

libatomprobe

Generated by Doxygen 1.8.13

Contents

- 1 Namespace Index** **1**
 - 1.1 Namespace List 1

- 2 Hierarchical Index** **3**
 - 2.1 Class Hierarchy 3

- 3 Class Index** **5**
 - 3.1 Class List 5

- 4 File Index** **7**
 - 4.1 File List 7

- 5 Namespace Documentation** **11**
 - 5.1 AtomProbe Namespace Reference 11
 - 5.1.1 Typedef Documentation 21
 - 5.1.1.1 ATO_ENTRY 21
 - 5.1.1.2 IONHIT 21
 - 5.1.2 Enumeration Type Documentation 21
 - 5.1.2.1 anonymous enum 21
 - 5.1.2.2 anonymous enum 22
 - 5.1.2.3 anonymous enum 22
 - 5.1.2.4 anonymous enum 22
 - 5.1.2.5 anonymous enum 23
 - 5.1.2.6 anonymous enum 23
 - 5.1.2.7 anonymous enum 24

5.1.2.8	anonymous enum	25
5.1.2.9	anonymous enum	25
5.1.2.10	anonymous enum	26
5.1.2.11	anonymous enum	26
5.1.2.12	anonymous enum	26
5.1.2.13	anonymous enum	27
5.1.2.14	anonymous enum	27
5.1.2.15	anonymous enum	27
5.1.2.16	anonymous enum	27
5.1.2.17	anonymous enum	28
5.1.2.18	anonymous enum	28
5.1.2.19	anonymous enum	29
5.1.2.20	anonymous enum	29
5.1.2.21	anonymous enum	29
5.1.2.22	anonymous enum	30
5.1.2.23	anonymous enum	30
5.1.2.24	anonymous enum	30
5.1.2.25	anonymous enum	31
5.1.2.26	PointDir	31
5.1.3	Function Documentation	31
5.1.3.1	accumulateCorrelationHistogram()	31
5.1.3.2	andersonDarlingStatistic()	32
5.1.3.3	antiRotateMatch()	33
5.1.3.4	applyQuaternionRotation()	33
5.1.3.5	askAssert()	33
5.1.3.6	buildFrequencyTable()	33
5.1.3.7	castVoxels()	34
5.1.3.8	checkMassRangingCorrectness()	34
5.1.3.9	chunkLoadEposFile()	35
5.1.3.10	computeComposition()	35

5.1.3.11	computeConvexHull()	36
5.1.3.12	computeIonDistAdjacency()	36
5.1.3.13	computeRangeAdjacency()	37
5.1.3.14	computeRotationMatrix()	38
5.1.3.15	convertEPOStoPos()	39
5.1.3.16	convertFileStringToCanonical()	39
5.1.3.17	convertFileStringToNative()	40
5.1.3.18	convertMassToMol()	40
5.1.3.19	convertMolToMass()	41
5.1.3.20	correlationHistogram()	41
5.1.3.21	countDataDistanceWeight()	42
5.1.3.22	countIntensityEvents()	43
5.1.3.23	createMassBackground()	43
5.1.3.24	cumTrapezoid()	44
5.1.3.25	det3by3()	44
5.1.3.26	Determinant()	45
5.1.3.27	diff()	45
5.1.3.28	digitString()	45
5.1.3.29	distanceToFacet()	46
5.1.3.30	distanceToSegment()	47
5.1.3.31	doFitBackground()	47
5.1.3.32	doHull()	48
5.1.3.33	dotProduct()	49
5.1.3.34	edgeldx()	49
5.1.3.35	estimateRank()	50
5.1.3.36	filterBySolutionPPM()	50
5.1.3.37	filterPeakNeedBiggerObs()	51
5.1.3.38	findMaxLessThanOrEq()	51
5.1.3.39	findNearVertices() [1/2]	52
5.1.3.40	findNearVertices() [2/2]	52

5.1.3.41	findOverlaps() [1/4]	52
5.1.3.42	findOverlaps() [2/4]	53
5.1.3.43	findOverlaps() [3/4]	53
5.1.3.44	findOverlaps() [4/4]	54
5.1.3.45	findOverlapsFromSpectra() [1/2]	54
5.1.3.46	findOverlapsFromSpectra() [2/2]	54
5.1.3.47	findPeaks()	55
5.1.3.48	fixedRecordChunkReader()	56
5.1.3.49	fixedRecordReader()	56
5.1.3.50	fourDeterminant()	57
5.1.3.51	fpeek()	57
5.1.3.52	freeConvexHull()	58
5.1.3.53	gauss_ratio_pdf()	58
5.1.3.54	gcd()	58
5.1.3.55	genColString() [1/2]	58
5.1.3.56	genColString() [2/2]	59
5.1.3.57	generate1DAxialDistHist()	59
5.1.3.58	generate1DAxialDistHistSweep()	60
5.1.3.59	getAtoErrString()	61
5.1.3.60	getFileSize()	62
5.1.3.61	getFitErrorMsg()	62
5.1.3.62	getHardAssert()	62
5.1.3.63	getPosFileErrString()	62
5.1.3.64	getRangeMolecule()	63
5.1.3.65	getRecordReadErrString()	63
5.1.3.66	GetReducedHullPts()	63
5.1.3.67	gsl_determinant()	64
5.1.3.68	gsl_matrix_mult()	64
5.1.3.69	gsl_print_matrix()	65
5.1.3.70	gsl_print_vector()	65

5.1.3.71	hexStrToUChar()	65
5.1.3.72	incrementCorrelationHist()	65
5.1.3.73	incrementDataDistanceWeight()	66
5.1.3.74	intersect_RayTriangle()	66
5.1.3.75	isNotDirectory()	67
5.1.3.76	kahansum()	68
5.1.3.77	leastSquaresDeconvolve()	68
5.1.3.78	linearHistogram()	69
5.1.3.79	linkIdentifiers()	69
5.1.3.80	loadATOFile()	69
5.1.3.81	loadEposFile()	70
5.1.3.82	loadPosFile() [1/2]	71
5.1.3.83	loadPosFile() [2/2]	72
5.1.3.84	loadTapsimBinFile()	72
5.1.3.85	loadTextData()	73
5.1.3.86	lowercase()	73
5.1.3.87	makeHistogram()	74
5.1.3.88	marchingCubes()	74
5.1.3.89	matchComposedName()	75
5.1.3.90	maxExplainedFraction() [1/2]	76
5.1.3.91	maxExplainedFraction() [2/2]	76
5.1.3.92	meanAndStdev()	77
5.1.3.93	myStrDup()	78
5.1.3.94	nullBackground()	78
5.1.3.95	nullifyMarker()	78
5.1.3.96	numericalEstimateGaussRatioConf()	78
5.1.3.97	numericalEstimatePoissRatioConf()	79
5.1.3.98	numericalEstimateSkellamConf()	80
5.1.3.99	onlyDir()	80
5.1.3.100	onlyFilename()	80

5.1.3.101 operator<()	81
5.1.3.102 operator<<() [1/3]	81
5.1.3.103 operator<<() [2/3]	81
5.1.3.104 operator<<() [3/3]	82
5.1.3.105 pairContains()	82
5.1.3.106 pairOverlaps()	82
5.1.3.107 parseColString()	83
5.1.3.108 parseCompositionData()	83
5.1.3.109 poissonConfidenceObservation()	84
5.1.3.110 popLocale()	85
5.1.3.111 pushLocale()	85
5.1.3.112 pyramidVol()	85
5.1.3.113 quat_get_rot_quat()	86
5.1.3.114 quat_invert()	86
5.1.3.115 quat_mult_no_second_a()	86
5.1.3.116 quat_pointmult()	87
5.1.3.117 quat_rot() [1/2]	87
5.1.3.118 quat_rot() [2/2]	87
5.1.3.119 quat_rot_apply_quat()	88
5.1.3.120 quat_rot_array() [1/2]	89
5.1.3.121 quat_rot_array() [2/2]	89
5.1.3.122 randomIndices()	90
5.1.3.123 randomSelect()	90
5.1.3.124 rangeOverlaps()	91
5.1.3.125 rangeToStr()	91
5.1.3.126 readEposRecord()	91
5.1.3.127 readPosapOps()	92
5.1.3.128 reconstructTest()	93
5.1.3.129 removeElements()	93
5.1.3.130 rotateMatch()	93

5.1.3.131 sampleIons()	94
5.1.3.132 savePosFile() [1/2]	94
5.1.3.133 savePosFile() [2/2]	95
5.1.3.134 saveTapsimBin()	95
5.1.3.135 selectElements()	96
5.1.3.136 setHardAssert()	96
5.1.3.137 signedDistanceToFacet()	96
5.1.3.138 signVal()	97
5.1.3.139 solveLeastSquares()	97
5.1.3.140 SphericProjectionEqn()	98
5.1.3.141 splitStrsRef() [1/2]	98
5.1.3.142 splitStrsRef() [2/2]	98
5.1.3.143 stlStrToStlWStr()	99
5.1.3.144 stlWStrToStlStr()	99
5.1.3.145 strAppend()	99
5.1.3.146 stream_cast()	100
5.1.3.147 strhas()	101
5.1.3.148 stripChars()	101
5.1.3.149 stripWhite()	101
5.1.3.150 stripZeroEntries()	102
5.1.3.151 sumVoxels()	102
5.1.3.152 tabs()	102
5.1.3.153 tolEqual()	103
5.1.3.154 transposeVector()	103
5.1.3.155 trilsDegenerate()	103
5.1.3.156 ucharToHexStr()	104
5.1.3.157 uppercase()	104
5.1.3.158 vectorMultiErase()	105
5.1.3.159 vectorPointDir()	105
5.1.3.160 weightedMean()	105

5.1.3.161	XMLFreeDoc()	106
5.1.3.162	XMLGetNextElemAttrib()	106
5.1.3.163	XMLHelpFwdNotElem()	106
5.1.3.164	XMLHelpFwdToElem()	107
5.1.3.165	XMLHelpFwdToList()	107
5.1.3.166	XMLHelpGetProp()	107
5.1.3.167	XMLHelpGetText() [1/2]	108
5.1.3.168	XMLHelpGetText() [2/2]	108
5.1.3.169	XMLHelpNextType()	108
5.1.3.170	zechConfidenceLimits()	109
5.1.3.171	zechRoot()	109
5.1.4	Variable Documentation	110
5.1.4.1	a2iTriangleConnectionTable	110
5.1.4.2	ABUNDANCE_ERROR	110
5.1.4.3	aiCubeEdgeFlags	110
5.1.4.4	ATO_ERR_STRINGS	111
5.1.4.5	edgeRemap	111
5.1.4.6	ELEM_FOUR_NODE_TETRAHEDRON	111
5.1.4.7	ELEM_SINGLE_NODE_POINT	111
5.1.4.8	ELEM_THREE_NODE_TRIANGLE	112
5.1.4.9	ELEM_TWO_NODE_LINE	112
5.1.4.10	elementList	112
5.1.4.11	EPOS_RECORD_SIZE	112
5.1.4.12	hardAssert	113
5.1.4.13	HULL_GRAB_SIZE	113
5.1.4.14	libVersion	113
5.1.4.15	MAX_LINE_SIZE	113
5.1.4.16	MAX_RANGEFILE_SIZE	113
5.1.4.17	maximumLinearTable	114
5.1.4.18	MESH_LOAD_ERRS	114
5.1.4.19	MULTIRANGE_FORMAT_VERSION	114
5.1.4.20	NUM_ELEMENTS	114
5.1.4.21	OPS_ENUM_ERRSTRINGS	115
5.1.4.22	PROGRESS_REDUCE	115
5.1.4.23	qhullInited	115
5.1.4.24	randGen	115
5.1.4.25	RANGE_EXTS	116
5.1.4.26	RECORDREAD_ERR_STRINGS	116
5.1.4.27	VERTEX_OFFSET	116

6	Class Documentation	117
6.1	AtomProbe::AbundanceData Class Reference	117
6.1.1	Detailed Description	118
6.1.2	Member Function Documentation	118
6.1.2.1	abundanceBetweenLimits()	119
6.1.2.2	elementName()	119
6.1.2.3	elementNames()	120
6.1.2.4	generateGroupedIsotopeDist()	120
6.1.2.5	generateIsotopeDist()	121
6.1.2.6	generateSingleAtomDist()	122
6.1.2.7	getAtomicNumber()	122
6.1.2.8	getErrorText()	122
6.1.2.9	getMajorIsotopeFromElemIdx()	122
6.1.2.10	getNearestCharge()	123
6.1.2.11	getNominalMass()	123
6.1.2.12	getSymbolIndices() [1/2]	124
6.1.2.13	getSymbolIndices() [2/2]	124
6.1.2.14	isotope()	124
6.1.2.15	isotopeIndex() [1/2]	125
6.1.2.16	isotopeIndex() [2/2]	125
6.1.2.17	isotopes()	125
6.1.2.18	numElements()	126
6.1.2.19	numIsotopes()	126
6.1.2.20	open()	126
6.1.2.21	symbolIdxFromAtomicNumber()	127
6.1.2.22	symbolIndex()	127
6.2	AtomProbe::ATO_ENTRY Struct Reference	128
6.2.1	Detailed Description	128
6.2.2	Member Data Documentation	128
6.2.2.1	approxPulse	128

6.2.2.2	detectorX	128
6.2.2.3	detectorY	128
6.2.2.4	mass	129
6.2.2.5	pulseVoltage	129
6.2.2.6	tof	129
6.2.2.7	voltage	129
6.2.2.8	x	129
6.2.2.9	y	129
6.2.2.10	z	130
6.3	AtomProbe::AxisCompare Class Reference	130
6.3.1	Detailed Description	130
6.3.2	Constructor & Destructor Documentation	130
6.3.2.1	AxisCompare()	130
6.3.3	Member Function Documentation	130
6.3.3.1	operator()()	131
6.3.3.2	setAxis()	131
6.4	AxisCompareExact Class Reference	131
6.4.1	Detailed Description	131
6.4.2	Member Function Documentation	131
6.4.2.1	operator()()	132
6.4.2.2	setAxis()	132
6.5	AtomProbe::BACKGROUND_PARAMS Struct Reference	132
6.5.1	Detailed Description	132
6.5.2	Member Enumeration Documentation	132
6.5.2.1	anonymous enum	132
6.5.3	Member Data Documentation	133
6.5.3.1	binWidth	133
6.5.3.2	intensity	133
6.5.3.3	massEnd	133
6.5.3.4	massStart	134

6.5.3.5	mode	134
6.5.3.6	stdev	134
6.6	AtomProbe::BodyCentredCubicGen Class Reference	134
6.6.1	Detailed Description	135
6.6.2	Constructor & Destructor Documentation	135
6.6.2.1	BodyCentredCubicGen()	135
6.6.3	Member Function Documentation	136
6.6.3.1	generateLattice()	136
6.7	AtomProbe::BoundCube Class Reference	136
6.7.1	Detailed Description	138
6.7.2	Constructor & Destructor Documentation	138
6.7.2.1	BoundCube() [1/4]	138
6.7.2.2	BoundCube() [2/4]	139
6.7.2.3	BoundCube() [3/4]	139
6.7.2.4	BoundCube() [4/4]	140
6.7.3	Member Function Documentation	140
6.7.3.1	containedInSphere()	140
6.7.3.2	contains()	141
6.7.3.3	containsPt()	141
6.7.3.4	expand() [1/3]	141
6.7.3.5	expand() [2/3]	142
6.7.3.6	expand() [3/3]	142
6.7.3.7	getBound()	142
6.7.3.8	getBounds()	143
6.7.3.9	getCentroid()	143
6.7.3.10	getLargestDim()	144
6.7.3.11	getMaxDistanceToBox()	144
6.7.3.12	getSize()	145
6.7.3.13	getVolume()	145
6.7.3.14	intersects() [1/2]	146

6.7.3.15	<code>intersects()</code> [2/2]	146
6.7.3.16	<code>isFlat()</code>	147
6.7.3.17	<code>makeIntersection()</code>	147
6.7.3.18	<code>makeUnion()</code>	147
6.7.3.19	<code>max()</code>	148
6.7.3.20	<code>min()</code>	148
6.7.3.21	<code>operator=()</code>	149
6.7.3.22	<code>operator==(())</code>	149
6.7.3.23	<code>segmentTriple()</code>	149
6.7.3.24	<code>setBound()</code>	150
6.7.3.25	<code>setBounds()</code> [1/7]	150
6.7.3.26	<code>setBounds()</code> [2/7]	150
6.7.3.27	<code>setBounds()</code> [3/7]	151
6.7.3.28	<code>setBounds()</code> [4/7]	151
6.7.3.29	<code>setBounds()</code> [5/7]	151
6.7.3.30	<code>setBounds()</code> [6/7]	152
6.7.3.31	<code>setBounds()</code> [7/7]	152
6.7.3.32	<code>setInverseLimits()</code>	152
6.7.3.33	<code>setLimits()</code>	153
6.7.4	Friends And Related Function Documentation	153
6.7.4.1	<code>K3DTreeApprox</code>	153
6.7.4.2	<code>K3DTreeExact</code>	153
6.7.4.3	<code>operator<<</code>	153
6.8	<code>AtomProbe::ComparePairFirst</code> Class Reference	153
6.8.1	Detailed Description	154
6.8.2	Member Function Documentation	154
6.8.2.1	<code>operator()</code>	154
6.9	<code>AtomProbe::ComparePairFirstReverse</code> Class Reference	154
6.9.1	Detailed Description	154
6.9.2	Member Function Documentation	154

6.9.2.1	operator()	155
6.10	AtomProbe::ComparePairSecond Class Reference	155
6.10.1	Detailed Description	155
6.10.2	Member Function Documentation	156
6.10.2.1	operator()	156
6.11	AtomProbe::CrystalGen Class Reference	156
6.11.1	Detailed Description	157
6.11.2	Constructor & Destructor Documentation	157
6.11.2.1	CrystalGen()	157
6.11.2.2	~CrystalGen()	158
6.11.3	Member Function Documentation	158
6.11.3.1	extractPositions()	158
6.11.3.2	generateLattice()	158
6.11.3.3	mirrorOut()	159
6.11.3.4	swap()	159
6.11.4	Member Data Documentation	159
6.11.4.1	farPoint	159
6.11.4.2	localData	160
6.12	AtomProbe::EPOS_ENTRY Class Reference	160
6.12.1	Detailed Description	160
6.12.2	Member Function Documentation	160
6.12.2.1	getIonHit() [1/2]	161
6.12.2.2	getIonHit() [2/2]	161
6.12.2.3	operator==()	161
6.12.3	Member Data Documentation	161
6.12.3.1	deltaPulse	162
6.12.3.2	hitMultiplicity	162
6.12.3.3	massToCharge	162
6.12.3.4	timeOfFlight	162
6.12.3.5	voltDC	162

6.12.3.6	voltPulse	163
6.12.3.7	x	163
6.12.3.8	xDetector	163
6.12.3.9	y	163
6.12.3.10	yDetector	163
6.12.3.11	z	164
6.13	AtomProbe::FaceCentredCubicGen Class Reference	164
6.13.1	Detailed Description	165
6.13.2	Constructor & Destructor Documentation	165
6.13.2.1	FaceCentredCubicGen()	165
6.13.3	Member Function Documentation	166
6.13.3.1	generateLattice()	166
6.14	FixedStack< T > Class Template Reference	166
6.14.1	Detailed Description	167
6.14.2	Constructor & Destructor Documentation	167
6.14.2.1	FixedStack() [1/2]	167
6.14.2.2	FixedStack() [2/2]	167
6.14.2.3	~FixedStack()	167
6.14.3	Member Function Documentation	168
6.14.3.1	empty()	168
6.14.3.2	pop()	168
6.14.3.3	push()	168
6.14.3.4	top()	168
6.15	AtomProbe::FLATTENED_RANGE Struct Reference	169
6.15.1	Detailed Description	169
6.15.2	Member Data Documentation	169
6.15.2.1	containedIonIDs	169
6.15.2.2	containedRangeIDs	170
6.15.2.3	endMass	170
6.15.2.4	startMass	170

6.16	AtomProbe::GnomonicProjection Class Reference	170
6.16.1	Detailed Description	171
6.16.2	Member Function Documentation	171
6.16.2.1	scaleDown()	171
6.16.2.2	scaleUp()	172
6.16.2.3	toAzimuthal()	172
6.16.2.4	toPlanar()	172
6.17	AtomProbe::IonHit Class Reference	173
6.17.1	Detailed Description	174
6.17.2	Constructor & Destructor Documentation	174
6.17.2.1	IonHit() [1/5]	174
6.17.2.2	IonHit() [2/5]	174
6.17.2.3	IonHit() [3/5]	174
6.17.2.4	IonHit() [4/5]	174
6.17.2.5	IonHit() [5/5]	175
6.17.3	Member Function Documentation	175
6.17.3.1	getBoundCube()	175
6.17.3.2	getCentroid()	176
6.17.3.3	getIonDataLimits()	176
6.17.3.4	getMassToCharge()	177
6.17.3.5	getPoints()	177
6.17.3.6	getPos()	178
6.17.3.7	getPosRef()	178
6.17.3.8	hasNaN()	179
6.17.3.9	operator+()	179
6.17.3.10	operator=()	179
6.17.3.11	operator==(())	179
6.17.3.12	operator[]() [1/2]	179
6.17.3.13	operator[]() [2/2]	180
6.17.3.14	setHit()	180

6.17.3.15	setMassToCharge()	180
6.17.3.16	setPos() [1/3]	181
6.17.3.17	setPos() [2/3]	181
6.17.3.18	setPos() [3/3]	181
6.17.4	Friends And Related Function Documentation	182
6.17.4.1	operator<<	182
6.18	AtomProbe::!ONHIT Struct Reference	182
6.18.1	Detailed Description	182
6.18.2	Member Data Documentation	182
6.18.2.1	massToCharge	182
6.18.2.2	pos	183
6.19	AtomProbe::!ISOTOPE_ENTRY Struct Reference	183
6.19.1	Detailed Description	184
6.19.2	Member Data Documentation	184
6.19.2.1	abundance	184
6.19.2.2	abundanceError	184
6.19.2.3	atomicNumber	184
6.19.2.4	mass	184
6.19.2.5	massError	185
6.19.2.6	massNumber	185
6.19.2.7	symbol	185
6.20	AtomProbe::K3DNodeApprox Class Reference	185
6.20.1	Detailed Description	186
6.20.2	Member Function Documentation	186
6.20.2.1	deleteChildren()	186
6.20.2.2	dump()	187
6.20.2.3	getAxis()	187
6.20.2.4	getAxisVal()	187
6.20.2.5	getLoc()	188
6.20.2.6	getLocRef()	188

6.20.2.7	getLocVal()	188
6.20.2.8	left()	189
6.20.2.9	right()	189
6.20.2.10	setAxis()	189
6.20.2.11	setLeft()	189
6.20.2.12	setLoc()	190
6.20.2.13	setRight()	190
6.20.2.14	sqrDist()	190
6.21	AtomProbe::K3DNodeExact Class Reference	191
6.21.1	Detailed Description	191
6.21.2	Member Data Documentation	191
6.21.2.1	childLeft	191
6.21.2.2	childRight	191
6.21.2.3	tagged	191
6.22	AtomProbe::K3DTreeApprox Class Reference	192
6.22.1	Detailed Description	192
6.22.2	Constructor & Destructor Documentation	192
6.22.2.1	K3DTreeApprox()	192
6.22.2.2	~K3DTreeApprox()	193
6.22.3	Member Function Documentation	193
6.22.3.1	build()	193
6.22.3.2	buildByRef()	194
6.22.3.3	dump()	194
6.22.3.4	findKNearest()	195
6.22.3.5	findNearest()	196
6.22.3.6	kill()	196
6.22.3.7	nodeCount()	197
6.23	AtomProbe::K3DTreeExact Class Reference	197
6.23.1	Detailed Description	198
6.23.2	Constructor & Destructor Documentation	198

6.23.2.1	K3DTreeExact()	198
6.23.2.2	~K3DTreeExact()	198
6.23.3	Member Function Documentation	199
6.23.3.1	build()	199
6.23.3.2	clear()	199
6.23.3.3	clearAllTags()	200
6.23.3.4	clearTags()	200
6.23.3.5	dump()	200
6.23.3.6	findNearestUntagged()	201
6.23.3.7	findUntaggedInRadius()	201
6.23.3.8	getBoundCube()	202
6.23.3.9	getOrigIndex()	202
6.23.3.10	getPt()	203
6.23.3.11	getPtRef()	203
6.23.3.12	getTag()	203
6.23.3.13	ptsInSphere()	204
6.23.3.14	resetPts() [1/2]	204
6.23.3.15	resetPts() [2/2]	205
6.23.3.16	rootIdx()	205
6.23.3.17	setCallback()	205
6.23.3.18	setProgressPointer()	206
6.23.3.19	size()	206
6.23.3.20	tag() [1/2]	206
6.23.3.21	tag() [2/2]	206
6.23.3.22	tagCount()	207
6.24	AtomProbe::LibVersion Class Reference	207
6.24.1	Detailed Description	207
6.24.2	Constructor & Destructor Documentation	207
6.24.2.1	LibVersion()	208
6.24.3	Member Function Documentation	208

6.24.3.1	checkDebug()	208
6.24.3.2	getMajor()	208
6.24.3.3	getMinor()	209
6.24.3.4	getRevision()	209
6.24.3.5	getVersionStr()	209
6.25	AtomProbe::LINE Class Reference	210
6.25.1	Detailed Description	210
6.25.2	Member Data Documentation	210
6.25.2.1	p	210
6.25.2.2	physGroup	210
6.26	AtomProbe::LinearFeedbackShiftReg Class Reference	210
6.26.1	Detailed Description	211
6.26.2	Member Function Documentation	211
6.26.2.1	clock()	211
6.26.2.2	setMaskPeriod()	211
6.26.2.3	setState()	212
6.26.2.4	verifyTable()	212
6.27	AtomProbe::MassTool Class Reference	213
6.27.1	Detailed Description	213
6.27.2	Member Function Documentation	213
6.27.2.1	bruteKnapsack() [1/2]	213
6.27.2.2	bruteKnapsack() [2/2]	214
6.28	AtomProbe::Mesh Class Reference	215
6.28.1	Detailed Description	217
6.28.2	Member Function Documentation	217
6.28.2.1	clear()	217
6.28.2.2	countTriNodes()	217
6.28.2.3	divideMeshSurface()	218
6.28.2.4	elementCount()	218
6.28.2.5	erasePhysGroup()	218

6.28.2.6	getBounds()	219
6.28.2.7	getContainedNodes()	219
6.28.2.8	getCurPhysGroups()	220
6.28.2.9	getIntersectingPrimitives()	220
6.28.2.10	getNearestTri()	220
6.28.2.11	getTriNormal()	221
6.28.2.12	getVolume()	221
6.28.2.13	isOrientedCoherently()	222
6.28.2.14	isSane()	222
6.28.2.15	killOrphanNodes()	223
6.28.2.16	loadGmshMesh()	223
6.28.2.17	mergeDuplicateVertices()	224
6.28.2.18	numDupTris()	224
6.28.2.19	numDupVertices()	225
6.28.2.20	pointsInside()	225
6.28.2.21	print()	226
6.28.2.22	reassignGroups()	226
6.28.2.23	removeDuplicateTris()	226
6.28.2.24	removeStrayTris()	227
6.28.2.25	resurface()	227
6.28.2.26	rotate()	227
6.28.2.27	saveGmshMesh()	228
6.28.2.28	scale() [1/2]	228
6.28.2.29	scale() [2/2]	228
6.28.2.30	setTriangleMesh()	229
6.28.2.31	translate() [1/2]	229
6.28.2.32	translate() [2/2]	229
6.28.3	Member Data Documentation	230
6.28.3.1	lines	230
6.28.3.2	nodes	230

6.28.3.3	physGroupNames	230
6.28.3.4	points	230
6.28.3.5	tetrahedra	231
6.28.3.6	triangles	231
6.29	AtomProbe::MILLER_TRIPLET Class Reference	231
6.29.1	Detailed Description	232
6.29.2	Constructor & Destructor Documentation	232
6.29.2.1	MILLER_TRIPLET()	232
6.29.3	Member Function Documentation	232
6.29.3.1	h()	232
6.29.3.2	k()	232
6.29.3.3	l()	232
6.29.3.4	operator==()	233
6.29.3.5	simplify()	233
6.29.3.6	tripletToNormal() [1/2]	233
6.29.3.7	tripletToNormal() [2/2]	234
6.30	AtomProbe::ModifiedFocusSphericProjection Class Reference	234
6.30.1	Detailed Description	235
6.30.2	Constructor & Destructor Documentation	235
6.30.2.1	ModifiedFocusSphericProjection()	235
6.30.3	Member Function Documentation	235
6.30.3.1	etaToTheta()	236
6.30.3.2	getFOVRadius()	236
6.30.3.3	getMaxFOV()	236
6.30.3.4	scaleDown()	237
6.30.3.5	scaleUp()	237
6.30.3.6	setFocus()	238
6.30.3.7	thetaToEta()	238
6.30.3.8	toAzimuthal()	238
6.30.3.9	toPlanar()	239

