0])==RIGHT)) {
00153                             output_.push_back(stack[stack.size()-1]);
00154                             stack.pop_back();
00155                             //if stack is empty, push new operator on stack and break loop
00156                             if(stack.size()==0) {
00157                                 stack.push_back(tempPair);
00158                                 break;
00159                             }
00160                             } else {
00161                                 //if conditions aren't met, push new operator on stack and break loop
00162                                 stack.push_back(tempPair);
00163                                 break;
00164                             }
00165                             } else {
00166                                 //if top of stack is not operator, push new operator on stack and break loop
00167                                 stack.push_back(tempPair);
00168                                 break;
00169                             }
00170                             }
00171                             } else if(tempPair.first==LEFTPAR) {
00172                                 //left parentheses go directly to the stack
00173                                 stack.push_back(tempPair);
00174                                 } else if(tempPair.first==RIGHTPAR) {
00175                                 //right parentheses initiate push of stack to output until left parenthesis is found
00176                                 while(stack.size()>0&&stack[stack.size()-1].first!=LEFTPAR) {
00177                                     output_.push_back(stack[stack.size()-1]);
00178                                     stack.pop_back();
00179                                 }
00180                                 if(stack.size()==0) {
00181                                     //if stack is empty, parentheses were mismatched
00182                                     throw SyntaxError(infixEquation_,2,position-1);
00183                                 } else if(stack[stack.size()-1].first==LEFTPAR) stack.pop_back();
00184                                 }
00185                                 }
00186                                 //push remaining stack to output after all tokens are read
00187                                 while(stack.size()>0) {
00188                                     //if left parenthesis is found, parentheses were mismatched
00189                                     if(stack[stack.size()-1].first==LEFTPAR) throw SyntaxError(infixEquation_,2);
00190                                     output_.push_back(stack[stack.size()-1]);
00191                                 }
00192                                 }

```

```

00189     stack.pop_back();
00190 }
00191 } catch (SyntaxError e) {
00192     configure.outStream « e.what() « std::endl;
00193     std::exit(-1);
00194 }
00195 }
00200 std::vector<double> Equation::GetParameters() const {
00201     return parameters_;
00202 }
00203
00208 void Equation::SetParameter(unsigned int index, double value, const Config& configure) {
00209     if(index<parameters_.size()) {
00210         parameters_[index]=value;
00211         for(unsigned int i = 0; i<subEquations_.size(); i++)
00212             subEquations_[i].SetParameter(index,value,configure);
00213     } else configure.outStream « "Error: Parameter index " « index « " greater than vector size." «
00214         std::endl;
00215 }
00220 bool Equation::IsOperator(char c) const {
00221     return (c=='+'||c=='-'||c=='*'||c=='/'||c=='^') ? (true) : (false);
00222 }
00223
00229 bool Equation::IsDigit(char c) const {
00230     return (c>='0' && c<='9') ? (true) : (false);
00231 }
00232
00239 unsigned int Equation::FindFunction(unsigned int &position) {
00240     unsigned int length = 0;
00241     for(std::map<std::string,GenericFunction>::iterator it = functionList_.begin();
00242         it!=functionList_.end();it++) {
00243         std::string searchKey = it->first+'(';
00244         if(infixEquation_.substr(position,searchKey.length()) == searchKey) {
00245             length = searchKey.length()-1;
00246             break;
00247         }
00248     }
00249     return length;
00250 }
00251
00258 Equation::TokenPair Equation::GetToken(unsigned int &position, const Config& configure) {
00259     try {
00260         TokenType tempType;
00261         std::string tempString;
00262         //Number (positive decimal, integer, or exponential notation)
00263         if(IsDigit(infixEquation_[position])||infixEquation_[position]=='.') {
00264             tempType=NUMBER;
00265
00266             while((IsDigit(infixEquation_[position])||infixEquation_[position]=='.'||infixEquation_[position]=='e'||infixEquation_[position]=='E')
00267                 ||(infixEquation_[position]=='-'&&(infixEquation_[position-1]=='e'||infixEquation_[position-1]=='E'))
00268                 ||(infixEquation_[position]=='+'&&(infixEquation_[position-1]=='e'||infixEquation_[position-1]=='E'))
00269                 &&position<infixEquation_.length()) {
00270                 tempString+=infixEquation_[position];
00271                 position++;
00272             }
00273             //Check for negation. Implimented as a function. MUST be checked before minus is parsed as an
00274             //operator.
00275             else if(infixEquation_[position]=='-'&&(position==0||IsOperator(infixEquation_[position-1])||
00276                 infixEquation_[position-1]=='(')) {
00277                 tempType=FUNCTION;
00278                 position++;
00279                 tempString="neg";
00280             }
00281             //Operator (+,-,*,/, or ^)
00282             else if(IsOperator(infixEquation_[position])) {
00283                 tempType=OPERATOR;
00284                 tempString+=infixEquation_[position];
00285                 position++;
00286             }
00287             //Parameter (must be of form a0,a1,...)
00288             else if(infixEquation_[position]=='a'&&IsDigit(infixEquation_[position+1])) {
00289                 tempType=PARAMETER;
00290                 position++;
00291                 while(IsDigit(infixEquation_[position])&&position<infixEquation_.length()) {
00292                     tempString+=infixEquation_[position];
00293                     position++;
00294                 }
00295                 std::stringstream stm;
00296                 stm.str(tempString);
00297                 unsigned int paramNumber;stm>>paramNumber;
00298                 if(paramNumber+1>parameters_.size()) throw SyntaxError(infixEquation_,1,position-1);
00299             }
00300             //Dependent variable

```



```

00300     else if(infixEquation_[position]=='x') {
00301         tempType=VARIABLE;
00302         tempString+=infixEquation_[position];
00303         position++;
00304     }
00305     //Left Parenthesis
00306     else if(infixEquation_[position]=='(') {
00307         tempType=LEFTPAR;
00308         tempString+=infixEquation_[position];
00309         position++;
00310     }
00311     //Right Parenthesis
00312     else if(infixEquation_[position]==')') {
00313         tempType=RIGHTPAR;
00314         tempString+=infixEquation_[position];
00315         position++;
00316     }
00317     //Functions (currently supports whatever is linked in BuildFunctionList())
00318     else if(FindFunction(position)) {
00319         tempType=FUNCTION;
00320         unsigned int length = FindFunction(position);
00321         tempString=infixEquation_.substr(position,length);
00322         position+=(length+1);
00323     }
00324     //Unrecognized Tokens
00325     else throw SyntaxError(infixEquation_,0,position);
00326     //Read subequation if token is function
00327     if(tempType==FUNCTION) {
00328         std::string subString;
00329         //for negation, read until an operator (except ^), or unmatched right paranthesis, is found
00330         // and create subequation
00331         if(tempString=="neg") {
00332             int parenCount=0;
00333             while(((infixEquation_[position]!='')&&
00334                 !((IsOperator(infixEquation_[position])&&infixEquation_[position]!='^') &&
00335                 !(infixEquation_[position-1]=='e' || infixEquation_[position-1]=='E')) ||
00336                 parenCount!=0)&&
00337                 position<infixEquation_.length()) {
00338                 if(infixEquation_[position]=='(') parenCount++;
00339                 else if(infixEquation_[position]==')') parenCount--;
00340                 subString+=infixEquation_[position];
00341                 position++;
00342             }
00343             if(parenCount>0&&position>=infixEquation_.length()) throw SyntaxError(infixEquation_,2);
00344             } else {
00345                 //for regular functions, read until closing right parenthesis is found
00346                 // and create subequation
00347                 int parenCount=1;
00348                 while(parenCount>0&&position<infixEquation_.length()) {
00349                     if(infixEquation_[position]=='(') parenCount++;
00350                     else if(infixEquation_[position]==')') parenCount--;
00351                     if(parenCount!=0) subString+=infixEquation_[position];
00352                     position++;
00353                 }
00354                 if(parenCount>0&&position>=infixEquation_.length()) throw SyntaxError(infixEquation_,2);
00355             }
00356             Equation subEquation(subString,parameters_,configure);
00357             subEquations_.push_back(subEquation);
00358             int subIndex=subEquations_.size()-1;
00359             std::ostringstream stm;
00360             stm << subIndex;
00361             tempString+=stm.str();
00362         }
00363         return TokenPair(tempType,tempString);
00364     } catch (SyntaxError e) {
00365         configure.outStream << e.what() << std::endl;
00366         std::exit(-1);
00367     }
00368 }
00369
00374 Equation::OperatorType Equation::GetOperatorType(char c) const {
00375     switch (c) {
00376         case '+': return ADD;
00377         case '-': return SUBTRACT;
00378         case '*': return MULT;
00379         case '/': return DIVIDE;
00380         case '^': return POWER;
00381         default : return BADTYPE;
00382     }
00383 }
00384
00389 Equation::Associativity Equation::GetOperatorAssociativity(char c) const {
00390     switch (c) {
00391         case '+': return LEFT;
00392         case '-': return LEFT;
00393         case '*': return LEFT;
00394         case '/': return LEFT;

```

```

00395     case '^': return RIGHT;
00396     default : return LEFT;
00397 }
00398 }
00399
00404 std::string Equation::BinaryOperation(double left, double right, char op, const Config& configure)
const {
00405     std::ostringstream stm;
00406     stm.precision(15);
00407     double result;
00408     switch(op) {
00409         case '+':
00410             result=left+right;
00411             break;
00412         case '-':
00413             result=left-right;
00414             break;
00415         case '*':
00416             result=left*right;
00417             break;
00418         case '/':
00419             result=left/right;
00420             break;
00421         case '^':
00422             if(left<0.0&&fabs(int(right)-right)>0.0) {
00423                 configure.outStream << "Warning: Exponent results in unsupported imaginary number." << std::endl;
00424                 result=0.0;
00425             } else result=pow(left,right);
00426             break;
00427         default:
00428             result=0.0;
00429     }
00430     stm<<result;
00431     return stm.str();
00432 }
00433
00438 double Equation::FunctionOperation(TokenPair token, double x, const Config& configure) const {
00439     double result=0.0;
00440     if(token.second.substr(0,3)=="neg") {
00441         int subEquationIndex;
00442         std::ostringstream stm;
00443         stm.str(token.second.substr(3));
00444         stm<>subEquationIndex;
00445         result=-1.*subEquations_[subEquationIndex].Evaluate(configure,x);
00446     } else {
00447         for(std::map<std::string,GenericFunction>::const_iterator it = functionList_.begin();
00448             it!=functionList_.end();it++) {
00449             if(it->first==token.second.substr(0,it->first.length())) {
00450                 int subEquationIndex;
00451                 std::ostringstream stm;
00452                 stm.str(token.second.substr(it->first.length()));
00453                 stm<>subEquationIndex;
00454                 result=it->second.Evaluate(subEquations_[subEquationIndex].Evaluate(configure,x));
00455                 break;
00456             }
00457         }
00458     }
00459     return result;
00460 }
00461
00466 double Equation::GetTokenValue(TokenPair token, double x, const Config& configure) const {
00467     double value;
00468     std::ostringstream stm;
00469     if(token.first==NUMBER) {
00470         stm.clear();
00471         stm.str(token.second);
00472         stm >> value;
00473     } else if(token.first==VARIABLE) {
00474         value=x;
00475     } else if(token.first==PARAMETER) {
00476         unsigned int paramNumber;
00477         stm.clear();
00478         stm.str(token.second);
00479         stm >> paramNumber;
00480         value=parameters_[paramNumber];
00481     } else if(token.first==FUNCTION)
00482         value=FunctionOperation(token,x,configure);
00483     else value=0.0;
00484     return value;
00485 }
00486
00491 double Equation::Evaluate(const Config& configure,double x) const {
00492     std::vector<TokenPair> localOutput=output_;
00493     std::ostringstream stm;
00494     double result=0.0;
00495     int i=0;
00496     if(localOutput.size()==1) {

```

```

00497     result=GetTokenValue(localOutput[0],x,configure);
00498 } else {
00499     while(localOutput.size()!=1) {
00500         if(localOutput[i].first==OPERATOR) {
00501             char op=localOutput[i].second[0];
00502             double left=GetTokenValue(localOutput[i-2],x,configure);
00503             double right=GetTokenValue(localOutput[i-1],x,configure);
00504             localOutput.erase(localOutput.begin()+i-2,localOutput.begin()+i+1);
00505             localOutput.insert(localOutput.begin()+i-2,
00506                               TokenPair(NUMBER,BinaryOperation(left,right,op,configure)));
00507             i--;
00508         }
00509         else i++;
00510     }
00511     stm.clear();
00512     stm.str(localOutput[0].second);
00513     stm >> result;
00514 }
00515 return result;
00516 }
00517

```

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.](#)

```

00001 #include "CNuc.h"
00002 #include "DataLine.h"
00003 #include "EData.h"
00004 #include "ESegment.h"
00005 #include "ExtrapLine.h"
00006 #include "SegLine.h"
00007
00013 ESegment::ESegment(SegLine segLine) {
00014     entrancekey_=segLine.entranceKey();
00015     exitkey_=segLine.exitKey();
00016     min_e_=segLine.minE();
00017     max_e_=segLine.maxE();
00018     min_a_=segLine.minA();
00019     max_a_=segLine.maxA();
00020     e_step_=0.0;
00021     a_step_=0.0;
00022     segment_chi_squared_=0.0;
00023     if(segLine.isDiff()==1) isdifferential_=true;
00024     else isdifferential_=false;
00025     if(segLine.isDiff()==2) {
00026         isphase_=true;
00027         j_=segLine.phaseJ();
00028         l_=segLine.phaseL();
00029     } else {
00030         isphase_=false;
00031         j_=0.0;
00032         l_=0;
00033     }
00034     isTotalCapture_ = (segLine.isDiff()==3) ? 1 : 0;
00035     isAngDist_=false;
00036     maxAngDistOrder_=0;
00037     datafile_=segLine.dataFile();
00038     dataNorm_= dataNormNominal_ = segLine.dataNorm();
00039     dataNormError_=segLine.dataNormError();
00040     if(segLine.varyNorm()==1) varyNorm_=true;

```

```

00041     else varyNorm_=false;
00042     targetEffectNum_=0;
00043     isTargetEffect_=false;
00044 }
00045
00052 ESegment::ESegment(ExtrapLine extrapLine) {
00053     entrancekey_=extrapLine.entranceKey();
00054     exitkey_=extrapLine.exitKey();
00055     min_e_=extrapLine.minE();
00056     max_e_=extrapLine.maxE();
00057     min_a_=extrapLine.minA();
00058     max_a_=extrapLine.maxA();
00059     e_step_=extrapLine.eStep();
00060     a_step_=extrapLine.aStep();
00061     segment_chi_squared_=0.0;
00062     if(extrapLine.isDiff()==1) isdifferential_=true;
00063     else isdifferential_=false;
00064     if(extrapLine.isDiff()==2) {
00065         isphase_=true;
00066         j_=extrapLine.phaseJ();
00067         l_=extrapLine.phaseL();
00068     } else {
00069         isphase_=false;
00070         j_=0.0;
00071         l_=0;
00072     }
00073     if(extrapLine.isDiff()==3) {
00074         isAngDist_=true;
00075         maxAngDistOrder_=extrapLine.maxAngDistOrder();
00076     } else {
00077         isAngDist_=false;
00078         maxAngDistOrder_=0;
00079     }
00080     isTotalCapture_ = (extrapLine.isDiff()==4) ? 1 : 0;
00081     datafile_="";
00082     dataNorm_ = dataNormNominal_ = 1.;
00083     dataNormError_=0.;
00084     varyNorm_=false;
00085     targetEffectNum_=0;
00086     isTargetEffect_=false;
00087 }
00088
00094 bool ESegment::IsInSegment(EPoint point) {
00095     bool b=false;
00096     if(point.GetLabEnergy()>=this->GetMinEnergy()&&
00097        point.GetLabEnergy()<=this->GetMaxEnergy()) {
00098         if(this->IsDifferential()) {
00099             if(point.GetLabAngle()>=this->GetMinAngle()&&
00100                point.GetLabAngle()<=this->GetMaxAngle()) b=true;
00101         } else b=true;
00102     }
00103     return b;
00104 }
00105
00110 bool ESegment::IsDifferential() const {
00111     return isdifferential_;
00112 }
00113
00118 bool ESegment::IsPhase() const {
00119     return isphase_;
00120 }
00121
00128 int ESegment::IsTotalCapture() const {
00129     return isTotalCapture_;
00130 }
00131
00136 bool ESegment::IsAngularDist() const {
00137     return isAngDist_;
00138 }
00139
00145 bool ESegment::IsTargetEffect() const {
00146     return isTargetEffect_;
00147 }
00148
00154 bool ESegment::IsVaryNorm() const {
00155     return varyNorm_;
00156 }
00157
00162 int ESegment::NumPoints() const {
00163     return points_.size();
00164 }
00165
00170 int ESegment::GetEntranceKey() const {
00171     return entrancekey_;
00172 }
00173
00178 int ESegment::GetExitKey() const {

```

```

00179     return exitkey_;
00180 }
00181
00182 int ESegment::Fill(CNuc *theCNuc, EData *theData, const Config& configure) {
00183     std::string infile=this->GetDataFile();
00184     std::ifstream in(infile.c_str());
00185     if(!in) return -1;
00186     while(!in.eof()) {
00187         DataLine line(in);
00188         if(!in.eof()) {
00189             EPoint NewEPoint(line,this);
00190             if(this->IsInSegment(NewEPoint)) {
00191                 this->AddPoint(NewEPoint);
00192                 PPair *entrancePair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetEntranceKey()));
00193                 PPair *exitPair=theCNuc->GetPair(theCNuc->GetPairNumFromKey(this->GetExitKey()));
00194                 this->GetPoint(this->NumPoints()-1)->SetParentData(theData);
00195                 if(entrancePair->GetPType()==20) this->GetPoint(this->NumPoints()-1)->ConvertDecayEnergy(exitPair);
00196                 else this->GetPoint(this->NumPoints()-1)->ConvertLabEnergy(entrancePair);
00197                 if(exitPair->GetPType()==0&&this->IsDifferential()&&!this->IsPhase()) {
00198                     if(this->GetEntranceKey()==this->GetExitKey()) {
00199                         this->GetPoint(this->NumPoints()-1)->ConvertLabAngle(entrancePair);
00200                     } else {
00201                         this->GetPoint(this->NumPoints()-1)->ConvertLabAngle(entrancePair,exitPair,configure);
00202                     }
00203                     this->GetPoint(this->NumPoints()-1)->ConvertCrossSection(entrancePair,exitPair);
00204                 }
00205             }
00206         }
00207     }
00208     in.close();
00209     return 0;
00210 }
00211
00212 int ESegment::GetL() const {
00213     return l_;
00214 }
00215
00216 int ESegment::GetTargetEffectNum() const {
00217     return targetEffectNum_;
00218 }
00219
00220 int ESegment::GetSegmentKey() const {
00221     return segmentKey_;
00222 }
00223
00224 int ESegment::GetMaxAngDistOrder() const {
00225     return maxAngDistOrder_;
00226 }
00227
00228 double ESegment::GetMinEnergy() const {
00229     return min_e_;
00230 }
00231
00232 double ESegment::GetMaxEnergy() const {
00233     return max_e_;
00234 }
00235
00236 double ESegment::GetMinAngle() const {
00237     return min_a_;
00238 }
00239
00240 double ESegment::GetMaxAngle() const {
00241     return max_a_;
00242 }
00243
00244 double ESegment::GetSegmentChiSquared() const {
00245     return segment_chi_squared_;
00246 }
00247
00248 double ESegment::GetEStep() const {
00249     return e_step_;
00250 }
00251
00252 double ESegment::GetAStep() const {
00253     return a_step_;
00254 }
00255
00256 double ESegment::GetJ() const {
00257     return j_;
00258 }
00259
00260 double ESegment::GetNorm() const {
00261     return dataNorm_;
00262 }
00263
00264 double ESegment::GetNominalNorm() const {
00265     return dataNormNominal_;
00266 }

```

```

00330 }
00331
00336 double ESegment::GetNormError() const {
00337     return dataNormError_;
00338 }
00339
00344 std::string ESegment::GetDataFile() const {
00345     return datafile_;
00346 }
00347
00352 void ESegment::AddPoint(EPoint point) {
00353     points_.push_back(point);
00354 }
00355
00360 void ESegment::SetSegmentChiSquared(double chiSquared) {
00361     segment_chi_squared_=chiSquared;
00362 }
00363
00369 void ESegment::SetTargetEffectNum(int targetEffectNum) {
00370     targetEffectNum_=targetEffectNum;
00371     isTargetEffect_=true;
00372 }
00373
00378 void ESegment::SetSegmentKey (int segmentKey) {
00379     segmentKey_=segmentKey;
00380 }
00381
00386 void ESegment::SetNorm(double norm) {
00387     dataNorm_=norm;
00388 }
00389
00394 void ESegment::SetExitKey(int key) {
00395     exitkey_=key;
00396     for(int i=1;i<=this->NumPoints();i++)
00397         this->GetPoint(i)->SetExitKey(key);
00398 }
00399
00406 void ESegment::SetIsTotalCapture(int num) {
00407     isTotalCapture_=num;
00408 }
00409
00414 void ESegment::SetVaryNorm(bool varyNorm) {
00415     varyNorm_=varyNorm;
00416 }
00417
00422 EPoint *ESegment::GetPoint(int pointNum) {
00423     EPoint *b=&points_[pointNum-1];
00424     return b;
00425 }
00426
00431 std::vector<EPoint>& ESegment::GetPoints() {
00432     return points_;
00433 }

```

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.](#)

```

00001 #include "CNuc.h"
00002 #include "EPoint.h"
00003 #include "GenMatrixFunc.h"
00004 #include <assert.h>
00005
00012 void GenMatrixFunc::CalculateCrossSection(EPoint *point) {

```

```

00013     complex sum(0.,0.);
00014     int aa=compound()->GetPairNumFromKey(point->GetEntranceKey());
00015     int ir=0;
00016     while(ir<compound()->GetPair(aa)->NumDecays()) {
00017         ir++;
00018
00019         if(compound()->GetPair(aa)->GetDecay(ir)->GetPairNum()==compound()->GetPairNumFromKey(point->GetExitKey()))
00020             break;
00021     }
00022     Decay *theDecay=compound()->GetPair(aa)->GetDecay(ir);
00023     if(compound()->GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->GetPType()==10 &&
00024         (configure().paramMask & Config::USE_RMC_FORMALISM)) {
00025         int decayNum=0;
00026         while(decayNum<compound()->GetPair(aa)->NumDecays()) {
00027             decayNum++;
00028             if(compound()->GetPair(aa)->GetDecay(decayNum)->GetPairNum()==aa) break;
00029         }
00030         for(int k=1;k<=compound()->GetPair(aa)->GetDecay(decayNum)->NumKGroups();k++) {
00031             for(int m=1;m<=compound()->GetPair(aa)->GetDecay(decayNum)->GetKGroup(k)->NumMGroups();m++) {
00032                 MGroup *theMGroup=compound()->GetPair(aa)->GetDecay(decayNum)->GetKGroup(k)->GetMGroup(m);
00033                 if(theMGroup->GetChNum()==theMGroup->GetChpNum()) {
00034                     double jValue=compound()->GetJGroup(theMGroup->GetJNum())->GetJ();
00035                     sum+=2.*point->GetGeometricalFactor()*
00036                         (2.*jValue+1.)*compound()->GetPair(aa)->GetI1I2Factor()*
00037                         imag(this->GetTMatrixElement(k,m,decayNum));
00038                 }
00039             }
00040         }
00041         for(int dp=1;dp<=compound()->GetPair(aa)->NumDecays();dp++) {
00042             if(compound()->GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->GetPairNum()->GetPType()==0) {
00043                 for(int k=1;k<=compound()->GetPair(aa)->GetDecay(dp)->NumKGroups();k++) {
00044                     this->ClearTempTMatrices();
00045                     for(int m=1;m<=compound()->GetPair(aa)->GetDecay(dp)->GetKGroup(k)->NumMGroups();m++) {
00046                         MGroup *theMGroup=compound()->GetPair(aa)->GetDecay(dp)->GetKGroup(k)->GetMGroup(m);
00047                         int
00048                         lValue=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChNum())->GetL();
00049                         int
00050                         lpValue=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChpNum())->GetL();
00051                         double jValue=compound()->GetJGroup(theMGroup->GetJNum())->GetJ();
00052                         int tempTNum=this->IsTempTMatrix(jValue,lValue,lpValue);
00053                         if(!tempTNum) {
00054                             TempTMatrix temptmatrix={jValue,lValue,lpValue,this->GetTMatrixElement(k,m,dp)};
00055                             this->NewTempTMatrix(temptmatrix);
00056                         } else this->AddToTempTMatrix(tempTNum,this->GetTMatrixElement(k,m,dp));
00057                     }
00058                     for(int temp=1;temp<=this->NumTempTMatrices();temp++) {
00059                         sum+=point->GetGeometricalFactor()*
00060                             (2.*this->GetTempTMatrix(temp)->jValue+1.)*
00061                             compound()->GetPair(aa)->GetI1I2Factor()*
00062                             (this->GetTempTMatrix(temp)->TMatrix)*conj(this->GetTempTMatrix(temp)->TMatrix);
00063                     }
00064                 }
00065             }
00066             point->SetFitCrossSection(real(sum)/100.);
00067         } else {
00068             if(!point->IsPhase()) {
00069                 double angleIntegratedXS=0.;
00070                 if(!point->IsDifferential()) {
00071                     for(int k=1;k<=theDecay->NumKGroups();k++) {
00072                         this->ClearTempTMatrices();
00073                         for(int m=1;m<=theDecay->GetKGroup(k)->NumMGroups();m++) {
00074                             if(compound()->GetPair(aa)->GetPType()==20) {
00075                                 sum+=25.*this->GetTMatrixElement(k,m)*conj(this->GetTMatrixElement(k,m));
00076                             } else {
00077                                 MGroup *theMGroup=theDecay->GetKGroup(k)->GetMGroup(m);
00078                                 int
00079                                 lValue=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChNum())->GetL();
00080                                 int
00081                                 lpValue=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChpNum())->GetL();
00082                                 double jValue=compound()->GetJGroup(theMGroup->GetJNum())->GetJ();
00083                                 int tempTNum=this->IsTempTMatrix(jValue,lValue,lpValue);
00084                                 if(!tempTNum) {
00085                                     TempTMatrix temptmatrix={jValue,lValue,lpValue,this->GetTMatrixElement(k,m)};
00086                                     this->NewTempTMatrix(temptmatrix);
00087                                 } else this->AddToTempTMatrix(tempTNum,this->GetTMatrixElement(k,m));
00088                             }
00089                         }
00090                     }
00091                     if(compound()->GetPair(aa)->GetPType()==20) continue;
00092                     for(int m=1;m<=theDecay->GetKGroup(k)->NumECMGroups();m++) {
00093                         ECMGroup *theECMGroup=theDecay->GetKGroup(k)->GetECMGroup(m);
00094                         int lValue=theECMGroup->GetL();
00095                         int lpValue=theECMGroup->GetMult();
00096                         double jValue=theECMGroup->GetJ();
00097                         int tempTNum=this->IsTempTMatrix(jValue,lValue,lpValue);
00098                         if(!tempTNum) {
00099                             TempTMatrix temptmatrix={jValue,lValue,lpValue,this->GetTMatrixElement(k,m)};

```

```

00094         this->NewTempTMatrix(temptmatrix);
00095     } else this->AddToTempTMatrix(temptNum,this->GetECTMatrixElement(k,m));
00096 }
00097 for(int temp=1;temp<=this->NumTempTMatrices();temp++) {
00098     sum+=point->GetGeometricalFactor()*
00099         (2.*this->GetTempTMatrix(temp)->jValue+1.)*
00100         compound()->GetPair(aa)->GetI1I2Factor()*
00101         (this->GetTempTMatrix(temp)->TMatrix)*conj(this->GetTempTMatrix(temp)->TMatrix);
00102 }
00103 }
00104 angleIntegratedXS=real(sum)/100.;
00105 if(!point->IsAngularDist()) {
00106     point->SetFitCrossSection(angleIntegratedXS);
00107     return;
00108 }
00109 }
00110 std::vector<double>
angularCoeff(std::min(point->GetMaxLOrder()+1,point->GetMaxAngDistOrder()+1),0.);
00111 for(int kL=1;kL<=theDecay->NumKLGrouPs();kL++) {
00112     for(int inter=1;inter<=theDecay->GetKLGroup(kL)
00113         ->NumInterferences();inter++) {
00114         Interference *theInterference=theDecay->GetKLGroup(kL)
00115             ->GetInterference(inter);
00116         complex T1(0.0,0.0),T2(0.0,0.0);
00117         std::string interferenceType=theInterference->GetInterferenceType();
00118         if(interferenceType=="RR") {
00119             T1=this->GetTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM1());
00120             T2=this->GetTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM2());
00121         } else if(interferenceType=="ER") {
00122             T1=this->GetECTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM1());
00123             T2=this->GetTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM2());
00124         } else if(interferenceType=="RE") {
00125             T1=this->GetTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM1());
00126             T2=this->GetECTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM2());
00127         } else if(interferenceType=="EE") {
00128             T1=this->GetECTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM1());
00129             T2=this->GetECTMatrixElement(theDecay->GetKLGroup(kL)->GetK(),theInterference->GetM2());
00130         }
00131         int lOrder = theDecay->GetKLGroup(kL)->GetLOrder();
00132         sum+=theInterference->GetZ1Z2()*T1*conj(T2)*
00133             point->GetLegendreP(lOrder);
00134         if((lOrder < angularCoeff.size()) && point->IsAngularDist()) {
00135             double tempCoeff=angularCoeff[lOrder]+
00136                 real(theInterference->GetZ1Z2()*T1*conj(T2))*point->GetGeometricalFactor()*
00137                 compound()->GetPair(aa)->GetI1I2Factor()/100.*4./angleIntegratedXS;
00138             angularCoeff[lOrder]=tempCoeff;
00139         }
00140     }
00141 }
00142 if(point->IsAngularDist()) {
00143     point->SetAngularDists(angularCoeff);
00144     return;
00145 }
00146 complex RT=sum/pi*point->GetGeometricalFactor()*
00147 compound()->GetPair(aa)->GetI1I2Factor();
00148
00149 complex CT(0.,0.), IT(0.,0.);
00150 if(aa==ir) {
00151     complex coulombAmplitude=point->GetCoulombAmplitude();
00152     CT=coulombAmplitude*conj(coulombAmplitude)*point->GetGeometricalFactor();
00153 }
00154 sum=complex(0.,0.);
00155 for(int k=1;k<=theDecay->NumKGroups();k++) {
00156     for(int m=1;m<=theDecay->GetKGroup(k)->NumMGroups();m++) {
00157         MGroup *theMGroup=theDecay->GetKGroup(k)->GetMGroup(m);
00158         AChannel
*entranceChannel=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChNum());
00159         AChannel
*exitChannel=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChpNum());
00160         if(entranceChannel==exitChannel)
00161             sum+=theMGroup->GetStatSpinFactor()*
00162                 coulombAmplitude*conj(this->GetTMatrixElement(k,m))*
00163                 point->GetLegendreP(compound()->GetJGroup(theMGroup->GetJNum())->
00164                     GetChannel(theMGroup->GetChNum())->GetL());
00165     }
00166 }
00167 IT=complex(0.,1.)/sqrt(pi)*sum*point->GetGeometricalFactor();
00168 }
00169 point->SetFitCrossSection((real(CT)+real(RT)+real(IT))/100.);
00170 } else if(aa==ir) {
00171     double segmentJ=point->GetJ();
00172     int segmentL=point->GetL();
00173     this->ClearTempTMatrices();
00174     for(int k=1;k<=theDecay->NumKGroups();k++) {
00175         for(int m=1;m<=theDecay->GetKGroup(k)->NumMGroups();m++) {
00176             MGroup *theMGroup=theDecay->GetKGroup(k)->GetMGroup(m);
00177             double jValue=compound()->GetJGroup(theMGroup->GetJNum())->GetJ();

```



```

00178     AChannel
*entranceChannel=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChNum());
00179     int lValue=entranceChannel->GetL();
00180     AChannel
*exitChannel=compound()->GetJGroup(theMGroup->GetJNum())->GetChannel(theMGroup->GetChNum());
00181     if(jValue==segmentJ&&lValue==segmentL&&entranceChannel==exitChannel) {
00182         complex
expCoulPhaseSquared=point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChNum())*
00183         point->GetExpCoulombPhase(theMGroup->GetJNum(),theMGroup->GetChNum());
00184         complex theUMatrix=(expCoulPhaseSquared-this->GetTMatrixElement(k,m))/expCoulPhaseSquared;
00185         int tempTNum=this->IsTempTMatrix(jValue,lValue,lValue);
00186         if(!tempTNum) {
00187             TempTMatrix temptmatrix={jValue,lValue,lValue,theUMatrix};
00188             this->NewTempTMatrix(temptmatrix);
00189         } else this->AddToTempTMatrix(tempTNum,theUMatrix);
00190     }
00191 }
00192 }
00193 assert(this->NumTempTMatrices()<=1);
00194 double phase=0.0;
00195 if(this->NumTempTMatrices()==1) phase = 180.0/pi/2.0*
00196 atan2(imag(this->GetTempTMatrix(1)->TMatrix),real(this->GetTempTMatrix(1)->TMatrix));
00197 //if(segmentL%2!=0&&phase<0) phase+=180.0;
00198 point->SetFitCrossSection(phase);
00199 }
00200 }
00201 }
00202
00203
00208 void GenMatrixFunc::NewTempTMatrix(TempTMatrix tempTMatrix) {
00209     temp_t_matrices_.push_back(tempTMatrix);
00210 }
00211
00216 void GenMatrixFunc::AddToTempTMatrix(int tempTMatrixNum, complex tempValue) {
00217     this->GetTempTMatrix(tempTMatrixNum)->TMatrix+=tempValue;
00218 }
00219 }
00220
00225 void GenMatrixFunc::ClearTempTMatrices() {
00226     temp_t_matrices_.clear();
00227 }
00228
00234 void GenMatrixFunc::AddTMatrixElement(int kGroupNum ,int mGroupNum,complex tMatrixElement, int
decayNum) {
00235     matrix_c d;
00236     vector_c e;
00237     while(decayNum>tmatrix_.size()) tmatrix_.push_back(d);
00238     while(kGroupNum>tmatrix_[decayNum-1].size()) tmatrix_[decayNum-1].push_back(e);
00239     tmatrix_[decayNum-1][kGroupNum-1].push_back(tMatrixElement);
00240     assert(kGroupNum==tmatrix_[decayNum-1].size());
00241     assert(mGroupNum==tmatrix_[decayNum-1][kGroupNum-1].size());
00242 }
00243
00249 void GenMatrixFunc::AddECTMatrixElement(int kGroupNum ,int mGroupNum,complex tMatrixElement) {
00250     vector_c d;
00251     while(kGroupNum>ec_tmatrix_.size()) ec_tmatrix_.push_back(d);
00252     ec_tmatrix_[kGroupNum-1].push_back(tMatrixElement);
00253     assert(mGroupNum==ec_tmatrix_[kGroupNum-1].size());
00254 }
00255
00261 int GenMatrixFunc::IsTempTMatrix(double jValue, int lValue, int lPrimeValue) {
00262     int d=0;
00263     bool e=false;
00264     while(!e&&d<this->NumTempTMatrices()) {
00265         if(jValue==this->GetTempTMatrix(d+1)->jValue&&
00266             lValue==this->GetTempTMatrix(d+1)->lValue&&
00267             lPrimeValue==this->GetTempTMatrix(d+1)->lpValue) e=true;
00268         d++;
00269     }
00270     if(!e) return 0;
00271     else return d;
00272 }
00273
00278 int GenMatrixFunc::NumTempTMatrices() const {
00279     return temp_t_matrices_.size();
00280 }
00281
00286 TempTMatrix *GenMatrixFunc::GetTempTMatrix(int tempTMatrixNum) {
00287     TempTMatrix *b=&temp_t_matrices_[tempTMatrixNum-1];
00288     return b;
00289 }
00290
00295 complex GenMatrixFunc::GetTMatrixElement(int kGroupNum, int mGroupNum, int decayNum) const {
00296     return tmatrix_[decayNum-1][kGroupNum-1][mGroupNum-1];
00297 }
00298

```

```

00303 complex GenMatrixFunc::GetECTMatrixElement(int kGroupNum, int ecMGroupNum) const {
00304     return ec_tmatrix_[kGroupNum-1][ecMGroupNum-1];
00305 }

```

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.](#)

```

00001 #include "GSLException.h"
00002 #include <gsl/gsl_errno.h>
00003
00004 void GSLException::GSLErrorHandler(const char* reason,
00005                                     const char* file,
00006                                     int line,
00007                                     int errorCode) {
00008     if(errorCode==GSL_EOVRFLW) return;
00009     std::ostringstream stm;
00010     stm << "GSL Error on line " << line << " of " << file << std::endl
00011         << reason << std::endl;
00012     std::string message = stm.str();
00013     throw GSLException(message);
00014 }

```

8.272 /Users/kuba/Desktop/R-Matrix/AZURE2/src/IntegratedFermiFunc.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.](#)

```

00001 #include "IntegratedFermiFunc.h"
00002 #include <math.h>
00003 #include <gsl/gsl_sf_gamma.h>
00004 #include <gsl/gsl_integration.h>
00005
00006 const double IntegratedFermiFunc::alpha_=1./137.036;
00007 const double IntegratedFermiFunc::pi_=3.14159;
00008 const double IntegratedFermiFunc::electronMass_=0.51099891;
00009 const double IntegratedFermiFunc::hbarc_=197.327;
00010
00019 IntegratedFermiFunc::IntegratedFermiFunc(int charge, double V0) :
00020     charge_(charge), V0_(V0) {
00021 }
00022
00029 double IntegratedFermiFunc::operator()(double W0,
00030                                         double Z,
00031                                         double radius) {

```

```

00032
00033     if(W0<=1) return 0.;
00034
00035     double gamma0 = sqrt(1.-alpha_*alpha_*Z*Z);
00036     double GammaDenom = gsl_sf_gamma(2.*gamma0+1.);
00037     Params_ params = {charge_,
00038                       gamma0,
00039                       Z,
00040                       radius,
00041                       W0,
00042                       GammaDenom*GammaDenom,
00043                       V0_};
00044
00045     gsl_integration_workspace * w
00046     = gsl_integration_workspace_alloc (1000);
00047     gsl_function f;
00048     f.function = &IntegratedFermiFunc::Integrand;
00049     f.params = &params;
00050
00051     double result, error;
00052     gsl_integration_qags (&f, 1., params.W0, 0., 1.e-6, 1000, w, &result, &error);
00053
00054     gsl_integration_workspace_free (w);
00055
00056     return result;
00057 }
00058
00064 double IntegratedFermiFunc::Integrand(double x, void* p) {
00065     Params_* params = (Params_*)p;
00066
00067     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.);
00068     double eta = (params->charge<0) ? alpha_*params->Z*W/sqrt(W*W-1.) :
00069     -alpha_*params->Z*W/sqrt(W*W-1.);
00070     gsl_sf_result GammaNumValue;
00071     gsl_sf_result GammaNumArg;
00072     gsl_sf_lngamma_complex_e(params->gamma0,eta,&GammaNumValue,&GammaNumArg);
00073     double GammaNum2 = exp(2.*GammaNumValue.val);
00074
00075     double result =
00076     2.*(1.+params->gamma0)*
00077     exp(pi_*eta-2.*(1.-params->gamma0)*
00078     log(2.*electronMass_*params->radius*sqrt(W*W-1.)/hbarc_))*
00079     GammaNum2/params->GammaDenom2*sqrt(W*W-1.)*W*(params->W0-W)*(params->W0-W);
00080
00081     if(params->V0!=0.) {
00082         result*=sqrt((W*W-1.)/(x*x-1.))*W/x;
00083     }
00084
00085     return result;
00086 }

```

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.](#)

```

00001 #include "Interference.h"
00002
00011 Interference::Interference(int mGroupNum1, int mGroupNum2, double z1z2Coeff, std::string
interferenceType) :
00012     m1_(mGroupNum1), m2_(mGroupNum2), z1z2_(z1z2Coeff), intertype_(interferenceType) {};
00013
00018 std::string Interference::GetInterferenceType() const {
00019     return intertype_;
00020 }
00021
00026 int Interference::GetM1() const {
00027     return m1_;

```

```

00028 }
00029
00034 int Interference::GetM2() const {
00035     return m2_;
00036 }
00037
00042 double Interference::GetZ1Z2() const {
00043     return z1z2_;
00044 }
00045

```

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.](#)

```

00001 #include "JGroup.h"
00002 #include "NucLine.h"
00003
00008 JGroup::JGroup(NucLine nucLine):
00009     pi_(nucLine.levelPi()), j_(nucLine.levelJ()), isinrmatrix_(true) {};
00010
00015 JGroup::JGroup(double j,int pi):
00016     pi_(pi),j_(j),isinrmatrix_(false) {};
00017
00023 bool JGroup::IsInRMatrix() const {
00024     return isinrmatrix_;
00025 }
00026
00032 int JGroup::IsLevel(ALevel level) {
00033     bool b=false;
00034     int c=0;
00035     double tol=1e-3;
00036     while(!b&& c<this->NumLevels())
00037     {
00038         if(this->GetLevel(c+1)->GetE()-tol<=level.GetE() &&
00039             level.GetE() <=this->GetLevel(c+1)->GetE()+tol) b=true;
00040         c++;
00041     }
00042     if(b) return c;
00043     else return 0;
00044 }
00045
00050 int JGroup::GetPi() const {
00051     return pi_;
00052 }
00053
00058 int JGroup::NumLevels() const {
00059     return levels_.size();
00060 }
00061
00066 int JGroup::NumChannels() {
00067     return channels_.size();
00068 }
00069
00075 int JGroup::IsChannel(AChannel channel) {
00076     bool b=false;
00077     int c=0;
00078     while(!b&& c<this->NumChannels()) {
00079         if(channel.GetL()==this->GetChannel(c+1)->GetL() &&
00080             channel.GetS()==this->GetChannel(c+1)->GetS() &&
00081             channel.GetPairNum()==this->GetChannel(c+1)->GetPairNum()) b=true;
00082         c++;
00083     }
00084     if(b) return c;
00085     else return 0;
00086 }
00087

```

```

00092 double JGroup::GetJ() const {
00093     return j_;
00094 }
00095
00100 void JGroup::AddLevel(ALevel level) {
00101     levels_.push_back(level);
00102 }
00103
00108 void JGroup::AddChannel(AChannel channel) {
00109     channels_.push_back(channel);
00110 }
00111
00117 AChannel *JGroup::GetChannel(int channelNum) {
00118     AChannel *b=&channels_[channelNum-1];
00119     return b;
00120 }
00121
00126 ALevel *JGroup::GetLevel(int levelNum) {
00127     ALevel *b=&levels_[levelNum-1];
00128     return b;
00129 }

```

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.](#)

```

00001 #include "KGroup.h"
00002
00007 KGroup::KGroup(double s, double sPrime) :
00008     s_(s), sp_(sPrime) {};
00009
00014 int KGroup::NumMGroups() const {
00015     return mgroups_.size();
00016 }
00017
00022 int KGroup::NumECMGroups() const {
00023     return ec_mgroups_.size();
00024 }
00025
00032 int KGroup::IsMGroup(MGroup mGroup) {
00033     bool b=false;
00034     int c=0;
00035     while(!b&& c<this->NumMGroups())
00036     {
00037         if(mGroup.GetChNum()==this->GetMGroup(c+1)->GetChNum() &&
00038            mGroup.GetChpNum()==this->GetMGroup(c+1)->GetChpNum() &&
00039            mGroup.GetJNum()==this->GetMGroup(c+1)->GetJNum()) b=true;
00040         c++;
00041     }
00042     if(b) return c;
00043     else return 0;
00044 }
00045
00050 double KGroup::GetS() const {
00051     return s_;
00052 }
00053
00058 double KGroup::GetSp() const {
00059     return sp_;
00060 }
00061
00066 void KGroup::AddMGroup(MGroup mGroup) {
00067     mgroups_.push_back(mGroup);
00068 }
00069
00074 void KGroup::AddECMGroup(ECMGroup ecMGroup) {
00075     ec_mgroups_.push_back(ecMGroup);
00076 }

```