6.31 AtomProbe::MultiRange Class Reference	239
6.31.1 Detailed Description	240
6.31.2 Constructor & Destructor Documentation	240
6.31.2.1 MultiRange() [1/2]	241
6.31.2.2 MultiRange() [2/2]	241
6.31.3 Member Function Documentation	241
6.31.3.1 addlon() [1/2]	242
6.31.3.2 addlon() [2/2]	242
6.31.3.3 addRange() [1/2]	243
6.31.3.4 addRange() [2/2]	243
6.31.3.5 clear()	243
6.31.3.6 copyDataFromRange()	244
6.31.3.7 flattenToMassAxis()	245
6.31.3.8 getColour()	245
6.31.3.9 getErrString()	245
6.31.3.10 getlonID()	245
6.31.3.11 getlonName()	246
6.31.3.12 getMolecule()	246
6.31.3.13 getNumlons()	246
6.31.3.14 getNumRanges() [1/2]	246
6.31.3.15 getNumRanges() [2/2]	247
6.31.3.16 getRange()	247
6.31.3.17 getRangeldsFromlon()	247
6.31.3.18 getRanges()	247
6.31.3.19 isRanged() [1/2]	248
6.31.3.20 isRanged() [2/2]	248
6.31.3.21 isSelfConsistent()	249
6.31.3.22 open()	249
6.31.3.23 range()	250
6.31.3.24 setColour()	250

6.31.3.25	setlonID()	250
6.31.3.26	setRangeGroups()	251
6.31.3.27	splitOverlapping()	251
6.31.3.28	write()	252
6.32	NodeWalk Class Reference	252
6.32.1	Detailed Description	253
6.32.2	Constructor & Destructor Documentation	253
6.32.2.1	NodeWalk()	253
6.32.3	Member Data Documentation	253
6.32.3.1	cube	253
6.32.3.2	depth	253
6.32.3.3	index	254
6.33	AtomProbe::OVERLAP_PROBLEM_SETTINGS Struct Reference	254
6.33.1	Detailed Description	254
6.33.2	Member Data Documentation	254
6.33.2.1	intensityTolerance	254
6.33.2.2	massTolerance	255
6.33.2.3	maxDefaultCharge	255
6.34	AtomProbe::Point3D Class Reference	255
6.34.1	Detailed Description	257
6.34.2	Constructor & Destructor Documentation	257
6.34.2.1	Point3D() [1/3]	258
6.34.2.2	Point3D() [2/3]	258
6.34.2.3	Point3D() [3/3]	258
6.34.3	Member Function Documentation	258
6.34.3.1	add()	259
6.34.3.2	angle()	259
6.34.3.3	copyValueArr()	260
6.34.3.4	crossProd()	260
6.34.3.5	dotProd()	260

6.34.3.6	<code>extend()</code>	261
6.34.3.7	<code>getCentroid()</code>	261
6.34.3.8	<code>getISOspherical()</code>	262
6.34.3.9	<code>getValue()</code>	263
6.34.3.10	<code>getValueArr()</code>	263
6.34.3.11	<code>hasNaN()</code>	264
6.34.3.12	<code>insideBox()</code> [1/2]	265
6.34.3.13	<code>insideBox()</code> [2/2]	265
6.34.3.14	<code>mag()</code>	265
6.34.3.15	<code>negate()</code>	266
6.34.3.16	<code>normal()</code>	266
6.34.3.17	<code>normalise()</code>	267
6.34.3.18	<code>operator*()</code> [1/2]	268
6.34.3.19	<code>operator*()</code> [2/2]	268
6.34.3.20	<code>operator*=(())</code>	269
6.34.3.21	<code>operator+()</code> [1/2]	269
6.34.3.22	<code>operator+()</code> [2/2]	269
6.34.3.23	<code>operator+=()</code>	269
6.34.3.24	<code>operator-()</code> [1/2]	270
6.34.3.25	<code>operator-()</code> [2/2]	270
6.34.3.26	<code>operator-=()</code>	270
6.34.3.27	<code>operator/()</code> [1/2]	270
6.34.3.28	<code>operator/()</code> [2/2]	271
6.34.3.29	<code>operator=()</code>	271
6.34.3.30	<code>operator==(())</code>	271
6.34.3.31	<code>operator[]()</code> [1/2]	271
6.34.3.32	<code>operator[]()</code> [2/2]	272
6.34.3.33	<code>orthogonalise()</code>	272
6.34.3.34	<code>parse()</code>	273
6.34.3.35	<code>reciprocal()</code>	274

6.34.3.36 setISOSpherical()	274
6.34.3.37 setValue() [1/2]	274
6.34.3.38 setValue() [2/2]	275
6.34.3.39 setValueArr()	275
6.34.3.40 sqrDist()	275
6.34.3.41 sqrMag()	276
6.34.3.42 transform3x3()	276
6.34.4 Friends And Related Function Documentation	276
6.34.4.1 operator<<	276
6.35 AtomProbe::Point3f Struct Reference	277
6.35.1 Detailed Description	277
6.35.2 Member Data Documentation	277
6.35.2.1 fx	277
6.35.2.2 fy	277
6.35.2.3 fz	278
6.36 AtomProbe::ProgressBar Class Reference	278
6.36.1 Detailed Description	278
6.36.2 Constructor & Destructor Documentation	278
6.36.2.1 ProgressBar()	279
6.36.2.2 ~ProgressBar()	279
6.36.3 Member Function Documentation	279
6.36.3.1 abort()	279
6.36.3.2 finish()	280
6.36.3.3 init()	280
6.36.3.4 reset()	280
6.36.3.5 setLength()	281
6.36.3.6 update()	281
6.37 AtomProbe::Quaternion Struct Reference	282
6.37.1 Detailed Description	282
6.37.2 Member Data Documentation	282

6.37.2.1	a	282
6.37.2.2	b	282
6.37.2.3	c	283
6.37.2.4	d	283
6.38	AtomProbe::RandNumGen Class Reference	283
6.38.1	Detailed Description	283
6.38.2	Constructor & Destructor Documentation	283
6.38.2.1	RandNumGen()	284
6.38.2.2	~RandNumGen()	284
6.38.3	Member Function Documentation	284
6.38.3.1	getRng()	284
6.39	AtomProbe::RANGE_MOLECULE Struct Reference	285
6.39.1	Detailed Description	285
6.39.2	Member Data Documentation	285
6.39.2.1	components	285
6.39.2.2	isOK	286
6.39.2.3	name	286
6.40	AtomProbe::RangeFile Class Reference	286
6.40.1	Detailed Description	289
6.40.2	Constructor & Destructor Documentation	289
6.40.2.1	RangeFile()	289
6.40.3	Member Function Documentation	289
6.40.3.1	addlon()	289
6.40.3.2	addRange()	290
6.40.3.3	decomposelonByld()	290
6.40.3.4	decomposelonNames()	291
6.40.3.5	detectFileType()	291
6.40.3.6	eraselon()	292
6.40.3.7	eraseRange()	293
6.40.3.8	extensionIsRange()	293

6.40.3.9	extractIons()	293
6.40.3.10	getAllExts()	294
6.40.3.11	getColour()	294
6.40.3.12	getErrState()	294
6.40.3.13	getErrString()	295
6.40.3.14	getIonID() [1/5]	296
6.40.3.15	getIonID() [2/5]	296
6.40.3.16	getIonID() [3/5]	297
6.40.3.17	getIonID() [4/5]	297
6.40.3.18	getIonID() [5/5]	297
6.40.3.19	getName() [1/2]	298
6.40.3.20	getName() [2/2]	299
6.40.3.21	getNumIons()	299
6.40.3.22	getNumRanges() [1/2]	300
6.40.3.23	getNumRanges() [2/2]	300
6.40.3.24	getRange()	300
6.40.3.25	getRangeByRef()	301
6.40.3.26	getRangeID()	301
6.40.3.27	getRangeVolume()	301
6.40.3.28	guessChargeState()	302
6.40.3.29	haveRangeVolumes()	303
6.40.3.30	isRanged() [1/3]	303
6.40.3.31	isRanged() [2/3]	303
6.40.3.32	isRanged() [3/3]	304
6.40.3.33	isSelfConsistent() [1/2]	304
6.40.3.34	isSelfConsistent() [2/2]	304
6.40.3.35	makeSelfConsistent()	305
6.40.3.36	moveBothRanges()	306
6.40.3.37	moveRange()	306
6.40.3.38	open()	306

6.40.3.39	<code>openFormat()</code>	307
6.40.3.40	<code>operator=()</code>	308
6.40.3.41	<code>range()</code> [1/2]	308
6.40.3.42	<code>range()</code> [2/2]	309
6.40.3.43	<code>rangeByID()</code>	309
6.40.3.44	<code>rangeByRangeID()</code>	309
6.40.3.45	<code>rangeInvertable()</code>	310
6.40.3.46	<code>rangeTypeString()</code>	310
6.40.3.47	<code>setColour()</code>	311
6.40.3.48	<code>setEnforceConsistent()</code>	311
6.40.3.49	<code>setIonID()</code>	312
6.40.3.50	<code>setIonLongName()</code>	312
6.40.3.51	<code>setIonShortName()</code>	313
6.40.3.52	<code>setRangeEnd()</code>	313
6.40.3.53	<code>setRangeStart()</code>	314
6.40.3.54	<code>setRangeVolume()</code>	314
6.40.3.55	<code>swap()</code>	314
6.40.3.56	<code>write()</code> [1/2]	315
6.40.3.57	<code>write()</code> [2/2]	315
6.41	<code>AtomProbe::ReconstructionSphereOnCone</code> Class Reference	316
6.41.1	Detailed Description	316
6.41.2	Constructor & Destructor Documentation	316
6.41.2.1	<code>ReconstructionSphereOnCone()</code>	316
6.41.3	Member Function Documentation	317
6.41.3.1	<code>reconstruct()</code>	317
6.41.3.2	<code>setDetectorEfficiency()</code>	318
6.41.3.3	<code>setFlightPath()</code>	319
6.41.3.4	<code>setInitialRadius()</code>	319
6.41.3.5	<code>setProjModel()</code>	319
6.41.3.6	<code>setRadiusEvolutionMode()</code>	319

6.41.3.7	setReconFOV()	320
6.41.3.8	setShankAngle()	320
6.42	AtomProbe::RGBf Class Reference	320
6.42.1	Detailed Description	321
6.42.2	Member Function Documentation	321
6.42.2.1	fromHex()	321
6.42.2.2	toHex()	321
6.42.3	Member Data Documentation	322
6.42.3.1	blue	322
6.42.3.2	green	322
6.42.3.3	red	322
6.43	AtomProbe::SIMPLE_SPECIES Struct Reference	322
6.43.1	Detailed Description	323
6.43.2	Member Function Documentation	323
6.43.2.1	operator==()	323
6.43.3	Member Data Documentation	323
6.43.3.1	atomicNumber	323
6.43.3.2	count	324
6.44	AtomProbe::SimpleCubicGen Class Reference	324
6.44.1	Detailed Description	325
6.44.2	Constructor & Destructor Documentation	325
6.44.2.1	SimpleCubicGen()	325
6.44.3	Member Function Documentation	326
6.44.3.1	generateLattice()	326
6.45	AtomProbe::SINGLE_HIT Struct Reference	326
6.45.1	Detailed Description	327
6.45.2	Member Data Documentation	327
6.45.2.1	tof	327
6.45.2.2	x	327
6.45.2.3	y	327

6.46	AtomProbe::SphericPlaneProjection Class Reference	328
6.46.1	Detailed Description	328
6.46.2	Member Function Documentation	328
6.46.2.1	scaleDown()	328
6.46.2.2	scaleUp()	329
6.46.2.3	toAzimuthal()	329
6.46.2.4	toPlanar()	329
6.47	AtomProbe::SphericProjectionParams Struct Reference	330
6.47.1	Detailed Description	330
6.47.2	Member Data Documentation	330
6.47.2.1	focusDist	330
6.47.2.2	theta	330
6.48	AtomProbe::StereographicProjection Class Reference	331
6.48.1	Detailed Description	332
6.48.2	Member Function Documentation	332
6.48.2.1	scaleDown()	332
6.48.2.2	scaleUp()	332
6.48.2.3	thetaToEta()	333
6.48.2.4	toAzimuthal()	333
6.48.2.5	toPlanar()	333
6.49	AtomProbe::TETRAHEDRON Class Reference	334
6.49.1	Detailed Description	334
6.49.2	Member Data Documentation	334
6.49.2.1	p	334
6.49.2.2	physGroup	334
6.50	AtomProbe::THREEDAP_DATA Struct Reference	334
6.50.1	Detailed Description	335
6.50.2	Member Data Documentation	335
6.50.2.1	alpha	335
6.50.2.2	beta	335

6.50.2.3	detectorChannels	336
6.50.2.4	detectorRadius	336
6.50.2.5	flightPath	336
6.50.2.6	tZero	336
6.51	AtomProbe::THREEDAP_EXPERIMENT Struct Reference	337
6.51.1	Detailed Description	337
6.51.2	Member Data Documentation	337
6.51.2.1	eventData	337
6.51.2.2	eventPulseNumber	338
6.51.2.3	params	338
6.51.2.4	voltageData	338
6.52	AtomProbe::TRIANGLE Class Reference	338
6.52.1	Detailed Description	339
6.52.2	Member Function Documentation	339
6.52.2.1	edgesMismatch()	339
6.52.2.2	isSane()	339
6.52.3	Member Data Documentation	340
6.52.3.1	p	340
6.52.3.2	physGroup	340
6.53	AtomProbe::TriangleWithIndexedVertices Struct Reference	340
6.53.1	Detailed Description	340
6.53.2	Member Data Documentation	340
6.53.2.1	p	341
6.54	AtomProbe::TriangleWithVertexNorm Class Reference	341
6.54.1	Detailed Description	341
6.54.2	Member Function Documentation	342
6.54.2.1	computeACWNormal()	342
6.54.2.2	computeArea()	342
6.54.2.3	getCentroid()	343
6.54.2.4	isDegenerate()	343

6.54.2.5	safeComputeACWNormal()	344
6.54.3	Member Data Documentation	344
6.54.3.1	normal	344
6.54.3.2	p	345
6.55	AtomProbe::VOLTAGE_DATA Struct Reference	345
6.55.1	Detailed Description	345
6.55.2	Member Data Documentation	345
6.55.2.1	beta	345
6.55.2.2	nextHitGroupOffset	346
6.55.2.3	pulseVolt	346
6.55.2.4	voltage	346
6.56	AtomProbe::Voxels< T > Class Template Reference	346
6.56.1	Detailed Description	349
6.56.2	Constructor & Destructor Documentation	349
6.56.2.1	Voxels()	349
6.56.2.2	~Voxels()	350
6.56.3	Member Function Documentation	350
6.56.3.1	applyMask()	350
6.56.3.2	binarise()	351
6.56.3.3	calculateDensity()	351
6.56.3.4	clear()	352
6.56.3.5	convolve()	353
6.56.3.6	copy()	353
6.56.3.7	count()	353
6.56.3.8	countPoints()	354
6.56.3.9	deprecatedGetEdgeUniqueIndex()	354
6.56.3.10	fill()	355
6.56.3.11	getAxisBounds()	355
6.56.3.12	getBinVolume()	355
6.56.3.13	getBounds() [1/2]	356

6.56.3.14	<code>getBounds()</code> [2/2]	356
6.56.3.15	<code>getCellUniqueEdgeIndex()</code>	357
6.56.3.16	<code>getData()</code> [1/2]	357
6.56.3.17	<code>getData()</code> [2/2]	357
6.56.3.18	<code>getEdgeCell()</code>	358
6.56.3.19	<code>getEdgeEndApproxVals()</code>	358
6.56.3.20	<code>getEdgeEnds()</code>	359
6.56.3.21	<code>getIndex()</code>	360
6.56.3.22	<code>getIndexWithUpper()</code>	361
6.56.3.23	<code>getInterpolatedData()</code>	361
6.56.3.24	<code>getInterpSlice()</code>	362
6.56.3.25	<code>getMaxBounds()</code>	363
6.56.3.26	<code>getMinBounds()</code>	363
6.56.3.27	<code>getOffset()</code>	363
6.56.3.28	<code>getPitch()</code>	363
6.56.3.29	<code>getPoint()</code>	364
6.56.3.30	<code>getPointData()</code>	364
6.56.3.31	<code>getSize()</code>	364
6.56.3.32	<code>getSlice()</code>	365
6.56.3.33	<code>getSum()</code>	365
6.56.3.34	<code>init()</code> [1/2]	366
6.56.3.35	<code>init()</code> [2/2]	366
6.56.3.36	<code>loadFile()</code>	367
6.56.3.37	<code>max()</code>	367
6.56.3.38	<code>min()</code>	368
6.56.3.39	<code>minMax()</code>	368
6.56.3.40	<code>operator/=()</code> [1/2]	368
6.56.3.41	<code>operator/=()</code> [2/2]	369
6.56.3.42	<code>operator==()</code>	369
6.56.3.43	<code>rebin()</code>	369

6.56.3.44	resize() [1/2]	369
6.56.3.45	resize() [2/2]	370
6.56.3.46	resizeKeepData()	370
6.56.3.47	separableConvolve()	371
6.56.3.48	setBounds() [1/2]	371
6.56.3.49	setBounds() [2/2]	372
6.56.3.50	setData() [1/2]	372
6.56.3.51	setData() [2/2]	373
6.56.3.52	setEntry()	373
6.56.3.53	setPoint()	374
6.56.3.54	size()	375
6.56.3.55	sizeofType()	375
6.56.3.56	swap()	376
6.56.3.57	threshold()	376
6.56.3.58	thresholdForPosition()	377
6.56.3.59	thresholdToBoolMask()	377
6.56.3.60	trapezIntegral()	378
6.56.3.61	writeFile()	378
6.57	AtomProbe::Weight Class Reference	379
6.57.1	Detailed Description	379
6.57.2	Constructor & Destructor Documentation	379
6.57.2.1	Weight() [1/2]	379
6.57.2.2	Weight() [2/2]	379
6.57.3	Member Function Documentation	380
6.57.3.1	operator<()	380
6.57.3.2	operator==(())	380
6.57.4	Member Data Documentation	380
6.57.4.1	mass	380
6.57.4.2	uniqueId	380
6.58	WEIGHT_SEARCH_ENTRY Class Reference	381
6.58.1	Detailed Description	381
6.58.2	Member Data Documentation	381
6.58.2.1	allowableWeights	381
6.58.2.2	cumulativeWeight	382
6.58.2.3	curWeights	382
6.58.2.4	offset	382
6.59	AtomProbe::ZECH_ROOT Struct Reference	382
6.59.1	Detailed Description	382
6.59.2	Member Data Documentation	382
6.59.2.1	alpha	383
6.59.2.2	lambdaBack	383
6.59.2.3	observation	383

7 File Documentation	385
7.1 CMakeFiles/3.10.2/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference	385
7.1.1 Macro Definition Documentation	385
7.1.1.1 ARCHITECTURE_ID	385
7.1.1.2 COMPILER_ID	386
7.1.1.3 CXX_STD	386
7.1.1.4 DEC	386
7.1.1.5 HEX	386
7.1.1.6 PLATFORM_ID	387
7.1.1.7 STRINGIFY	387
7.1.1.8 STRINGIFY_HELPER	387
7.1.2 Function Documentation	387
7.1.2.1 main()	387
7.1.3 Variable Documentation	387
7.1.3.1 info_arch	387
7.1.3.2 info_compiler	388
7.1.3.3 info_language_dialect_default	388
7.1.3.4 info_platform	388
7.2 CMakeFiles/CheckTypeSize/SIZEOF_SIZE_T.cpp File Reference	388
7.2.1 Macro Definition Documentation	389
7.2.1.1 SIZE	389
7.2.2 Function Documentation	389
7.2.2.1 main()	389
7.2.3 Variable Documentation	389
7.2.3.1 info_size	390
7.3 example/correlationhist.cpp File Reference	390
7.3.1 Function Documentation	390
7.3.1.1 dumpHistogram()	390
7.3.1.2 main()	391
7.4 example/countepos.cpp File Reference	391

7.4.1	Function Documentation	391
7.4.1.1	main()	392
7.5	example/kd-example.cpp File Reference	392
7.5.1	Macro Definition Documentation	393
7.5.1.1	M_PI	393
7.5.1.2	TIME_END	393
7.5.1.3	TIME_START	393
7.5.2	Function Documentation	393
7.5.2.1	callback()	394
7.5.2.2	main()	394
7.5.3	Variable Documentation	394
7.5.3.1	progress	394
7.6	example/massFit.cpp File Reference	395
7.6.1	Function Documentation	395
7.6.1.1	main()	395
7.6.1.2	printSolution()	396
7.7	example/massHist.cpp File Reference	396
7.7.1	Function Documentation	396
7.7.1.1	main()	396
7.8	example/multiplesplit.cpp File Reference	397
7.8.1	Function Documentation	397
7.8.1.1	main()	397
7.9	example/plot-overlaps.cpp File Reference	398
7.9.1	Function Documentation	398
7.9.1.1	main()	398
7.10	example/polefigure.cpp File Reference	399
7.10.1	Function Documentation	399
7.10.1.1	gsl_print_matrix()	400
7.10.1.2	main()	400
7.11	example/proxigram.cpp File Reference	401

7.11.1	Function Documentation	402
7.11.1.1	main()	402
7.12	example/pulse-filter.cpp File Reference	403
7.12.1	Enumeration Type Documentation	404
7.12.1.1	anonymous enum	404
7.12.2	Function Documentation	404
7.12.2.1	convertToPos()	404
7.12.2.2	dumpHistogram()	405
7.12.2.3	filterEposByPulse()	405
7.12.2.4	main()	405
7.13	example/signedDistance.cpp File Reference	406
7.13.1	Function Documentation	406
7.13.1.1	main()	406
7.14	example/specsim.cpp File Reference	406
7.14.1	Typedef Documentation	407
7.14.1.1	FRAGMENT	407
7.14.2	Function Documentation	407
7.14.2.1	getMassDistributions()	407
7.14.2.2	main()	408
7.14.2.3	normaliseVec()	408
7.14.3	Variable Documentation	408
7.14.3.1	MASS_TOL	408
7.15	example/zechBackground.cpp File Reference	409
7.15.1	Macro Definition Documentation	409
7.15.1.1	ASSERT	409
7.15.2	Function Documentation	411
7.15.2.1	dumpHistogramToFile()	411
7.15.2.2	main()	411
7.15.2.3	zechCorrect()	412
7.16	extras/autocorrelate/autocorrelate.cpp File Reference	412

7.16.1	Macro Definition Documentation	413
7.16.1.1	ASSERT	413
7.16.2	Function Documentation	413
7.16.2.1	callback()	413
7.16.2.2	dumpAutocorrelation()	414
7.16.2.3	main()	414
7.16.3	Variable Documentation	415
7.16.3.1	PROGRESS_BAR_SIZE	415
7.16.3.2	treeProgressBar	416
7.16.3.3	treeProgressValue	416
7.17	extras/readpos_c/readpos.h File Reference	416
7.17.1	Function Documentation	416
7.17.1.1	readPos()	416
7.18	include/atomprobe/algorithm/axialdf.h File Reference	417
7.19	include/atomprobe/algorithm/componentAnalysis.h File Reference	418
7.20	include/atomprobe/algorithm/convexHull.h File Reference	419
7.21	include/atomprobe/algorithm/filter.h File Reference	420
7.22	include/atomprobe/algorithm/histogram.h File Reference	421
7.23	include/atomprobe/algorithm/isoSurface.h File Reference	422
7.24	include/atomprobe/algorithm/K3DTree-approx.h File Reference	422
7.25	include/atomprobe/algorithm/K3DTree-exact.h File Reference	423
7.26	include/atomprobe/algorithm/massTool.h File Reference	424
7.27	include/atomprobe/algorithm/rangeCheck.h File Reference	425
7.28	include/atomprobe/algorithm/rotations.h File Reference	426
7.29	include/atomprobe/atomprobe.h File Reference	426
7.30	include/atomprobe/deconvolution/deconvolution.h File Reference	428
7.31	include/atomprobe/helper/aptAssert.h File Reference	428
7.31.1	Macro Definition Documentation	429
7.31.1.1	ASSERT	429
7.31.1.2	TEST	429

7.31.1.3	WARN	430
7.32	include/atomprobe/helper/composition.h File Reference	430
7.33	include/atomprobe/helper/constants.h File Reference	431
7.34	include/atomprobe/helper/convert.h File Reference	431
7.35	include/atomprobe/helper/endianTest.h File Reference	432
7.35.1	Function Documentation	432
7.35.1.1	floatSwapBytes()	433
7.35.1.2	int4SwapBytes()	433
7.35.1.3	is_bigendian()	433
7.35.1.4	is_littleendian()	433
7.35.2	Variable Documentation	433
7.35.2.1	ENDIAN_TEST	433
7.36	include/atomprobe/helper/maths/ffsr.h File Reference	434
7.37	include/atomprobe/helper/maths/misc.h File Reference	434
7.38	include/atomprobe/helper/misc.h File Reference	436
7.39	include/atomprobe/helper/maths/quat.h File Reference	437
7.40	include/atomprobe/helper/progress.h File Reference	438
7.41	include/atomprobe/helper/sampling.h File Reference	438
7.42	include/atomprobe/helper/stringFuncs.h File Reference	439
7.43	include/atomprobe/helper/version.h File Reference	441
7.43.1	Macro Definition Documentation	442
7.43.1.1	LIBATOMPROBE_MAJOR	442
7.43.1.2	LIBATOMPROBE_MINOR	442
7.43.1.3	LIBATOMPROBE_REVISION	442
7.44	include/atomprobe/helper/voxels.h File Reference	442
7.44.1	Macro Definition Documentation	444
7.44.1.1	XOR	444
7.45	include/atomprobe/helper/XMLHelper.h File Reference	444
7.46	include/atomprobe/io/dataFiles.h File Reference	445
7.47	include/atomprobe/io/multiRange.h File Reference	448

7.48	include/atomprobe/io/ranges.h File Reference	449
7.49	include/atomprobe/isotopes/abundance.h File Reference	451
7.50	include/atomprobe/isotopes/massconvert.h File Reference	451
7.51	include/atomprobe/isotopes/overlaps.h File Reference	453
7.52	include/atomprobe/latticeplanes/generate.h File Reference	454
7.53	include/atomprobe/latticeplanes/millerIndex.h File Reference	455
7.54	include/atomprobe/primitives/boundcube.h File Reference	455
7.55	include/atomprobe/primitives/ionHit.h File Reference	456
7.56	include/atomprobe/primitives/mesh.h File Reference	457
7.57	include/atomprobe/primitives/point3D.h File Reference	458
7.58	include/atomprobe/projection/projection.h File Reference	459
7.59	include/atomprobe/reconstruction/reconstruction-simple.h File Reference	459
7.60	include/atomprobe/spectrum/processing.h File Reference	460
7.61	include/atomprobe/statistics/confidence.h File Reference	461
7.62	math.cpp File Reference	462
7.62.1	Macro Definition Documentation	463
7.62.1.1	ASSERT	463
7.62.2	Function Documentation	463
7.62.2.1	gsl_matrix_mult()	463
7.62.2.2	gsl_pseudo_inverse()	463
7.62.2.3	psuedoInverseFromSVD()	464
7.63	math.h File Reference	464
7.63.1	Function Documentation	465
7.63.1.1	computeMeanAndVariance()	465
7.63.1.2	gsl_matrix_mult()	465
7.63.1.3	gsl_pseudo_inverse()	466
7.64	src/algorithm/axialdf.cpp File Reference	466
7.65	src/algorithm/componentAnalysis.cpp File Reference	467
7.66	src/algorithm/convexHull.cpp File Reference	467
7.67	src/algorithm/filter.cpp File Reference	468

7.68	src/algorithm/histogram.cpp File Reference	469
7.69	src/algorithm/isoSurface.cpp File Reference	470
7.70	src/algorithm/K3DTree-approx.cpp File Reference	471
7.71	src/algorithm/K3DTree-exact.cpp File Reference	472
7.71.1	Variable Documentation	473
7.71.1.1	PROGRESS_REDUCE	473
7.72	src/algorithm/massTool.cpp File Reference	473
7.73	src/algorithm/rangeCheck.cpp File Reference	474
7.74	src/algorithm/rotations.cpp File Reference	474
7.75	src/atomprobe.cpp File Reference	475
7.76	src/deconvolution/deconvolution.cpp File Reference	476
7.77	src/helper/aptAssert.cpp File Reference	476
7.78	src/helper/composition.cpp File Reference	477
7.79	src/helper/convert.cpp File Reference	478
7.79.1	Function Documentation	478
7.79.1.1	convertEPOStoPos()	478
7.80	src/helper/helpFuncs.cpp File Reference	479
7.81	src/helper/helpFuncs.h File Reference	479
7.81.1	Macro Definition Documentation	481
7.81.1.1	ARRAYSIZE	481
7.81.1.2	XOR	481
7.82	src/helper/math/lfsr.cpp File Reference	481
7.83	src/helper/math/math.cpp File Reference	482
7.83.1	Macro Definition Documentation	482
7.83.1.1	ASSERT	483
7.83.2	Function Documentation	483
7.83.2.1	gsI_matrix_mult()	483
7.83.2.2	gsI_pseudo_inverse()	483
7.83.2.3	psuedoInverseFromSVD()	484
7.84	src/helper/math/misc.cpp File Reference	484

7.85	src/helper/maths/quat.cpp File Reference	485
7.86	src/helper/progress.cpp File Reference	486
7.87	src/helper/sampling.cpp File Reference	487
7.88	src/helper/stringFuncs.cpp File Reference	487
7.89	src/helper/voxels.cpp File Reference	488
7.90	src/helper/XMLHelper.cpp File Reference	489
7.91	src/io/dataFiles.cpp File Reference	490
7.92	src/io/multiRange.cpp File Reference	491
7.93	src/io/ranges.cpp File Reference	493
7.94	src/isotopes/abundance.cpp File Reference	494
7.95	src/isotopes/massconvert.cpp File Reference	494
7.96	src/isotopes/overlaps.cpp File Reference	495
7.96.1	Macro Definition Documentation	496
7.96.1.1	EQ_TOLV	496
7.97	src/lattice/generate.cpp File Reference	496
7.98	src/lattice/millerIndex.cpp File Reference	497
7.99	src/primitives/boundcube.cpp File Reference	497
7.100	src/primitives/ionhit.cpp File Reference	498
7.101	src/primitives/mesh.cpp File Reference	498
7.102	src/primitives/point3D.cpp File Reference	500
7.103	src/projection/projection.cpp File Reference	500
7.104	src/reconstruction/reconstruction-simple.cpp File Reference	501
7.105	src/spectrum/processing.cpp File Reference	502
7.105.1	Macro Definition Documentation	503
7.105.1.1	USE_CENTRAL	503
7.106	src/statistics/confidence.cpp File Reference	503
7.107	test/KDTest.cpp File Reference	504
7.107.1	Function Documentation	505
7.107.1.1	callback()	505
7.107.1.2	kdExactFuzz()	505

7.107.1.3 main()	506
7.107.1.4 testInexactKDTree()	506
7.107.2 Variable Documentation	507
7.107.2.1 NUM_IONS	507
7.107.2.2 progress	507
7.107.2.3 SCALE	507
7.107.2.4 SEARCH_RAD	507
7.108test/rangetest.cpp File Reference	508
7.108.1 Function Documentation	508
7.108.1.1 main()	508
7.109test/test.h File Reference	509
7.109.1 Macro Definition Documentation	510
7.109.1.1 TEST	510
7.110test/unittests.cpp File Reference	510
7.110.1 Macro Definition Documentation	510
7.110.1.1 ASSERT	510
7.110.2 Function Documentation	511
7.110.2.1 main()	511
7.110.2.2 rangePaths()	511
7.110.2.3 runTests()	512
Index	513

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

[AtomProbe](#) 11

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

AtomProbe::AbundanceData	117
AtomProbe::ATO_ENTRY	128
AtomProbe::AxisCompare	130
AxisCompareExact	131
AtomProbe::BACKGROUND_PARAMS	132
AtomProbe::BoundCube	136
AtomProbe::ComparePairFirst	153
AtomProbe::ComparePairFirstReverse	154
AtomProbe::ComparePairSecond	155
AtomProbe::CrystalGen	156
AtomProbe::BodyCentredCubicGen	134
AtomProbe::FaceCentredCubicGen	164
AtomProbe::SimpleCubicGen	324
AtomProbe::EPOS_ENTRY	160
FixedStack< T >	166
AtomProbe::FLATTENED_RANGE	169
AtomProbe::IonHit	173
AtomProbe::IONHIT	182
AtomProbe::ISOTOPE_ENTRY	183
AtomProbe::K3DNodeApprox	185
AtomProbe::K3DNodeExact	191
AtomProbe::K3DTreeApprox	192
AtomProbe::K3DTreeExact	197
AtomProbe::LibVersion	207
AtomProbe::LINE	210
AtomProbe::LinearFeedbackShiftReg	210
AtomProbe::MassTool	213
AtomProbe::Mesh	215
AtomProbe::MILLER_TRIPLET	231
AtomProbe::MultiRange	239
NodeWalk	252
AtomProbe::OVERLAP_PROBLEM_SETTINGS	254
AtomProbe::Point3D	255
AtomProbe::Point3f	277
AtomProbe::ProgressBar	278

AtomProbe::Quaternion	282
AtomProbe::RandNumGen	283
AtomProbe::RANGE_MOLECULE	285
AtomProbe::RangeFile	286
AtomProbe::ReconstructionSphereOnCone	316
AtomProbe::RGBf	320
AtomProbe::SIMPLE_SPECIES	322
AtomProbe::SINGLE_HIT	326
AtomProbe::SphericPlaneProjection	328
AtomProbe::GnomonicProjection	170
AtomProbe::ModifiedFocusSphericProjection	234
AtomProbe::StereographicProjection	331
AtomProbe::SphericProjectionParams	330
AtomProbe::TETRAHEDRON	334
AtomProbe::THREEDAP_DATA	334
AtomProbe::THREEDAP_EXPERIMENT	337
AtomProbe::TRIANGLE	338
AtomProbe::TriangleWithIndexedVertices	340
AtomProbe::TriangleWithVertexNorm	341
AtomProbe::VOLTAGE_DATA	345
AtomProbe::Voxels< T >	346
AtomProbe::Weight	379
WEIGHT_SEARCH_ENTRY	381
AtomProbe::ZECH_ROOT	382

Chapter 3

Class Index

3.1 Class List

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

AtomProbe::AbundanceData	Class to load abundance information for natural isotopes	117
AtomProbe::ATO_ENTRY	128
AtomProbe::AxisCompare	Functor allowing for sorting of points in 3D	130
AxisCompareExact	Functor allowing for sorting of points in 3D	131
AtomProbe::BACKGROUND_PARAMS	132
AtomProbe::BodyCentredCubicGen	134
AtomProbe::BoundCube	Helper class to define a bounding cube	136
AtomProbe::ComparePairFirst	153
AtomProbe::ComparePairFirstReverse	154
AtomProbe::ComparePairSecond	155
AtomProbe::CrystalGen	156
AtomProbe::EPOS_ENTRY	160
AtomProbe::FaceCentredCubicGen	164
FixedStack< T >	166
AtomProbe::FLATTENED_RANGE	Structure that allows for the multirange data to be mapped into	169
AtomProbe::GnomonicProjection	170
AtomProbe::IonHit	This is a data holding class for POS file ions, from	173
AtomProbe::IONHIT	Record as stored in a .POS file	182
AtomProbe::ISOTOPE_ENTRY	183
AtomProbe::K3DNodeApprox	Node Class for storing point	185
AtomProbe::K3DNodeExact	Node Class for storing point	191
AtomProbe::K3DTreeApprox	3D specific KD tree	192
AtomProbe::K3DTreeExact	3D specific KD tree	197
AtomProbe::LibVersion	Class to hold the library version data	207

AtomProbe::LINE	210
AtomProbe::LinearFeedbackShiftReg	
This class implements a Linear Feedback Shift Register (in software)	210
AtomProbe::MassTool	
Class that brute-force solves the knapsack problem	213
AtomProbe::Mesh	
Simple mesh class for storing and manipulating meshes consisting of 2 and 3D simplexes (triangles or tetrahedra)	215
AtomProbe::MILLER_TRIPLET	231
AtomProbe::ModifiedFocusSphericProjection	234
AtomProbe::MultiRange	
Data storage and retrieval class for "ranging" a spectra, where overlapping ranges are permitted	239
NodeWalk	252
AtomProbe::OVERLAP_PROBLEM_SETTINGS	254
AtomProbe::Point3D	
A 3D point data class storage	255
AtomProbe::Point3f	
Data storage structure for points	277
AtomProbe::ProgressBar	278
AtomProbe::Quaternion	
Data storage structure for quaternions	282
AtomProbe::RandNumGen	283
AtomProbe::RANGE_MOLECULE	285
AtomProbe::RangeFile	
Data storage and retrieval class for various range files	286
AtomProbe::ReconstructionSphereOnCone	316
AtomProbe::RGBf	
Data holder for colour as float	320
AtomProbe::SIMPLE_SPECIES	322
AtomProbe::SimpleCubicGen	324
AtomProbe::SINGLE_HIT	
Single 3DAP hit event	326
AtomProbe::SphericPlaneProjection	328
AtomProbe::SphericProjectionParams	330
AtomProbe::StereographicProjection	331
AtomProbe::TETRAHEDRON	334
AtomProbe::THREEDAP_DATA	
Experimental setup data	334
AtomProbe::THREEDAP_EXPERIMENT	
Data structure that contains the experiment information present in a 3Dap file	337
AtomProbe::TRIANGLE	338
AtomProbe::TriangleWithIndexedVertices	340
AtomProbe::TriangleWithVertexNorm	341
AtomProbe::VOLTAGE_DATA	
Voltage data structure – these updates occur periodically during the experiment	345
AtomProbe::Voxels< T >	
Template class that stores 3D voxel data	346
AtomProbe::Weight	
Placeholder class for containing input weights for MassTool	379
WEIGHT_SEARCH_ENTRY	381
AtomProbe::ZECH_ROOT	382

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

math.cpp	462
math.h	464
CMakeFiles/3.10.2/CompilerIdCXX/CompilerIdCXX.cpp	385
CMakeFiles/CheckTypeSize/SIZEOF_SIZE_T.cpp	388
example/correlationhist.cpp	390
example/countepos.cpp	391
example/kd-example.cpp	392
example/massFit.cpp	395
example/massHist.cpp	396
example/multiplesplit.cpp	397
example/plot-overlaps.cpp	398
example/polefigure.cpp	399
example/proxigram.cpp	401
example/pulse-filter.cpp	403
example/signedDistance.cpp	406
example/specsim.cpp	406
example/zechBackground.cpp	409
extras/autocorrelate/autocorrelate.cpp	412
extras/readpos_c/readpos.h	416
include/atomprobe/atomprobe.h	426
include/atomprobe/algorithm/axialdf.h	417
include/atomprobe/algorithm/componentAnalysis.h	418
include/atomprobe/algorithm/convexHull.h	419
include/atomprobe/algorithm/filter.h	420
include/atomprobe/algorithm/histogram.h	421
include/atomprobe/algorithm/isoSurface.h	422
include/atomprobe/algorithm/K3DTree-approx.h	422
include/atomprobe/algorithm/K3DTree-exact.h	423
include/atomprobe/algorithm/massTool.h	424
include/atomprobe/algorithm/rangeCheck.h	425
include/atomprobe/algorithm/rotations.h	426
include/atomprobe/deconvolution/deconvolution.h	428
include/atomprobe/helper/aptAssert.h	428
include/atomprobe/helper/composition.h	430
include/atomprobe/helper/constants.h	431

include/atomprobe/helper/convert.h	431
include/atomprobe/helper/EndianTest.h	432
include/atomprobe/helper/misc.h	436
include/atomprobe/helper/progress.h	438
include/atomprobe/helper/sampling.h	438
include/atomprobe/helper/stringFuncs.h	439
include/atomprobe/helper/version.h	441
include/atomprobe/helper/voxels.h	442
include/atomprobe/helper/XMLHelper.h	444
include/atomprobe/helper/mathsf/fsr.h	434
include/atomprobe/helper/mathsf/misc.h	434
include/atomprobe/helper/mathsf/quat.h	437
include/atomprobe/io/dataFiles.h	445
include/atomprobe/io/multiRange.h	448
include/atomprobe/io/ranges.h	449
include/atomprobe/isotopes/abundance.h	451
include/atomprobe/isotopes/massconvert.h	451
include/atomprobe/isotopes/overlaps.h	453
include/atomprobe/latticeplanes/generate.h	454
include/atomprobe/latticeplanes/millerIndex.h	455
include/atomprobe/primitives/boundcube.h	455
include/atomprobe/primitives/ionHit.h	456
include/atomprobe/primitives/mesh.h	457
include/atomprobe/primitives/point3D.h	458
include/atomprobe/projection/projection.h	459
include/atomprobe/reconstruction/reconstruction-simple.h	459
include/atomprobe/spectrum/processing.h	460
include/atomprobe/statistics/confidence.h	461
src/atomprobe.cpp	475
src/algorithm/axialdf.cpp	466
src/algorithm/componentAnalysis.cpp	467
src/algorithm/convexHull.cpp	467
src/algorithm/filter.cpp	468
src/algorithm/histogram.cpp	469
src/algorithm/isoSurface.cpp	470
src/algorithm/K3DTree-approx.cpp	471
src/algorithm/K3DTree-exact.cpp	472
src/algorithm/massTool.cpp	473
src/algorithm/rangeCheck.cpp	474
src/algorithm/rotations.cpp	474
src/deconvolution/deconvolution.cpp	476
src/helper/aptAssert.cpp	476
src/helper/composition.cpp	477
src/helper/convert.cpp	478
src/helper/helpFuncs.cpp	479
src/helper/helpFuncs.h	479
src/helper/progress.cpp	486
src/helper/sampling.cpp	487
src/helper/stringFuncs.cpp	487
src/helper/voxels.cpp	488
src/helper/XMLHelper.cpp	489
src/helper/mathsf/fsr.cpp	481
src/helper/mathsf/mathsf.cpp	482
src/helper/mathsf/misc.cpp	484
src/helper/mathsf/quat.cpp	485
src/io/dataFiles.cpp	490
src/io/multiRange.cpp	491
src/io/ranges.cpp	493

src/isotopes/abundance.cpp	494
src/isotopes/massconvert.cpp	494
src/isotopes/overlaps.cpp	495
src/lattice/generate.cpp	496
src/lattice/millerIndex.cpp	497
src/primitives/boundcube.cpp	497
src/primitives/ionhit.cpp	498
src/primitives/mesh.cpp	498
src/primitives/point3D.cpp	500
src/projection/projection.cpp	500
src/reconstruction/reconstruction-simple.cpp	501
src/spectrum/processing.cpp	502
src/statistics/confidence.cpp	503
test/KDTest.cpp	504
test/rangetest.cpp	508
test/test.h	509
test/unittests.cpp	510

Chapter 5

Namespace Documentation

5.1 AtomProbe Namespace Reference

Classes

- class [AbundanceData](#)
Class to load abundance information for natural isotopes.
- struct [ATO_ENTRY](#)
- class [AxisCompare](#)
Functor allowing for sorting of points in 3D.
- struct [BACKGROUND_PARAMS](#)
- class [BodyCentredCubicGen](#)
- class [BoundCube](#)
Helper class to define a bounding cube.
- class [ComparePairFirst](#)
- class [ComparePairFirstReverse](#)
- class [ComparePairSecond](#)
- class [CrystalGen](#)
- class [EPOS_ENTRY](#)
- class [FaceCentredCubicGen](#)
- struct [FLATTENED_RANGE](#)
Structure that allows for the multirange data to be mapped into.
- class [GnomonicProjection](#)
- class [IonHit](#)
This is a data holding class for POS file ions, from.
- struct [IONHIT](#)
Record as stored in a .POS file.
- struct [ISOTOPE_ENTRY](#)
- class [K3DNodeApprox](#)
Node Class for storing point.
- class [K3DNodeExact](#)
Node Class for storing point.
- class [K3DTreeApprox](#)
3D specific KD tree
- class [K3DTreeExact](#)
3D specific KD tree

- class [LibVersion](#)
Class to hold the library version data.
- class [LINE](#)
- class [LinearFeedbackShiftReg](#)
This class implements a Linear Feedback Shift Register (in software)
- class [MassTool](#)
Class that brute-force solves the knapsack problem.
- class [Mesh](#)
Simple mesh class for storing and manipulating meshes consisting of 2 and 3D simplexes (triangles or tetrahedra)
- class [MILLER_TRIPLET](#)
- class [ModifiedFocusSphericProjection](#)
- class [MultiRange](#)
Data storage and retrieval class for "ranging" a spectra, where overlapping ranges are permitted.
- struct [OVERLAP_PROBLEM_SETTINGS](#)
- class [Point3D](#)
A 3D point data class storage.
- struct [Point3f](#)
Data storage structure for points.
- class [ProgressBar](#)
- struct [Quaternion](#)
Data storage structure for quaternions.
- class [RandNumGen](#)
- struct [RANGE_MOLECULE](#)
- class [RangeFile](#)
Data storage and retrieval class for various range files.
- class [ReconstructionSphereOnCone](#)
- class [RGBf](#)
Data holder for colour as float.
- struct [SIMPLE_SPECIES](#)
- class [SimpleCubicGen](#)
- struct [SINGLE_HIT](#)
Single 3DAP hit event.
- class [SphericPlaneProjection](#)
- struct [SphericProjectionParams](#)
- class [StereographicProjection](#)
- class [TETRAHEDRON](#)
- struct [THREEDAP_DATA](#)
experimental setup data
- struct [THREEDAP_EXPERIMENT](#)
Data structure that contains the experiment information present in a 3Dap file.
- class [TRIANGLE](#)
- struct [TriangleWithIndexedVertices](#)
- class [TriangleWithVertexNorm](#)
- struct [VOLTAGE_DATA](#)
Voltage data structure – these updates occur periodically during the experiment.
- class [Voxels](#)
Template class that stores 3D voxel data.
- class [Weight](#)
Placeholder class for containing input weights for [MassTool](#).
- struct [ZECH_ROOT](#)

Typedefs

- typedef struct `AtomProbe::ATO_ENTRY` `ATO_ENTRY`
- typedef struct `AtomProbe::IONHIT` `IONHIT`

Record as stored in a .POS file.

Enumerations

- enum { `HULL_SURFREDUCE_NEGATIVE_SCALE_FACT = 1`, `HULL_ERR_USER_ABORT`, `HULL_ERR_NO_MEM` }
- enum `PointDir` { `POINTDIR_TOGETHER = 0`, `POINTDIR_IN_COMMON`, `POINTDIR_APART` }
- enum { `BOUND_CLIP = 1`, `BOUND_HOLD`, `BOUND_DERIV_HOLD`, `BOUND_MIRROR`, `BOUND_ZERO`, `BOUND_ENUM_END` }
Boundary clipping mode.
- enum { `VOX_INTERP_NONE`, `VOX_INTERP_LINEAR`, `VOX_INTERP_ENUM_END` }
- enum { `ADJ_ALL`, `ADJ_PLUS` }
- enum { `VOXELS_BAD_FILE_READ = 1`, `VOXELS_BAD_FILE_OPEN`, `VOXELS_BAD_FILE_SIZE`, `VOXELS_OUT_OF_MEMORY` }
- enum { `CLIP_NONE = 0`, `CLIP_LOWER_SOUTH_WEST`, `CLIP_UPPER_NORTH_EAST` }
Clipping direction constants.
- enum { `VOXEL_ABORT_ERR`, `VOXEL_MEMORY_ERR`, `VOXEL_BOUNDS_INVALID_ERR` }
- enum { `PROP_PARSE_ERR = 1`, `PROP_BAD_ATT` }
- enum { `RECORDREAD_ERR_GET_FILESIZE = 1`, `RECORDREAD_ERR_FILESIZE_MODULO`, `RECORDREAD_ERR_FILE_OPEN`, `RECORDREAD_ERR_NOMEM`, `RECORDREAD_BAD_FILEREAD`, `RECORDREAD_ERR_CHUNKOFFSET`, `RECORDREAD_BAD_RECORD` }
- enum { `ATO_OPEN_FAIL = 1`, `ATO_EMPTY_FAIL`, `ATO_SIZE_ERR`, `ATO_VERSIONCHECK_ERR`, `ATO_MEM_ERR`, `ATO_BAD_ENDIAN_DETECT`, `ATO_ENUM_END` }
- enum { `TEXT_ERR_FILE_OPEN = 1`, `TEXT_ERR_FILE_NUM_FIELDS`, `TEXT_ERR_FILE_FORMAT` }
- enum { `OPSREADER_FORMAT_CLINE_ERR = 1`, `OPSREADER_FORMAT_DETECTORLINE_ERR`, `OPSREADER_FORMAT_LINE_DASH_ERR`, `OPSREADER_FORMAT_LINE_VOLTAGE_ERR`, `OPSREADER_FORMAT_LINE_VOLTAGE_NOBETA`, `OPSREADER_FORMAT_LINE_VOLTAGE_DATA_ERR`, `OPSREADER_FORMAT_LINE_VOLTAGE_DATACOUNT_ERR`, `OPSREADER_FORMAT_LINETYPE_ERR`, `OPSREADER_FORMAT_CHANNELS_ERR`, `OPSREADER_CHANNELS_DATA_ERR`, `OPSREADER_FORMAT_SLINE_EVENTCOUNT_ERR`, `OPSREADER_FORMAT_SLINE_EVENTDATA_ERR`, `OPSREADER_FORMAT_SLINE_FORMAT_ERR`, `OPSREADER_FORMAT_SLINE_PREFIX_ERR`, `OPSREADER_FORMAT_DUPLICATE_SYSDATA`, `OPSREADER_FORMAT_DUPLICATE_DETECTORSIZE`, `OPSREADER_FORMAT_TRAILING_DASH_ERR`, `OPSREADER_FORMAT_DOUBLEDASH`, `OPSREADER_OPEN_ERR`, `OPSREADER_READ_ERR`, `OPSREADER_ABORT_ERR`, `OPSREADER_ENUM_END` }
Error codes for OPS file loading.
- enum { `MULTIRANGE_FORMAT_XML` }
- enum { `RANGE_FORMAT_ORNL`, `RANGE_FORMAT_DBL_ORNL`, `RANGE_FORMAT_ENV`, `RANGE_FORMAT_RRNG`, `RANGE_FORMAT_END_OF_ENUM` }

- enum {
 - RANGE_ERR_FORMAT =1, RANGE_ERR_FORMAT_HEADER, RANGE_ERR_EMPTY, RANGE_ERR_FORMAT_LONGNAME,
 - RANGE_ERR_FORMAT_SHORTNAME, RANGE_ERR_FORMAT_COLOUR, RANGE_ERR_FORMAT_TABLESEPARATOR, RANGE_ERR_FORMAT_MASS_PAIR,
 - RANGE_ERR_FORMAT_TABLE_ENTRY, PARSE_RANGE_FORMAT_GENERIC_FAIL, RANGE_ERR_DATA_TOO_MANY_USELESS_RANGES, RANGE_ERR_DATA_FLIPPED,
 - RANGE_ERR_DATA_INCONSISTENT, RANGE_ERR_DATA_NOMAPPED_IONNAME, RANGE_ERR_OPEN, RANGE_ERR_FORMAT_RANGETABLE,
 - RANGE_ERR_FORMAT_TABLEHEADER_NUMIONS, RANGE_ERR_NONUNIQUE_POLYATOMIC, RANGE_ERR_TOO_LARGE, RANGE_ERR_DASHHEADER,
 - RANGE_ERR_IONBLOCK_CONTENT, RANGE_ERR_NUMIONS, RANGE_ERR_NUMIONS_DUPLICATED, RANGE_ERR_TOO_MANYIONS,
 - RANGE_ERR_ION_BLOCK_NOT_PRESENT, RANGE_ERR_DUPLICATE_NUMRANGES, RANGE_ERR_NUMRANGE_PARSE, RANGE_ERR_RRNG_IONS_TOOSHORT,
 - RANGE_ERR_RRNG_COLON_SEPARATOR, RANGE_ERR_BADCOLOUR, RANGE_ERR_ION_NOT_MAPPED, RANGE_ERR_BADSTART,
 - RANGE_ERR_BADEND, RANGE_ERR_NAME_EMPTY, RANGE_ERR_BAD_LINE_RANGEBLOCK, RANGE_ERR_MISSING_IONBLOCK,
 - RANGE_ERR_NO_RANGES, RANGE_ERR_NO_BASIC_IONS, RANGE_ERR_MISMATCHED_NUMRANGES, RANGE_ERR_RANGEBLOCK_FORMAT,
 - RANGE_ERR_BAD_MULTIPPLICITY, RANGE_ERR_FORMAT_EMPTY_RANGEROW, RANGE_ERR_FILESIZE, RANGE_ERR_ENUM_END }
- enum {
 - COMPPARSE_BAD_INPUT_FORMAT =1, COMPPARSE_BAD_COMP_VALUE, COMPPARSE_BAD_BAL_USAGE, COMPPARSE_BAD_SYMBOL,
 - COMPPARSE_DUPLICATE_SYMBOL, COMPPARSE_BAD_TOTAL_COMP }

composition parsing error messages
- enum { CRYSTAL_BAD_UNIT_CELL, CRYSTAL_ENUM_END }
- enum { LATTICE_FCC, LATTICE_BCC, LATTICE_HCP, LATTICE_END_OF_ENUM }
- enum { ELEMENT_TRIANGLE =1, ELEMENT_TETRAHEDRON =2, ELEMENT_LINE =4 }
- enum { EVOLUTION_MODE_SHANKANGLE }
- enum { BACK_FIT_MODE_CONST_TOF, BACK_FIT_MODE_ENUM_END }

Background fitting modes.

- enum {
 - POS_ALLOC_FAIL =1, POS_OPEN_FAIL, POS_SIZE_MODULUS_ERR, POS_SIZE_EMPTY_ERR, POS_READ_FAIL, POS_NAN_LOAD_ERROR, POS_FILE_ENUM_END }
- enum { TAPSIM_FILE_FORMAT_FAIL =1, TAPSIM_OPEN_FAIL }
- enum {
 - MULTIRANGE_ERR_NO_PARSER =1, MULTIRANGE_ERR_NO_XML_DOC, MULTIRANGE_ERR_NO_XML_CONTEXT, MULTIRANGE_ERR_NO_NODEPTR,
 - MULTIRANGE_NOT_VALID_ROOTNODE, MULTIRANGE_ERR_NO_CONTENT, MULTIRANGE_ERR_NO_TIMESTAMP, MULTIRANGE_ERR_NO_MOLECULES,
 - MULTIRANGE_ERR_NO_MOLECULE_ENTRIES, MULTIRANGE_ERR_MOL_MISSING_ENTRY, MULTIRANGE_ERR_MOL_MISSING_NAME, MULTIRANGE_ERR_MOL_MISSING_COMPONENTS,
 - MULTIRANGE_ERR_BAD_RANGE, MULTIRANGE_ERR_BAD_RANGES, MULTIRANGE_ERR_MOL_MISSING_COLOUR, MULTIRANGE_ERR_MOL_BAD_COLOUR,
 - MULTIRANGE_ERR_NO_RANGES, MULTIRANGE_ERR_BAD_GROUP, MULTIRANGE_ERR_SELF_INCONSISTENT, MULTIRANGE_ERR_RANGE_NOPARENT,
 - MULTIRANGE_ERR_ENUM_END }
- enum {
 - MESH_LOAD_UNSPECIFIED_ERROR =1, MESH_LOAD_BAD_NODECOUNT, MESH_LOAD_BAD_ELEMENTCOUNT, MESH_LOAD_IS_INSANE,
 - MESH_LOAD_ENUM_END }

Functions

- unsigned int [generate1DaxialDistHist](#) (const std::vector< [Point3D](#) > &pointList, [K3DTreeExact](#) &tree, const [Point3D](#) &axisDir, float distMax, std::vector< unsigned int > &histogram)
Generate a 1D axial distribution function,.
- unsigned int [generate1DaxialDistHistSweep](#) (const std::vector< [Point3D](#) > &pointList, [K3DTreeExact](#) &tree, float distMax, float dTheta, float dPhi, [ProgressBar](#) &prog, std::vector< std::vector< std::vector< unsigned int > > > &histogram)
Generate a series of 1D distribution functions, one per pixel in a 2D grid of spherical coordinate directions.
- void [computeIonDistAdjacency](#) (const [MultiRange](#) &mrf, const [AbundanceData](#) &abundance, const [OVERLAP_PROBLEM_SETTINGS](#) &settings, gsl_matrix *m, gsl_vector *vOwnership, gsl_vector *vMass)
Calculate the adjacency matrix for the ions in a multirange.
- unsigned int [GetReducedHullPts](#) (const std::vector< [IonHit](#) > &points, float reductionDim, unsigned int *progress, bool(*callback)(bool), std::vector< [IonHit](#) > &pointResult)
Obtain a set of points that are a subset of points the convex hull that are at least "reductionDim" away from the hull itself.
- unsigned int [computeConvexHull](#) (const std::vector< [IonHit](#) > &data, unsigned int *progress, bool(*callback)(bool), std::vector< [Point3D](#) > &curHull, bool freeHull)
Obtain the convex hull of a set of ions.
- void [filterPeakNeedBiggerObs](#) (const [AbundanceData](#) &massTable, const std::vector< float > &peakData, float tolerance, size_t solutionCharge, std::vector< std::vector< [ISOTOPE_ENTRY](#) > > &solutions)
- void [filterBySolutionPPM](#) (const [AbundanceData](#) &massTable, float minPpm, std::vector< std::vector< [ISOTOPE_ENTRY](#) > > &solutions)
Use the maximum possible PPM for each isotopic combination to filter possible solutions.
- std::vector< float > [maxExplainedFraction](#) (const std::vector< std::pair< float, float > > &massData, float peakMass, float massWidth, const std::vector< std::vector< [ISOTOPE_ENTRY](#) > > &solutions, const [AbundanceData](#) &massTable, float massDistTol, unsigned int solutionCharge)
Compute the fraction of the data that has been explained, using the natural abundance information, and intensity data.
- bool [correlationHistogram](#) (const std::vector< [EPOS_ENTRY](#) > &eposIons, std::vector< std::vector< unsigned int > > &histogram, float stepMass, float endMass, bool lowerTriangular=false)
Generates an ion "correlation histogram" from a vector of EPOS ions. The input vector MUST
- bool [accumulateCorrelationHistogram](#) (const std::vector< [EPOS_ENTRY](#) > &eposIons, std::vector< std::vector< unsigned int > > &histogram, float stepMass, float endMass, bool lowerTriangular=true)
Increments a correlation histogram, as per correlationHistogram,.
- template<class T >
void [linearHistogram](#) (const std::vector< T > &data, T start, T end, T step, std::vector< T > &histVals)
- void [marchingCubes](#) (const [Voxels](#)< float > &v, float isoValue, std::vector< [TriangleWithVertexNorm](#) > &tVec)
- void [checkMassRangingCorrectness](#) (const [RangeFile](#) &rng, [AbundanceData](#) &massTable, float massTolerance, unsigned int maxChargeState, unsigned int maxComponents, std::vector< bool > &badRanges)
Ensure that each mass given spans a peak that should exist.
- void [computeRotationMatrix](#) (const [Point3D](#) &ur1, const [Point3D](#) &ur2, const [Point3D](#) &r1, const [Point3D](#) &r2, gsl_matrix *m)
- void [setHardAssert](#) (bool enabled)
Do assertions cause a straight up crash (enabled), or write a message to stderr (disabled)? By default, hard crash.
- void [askAssert](#) (const char *, unsigned int)
Either abort program, or ask the user what to do for an assertion. depending on hardAssert setting.
- void [computeComposition](#) (const std::vector< unsigned int > &countData, std::vector< float > &compositionData)
- void [convertEPOSstoPos](#) (const std::vector< [EPOS_ENTRY](#) > &eposEntry, std::vector< [IonHit](#) > &posFile)
Convert an incoming entry of EPOS files to pos, and the append this to the pos vector given.
- unsigned int [estimateRank](#) (const gsl_matrix *m, float tolerance=sqrt(std::numeric_limits< float >::epsilon()))
Estimate the rank of the given matrix.