```

00077
00082 MGroup *KGroup::GetMGroup(int mGroupNum) {
00083     MGroup *b=&mgroups_[mGroupNum-1];
00084     return b;
00085 }
00090 ECMGroup *KGroup::GetECMGroup(int ecMGroupNum) {
00091     ECMGroup *b=&ec_mgroups_[ecMGroupNum-1];
00092     return b;
00093 }

```

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.](#)

```

00001 #include "KLGroup.h"
00002
00007 KLGroup::KLGroup(int kGroupNum,int lOrder) :
00008     k_(kGroupNum), lorder_(lOrder) {};
00009
00014 int KLGroup::GetK() const {
00015     return k_;
00016 }
00017
00022 int KLGroup::GetLOrder() const {
00023     return lorder_;
00024 }
00025
00030 int KLGroup::NumInterferences() const {
00031     return interferences_.size();
00032 }
00033
00039 int KLGroup::IsInterference(Interference interference) {
00040     bool b=false;
00041     int c=0;
00042     while(!b&& c<this->NumInterferences())
00043     {
00044         if(interference.GetM1()==this->GetInterference(c+1)->GetM1() &&
00045            interference.GetM2()==this->GetInterference(c+1)->GetM2() &&
00046            interference.GetInterferenceType()==this->GetInterference(c+1)->GetInterferenceType()) b=true;
00047         c++;
00048     }
00049     if(b) return c;
00050     else return 0;
00051 }
00052
00057 void KLGroup::AddInterference(Interference interference) {
00058     interferences_.push_back(interference);
00059 }
00060
00065 Interference *KLGroup::GetInterference(int interferenceNum) {
00066     Interference *b=&interferences_[interferenceNum-1];
00067     return b;
00068 }

```

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.](#)

```
00001 #include "MatrixInv.h"
00002 #include <gsl/gsl_linalg.h>
00003 #include <gsl/gsl_matrix_complex_double.h>
00004 #include <iostream>
00005
00011 MatrixInv::MatrixInv(const matrix_c &A) {
00012     inverse_.clear();
00013
00014     gsl_complex x;
00015     gsl_matrix_complex * m = gsl_matrix_complex_alloc (A.size(), A.size());
00016     for(int i=0;i<A.size();i++) {
00017         for(int ii=0;ii<A.size();ii++) {
00018             GSL_SET_COMPLEX(&x,real(A[i][ii]),imag(A[i][ii]));
00019             gsl_matrix_complex_set (m,i,ii,x);
00020         }
00021     }
00022     int psign;
00023     gsl_permutation *p = gsl_permutation_alloc(A.size());
00024     gsl_matrix_complex * mi = gsl_matrix_complex_alloc (A.size(), A.size());
00025     gsl_linalg_complex_LU_decomp(m,p,&psign);
00026     gsl_linalg_complex_LU_invert (m,p,mi);
00027
00028     vector_c AI_row;
00029     for(int i=0;i<A.size();i++) {
00030         inverse_.push_back(AI_row);
00031         for(int ii=0;ii<A.size();ii++) {
00032             x=gsl_matrix_complex_get(mi,i,ii);
00033             complex inv (GSL_REAL(x),GSL_IMAG(x));
00034             inverse_[i].push_back(inv);
00035         }
00036     }
00037
00038     gsl_matrix_complex_free(m);
00039     gsl_matrix_complex_free(mi);
00040     gsl_permutation_free(p);
00041 }
```

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.](#)

```
00001 #include "MGroup.h"
00002
00007 MGroup::MGroup(int jGroupNum, int channelNum, int channelPrimeNum) :
00008     jnum_(jGroupNum), ch_(channelNum), chp_(channelPrimeNum), statspinfactor_(0.0) {
00009 }
00010
00015 int MGroup::GetChNum() const {
00016     return ch_;
00017 }
00018
00023 int MGroup::GetChpNum() const {
00024     return chp_;
00025 }
00026
00031 int MGroup::GetJNum() const {
00032     return jnum_;
00033 }
00034
00039 double MGroup::GetStatSpinFactor() const {
00040     return statspinfactor_;
00041 }
00042
00047 void MGroup::SetStatSpinFactor(double spinFactor) {
00048     statspinfactor_=spinFactor;
00049 }
```

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.](#)

```
00001 #include "NFIntegral.h"
00002 #include <gsl/gsl_integration.h>
00003
00004 double NFIntegral::Integrand(double x, void * p) {
00005     Params *params = (Params*)p;
00006     WhitFunc *whitFunc=(params->whitFunc);
00007     int lfValue=(params->lfValue);
00008     double bindingEnergy=(params->bindingEnergy);
00009     double whitChRadSquaredValue=(params->whitChRadSquaredValue);
00010
00011     double whit=whitFunc->operator() (lfValue,x,bindingEnergy);
00012     return pow(whit,2.0)/whitChRadSquaredValue;
00013 }
00014
00015 double NFIntegral::operator() (int lf,double levelEnergy) {
00016     params_.lfValue = lf;
00017     params_.bindingEnergy = fabs(levelEnergy - totalSepE());
00018     params_.whitChRadSquaredValue =
00019     pow(params_.whitFunc->operator() (lf,chanRad()),params_.bindingEnergy),2.0);
00020
00021     gsl_integration_workspace * w
00022     = gsl_integration_workspace_alloc (1000);
00023
00024     gsl_function F;
00025     F.function = &Integrand;
00026     F.params= &params_;
00027
00028     double intresult,interror;
00029
00030     gsl_integration_qagiu(&F,chanRad(),0.0,1e-4,1000,w,&intresult,&interror);
00031
00032     gsl_integration_workspace_free (w);
00033
00034     return intresult;
00035 }
```

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.](#)

```
00001 #include "NucLine.h"
00002 #include "PPair.h"
00003 #include <assert.h>
00004
```



```

00009 PPair::PPair(NucLine nucLine)
00010 {
00011     pair_z_[0]=nucLine.z1();
00012     pair_z_[1]=nucLine.z2();
00013     pair_m_[0]=nucLine.m1();
00014     pair_m_[1]=nucLine.m2();
00015     pair_pi_[0]=nucLine.pi1();
00016     pair_pi_[1]=nucLine.pi2();
00017     pair_g_[0]=nucLine.g1();
00018     pair_g_[1]=nucLine.g2();
00019     pair_j_[0]=nucLine.j1();
00020     pair_j_[1]=nucLine.j2();
00021     pair_ex_e_=nucLine.e2();
00022     pair_sep_e_=nucLine.sepE();
00023     pair_ch_rad_=nucLine.chRad();
00024     pair_ptype_=nucLine.pType();
00025     pair_key_=nucLine.ir();
00026     red_mass_=nucLine.m1()*nucLine.m2()/(nucLine.m1()+nucLine.m2());
00027     ili2factor_=1.0/(2.*nucLine.j1()+1.0)/(2.*nucLine.j2()+1.0);
00028     entrance_=false;
00029     ec_entrance_=false;
00030 }
00031
00036 bool PPair::IsEntrance() const {
00037     return entrance_;
00038 }
00039
00044 int PPair::GetZ(int particle) const {
00045     return pair_z_[particle-1];
00046 }
00047
00052 int PPair::GetPi(int particle) const {
00053     return pair_pi_[particle-1];
00054 }
00055
00060 int PPair::GetPType() const {
00061     return pair_ptype_;
00062 }
00063
00069 int PPair::NumDecays() const {
00070     return decays_.size();
00071 }
00072
00078 int PPair::IsDecay(Decay decay) {
00079     bool b=false;
00080     int c=0;
00081     while(!b&& c<this->NumDecays())
00082     {
00083         if(decay.GetPairNum()==this->GetDecay(c+1)->GetPairNum()) b=true;
00084         c++;
00085     }
00086     if(b) return c;
00087     else return 0;
00088 }
00089
00095 int PPair::IsDecay(int pairNum) {
00096     bool b=false;
00097     int c=0;
00098     while(!b&& c<this->NumDecays())
00099     {
00100         if(pairNum==this->GetDecay(c+1)->GetPairNum()) b=true;
00101         c++;
00102     }
00103     if(b) return c;
00104     else return 0;
00105 }
00106
00111 int PPair::GetPairKey() const {
00112     return pair_key_;
00113 }
00114
00119 double PPair::GetM(int particle) const {
00120     return pair_m_[particle-1];
00121 }
00122
00127 double PPair::GetG(int particle) const {
00128     return pair_g_[particle-1];
00129 }
00130
00135 double PPair::GetJ(int particle) const {
00136     return pair_j_[particle-1];
00137 }
00138
00143 double PPair::GetExE() const {
00144     return pair_ex_e_;
00145 }
00146

```

```

00151 double PPair::GetSepE() const {
00152     return pair_sep_e_;
00153 }
00154
00159 double PPair::GetChRad() const {
00160     return pair_ch_rad_;
00161 }
00162
00167 double PPair::GetRedMass() const {
00168     return red_mass_;
00169 }
00170
00175 double PPair::GetIli2Factor() const {
00176     return ili2factor_;
00177 }
00178
00183 void PPair::AddDecay(Decay decay) {
00184     decays_.push_back(decay);
00185 }
00186
00191 void PPair::SetEntrance() {
00192     entrance_=true;
00193 }
00194
00199 Decay *PPair::GetDecay(int decayNum) {
00200     Decay *b=&decays_[decayNum-1];
00201     return b;
00202 }
00203

```

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*
```

```

00063     sqrt(8.0/pi/compound->GetPair(compound->GetPairNumFromKey(entranceKey))->GetRedMass()/uconv)/
00064     pow(boltzConst*temperature,1.5)*result;
00065
00066     gsl_integration_workspace_free (w);
00067
00068     return rate;
00069 }
00070
00071 ReactionRate::ReactionRate(CNuc *compound, const vector_r &params,
00072     const Config &configure, int entranceKey, int exitKey) :
00073     configure_(configure) {
00074     compound_=compound;
00075     compound_->FillCompoundFromParams(params);
00076     entrance_key_=entranceKey;
00077     exit_key_=exitKey;
00078 }
00079
00084 void ReactionRate::CalculateRates() {
00085     int numSteps = (configure().rateParams.tempStep!=0.) ?
00086
00087     int ((configure().rateParams.maxTemp-configure().rateParams.minTemp)/configure().rateParams.tempStep)+1
00088     : 1;
00087     configure().outStream << std::setw(0) << "\t[
00088     0%";configure().outStream.flush();
00088     int pointIndex=0;
00089     time_t startTime = time(NULL);
00090     #pragma omp parallel for
00091     for(int i=0;i<numSteps;++i) {
00092         CNuc* localCompound = compound()->Clone();
00093         int localEntranceKey = entranceKey();
00094         int localExitKey = exitKey();
00095         const Config localConfigure = configure();
00096         double temp = localConfigure.rateParams.minTemp+i*localConfigure.rateParams.tempStep;
00097
00098         double
00099         rate=gsl_reactionrate_integration(temp,localCompound,localConfigure,localEntranceKey,localExitKey);
00100         rates_.push_back(RateData(temp,rate));
00101         delete localCompound;
00102         ++pointIndex;
00103         if(diffTime(time(NULL),startTime)>0.25) {
00104             startTime=time(NULL);
00105             std::string progress="";
00106             double percent=0.;
00107             for(int j = 1;j<=25;j++) {
00108                 if(pointIndex>=percent*numSteps&&percent<1.) {
00109                     percent+=0.04;
00110                     progress+="'*';
00111                 } else progress+="' ";
00112                 } progress+="] ";
00113                 localConfigure.outStream << std::setw(0) << "\r\t" << progress << percent*100 <<
00114                 '%';localConfigure.outStream.flush();
00115             }
00116             configure().outStream << std::setw(0) << "\r\t[*****] 100%" << std::endl;
00117             std::sort(rates_.begin(),rates_.end());
00118         }
00119     }
00120
00123 void ReactionRate::CalculateFileRates() {
00124     std::ifstream inFile(configure().rateParams.temperatureFile.c_str());
00125     if(inFile) {
00126         while(!inFile.eof()) {
00127             std::string line;
00128             getline(inFile,line);
00129             if(!inFile.eof()) {
00130                 double temp = 0.;
00131                 std::istringstream stm;
00132                 stm.str(line);
00133                 if(stm > temp) {
00134                     rates_.push_back(RateData(temp,0.));
00135                 }
00136             }
00137         }
00138         inFile.close();
00139         configure().outStream << std::setw(0) << "\t[
00140         0%";configure().outStream.flush();
00141         int pointIndex=0;
00142         int numSteps = rates_.size();
00143         time_t startTime = time(NULL);
00144         #pragma omp parallel for
00145         for(int i=0;i<numSteps;++i) {
00146             CNuc* localCompound = compound()->Clone();
00147             int localEntranceKey = entranceKey();
00148             int localExitKey = exitKey();
00149             const Config localConfigure = configure();
00150
00151             rates_[i].rate=gsl_reactionrate_integration(rates_[i].temperature,localCompound,localConfigure,localEntranceKey,localExitKey);

```

```

00151         delete localCompound;
00152         ++pointIndex;
00153         if(diffTime(time(NULL),startTime)>0.25) {
00154             startTime=time(NULL);
00155             std::string progress="[";
00156             double percent=0.;
00157             for(int j = 1;j<=25;j++) {
00158                 if(pointIndex>=percent*numSteps&&percent<1.) {
00159                     percent+=0.04;
00160                     progress+='*';
00161                 } else progress+=' ';
00162             } progress+="] ";
00163             localConfigure.outStream << std::setw(0) << "\r\t" << progress << percent*100 <<
00164             '\%';localConfigure.outStream.flush();
00165         }
00166         configure().outStream << std::setw(0) << "\r\t[*****] 100%" << std::endl;
00167     } else configure().outStream << "Couldn't open temperature file." << std::endl;
00168 }
00169
00170 void ReactionRate::WriteRates() {
00171     std::string outputfile=configure().outputdir+"reactionrates.out";
00172     std::ofstream out;
00173     out.open(outputfile.c_str());
00174     if(out) {
00175         out << std::setw(20) << "T9" << std::setw(20) << "Rate" << std::endl;
00176         for(int i=0;i<rates_.size();i++) {
00177             out << std::setw(20) << rates_[i].temperature << std::setw(20) << rates_[i].rate << std::endl;
00178         }
00179         out.flush();
00180         out.close();
00181     } else configure().outStream << "Could not write reaction rate file." << std::endl;
00182 }

```

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.](#)

```

00001 #include "EPoint.h"
00002 #include "CNuc.h"
00003 #include "MatrixInv.h"
00004 #include "RMatrixFunc.h"
00005 #include <assert.h>
00006
00011 RMatrixFunc::RMatrixFunc(CNuc* compound, const Config &configure) :
00012     compound_(compound), configure_(configure) {}
00013
00018 complex RMatrixFunc::GetRMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum) const {
00019     return r_matrices_[jGroupNum-1][channelNum-1][channelPrimeNum-1];
00020 }
00021
00026 complex RMatrixFunc::GetRLMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum) const {
00027     return rl_matrices_[jGroupNum-1][channelNum-1][channelPrimeNum-1];
00028 }
00029
00034 complex RMatrixFunc::GetRLInvMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum) const {
00035     return rl_inv_matrices_[jGroupNum-1][channelNum-1][channelPrimeNum-1];
00036 }
00037
00042 complex RMatrixFunc::GetRLInvRMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum) const
00043 {
00044     return rl_inv_r_matrices_[jGroupNum-1][channelNum-1][channelPrimeNum-1];
00045 }

```

```

00044 }
00045
00050 matrix_c *RMatrixFunc::GetJSpecRLMatrix(int jGroupNum) {
00051     matrix_c *b=&rl_matrices_[jGroupNum-1];
00052     return b;
00053 }
00054
00059 void RMatrixFunc::ClearMatrices() {
00060     r_matrices_.clear();
00061     rl_matrices_.clear();
00062     rl_inv_matrices_.clear();
00063     rl_inv_r_matrices_.clear();
00064     tmatrix_.clear();
00065     ec_tmatrix_.clear();
00066 }
00067
00072 void RMatrixFunc::FillMatrices(EPoint *point) {
00073     double inEnergy;
00074     if(compound()->
00075         GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->
00076         GetPType()==20)
00077         inEnergy=point->GetCMEnergy()+
00078         compound()->
00079         GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->
00080         GetSepE()+
00081         compound()->
00082         GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->
00083         GetExE();
00084     else inEnergy=point->GetCMEnergy()+
00085         compound()->GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->GetSepE()+
00086         compound()->GetPair(compound()->GetPairNumFromKey(point->GetEntranceKey()))->GetExE();
00087     for(int j=1; j<=compound()->NumJGroups(); j++) {
00088         if(compound()->GetJGroup(j)->IsInRMatrix()) {
00089             if(configure().paramMask & Config::USE_BRUNE_FORMALISM) {
00090                 matrix_c qMatrixInverse;
00091                 for(int la=1; la<=compound()->GetJGroup(j)->NumLevels(); la++) {
00092                     if(!compound()->GetJGroup(j)->GetLevel(la)->IsInRMatrix()) continue;
00093                     ALevel *level=compound()->GetJGroup(j)->GetLevel(la);
00094                     vector_c tempVector;
00095                     for(int lap=1; lap<=compound()->GetJGroup(j)->NumLevels(); lap++) {
00096                         if(!compound()->GetJGroup(j)->GetLevel(lap)->IsInRMatrix()) continue;
00097                         ALevel *levelp=compound()->GetJGroup(j)->GetLevel(lap);
00098                         complex sum = (la==lap) ? complex(level->GetFitE()-inEnergy,0.0) : complex(0.0,0.0);
00099                         for(int ch=1; ch<=compound()->GetJGroup(j)->NumChannels(); ch++) {
00100                             if(compound()->GetJGroup(j)->GetChannel(ch)->GetRadType()!='P') continue;
00101                             double gammaCh=level->GetFitGamma(ch);
00102                             double gammaChp=levelp->GetFitGamma(ch);
00103                             complex channelShift=point->GetLoElement(j,ch)
00104                             +complex(compound()->GetJGroup(j)->GetChannel(ch)->GetBoundaryCondition(),
00105                                     -1.*pow(point->GetSqrtPenetrability(j,ch),2.));
00106                             sum-=gammaCh*gammaChp*channelShift;
00107                             if(la==lap) sum+=gammaCh*gammaChp*level->GetShiftFunction(ch);
00108                             else sum+=gammaCh*gammaChp*
00109                                 (level->GetShiftFunction(ch)*(inEnergy-levelp->GetFitE())
00110                                  -levelp->GetShiftFunction(ch)*(inEnergy-level->GetFitE()))/
00111                                 (level->GetFitE()-levelp->GetFitE());
00112                         }
00113                         tempVector.push_back(sum);
00114                     }
00115                     qMatrixInverse.push_back(tempVector);
00116                 }
00117                 MatrixInv matrixInv(qMatrixInverse);
00118                 for(int ch=1; ch<=compound()->GetJGroup(j)->NumChannels(); ch++) {
00119                     for(int chp=1; chp<=compound()->GetJGroup(j)->NumChannels(); chp++) {
00120                         complex rTemp(0.0,0.0);
00121                         for(int la=1; la<=compound()->GetJGroup(j)->NumLevels(); la++) {
00122                             if(!compound()->GetJGroup(j)->GetLevel(la)->IsInRMatrix()) continue;
00123                             complex temp(0.0,0.0);
00124                             for(int lap=1; lap<=compound()->GetJGroup(j)->NumLevels(); lap++) {
00125                                 if(!compound()->GetJGroup(j)->GetLevel(lap)->IsInRMatrix()) continue;
00126                                 temp+=matrixInv.inverse()[la-1][lap-1]*compound()->GetJGroup(j)->GetLevel(lap)->GetFitGamma(chp);
00127                             }
00128                             rTemp+=temp*compound()->GetJGroup(j)->GetLevel(la)->GetFitGamma(ch);
00129                         }
00130                         this->AddRMatrixElement(j,ch,chp,rTemp);
00131                         double tempPene = 0.;
00132                         if(compound()->GetJGroup(j)->GetChannel(chp)->GetRadType()=='P')
00133                             tempPene=pow(point->GetSqrtPenetrability(j,chp),2.);
00134                         if(ch==chp) this->AddRLMatrixElement(j,ch,chp,1.-complex(0.,1.)*rTemp*tempPene);
00135                         else this->AddRLMatrixElement(j,ch,chp,-complex(0.,1.)*rTemp*tempPene);
00136                     }
00137                 }
00138             } else {
00139                 for(int ch=1; ch<=compound()->GetJGroup(j)->NumChannels(); ch++) {
00140                     for(int chp=1; chp<=compound()->GetJGroup(j)->NumChannels(); chp++) {
00141                         complex sum(0.0,0.0);