- bool [solveLeastSquares](#) (gsl_matrix *m, gsl_vector *b, gsl_vector *&x)
Use an SVD based least-squares solver to solve $Mx=b$ (for x).
- template<class T >
T [weightedMean](#) (const std::vector< T > &values, const std::vector< T > &weight)
- template<typename T >
void [meanAndStdev](#) (const std::vector< T > &f, float &meanVal, float &stdevVal, bool normalCorrection=true)
- unsigned int [vectorPointDir](#) (const [Point3D](#) &pA, const [Point3D](#) &pB, const [Point3D](#) &vC, const [Point3D](#) &vD)
Check which way vectors attached to two 3D points "point",.
- float [distanceToSegment](#) (const [Point3D](#) &fA, const [Point3D](#) &fB, const [Point3D](#) &p)
- float [signedDistanceToFacet](#) (const [Point3D](#) &fA, const [Point3D](#) &fB, const [Point3D](#) &fC, const [Point3D](#) &normal, const [Point3D](#) &p)
Find the distance between a point, and a triangular facet – may be positive or negative.
- float [distanceToFacet](#) (const [Point3D](#) &fA, const [Point3D](#) &fB, const [Point3D](#) &fC, const [Point3D](#) &normal, const [Point3D](#) &p)
- float [dotProduct](#) (float a1, float a2, float a3, float b1, float b2, float b3)
Inline func for calculating $a(\text{dot})b$.
- double [det3by3](#) (const double *ptArray)
- double [pyramidVol](#) (const [Point3D](#) *planarPts, const [Point3D](#) &apex)
- bool [trilsDegenerate](#) (const [Point3D](#) &fA, const [Point3D](#) &fB, const [Point3D](#) &fC)
- void [quat_rot](#) ([Point3D](#) &p, const [Point3D](#) &r, float angle)
Rotate a point around a given rotation axis by a specified angle.
- void [quat_rot](#) ([Point3f](#) *point, const [Point3f](#) *rotVec, float angle)
Rotate a point around a given vector, with specified angle.
- void [quat_rot_array](#) ([Point3f](#) *point, unsigned int n, const [Point3f](#) *rotVec, float angle)
Rotate each point in array of size n around a given vector, with specified angle.
- void [quat_rot_array](#) ([Point3D](#) *point, unsigned int n, const [Point3f](#) *rotVec, float angle)
Rotate each point in array of size n around a given vector, with specified angle.
- void [quat_get_rot_quat](#) (const [Point3f](#) *rotVec, float angle, [Quaternion](#) *rotQuat)
Compute the quaternion for specified rotation.
- void [quat_rot_apply_quat](#) ([Point3f](#) *point, const [Quaternion](#) *rotQuat)
Use previously generated quats from `quat_get_rot_quats` to rotate a point.
- void [applyQuaternionRotation](#) (std::vector< [Point3D](#) > &pts, const [Quaternion](#) &q)
Apply the given quaternion rotation to the input points.
- template<class T >
bool [tolEqual](#) (const T &a, const T &b, const T &f)
- template<class T >
void [selectElements](#) (const std::vector< T > &in, const std::vector< unsigned int > &indices, std::vector< T > &out)
- template<class T >
void [transposeVector](#) (std::vector< std::vector< T > > &v)
- template<class T >
void [vectorMultiErase](#) (std::vector< T > &vec, const std::vector< bool > &wantKill)
- void [sampleIons](#) (const std::vector< [IonHit](#) > &ions, float sampleFactor, std::vector< [IonHit](#) > &sampld, bool strongRandom=true)
- template<class T >
size_t [randomSelect](#) (std::vector< T > &result, const std::vector< T > &source, size_t num, gsl_rng *rng)
- template<class T >
void [randomIndices](#) (std::vector< T > &res, size_t num, size_t nMax, gsl_rng *rng)
- template<class T1, class T2 >
bool [stream_cast](#) (T1 &result, const T2 &obj)
Template function to cast and object to another by the stringstream.
- template<class T >
void [strAppend](#) (std::string s, const T &a)
- void [genColString](#) (unsigned char r, unsigned char g, unsigned char b, unsigned char a, std::string &s)

- void [genColString](#) (unsigned char r, unsigned char g, unsigned char b, std::string &s)
- bool [parseColString](#) (const std::string &str, unsigned char &r, unsigned char &g, unsigned char &b, unsigned char &a)
 - Parse a colour string containing rgb[a]; hex for with leading #.*
- std::string [digitString](#) (unsigned int thisDigit, unsigned int maxDigit)
 - Generate a string with leading digits up to maxDigit (eg, if maxDigit is 424, and thisDigit is 1.*
- std::string [stripWhite](#) (const std::string &str)
 - Strip whitespace, (eg tab,space) from either side of a string.*
- std::string [stripChars](#) (const std::string &Str, const char *chars)
- std::string [lowercase](#) (std::string s)
 - Return a lowercase version for a given string.*
- std::string [uppercase](#) (std::string s)
 - Return a uppercase version for a given string.*
- void [stripZeroEntries](#) (std::vector< std::string > &s)
- void [splitStrsRef](#) (const char *cpStr, const char delim, std::vector< std::string > &v)
 - Split string references using a single delimiter.*
- void [splitStrsRef](#) (const char *cpStr, const char *delim, std::vector< std::string > &v)
 - Split string references using any of a given string of delimiters.*
- std::string [onlyFilename](#) (const std::string &path)
 - Return only the filename component.*
- std::string [onlyDir](#) (const std::string &path)
 - Return only the directory name component of the full path.*
- std::string [convertFileStringToNative](#) (const std::string &s)
 - Convert a path format into a native path from unix format.*
- std::string [convertFileStringToCanonical](#) (const std::string &s)
 - Convert a path format into a unix path from native format.*
- std::string [tabs](#) (unsigned int nTabs)
- std::string [stlWStrToStlStr](#) (const std::wstring &s)
- std::wstring [stlStrToStlWStr](#) (const std::string &s)
- void [nullifyMarker](#) (char *buffer, char marker)
- template<class T, class U >
 - void [castVoxels](#) (const [Voxels](#)< T > &src, [Voxels](#)< U > &dest)
 - Convert one type of voxel into another by assignment operator.*
- template<class T, class U >
 - void [sumVoxels](#) (const [Voxels](#)< T > &src, U &counter)
 - Use one counting type to sum counts in a voxel of given type.*
- unsigned int [XMLHelpNextType](#) (xmlNodePtr &node, int)
- unsigned int [XMLHelpFwdToElem](#) (xmlNodePtr &node, const char *nodeName)
- unsigned int [XMLHelpFwdNotElem](#) (xmlNodePtr &node, const char *nodeName)
- unsigned int [XMLHelpFwdToList](#) (xmlNodePtr &node, const std::vector< std::string > &nodeList)
- std::string [XMLHelpGetText](#) (xmlNodePtr &node)
- void [XMLFreeDoc](#) (void *data)
 - Free a xmlDoc pointer. For use in conjunction with std::unique_ptr for auto-deallocation.*
- template<class T >
 - unsigned int [XMLHelpGetProp](#) (T &prop, xmlNodePtr node, std::string propName)
- template<class T >
 - bool [XMLGetNextElemAttrib](#) (xmlNodePtr &nodePtr, T &v, const char *nodeName, const char *attrib)
 - Grab the specified attribute from the next element, then [stream_cast\(\)](#) it into the passed-in object. Returns true on success, false on error.*
- const char * [getRecordReadErrString](#) (unsigned int errCode)
- const char * [getAtoErrString](#) (unsigned int errCode)
- unsigned int [loadPosFile](#) (std::vector< [IonHit](#) > &poslons, const char *posFile)

- Load a pos file directly into a single ion list.*

 - unsigned int [loadPosFile](#) (std::vector< [IonHit](#) > &posIons, const char *posFile, unsigned int nSamplesMax)

As per loadPosFile, but with an additional setting for the maximum number of ions to load.

 - const char * [getPosFileErrMsg](#) (unsigned int errMesg)
 - unsigned int [savePosFile](#) (const std::vector< [Point3D](#) > &points, float mass, const char *name, bool append=false)

Save a vector of Point3Ds into a pos file, using a fixed mass, return nonzero on error.

 - unsigned int [savePosFile](#) (const std::vector< [IonHit](#) > &data, const char *name, bool append=false)

Save a vector of IonHits into a "pos" file, return nonzero on error.

 - unsigned int [saveTapsimBin](#) (std::ostream &f, std::vector< [IonHit](#) > &posIons)

Write a tapsim file from a bunch of IonHits.

 - size_t [loadEposFile](#) (std::vector< [EPOS_ENTRY](#) > &outData, const char *filename)

Load an entire "EPOS" File.

 - size_t [chunkLoadEposFile](#) (std::vector< [EPOS_ENTRY](#) > &outData, const char *filename, unsigned int chunkSize, unsigned int chunkOffset, unsigned int &nEntriesLeft)

Load an "EPOS" file, with a maximum chunk size.

 - unsigned int [loadTextData](#) (const char *cpFilename, std::vector< std::vector< float > > &dataVec, std::vector< std::string > &headerVec, const char *delim, bool allowNan=true)

Load a CSV, TSV or similar text file. Assumes "C" Locale for input (ie "." as decimal separator).

 - unsigned int [readPosapOps](#) (const char *file, [THREEDAP_EXPERIMENT](#) &data, unsigned int &badLine, unsigned int &progress, std::atomic< bool > &wantAbort, unsigned int nDelayLines=2, bool strictMode=false)

Function to read POSAP "OPS" files.

 - unsigned int [loadATOFFile](#) (const char *fileName, std::vector< [ATO_ENTRY](#) > &ions, unsigned int forceEndian=0)

Load a LAWATAP "ATO" file.

 - bool [operator<](#) (const [SIMPLE_SPECIES](#) &a, const [SIMPLE_SPECIES](#) &b)
 - unsigned int [parseCompositionData](#) (const std::vector< std::string > &s, const [AbundanceData](#) &atomTable, std::map< unsigned int, double > &atomData)

Convert atomic string label data, in the form "Fe 0.81" to fraction map.

 - void [convertMolToMass](#) (const [AbundanceData](#) &atomTable, const std::map< unsigned int, double > &compositionMols, std::map< unsigned int, double > &compositionMass)

Convert the given composition from molar fraction data to mass fraction.

 - void [convertMassToMol](#) (const [AbundanceData](#) &atomTable, const std::map< unsigned int, double > &compositionMass, std::map< unsigned int, double > &compositionMols)

Convert the given composition from mass fraction data to molar fraction.

 - void [findOverlaps](#) (const [AbundanceData](#) &natData, const [RangeFile](#) &rng, float massDelta, unsigned int maxCharge, std::vector< std::pair< size_t, size_t > > &overlapIdx, std::vector< std::pair< float, float > > &overlapMasses)

Find the overlaps stemming from a given rangefile.

 - void [findOverlaps](#) (const [AbundanceData](#) &natData, const std::vector< std::string > &ions, float massDelta, unsigned int maxCharge, std::vector< std::pair< size_t, size_t > > &overlapIdx, std::vector< std::pair< float, float > > &overlapMasses)

As per findOverlaps(...Rangefile ...) , but uses a vector of ions instead.

 - void [findOverlapsFromSpectra](#) (const std::vector< std::vector< std::pair< float, float > > > &massDistributions, float massDelta, const std::set< unsigned int > &skipIDs, std::vector< std::pair< size_t, size_t > > &overlapIdx, std::vector< std::pair< float, float > > &overlapMasses)

Find overlaps in the given mass distributions, as per findOverlaps(...Rangefile...)

 - void [gsl_matrix_mult](#) (const gsl_matrix *a, const gsl_matrix *b, gsl_matrix *&res)
 - bool [reconstructTest](#) ()
 - unsigned int [doFitBackground](#) (const std::vector< float > &massData, [BACKGROUND_PARAMS](#) ¶ms)

Perform a background fit, assuming constant TOF noise, from 0->inf time.

 - std::string [getFitErrorMsg](#) (unsigned int errCode)

- void [createMassBackground](#) (float massStart, float massEnd, unsigned int nBinsMass, float tofBackIntensity, std::vector< float > &histogram)
 - Build a histogram of the background counts.*
- void [findPeaks](#) (const std::vector< float > &x0, std::vector< unsigned int > &peakInds, float sel, bool autoSel=true, bool includeEndpoints=true)
 - Simple peak-finding algorithm.*
- void [poissonConfidenceObservation](#) (float counts, float alpha, float &lBound, float &uBound)
 - Obtain poisson confidence limits for the mean of a poisson distribution.*
- bool [numericalEstimatePoissRatioConf](#) (float lambda1, float lambda2, float alpha, unsigned int nTrials, gsl_rng *r, float &lBound, float &uBound)
 - Brute-force poisson ratio confidence estimator. Returns the confidence interval in the estimate of the mean, at the given rates.*
- bool [numericalEstimateGaussRatioConf](#) (float mu1, float mu2, float var1, float var2, float alpha, unsigned int nTrials, gsl_rng *r, float &lBound, float &uBound)
 - Brute-force gaussian ratio confidence estimator.*
- bool [numericalEstimateSkellamConf](#) (float lambda1, float lambda2, float alpha, unsigned int nTrials, gsl_rng *r, float &lBound, float &uBound)
 - Brute-force the confidence bounds of a skellam distribution.*
- bool [zechConfidenceLimits](#) (float lambdaBack, unsigned int observation, float alpha, float &estimate)
 - Provides a best estimate for true signal when true signal, background.*
- bool [rangeOverlaps](#) (const pair< float, float > &r1, const pair< float, float > &r2)
- void [computeRangeAdjacency](#) (const [MultiRange](#) &mrf, const [OVERLAP_PROBLEM_SETTINGS](#) &settings, gsl_matrix *&m)
- void [freeConvexHull](#) ()
- unsigned int [doHull](#) (unsigned int bufferSize, double *buffer, vector< [Point3D](#) > &resHull, [Point3D](#) &midPoint, bool freeHullOnExit)
- unsigned int [countIntensityEvents](#) (const vector< pair< float, float > > &data, float minV, float maxV)
- void [buildFrequencyTable](#) (const vector< [ISOTOPE_ENTRY](#) > &solutionVec, vector< size_t > &solutionElements, vector< size_t > &solutionFrequency)
- vector< float > [maxExplainedFraction](#) (const vector< pair< float, float > > &intensityData, float peakMass, float massWidth, const vector< vector< [ISOTOPE_ENTRY](#) > > &solutions, const [AbundanceData](#) &massTable, float massDistTol, unsigned int solutionCharge)
- bool [incrementCorrelationHist](#) (const std::vector< [EPOS_ENTRY](#) > &eposIons, std::vector< std::vector< unsigned int > > &histogram, float stepMass, float endMass)
- template<class T >
 - void [removeElements](#) (const std::vector< size_t > &elems, std::vector< T > &vec)
- bool [pairContains](#) (const pair< float, float > &p, float m)
- [RANGE_MOLECULE](#) [getRangeMolecule](#) (const [AbundanceData](#) &massTable, const [RangeFile](#) &rng, unsigned int rangeld)
- float [nullBackground](#) (float rangeStart, float rangeEnd)
- bool [leastSquaresDeconvolve](#) (const [MultiRange](#) &rangeData, const [AbundanceData](#) &abundance, const float massTol, const vector< [IonHit](#) > &hits, float(*backgroundEstimator)(float, float), vector< pair< unsigned int, float > > &decomposedHits)
- bool [getHardAssert](#) ()
- char * [myStrDup](#) (const char *s)
- void [pushLocale](#) (const char *newLocale, int type)
- void [popLocale](#) ()
- bool [getFileSize](#) (const char *fname, size_t &size)
- void [gsl_print_matrix](#) (const gsl_matrix *m)
- void [gsl_print_vector](#) (const gsl_vector *v)
- float [gsl_determinant](#) (const gsl_matrix *m)
- bool [isNotDirectory](#) (const char *filename)
- int [fpeek](#) (FILE *stream)
- void [quat_mult_no_second_a](#) ([Quaternion](#) *result, const [Quaternion](#) *q1, const [Quaternion](#) *q2)
- void [quat_pointmult](#) ([Point3f](#) *result, const [Quaternion](#) *q1, const [Quaternion](#) *q2)

- void [quat_invert](#) ([Quaternion](#) *quat)
- void [ucharToHexStr](#) (unsigned char c, std::string &s)
- void [hexStrToUChar](#) (const std::string &s, unsigned char &c)
- void [incrementDataDistanceWeight](#) (const [Point3D](#) &p, [Voxels](#)< float > &v)
- void [countDataDistanceWeight](#) (const vector< [Point3D](#) > &pts, [Voxels](#)< float > &v)
- string [XMLHelpGetText](#) (xmlNodePtr node)
- template<class T >
unsigned int [fixedRecordReader](#) (const char *filename, bool(*recordReader)(const char *bufRead, const char *destBuf), size_t recordSize, std::vector< T > &outputData)
- template<class T >
unsigned int [fixedRecordChunkReader](#) (const char *filename, bool(*recordReader)(const char *bufRead, const char *destBuf), size_t recordSize, std::vector< T > &outputData, unsigned int chunkSize, unsigned int chunkOffset, unsigned int &nEntriesLeft)
- unsigned int [loadTapsimBinFile](#) (vector< [IonHit](#) > &posIons, const char *posfile)
- bool [readEposRecord](#) (const char *src, const char *dest)
- bool [strhas](#) (const char *cpTest, const char *cpPossible)
- template<class T >
void [linkIdentifiers](#) (vector< T > &link)
- bool [pairOverlaps](#) (float aStart, float aEnd, float bStart, float bEnd)
- string [rangeToStr](#) (const pair< float, float > &rng)
- bool [matchComposedName](#) (const std::map< string, size_t > &composedNames, const vector< pair< string, size_t > > &namesToFind, size_t &matchOffset)
- void [findOverlapsFromSpectra](#) (const vector< vector< pair< float, float > > > &massDistributions, float massDelta, const set< unsigned int > &skipIDs, vector< pair< size_t, size_t > > &overlapIdx, vector< pair< float, float > > &overlapMasses)
- void [findOverlaps](#) (const [AbundanceData](#) &natData, const [RangeFile](#) &rng, float massDelta, unsigned int maxCharge, vector< pair< size_t, size_t > > &overlapIdx, vector< pair< float, float > > &overlapMasses)
- void [findOverlaps](#) (const [AbundanceData](#) &natData, const vector< string > &ionNames, float massDelta, unsigned int maxCharge, vector< pair< size_t, size_t > > &overlapIdx, vector< pair< float, float > > &overlapMasses)
- int [gcd](#) (int a, int b)
- std::ostream & [operator<<](#) (std::ostream &stream, const [BoundCube](#) &b)
- std::ostream & [operator<<](#) (std::ostream &stream, const [IonHit](#) &ion)
- bool [antiRotateMatch](#) (const unsigned int *a, const unsigned int *b, size_t n)
- bool [rotateMatch](#) (const unsigned int *a, const unsigned int *b, size_t n, bool directionForwards=true)
- float [signVal](#) (unsigned int val)
- size_t [findMaxLessThanOrEq](#) (const vector< std::pair< size_t, size_t > > &v, size_t value)
- float [Determinant](#) (float **a, int n)
- float [fourDeterminant](#) (const [Point3D](#) &a, const [Point3D](#) &b, const [Point3D](#) &c, const [Point3D](#) &d)
- unsigned int [edgeldx](#) (unsigned int i, unsigned int j)
- int [intersect_RayTriangle](#) (const [Point3D](#) &rayStart, const [Point3D](#) &rayEnd, [Point3D](#) *tri, [Point3D](#) &l)
- void [findNearVertices](#) (float tolerance, const vector< [Point3D](#) > &ptVec, vector< std::pair< size_t, vector< size_t > > > &clusterList)
- void [findNearVertices](#) (float tolerance, const vector< [Point3D](#) > &ptVec, std::list< std::pair< size_t, vector< size_t > > > &clusterList)
- std::ostream & [operator<<](#) (std::ostream &stream, const [Point3D](#) &pt)
- double [SphericProjectionEqn](#) (double eta, void *p)
- template<class T >
bool [andersonDarlingStatistic](#) (std::vector< T > vals, float &meanV, float &stdevVal, float &statistic, size_t &undefCount, bool computeMeanAndStdev=true)
- void [makeHistogram](#) (const vector< float > &data, float start, float end, float step, vector< float > &histVals)
- void [diff](#) (const vector< float > &in, vector< float > &out)
- template<class T, class T2 >
std::iterator_traits< T >::value_type [kahansum](#) (T begin, T end, T2 other)
- double [zechRoot](#) (double sigGuess, void *params)
- template<class T >
void [cumTrapezoid](#) (const vector< T > &x, const vector< T > &vals, vector< T > &res)
- double [gauss_ratio_pdf](#) (double x, double muX, double muY, double varX, double varY)

Variables

- [LibVersion](#) libVersion
- [RandNumGen](#) randGen
- const char * [RECORDREAD_ERR_STRINGS](#) []
- const char * [ATO_ERR_STRINGS](#) []
- *Human readable error messages for use with ATO reader return values.*
- const char * [OPS_ENUM_ERRSTRINGS](#) []
- const unsigned int [NUM_ELEMENTS](#) =119
- const size_t [EPOS_RECORD_SIZE](#) = 11*4
- const unsigned int [ELEM_SINGLE_NODE_POINT](#) =15
- const unsigned int [ELEM_TWO_NODE_LINE](#) =1
- const unsigned int [ELEM_THREE_NODE_TRIANGLE](#) =2
- const unsigned int [ELEM_FOUR_NODE_TETRAHEDRON](#) =4
- const char * [MESH_LOAD_ERRS](#) []
- bool [qhullInited](#) =false
- const unsigned int [HULL_GRAB_SIZE](#) =4096
- int [aiCubeEdgeFlags](#) [256]
- int [a2iTriangleConnectionTable](#) [256][16]
- const unsigned int [VERTEX_OFFSET](#) [8][3]
- int [edgeRemap](#) [12]
- bool [hardAssert](#) =true
- const size_t [maximumLinearTable](#) []
- const char [MULTIRANGE_FORMAT_VERSION](#) [] = "0.0.1"
- const size_t [MAX_LINE_SIZE](#) = 16536
- const size_t [MAX_RANGEFILE_SIZE](#) = 50*1024*1024
- const char * [RANGE_EXTS](#) []
- const char * [elementList](#) []
- const char * [ABUNDANCE_ERROR](#) []
- const size_t [PROGRESS_REDUCE](#) =500

5.1.1 Typedef Documentation

5.1.1.1 ATO_ENTRY

```
typedef struct AtomProbe::ATO_ENTRY AtomProbe::ATO_ENTRY
```

5.1.1.2 IONHIT

```
typedef struct AtomProbe::IONHIT AtomProbe::IONHIT
```

Record as stored in a .POS file.

5.1.2 Enumeration Type Documentation

5.1.2.1 anonymous enum

```
anonymous enum
```

Enumerator

HULL_SURFREDUCE_NEGATIVE_SCALE_FACT	
HULL_ERR_USER_ABORT	
HULL_ERR_NO_MEM	

Definition at line 11 of file convexHull.h.

5.1.2.2 anonymous enum

anonymous enum

Enumerator

ATO_OPEN_FAIL	
ATO_EMPTY_FAIL	
ATO_SIZE_ERR	
ATO_VERSIONCHECK_ERR	
ATO_MEM_ERR	
ATO_BAD_ENDIAN_DETECT	
ATO_ENUM_END	

Definition at line 49 of file dataFiles.h.

5.1.2.3 anonymous enum

anonymous enum

Enumerator

TEXT_ERR_FILE_OPEN	
TEXT_ERR_FILE_NUM_FIELDS	
TEXT_ERR_FILE_FORMAT	

Definition at line 60 of file dataFiles.h.

5.1.2.4 anonymous enum

anonymous enum

Error codes for OPS file loading.

Enumerator

OPSREADER_FORMAT_CLINE_ERR	
OPSREADER_FORMAT_DETECTORLINE_ERR	
OPSREADER_FORMAT_LINE_DASH_ERR	
OPSREADER_FORMAT_LINE_VOLTAGE_ERR	
OPSREADER_FORMAT_LINE_VOLTAGE_NOBETA	
OPSREADER_FORMAT_LINE_VOLTAGE_DATA_ERR	
OPSREADER_FORMAT_LINE_VOLTAGE_DATACOUNT_ERR	
OPSREADER_FORMAT_LINETYPE_ERR	
OPSREADER_FORMAT_CHANNELS_ERR	
OPSREADER_CHANNELS_DATA_ERR	
OPSREADER_FORMAT_SLINE_EVENTCOUNT_ERR	
OPSREADER_FORMAT_SLINE_EVENTDATA_ERR	
OPSREADER_FORMAT_SLINE_FORMAT_ERR	
OPSREADER_FORMAT_SLINE_PREFIX_ERR	
OPSREADER_FORMAT_DUPLICATE_SYSDATA	
OPSREADER_FORMAT_DUPLICATE_DETECTORSIZE	
OPSREADER_FORMAT_TRAILING_DASH_ERR	
OPSREADER_FORMAT_DOUBLEDASH	
OPSREADER_OPEN_ERR	
OPSREADER_READ_ERR	
OPSREADER_ABORT_ERR	
OPSREADER_ENUM_END	

Definition at line 168 of file dataFiles.h.

5.1.2.5 anonymous enum

anonymous enum

Enumerator

MULTIRANGE_FORMAT_XML	
-----------------------	--

Definition at line 57 of file multiRange.h.

5.1.2.6 anonymous enum

anonymous enum

Enumerator

RANGE_FORMAT_ORNL	
RANGE_FORMAT_DBL_ORNL	

Enumerator

RANGE_FORMAT_ENV	
RANGE_FORMAT_RRNG	
RANGE_FORMAT_END_OF_ENUM	

Definition at line 36 of file ranges.h.

5.1.2.7 anonymous enum

anonymous enum

Enumerator

RANGE_ERR_FORMAT	
RANGE_ERR_FORMAT_HEADER	
RANGE_ERR_EMPTY	
RANGE_ERR_FORMAT_LONGNAME	
RANGE_ERR_FORMAT_SHORTNAME	
RANGE_ERR_FORMAT_COLOUR	
RANGE_ERR_FORMAT_TABLESEPARATOR	
RANGE_ERR_FORMAT_MASS_PAIR	
RANGE_ERR_FORMAT_TABLE_ENTRY	
PARSE_RANGE_FORMAT_GENERIC_FAIL	
RANGE_ERR_DATA_TOO_MANY_USELESS_RANGES	
RANGE_ERR_DATA_FLIPPED	
RANGE_ERR_DATA_INCONSISTENT	
RANGE_ERR_DATA_NOMAPPED_IONNAME	
RANGE_ERR_OPEN	
RANGE_ERR_FORMAT_RANGETABLE	
RANGE_ERR_FORMAT_TABLEHEADER_NUMIONS	
RANGE_ERR_NONUNIQUE_POLYATOMIC	
RANGE_ERR_TOO_LARGE	
RANGE_ERR_DASHHEADER	
RANGE_ERR_IONBLOCK_CONTENT	
RANGE_ERR_NUMIONS	
RANGE_ERR_NUMIONS_DUPLICATED	
RANGE_ERR_TOO_MANYIONS	
RANGE_ERR_ION_BLOCK_NOT_PRESENT	
RANGE_ERR_DUPLICATE_NUMRANGES	
RANGE_ERR_NUMRANGE_PARSE	
RANGE_ERR_RRNG_IONS_TOOSHORT	
RANGE_ERR_RRNG_COLON_SEPARATOR	
RANGE_ERR_BADCOLOUR	
RANGE_ERR_ION_NOT_MAPPED	
RANGE_ERR_BADSTART	
RANGE_ERR_BADEND	
RANGE_ERR_NAME_EMPTY	

Enumerator

RANGE_ERR_BAD_LINE_RANGEBLOCK	
RANGE_ERR_MISSING_IONBLOCK	
RANGE_ERR_NO_RANGES	
RANGE_ERR_NO_BASIC_IONS	
RANGE_ERR_MISMATCHED_NUMRANGES	
RANGE_ERR_RANGEBLOCK_FORMAT	
RANGE_ERR_BAD_MULTIPLICITY	
RANGE_ERR_FORMAT_EMPTY_RANGEROW	
RANGE_ERR_FILESIZE	
RANGE_ERR_ENUM_END	

Definition at line 43 of file ranges.h.

5.1.2.8 anonymous enum

anonymous enum

composition parsing error messages

Enumerator

COMPPARSE_BAD_INPUT_FORMAT	
COMPPARSE_BAD_COMP_VALUE	
COMPPARSE_BAD_BAL_USAGE	
COMPPARSE_BAD_SYMBOL	
COMPPARSE_DUPLICATE_SYMBOL	
COMPPARSE_BAD_TOTAL_COMP	

Definition at line 10 of file massconvert.h.

5.1.2.9 anonymous enum

anonymous enum

Enumerator

CRYSTAL_BAD_UNIT_CELL	
CRYSTAL_ENUM_END	

Definition at line 80 of file generate.h.

5.1.2.10 anonymous enum

anonymous enum

Enumerator

LATTICE_FCC	
LATTICE_BCC	
LATTICE_HCP	
LATTICE_END_OF_ENUM	

Definition at line 11 of file millerIndex.h.

5.1.2.11 anonymous enum

anonymous enum

Boundary clipping mode.

These constants defines the behaviour of the routines when presented with a boundary value problem, such as in convolution.

Enumerator

BOUND_CLIP	
BOUND_HOLD	
BOUND_DERIV_HOLD	
BOUND_MIRROR	
BOUND_ZERO	
BOUND_ENUM_END	

Definition at line 46 of file voxels.h.

5.1.2.12 anonymous enum

anonymous enum

Enumerator

ELEMENT_TRIANGLE	
ELEMENT_TETRAHEDRON	
ELEMENT_LINE	

Definition at line 44 of file mesh.h.

5.1.2.13 anonymous enum

anonymous enum

Enumerator

EVOLUTION_MODE_SHANKANGLE	
---------------------------	--

Definition at line 29 of file reconstruction-simple.h.

5.1.2.14 anonymous enum

anonymous enum

Background fitting modes.

Enumerator

BACK_FIT_MODE_CONST_TOF	
BACK_FIT_MODE_ENUM_END	

Definition at line 22 of file processing.h.

5.1.2.15 anonymous enum

anonymous enum

Enumerator

POS_ALLOC_FAIL	
POS_OPEN_FAIL	
POS_SIZE_MODULUS_ERR	
POS_SIZE_EMPTY_ERR	
POS_READ_FAIL	
POS_NAN_LOAD_ERROR	
POS_FILE_ENUM_END	

Definition at line 53 of file dataFiles.cpp.

5.1.2.16 anonymous enum

anonymous enum

Enumerator

TAPSIM_FILE_FORMAT_FAIL	
TAPSIM_OPEN_FAIL	

Definition at line 65 of file dataFiles.cpp.

5.1.2.17 anonymous enum

anonymous enum

Enumerator

MULTIRANGE_ERR_NO_PARSER	
MULTIRANGE_ERR_NO_XML_DOC	
MULTIRANGE_ERR_NO_XML_CONTEXT	
MULTIRANGE_ERR_NO_NODEPTR	
MULTIRANGE_NOT_VALID_ROOTNODE	
MULTIRANGE_ERR_NO_CONTENT	
MULTIRANGE_ERR_NO_TIMESTAMP	
MULTIRANGE_ERR_NO_MOLECULES	
MULTIRANGE_ERR_NO_MOLECULE_ENTRIES	
MULTIRANGE_ERR_MOL_MISSING_ENTRY	
MULTIRANGE_ERR_MOL_MISSING_NAME	
MULTIRANGE_ERR_MOL_MISSING_COMPONENTS	
MULTIRANGE_ERR_BAD_RANGE	
MULTIRANGE_ERR_BAD_RANGES	
MULTIRANGE_ERR_MOL_MISSING_COLOUR	
MULTIRANGE_ERR_MOL_BAD_COLOUR	
MULTIRANGE_ERR_NO_RANGES	
MULTIRANGE_ERR_BAD_GROUP	
MULTIRANGE_ERR_SELF_INCONSISTENT	
MULTIRANGE_ERR_RANGE_NOPARENT	
MULTIRANGE_ERR_ENUM_END	

Definition at line 48 of file multiRange.cpp.

5.1.2.18 anonymous enum

anonymous enum

Enumerator

MESH_LOAD_UNSPECIFIED_ERROR	
-----------------------------	--

Enumerator

MESH_LOAD_BAD_NODECOUNT	
MESH_LOAD_BAD_ELEMENTCOUNT	
MESH_LOAD_IS_INSANE	
MESH_LOAD_ENUM_END	

Definition at line 44 of file mesh.cpp.

5.1.2.19 anonymous enum

anonymous enum

Enumerator

VOX_INTERP_NONE	
VOX_INTERP_LINEAR	
VOX_INTERP_ENUM_END	

Definition at line 56 of file voxels.h.

5.1.2.20 anonymous enum

anonymous enum

Enumerator

ADJ_ALL	
ADJ_PLUS	

Definition at line 63 of file voxels.h.

5.1.2.21 anonymous enum

anonymous enum

Enumerator

VOXELS_BAD_FILE_READ	
VOXELS_BAD_FILE_OPEN	
VOXELS_BAD_FILE_SIZE	
VOXELS_OUT_OF_MEMORY	

Definition at line 69 of file voxels.h.

5.1.2.22 anonymous enum

anonymous enum

Clipping direction constants.

Controls the clipping direction when performing clipping operations

Enumerator

CLIP_NONE	
CLIP_LOWER_SOUTH_WEST	
CLIP_UPPER_NORTH_EAST	

Definition at line 80 of file voxels.h.

5.1.2.23 anonymous enum

anonymous enum

Enumerator

VOXEL_ABORT_ERR	
VOXEL_MEMORY_ERR	
VOXEL_BOUNDS_INVALID_ERR	

Definition at line 87 of file voxels.h.

5.1.2.24 anonymous enum

anonymous enum

Enumerator

PROP_PARSE_ERR	
PROP_BAD_ATT	

Definition at line 36 of file XMLHelper.h.

5.1.2.25 anonymous enum

anonymous enum

Enumerator

RECORDREAD_ERR_GET_FILESIZE	
RECORDREAD_ERR_FILESIZE_MODULO	
RECORDREAD_ERR_FILE_OPEN	
RECORDREAD_ERR_NOMEM	
RECORDREAD_BAD_FILEREAD	
RECORDREAD_ERR_CHUNKOFFSET	
RECORDREAD_BAD_RECORD	

Definition at line 34 of file dataFiles.h.

5.1.2.26 PointDir

enum `AtomProbe::PointDir`

Enumerator

POINTDIR_TOGETHER	
POINTDIR_IN_COMMON	
POINTDIR_APART	

Definition at line 16 of file misc.h.

5.1.3 Function Documentation

5.1.3.1 accumulateCorrelationHistogram()

```
bool AtomProbe::accumulateCorrelationHistogram (
    const std::vector< EPOS_ENTRY > & eposIons,
    std::vector< std::vector< unsigned int > > & histogram,
    float stepMass,
    float endMass,
    bool lowerTriangular = true )
```

Increments a correlation histogram, as per correlationHistogram,.

For input parameters, see correlationHistogram(...). This version will have lower peak memory input, as eposIons can be chunked.

- The input histogram must be zero sized, or have the correct size when used. If zero sized, it will be resized and zeroed appropriately to contents
- The default here is that the lower triangular is true, unlike in `correlationHistogram(...)` so when performing the final accumulation, you may wish to set this to false, to ensure that `histogram_ij = histogram_ji`.
- Do not change `endMass` or `stepMass` between calls on the same histogram

Definition at line 119 of file `histogram.cpp`.

References `incrementCorrelationHist()`.

Referenced by `main()`.

Here is the call graph for this function:



5.1.3.2 `andersonDarlingStatistic()`

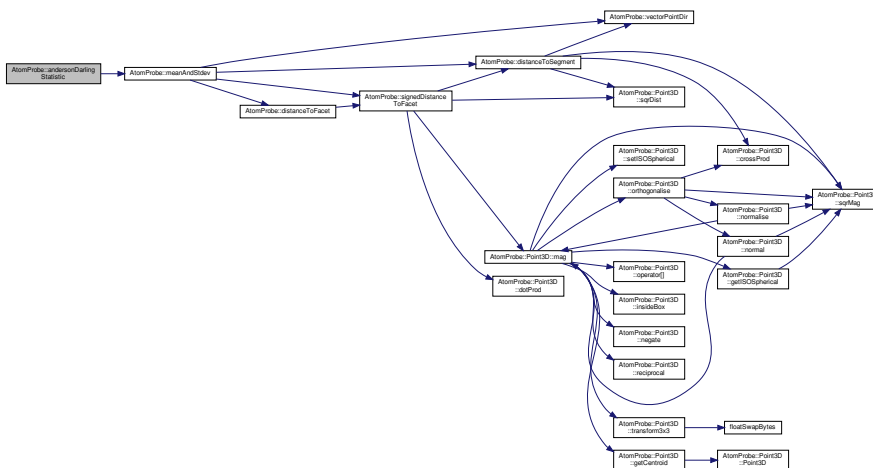
```
template<class T >
bool AtomProbe::andersonDarlingStatistic (
    std::vector< T > vals,
    float & meanV,
    float & stdevVal,
    float & statistic,
    size_t & undefCount,
    bool computeMeanAndStdev = true )
```

Definition at line 35 of file `processing.cpp`.

References `meanAndStdev()`.

Referenced by `doFitBackground()`.

Here is the call graph for this function:



5.1.3.3 antiRotateMatch()

```
bool AtomProbe::antiRotateMatch (
    const unsigned int * a,
    const unsigned int * b,
    size_t n )
```

Definition at line 63 of file mesh.cpp.

Referenced by AtomProbe::TRIANGLE::edgesMismatch().

5.1.3.4 applyQuaternionRotation()

```
void AtomProbe::applyQuaternionRotation (
    std::vector< Point3D > & pts,
    const Quaternion & q )
```

Apply the given quaternion rotation to the input points.

Rotation is in-place

5.1.3.5 askAssert()

```
void AtomProbe::askAssert (
    const char * filename,
    unsigned int lineNumber )
```

Either abort program, or ask the user what to do for an assertion. depending on hardAssert setting.

Definition at line 35 of file aptAssert.cpp.

5.1.3.6 buildFrequencyTable()

```
void AtomProbe::buildFrequencyTable (
    const vector< ISOTOPE_ENTRY > & solutionVec,
    vector< size_t > & solutionElements,
    vector< size_t > & solutionFrequency )
```

Definition at line 35 of file filter.cpp.

References ASSERT.

Referenced by filterPeakNeedBiggerObs(), and maxExplainedFraction().

5.1.3.7 castVoxels()

```
template<class T , class U >
void AtomProbe::castVoxels (
    const Voxels< T > & src,
    Voxels< U > & dest )
```

Convert one type of voxel into another by assignment operator.

Definition at line 383 of file voxels.h.

5.1.3.8 checkMassRangingCorrectness()

```
void AtomProbe::checkMassRangingCorrectness (
    const RangeFile & rng,
    AbundanceData & massTable,
    float massTolerance,
    unsigned int maxChargeState,
    unsigned int maxComponents,
    std::vector< bool > & badRanges )
```

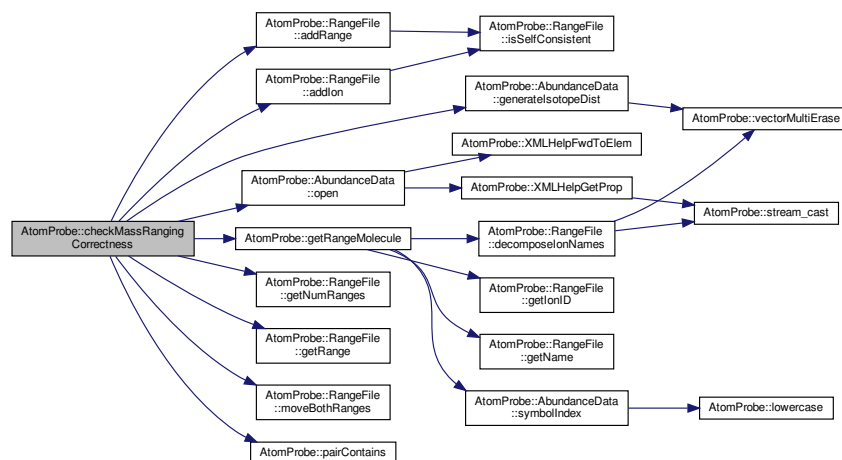
Ensure that each mass given spans a peak that should exist.

Each entry in badRanges is true if an inconsistent range is detected. There will be one entry per range in the parent rangefile Some ranges may be skipped if they cannot be understood maxComponents will skip checks for ions with more than this many atoms in the ionic molecule. Eg H2 would not be checked if maxComponents=1

Definition at line 101 of file rangeCheck.cpp.

References AtomProbe::RangeFile::addIon(), AtomProbe::RangeFile::addRange(), AtomProbe::RGBf::blue, AtomProbe::RANGE_MOLECULE::components, AtomProbe::AbundanceData::generateIsotopeDist(), AtomProbe::RangeFile::getNumRanges(), AtomProbe::RangeFile::getRange(), getRangeMolecule(), AtomProbe::RGBf::green, AtomProbe::RANGE_MOLECULE::isOK, AtomProbe::RangeFile::moveBothRanges(), AtomProbe::AbundanceData::open(), pairContains(), AtomProbe::RGBf::red, and TEST.

Here is the call graph for this function:



5.1.3.9 chunkLoadEposFile()

```
size_t AtomProbe::chunkLoadEposFile (
    std::vector< EPOS_ENTRY > & outData,
    const char * filename,
    unsigned int chunkSize,
    unsigned int chunkOffset,
    unsigned int & nEntriesLeft )
```

Load an "EPOS" file, with a maximum chunk size.

This is useful to avoid out-of-memory situations when working with large datasets. If the file has fewer EPOS entries left than the chunk size, only the remaining entries will be read. Increment chunkOffset to seek to the next part of the file. The file will be re-opened each time, so larger chunk sizes will increase speed at the cost of memory.

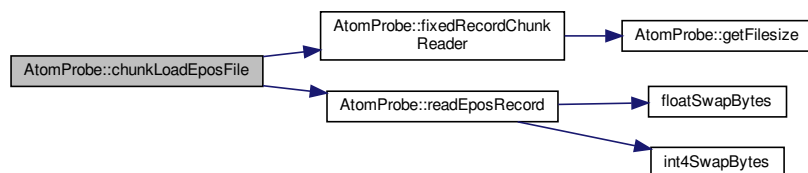
outData : the output to write the epos data to
 filename : the name of the EPOS formatted file to open
 chunkSize : the maximum number of entries to read in one go
 chunkOffset : the position (in chunks) of the file to start reading from, the caller must increment this to prevent the same section being re-read
 nEntriesLeft : returns the number of entries left in the file, The return value is nonzero on error, and is part of the RECORDREAD_ERR enum

Definition at line 757 of file dataFiles.cpp.

References EPOS_RECORD_SIZE, fixedRecordChunkReader(), and readEposRecord().

Referenced by getAtoErrString(), and main().

Here is the call graph for this function:



5.1.3.10 computeComposition()

```
void AtomProbe::computeComposition (
    const std::vector< unsigned int > & countData,
    std::vector< float > & compositionData )
```

Definition at line 13 of file composition.cpp.

Referenced by main().

5.1.3.11 computeConvexHull()

```
unsigned int AtomProbe::computeConvexHull (
    const std::vector< IonHit > & data,
    unsigned int * progress,
    bool(*) (bool) callback,
    std::vector< Point3D > & curHull,
    bool freeHull )
```

Obtain the convex hull of a set of ions.

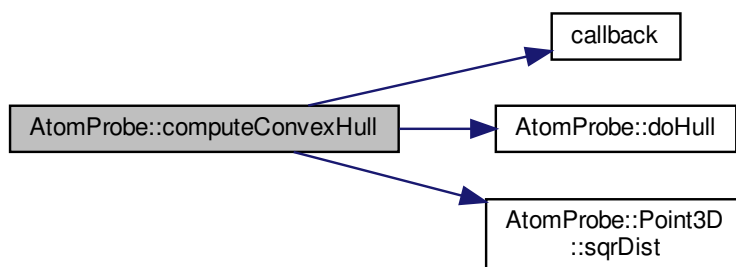
An existing set of points on the convex hull, "curHull", can be given as a starting set, optionally. For the last call to this routine, set "freeHull" to true. Returns 0 on success, nonzero on error, with error codes in above enum

Definition at line 149 of file convexHull.cpp.

References callback(), doHull(), HULL_ERR_NO_MEM, HULL_ERR_USER_ABORT, PROGRESS_REDUCE, and AtomProbe::Point3D::sqrDist().

Referenced by GetReducedHullPts().

Here is the call graph for this function:



5.1.3.12 computeIonDistAdjacency()

```
void AtomProbe::computeIonDistAdjacency (
    const MultiRange & mrf,
    const AbundanceData & abundance,
    const OVERLAP_PROBLEM_SETTINGS & settings,
    gsl_matrix *& m,
    gsl_vector *& vOwnership,
    gsl_vector *& vMass )
```

Calculate the adjacency matrix for the ions in a multirange.

Uses tolerance and other data in settings. Range data is NOT used. Vectors and matrices should not be allocated, but must be freed using `gsl_matrix_free` and `gsl_vector_free` after use. Return values are a 0/1 adjacency matrix,

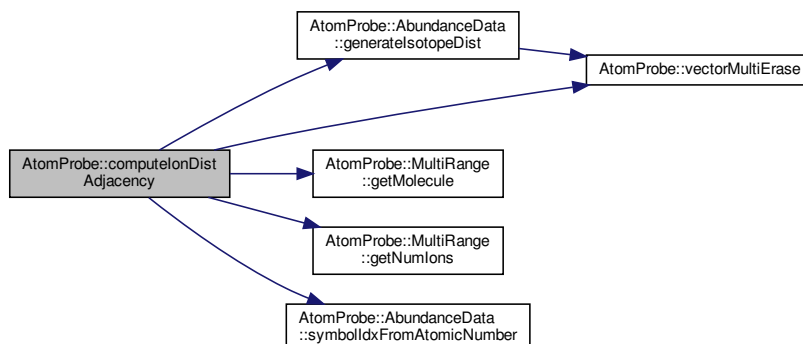
and two vectors, one describing the owner index of each ion and the other the mass. return matrix will be square, and vectors will have same size as matrix

Definition at line 40 of file componentAnalysis.cpp.

References ASSERT, AtomProbe::AbundanceData::generateIsotopeDist(), AtomProbe::MultiRange::getMolecule(), AtomProbe::MultiRange::getNumIons(), AtomProbe::OVERLAP_PROBLEM_SETTINGS::intensityTolerance, AtomProbe::OVERLAP_PROBLEM_SETTINGS::massTolerance, AtomProbe::OVERLAP_PROBLEM_SETTINGS::maxDefaultCharge, AtomProbe::AbundanceData::symbolIdxFromAtomicNumber(), and vectorMultiErase().

Referenced by computeRangeAdjacency().

Here is the call graph for this function:



5.1.3.13 computeRangeAdjacency()

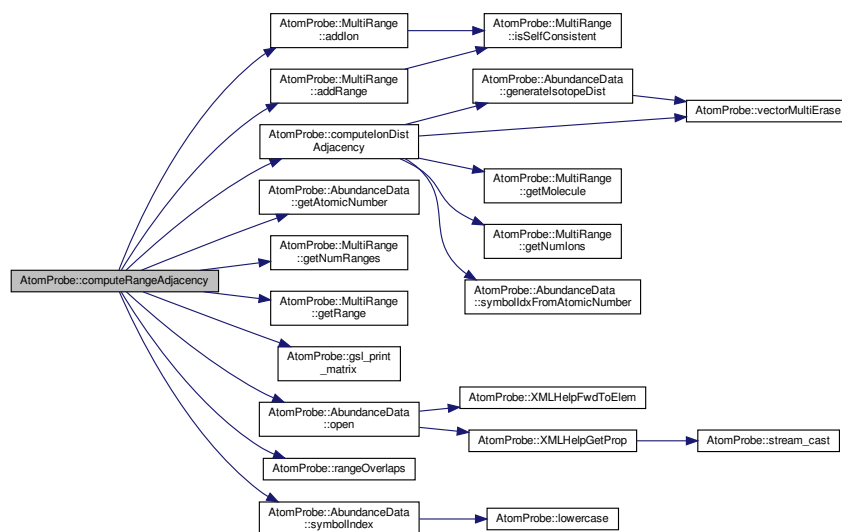
```

void AtomProbe::computeRangeAdjacency (
    const MultiRange & mrf,
    const OVERLAP_PROBLEM_SETTINGS & settings,
    gsl_matrix *& m )
  
```

Definition at line 172 of file componentAnalysis.cpp.

References AtomProbe::MultiRange::addIon(), AtomProbe::MultiRange::addRange(), ASSERT, AtomProbe::SIMPLE_SPECIES::atomicNumber, AtomProbe::RGBf::blue, computeIonDistAdjacency(), AtomProbe::SIMPLE_SPECIES::count, AtomProbe::AbundanceData::getAtomicNumber(), AtomProbe::MultiRange::getNumRanges(), AtomProbe::MultiRange::getRange(), AtomProbe::RGBf::green, gsl_print_matrix(), AtomProbe::OVERLAP_PROBLEM_SETTINGS::massTolerance, AtomProbe::OVERLAP_PROBLEM_SETTINGS::maxDefaultCharge, AtomProbe::AbundanceData::open(), rangeOverlaps(), AtomProbe::RGBf::red, AtomProbe::AbundanceData::symbolIndex(), TEST, and WARN.

Here is the call graph for this function:



5.1.3.14 computeRotationMatrix()

```

void AtomProbe::computeRotationMatrix (
    const Point3D & ur1,
    const Point3D & ur2,
    const Point3D & r1,
    const Point3D & r2,
    gsl_matrix * m )
  
```

Given two orthogonal and normal basis vectors, r1 and r2, and two matching rotated vectors ur1 and ur2, compute the rotation matrix that will transform between them.

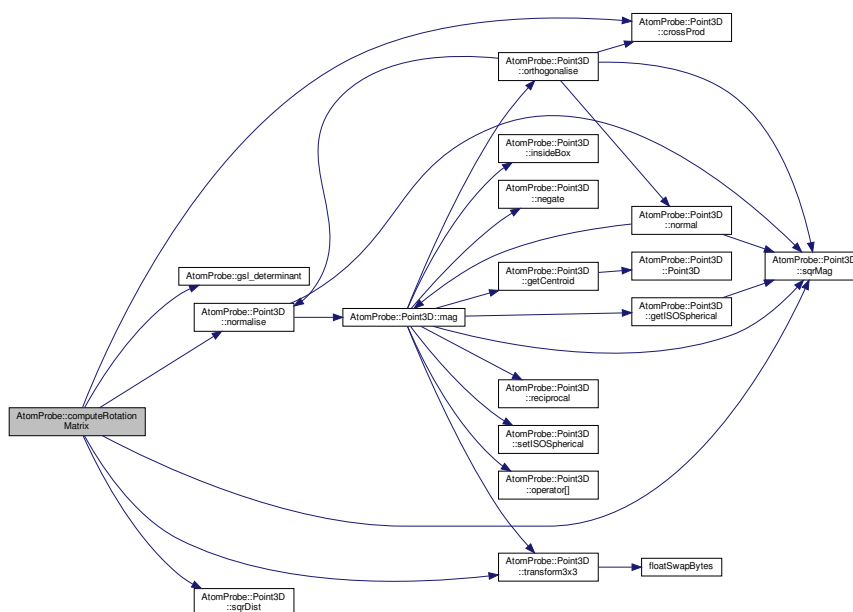
i.e. $r = M * ur$, vector pairs (ur1,ur2) and (r1,r2) should be orthogonal and normalised matrix m must be pre-allocated as a 3x3 matrix, using e.g `gsl_matrix_alloc(3,3)`

Definition at line 12 of file rotations.cpp.

References ASSERT, AtomProbe::Point3D::crossProd(), gsl_determinant(), AtomProbe::Point3D::normalise(), AtomProbe::Point3D::sqrDist(), AtomProbe::Point3D::sqrMag(), TEST, and AtomProbe::Point3D::transform3x3().

Referenced by main().

Here is the call graph for this function:



5.1.3.15 convertEPOStoPos()

```

void AtomProbe::convertEPOStoPos (
    const std::vector< EPOS_ENTRY > & eposEntry,
    std::vector< IonHit > & posFile )
  
```

Convert an incoming entry of EPOS files to pos, and the append this to the pos vector given.

Definition at line 29 of file convert.cpp.

5.1.3.16 convertFileStringToCanonical()

```

std::string AtomProbe::convertFileStringToCanonical (
    const std::string & s )
  
```

Convert a path format into a unix path from native format.

Referenced by strAppend().

5.1.3.17 convertFileStringToNative()

```
std::string AtomProbe::convertFileStringToNative (
    const std::string & s )
```

Convert a path format into a native path from unix format.

Referenced by strAppend().

5.1.3.18 convertMassToMol()

```
void AtomProbe::convertMassToMol (
    const AbundanceData & atomTable,
    const std::map< unsigned int, double > & compositionMass,
    std::map< unsigned int, double > & compositionMols )
```

Convert the given composition from mass fraction data to molar fraction.

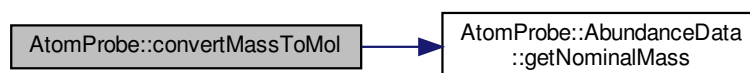
inputs are: compositionMass - relative mass fractions (this will be normalised automatically) massValues - weights of the masses input compositionMols - output results

Definition at line 74 of file massconvert.cpp.

References AtomProbe::AbundanceData::getNominalMass().

Referenced by parseCompositionData().

Here is the call graph for this function:



5.1.3.19 convertMolToMass()

```
void AtomProbe::convertMolToMass (
    const AbundanceData & atomTable,
    const std::map< unsigned int, double > & compositionMols,
    std::map< unsigned int, double > & compositionMass )
```

Convert the given composition from molar fraction data to mass fraction.

Inputs are:

- atomTable - the natural abundance table, to provide weights
- compositionMols : relative molar fractions (this will be normalised automatically). First entry in map is ion index in abundance table. Second is relative fraction.

Outputs are:

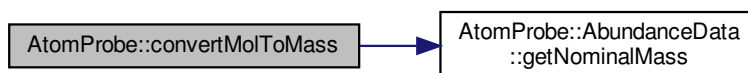
- compositionMass : output results, as mass fractions First entry is the position in the abundance table, second is relative fraction

Definition at line 41 of file massconvert.cpp.

References ASSERT, and AtomProbe::AbundanceData::getNominalMass().

Referenced by parseCompositionData().

Here is the call graph for this function:



5.1.3.20 correlationHistogram()

```
bool AtomProbe::correlationHistogram (
    const std::vector< EPOS_ENTRY > & eposIons,
    std::vector< std::vector< unsigned int > > & histogram,
    float stepMass,
    float endMass,
    bool lowerTriangular = false )
```

Generates an ion "correlation histogram" from a vector of EPOS ions. The input vector *MUST*

The nature of the plot is described in the following paper (Saxey et al, 2011): DOI : 10.1016/j.ultramic.2010.11.021, see e.g, Figure 3a.

The histogram will be allocated for you - do not pre-allocate

As the plot is symmetric around the $m_1=m_2$ line, the `lowerTriangular` setting controls if mirroring of the calculation is done. Setting this to true will ensure that only the lower half of the data field is calculated (faster).

`StepMass > 0` `EndMass > 0`

in both cases, the histogram will use the full rectangular memory space (arrays will not be ragged)

Definition at line 85 of file `histogram.cpp`.

References `incrementCorrelationHist()`.

Here is the call graph for this function:



5.1.3.21 `countDataDistanceWeight()`

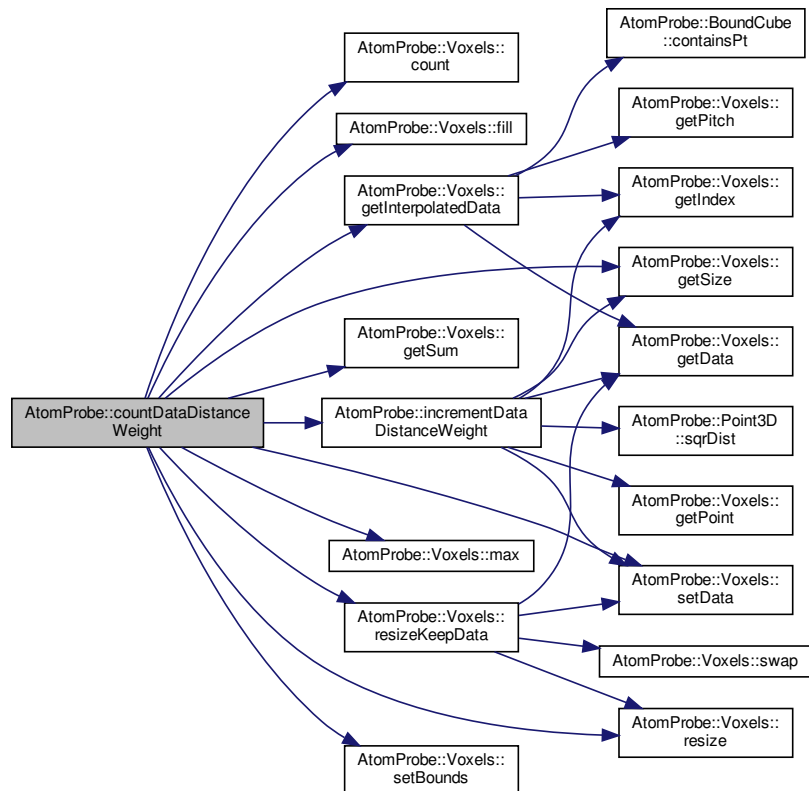
```

void AtomProbe::countDataDistanceWeight (
    const vector< Point3D > & pts,
    Voxels< float > & v )
  
```

Definition at line 120 of file `voxels.cpp`.