```

```

00142         for(int la=1; la<=compound()->GetJGroup(j)->NumLevels(); la++) {
00143             if(compound()->GetJGroup(j)->GetLevel(la)->IsInRMatrix()) {
00144                 ALevel *level=compound()->GetJGroup(j)->GetLevel(la);
00145                 double gammaCh=level->GetFitGamma(ch);
00146                 double gammaChp=level->GetFitGamma(chp);
00147                 double resenergy=level->GetFitE();
00148                 double gammaSum=0.;
00149                 if(configure().paramMask & Config::USE_RMC_FORMALISM)
00150                     for(int chpp=1; chpp<=compound()->GetJGroup(j)->NumChannels(); chpp++)
00151                         if(compound()->GetJGroup(j)->GetChannel(chpp)->GetRadType()=='M' ||
00152                            compound()->GetJGroup(j)->GetChannel(chpp)->GetRadType()=='E')
00153                             gammaSum+=pow(compound()->GetJGroup(j)->GetLevel(la)->GetFitGamma(chpp), 2.0);
00154                 sum+=gammaCh*gammaChp/(resenergy-inEnergy-complex(0.0,1.0)*gammaSum);
00155             }
00156         }
00157         this->AddRMatrixElement(j, ch, chp, sum);
00158         complex loElement =point->GetLoElement(j, chp);
00159         if(ch==chp) this->AddRLMatrixElement(j, ch, chp, 1.0-sum*loElement);
00160         else this->AddRLMatrixElement(j, ch, chp, -sum*loElement);
00161     }
00162 }
00163 }
00164 }
00165 }
00166 }
00167
00168
00173 void RMatrixFunc::InvertMatrices() {
00174     for(int j=1; j<=compound()->NumJGroups(); j++) {
00175         if(compound()->GetJGroup(j)->IsInRMatrix()) {
00176             matrix_c *theRLMatrix = this->GetJSpecRLMatrix(j);
00177             MatrixInv matrixInv(*theRLMatrix);
00178             this->AddRLInvMatrix(matrixInv.inverse());
00179             for(int ch=1; ch<=compound()->GetJGroup(j)->NumChannels(); ch++) {
00180                 for(int chp=1; chp<=compound()->GetJGroup(j)->NumChannels(); chp++) {
00181                     complex rlinvrElement(0., 0.);
00182                     for(int chpp=1; chpp<=compound()->GetJGroup(j)->NumChannels(); chpp++) {
00183                         rlinvrElement+=this->GetRLInvMatrixElement(j, ch, chpp)*
00184                             this->GetRMatrixElement(j, chpp, chp);
00185                     }
00186                     this->AddRLInvRMatrixElement(j, ch, chp, rlinvrElement);
00187                 }
00188             }
00189         }
00190     }
00191 }
00192
00197 void RMatrixFunc::CalculateTMatrix(EPoint *point) {
00198     int aa=compound()->GetPairNumFromKey(point->GetEntranceKey());
00199     int irEnd;
00200     int irStart;
00201     bool isRMC=false;
00202     if((configure().paramMask & Config::USE_RMC_FORMALISM) &&
00203        compound()->GetPair(compound()->GetPairNumFromKey(point->GetExitKey()))->GetPType()==10) {
00204         irStart=1;
00205         irEnd=compound()->GetPair(aa)->NumDecays();
00206         isRMC=true;
00207     } else {
00208         irStart=0;
00209         while(irStart<compound()->GetPair(aa)->NumDecays()) {
00210             irStart++;
00211             if(compound()->GetPair(aa)->GetDecay(irStart)->GetPairNum()==
00212                compound()->GetPairNumFromKey(point->GetExitKey())) break;
00213         }
00214         irEnd=irStart;
00215     }
00216     for(int ir=irStart; ir<=irEnd; ir++) {
00217         Decay *theDecay=compound()->GetPair(aa)->GetDecay(ir);
00218         for(int k=1; k<=theDecay->NumKGroups(); k++) {
00219             for(int m=1; m<=theDecay->GetKGroup(k)->NumMGroups(); m++) {
00220                 MGroup *theMGroup=theDecay->GetKGroup(k)->GetMGroup(m);
00221                 JGroup *theJGroup=compound()->GetJGroup(theMGroup->GetJNum());
00222                 AChannel *entranceChannel=theJGroup->GetChannel(theMGroup->GetChNum());
00223                 AChannel *exitChannel=theJGroup->GetChannel(theMGroup->GetChpNum());
00224                 complex uphase=point->GetExpCoulombPhase(theMGroup->GetJNum(), theMGroup->GetChNum())*
00225                     point->GetExpHardSpherePhase(theMGroup->GetJNum(), theMGroup->GetChNum())*
00226                     point->GetExpCoulombPhase(theMGroup->GetJNum(), theMGroup->GetChpNum())*
00227                     point->GetExpHardSpherePhase(theMGroup->GetJNum(), theMGroup->GetChpNum());
00228                 complex umatrix=2.0*std::complex<double>(0.0,1.0)*
00229                     point->GetSqrtPenetrability(theMGroup->GetJNum(), theMGroup->GetChNum())*
00230                     point->GetSqrtPenetrability(theMGroup->GetJNum(), theMGroup->GetChpNum())*
00231                     this->GetRLInvRMatrixElement(theMGroup->GetJNum(),
00232                                             theMGroup->GetChpNum(),
00233                                             theMGroup->GetChNum());
00234                 complex tphase=point->GetExpCoulombPhase(theMGroup->GetJNum(), theMGroup->GetChNum())*
00235                     point->GetExpCoulombPhase(theMGroup->GetJNum(), theMGroup->GetChNum());
00236                 complex tmatrix;

```

```

00237     if (isRMC) this->AddTMatrixElement(k,m,complex(0.0,-1.0)*umatrix,ir);
00238     else {
00239         if (theMGroup->GetChNum()==theMGroup->GetChpNum()) {
00240             tmatrix=tphase-uphase*(1.0+umatrix);
00241         } else tmatrix=-uphase*umatrix;
00242         this->AddTMatrixElement(k,m,tmatrix);
00243     }
00244 }
00245 for (int m=1;m<=theDecay->GetKGroup(k)->NumECMGroups();m++) {
00246     ECMGroup *theECMGroup=theDecay->GetKGroup(k)->GetECMGroup(m);
00247     ALevel *finalLevel=compound()->GetJGroup(theECMGroup->GetJGroupNum())
00248         ->GetLevel(theECMGroup->GetLevelNum());
00249     double ecNormParam=finalLevel->GetFitGamma(theECMGroup->GetFinalChannel())*
00250         finalLevel->GetSqrtNFFactor()*finalLevel->GetECConversionFactor(theECMGroup->GetFinalChannel());
00251     complex tmatrix=ecNormParam*point->GetECAmplitude(k,m);
00252     if (theECMGroup->IsChannelCapture()) {
00253         MGroup *chanMGroup=compound()->GetPair(aa)->GetDecay(theECMGroup->GetChanCapDecay())
00254             ->GetKGroup(theECMGroup->GetChanCapKGroup())->GetMGroup(theECMGroup->GetChanCapMGroup());
00255         AChannel *chanEntranceChannel=compound()->GetJGroup(chanMGroup->GetJNum())
00256             ->GetChannel(chanMGroup->GetChNum());
00257         AChannel *chanExitChannel=compound()->GetJGroup(chanMGroup->GetJNum())
00258             ->GetChannel(chanMGroup->GetChpNum());
00259         complex umatrix=2.0*std::complex<double>(0.0,1.0)*
00260             point->GetSqrtPenetrability(chanMGroup->GetJNum(),chanMGroup->GetChNum())*
00261             this->GetRLInvRMatrixElement(chanMGroup->GetJNum(),
00262                 chanMGroup->GetChpNum(),
00263                 chanMGroup->GetChNum());
00264         tmatrix=tmatrix*umatrix;
00265     }
00266     this->AddECTMatrixElement(k,m,tmatrix);
00267 }
00268 }
00269 }
00270 }
00271 }
00276 void RMatrixFunc::AddRMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum, complex
matrixElement) {
00277     matrix_c e;
00278     vector_c f;
00279     while (jGroupNum>r_matrices_.size()) r_matrices_.push_back(e);
00280     while (channelNum>r_matrices_[jGroupNum-1].size()) r_matrices_[jGroupNum-1].push_back(f);
00281     r_matrices_[jGroupNum-1][channelNum-1].push_back(matrixElement);
00282     assert(channelPrimeNum==r_matrices_[jGroupNum-1][channelNum-1].size());
00283 }
00284
00289 void RMatrixFunc::AddRLMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum, complex
matrixElement) {
00290     matrix_c e;
00291     vector_c f;
00292     while (jGroupNum>rl_matrices_.size()) rl_matrices_.push_back(e);
00293     while (channelNum>rl_matrices_[jGroupNum-1].size()) rl_matrices_[jGroupNum-1].push_back(f);
00294     rl_matrices_[jGroupNum-1][channelNum-1].push_back(matrixElement);
00295     assert(channelPrimeNum==rl_matrices_[jGroupNum-1][channelNum-1].size());
00296 }
00297
00302 void RMatrixFunc::AddRLInvRMatrixElement(int jGroupNum, int channelNum, int channelPrimeNum, complex
matrixElement) {
00303     matrix_c e;
00304     vector_c f;
00305     while (jGroupNum>rl_inv_r_matrices_.size()) rl_inv_r_matrices_.push_back(e);
00306     while (channelNum>rl_inv_r_matrices_[jGroupNum-1].size())
        rl_inv_r_matrices_[jGroupNum-1].push_back(f);
00307     rl_inv_r_matrices_[jGroupNum-1][channelNum-1].push_back(matrixElement);
00308     assert(channelPrimeNum==rl_inv_r_matrices_[jGroupNum-1][channelNum-1].size());
00309 }
00310
00315 void RMatrixFunc::AddRLInvMatrix(matrix_c matrix) {
00316     rl_inv_matrices_.push_back(matrix);
00317 }

```

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.](#)

```
00001 #include "ShftFunc.h"
00002 #include <math.h>
00003 #include <gsl/gsl_deriv.h>
00004
00005 double ShftFunc::thisShftFunc(double x, void *p) {
00006     ShftFunc *shift = (ShftFunc*)p;
00007     int l= shift->params_.lValue;
00008
00009     return shift->operator()(l,x);
00010 }
00011
00012 double ShftFunc::theWhitFunc(double x, void * p) {
00013     Params *params =(Params *)p;
00014     int l = (params->lValue);
00015     double bindingenergy = (params->bindingEnergy);
00016     WhitFunc *whitFunc = (params->whitFunc);
00017
00018     return whitFunc->operator()(l,x,bindingenergy);
00019 }
00020
00021 double ShftFunc::operator()(int l, double energy) {
00022     params_.bindingEnergy=fabs(energy-totalSepE());
00023     params_.lValue=l;
00024
00025     double result;
00026     double error;
00027
00028     gsl_function F;
00029     F.function = &theWhitFunc;
00030     F.params = &params_;
00031
00032     gsl_deriv_central (&F, radius(), 1e-4, &result, &error);
00033
00034     return radius()*result/theWhitFunc(radius(),&params_);
00035 }
00036
00037 double ShftFunc::EnergyDerivative(int l, double energy) {
00038
00039     double result;
00040     double error;
00041
00042     params_.lValue = l;
00043
00044     gsl_function F;
00045     F.function = &thisShftFunc;
00046     F.params = this;
00047
00048     gsl_deriv_central (&F, energy, 1e-6, &result, &error);
00049
00050     return result;
00051 }
```

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.](#)

```
00001 #include "TargetEffect.h"
00002 #include <sstream>
00003 #include <iostream>
00004
```

```

00011 TargetEffect::TargetEffect(std::istream &stream, const Config& configure) {
00012     int isActive;
00013     std::string segmentList;
00014     int numIntegrationPoints;
00015     int isConvolution;
00016     double sigma;
00017     int isTargetIntegration;
00018     double density;
00019     std::string stoppingPowerEq;
00020     int numParameters;
00021     vector_r parameters;
00022     int isQCoefficients;
00023     int numQCoefficients;
00024     vector_r qCoefficients;
00025     stream » isActive » segmentList » numIntegrationPoints » isConvolution
00026         » sigma » isTargetIntegration » density » stoppingPowerEq
00027         » numParameters;
00028     if(!stream.eof()) {
00029         for(int i=0; i<numParameters; i++) {
00030             double tempParameter;
00031             stream » tempParameter;
00032             parameters.push_back(tempParameter);
00033         }
00034         stream » isQCoefficients » numQCoefficients;
00035         for(int i=0; i<numQCoefficients; i++) {
00036             double tempQCoefficient;
00037             stream » tempQCoefficient;
00038             qCoefficients.push_back(tempQCoefficient);
00039         }
00040         isQCoefficients_ = (isQCoefficients==1) ? true : false;
00041         qCoefficients_ = qCoefficients;
00042         size_t found=0;
00043         while(found!=std::string::npos) {
00044             found=segmentList.find('"');
00045             if(found!=std::string::npos) segmentList.erase(found,1);
00046         }
00047         found=0;
00048         while(found!=std::string::npos) {
00049             found=stoppingPowerEq.find('"');
00050             if(found!=std::string::npos) stoppingPowerEq.erase(found,1);
00051         }
00052         if(isActive==1) isActive_=true;
00053         else isActive_=false;
00054         segmentsList_=segmentList;
00055         numIntegrationPoints_=numIntegrationPoints;
00056         if(isConvolution==1) isConvolution_=true;
00057         else isConvolution_=false;
00058         sigma_=sigma;
00059         if(isTargetIntegration==1) isTargetIntegration_=true;
00060         else isTargetIntegration_=false;
00061         density_=density;
00062         if(isTargetIntegration_) {
00063             stoppingPowerEq_.Initialize(stoppingPowerEq, numParameters, configure);
00064             for(int i=0; i<numParameters; i++) {
00065                 stoppingPowerEq_.SetParameter(i, parameters[i], configure);
00066             }
00067         }
00068     }
00069 }
00070
00076 bool TargetEffect::IsActive() const {
00077     return isActive_;
00078 }
00079
00085 bool TargetEffect::IsConvolution() const {
00086     return isConvolution_;
00087 }
00088
00094 bool TargetEffect::IsTargetIntegration() const {
00095     return isTargetIntegration_;
00096 }
00097
00103 bool TargetEffect::IsQCoefficients() const {
00104     return isQCoefficients_;
00105 }
00106
00112 int TargetEffect::NumSubPoints() const {
00113     return numIntegrationPoints_;
00114 }
00115
00121 int TargetEffect::NumQCoefficients() const {
00122     return qCoefficients_.size();
00123 }
00124
00129 double TargetEffect::GetSigma() const {
00130     return sigma_;
00131 }

```

```

00132
00138 double TargetEffect::GetDensity() const {
00139     return density_;
00140 }
00141
00147 double TargetEffect::TargetThickness(double energy, const Config& configure) {
00148     return this->GetStoppingPowerEq()->Evaluate(configure,energy)*this->GetDensity();
00149 }
00150
00155 double TargetEffect::GetQCoefficient(int order) const {
00156     return (qCoefficients_.size()>order) ? qCoefficients_[order] : 1.;
00157 }
00158
00163 void TargetEffect::SetSigma(double sigma) {
00164     sigma_=sigma;
00165 }
00166
00171 void TargetEffect::SetNumSubPoints(int numPoints) {
00172     numIntegrationPoints_=numPoints;
00173 }
00174
00181 std::vector<int> TargetEffect::GetSegmentsList() const {
00182     std::vector<int> tempList;
00183     int i=0;
00184     int lastSegNum=0;
00185     bool inclusive=false;
00186     while(i<segmentsList_.length()) {
00187         if(segmentsList_[i]>='0'&&segmentsList_[i]<='9') {
00188             std::string tempString;
00189             while(segmentsList_[i]!=','&&segmentsList_[i]!='-'&&
00190                 i<segmentsList_.length()) {
00191                 tempString+=segmentsList_[i];
00192                 i++;
00193             }
00194             std::istringstream stm;
00195             stm.str(tempString);
00196             int tempSegNum;stm>>tempSegNum;
00197             if(inclusive==true) for(int j=lastSegNum+1;j<=tempSegNum;j++)
00198                 tempList.push_back(j);
00199             else tempList.push_back(tempSegNum);
00200             lastSegNum=tempSegNum;
00201         }
00202         if(segmentsList_[i]=='-') inclusive=true;
00203         else inclusive =false;
00204         i++;
00205     }
00206     return tempList;
00207 }
00208
00214 Equation *TargetEffect::GetStoppingPowerEq() {
00215     Equation *tempEquation;
00216     tempEquation=&stoppingPowerEq_;
00217     return tempEquation;
00218 }
00219
00225 double TargetEffect::GetConvolutionFactor(double energy, double centroid) const {
00226     double sigma=this->GetSigma();
00227     return pow(2.*pi,-0.5)/sigma*exp(-pow(energy-centroid,2.0)/2.0/pow(sigma,2.0));
00228 }

```


Index

/Users/kuba/Desktop/R-Matrix/AZURE2/README.md, 488

/Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/complex_functions.h, 275, 283

/Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/cwfcouplers.h, 285

/Users/kuba/Desktop/R-Matrix/AZURE2/coul/include/ode_integrators.h, 288

/Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/complex_functions.cpp, 289, 291

/Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/cwfcouplers.cpp, 296

/Users/kuba/Desktop/R-Matrix/AZURE2/coul/src/ode_integrators.cpp, 315

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZURE2MainThread.h, 323

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZURE2Pairs.h, 324, 325

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AZURE2Setup.h, 326

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AboutAZURE2Dialog.h, 318

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddLevelDialog.h, 318, 319

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddPairDialog.h, 319

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddSegDataDialog.h, 320

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddSegTestDialog.h, 321

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/AddTargetIntDialog.h, 322

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelDetails.h, 328

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChannelModel.h, 328, 329

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ChooseFileButton.h, 329

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditChannelDialog.h, 330

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditDirDialog.h, 331

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/EditOptionsDialog.h, 331, 332

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/ElementMarsh.h, 332

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/FilteredTextEditor.h, 334

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/InfoDialog.h, 335

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsHeaderView.h, 335, 336

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsModel.h, 336

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/LevelsTab.h, 337

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsModel.h, 338, 339

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PairsTab.h, 339, 340

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/PlotTab.h, 340, 341

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RichTextDelegate.h, 342

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/RunTab.h, 342

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsDataModel.h, 343, 344

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTab.h, 344, 345

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/SegmentsTestModel.h, 346

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntModel.h, 347

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TargetIntTab.h, 348

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/include/TextEditBuffer.h, 349

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZUREPlot.cpp, 362, 363

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AZURESetup.cpp, 367, 369

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AboutAZURE2Dialog.cpp, 350

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddLevelDialog.cpp, 351

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddPairDialog.cpp, 351, 352

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddSegDataDialog.cpp, 354

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddSegTestDialog.cpp, 357

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/AddTargetIntDialog.cpp, 359

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChannelDetails.cpp, 381

/Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChannelsModel.cpp, 382
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/ChooseFileDialog.cpp, 385
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditChecksDialog.cpp, 385
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditDirsDialog.cpp, 386, 387
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/EditOptionsDialog.cpp, 387
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InTabDocs.cpp, 389, 390
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/InfoDialog.cpp, 388
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/LevelsModel.cpp, 392
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/LevelsTab.cpp, 394
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PairsModel.cpp, 405
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PairsTab.cpp, 410
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/PlotTab.cpp, 415
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RichTextDialog.cpp, 418
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/RunTab.cpp, 419
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsDataModel.cpp, 422, 423
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTab.cpp, 426
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/SegmentsTableModel.cpp, 437
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntModel.cpp, 440
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/TargetIntTab.cpp, 442
 /Users/kuba/Desktop/R-Matrix/AZURE2/gui/src/main.cpp, 404, 405
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AChannel.h, 447
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ALevel.h, 448
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AMatrixFunc.h, 449
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZURECalc.h, 450
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREFB.h, 451
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREMain.h, 452
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREOutput.h, 452, 453
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AZUREPairs.h, 453
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/AngCoeff.h, 449, 450
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/CNuc.h, 454
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Config.h, 455
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Constants.h, 456, 460
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/CoulFunc.h, 460
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/DataLine.h, 461
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Decay.h, 462
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ECIntegral.h, 462, 463
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ECMGroup.h, 463, 464
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EData.h, 464, 465
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EDataIterator.h, 465, 466
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EPoint.h, 468
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ESegment.h, 471
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EffectiveCharge.h, 467
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/EigenFunc.h, 467
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Equation.h, 470
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ExtrapLine.h, 472, 473
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/GSLEException.h, 474, 475
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/GenMatrixFunc.h, 473, 474
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/IntegratedFermiFunc.h, 475
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/Interference.h, 476
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/JGroup.h, 476, 477
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/KGroup.h, 477
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/KLGroup.h, 478
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/MGroup.h, 479
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/MatrixInv.h, 478, 479
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/NFIntegral.h, 480
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/NucLine.h, 480, 481
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/PPair.h, 482
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/RMatrixFunc.h, 484

/Users/kuba/Desktop/R-Matrix/AZURE2/include/ReactionRate.h, 482, 483
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/SegLine.h, 484, 485
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/ShftFunc.h, 485, 486
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/TargetEffect.h, 486
 /Users/kuba/Desktop/R-Matrix/AZURE2/include/WhitFunc.h, 487, 488
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AChannel.cpp, 488
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ALevel.cpp, 489
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AMatrixFunc.cpp, 491, 492
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZURE2.cpp, 495, 499
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZURECalc.cpp, 506
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREMain.cpp, 507
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREOutput.cpp, 511
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AZUREParams.cpp, 512
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/AngCoeff.cpp, 494
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/CNuc.cpp, 513
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/Config.cpp, 531
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/CoulFunc.cpp, 533
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/Decay.cpp, 535
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/DoubleFactorial.cpp, 536
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ECIntegral.cpp, 536, 537
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ECMGroup.cpp, 539
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EData.cpp, 540
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EDataIterator.cpp, 552
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EPoint.cpp, 555, 556
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ESegment.cpp, 571
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EffectiveCharge.cpp, 553, 554
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/EigenFunc.cpp, 554
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/Equation.cpp, 566
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/GSLEException.cpp, 578
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/GenMatrixFunc.cpp, 574
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/IntegratedFermiFunc.cpp, 578
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/Interference.cpp, 579
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/JGroup.cpp, 580
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/KGroup.cpp, 581
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/KLGroup.cpp, 582
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/MGroup.cpp, 583
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/MatrixInv.cpp, 582, 583
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/NFIntegral.cpp, 584
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/PPair.cpp, 584
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/RMatrixFunc.cpp, 589
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ReactionRate.cpp, 586, 587
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/ShftFunc.cpp, 592, 593
 /Users/kuba/Desktop/R-Matrix/AZURE2/src/TargetEffect.cpp, 593
 ~AZURECalc, AZURECalc, 53
 ~AZUREFBuffer, AZUREFBuffer, 55
 ~AZUREMain, AZUREMain, 57
 ~AZUREOutput, AZUREOutput, 61
 ~Coulomb_wave_functions, Coulomb_wave_functions, 93
 ~ECIntegral, ECIntegral, 102
 ~GSLEException, GSLEException, 167
 ~GenMatrixFunc, GenMatrixFunc, 161
 ~NFIntegral, NFIntegral, 192
 ~PlotEntry, PlotEntry, 210
 ~ShftFunc, ShftFunc, 256
 ~SyntaxError, SyntaxError, 257
 aa, NucLine, 194
 AboutAZURE2Dialog, 19
 AboutAZURE2Dialog, 19
 AChannel, 20
 AChannel, 20

- GetBoundaryCondition, 21
- GetL, 21
- GetPairNum, 21
- GetRadType, 21
- GetS, 21
- SetBoundaryCondition, 21
- AddAInvMatrixElement
 - AMatrixFunc, 49
- AddAMatrix
 - AMatrixFunc, 49
- AddAZUREFBuffer
 - AZUREOutput, 61
- AddChannel
 - JGroup, 172
- AddDecay
 - PPair, 216
- AddECAmplitude
 - EPoint, 129
- AddECConversionFactor
 - ALevel, 40
- AddECMGroup
 - KGroup, 175
- AddECTMatrixElement
 - GenMatrixFunc, 161
- AddExpCoulombPhase
 - EPoint, 129
- AddExpHardSpherePhase
 - EPoint, 129
- AddGamma
 - ALevel, 40
- AddInterference
 - KLGroup, 177
- AddJGroup
 - CNuc, 78
- AddKGroup
 - Decay, 99
- AddKLGroup
 - Decay, 99
- AddLegendreP
 - EPoint, 130
- AddLevel
 - JGroup, 172
- addLevel
 - LevelsTab, 185
- AddLevelDialog, 22
 - AddLevelDialog, 22
 - energyText, 23
 - jValueText, 23
 - piValueCombo, 23
- addLine
 - TargetIntTab, 267
- AddLocalMappedPoint
 - EPoint, 130
- AddLoElement
 - EPoint, 130
- AddMGroup
 - KGroup, 175
- AddNFIntegral
 - ALevel, 41
- AddPair
 - CNuc, 78
- addPair
 - PairsTab, 207
- AddPairDialog, 23
 - AddPairDialog, 24
 - channelRadiusText, 24
 - e1Check, 24
 - e2Check, 25
 - excitationEnergyText, 25
 - heavyJText, 25
 - heavyMText, 25
 - heavyPiCombo, 25
 - heavyZText, 25
 - lightJText, 25
 - lightMText, 25
 - lightPiCombo, 26
 - lightZText, 26
 - multBox, 26
 - pairTypeCombo, 26
 - seperationEnergyText, 26
 - updateLightParticle, 24
- AddPoint
 - ESegment, 147
- AddRLInvMatrix
 - RMatrixFunc, 228
- AddRLInvRMatrixElement
 - RMatrixFunc, 228
- AddRLMatrixElement
 - RMatrixFunc, 228
- AddRMatrixElement
 - RMatrixFunc, 228
- AddSegDataDialog, 27
 - AddSegDataDialog, 28
 - dataFileText, 28
 - dataNormErrorLabel, 28
 - dataNormErrorText, 28
 - dataNormText, 29
 - dataTypeChanged, 28
 - dataTypeCombo, 29
 - entrancePairIndexSpin, 29
 - exitPairIndexSpin, 29
 - highAngleText, 29
 - highEnergyText, 29
 - lowAngleText, 29
 - lowEnergyText, 29
 - phaseJValueLabel, 30
 - phaseJValueText, 30
 - phaseLValueLabel, 30
 - phaseLValueText, 30
 - setChooseFile, 28
 - totalCaptureLabel, 30
 - varyNormChanged, 28
 - varyNormCheck, 30
- addSegDataLine
 - SegmentsTab, 244
- AddSegment

- EData, 108
- AddSegTestDialog, 31
 - AddSegTestDialog, 31
 - angDistLabel, 32
 - angDistSpin, 32
 - angleStepText, 32
 - dataTypeChanged, 32
 - dataTypeCombo, 32
 - energyStepText, 32
 - entrancePairIndexSpin, 32
 - exitPairIndexSpin, 32
 - highAngleText, 33
 - highEnergyText, 33
 - lowAngleText, 33
 - lowEnergyText, 33
 - phaseJValueLabel, 33
 - phaseJValueText, 33
 - phaseLValueLabel, 33
 - phaseLValueText, 33
 - totalCaptureLabel, 34
- addSegTestLine
 - SegmentsTab, 245
- AddSqrtPenetrability
 - EPoint, 130
- AddSubPoint
 - EPoint, 130
- AddTargetEffect
 - EData, 108
- AddTargetIntDialog, 34
 - AddTargetIntDialog, 35
 - convolutionCheckChanged, 35
 - createParameterItem, 35
 - createQCoefficientItem, 35
 - densityText, 37
 - isConvolutionCheck, 37
 - isQCoefficientCheck, 37
 - isTargetIntegrationCheck, 37
 - numParametersSpin, 37
 - numPointsSpin, 37
 - numQCoefficientSpin, 37
 - parameterChanged, 36
 - parameterSpinChanged, 36
 - parametersTable, 38
 - qCoefficientChanged, 36
 - qCoefficientCheckChanged, 36
 - qCoefficientSpinChanged, 36
 - qCoefficientTable, 38
 - segmentsListText, 38
 - sigmaText, 38
 - stoppingPowerEqText, 38
 - targetIntCheckChanged, 36
 - tempParameters, 38
 - tempQCoefficients, 38
- AddTMatrixElement
 - GenMatrixFunc, 161
- AddToTempTMatrix
 - GenMatrixFunc, 161
- ALevel, 39
- AddECConversionFactor, 40
- AddGamma, 40
- AddNFIntegral, 41
- ALevel, 40
- ChannelFixed, 41
- EnergyFixed, 41
- GetBigGamma, 41
- GetE, 41
- GetECConversionFactor, 42
- GetECMultMask, 42
- GetECPairNum, 42
- GetExternalGamma, 42
- GetFitE, 42
- GetFitGamma, 42
- GetGamma, 43
- GetNFIntegral, 43
- GetShiftFunction, 43
- GetSqrtNFFactor, 43
- GetTransformE, 43
- GetTransformGamma, 43
- GetTransformIterations, 44
- IsECLLevel, 44
- IsInRMatrix, 44
- NumNFIntegrals, 44
- SetBigGamma, 44
- SetE, 44
- SetECParams, 45
- SetExternalGamma, 45
- SetFitE, 45
- SetFitGamma, 45
- SetGamma, 45
- SetShiftFunction, 46
- SetSqrtNFFactor, 46
- SetTransformE, 46
- SetTransformGamma, 46
- SetTransformIterations, 46
- AMatrixFunc, 47
 - AddAInvMatrixElement, 49
 - AddAMatrix, 49
 - AMatrixFunc, 48
 - CalculateCrossSection, 49
 - CalculateTMatrix, 49
 - ClearMatrices, 49
 - compound, 49
 - configure, 50
 - FillMatrices, 50
 - GetAMatrixElement, 50
 - GetJSpecAInvMatrix, 50
 - InvertMatrices, 50
- AngCoeff, 51
 - ClebGord, 51
 - Racah, 51
- angDistLabel
 - AddSegTestDialog, 32
- angDistsCheckCombo
 - EditChecksDialog, 119
- angDistSpin
 - AddSegTestDialog, 32

- angle
 - DataLine, 97
 - PlotPoint, 211
- angleStep
 - SegmentsTestData, 248
- angleStepText
 - AddSegTestDialog, 32
- aStep
 - ExtrapLine, 154
- attach
 - PlotEntry, 210
- avagadroNum
 - Constants.h, 458
- AZURE2, 1
- AZURE2.cpp
 - checkExternalCapture, 496
 - commandShell, 496
 - exitMessage, 496
 - getExternalCaptureFile, 496
 - getParameterFile, 496
 - getRateParams, 496
 - getTemperatureFile, 497
 - main, 497
 - parseOptions, 497
 - printHelp, 497
 - processCommand, 497
 - readSegmentFile, 498
 - startMessage, 498
 - welcomeMessage, 498
- AZURECalc, 52
 - ~AZURECalc, 53
 - AZURECalc, 53
 - compound, 53
 - configure, 53
 - data, 53
 - operator(), 53
 - SetErrorDef, 54
 - Up, 54
- AZUREFBuffer, 54
 - ~AZUREFBuffer, 55
 - AZUREFBuffer, 55
 - GetEntranceKey, 55
 - GetExitKey, 55
 - GetFBuffer, 55
 - IsAngDist, 56
- AZUREMain, 56
 - ~AZUREMain, 57
 - AZUREMain, 57
 - compound, 57
 - configure, 57
 - data, 57
 - operator(), 57
- AZUREMainThread, 58
 - AZUREMainThread, 59
 - configure, 59
 - readyToRun, 59
 - run, 59
 - RunTab, 233
 - stopAZURE, 59
- AZUREMainThreadWorker, 59
 - AZUREMainThreadWorker, 60
 - done, 60
 - run, 60
- AZUREOutput, 60
 - ~AZUREOutput, 61
 - AddAZUREFBuffer, 61
 - AZUREOutput, 61
 - GetAZUREFBuffer, 61
 - GetOutputDir, 62
 - IsAZUREFBuffer, 62
 - IsExtrap, 62
 - NumAZUREFBuffers, 62
 - operator(), 62
 - SetExtrap, 63
- AZUREParams, 63
 - GetMinuitParams, 64
 - ReadUserParameters, 64
 - WriteParameterErrors, 64
 - WriteUserParameters, 64
- AZUREPlot, 65
 - AZUREPlot, 65
 - clearEntries, 65
 - draw, 65
 - exportPlot, 66
 - PlotEntry, 211
 - PlotTab, 215
 - print, 66
 - setXAxisLog, 66
 - setXAxisType, 66
 - setYAxisLog, 66
 - setYAxisType, 66
 - update, 66
- AZURESetup, 67
 - AZURESetup, 67
 - DeleteThread, 68
 - GetConfig, 68
 - open, 68
 - RunTab, 233
 - SaveAndRun, 68
- AZURESetup.cpp
 - checkExternalCapture, 368
 - exitMessage, 368
 - readSegmentFile, 368
 - startMessage, 369
- AZUREZoomer, 68
 - AZUREZoomer, 69
 - trackerTextF, 69
- begin
 - EData, 109
- boltzConst
 - Constants.h, 458
- boundaryCheckCombo
 - EditChecksDialog, 119
- CalcAngularDists
 - CNuc, 78

- CalcBoundaryConditions
 - CNuc, [78](#)
- CalcCoulombAmplitude
 - EData, [109](#)
 - EPoint, [131](#)
- CalcEDependentValues
 - EData, [109](#)
 - EPoint, [131](#)
- CalcExternalWidth
 - CNuc, [78](#)
- CalcLegendreP
 - EData, [109](#)
 - EPoint, [131](#)
- CalcShiftFunctions
 - CNuc, [79](#)
- Calculate
 - EPoint, [131](#)
- CALCULATE_REACTION_RATE
 - Config, [86](#)
- CALCULATE_WITH_DATA
 - Config, [86](#)
- calculateChannels
 - LevelsTab, [185](#)
- CalculateCrossSection
 - AMatrixFunc, [49](#)
 - GenMatrixFunc, [161](#)
 - RMatrixFunc, [229](#)
- CalculateECAmplitudes
 - EData, [109](#)
 - EPoint, [131](#)
- CalculateFileRates
 - ReactionRate, [224](#)
- CalculateRates
 - ReactionRate, [224](#)
- CalculateTMatrix
 - AMatrixFunc, [49](#)
 - GenMatrixFunc, [162](#)
 - RMatrixFunc, [229](#)
- ChannelDetails, [69](#)
 - ChannelDetails, [70](#)
 - details, [70](#)
 - reducedWidthText, [70](#)
 - setNormParam, [70](#)
- channelFix
 - NucLine, [194](#)
- ChannelFixed
 - ALevel, [41](#)
- channelRadius
 - PairsData, [201](#)
- channelRadiusText
 - AddPairDialog, [24](#)
- ChannelsData, [71](#)
 - isFixed, [71](#)
 - levelIndex, [71](#)
 - IValue, [71](#)
 - pairIndex, [71](#)
 - radType, [71](#)
 - reducedWidth, [72](#)
- SIZE, [72](#)
- sValue, [72](#)
- ChannelsModel, [72](#)
 - ChannelsModel, [73](#)
 - columnCount, [73](#)
 - data, [73](#)
 - flags, [73](#)
 - getChannels, [74](#)
 - getSpinLabel, [74](#)
 - headerData, [74](#)
 - insertRows, [74](#)
 - isChannel, [74](#)
 - removeRows, [74](#)
 - rowCount, [75](#)
 - setData, [75](#)
 - setPairsModel, [75](#)
- chanRad
 - NFIntegral, [192](#)
- CHECK_ANGULAR_DISTS
 - Config, [86](#)
- CHECK_BOUNDARY_CONDITIONS
 - Config, [86](#)
- CHECK_COMPOUND_NUCLEUS
 - Config, [86](#)
- CHECK_COUL_AMPLITUDES
 - Config, [86](#)
- CHECK_DATA
 - Config, [86](#)
- CHECK_ENERGY_DEP
 - Config, [86](#)
- CHECK_LEGENDRE
 - Config, [86](#)
- CHECK_PATHWAYS
 - Config, [86](#)
- checkdir
 - Config, [87](#)
- checkExternalCapture
 - AZURE2.cpp, [496](#)
 - AZURESetup.cpp, [368](#)
- CheckFileFlags
 - Config, [85](#)
- CheckForInputFiles
 - Config, [87](#)
- checksDir
 - Directories, [101](#)
- checksDirectoryText
 - EditDirsDialog, [121](#)
- chiVariance
 - Config, [87](#)
- ChooseFileButton, [75](#)
 - ChooseFileButton, [76](#)
 - click, [76](#)
 - clicked, [76](#)
 - setLineEdit, [76](#)
- chRad
 - NucLine, [194](#)
- clearEntries
 - AZUREPlot, [65](#)

- ClearLocalMappedPoints
 - EPoint, [132](#)
- ClearMatrices
 - AMatrixFunc, [49](#)
 - GenMatrixFunc, [162](#)
 - RMatrixFunc, [229](#)
- ClearTempTMatrices
 - GenMatrixFunc, [162](#)
- ClebGord
 - AngCoeff, [51](#)
- click
 - ChooseFileButton, [76](#)
- clicked
 - ChooseFileButton, [76](#)
- Clone
 - CNuc, [79](#)
 - EData, [110](#)
- CNuc, [77](#)
 - AddJGroup, [78](#)
 - AddPair, [78](#)
 - CalcAngularDists, [78](#)
 - CalcBoundaryConditions, [78](#)
 - CalcExternalWidth, [78](#)
 - CalcShiftFunctions, [79](#)
 - Clone, [79](#)
 - Fill, [79](#)
 - FillCompoundFromParams, [79](#)
 - FillMnParams, [79](#)
 - GetJGroup, [80](#)
 - GetMaxLValue, [80](#)
 - GetPair, [80](#)
 - GetPairNumFromKey, [80](#)
 - Initialize, [80](#)
 - IsJGroup, [81](#)
 - IsPair, [81](#)
 - IsPairKey, [81](#)
 - NumJGroups, [81](#)
 - NumPairs, [81](#)
 - ParseExternalCapture, [82](#)
 - PrintAngularDists, [82](#)
 - PrintBoundaryConditions, [82](#)
 - PrintNuc, [82](#)
 - PrintPathways, [82](#)
 - PrintTransformParams, [83](#)
 - SetMaxLValue, [83](#)
 - SortPathways, [83](#)
 - TransformIn, [83](#)
 - TransformOut, [83](#)
- CNuc.h
 - DoubleFactorial, [454](#)
- columnCount
 - ChannelsModel, [73](#)
 - LevelsModel, [182](#)
 - PairsModel, [204](#)
 - SegmentsDataModel, [241](#)
 - SegmentsTestModel, [251](#)
 - TargetIntModel, [265](#)
- commandShell
 - AZURE2.cpp, [496](#)
- complex
 - Constants.h, [457](#)
- complex_functions.cpp
 - exp_l_omega_chi_calc, [289](#)
 - expm1, [289](#)
 - log1p, [289](#)
 - log_Cl_eta_calc, [290](#)
 - log_cut_constant_AS_calc, [290](#)
 - log_cut_constant_CFa_calc, [290](#)
 - log_cut_constant_CFb_calc, [290](#)
 - log_Gamma, [290](#)
 - sigma_l_calc, [290](#)
 - sin_chi_calc, [291](#)
- complex_functions.H
 - exp_l_omega_chi_calc, [276](#)
 - expm1, [276](#)
 - inf_norm, [277](#)
 - isfinite, [277](#)
 - log1p, [277](#)
 - log_Cl_eta_calc, [277](#)
 - log_cut_constant_AS_calc, [277](#)
 - log_cut_constant_CFa_calc, [277](#)
 - log_cut_constant_CFb_calc, [278](#)
 - log_Gamma, [278](#)
 - operator!=, [278](#)
 - operator+, [279](#), [280](#)
 - operator-, [280](#)
 - operator/, [281](#)
 - operator==, [281](#), [282](#)
 - operator*, [279](#)
 - precision, [282](#)
 - sigma_l_calc, [282](#)
 - SIGN, [276](#)
 - sin_chi_calc, [282](#)
 - sqrt_precision, [282](#)
- compound
 - AMatrixFunc, [49](#)
 - AZURECalc, [53](#)
 - AZUREMain, [57](#)
 - GenMatrixFunc, [162](#)
 - gsl_reactionrate_params, [165](#)
 - ReactionRate, [224](#)
 - RMatrixFunc, [229](#)
- compoundCheckCombo
 - EditChecksDialog, [119](#)
- Config, [84](#)
 - CALCULATE_REACTION_RATE, [86](#)
 - CALCULATE_WITH_DATA, [86](#)
 - CHECK_ANGULAR_DISTs, [86](#)
 - CHECK_BOUNDARY_CONDITIONS, [86](#)
 - CHECK_COMPOUND_NUCLEUS, [86](#)
 - CHECK_COUL_AMPLITUDES, [86](#)
 - CHECK_DATA, [86](#)
 - CHECK_ENERGY_DEP, [86](#)
 - CHECK_LEGENDRE, [86](#)
 - CHECK_PATHWAYS, [86](#)
 - checkdir, [87](#)

- CheckFileFlags, 85
- CheckForInputFiles, 87
- chiVariance, 87
- Config, 86
- configfile, 87
- fileCheckMask, 88
- IGNORE_ZERO_WIDTHS, 86
- integralsfile, 88
- maxLOrder, 88
- outputdir, 88
- outStream, 88
- ParameterFlags, 86
- paramfile, 88
- paramMask, 89
- PERFORM_ERROR_ANALYSIS, 86
- PERFORM_FIT, 86
- rateParams, 89
- ReadConfigFile, 87
- Reset, 87
- screenCheckMask, 89
- stopFlag, 89
- TRANSFORM_PARAMETERS, 86
- USE_AMATRIX, 86
- USE_BRUNE_FORMALISM, 86
- USE_EXTERNAL_CAPTURE, 86
- USE_GSL_COULOMB_FUNC, 86
- USE_LONGWAVELENGTH_APPROX, 86
- USE_PREVIOUS_INTEGRALS, 86
- USE_PREVIOUS_PARAMETERS, 86
- USE_RMC_FORMALISM, 86
- configfile
 - Config, 87
- configure
 - AMatrixFunc, 50
 - AZURECalc, 53
 - AZUREMain, 57
 - AZUREMainThread, 59
 - GenMatrixFunc, 162
 - gsl_reactionrate_params, 165
 - ReactionRate, 224
 - RMatrixFunc, 229
- Constants.h
 - avagadroNum, 458
 - boltzConst, 458
 - complex, 457
 - fstruc, 458
 - hbarc, 458
 - isE1, 458
 - isE2, 458
 - isM1, 459
 - lightSpeedInCmPerS, 459
 - matrix_c, 457
 - matrix_r, 457
 - maxECMult, 459
 - nuclearMagneon, 459
 - pi, 459
 - uconv, 459
 - vector_c, 457
 - vector_matrix_c, 457
 - vector_matrix_r, 457
 - vector_r, 458
- ConvertCrossSection
 - EPoint, 132
- ConvertDecayEnergy
 - EPoint, 132
- ConvertLabAngle
 - EPoint, 132
- ConvertLabEnergy
 - EPoint, 133
- convolutionCheckChanged
 - AddTargetIntDialog, 35
- convolutionRange
 - TargetEffect, 262
- coulAmpCheckCombo
 - EditChecksDialog, 119
- CoulFunc, 89
 - CoulFunc, 90
 - coulLast, 90
 - energyLast, 90
 - lLast, 91
 - operator(), 91
 - Penetrability, 91
 - PEShift, 91
 - PEShift_dE, 91
 - radiusLast, 92
 - redmass, 92
 - setLast, 92
 - z1, 92
 - z2, 92
- coulLast
 - CoulFunc, 90
- Coulomb_wave_functions, 93
 - ~Coulomb_wave_functions, 93
 - Coulomb_wave_functions, 93
 - eta, 95
 - F_dF, 94
 - F_dF_init, 94
 - G_dG, 94
 - H_dH, 94
 - H_dH_scaled, 94
 - is_it_normalized, 95
 - l, 95
- CoulWaves, 95
 - dF, 96
 - dG, 96
 - F, 96
 - G, 96
- createParameterItem
 - AddTargetIntDialog, 35
- createQCoefficientItem
 - AddTargetIntDialog, 35
- crossSection
 - DataLine, 97
- data
 - AZURECalc, 53
 - AZUREMain, 57