References `ASSERT`, `AtomProbe::Voxels< T >::count()`, `AtomProbe::Voxels< T >::fill()`, `AtomProbe::Voxels< T >::getInterpolatedData()`, `AtomProbe::Voxels< T >::getSize()`, `AtomProbe::Voxels< T >::getSum()`, `incrementDataDistanceWeight()`, `AtomProbe::Voxels< T >::max()`, `AtomProbe::Voxels< T >::resize()`, `AtomProbe::Voxels< T >::resizeKeepData()`, `AtomProbe::Voxels< T >::setBounds()`, `AtomProbe::Voxels< T >::setData()`, and `TEST`.

Here is the call graph for this function:



5.1.3.22 countIntensityEvents()

```

unsigned int AtomProbe::countIntensityEvents (
    const vector< pair< float, float > > & data,
    float minV,
    float maxV )

```

Definition at line 21 of file filter.cpp.

Referenced by maxExplainedFraction().

5.1.3.23 createMassBackground()

```

void AtomProbe::createMassBackground (
    float massStart,
    float massEnd,
    unsigned int nBinsMass,
    float tofBackIntensity,
    std::vector< float > & histogram )

```

Build a histogram of the background counts.

- Inputs : Start and end mass for background window, and bin count
- `tofBackIntensity` is the intensity level per unit time (proportional to sqrt mass) in the background, as obtained by `doFitBackground`.

Note : You may need to divide by bin width, depending on your application

Definition at line 208 of file `processing.cpp`.

Referenced by `findPeaks()`.

5.1.3.24 `cumTrapezoid()`

```
template<class T >
void AtomProbe::cumTrapezoid (
    const vector< T > & x,
    const vector< T > & vals,
    vector< T > & res )
```

Definition at line 449 of file `confidence.cpp`.

References `ASSERT`, and `kahansum()`.

Here is the call graph for this function:



5.1.3.25 `det3by3()`

```
double AtomProbe::det3by3 (
    const double * ptArray )
```

Definition at line 205 of file `misc.cpp`.

Referenced by `dotProduct()`, and `pyramidVol()`.

5.1.3.26 Determinant()

```
float AtomProbe::Determinant (
    float ** a,
    int n )
```

Definition at line 172 of file mesh.cpp.

References ASSERT, and signVal().

Referenced by fourDeterminant().

Here is the call graph for this function:



5.1.3.27 diff()

```
void AtomProbe::diff (
    const vector< float > & in,
    vector< float > & out )
```

Definition at line 233 of file processing.cpp.

Referenced by findPeaks().

5.1.3.28 digitString()

```
std::string AtomProbe::digitString (
    unsigned int thisDigit,
    unsigned int maxDigit )
```

Generate a string with leading digits up to maxDigit (eg, if maxDigit is 424, and thisDigit is 1.

Definition at line 115 of file stringFuncs.cpp.

References stream_cast().

Referenced by strAppend().

Here is the call graph for this function:



5.1.3.29 distanceToFacet()

```

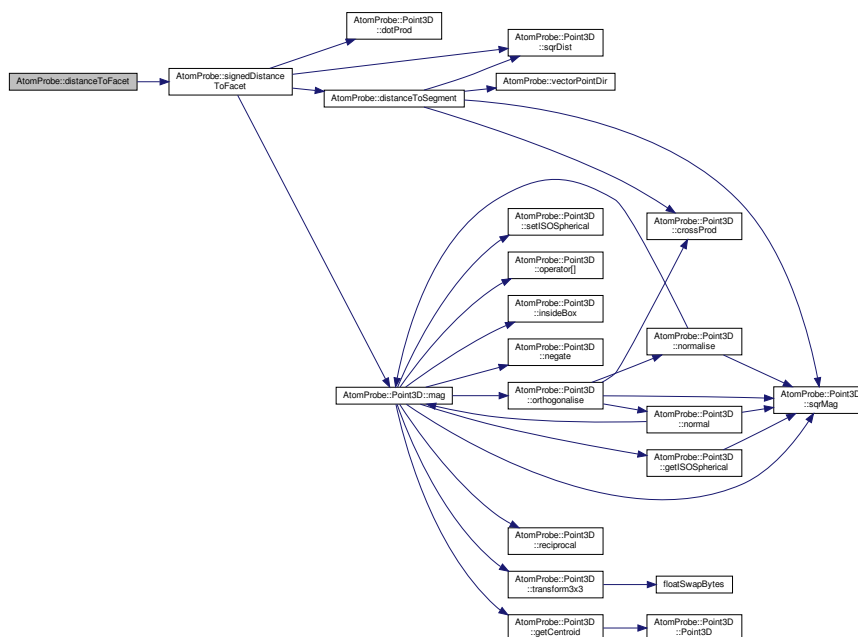
float AtomProbe::distanceToFacet (
    const Point3D & fA,
    const Point3D & fB,
    const Point3D & fC,
    const Point3D & normal,
    const Point3D & p )
  
```

Definition at line 199 of file misc.cpp.

References signedDistanceToFacet().

Referenced by GetReducedHullPts(), and meanAndStdev().

Here is the call graph for this function:



5.1.3.30 distanceToSegment()

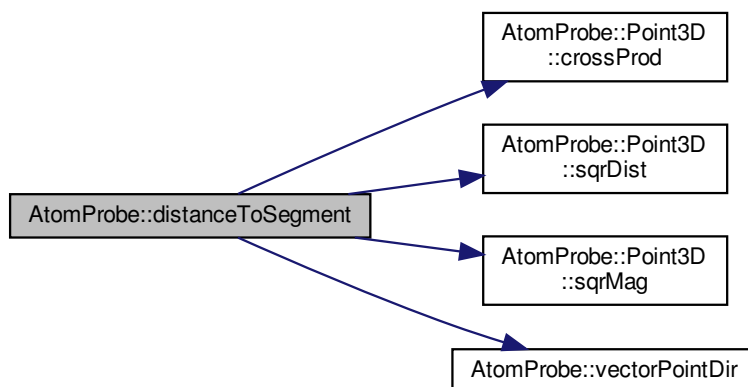
```
float AtomProbe::distanceToSegment (
    const Point3D & fA,
    const Point3D & fB,
    const Point3D & p )
```

Definition at line 105 of file misc.cpp.

References AtomProbe::Point3D::crossProd(), POINTDIR_TOGETHER, AtomProbe::Point3D::sqrDist(), AtomProbe::Point3D::sqrMag(), and vectorPointDir().

Referenced by meanAndStdev(), and signedDistanceToFacet().

Here is the call graph for this function:



5.1.3.31 doFitBackground()

```
unsigned int AtomProbe::doFitBackground (
    const std::vector< float > & massData,
    BACKGROUND_PARAMS & params )
```

Perform a background fit, assuming constant TOF noise, from 0->inf time.

- Background params has the input and output data for the fit.

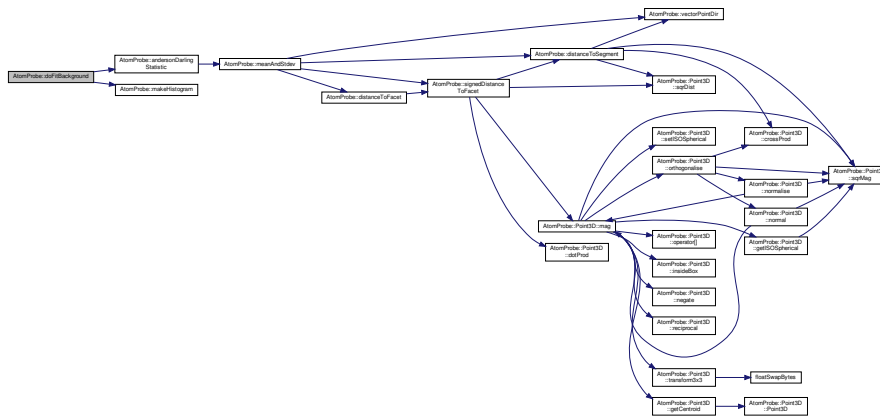
dataIn requires raw mass data, not histogram counts.

- returns zero on success, nonzero on error Return value is stored in params.intensity, as counts/bin. Use createMassBackground to re-generate the histogram

Definition at line 144 of file processing.cpp.

References `andersonDarlingStatistic()`, `ASSERT`, `AtomProbe::BACKGROUND_PARAMS::binWidth`, `AtomProbe::BACKGROUND_PARAMS::FIT_FAIL_AVG_COUNTS`, `AtomProbe::BACKGROUND_PARAMS::FIT_FAIL_DATA_NON_GAUSSIAN`, `AtomProbe::BACKGROUND_PARAMS::FIT_FAIL_INSUFF_DATA`, `AtomProbe::BACKGROUND_PARAMS::FIT_FAIL_MIN_REQ_BINS`, `AtomProbe::BACKGROUND_PARAMS::intensity`, `makeHistogram()`, `AtomProbe::BACKGROUND_PARAMS::massEnd`, `AtomProbe::BACKGROUND_PARAMS::massStart`, and `AtomProbe::BACKGROUND_PARAMS::stdev`.

Here is the call graph for this function:



5.1.3.32 doHull()

```
unsigned int AtomProbe::doHull (
    unsigned int bufferSize,
    double * buffer,
    vector< Point3D > & resHull,
    Point3D & midPoint,
    bool freeHullOnExit )
```

Definition at line 53 of file convexHull.cpp.

References `HULL_ERR_NO_MEM`.

Referenced by `computeConvexHull()`.

5.1.3.33 dotProduct()

```
float AtomProbe::dotProduct (
    float a1,
    float a2,
    float a3,
    float b1,
    float b2,
    float b3 ) [inline]
```

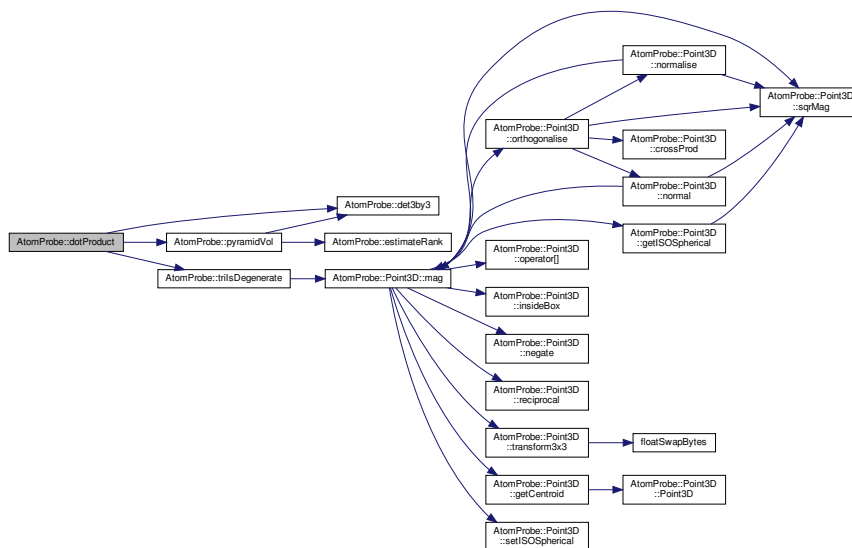
Inline func for calculating $a \cdot b$.

Definition at line 107 of file misc.h.

References `det3by3()`, `pyramidVol()`, and `trilsDegenerate()`.

Referenced by `GetReducedHullPts()`.

Here is the call graph for this function:



5.1.3.34 edgIdx()

```
unsigned int AtomProbe::edgeIdx (
    unsigned int i,
    unsigned int j )
```

Definition at line 249 of file mesh.cpp.

References `ASSERT`.

5.1.3.35 estimateRank()

```
unsigned int AtomProbe::estimateRank (
    const gsl_matrix * m,
    float tolerance = sqrt(std::numeric_limits<float>::epsilon()) )
```

Estimate the rank of the given matrix.

Definition at line 7 of file misc.cpp.

Referenced by leastSquaresDeconvolve(), and pyramidVol().

5.1.3.36 filterBySolutionPPM()

```
void AtomProbe::filterBySolutionPPM (
    const AbundanceData & massTable,
    float minPpm,
    std::vector< std::vector< ISOTOPE_ENTRY > > & solutions )
```

Use the maximum possible PPM for each isotopic combination to filter possible solutions.

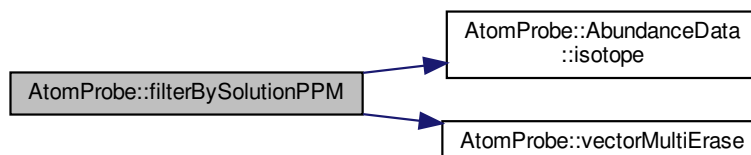
For example, $^{17}\text{O}:^{17}\text{O}:^{17}\text{O}$ in a naturally abundant mix has a maximum possible concentration of $5\text{e-}11$ at. fraction. Isotope group tolerance is used to group the abundance distribution of nearby isotopes

massTable - a table of natural abundances minPpm - Minimum PPM below which to discard solutions solutions - list of solutions, which will be culled.

Definition at line 68 of file filter.cpp.

References AtomProbe::ISOTOPE_ENTRY::abundance, AtomProbe::AbundanceData::isotope(), and vectorMultiErase().

Here is the call graph for this function:



5.1.3.37 filterPeakNeedBiggerObs()

```
void AtomProbe::filterPeakNeedBiggerObs (
    const AbundanceData & massTable,
    const std::vector< float > & peakData,
    float tolerance,
    size_t solutionCharge,
    std::vector< std::vector< ISOTOPE_ENTRY > > & solutions )
```

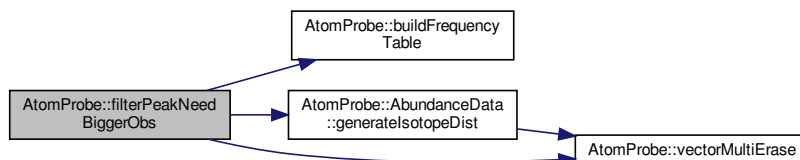
Remove peaks for which an experimental set of observed peaks do not show a peak at a position which is expected from the peak distribution for this element. Only works with single-atom peaks

- Solutions are the input solutions,
- peakData is a list of peak mass positions
- tolerance is the error allowed to fit a peak (+/-)
- solutionCharge - charge state for the input isotope entries

Definition at line 104 of file filter.cpp.

References ASSERT, buildFrequencyTable(), AtomProbe::AbundanceData::generateIsotopeDist(), and vector::MultiErase().

Here is the call graph for this function:



5.1.3.38 findMaxLessThanOrEq()

```
size_t AtomProbe::findMaxLessThanOrEq (
    const vector< std::pair< size_t, size_t > > & v,
    size_t value )
```

Definition at line 155 of file mesh.cpp.

References ASSERT.

5.1.3.39 findNearVerticies() [1/2]

```
void AtomProbe::findNearVerticies (
    float tolerance,
    const vector< Point3D > & ptVec,
    vector< std::pair< size_t, vector< size_t > > > & clusterList )
```

Definition at line 365 of file mesh.cpp.

References ASSERT.

Referenced by AtomProbe::Mesh::mergeDuplicateVertices(), and AtomProbe::Mesh::setTriangleMesh().

5.1.3.40 findNearVerticies() [2/2]

```
void AtomProbe::findNearVerticies (
    float tolerance,
    const vector< Point3D > & ptVec,
    std::list< std::pair< size_t, vector< size_t > > > & clusterList )
```

Definition at line 404 of file mesh.cpp.

References ASSERT.

5.1.3.41 findOverlaps() [1/4]

```
void AtomProbe::findOverlaps (
    const AbundanceData & natData,
    const RangeFile & rng,
    float massDelta,
    unsigned int maxCharge,
    std::vector< std::pair< size_t, size_t > > & overlapIdx,
    std::vector< std::pair< float, float > > & overlapMasses )
```

Find the overlaps stemming from a given rangefile.

An overlap is defined as "natural abundances within +-massDelta at same mass to charge"

- Species not in the rangefile will not be marked as overlapping
- FIXME: The output index is abused at this time, the range's index is actually given by dividing the output by the mass to charge. The charge state is given by modulo (massToCharge) +1
- The range mass information in the rangefile is not used - it is abused as an ion list

Inputs are:

- rng: The rangefile to use. The species in the rangefile are used to compute the overlaps
- massDelta: tolerance in amu,
- maxCharge: the maximum charge that species can be estimated to have

Outputs are:

- overlapIdx: the range indices of the overlapping species (ion index in rangefile)
- overlapMasses: the masses of the overlapping species, which should be within +-massDelta of one another

Note that ion's names will be decomposed, so Fe2H will be considered an Fe-Fe-H molecule

Referenced by findOverlaps(), and main().

5.1.3.42 findOverlaps() [2/4]

```
void AtomProbe::findOverlaps (
    const AbundanceData & natData,
    const std::vector< std::string > & ions,
    float massDelta,
    unsigned int maxCharge,
    std::vector< std::pair< size_t, size_t > > & overlapIdx,
    std::vector< std::pair< float, float > > & overlapMasses )
```

As per findOverlaps(...Rangefile ...) , but uses a vector of ions instead.

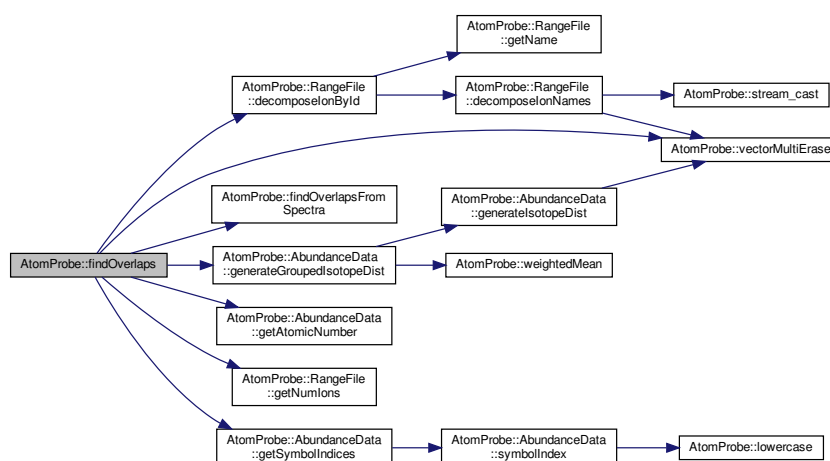
5.1.3.43 findOverlaps() [3/4]

```
void AtomProbe::findOverlaps (
    const AbundanceData & natData,
    const RangeFile & rng,
    float massDelta,
    unsigned int maxCharge,
    vector< pair< size_t, size_t > > & overlapIdx,
    vector< pair< float, float > > & overlapMasses )
```

Definition at line 78 of file overlaps.cpp.

References ASSERT, AtomProbe::RangeFile::decomposeIonById(), findOverlapsFromSpectra(), AtomProbe::AbundanceData::generateGroupedIsotopeDist(), AtomProbe::AbundanceData::getAtomicNumber(), AtomProbe::RangeFile::getNumIons(), AtomProbe::AbundanceData::getSymbolIndices(), and vectorMultiErase().

Here is the call graph for this function:



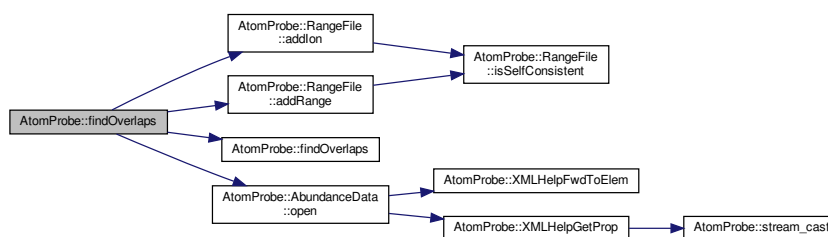
5.1.3.44 findOverlaps() [4/4]

```
void AtomProbe::findOverlaps (
    const AbundanceData & natData,
    const vector< string > & ionNames,
    float massDelta,
    unsigned int maxCharge,
    vector< pair< size_t, size_t > > & overlapIdx,
    vector< pair< float, float > > & overlapMasses )
```

Definition at line 202 of file overlaps.cpp.

References AtomProbe::RangeFile::addIon(), AtomProbe::RangeFile::addRange(), AtomProbe::RGBf::blue, findOverlaps(), AtomProbe::RGBf::green, AtomProbe::AbundanceData::open(), AtomProbe::RGBf::red, and TEST.

Here is the call graph for this function:



5.1.3.45 findOverlapsFromSpectra() [1/2]

```
void AtomProbe::findOverlapsFromSpectra (
    const vector< vector< pair< float, float > > > & massDistributions,
    float massDelta,
    const set< unsigned int > & skipIDs,
    vector< pair< size_t, size_t > > & overlapIdx,
    vector< pair< float, float > > & overlapMasses )
```

Definition at line 30 of file overlaps.cpp.

References EQ_TOLV.

5.1.3.46 findOverlapsFromSpectra() [2/2]

```
void AtomProbe::findOverlapsFromSpectra (
    const std::vector< std::vector< std::pair< float, float > > > & massDistributions,
    float massDelta,
    const std::set< unsigned int > & skipIDs,
    std::vector< std::pair< size_t, size_t > > & overlapIdx,
    std::vector< std::pair< float, float > > & overlapMasses )
```

Find overlaps in the given mass distributions, as per findOverlaps(...Rangefile...)

Referenced by findOverlaps().

5.1.3.47 findPeaks()

```
void AtomProbe::findPeaks (
    const std::vector< float > & x0,
    std::vector< unsigned int > & peakInds,
    float sel,
    bool autoSel = true,
    bool includeEndpoints = true )
```

Simple peak-finding algorithm.

Adapted from Nathaniel Yoder's: <https://uk.mathworks.com/matlabcentral/fileexchange/25500-peakfind> which is BSD licenced.

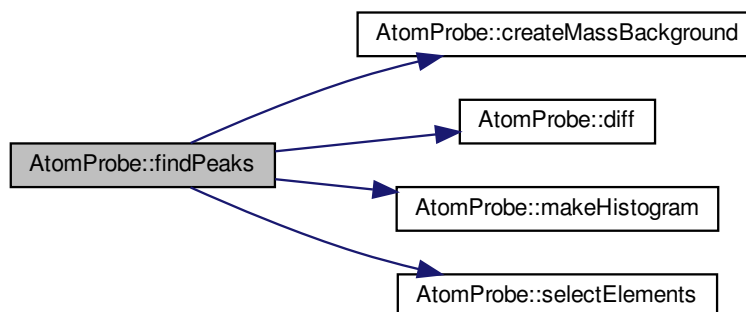
The performance of this function may or may not be suitable for your application, and your histogram may require pre-processing to optimise false positive/false-negative performance

- x0 : input mass histogram, must be > 2 elements
- sel : Selectivity for peaks - larger is more selective
- autosel : whether to determine sel automatically, or not
- includeEndpoints : allow edge values to be peaks

Definition at line 252 of file processing.cpp.

References ASSERT, createMassBackground(), diff(), makeHistogram(), NUM_IONS, and selectElements().

Here is the call graph for this function:



5.1.3.48 fixedRecordChunkReader()

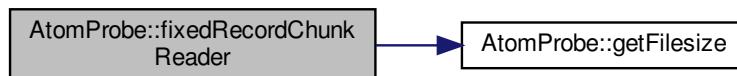
```
template<class T >
unsigned int AtomProbe::fixedRecordChunkReader (
    const char * filename,
    bool(*) (const char *bufRead, const char *destBuf) recordReader,
    size_t recordSize,
    std::vector< T > & outputData,
    unsigned int chunkSize,
    unsigned int chunkOffset,
    unsigned int & nEntriesLeft )
```

Definition at line 144 of file dataFiles.cpp.

References `getFileSize()`, `RECORDREAD_BAD_FILEREAD`, `RECORDREAD_BAD_RECORD`, `RECORDREAD_ERR_CHUNKOFFSET`, `RECORDREAD_ERR_FILE_OPEN`, `RECORDREAD_ERR_FILESIZE_MODULO`, `RECORDREAD_ERR_GET_FILESIZE`, and `RECORDREAD_ERR_NOMEM`.

Referenced by `chunkLoadEposFile()`.

Here is the call graph for this function:



5.1.3.49 fixedRecordReader()

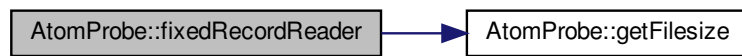
```
template<class T >
unsigned int AtomProbe::fixedRecordReader (
    const char * filename,
    bool(*) (const char *bufRead, const char *destBuf) recordReader,
    size_t recordSize,
    std::vector< T > & outputData )
```

Definition at line 93 of file dataFiles.cpp.

References `getFileSize()`, `RECORDREAD_BAD_FILEREAD`, `RECORDREAD_BAD_RECORD`, `RECORDREAD_ERR_FILE_OPEN`, `RECORDREAD_ERR_FILESIZE_MODULO`, `RECORDREAD_ERR_GET_FILESIZE`, and `RECORDREAD_ERR_NOMEM`.

Referenced by `loadEposFile()`.

Here is the call graph for this function:



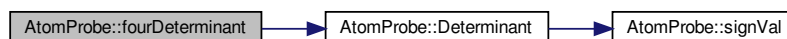
5.1.3.50 fourDeterminant()

```
float AtomProbe::fourDeterminant (
    const Point3D & a,
    const Point3D & b,
    const Point3D & c,
    const Point3D & d )
```

Definition at line 212 of file mesh.cpp.

References Determinant().

Here is the call graph for this function:



5.1.3.51 fpeek()

```
int AtomProbe::fpeek (
    FILE * stream ) [inline]
```

Definition at line 69 of file helpFuncs.h.

Referenced by `AtomProbe::RangeFile::rangeTypeString()`.

5.1.3.52 freeConvexHull()

```
void AtomProbe::freeConvexHull ( )
```

Definition at line 47 of file convexHull.cpp.

Referenced by GetReducedHullPts().

5.1.3.53 gauss_ratio_pdf()

```
double AtomProbe::gauss_ratio_pdf (
    double x,
    double muX,
    double muY,
    double varX,
    double varY )
```

Definition at line 471 of file confidence.cpp.

References M_PI.

5.1.3.54 gcd()

```
int AtomProbe::gcd (
    int a,
    int b )
```

Definition at line 12 of file millerIndex.cpp.

Referenced by AtomProbe::MILLER_TRIPLET::simplify().

5.1.3.55 genColString() [1/2]

```
void AtomProbe::genColString (
    unsigned char r,
    unsigned char g,
    unsigned char b,
    unsigned char a,
    std::string & s )
```

Referenced by strAppend(), AtomProbe::RGBf::toHex(), and AtomProbe::RangeFile::write().

5.1.3.56 genColString() [2/2]

```
void AtomProbe::genColString (
    unsigned char r,
    unsigned char g,
    unsigned char b,
    std::string & s )
```

Definition at line 308 of file stringFuncs.cpp.

References ucharToHexStr().

Here is the call graph for this function:



5.1.3.57 generate1DAxialDistHist()

```
unsigned int AtomProbe::generate1DAxialDistHist (
    const std::vector< Point3D > & pointList,
    K3DTreeExact & tree,
    const Point3D & axisDir,
    float distMax,
    std::vector< unsigned int > & histogram )
```

Generate a 1D axial distribution function,.

This generates a histogram of point-point distances, from the points in the point list to the points in the tree, within a specified radius of the source point.

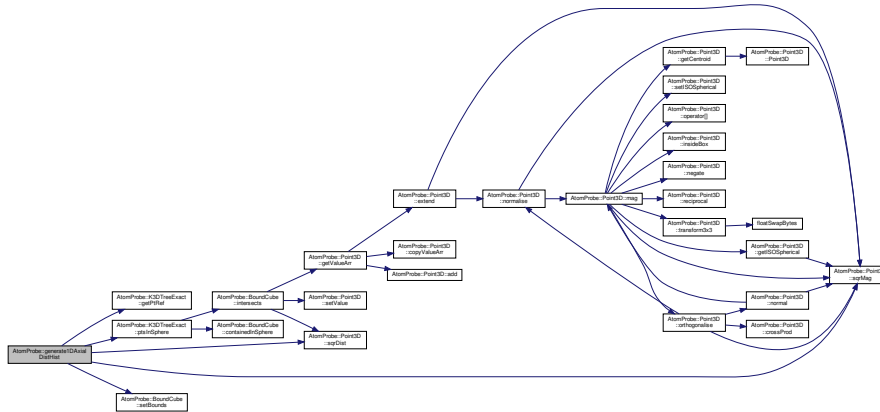
This is variously called a 1D "Spatial Distribution Map (SDM)" "Atom Vicinity" or similar

- pointList : the source points for the distribution function
- tree : a pre-generated KD tree with the target points
- distMax : radius of the points around which to search
- histogram : a vector with the number of bins that are required for output. Does not need to be zero filled

Definition at line 22 of file axialdf.cpp.

References ASSERT, AtomProbe::K3DTreeExact::getPtRef(), AtomProbe::K3DTreeExact::ptsInSphere(), AtomProbe::BoundCube::setBounds(), AtomProbe::Point3D::sqrDist(), and AtomProbe::Point3D::sqrMag().

Here is the call graph for this function:



5.1.3.58 generate1DAxialDistHistSweep()

```
unsigned int AtomProbe::generate1DAxialDistHistSweep (
    const std::vector< Point3D > & pointList,
    K3DTreeExact & tree,
    float distMax,
    float dTheta,
    float dPhi,
    ProgressBar & prog,
    std::vector< std::vector< std::vector< unsigned int > > > & histogram )
```

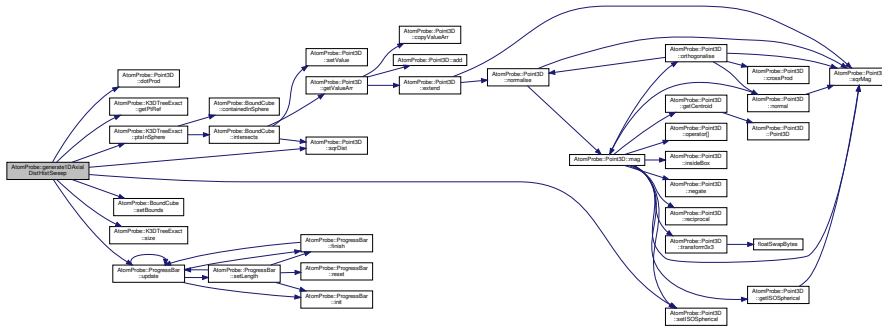
Generate a series of 1D distribution functions, one per pixel in a 2D grid of spherical coordinate directions.