- ChannelsModel, 73
- LevelsModel, 182
- PairsModel, 204
- SegDataProxyModel, 234
- SegmentsDataModel, 241
- SegmentsTestModel, 251
- SegTestProxyModel, 255
- TargetIntModel, 265
- dataCheckCombo
 - EditChecksDialog, 119
- dataCrossSection
 - PlotPoint, 211
- dataErrorCrossSection
 - PlotPoint, 211
- dataErrorSFactor
 - PlotPoint, 212
- dataFile
 - SegLine, 235
 - SegmentsDataData, 238
- dataFileText
 - AddSegDataDialog, 28
- DataLine, 97
 - angle, 97
 - crossSection, 97
 - DataLine, 97
 - energy, 97
 - error, 98
- dataNorm
 - SegLine, 235
 - SegmentsDataData, 238
- dataNormError
 - SegLine, 235
 - SegmentsDataData, 238
- dataNormErrorLabel
 - AddSegDataDialog, 28
- dataNormErrorText
 - AddSegDataDialog, 28
- dataNormText
 - AddSegDataDialog, 29
- dataSFactor
 - PlotPoint, 212
- dataType
 - SegmentsDataData, 238
 - SegmentsTestData, 248
- dataTypeChanged
 - AddSegDataDialog, 28
 - AddSegTestDialog, 32
- dataTypeCombo
 - AddSegDataDialog, 29
 - AddSegTestDialog, 32
- Decay, 98
 - AddKGroup, 99
 - AddKLGroup, 99
 - Decay, 99
 - GetKGroup, 99
 - GetKLGroup, 99
 - GetPairNum, 99
 - IsKGroup, 100
 - IsKLGroup, 100
 - NumKGroups, 100
 - NumKLGroups, 100
- DeleteLastSegment
 - EData, 110
- deleteLine
 - TargetIntTab, 268
- deleteSegDataLine
 - SegmentsTab, 245
- deleteSegTestLine
 - SegmentsTab, 245
- DeleteThread
 - AZURESetup, 68
- density
 - TargetIntData, 262
- densityText
 - AddTargetIntDialog, 37
- detach
 - PlotEntry, 210
- details
 - ChannelDetails, 70
- dF
 - CoulWaves, 96
- dG
 - CoulWaves, 96
- Directories, 101
 - checksDir, 101
 - Directories, 101
 - outputDir, 101
- done
 - AZUREMainThreadWorker, 60
- DoubleFactorial
 - CNuc.h, 454
 - DoubleFactorial.cpp, 536
 - ECIntegral.cpp, 536
 - EffectiveCharge.cpp, 553
- DoubleFactorial.cpp
 - DoubleFactorial, 536
- draw
 - AZUREPlot, 65
 - PlotTab, 214
- e1Check
 - AddPairDialog, 24
- e2
 - NucLine, 194
- e2Check
 - AddPairDialog, 25
- e3
 - NucLine, 195
- ec_tmatrix_
 - GenMatrixFunc, 164
- ECIntegral, 102
 - ~ECIntegral, 102
 - ECIntegral, 102
 - operator(), 103
- ECIntegral.cpp
 - DoubleFactorial, 536
- ECMGroup, 103

- ECMGroup, [104](#)
- GetChanCapDecay, [105](#)
- GetChanCapKGroup, [105](#)
- GetChanCapMGroup, [105](#)
- GetFinalChannel, [105](#)
- GetIntChannelNum, [105](#)
- GetJ, [105](#)
- GetJGroupNum, [105](#)
- GetL, [106](#)
- GetLevelNum, [106](#)
- GetMult, [106](#)
- GetRadType, [106](#)
- GetStatSpinFactor, [106](#)
- IsChannelCapture, [106](#)
- SetStatSpinFactor, [107](#)
- ecMultMask
 - NucLine, [195](#)
 - PairsData, [201](#)
- EData, [107](#)
 - AddSegment, [108](#)
 - AddTargetEffect, [108](#)
 - begin, [109](#)
 - CalcCoulombAmplitude, [109](#)
 - CalcEDependentValues, [109](#)
 - CalcLegendreP, [109](#)
 - CalculateECAmplitudes, [109](#)
 - Clone, [110](#)
 - DeleteLastSegment, [110](#)
 - EData, [108](#)
 - end, [110](#)
 - Fill, [110](#)
 - FillMnParams, [110](#)
 - FillNormsFromParams, [111](#)
 - GetNormParamOffset, [111](#)
 - GetSegment, [111](#)
 - GetSegmentFromKey, [111](#)
 - GetSegments, [111](#)
 - GetTargetEffect, [111](#)
 - Initialize, [112](#)
 - IsErrorAnalysis, [112](#)
 - IsFit, [112](#)
 - IsSegmentKey, [112](#)
 - Iterate, [112](#)
 - Iterations, [113](#)
 - MakePoints, [113](#)
 - MapData, [113](#)
 - NumSegments, [113](#)
 - NumTargetEffects, [113](#)
 - PrintCoulombAmplitude, [113](#)
 - PrintData, [114](#)
 - PrintEDependentValues, [114](#)
 - PrintLegendreP, [114](#)
 - ReadTargetEffectsFile, [114](#)
 - ResetIterations, [114](#)
 - SetErrorAnalysis, [115](#)
 - SetFit, [115](#)
 - SetNormParamOffset, [115](#)
 - WriteOutputFiles, [115](#)
- EDatalterator, [116](#)
 - EDatalterator, [116](#)
 - operator!=, [117](#)
 - operator++, [117](#)
 - operator==, [117](#)
 - point, [117](#)
 - segment, [117](#)
 - SetEnd, [118](#)
- EDatalterator.h
 - EPointIterator, [466](#)
 - EPointMapIterator, [466](#)
 - ESegmentIterator, [466](#)
- EditChecksDialog, [118](#)
 - angDistsCheckCombo, [119](#)
 - boundaryCheckCombo, [119](#)
 - compoundCheckCombo, [119](#)
 - coulAmpCheckCombo, [119](#)
 - dataCheckCombo, [119](#)
 - EditChecksDialog, [119](#)
 - legendreCheckCombo, [119](#)
 - IMatrixCheckCombo, [120](#)
 - pathwaysCheckCombo, [120](#)
- EditDirsDialog, [120](#)
 - checksDirectoryText, [121](#)
 - EditDirsDialog, [121](#)
 - outputDirectoryText, [121](#)
- editLevel
 - LevelsTab, [186](#)
- editLine
 - TargetIntTab, [268](#)
- EditOptionsDialog, [121](#)
 - EditOptionsDialog, [122](#)
 - ignoreExternalsCheck, [122](#)
 - noTransformCheck, [122](#)
 - useBruneCheck, [122](#)
 - useGSLCoulCheck, [122](#)
 - useRMCCheck, [122](#)
- editPair
 - PairsTab, [207](#)
- editSegDataLine
 - SegmentsTab, [245](#)
- editSegTestLine
 - SegmentsTab, [245](#)
- EffectiveCharge, [123](#)
 - EffectiveCharge, [123](#)
 - operator(), [124](#)
- EffectiveCharge.cpp
 - DoubleFactorial, [553](#)
- EigenFunc, [124](#)
 - EigenFunc, [124](#)
 - eigenvalues, [125](#)
 - eigenvectors, [125](#)
- eigenvalues
 - EigenFunc, [125](#)
- eigenvectors
 - EigenFunc, [125](#)
- end
 - EData, [110](#)

- energy
 - DataLine, 97
 - LevelsData, 179
 - PlotPoint, 212
- EnergyDerivative
 - ShftFunc, 256
- EnergyFixed
 - ALevel, 41
- energyLast
 - CoulFunc, 90
- EnergyMap, 125
 - point, 126
 - segment, 126
- energyStep
 - SegmentsTestData, 249
- energyStepText
 - AddSegTestDialog, 32
- energyText
 - AddLevelDialog, 23
- entranceKey
 - ExtrapLine, 154
 - gsl_reactionrate_params, 165
 - ReactionRate, 224
 - SegLine, 235
- entrancePair
 - RateParams, 222
- entrancePairIndex
 - SegmentsDataData, 239
 - SegmentsTestData, 249
- entrancePairIndexSpin
 - AddSegDataDialog, 29
 - AddSegTestDialog, 32
- entranceSepE
 - NucLine, 195
- EPoint, 126
 - AddECAmplitude, 129
 - AddExpCoulombPhase, 129
 - AddExpHardSpherePhase, 129
 - AddLegendreP, 130
 - AddLocalMappedPoint, 130
 - AddLoElement, 130
 - AddSqrtPenetrability, 130
 - AddSubPoint, 130
 - CalcCoulombAmplitude, 131
 - CalcEDependentValues, 131
 - CalcLegendreP, 131
 - Calculate, 131
 - CalculateECAmplitudes, 131
 - ClearLocalMappedPoints, 132
 - ConvertCrossSection, 132
 - ConvertDecayEnergy, 132
 - ConvertLabAngle, 132
 - ConvertLabEnergy, 133
 - EPoint, 128, 129
 - GetAngularDist, 133
 - GetCMAngle, 133
 - GetCMCrossSection, 133
 - GetCMCrossSectionError, 133
 - GetCMEnergy, 134
 - GetCoulombAmplitude, 134
 - GetECAmplitude, 134
 - GetEntranceKey, 134
 - GetExcitationEnergy, 134
 - GetExitKey, 134
 - GetExpCoulombPhase, 135
 - GetExpHardSpherePhase, 135
 - GetFitCrossSection, 135
 - GetGeometricalFactor, 135
 - GetJ, 135
 - GetL, 136
 - GetLabAngle, 136
 - GetLabCrossSection, 136
 - GetLabCrossSectionError, 136
 - GetLabEnergy, 136
 - GetLegendreP, 136
 - GetLocalMappedPoint, 137
 - GetLoElement, 137
 - GetMap, 137
 - GetMappedPoints, 137
 - GetMaxAngDistOrder, 137
 - GetMaxLOrder, 138
 - GetNumAngularDists, 138
 - GetParentData, 138
 - GetSFactorConversion, 138
 - GetSqrtPenetrability, 138
 - GetStoppingPower, 138
 - GetSubPoint, 139
 - GetSubPoints, 139
 - GetTargetEffectNum, 139
 - GetTargetThickness, 139
 - Initialize, 139
 - IntegrateTargetEffect, 139
 - IsAngularDist, 140
 - IsDifferential, 140
 - IsMapped, 140
 - IsPhase, 140
 - IsTargetEffect, 140
 - NumLocalMappedPoints, 140
 - NumSubPoints, 141
 - SetAngularDists, 141
 - SetCoulombAmplitude, 141
 - SetExitKey, 141
 - SetFitCrossSection, 141
 - SetGeometricalFactor, 141
 - SetMap, 142
 - SetParentData, 142
 - SetSFactorConversion, 142
 - SetStoppingPower, 142
 - SetTargetEffectNum, 142
 - SetTargetThickness, 143
- EPointIterator
 - EDataIterator.h, 466
- EPointMapIterator
 - EDataIterator.h, 466
- Equation, 143
 - Equation, 144

- Evaluate, [145](#)
- GetParameters, [145](#)
- Initialize, [145](#)
- SetParameter, [145](#)
- error
 - DataLine, [98](#)
- ESegment, [146](#)
 - AddPoint, [147](#)
 - ESegment, [147](#)
 - Fill, [147](#)
 - GetAStep, [147](#)
 - GetDataFile, [148](#)
 - GetEntranceKey, [148](#)
 - GetEStep, [148](#)
 - GetExitKey, [148](#)
 - GetJ, [148](#)
 - GetL, [148](#)
 - GetMaxAngDistOrder, [149](#)
 - GetMaxAngle, [149](#)
 - GetMaxEnergy, [149](#)
 - GetMinAngle, [149](#)
 - GetMinEnergy, [149](#)
 - GetNominalNorm, [149](#)
 - GetNorm, [150](#)
 - GetNormError, [150](#)
 - GetPoint, [150](#)
 - GetPoints, [150](#)
 - GetSegmentChiSquared, [150](#)
 - GetSegmentKey, [150](#)
 - GetTargetEffectNum, [151](#)
 - IsAngularDist, [151](#)
 - IsDifferential, [151](#)
 - IsInSegment, [151](#)
 - IsPhase, [151](#)
 - IsTargetEffect, [151](#)
 - IsTotalCapture, [152](#)
 - IsVaryNorm, [152](#)
 - NumPoints, [152](#)
 - SetExitKey, [152](#)
 - SetIsTotalCapture, [152](#)
 - SetNorm, [152](#)
 - SetSegmentChiSquared, [153](#)
 - SetSegmentKey, [153](#)
 - SetTargetEffectNum, [153](#)
 - SetVaryNorm, [153](#)
- ESegmentIterator
 - EDataIterator.h, [466](#)
- eStep
 - ExtrapLine, [155](#)
- eta
 - Coulomb_wave_functions, [95](#)
- Evaluate
 - Equation, [145](#)
 - GenericFunction, [159](#)
- excitationEnergy
 - PairsData, [201](#)
 - PlotPoint, [212](#)
- excitationEnergyText
 - AddPairDialog, [25](#)
- exitKey
 - ExtrapLine, [155](#)
 - gsl_reactionrate_params, [165](#)
 - ReactionRate, [225](#)
 - SegLine, [235](#)
- exitMessage
 - AZURE2.cpp, [496](#)
 - AZURESetup.cpp, [368](#)
- exitPair
 - RateParams, [222](#)
- exitPairIndex
 - SegmentsDataData, [239](#)
 - SegmentsTestData, [249](#)
- exitPairIndexSpin
 - AddSegDataDialog, [29](#)
 - AddSegTestDialog, [32](#)
- exp_l_omega_chi_calc
 - complex_functions.cpp, [289](#)
 - complex_functions.H, [276](#)
- expm1
 - complex_functions.cpp, [289](#)
 - complex_functions.H, [276](#)
- exportPlot
 - AZUREPlot, [66](#)
- ExtrapLine, [154](#)
 - aStep, [154](#)
 - entranceKey, [154](#)
 - eStep, [155](#)
 - exitKey, [155](#)
 - ExtrapLine, [154](#)
 - isActive, [155](#)
 - isDiff, [155](#)
 - maxA, [155](#)
 - maxAngDistOrder, [155](#)
 - maxE, [156](#)
 - minA, [156](#)
 - minE, [156](#)
 - phaseJ, [156](#)
 - phaseL, [156](#)
- F
 - CoulWaves, [96](#)
- F_dF
 - Coulomb_wave_functions, [94](#)
- F_dF_init
 - Coulomb_wave_functions, [94](#)
- fileCheckMask
 - Config, [88](#)
- Fill
 - CNuc, [79](#)
 - EData, [110](#)
 - ESegment, [147](#)
- FillCompoundFromParams
 - CNuc, [79](#)
- FillMatrices
 - AMatrixFunc, [50](#)
 - GenMatrixFunc, [162](#)
 - RMatrixFunc, [230](#)

- FillMnParams
 - CNuc, [79](#)
 - EData, [110](#)
- FillNormsFromParams
 - EData, [111](#)
- filterAcceptsRow
 - SegTestProxyModel, [255](#)
- FilteredTextEdit, [157](#)
 - FilteredTextEdit, [157](#)
 - IsMouseFiltered, [157](#)
 - mouseDoubleClickEvent, [157](#)
 - mousePressEvent, [158](#)
 - SetMouseFiltered, [158](#)
 - write, [158](#)
- firstPair
 - SegPairs, [254](#)
- fitCrossSection
 - PlotPoint, [212](#)
- fitSFactor
 - PlotPoint, [212](#)
- flags
 - ChannelsModel, [73](#)
 - LevelsModel, [182](#)
 - SegmentsDataModel, [241](#)
 - SegmentsTestModel, [251](#)
 - TargetIntModel, [265](#)
- fstruc
 - Constants.h, [458](#)
- G
 - CoulWaves, [96](#)
- g1
 - NucLine, [195](#)
- g2
 - NucLine, [195](#)
- G_dG
 - Coulomb_wave_functions, [94](#)
- gamma
 - NucLine, [195](#)
- GenericFunction, [158](#)
 - Evaluate, [159](#)
 - GenericFunction, [159](#)
- GenMatrixFunc, [160](#)
 - ~GenMatrixFunc, [161](#)
 - AddECTMatrixElement, [161](#)
 - AddTMatrixElement, [161](#)
 - AddToTempTMatrix, [161](#)
 - CalculateCrossSection, [161](#)
 - CalculateTMatrix, [162](#)
 - ClearMatrices, [162](#)
 - ClearTempTMatrices, [162](#)
 - compound, [162](#)
 - configure, [162](#)
 - ec_tmatrix_, [164](#)
 - FillMatrices, [162](#)
 - GenMatrixFunc, [161](#)
 - GetECTMatrixElement, [163](#)
 - GetTempTMatrix, [163](#)
 - GetTMatrixElement, [163](#)
 - InvertMatrices, [163](#)
 - IsTempTMatrix, [163](#)
 - NewTempTMatrix, [164](#)
 - NumTempTMatrices, [164](#)
 - tmatrix_, [164](#)
- GetAMatrixElement
 - AMatrixFunc, [50](#)
- GetAngularDist
 - EPoint, [133](#)
- GetAStep
 - ESegment, [147](#)
- GetAZUREFBuffer
 - AZUREOutput, [61](#)
- GetBigGamma
 - ALevel, [41](#)
- GetBoundaryCondition
 - AChannel, [21](#)
- GetChanCapDecay
 - ECMGroup, [105](#)
- GetChanCapKGroup
 - ECMGroup, [105](#)
- GetChanCapMGroup
 - ECMGroup, [105](#)
- GetChannel
 - JGroup, [172](#)
- getChannels
 - ChannelsModel, [74](#)
- GetChNum
 - MGroup, [190](#)
- GetChpNum
 - MGroup, [190](#)
- GetChRad
 - PPair, [216](#)
- GetCMAngle
 - EPoint, [133](#)
- GetCMCrossSection
 - EPoint, [133](#)
- GetCMCrossSectionError
 - EPoint, [133](#)
- GetCMEnergy
 - EPoint, [134](#)
- GetConfig
 - AZURESetup, [68](#)
- GetConvolutionFactor
 - TargetEffect, [259](#)
- GetCoulombAmplitude
 - EPoint, [134](#)
- GetDataFile
 - ESegment, [148](#)
- getDataSegments
 - PlotTab, [214](#)
- GetDecay
 - PPair, [216](#)
- GetDensity
 - TargetEffect, [259](#)
- GetE
 - ALevel, [41](#)
- GetECAmplitude

- EPoint, [134](#)
- GetECConversionFactor
 - ALevel, [42](#)
- GetECMGroup
 - KGroup, [175](#)
- GetECMultMask
 - ALevel, [42](#)
- GetECPairNum
 - ALevel, [42](#)
- GetECTMatrixElement
 - GenMatrixFunc, [163](#)
- GetEntranceKey
 - AZUREFBuffer, [55](#)
 - EPoint, [134](#)
 - ESegment, [148](#)
- GetEStep
 - ESegment, [148](#)
- GetExcitationEnergy
 - EPoint, [134](#)
- GetExE
 - PPair, [216](#)
- GetExitKey
 - AZUREFBuffer, [55](#)
 - EPoint, [134](#)
 - ESegment, [148](#)
- GetExpCoulombPhase
 - EPoint, [135](#)
- GetExpHardSpherePhase
 - EPoint, [135](#)
- getExternalCaptureFile
 - AZURE2.cpp, [496](#)
- GetExternalGamma
 - ALevel, [42](#)
- GetFBuffer
 - AZUREFBuffer, [55](#)
- GetFinalChannel
 - ECMGroup, [105](#)
- GetFitCrossSection
 - EPoint, [135](#)
- GetFitE
 - ALevel, [42](#)
- GetFitGamma
 - ALevel, [42](#)
- GetG
 - PPair, [217](#)
- GetGamma
 - ALevel, [43](#)
- GetGeometricalFactor
 - EPoint, [135](#)
- GetI1I2Factor
 - PPair, [217](#)
- GetIntChannelNum
 - ECMGroup, [105](#)
- GetInterference
 - KLGroup, [177](#)
- GetInterferenceType
 - Interference, [170](#)
- GetJ
 - ECMGroup, [105](#)
 - EPoint, [135](#)
 - ESegment, [148](#)
 - JGroup, [172](#)
 - PPair, [217](#)
- GetJGroup
 - CNuc, [80](#)
- GetJGroupNum
 - ECMGroup, [105](#)
- GetJNum
 - MGroup, [190](#)
- GetJSpecAInvMatrix
 - AMatrixFunc, [50](#)
- GetJSpecRLMatrix
 - RMatrixFunc, [230](#)
- GetK
 - KLGroup, [178](#)
- GetKGroup
 - Decay, [99](#)
- GetKLGroup
 - Decay, [99](#)
- GetL
 - AChannel, [21](#)
 - ECMGroup, [106](#)
 - EPoint, [136](#)
 - ESegment, [148](#)
- GetLabAngle
 - EPoint, [136](#)
- GetLabCrossSection
 - EPoint, [136](#)
- GetLabCrossSectionError
 - EPoint, [136](#)
- GetLabEnergy
 - EPoint, [136](#)
- GetLegendreP
 - EPoint, [136](#)
- GetLevel
 - JGroup, [173](#)
- GetLevelNum
 - ECMGroup, [106](#)
- getLevels
 - LevelsModel, [183](#)
- getLines
 - SegmentsDataModel, [242](#)
 - SegmentsTestModel, [252](#)
 - TargetIntModel, [265](#)
- GetLocalMappedPoint
 - EPoint, [137](#)
- GetLoElement
 - EPoint, [137](#)
- GetLOrder
 - KLGroup, [178](#)
- GetM
 - PPair, [217](#)
- GetM1
 - Interference, [170](#)
- GetM2
 - Interference, [170](#)

GetMap
 EPoint, 137
 GetMappedPoints
 EPoint, 137
 GetMaxAngDistOrder
 EPoint, 137
 ESegment, 149
 GetMaxAngle
 ESegment, 149
 GetMaxEnergy
 ESegment, 149
 GetMaxLOrder
 EPoint, 138
 GetMaxLValue
 CNuc, 80
 GetMGroup
 KGroup, 175
 GetMinAngle
 ESegment, 149
 GetMinEnergy
 ESegment, 149
 GetMinuitParams
 AZUREParams, 64
 GetMult
 ECMGroup, 106
 GetNFIintegral
 ALevel, 43
 GetNominalNorm
 ESegment, 149
 GetNorm
 ESegment, 150
 GetNormError
 ESegment, 150
 GetNormParamOffset
 EData, 111
 GetNumAngularDists
 EPoint, 138
 GetOutputDir
 AZUREOutput, 62
 GetPair
 CNuc, 80
 GetPairKey
 PPair, 217
 GetPairNum
 AChannel, 21
 Decay, 99
 GetPairNumFromKey
 CNuc, 80
 getPairs
 PairsModel, 204
 getPairsModel
 PairsTab, 208
 getParameterFile
 AZURE2.cpp, 496
 GetParameters
 Equation, 145
 GetParentData
 EPoint, 138
 getParticleLabel
 PairsModel, 204
 GetPi
 JGroup, 173
 PPair, 217
 GetPoint
 ESegment, 150
 GetPoints
 ESegment, 150
 GetPType
 PPair, 218
 GetQCoefficient
 TargetEffect, 259
 GetRadType
 AChannel, 21
 ECMGroup, 106
 getRateParams
 AZURE2.cpp, 496
 getReactionLabel
 PairsModel, 204
 SegmentsDataModel, 242
 SegmentsTestModel, 252
 getReactionLabelTotalCapture
 PairsModel, 204
 GetRedMass
 PPair, 218
 GetRLInvMatrixElement
 RMatrixFunc, 230
 GetRLInvRMatrixElement
 RMatrixFunc, 230
 GetRLMatrixElement
 RMatrixFunc, 230
 GetRMatrixElement
 RMatrixFunc, 231
 GetS
 AChannel, 21
 KGroup, 176
 GetSegment
 EData, 111
 GetSegmentChiSquared
 ESegment, 150
 GetSegmentFromKey
 EData, 111
 GetSegmentKey
 ESegment, 150
 GetSegments
 EData, 111
 getSegmentsDataModel
 SegmentsTab, 245
 GetSegmentsList
 TargetEffect, 259
 getSegmentsTestModel
 SegmentsTab, 246
 GetSepE
 PPair, 218
 GetSFactorConversion
 EPoint, 138
 GetShiftFunction

- ALevel, [43](#)
- GetSigma
 - TargetEffect, [259](#)
- GetSp
 - KGroup, [176](#)
- getSpinLabel
 - ChannelsModel, [74](#)
 - LevelsModel, [183](#)
 - PairsModel, [205](#)
- GetSqrtNFFactor
 - ALevel, [43](#)
- GetSqrtPenetrability
 - EPoint, [138](#)
- GetStatSpinFactor
 - ECMGroup, [106](#)
 - MGroup, [191](#)
- GetStoppingPower
 - EPoint, [138](#)
- GetStoppingPowerEq
 - TargetEffect, [260](#)
- GetSubPoint
 - EPoint, [139](#)
- GetSubPoints
 - EPoint, [139](#)
- GetTargetEffect
 - EData, [111](#)
- GetTargetEffectNum
 - EPoint, [139](#)
 - ESegment, [151](#)
- getTargetIntModel
 - TargetIntTab, [268](#)
- GetTargetThickness
 - EPoint, [139](#)
- getTemperatureFile
 - AZURE2.cpp, [497](#)
- GetTempTMatrix
 - GenMatrixFunc, [163](#)
- getTestSegments
 - PlotTab, [214](#)
- GetTMatrixElement
 - GenMatrixFunc, [163](#)
- GetTransformE
 - ALevel, [43](#)
- GetTransformGamma
 - ALevel, [43](#)
- GetTransformIterations
 - ALevel, [44](#)
- GetZ
 - PPair, [218](#)
- GetZ1Z2
 - Interference, [170](#)
- gsl_reactionrate_integrand
 - ReactionRate.cpp, [586](#)
- gsl_reactionrate_integration
 - ReactionRate.cpp, [586](#)
 - ReactionRate.h, [483](#)
- gsl_reactionrate_params, [165](#)
 - compound, [165](#)
 - configure, [165](#)
 - entranceKey, [165](#)
 - exitKey, [165](#)
 - gsl_reactionrate_params, [165](#)
 - temperature, [166](#)
- gsl_whit_function
 - WhitFunc.h, [487](#)
- GSLErrorHandler
 - GSLErrorException, [167](#)
- GSLErrorException, [166](#)
 - ~GSLErrorException, [167](#)
 - GSLErrorHandler, [167](#)
 - GSLErrorException, [167](#)
 - what, [167](#)
- H_dH
 - Coulomb_wave_functions, [94](#)
- H_dH_scaled
 - Coulomb_wave_functions, [94](#)
- hbarc
 - Constants.h, [458](#)
- headerData
 - ChannelsModel, [74](#)
 - LevelsModel, [183](#)
 - PairsModel, [205](#)
 - SegmentsDataModel, [242](#)
 - SegmentsTestModel, [252](#)
 - TargetIntModel, [265](#)
- heavyG
 - PairsData, [201](#)
- heavyJ
 - PairsData, [201](#)
- heavyJText
 - AddPairDialog, [25](#)
- heavyM
 - PairsData, [201](#)
- heavyMText
 - AddPairDialog, [25](#)
- heavyPi
 - PairsData, [201](#)
- heavyPiCombo
 - AddPairDialog, [25](#)
- heavyZ
 - PairsData, [202](#)
- heavyZText
 - AddPairDialog, [25](#)
- highAngle
 - SegmentsDataData, [239](#)
 - SegmentsTestData, [249](#)
- highAngleText
 - AddSegDataDialog, [29](#)
 - AddSegTestDialog, [33](#)
- highEnergy
 - SegmentsDataData, [239](#)
 - SegmentsTestData, [249](#)
- highEnergyText
 - AddSegDataDialog, [29](#)
 - AddSegTestDialog, [33](#)

- IGNORE_ZERO_WIDTHS
 - Config, [86](#)
- ignoreExternalsCheck
 - EditOptionsDialog, [122](#)
- inf_norm
 - complex_functions.H, [277](#)
- InfoDialog, [167](#)
 - InfoDialog, [168](#)
- Initialize
 - CNuc, [80](#)
 - EData, [112](#)
 - EPoint, [139](#)
 - Equation, [145](#)
- initResource
 - main.cpp, [404](#)
- insertRows
 - ChannelsModel, [74](#)
 - LevelsModel, [183](#)
 - PairsModel, [205](#)
 - SegmentsDataModel, [242](#)
 - SegmentsTestModel, [252](#)
 - TargetIntModel, [266](#)
- InTabDocs.cpp
 - setInfoStrings, [389](#)
- integralsfile
 - Config, [88](#)
- IntegratedFermiFunc, [168](#)
 - IntegratedFermiFunc, [169](#)
 - operator(), [169](#)
- IntegrateTargetEffect
 - EPoint, [139](#)
- Interference, [169](#)
 - GetInterferenceType, [170](#)
 - GetM1, [170](#)
 - GetM2, [170](#)
 - GetZ1Z2, [170](#)
 - Interference, [170](#)
- inverse
 - MatrixInv, [189](#)
- InvertMatrices
 - AMatrixFunc, [50](#)
 - GenMatrixFunc, [163](#)
 - RMatrixFunc, [231](#)
- ir
 - NucLine, [196](#)
- is_it_normalized
 - Coulomb_wave_functions, [95](#)
- IsActive
 - TargetEffect, [260](#)
- isActive
 - ExtrapLine, [155](#)
 - LevelsData, [179](#)
 - NucLine, [196](#)
 - SegLine, [236](#)
 - SegmentsDataData, [239](#)
 - SegmentsTestData, [249](#)
 - TargetIntData, [262](#)
- IsAngDist
 - AZUREFBuffer, [56](#)
- IsAngularDist
 - EPoint, [140](#)
 - ESegment, [151](#)
- IsAZUREFBuffer
 - AZUREOutput, [62](#)
- IsChannel
 - JGroup, [173](#)
- isChannel
 - ChannelsModel, [74](#)
- IsChannelCapture
 - ECMGroup, [106](#)
- IsConvolution
 - TargetEffect, [260](#)
- isConvolution
 - TargetIntData, [263](#)
- isConvolutionCheck
 - AddTargetIntDialog, [37](#)
- IsDecay
 - PPair, [218](#)
- isDiff
 - ExtrapLine, [155](#)
 - SegLine, [236](#)
- IsDifferential
 - EPoint, [140](#)
 - ESegment, [151](#)
- isE1
 - Constants.h, [458](#)
- isE2
 - Constants.h, [458](#)
- IsECLLevel
 - ALevel, [44](#)
- IsEntrance
 - PPair, [219](#)
- IsErrorAnalysis
 - EData, [112](#)
- IsExtrap
 - AZUREOutput, [62](#)
- isfinite
 - complex_functions.H, [277](#)
- IsFit
 - EData, [112](#)
- isFixed
 - ChannelsData, [71](#)
 - LevelsData, [179](#)
- IsInRMatrix
 - ALevel, [44](#)
 - JGroup, [173](#)
- IsInSegment
 - ESegment, [151](#)
- IsInterference
 - KLGroup, [178](#)
- IsJGroup
 - CNuc, [81](#)
- IsKGroup
 - Decay, [100](#)
- IsKLGroup
 - Decay, [100](#)

- IsLevel
 - JGroup, [173](#)
- isLevel
 - LevelsModel, [183](#)
- isM1
 - Constants.h, [459](#)
- IsMapped
 - EPoint, [140](#)
- IsMGroup
 - KGroup, [176](#)
- IsMouseFiltered
 - FilteredTextEdit, [157](#)
- IsPair
 - CNuc, [81](#)
- isPair
 - PairsModel, [205](#)
- IsPairKey
 - CNuc, [81](#)
- IsPhase
 - EPoint, [140](#)
 - ESegment, [151](#)
- isQCoefficientCheck
 - AddTargetIntDialog, [37](#)
- IsQCoefficients
 - TargetEffect, [260](#)
- isQCoefficients
 - TargetIntData, [263](#)
- isSegDataLine
 - SegmentsDataModel, [242](#)
- IsSegmentKey
 - EData, [112](#)
- isSegTestLine
 - SegmentsTestModel, [252](#)
- IsTargetEffect
 - EPoint, [140](#)
 - ESegment, [151](#)
- IsTargetIntegration
 - TargetEffect, [260](#)
- isTargetIntegration
 - TargetIntData, [263](#)
- isTargetIntegrationCheck
 - AddTargetIntDialog, [37](#)
- IsTempTMatrix
 - GenMatrixFunc, [163](#)
- IsTotalCapture
 - ESegment, [152](#)
- IsVaryNorm
 - ESegment, [152](#)
- Iterate
 - EData, [112](#)
- Iterations
 - EData, [113](#)
- j1
 - NucLine, [196](#)
- j2
 - NucLine, [196](#)
- j3
 - NucLine, [196](#)
- JGroup, [171](#)
 - AddChannel, [172](#)
 - AddLevel, [172](#)
 - GetChannel, [172](#)
 - GetJ, [172](#)
 - GetLevel, [173](#)
 - GetPi, [173](#)
 - IsChannel, [173](#)
 - IsInRMatrix, [173](#)
 - IsLevel, [173](#)
 - JGroup, [172](#)
 - NumChannels, [174](#)
 - NumLevels, [174](#)
- jValue
 - LevelsData, [179](#)
 - TempTMatrix, [270](#)
- jValueText
 - AddLevelDialog, [23](#)
- KGroup, [174](#)
 - AddECMGroup, [175](#)
 - AddMGroup, [175](#)
 - GetECMGroup, [175](#)
 - GetMGroup, [175](#)
 - GetS, [176](#)
 - GetSp, [176](#)
 - IsMGroup, [176](#)
 - KGroup, [175](#)
 - NumECMGroups, [176](#)
 - NumMGroups, [176](#)
- KLGroup, [177](#)
 - AddInterference, [177](#)
 - GetInterference, [177](#)
 - GetK, [178](#)
 - GetLOrder, [178](#)
 - IsInterference, [178](#)
 - KLGroup, [177](#)
 - NumInterferences, [178](#)
- I
 - Coulomb_wave_functions, [95](#)
 - NucLine, [196](#)
- legendreCheckCombo
 - EditChecksDialog, [119](#)
- levelE
 - NucLine, [197](#)
- levelFix
 - NucLine, [197](#)
- levelID
 - NucLine, [197](#)
- levelIndex
 - ChannelsData, [71](#)
- levelJ
 - NucLine, [197](#)
- levelPi
 - NucLine, [197](#)
- LevelsData, [179](#)
 - energy, [179](#)
 - isActive, [179](#)

- isFixed, 179
- jValue, 179
- piValue, 179
- SIZE, 180
- LevelsHeaderView, 180
 - LevelsHeaderView, 181
 - mouseMoveEvent, 181
 - mousePressEvent, 181
 - mouseReleaseEvent, 181
- LevelsModel, 181
 - columnCount, 182
 - data, 182
 - flags, 182
 - getLevels, 183
 - getSpinLabel, 183
 - headerData, 183
 - insertRows, 183
 - isLevel, 183
 - LevelsModel, 182
 - removeRows, 183
 - rowCount, 184
 - setData, 184
- LevelsTab, 184
 - addLevel, 185
 - calculateChannels, 185
 - editLevel, 186
 - LevelsTab, 185
 - readExistingPair, 186
 - readNewPair, 186
 - readNuclearFile, 186
 - removeLevel, 186
 - reset, 186
 - setPairsModel, 186
 - showInfo, 187
 - updateButtons, 187
 - updateChannelsLevelAdded, 187
 - updateChannelsLevelDeleted, 187
 - updateChannelsLevelEdited, 187
 - updateChannelsPairAddedEdited, 187
 - updateChannelsPairRemoved, 187
 - updateDetails, 188
 - updateFilter, 188
 - updateReducedWidth, 188
 - writeNuclearFile, 188
- lightG
 - PairsData, 202
- lightJ
 - PairsData, 202
- lightJText
 - AddPairDialog, 25
- lightM
 - PairsData, 202
- lightMText
 - AddPairDialog, 25
- lightPi
 - PairsData, 202
- lightPiCombo
 - AddPairDialog, 26
- lightSpeedInCmPerS
 - Constants.h, 459
- lightZ
 - PairsData, 202
- lightZText
 - AddPairDialog, 26
- ILast
 - CoulFunc, 91
- IMatrixCheckCombo
 - EditChecksDialog, 120
- log1p
 - complex_functions.cpp, 289
 - complex_functions.H, 277
- log_Cl_eta_calc
 - complex_functions.cpp, 290
 - complex_functions.H, 277
- log_cut_constant_AS_calc
 - complex_functions.cpp, 290
 - complex_functions.H, 277
- log_cut_constant_CFa_calc
 - complex_functions.cpp, 290
 - complex_functions.H, 277
- log_cut_constant_CFb_calc
 - complex_functions.cpp, 290
 - complex_functions.H, 278
- log_Gamma
 - complex_functions.cpp, 290
 - complex_functions.H, 278
- lowAngle
 - SegmentsDataData, 239
 - SegmentsTestData, 249
- lowAngleText
 - AddSegDataDialog, 29
 - AddSegTestDialog, 33
- lowEnergy
 - SegmentsDataData, 239
 - SegmentsTestData, 249
- lowEnergyText
 - AddSegDataDialog, 29
 - AddSegTestDialog, 33
- lpValue
 - TempTMatrix, 270
- lValue
 - ChannelsData, 71
 - TempTMatrix, 270
- m1
 - NucLine, 197
- m2
 - NucLine, 198
- main
 - AZURE2.cpp, 497
- main.cpp
 - initResource, 404
 - start_gui, 404
- MakePoints
 - EData, 113
- MapData
 - EData, 113

- matrix_c
 - Constants.h, [457](#)
- matrix_r
 - Constants.h, [457](#)
- MatrixInv, [188](#)
 - inverse, [189](#)
 - MatrixInv, [189](#)
- maxA
 - ExtrapLine, [155](#)
 - SegLine, [236](#)
- maxAngDistOrder
 - ExtrapLine, [155](#)
 - SegmentsTestData, [250](#)
- maxE
 - ExtrapLine, [156](#)
 - SegLine, [236](#)
- maxECMult
 - Constants.h, [459](#)
- maxLOrder
 - Config, [88](#)
- maxTemp
 - RateParams, [222](#)
- MGroup, [189](#)
 - GetChNum, [190](#)
 - GetChpNum, [190](#)
 - GetJNum, [190](#)
 - GetStatSpinFactor, [191](#)
 - MGroup, [190](#)
 - SetStatSpinFactor, [191](#)
- minA
 - ExtrapLine, [156](#)
 - SegLine, [236](#)
- minE
 - ExtrapLine, [156](#)
 - SegLine, [236](#)
- minTemp
 - RateParams, [222](#)
- mouseDoubleClickEvent
 - FilteredTextEdit, [157](#)
- mouseMoveEvent
 - LevelsHeaderView, [181](#)
- mousePressEvent
 - FilteredTextEdit, [158](#)
 - LevelsHeaderView, [181](#)
- mouseReleaseEvent
 - LevelsHeaderView, [181](#)
- moveSegDataLineDown
 - SegmentsTab, [246](#)
- moveSegDataLineUp
 - SegmentsTab, [246](#)
- moveSegTestLineDown
 - SegmentsTab, [246](#)
- moveSegTestLineUp
 - SegmentsTab, [246](#)
- multBox
 - AddPairDialog, [26](#)
- NewTempTMatrix
 - GenMatrixFunc, [164](#)
- NFIntegral, [191](#)
 - ~NFIntegral, [192](#)
 - chanRad, [192](#)
 - NFIntegral, [192](#)
 - operator(), [192](#)
 - totalSepE, [192](#)
- noTransformCheck
 - EditOptionsDialog, [122](#)
- nuclearMagnetron
 - Constants.h, [459](#)
- NucLine, [193](#)
 - aa, [194](#)
 - channelFix, [194](#)
 - chRad, [194](#)
 - e2, [194](#)
 - e3, [195](#)
 - ecMultMask, [195](#)
 - entranceSepE, [195](#)
 - g1, [195](#)
 - g2, [195](#)
 - gamma, [195](#)
 - ir, [196](#)
 - isActive, [196](#)
 - j1, [196](#)
 - j2, [196](#)
 - j3, [196](#)
 - l, [196](#)
 - levelE, [197](#)
 - levelFix, [197](#)
 - levelID, [197](#)
 - levelJ, [197](#)
 - levelPi, [197](#)
 - m1, [197](#)
 - m2, [198](#)
 - NucLine, [194](#)
 - pi1, [198](#)
 - pi2, [198](#)
 - pi3, [198](#)
 - pType, [198](#)
 - s, [198](#)
 - sepE, [199](#)
 - z1, [199](#)
 - z2, [199](#)
- NumAZUREBuffers
 - AZUREOutput, [62](#)
- NumChannels
 - JGroup, [174](#)
- NumDecays
 - PPair, [219](#)
- NumECMGroups
 - KGroup, [176](#)
- NumInterferences
 - KLGroup, [178](#)
- NumJGroups
 - CNuc, [81](#)
- NumKGroups
 - Decay, [100](#)
- NumKLGroups

- Decay, [100](#)
- NumLevels
 - JGroup, [174](#)
- NumLocalMappedPoints
 - EPoint, [140](#)
- NumMGroups
 - KGroup, [176](#)
- NumNFIntegrals
 - ALevel, [44](#)
- NumPairs
 - CNuc, [81](#)
- numPairs
 - PairsModel, [205](#)
- numParameters
 - TargetIntData, [263](#)
- numParametersSpin
 - AddTargetIntDialog, [37](#)
- NumPoints
 - ESegment, [152](#)
- numPoints
 - TargetIntData, [263](#)
- numPointsSpin
 - AddTargetIntDialog, [37](#)
- NumQCoefficients
 - TargetEffect, [260](#)
- numQCoefficientSpin
 - AddTargetIntDialog, [37](#)
- NumSegments
 - EData, [113](#)
- NumSubPoints
 - EPoint, [141](#)
 - TargetEffect, [261](#)
- NumTargetEffects
 - EData, [113](#)
- NumTempTMatrices
 - GenMatrixFunc, [164](#)
- ODE_integration, [199](#)
 - ODE_integration, [200](#)
 - operator(), [200](#)
- open
 - AZURESetup, [68](#)
- operator!=
 - complex_functions.H, [278](#)
 - EDatalterator, [117](#)
- operator<
 - RateData, [220](#)
- operator()
 - AZURECalc, [53](#)
 - AZUREMain, [57](#)
 - AZUREOutput, [62](#)
 - CoulFunc, [91](#)
 - ECIntegral, [103](#)
 - EffectiveCharge, [124](#)
 - IntegratedFermiFunc, [169](#)
 - NFIntegral, [192](#)
 - ODE_integration, [200](#)
 - ShftFunc, [256](#)
 - WhitFunc, [273](#)
- operator+
 - complex_functions.H, [279](#), [280](#)
- operator++
 - EDatalterator, [117](#)
- operator-
 - complex_functions.H, [280](#)
- operator/
 - complex_functions.H, [281](#)
- operator==
 - complex_functions.H, [281](#), [282](#)
 - EDatalterator, [117](#)
- operator*
 - complex_functions.H, [279](#)
- outputDir
 - Directories, [101](#)
- outputdir
 - Config, [88](#)
- outputDirectoryText
 - EditDirsDialog, [121](#)
- outStream
 - Config, [88](#)
- overflow
 - TextEditBuffer, [271](#)
- paint
 - RichTextDelegate, [226](#)
- pairAdded
 - PairsTab, [208](#)
- pairEdited
 - PairsTab, [208](#)
- pairIndex
 - ChannelsData, [71](#)
- pairRemoved
 - PairsTab, [208](#)
- PairsData, [200](#)
 - channelRadius, [201](#)
 - ecMultMask, [201](#)
 - excitationEnergy, [201](#)
 - heavyG, [201](#)
 - heavyJ, [201](#)
 - heavyM, [201](#)
 - heavyPi, [201](#)
 - heavyZ, [202](#)
 - lightG, [202](#)
 - lightJ, [202](#)
 - lightM, [202](#)
 - lightPi, [202](#)
 - lightZ, [202](#)
 - pairType, [202](#)
 - seperationEnergy, [202](#)
 - SIZE, [203](#)
- PairsModel, [203](#)
 - columnCount, [204](#)
 - data, [204](#)
 - getPairs, [204](#)
 - getParticleLabel, [204](#)
 - getReactionLabel, [204](#)
 - getReactionLabelTotalCapture, [204](#)
 - getSpinLabel, [205](#)