Using a stepped spherical coordinates in equal delta-Theta, delta-Phi increments pointList : the source points from which to search tree : K3D Tree of target points distMax : maximum distance around which to search dTheta : step size in theta (radians). dPhi : step size in phi (radians) prog : Progress bar histogram : resultant data as a 2D vector, one bin per angular entry. Each vector will have the same size. The inner most vector contains the analysis data, over +-distMax

Definition at line 116 of file axialdf.cpp.

References ASSERT, AtomProbe::Point3D::dotProd(), AtomProbe::K3DTreeExact::getPtRef(), M_PI, AtomProbe::K3DTreeExact::ptsInSphere(), AtomProbe::BoundCube::setBounds(), AtomProbe::Point3D::setISOSpherical(), AtomProbe::K3DTreeExact::size(), AtomProbe::Point3D::sqrDist(), and AtomProbe::ProgressBar::update().

Here is the call graph for this function:



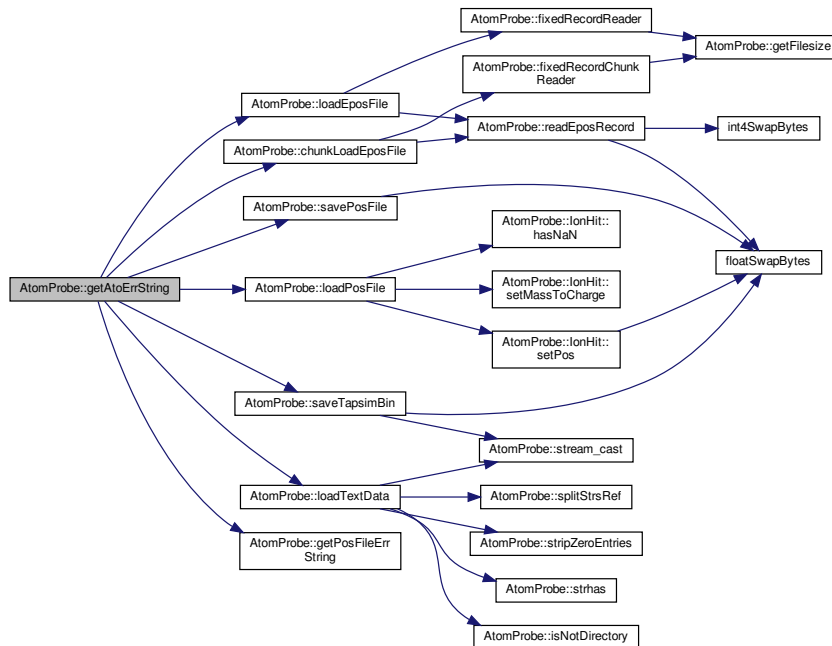
5.1.3.59 getAtoErrString()

```
const char* AtomProbe::getAtoErrString (
    unsigned int errCode ) [inline]
```

Definition at line 82 of file `dataFiles.h`.

References `chunkLoadEposFile()`, `getPosFileErrString()`, `loadEposFile()`, `loadPosFile()`, `loadTextData()`, `AtomProbe::ATO_ENTRY::mass`, `savePosFile()`, and `saveTapsimBin()`.

Here is the call graph for this function:



5.1.3.60 `getFileSize()`

```
bool AtomProbe::getFileSize (
    const char * fname,
    size_t & size )
```

Definition at line 107 of file `helpFuncs.cpp`.

Referenced by `fixedRecordChunkReader()`, `fixedRecordReader()`, `AtomProbe::RangeFile::openFormat()`, `AtomProbe::ComparePairFirstReverse::operator()`, and `readPosapOps()`.

5.1.3.61 `getFitErrorMsg()`

```
string AtomProbe::getFitErrorMsg (
    unsigned int errCode )
```

Obtain a human readable error, which has arisen when using `doFitBackground`. Input is the error code from `doFitBackground`.

Definition at line 130 of file `processing.cpp`.

References `ASSERT`, and `AtomProbe::BACKGROUND_PARAMS::FIT_FAIL_END`.

5.1.3.62 `getHardAssert()`

```
bool AtomProbe::getHardAssert ( )
```

Definition at line 25 of file `aptAssert.cpp`.

References `hardAssert`.

5.1.3.63 `getPosFileErrString()`

```
const char * AtomProbe::getPosFileErrString (
    unsigned int errMesg )
```

Definition at line 238 of file `dataFiles.cpp`.

References `ARRAYSIZE`, and `POS_FILE_ENUM_END`.

Referenced by `getAtoErrString()`, and `main()`.

5.1.3.64 `getRangeMolecule()`

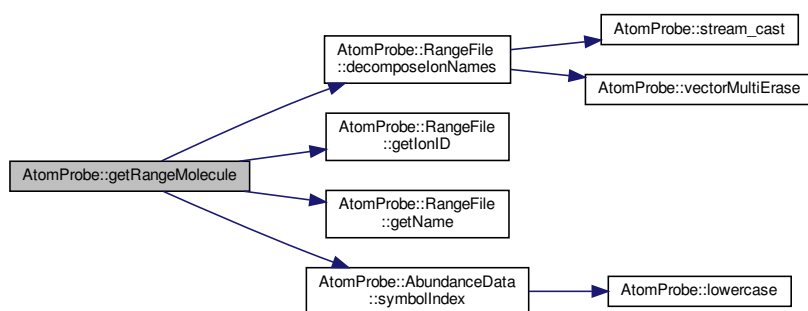
```
RANGE_MOLECULE AtomProbe::getRangeMolecule (
    const AbundanceData & massTable,
    const RangeFile & rng,
    unsigned int rangeId )
```

Definition at line 55 of file rangeCheck.cpp.

References `AtomProbe::RANGE_MOLECULE::components`, `AtomProbe::RangeFile::decomposeIonNames()`, `AtomProbe::RangeFile::getIonID()`, `AtomProbe::RangeFile::getName()`, `AtomProbe::RANGE_MOLECULE::isOK`, `AtomProbe::RANGE_MOLECULE::name`, and `AtomProbe::AbundanceData::symbolIndex()`.

Referenced by `checkMassRangingCorrectness()`.

Here is the call graph for this function:

5.1.3.65 `getRecordReadErrString()`

```
const char* AtomProbe::getRecordReadErrString (
    unsigned int errorCode ) [inline]
```

Definition at line 46 of file dataFiles.h.

5.1.3.66 `GetReducedHullPts()`

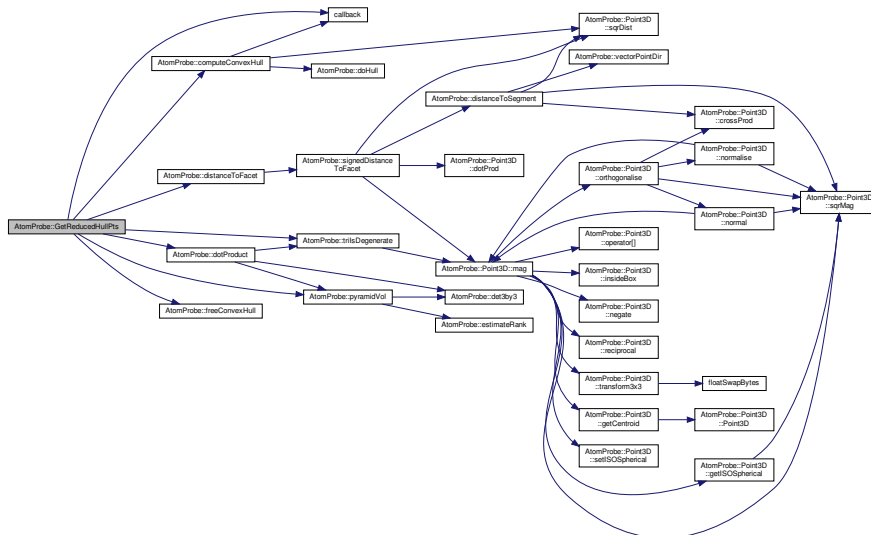
```
unsigned int AtomProbe::GetReducedHullPts (
    const std::vector< IonHit > & points,
    float reductionDim,
    unsigned int * progress,
    bool(*) (bool) callback,
    std::vector< IonHit > & pointResult )
```

Obtain a set of points that are a subset of points the convex hull that are at least "reductionDim" away from the hull itself.

Definition at line 289 of file convexHull.cpp.

References ASSERT, callback(), computeConvexHull(), distanceToFacet(), dotProduct(), freeConvexHull(), HULL ← _SURFREDUCE_NEGATIVE_SCALE_FACT, NUM_IONS, pyramidVol(), SCALE, and trisDegenerate().

Here is the call graph for this function:



5.1.3.67 gsl_determinant()

```
float AtomProbe::gsl_determinant (
    const gsl_matrix * m )
```

Definition at line 143 of file helpFuncs.cpp.

References ASSERT.

Referenced by computeRotationMatrix(), and AtomProbe::ComparePairFirstReverse::operator().

5.1.3.68 gsl_matrix_mult()

```
void AtomProbe::gsl_matrix_mult (
    const gsl_matrix * a,
    const gsl_matrix * b,
    gsl_matrix * & res )
```

Definition at line 40 of file point3D.cpp.

Referenced by main().

5.1.3.69 gsl_print_matrix()

```
void AtomProbe::gsl_print_matrix (
    const gsl_matrix * m )
```

Definition at line 122 of file helpFuncs.cpp.

Referenced by computeRangeAdjacency(), and AtomProbe::ComparePairFirstReverse::operator()().

5.1.3.70 gsl_print_vector()

```
void AtomProbe::gsl_print_vector (
    const gsl_vector * v )
```

Definition at line 134 of file helpFuncs.cpp.

Referenced by AtomProbe::ComparePairFirstReverse::operator()().

5.1.3.71 hexStrToUChar()

```
void AtomProbe::hexStrToUChar (
    const std::string & s,
    unsigned char & c )
```

Definition at line 92 of file stringFuncs.cpp.

References ASSERT.

Referenced by parseColString().

5.1.3.72 incrementCorrelationHist()

```
bool AtomProbe::incrementCorrelationHist (
    const std::vector< EPOS_ENTRY > & eposIons,
    std::vector< std::vector< unsigned int > > & histogram,
    float stepMass,
    float endMass )
```

Definition at line 29 of file histogram.cpp.

Referenced by accumulateCorrelationHistogram(), and correlationHistogram().

5.1.3.73 incrementDataDistanceWeight()

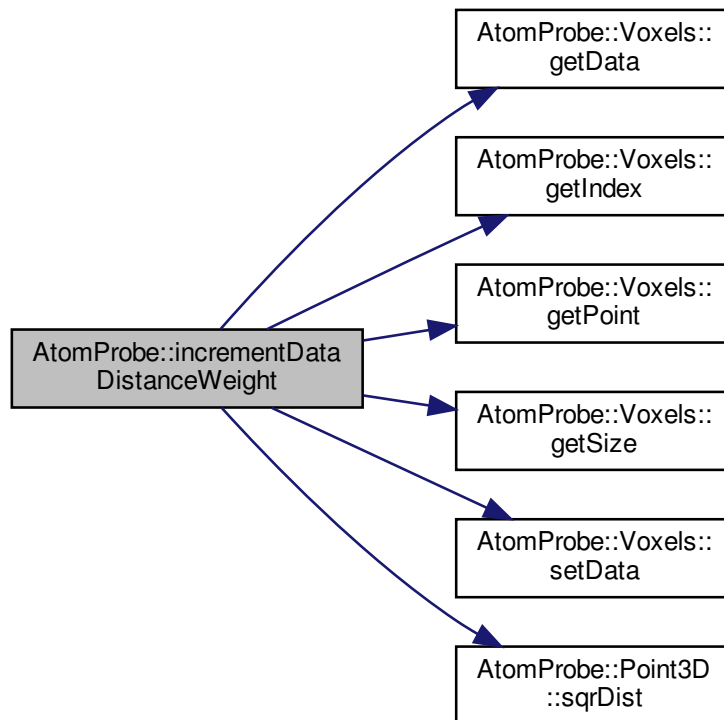
```
void AtomProbe::incrementDataDistanceWeight (
    const Point3D & p,
    Voxels< float > & v )
```

Definition at line 33 of file voxels.cpp.

References `AtomProbe::Voxels< T >::getData()`, `AtomProbe::Voxels< T >::getIndex()`, `AtomProbe::Voxels< T >::getPoint()`, `AtomProbe::Voxels< T >::getSize()`, `AtomProbe::Voxels< T >::setData()`, and `AtomProbe::Point3D::sqrDist()`.

Referenced by `countDataDistanceWeight()`.

Here is the call graph for this function:



5.1.3.74 intersect_RayTriangle()

```
int AtomProbe::intersect_RayTriangle (
    const Point3D & rayStart,
    const Point3D & rayEnd,
```



```

Point3D * tri,
Point3D & I )

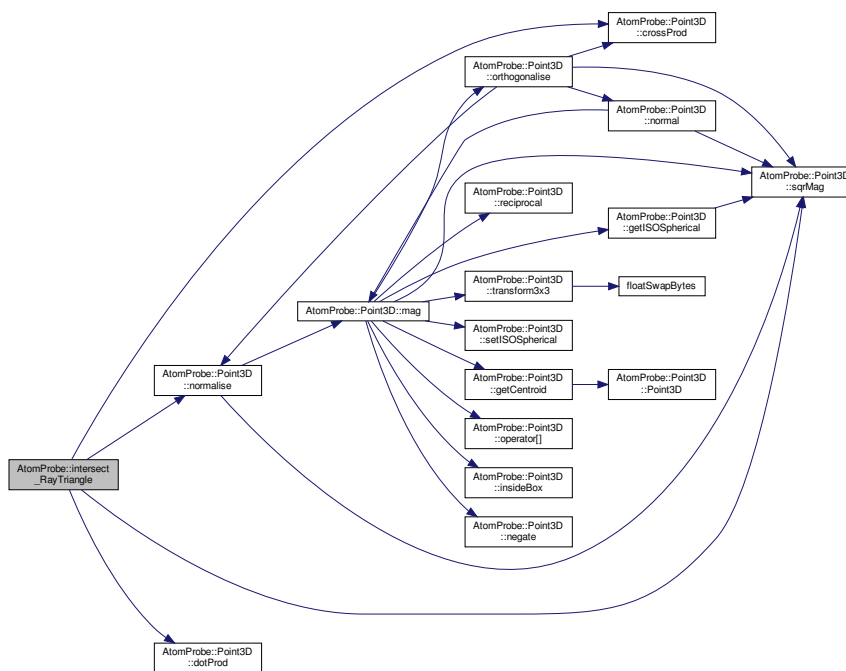
```

Definition at line 280 of file mesh.cpp.

References AtomProbe::Point3D::crossProd(), AtomProbe::Point3D::dotProd(), AtomProbe::Point3D::normalise(), and AtomProbe::Point3D::sqrMag().

Referenced by AtomProbe::Mesh::pointsInside().

Here is the call graph for this function:



5.1.3.75 isNotDirectory()

```

bool AtomProbe::isNotDirectory (
    const char * filename )

```

Definition at line 170 of file helpFuncs.cpp.

Referenced by AtomProbe::RangeFile::detectFileType(), loadTextData(), and AtomProbe::ComparePairFirstReverse::operator()().

5.1.3.76 kahansum()

```

template<class T , class T2 >
std::iterator_traits<T>::value_type AtomProbe::kahansum (
    T begin,
    T end,
    T2 other )

```

Definition at line 34 of file confidence.cpp.

Referenced by cumTrapezoid().

5.1.3.77 leastSquaresDeconvolve()

```

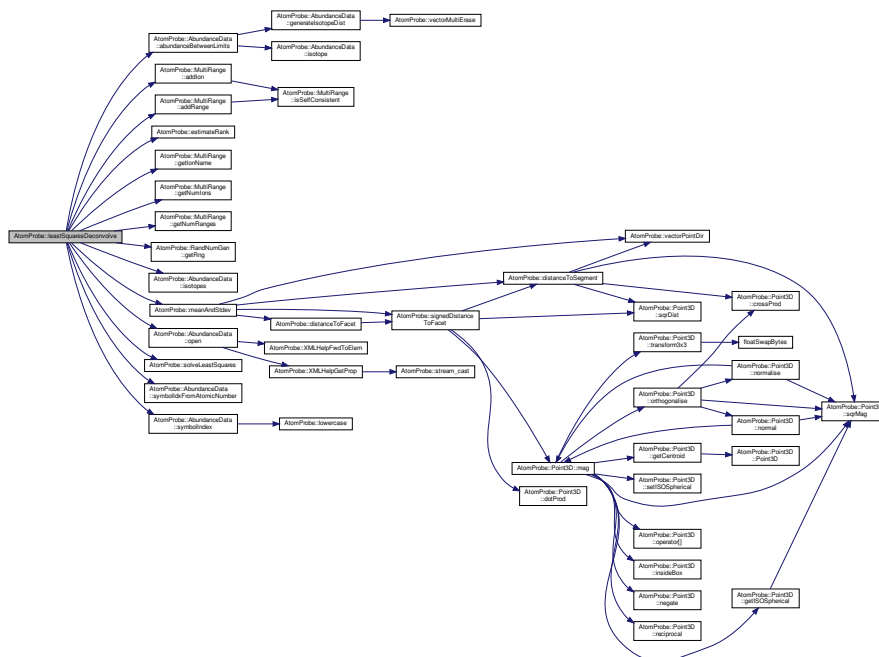
bool AtomProbe::leastSquaresDeconvolve (
    const MultiRange & rangeData,
    const AbundanceData & abundance,
    const float massTol,
    const vector< IonHit > & hits,
    float(*) (float, float) backgroundEstimator,
    vector< pair< unsigned int, float > > & decomposedHits )

```

Definition at line 32 of file deconvolution.cpp.

References AtomProbe::AbundanceData::abundanceBetweenLimits(), AtomProbe::MultiRange::addIon(), AtomProbe::MultiRange::addRange(), ASSERT, AtomProbe::SIMPLE_SPECIES::atomicNumber, AtomProbe::RGBF::blue, AtomProbe::SIMPLE_SPECIES::count, estimateRank(), AtomProbe::MultiRange::getIonName(), AtomProbe::MultiRange::getNumIons(), AtomProbe::MultiRange::getNumRanges(), AtomProbe::RandNumGen::getRng(), AtomProbe::RGBF::green, AtomProbe::AbundanceData::isotopes(), meanAndStdev(), AtomProbe::AbundanceData::open(), randGen, AtomProbe::RGBF::red, solveLeastSquares(), AtomProbe::AbundanceData::symbolIdxFromAtomicNumber(), AtomProbe::AbundanceData::symbolIndex(), and WARN.

Here is the call graph for this function:



5.1.3.78 linearHistogram()

```
template<class T >
void AtomProbe::linearHistogram (
    const std::vector< T > & data,
    T start,
    T end,
    T step,
    std::vector< T > & histVals )
```

Make a linearly spaced histogram with the given spacings Takes a vector of floats as input,

Parameters

<i>data</i>	
-------------	--

Definition at line 75 of file histogram.h.

References ASSERT.

Referenced by main().

5.1.3.79 linkIdentifiers()

```
template<class T >
void AtomProbe::linkIdentifiers (
    vector< T > & link )
```

Definition at line 88 of file multiRange.cpp.

Referenced by AtomProbe::MultiRange::splitOverlapping().

5.1.3.80 loadATOFile()

```
unsigned int AtomProbe::loadATOFile (
    const char * fileName,
    std::vector< ATO_ENTRY > & ions,
    unsigned int forceEndian = 0 )
```

Load a LAWATAP "ATO" file.

ATO files contain a mix of experiment and analysis data, including voltage, positioning and reconstructed data. There are specific algorithm generated data (eg clustering) which are not loaded in structures in this library to reduce memory consumption. Returns 0 on success, nonzero on error

- Filename : the name of the data file to read
- ions : the returned entries from the file

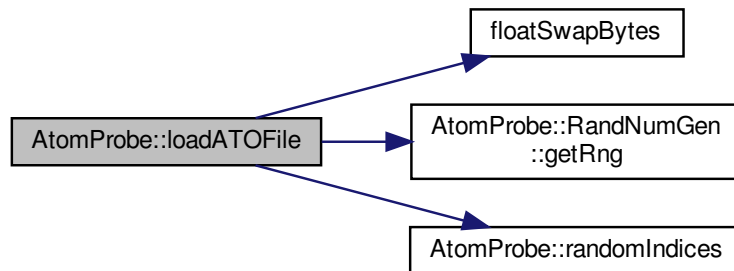
- `forceEndian` : if nonzero, this will force the use of either little-endian reading (1), or big-endian(2). If zero, the routine will try to automatically determine endianness from the file contents.

Returned code if nonzero can be converted to human readable form using `ATO_ERR_STRINGS`. Note if calling from outside C++, use `getAtoErrString(...)` to return the error string

Definition at line 1204 of file `dataFiles.cpp`.

References `ASSERT`, `ATO_EMPTY_FAIL`, `ATO_MEM_ERR`, `ATO_OPEN_FAIL`, `ATO_SIZE_ERR`, `ATO_VERSIONCHECK_ERR`, `floatSwapBytes()`, `AtomProbe::RandNumGen::getRng()`, `randGen`, and `randomIndices()`.

Here is the call graph for this function:



5.1.3.81 loadEposFile()

```

size_t AtomProbe::loadEposFile (
    std::vector< EPOS_ENTRY > & outData,
    const char * filename )
  
```

Load an entire "EPOS" File.

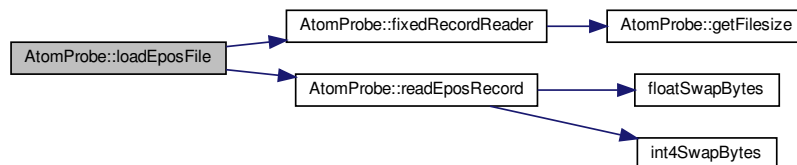
Epos files are very large and unwieldy. Ensure that you really want to do this, as it is possible you will run out of memory. Consider using the "chunkLoadEposFile" to process your data if possible. The file contains x,y,z,m-to-c,tof(uncorrected?). The return value is nonzero on error, and is part of the `RECORDREAD_ERR` enum

Definition at line 748 of file `dataFiles.cpp`.

References `EPOS_RECORD_SIZE`, `fixedRecordReader()`, and `readEposRecord()`.

Referenced by `getAtoErrString()`, and `main()`.

Here is the call graph for this function:



5.1.3.82 loadPosFile() [1/2]

```
unsigned int AtomProbe::loadPosFile (
    std::vector< IonHit > & posIons,
    const char * posFile )
```

Load a pos file directly into a single ion list.

Load a pos file from a file. Returns nonzero on exit.

Pos files are fixed record size files, with data stored as 4byte big endian floating point. (IEEE 754?). Data is stored as x,y,z,mass/charge.

Currently loads entire file in one gulp. This is bad from the point of view of gigabyte size pos files generated by laser atom probes.. Returns 0 on ok, nonzero on success.

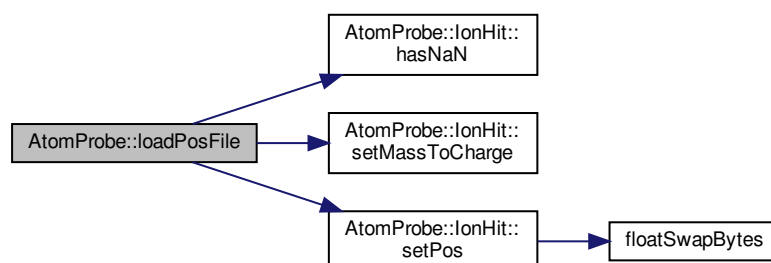
arguments : posfile | path to pos-formatted file posIons | will return any ions that are loaded from the dataset. Behaviour for loading into * a non-empty vector is undefined Return code can be decoded with getPosFileErrString

Definition at line 256 of file dataFiles.cpp.

References ASSERT, AtomProbe::IonHit::hasNaN(), AtomProbe::IONHIT::massToCharge, AtomProbe::IONHIT::pos, POS_ALLOC_FAIL, POS_NAN_LOAD_ERROR, POS_OPEN_FAIL, POS_READ_FAIL, POS_SIZE_EMPTY_ERR, POS_SIZE_MODULUS_ERR, AtomProbe::IonHit::setMassToCharge(), and AtomProbe::IonHit::setPos().

Referenced by getAtoErrString(), loadPosFile(), and main().

Here is the call graph for this function:



5.1.3.83 loadPosFile() [2/2]

```
unsigned int AtomProbe::loadPosFile (
    std::vector< IonHit > & posIons,
    const char * posFile,
    unsigned int nSamplesMax )
```

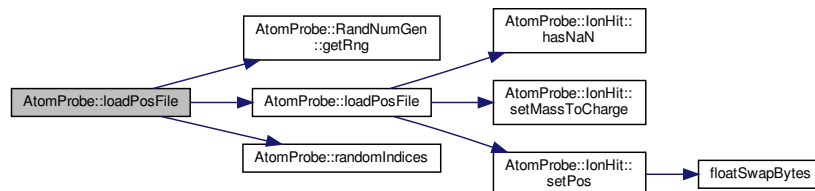
As per loadPosFile, but with an additional setting for the maximum number of ions to load.

Ions will be randomly sampled to form nSamples. This will be likely much slower than loading a full file, depending on the size of nSamples. The samples are not guaranteed to be the same for each load. Speed will be $n \log n$ in nSamplesMax, $o(1)$ in posIons.

Definition at line 364 of file dataFiles.cpp.

References AtomProbe::RandNumGen::getRng(), loadPosFile(), AtomProbe::IONHIT::massToCharge, AtomProbe::IONHIT::pos, POS_ALLOC_FAIL, POS_NAN_LOAD_ERROR, POS_OPEN_FAIL, POS_READ_FAIL, POS_SIZE_EMPTY_ERR, POS_SIZE_MODULUS_ERR, randGen, and randomIndices().

Here is the call graph for this function:



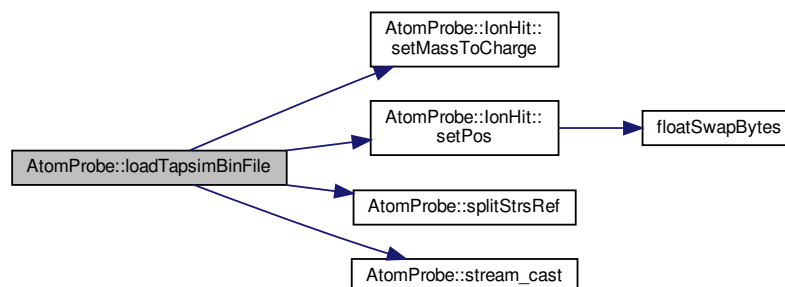
5.1.3.84 loadTapsimBinFile()

```
unsigned int AtomProbe::loadTapsimBinFile (
    vector< IonHit > & posIons,
    const char * posfile )
```

Definition at line 538 of file dataFiles.cpp.

References AtomProbe::IonHit::setMassToCharge(), AtomProbe::IonHit::setPos(), splitStrsRef(), stream_cast(), TAPSIM_FILE_FORMAT_FAIL, and TAPSIM_OPEN_FAIL.

Here is the call graph for this function:



5.1.3.85 loadTextData()

```
unsigned int AtomProbe::loadTextData (
    const char * cpFilename,
    std::vector< std::vector< float > > & dataVec,
    std::vector< std::string > & headerVec,
    const char * delim,
    bool allowNan = true )
```

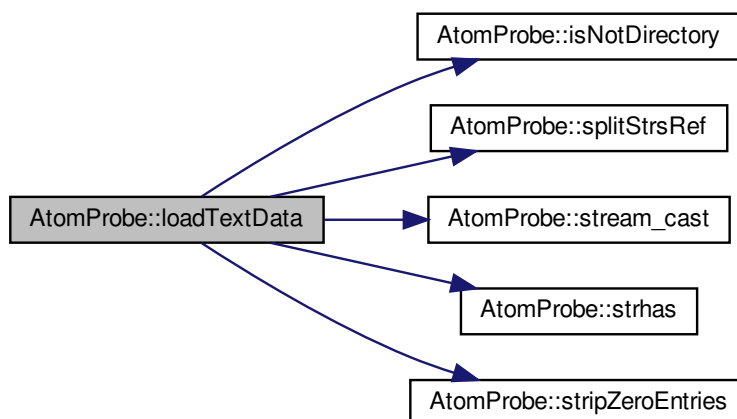
Load a CSV, TSV or similar text file. Assumes "C" Locale for input (ie "." as decimal separator).

Definition at line 1496 of file dataFiles.cpp.

References `isNotDirectory()`, `splitStrsRef()`, `stream_cast()`, `strhas()`, `stripZeroEntries()`, `TEXT_ERR_FILE_FORMAT`, `TEXT_ERR_FILE_NUM_FIELDS`, and `TEXT_ERR_FILE_OPEN`.

Referenced by `getAtoErrString()`, and `main()`.

Here is the call graph for this function:



5.1.3.86 lowercase()

```
std::string AtomProbe::lowercase (
    std::string s )
```

Return a lowercase version for a given string.