- headerData, 205
- insertRows, 205
- isPair, 205
- numPairs, 205
- PairsModel, 204
- removeRows, 205
- rowCount, 206
- setData, 206
- PairsTab, 206
 - addPair, 207
 - editPair, 207
 - getPairsModel, 208
 - pairAdded, 208
 - pairEdited, 208
 - pairRemoved, 208
 - PairsTab, 207
 - parseOldECSection, 208
 - removePair, 208
 - showInfo, 208
 - updateButtons, 209
- pairType
 - PairsData, 202
- pairTypeCombo
 - AddPairDialog, 26
- parameterChanged
 - AddTargetIntDialog, 36
- ParameterFlags
 - Config, 86
- parameters
 - TargetIntData, 263
- parameterSpinChanged
 - AddTargetIntDialog, 36
- parametersTable
 - AddTargetIntDialog, 38
- paramfile
 - Config, 88
- paramMask
 - Config, 89
- ParseExternalCapture
 - CNuc, 82
- parseOldECSection
 - PairsTab, 208
- parseOptions
 - AZURE2.cpp, 497
- pathwaysCheckCombo
 - EditChecksDialog, 120
- Penetrability
 - CoulFunc, 91
- PERFORM_ERROR_ANALYSIS
 - Config, 86
- PERFORM_FIT
 - Config, 86
- PEShift
 - CoulFunc, 91
- PEShift_dE
 - CoulFunc, 91
- phaseJ
 - ExtrapLine, 156
- SegLine, 237
- SegmentsDataData, 239
- SegmentsTestData, 250
- phaseJValueLabel
 - AddSegDataDialog, 30
 - AddSegTestDialog, 33
- phaseJValueText
 - AddSegDataDialog, 30
 - AddSegTestDialog, 33
- phaseL
 - ExtrapLine, 156
 - SegLine, 237
 - SegmentsDataData, 240
 - SegmentsTestData, 250
- phaseLValueLabel
 - AddSegDataDialog, 30
 - AddSegTestDialog, 33
- phaseLValueText
 - AddSegDataDialog, 30
 - AddSegTestDialog, 33
- pi
 - Constants.h, 459
- pi1
 - NucLine, 198
- pi2
 - NucLine, 198
- pi3
 - NucLine, 198
- piValue
 - LevelsData, 179
- piValueCombo
 - AddLevelDialog, 23
- PlotEntry, 209
 - ~PlotEntry, 210
 - attach, 210
 - AZUREPlot, 211
 - detach, 210
 - PlotEntry, 210
 - readData, 210
 - type, 210
- PlotPoint, 211
 - angle, 211
 - dataCrossSection, 211
 - dataErrorCrossSection, 211
 - dataErrorSFactor, 212
 - dataSFactor, 212
 - energy, 212
 - excitationEnergy, 212
 - fitCrossSection, 212
 - fitSFactor, 212
- PlotTab, 213
 - AZUREPlot, 215
 - draw, 214
 - getDataSegments, 214
 - getTestSegments, 214
 - PlotTab, 213
 - reset, 214
 - showInfo, 214

- xAxisLogScaleChanged, [214](#)
 - xAxisTypeChanged, [214](#)
 - yAxisLogScaleChanged, [214](#)
 - yAxisTypeChanged, [215](#)
- point
 - EDatalterator, [117](#)
 - EnergyMap, [126](#)
- PPair, [215](#)
 - AddDecay, [216](#)
 - GetChRad, [216](#)
 - GetDecay, [216](#)
 - GetExE, [216](#)
 - GetG, [217](#)
 - GetI1I2Factor, [217](#)
 - GetJ, [217](#)
 - GetM, [217](#)
 - GetPairKey, [217](#)
 - GetPi, [217](#)
 - GetPType, [218](#)
 - GetRedMass, [218](#)
 - GetSepE, [218](#)
 - GetZ, [218](#)
 - IsDecay, [218](#)
 - IsEntrance, [219](#)
 - NumDecays, [219](#)
 - PPair, [216](#)
 - SetEntrance, [219](#)
- precision
 - complex_functions.H, [282](#)
- print
 - AZUREPlot, [66](#)
- PrintAngularDists
 - CNuc, [82](#)
- PrintBoundaryConditions
 - CNuc, [82](#)
- PrintCoulombAmplitude
 - EData, [113](#)
- PrintData
 - EData, [114](#)
- PrintEDependentValues
 - EData, [114](#)
- printHelp
 - AZURE2.cpp, [497](#)
- PrintLegendreP
 - EData, [114](#)
- PrintNuc
 - CNuc, [82](#)
- PrintPathways
 - CNuc, [82](#)
- PrintTransformParams
 - CNuc, [83](#)
- processCommand
 - AZURE2.cpp, [497](#)
- pType
 - NucLine, [198](#)
- Q_DECLARE_METATYPE
 - TargetIntModel.h, [347](#)
- qCoefficientChanged
 - AddTargetIntDialog, [36](#)
- qCoefficientCheckChanged
 - AddTargetIntDialog, [36](#)
- qCoefficients
 - TargetIntData, [263](#)
- qCoefficientSpinChanged
 - AddTargetIntDialog, [36](#)
- qCoefficientTable
 - AddTargetIntDialog, [38](#)
- Racah
 - AngCoeff, [51](#)
- radiusLast
 - CoulFunc, [92](#)
- radType
 - ChannelsData, [71](#)
- rate
 - RateData, [221](#)
- RateData, [219](#)
 - operator<, [220](#)
 - rate, [221](#)
 - RateData, [220](#)
 - temperature, [221](#)
- RateParams, [221](#)
 - entrancePair, [222](#)
 - exitPair, [222](#)
 - maxTemp, [222](#)
 - minTemp, [222](#)
 - temperatureFile, [222](#)
 - tempStep, [222](#)
 - useFile, [223](#)
- rateParams
 - Config, [89](#)
- ReactionRate, [223](#)
 - CalculateFileRates, [224](#)
 - CalculateRates, [224](#)
 - compound, [224](#)
 - configure, [224](#)
 - entranceKey, [224](#)
 - exitKey, [225](#)
 - ReactionRate, [224](#)
 - WriteRates, [225](#)
- ReactionRate.cpp
 - gsl_reactionrate_integrand, [586](#)
 - gsl_reactionrate_integration, [586](#)
- ReactionRate.h
 - gsl_reactionrate_integration, [483](#)
- ReadConfigFile
 - Config, [87](#)
- readData
 - PlotEntry, [210](#)
- readExistingPair
 - LevelsTab, [186](#)
- readFile
 - TargetIntTab, [268](#)
- readNewPair
 - LevelsTab, [186](#)
- readNuclearFile
 - LevelsTab, [186](#)

- readSegDataFile
 - SegmentsTab, 246
- readSegmentFile
 - AZURE2.cpp, 498
 - AZURESetup.cpp, 368
- readSegTestFile
 - SegmentsTab, 246
- ReadTargetEffectsFile
 - EData, 114
- ReadUserParameters
 - AZUREParams, 64
- readyToRun
 - AZUREMainThread, 59
- redmass
 - CoulFunc, 92
 - WhitFunc, 273
- reducedWidth
 - ChannelsData, 72
- reducedWidthText
 - ChannelDetails, 70
- removeLevel
 - LevelsTab, 186
- removePair
 - PairsTab, 208
- removeRows
 - ChannelsModel, 74
 - LevelsModel, 183
 - PairsModel, 205
 - SegmentsDataModel, 242
 - SegmentsTestModel, 252
 - TargetIntModel, 266
- Reset
 - Config, 87
- reset
 - LevelsTab, 186
 - PlotTab, 214
 - RunTab, 232
 - SegmentsTab, 247
 - TargetIntTab, 268
- ResetIterations
 - EData, 114
- RichTextDelegate, 225
 - paint, 226
 - sizeHint, 226
- RMatrixFunc, 226
 - AddRLInvMatrix, 228
 - AddRLInvRMatrixElement, 228
 - AddRLMatrixElement, 228
 - AddRMatrixElement, 228
 - CalculateCrossSection, 229
 - CalculateTMatrix, 229
 - ClearMatrices, 229
 - compound, 229
 - configure, 229
 - FillMatrices, 230
 - GetJSpecRLMatrix, 230
 - GetRLInvMatrixElement, 230
 - GetRLInvRMatrixElement, 230
 - GetRLMatrixElement, 230
 - GetRMatrixElement, 231
 - InvertMatrices, 231
 - RMatrixFunc, 228
- ROOT, 17
- ROOT::Minuit2, 17
- rowCount
 - ChannelsModel, 75
 - LevelsModel, 184
 - PairsModel, 206
 - SegmentsDataModel, 243
 - SegmentsTestModel, 253
 - TargetIntModel, 266
- run
 - AZUREMainThread, 59
 - AZUREMainThreadWorker, 60
- RunTab, 231
 - AZUREMainThread, 233
 - AZURESetup, 233
 - reset, 232
 - RunTab, 232
 - showInfo, 232
- s
 - NucLine, 198
- SaveAndRun
 - AZURESetup, 68
- screenCheckMask
 - Config, 89
- secondPair
 - SegPairs, 254
- SegDataProxyModel, 233
 - data, 234
 - SegDataProxyModel, 233
- SegLine, 234
 - dataFile, 235
 - dataNorm, 235
 - dataNormError, 235
 - entranceKey, 235
 - exitKey, 235
 - isActive, 236
 - isDiff, 236
 - maxA, 236
 - maxE, 236
 - minA, 236
 - minE, 236
 - phaseJ, 237
 - phaseL, 237
 - SegLine, 235
 - varyNorm, 237
- segment
 - EDataIterator, 117
 - EnergyMap, 126
- SegmentsDataData, 237
 - dataFile, 238
 - dataNorm, 238
 - dataNormError, 238
 - dataType, 238
 - entrancePairIndex, 239

- exitPairIndex, [239](#)
- highAngle, [239](#)
- highEnergy, [239](#)
- isActive, [239](#)
- lowAngle, [239](#)
- lowEnergy, [239](#)
- phaseJ, [239](#)
- phaseL, [240](#)
- SIZE, [240](#)
- varyNorm, [240](#)
- SegmentsDataModel, [240](#)
 - columnCount, [241](#)
 - data, [241](#)
 - flags, [241](#)
 - getLines, [242](#)
 - getReactionLabel, [242](#)
 - headerData, [242](#)
 - insertRows, [242](#)
 - isSegDataLine, [242](#)
 - removeRows, [242](#)
 - rowCount, [243](#)
 - SegmentsDataModel, [241](#)
 - setData, [243](#)
 - setPairsModel, [243](#)
- segmentsList
 - TargetIntData, [263](#)
- segmentsListText
 - AddTargetIntDialog, [38](#)
- SegmentsTab, [243](#)
 - addSegDataLine, [244](#)
 - addSegTestLine, [245](#)
 - deleteSegDataLine, [245](#)
 - deleteSegTestLine, [245](#)
 - editSegDataLine, [245](#)
 - editSegTestLine, [245](#)
 - getSegmentsDataModel, [245](#)
 - getSegmentsTestModel, [246](#)
 - moveSegDataLineDown, [246](#)
 - moveSegDataLineUp, [246](#)
 - moveSegTestLineDown, [246](#)
 - moveSegTestLineUp, [246](#)
 - readSegDataFile, [246](#)
 - readSegTestFile, [246](#)
 - reset, [247](#)
 - SegmentsTab, [244](#)
 - setPairsModel, [247](#)
 - showInfo, [247](#)
 - updateSegDataButtons, [247](#)
 - updateSegTestButtons, [247](#)
 - writeSegDataFile, [247](#)
 - writeSegTestFile, [247](#)
- SegmentsTestData, [248](#)
 - angleStep, [248](#)
 - dataType, [248](#)
 - energyStep, [249](#)
 - entrancePairIndex, [249](#)
 - exitPairIndex, [249](#)
 - highAngle, [249](#)
 - highEnergy, [249](#)
 - isActive, [249](#)
 - lowAngle, [249](#)
 - lowEnergy, [249](#)
 - maxAngDistOrder, [250](#)
 - phaseJ, [250](#)
 - phaseL, [250](#)
 - SIZE, [250](#)
- SegmentsTestModel, [250](#)
 - columnCount, [251](#)
 - data, [251](#)
 - flags, [251](#)
 - getLines, [252](#)
 - getReactionLabel, [252](#)
 - headerData, [252](#)
 - insertRows, [252](#)
 - isSegTestLine, [252](#)
 - removeRows, [252](#)
 - rowCount, [253](#)
 - SegmentsTestModel, [251](#)
 - setData, [253](#)
 - setPairsModel, [253](#)
- SegPairs, [253](#)
 - firstPair, [254](#)
 - secondPair, [254](#)
- SegTestProxyModel, [254](#)
 - data, [255](#)
 - filterAcceptsRow, [255](#)
 - SegTestProxyModel, [254](#)
- sepE
 - NucLine, [199](#)
- seperationEnergy
 - PairsData, [202](#)
- seperationEnergyText
 - AddPairDialog, [26](#)
- SetAngularDists
 - EPoint, [141](#)
- SetBigGamma
 - ALevel, [44](#)
- SetBoundaryCondition
 - AChannel, [21](#)
- setChooseFile
 - AddSegDataDialog, [28](#)
- SetCoulombAmplitude
 - EPoint, [141](#)
- setData
 - ChannelsModel, [75](#)
 - LevelsModel, [184](#)
 - PairsModel, [206](#)
 - SegmentsDataModel, [243](#)
 - SegmentsTestModel, [253](#)
 - TargetIntModel, [266](#)
- SetE
 - ALevel, [44](#)
- SetECPParams
 - ALevel, [45](#)
- SetEnd
 - EDataIterator, [118](#)

- SetEntrance
 - PPair, 219
- SetErrorAnalysis
 - EData, 115
- SetErrorDef
 - AZURECalc, 54
- SetExitKey
 - EPoint, 141
 - ESegment, 152
- SetExternalGamma
 - ALevel, 45
- SetExtrap
 - AZUREOutput, 63
- SetFit
 - EData, 115
- SetFitCrossSection
 - EPoint, 141
- SetFitE
 - ALevel, 45
- SetFitGamma
 - ALevel, 45
- SetGamma
 - ALevel, 45
- SetGeometricalFactor
 - EPoint, 141
- setInfoStrings
 - InTabDocs.cpp, 389
- SetIsTotalCapture
 - ESegment, 152
- setLast
 - CoulFunc, 92
- setLineEdit
 - ChooseFileButton, 76
- SetMap
 - EPoint, 142
- SetMaxLValue
 - CNuc, 83
- SetMouseFiltered
 - FilteredTextEdit, 158
- SetNorm
 - ESegment, 152
- setNormParam
 - ChannelDetails, 70
- SetNormParamOffset
 - EData, 115
- SetNumSubPoints
 - TargetEffect, 261
- setPairsModel
 - ChannelsModel, 75
 - LevelsTab, 186
 - SegmentsDataModel, 243
 - SegmentsTab, 247
 - SegmentsTestModel, 253
- SetParameter
 - Equation, 145
- SetParentData
 - EPoint, 142
- SetSegmentChiSquared
 - ESegment, 153
- SetSegmentKey
 - ESegment, 153
- SetSFactorConversion
 - EPoint, 142
- SetShiftFunction
 - ALevel, 46
- SetSigma
 - TargetEffect, 261
- SetSqrtNFFactor
 - ALevel, 46
- SetStatSpinFactor
 - ECMGroup, 107
 - MGroup, 191
- SetStoppingPower
 - EPoint, 142
- SetTargetEffectNum
 - EPoint, 142
 - ESegment, 153
- SetTargetThickness
 - EPoint, 143
- SetTransformE
 - ALevel, 46
- SetTransformGamma
 - ALevel, 46
- SetTransformIterations
 - ALevel, 46
- SetVaryNorm
 - ESegment, 153
- setXAxisLog
 - AZUREPlot, 66
- setXAxisType
 - AZUREPlot, 66
- setYAxisLog
 - AZUREPlot, 66
- setYAxisType
 - AZUREPlot, 66
- ShftFunc, 255
 - ~ShftFunc, 256
 - EnergyDerivative, 256
 - operator(), 256
 - ShftFunc, 256
- showInfo
 - LevelsTab, 187
 - PairsTab, 208
 - PlotTab, 214
 - RunTab, 232
 - SegmentsTab, 247
 - TargetIntTab, 268
- sigma
 - TargetIntData, 264
- sigma_l_calc
 - complex_functions.cpp, 290
 - complex_functions.H, 282
- sigmaText
 - AddTargetIntDialog, 38
- SIGN
 - complex_functions.H, 276

- sin_chi_calc
 - complex_functions.cpp, 291
 - complex_functions.H, 282
- SIZE
 - ChannelsData, 72
 - LevelsData, 180
 - PairsData, 203
 - SegmentsDataData, 240
 - SegmentsTestData, 250
 - TargetIntData, 264
- sizeHint
 - RichTextDelegate, 226
- SortPathways
 - CNuc, 83
- sqrt_precision
 - complex_functions.H, 282
- start_gui
 - main.cpp, 404
- startMessage
 - AZURE2.cpp, 498
 - AZURESetup.cpp, 369
- stopAZURE
 - AZUREMainThread, 59
- stopFlag
 - Config, 89
- stoppingPowerEq
 - TargetIntData, 264
- stoppingPowerEqText
 - AddTargetIntDialog, 38
- sValue
 - ChannelsData, 72
- sync
 - TextEditBuffer, 271
- SyntaxError, 257
 - ~SyntaxError, 257
 - SyntaxError, 257
 - what, 258
- TargetEffect, 258
 - convolutionRange, 262
 - GetConvolutionFactor, 259
 - GetDensity, 259
 - GetQCoefficient, 259
 - GetSegmentsList, 259
 - GetSigma, 259
 - GetStoppingPowerEq, 260
 - IsActive, 260
 - IsConvolution, 260
 - IsQCoefficients, 260
 - IsTargetIntegration, 260
 - NumQCoefficients, 260
 - NumSubPoints, 261
 - SetNumSubPoints, 261
 - SetSigma, 261
 - TargetEffect, 259
 - TargetThickness, 261
- targetIntCheckChanged
 - AddTargetIntDialog, 38
- TargetIntData, 262
 - density, 262
 - isActive, 262
 - isConvolution, 263
 - isQCoefficients, 263
 - isTargetIntegration, 263
 - numParameters, 263
 - numPoints, 263
 - parameters, 263
 - qCoefficients, 263
 - segmentsList, 263
 - sigma, 264
 - SIZE, 264
 - stoppingPowerEq, 264
- TargetIntModel, 264
 - columnCount, 265
 - data, 265
 - flags, 265
 - getLines, 265
 - headerData, 265
 - insertRows, 266
 - removeRows, 266
 - rowCount, 266
 - setData, 266
 - TargetIntModel, 265
- TargetIntModel.h
 - Q_DECLARE_METATYPE, 347
- TargetIntTab, 267
 - addLine, 267
 - deleteLine, 268
 - editLine, 268
 - getTargetIntModel, 268
 - readFile, 268
 - reset, 268
 - showInfo, 268
 - TargetIntTab, 267
 - updateButtons, 268
 - writeFile, 269
- TargetThickness
 - TargetEffect, 261
- temperature
 - gsl_reactionrate_params, 166
 - RateData, 221
- temperatureFile
 - RateParams, 222
- tempParameters
 - AddTargetIntDialog, 38
- tempQCoefficients
 - AddTargetIntDialog, 38
- tempStep
 - RateParams, 222
- TempTMatrix, 269
 - jValue, 270
 - IpValue, 270
 - IValue, 270
 - TMatrix, 270
- TextEditBuffer, 270
 - overflow, 271
 - sync, 271

- TextEditBuffer, 271
- updateLog, 271
- TMatrix
 - TempTMatrix, 270
- tmatrix_
 - GenMatrixFunc, 164
- totalCaptureLabel
 - AddSegDataDialog, 30
 - AddSegTestDialog, 34
- totalSepE
 - NFIntegral, 192
- trackerTextF
 - AZUREZoomer, 69
- TRANSFORM_PARAMETERS
 - Config, 86
- TransformIn
 - CNuc, 83
- TransformOut
 - CNuc, 83
- type
 - PlotEntry, 210
- uconv
 - Constants.h, 459
- Up
 - AZURECalc, 54
- update
 - AZUREPlot, 66
- updateButtons
 - LevelsTab, 187
 - PairsTab, 209
 - TargetIntTab, 268
- updateChannelsLevelAdded
 - LevelsTab, 187
- updateChannelsLevelDeleted
 - LevelsTab, 187
- updateChannelsLevelEdited
 - LevelsTab, 187
- updateChannelsPairAddedEdited
 - LevelsTab, 187
- updateChannelsPairRemoved
 - LevelsTab, 187
- updateDetails
 - LevelsTab, 188
- updateFilter
 - LevelsTab, 188
- updateLightParticle
 - AddPairDialog, 24
- updateLog
 - TextEditBuffer, 271
- updateReducedWidth
 - LevelsTab, 188
- updateSegDataButtons
 - SegmentsTab, 247
- updateSegTestButtons
 - SegmentsTab, 247
- USE_AMATRIX
 - Config, 86
- USE_BRUNE_FORMALISM
 - Config, 86
- USE_EXTERNAL_CAPTURE
 - Config, 86
- USE_GSL_COULOMB_FUNC
 - Config, 86
- USE_LONGWAVELENGTH_APPROX
 - Config, 86
- USE_PREVIOUS_INTEGRALS
 - Config, 86
- USE_PREVIOUS_PARAMETERS
 - Config, 86
- USE_RMC_FORMALISM
 - Config, 86
- useBruneCheck
 - EditOptionsDialog, 122
- useFile
 - RateParams, 223
- useGSLCoulCheck
 - EditOptionsDialog, 122
- useRMCCheck
 - EditOptionsDialog, 122
- varyNorm
 - SegLine, 237
 - SegmentsDataData, 240
- varyNormChanged
 - AddSegDataDialog, 28
- varyNormCheck
 - AddSegDataDialog, 30
- vector_c
 - Constants.h, 457
- vector_matrix_c
 - Constants.h, 457
- vector_matrix_r
 - Constants.h, 457
- vector_r
 - Constants.h, 458
- welcomeMessage
 - AZURE2.cpp, 498
- what
 - GSLError, 167
 - SyntaxError, 258
- WhitFunc, 272
 - operator(), 273
 - redmass, 273
 - WhitFunc, 272
 - z1, 273
 - z2, 273
- WhitFunc.h
 - gsl_whit_function, 487
- write
 - FilteredTextEdit, 158
- writeFile
 - TargetIntTab, 269
- writeNuclearFile
 - LevelsTab, 188
- WriteOutputFiles
 - EData, 115

- WriteParameterErrors
 - AZUREParams, [64](#)
- WriteRates
 - ReactionRate, [225](#)
- writeSegDataFile
 - SegmentsTab, [247](#)
- writeSegTestFile
 - SegmentsTab, [247](#)
- WriteUserParameters
 - AZUREParams, [64](#)
- xAxisLogScaleChanged
 - PlotTab, [214](#)
- xAxisTypeChanged
 - PlotTab, [214](#)
- yAxisLogScaleChanged
 - PlotTab, [214](#)
- yAxisTypeChanged
 - PlotTab, [215](#)
- z1
 - CoulFunc, [92](#)
 - NucLine, [199](#)
 - WhitFunc, [273](#)
- z2
 - CoulFunc, [92](#)
 - NucLine, [199](#)
 - WhitFunc, [273](#)