Definition at line 176 of file stringFuncs.cpp.

Referenced by `AtomProbe::RangeFile::detectFileType()`, `main()`, `parseCompositionData()`, `AtomProbe::RangeFile::rangeTypeString()`, `strAppend()`, and `AtomProbe::AbundanceData::symbolIndex()`.

5.1.3.87 makeHistogram()

```
void AtomProbe::makeHistogram (
    const vector< float > & data,
    float start,
    float end,
    float step,
    vector< float > & histVals )
```

Definition at line 110 of file processing.cpp.

References ASSERT.

Referenced by doFitBackground(), and findPeaks().

5.1.3.88 marchingCubes()

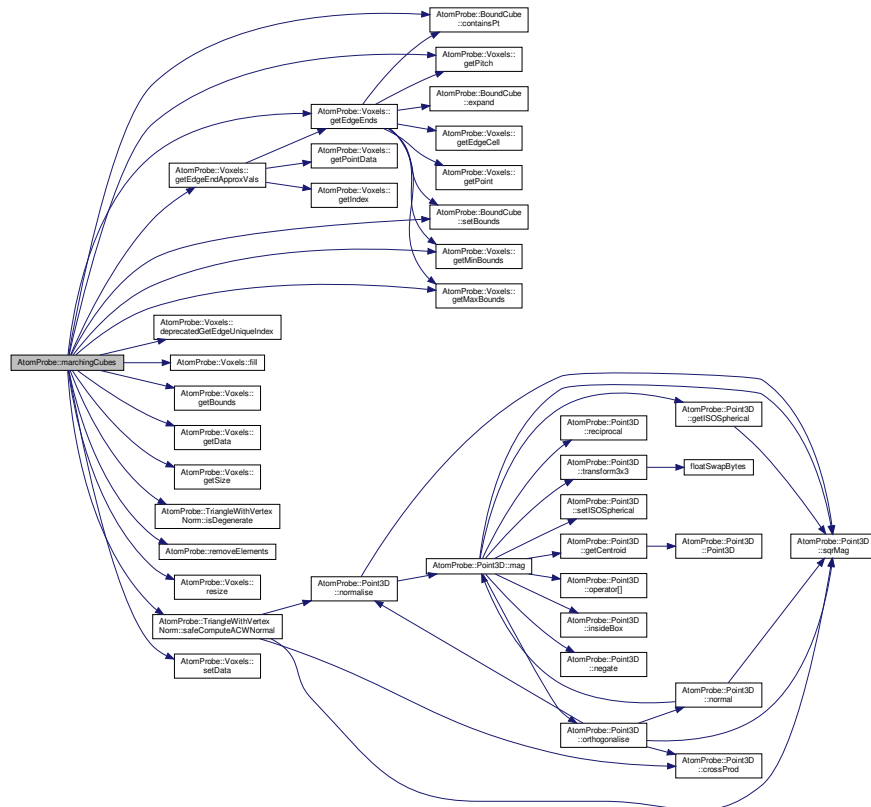
```
void AtomProbe::marchingCubes (
    const Voxels< float > & v,
    float isoValue,
    std::vector< TriangleWithVertexNorm > & tVec )
```

Definition at line 494 of file isoSurface.cpp.

References ASSERT, AtomProbe::BoundCube::containsPt(), AtomProbe::Voxels< T >::deprecatedGetEdge↔UniqueIndex(), AtomProbe::Voxels< T >::fill(), AtomProbe::Voxels< T >::getBounds(), AtomProbe::Voxels< T >::getData(), AtomProbe::Voxels< T >::getEdgeEndApproxVals(), AtomProbe::Voxels< T >::getEdgeEnds(), AtomProbe::Voxels< T >::getMaxBounds(), AtomProbe::Voxels< T >::getMinBounds(), AtomProbe::Voxels< T >::getPitch(), AtomProbe::Voxels< T >::getSize(), AtomProbe::TriangleWithVertexNorm::isDegenerate(), Atom↔Probe::TriangleWithVertexNorm::normal, AtomProbe::TriangleWithVertexNorm::p, AtomProbe::TriangleWith↔IndexedVertices::p, AtomProbe::TRIANGLE::p, AtomProbe::TRIANGLE::physGroup, removeElements(), Atom↔Probe::Voxels< T >::resize(), AtomProbe::TriangleWithVertexNorm::safeComputeACWNormal(), AtomProbe::↔BoundCube::setBounds(), AtomProbe::Voxels< T >::setData(), and TEST.

Referenced by main().

Here is the call graph for this function:



5.1.3.89 matchComposedName()

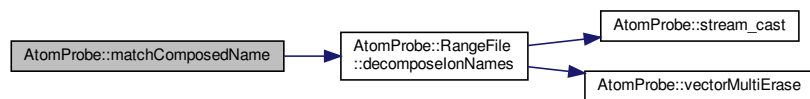
```
bool AtomProbe::matchComposedName (
    const std::map< string, size_t > & composedNames,
    const vector< pair< string, size_t > > & namesToFind,
    size_t & matchOffset )
```

Definition at line 253 of file ranges.cpp.

References `AtomProbe::RangeFile::decomposeNames()`.

Referenced by `AtomProbe::RangeFile::open()`.

Here is the call graph for this function:



5.1.3.90 maxExplainedFraction() [1/2]

```
std::vector<float> AtomProbe::maxExplainedFraction (
    const std::vector< std::pair< float, float > > & massData,
    float peakMass,
    float massWidth,
    const std::vector< std::vector< ISOTOPE_ENTRY > > & solutions,
    const AbundanceData & massTable,
    float massDistTol,
    unsigned int solutionCharge )
```

Compute the fraction of the data that has been explained, using the natural abundance information, and intensity data.

Specifically, this returns the maximum possible fraction of this peak that is explained by the isotopic fingerprint for a given solution set, and the count data given. This is returned as a vector with a value for each proposed solution. Note the solutions do not need to sum to 1.

massData - list of location and intensities for peaks in dataset peakMass - peak we wish to query. Must be present (within tolerance) in peak list (massData) massWidth - tolerance width (+-) when locating peaks solutions - list of candidate solutions we want to find the explained fraction for massDistTol - tolerance to use when checking for peaks in mass distribution solutionCharge - charge state to apply to isotope distribution

Referenced by maxExplainedFraction().

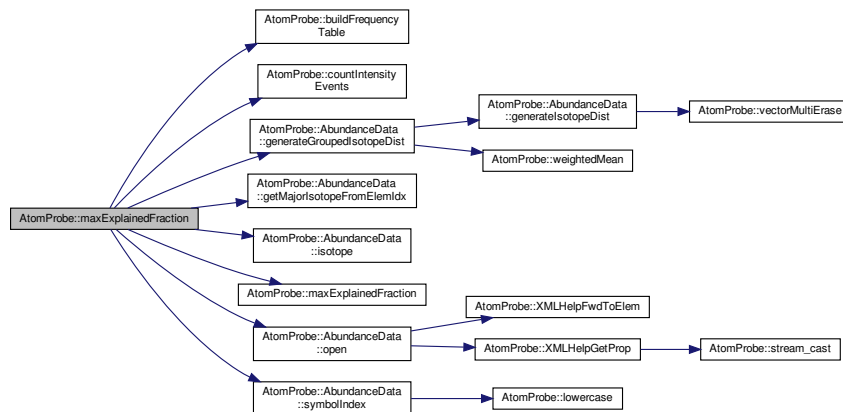
5.1.3.91 maxExplainedFraction() [2/2]

```
vector<float> AtomProbe::maxExplainedFraction (
    const vector< pair< float, float > > & intensityData,
    float peakMass,
    float massWidth,
    const vector< vector< ISOTOPE_ENTRY > > & solutions,
    const AbundanceData & massTable,
    float massDistTol,
    unsigned int solutionCharge )
```

Definition at line 206 of file filter.cpp.

References ASSERT, buildFrequencyTable(), countIntensityEvents(), AtomProbe::AbundanceData::generateGroupedIsotopeDist(), AtomProbe::AbundanceData::getMajorIsotopeFromElemIdx(), AtomProbe::AbundanceData::isotope(), AtomProbe::ISOTOPE_ENTRY::mass, maxExplainedFraction(), AtomProbe::AbundanceData::open(), AtomProbe::AbundanceData::symbolIndex(), and TEST.

Here is the call graph for this function:



5.1.3.92 meanAndStdev()

```

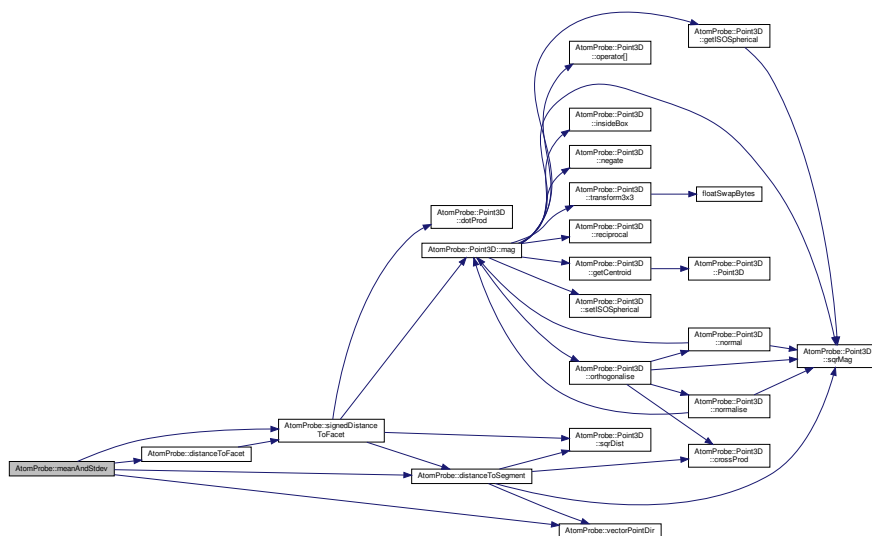
template<typename T >
void AtomProbe::meanAndStdev (
    const std::vector< T > & f,
    float & meanVal,
    float & stdevVal,
    bool normalCorrection = true )
  
```

Definition at line 55 of file misc.h.

References [distanceToFacet\(\)](#), [distanceToSegment\(\)](#), [signedDistanceToFacet\(\)](#), and [vectorPointDir\(\)](#).

Referenced by [andersonDarlingStatistic\(\)](#), and [leastSquaresDeconvolve\(\)](#).

Here is the call graph for this function:



5.1.3.93 myStrDup()

```
char* AtomProbe::myStrDup (
    const char * s )
```

Definition at line 39 of file helpFuncs.cpp.

Referenced by pushLocale().

5.1.3.94 nullBackground()

```
float AtomProbe::nullBackground (
    float rangeStart,
    float rangeEnd )
```

Definition at line 26 of file deconvolution.cpp.

5.1.3.95 nullifyMarker()

```
void AtomProbe::nullifyMarker (
    char * buffer,
    char marker )
```

Definition at line 55 of file stringFuncs.cpp.

Referenced by AtomProbe::RangeFile::rangeTypeString(), and stlStrToStlWStr().

5.1.3.96 numericalEstimateGaussRatioConf()

```
bool AtomProbe::numericalEstimateGaussRatioConf (
    float mu1,
    float mu2,
    float var1,
    float var2,
    float alpha,
    unsigned int nTrials,
    gsl_rng * r,
    float & lBound,
    float & uBound )
```

Brute-force gaussian ratio confidence estimator.

Performs monte-carlo trials to determine the distribution of $Z=X/Y$. where $X\sim\text{Gaussian}(L1)$, $Y\sim\max(\text{Gaussian}(L2),1)$

Biased, as it enforces 1, in the denominator of $L1/L2$ (ie $L2=0$ has $p=0$) returns false if either bound could not be computed

- mu1 : Mean 1 (must be +ve)
- mu2 : Mean 2 (must be +ve)
- var1 : Variance 1 (must be +ve)
- var2 : Variance 2 (must be +ve)
- alpha : confidence interval in distribution
- nTrials : number of computations of Z.
- lBound : returned lower bound at confidence alpha
- uBound : returned upper bound at confidence alpha

Note that the estimator seems to be biased to the number of trials, and needs to be improved.

Definition at line 260 of file confidence.cpp.

References ASSERT.

Referenced by poissonConfidenceObservation().

5.1.3.97 numericalEstimatePoissonRatioConf()

```
bool AtomProbe::numericalEstimatePoissonRatioConf (
    float lambda1,
    float lambda2,
    float alpha,
    unsigned int nTrials,
    gsl_rng * r,
    float & lBound,
    float & uBound )
```

Brute-force poisson ratio confidence estimator. Returns the confidence interval in the estimate of the mean, at the given rates.

Performs monte-carlo trials to determine the distribution of $Z=X/Y$. where $X\sim\text{Poisson}(L1)$, $Y\sim\text{Poisson}(L2)$ Biased, as it enforces 1, in the denominator of $L1/L2$ (ie $L2=0$ has $p=0$) returns false if either bound could not be computed

- lambda1 : Poisson rate 1 (must be +ve)
- lambda2 : Poisson rate 2 (must be +ve)
- alpha : confidence interval in distribution
- nTrials : number of computations of Z.
- lBound : returned lower bound at confidence alpha. This is interpolated, so is a real number
- uBound : returned upper bound at confidence alpha. This is interpolated, so is a real number.

Note that the estimator seems to be slightly biased to the number of trials, and needs to be improved.

Definition at line 53 of file confidence.cpp.

References ASSERT.

Referenced by poissonConfidenceObservation().

5.1.3.98 numericalEstimateSkellamConf()

```
bool AtomProbe::numericalEstimateSkellamConf (
    float lambda1,
    float lambda2,
    float alpha,
    unsigned int nTrials,
    gsl_rng * r,
    float & lBound,
    float & uBound )
```

Brute-force the confidence bounds of a skellam distribution.

Performs monte-carlo trials to determine $Z = X - Y$, where X, Y are poisson ($\text{Lambda1}, \text{Lambda2}$).

- Lambda1 : poisson rate for X
- Lambda2 : poisson rate for Y
- alpha : confidence bound
- nTrials : number of computations of Z .
- lBound : returned lower bound at confidence alpha . This is interpolated, so is a real number
- uBound : returned upper bound at confidence alpha . This is interpolated, so is a real number

Definition at line 158 of file `confidence.cpp`.

References ASSERT.

Referenced by `poissonConfidenceObservation()`.

5.1.3.99 onlyDir()

```
std::string AtomProbe::onlyDir (
    const std::string & path )
```

Return only the directory name component of the full path.

Definition at line 44 of file `stringFuncs.cpp`.

Referenced by `strAppend()`.

5.1.3.100 onlyFilename()

```
std::string AtomProbe::onlyFilename (
    const std::string & path )
```

Return only the filename component.

Definition at line 33 of file `stringFuncs.cpp`.

Referenced by `strAppend()`.

5.1.3.101 operator<()

```
bool AtomProbe::operator< (
    const SIMPLE_SPECIES & a,
    const SIMPLE_SPECIES & b )
```

Definition at line 117 of file multiRange.cpp.

References AtomProbe::SIMPLE_SPECIES::atomicNumber, and AtomProbe::SIMPLE_SPECIES::count.

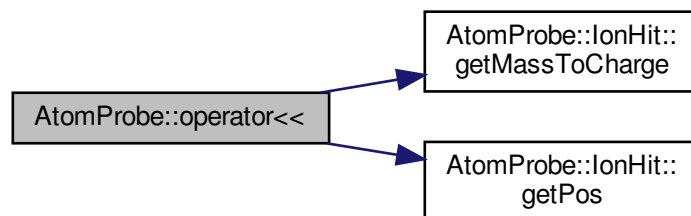
5.1.3.102 operator<<() [1/3]

```
std::ostream& AtomProbe::operator<< (
    std::ostream & stream,
    const IonHit & ion )
```

Definition at line 181 of file ionhit.cpp.

References AtomProbe::IonHit::getMassToCharge(), and AtomProbe::IonHit::getPos().

Here is the call graph for this function:



5.1.3.103 operator<<() [2/3]

```
std::ostream& AtomProbe::operator<< (
    std::ostream & stream,
    const Point3D & pt )
```

Definition at line 425 of file point3D.cpp.

5.1.3.104 operator<<() [3/3]

```
std::ostream& AtomProbe::operator<< (
    std::ostream & stream,
    const BoundCube & b )
```

Definition at line 563 of file boundcube.cpp.

5.1.3.105 pairContains()

```
bool AtomProbe::pairContains (
    const pair< float, float > & p,
    float m )
```

Definition at line 46 of file rangeCheck.cpp.

References ASSERT.

Referenced by checkMassRangingCorrectness().

5.1.3.106 pairOverlaps()

```
bool AtomProbe::pairOverlaps (
    float aStart,
    float aEnd,
    float bStart,
    float bEnd )
```

Definition at line 122 of file multiRange.cpp.

References ASSERT.

Referenced by AtomProbe::MultiRange::splitOverlapping().

5.1.3.107 parseColString()

```
bool AtomProbe::parseColString (
    const std::string & str,
    unsigned char & r,
    unsigned char & g,
    unsigned char & b,
    unsigned char & a )
```

Parse a colour string containing rgb[a]; hex for with leading #.

Definition at line 269 of file stringFuncs.cpp.

References hexStrToUChar().

Referenced by AtomProbe::RGBf::fromHex(), AtomProbe::MultiRange::open(), AtomProbe::RangeFile::range↔TypeString(), and strAppend().

Here is the call graph for this function:



5.1.3.108 parseCompositionData()

```
unsigned int AtomProbe::parseCompositionData (
    const std::vector< std::string > & s,
    const AbundanceData & atomTable,
    std::map< unsigned int, double > & atomData )
```

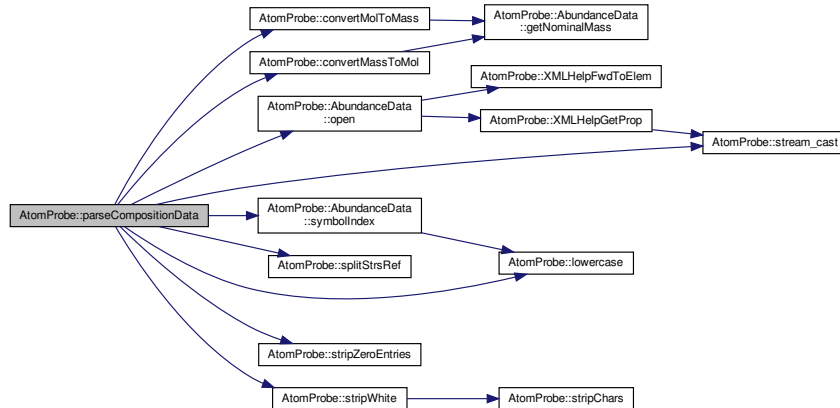
Convert atomic string label data, in the form "Fe 0.81" to fraction map.

The input type (mol/mass) is not relevant for this function, but should be in fraction the label "bal" can be used in place of a composition value exactly once to provide "balance" mass to sum to 1 returns 0 on success, nonzero on error (see COMPPARSE enum)

Definition at line 100 of file massconvert.cpp.

References COMPPARSE_BAD_BAL_USAGE, COMPPARSE_BAD_COMP_VALUE, COMPPARSE_BAD_INP↔UT_FORMAT, COMPPARSE_BAD_SYMBOL, COMPPARSE_BAD_TOTAL_COMP, COMPPARSE_DUPLICAT↔E_SYMBOL, convertMassToMol(), convertMolToMass(), lowercase(), AtomProbe::AbundanceData::open(), split↔StrsRef(), stream_cast(), stripWhite(), stripZeroEntries(), AtomProbe::AbundanceData::symbolIndex(), and TEST.

Here is the call graph for this function:



5.1.3.109 poissonConfidenceObservation()

```

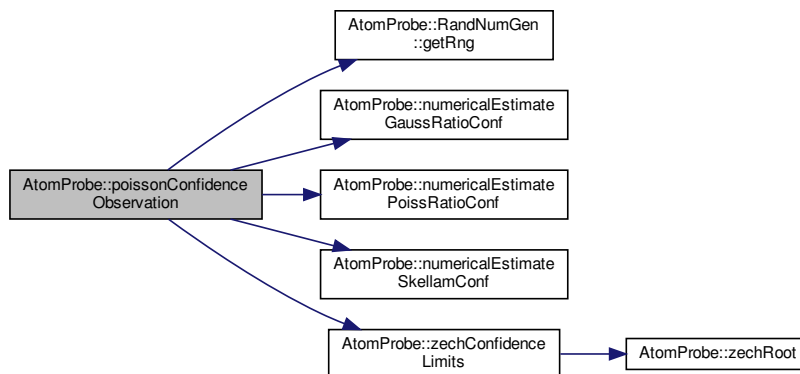
void AtomProbe::poissonConfidenceObservation (
    float counts,
    float alpha,
    float & lBound,
    float & uBound )
  
```

Obtain poisson confidence limits for the mean of a poisson distribution.

Definition at line 496 of file confidence.cpp.

References ASSERT, AtomProbe::RandNumGen::getRng(), numericalEstimateGaussRatioConf(), numericalEstimatePoissRatioConf(), numericalEstimateSkellamConf(), randGen, TEST, and zechConfidenceLimits().

Here is the call graph for this function:



5.1.3.110 popLocale()

```
void AtomProbe::popLocale ( )
```

Definition at line 96 of file helpFuncs.cpp.

Referenced by AtomProbe::RangeFile::openFormat().

5.1.3.111 pushLocale()

```
void AtomProbe::pushLocale (
    const char * newLocale,
    int type )
```

Definition at line 62 of file helpFuncs.cpp.

References ASSERT, and myStrDup().

Referenced by AtomProbe::RangeFile::openFormat().

Here is the call graph for this function:



5.1.3.112 pyramidVol()

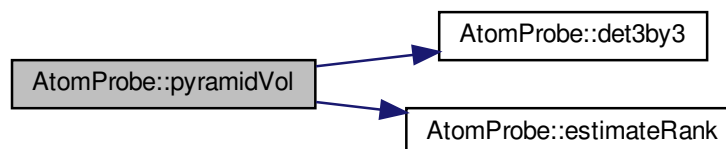
```
double AtomProbe::pyramidVol (
    const Point3D * planarPts,
    const Point3D & apex )
```

Definition at line 239 of file misc.cpp.

References ASSERT, det3by3(), and estimateRank().

Referenced by dotProduct(), and GetReducedHullPts().

Here is the call graph for this function:



5.1.3.113 quat_get_rot_quat()

```
void AtomProbe::quat_get_rot_quat (
    const Point3f * rotVec,
    float angle,
    Quaternion * rotQuat )
```

Compute the quaternion for specified rotation.

angle is in radians. pass to quat_rot_apply_quat to compute rotation

Definition at line 184 of file quat.cpp.

References AtomProbe::Quaternion::a, ASSERT, AtomProbe::Quaternion::b, AtomProbe::Quaternion::c, AtomProbe::Quaternion::d, AtomProbe::Point3f::fx, AtomProbe::Point3f::fy, and AtomProbe::Point3f::fz.

Referenced by AtomProbe::Mesh::rotate().

5.1.3.114 quat_invert()

```
void AtomProbe::quat_invert (
    Quaternion * quat )
```

Definition at line 219 of file quat.cpp.

References AtomProbe::Quaternion::b, AtomProbe::Quaternion::c, and AtomProbe::Quaternion::d.

5.1.3.115 quat_mult_no_second_a()

```
void AtomProbe::quat_mult_no_second_a (
    Quaternion * result,
    const Quaternion * q1,
    const Quaternion * q2 )
```

Definition at line 39 of file quat.cpp.

References AtomProbe::Quaternion::a, AtomProbe::Quaternion::b, AtomProbe::Quaternion::c, and AtomProbe::Quaternion::d.

Referenced by quat_rot(), quat_rot_apply_quat(), and quat_rot_array().

5.1.3.116 quat_pointmult()

```
void AtomProbe::quat_pointmult (
    Point3f * result,
    const Quaternion * q1,
    const Quaternion * q2 )
```

Definition at line 50 of file quat.cpp.

References AtomProbe::Quaternion::a, AtomProbe::Quaternion::b, AtomProbe::Quaternion::c, AtomProbe::Quaternion::d, AtomProbe::Point3f::fx, AtomProbe::Point3f::fy, and AtomProbe::Point3f::fz.

Referenced by quat_rot(), quat_rot_apply_quat(), and quat_rot_array().

5.1.3.117 quat_rot() [1/2]

```
void AtomProbe::quat_rot (
    Point3D & p,
    const Point3D & r,
    float angle )
```

Rotate a point around a given rotation axis by a specified angle.

This is not efficient for large numbers of points, as this will recompute the quaternion each time

Definition at line 59 of file quat.cpp.

References AtomProbe::Point3f::fx, AtomProbe::Point3f::fy, and AtomProbe::Point3f::fz.

5.1.3.118 quat_rot() [2/2]

```
void AtomProbe::quat_rot (
    Point3f * point,
    const Point3f * rotVec,
    float angle )
```

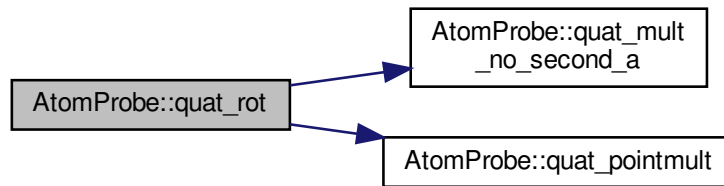
Rotate a point around a given vector, with specified angle.

This is not efficient for large numbers of points, as this will recompute the quaternion each time. angle (radians) is anti-clockwise as viewed from positive side of vector (right-handed)

Definition at line 77 of file quat.cpp.

References AtomProbe::Quaternion::a, ASSERT, AtomProbe::Quaternion::b, AtomProbe::Quaternion::c, AtomProbe::Quaternion::d, AtomProbe::Point3f::fx, AtomProbe::Point3f::fy, AtomProbe::Point3f::fz, quat_mult_no_second_a(), and quat_pointmult().

Here is the call graph for this function:



5.1.3.119 quat_rot_apply_quat()

```

void AtomProbe::quat_rot_apply_quat (
    Point3f * point,
    const Quaternion * rotQuat )
  
```

Use previously generated quats from `quat_get_rot_quats` to rotate a point.

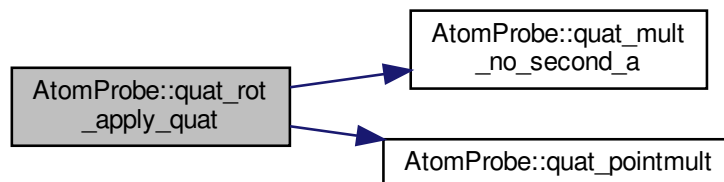
Rotation is in-place, on single point

Definition at line 206 of file `quat.cpp`.

References `AtomProbe::Quaternion::b`, `AtomProbe::Quaternion::c`, `AtomProbe::Quaternion::d`, `AtomProbe::Point3f::fx`, `AtomProbe::Point3f::fy`, `AtomProbe::Point3f::fz`, `quat_mult_no_second_a()`, and `quat_pointmult()`.

Referenced by `AtomProbe::Mesh::rotate()`.

Here is the call graph for this function:



5.1.3.120 quat_rot_array() [1/2]

```
void AtomProbe::quat_rot_array (
    Point3f * point,
    unsigned int n,
    const Point3f * rotVec,
    float angle )
```

Rotate each point in array of size n around a given vector, with specified angle.

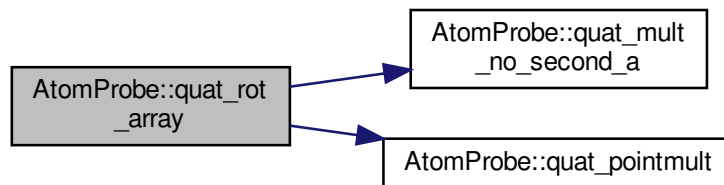
rotVec is the rotation, angle is the anti-clockwise angle as viewed from positive side of vector (right-handed) rotation of points is in-place

Definition at line 139 of file quat.cpp.

References AtomProbe::Quaternion::a, ASSERT, AtomProbe::Quaternion::b, AtomProbe::Quaternion::c, AtomProbe::Quaternion::d, AtomProbe::Point3f::fx, AtomProbe::Point3f::fy, AtomProbe::Point3f::fz, quat_mult_no_second_a(), and quat_pointmult().

Referenced by quat_rot_array().

Here is the call graph for this function:



5.1.3.121 quat_rot_array() [2/2]

```
void AtomProbe::quat_rot_array (
    Point3D * point,
    unsigned int n,
    const Point3f * rotVec,
    float angle )
```

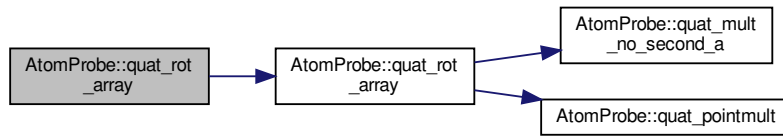
Rotate each point in array of size n around a given vector, with specified angle.

rotVec is the rotation, angle is the anti-clockwise angle as viewed from positive side of vector (right-handed). rotation of points is in-place

Definition at line 115 of file quat.cpp.

References AtomProbe::Point3f::fx, AtomProbe::Point3f::fy, AtomProbe::Point3f::fz, and quat_rot_array().

Here is the call graph for this function:



5.1.3.122 randomIndices()

```

template<class T >
void AtomProbe::randomIndices (
    std::vector< T > & res,
    size_t num,
    size_t nMax,
    gsl_rng * rng )
  
```

Definition at line 113 of file sampling.h.

Referenced by loadATOFile(), and loadPosFile().

5.1.3.123 randomSelect()

```

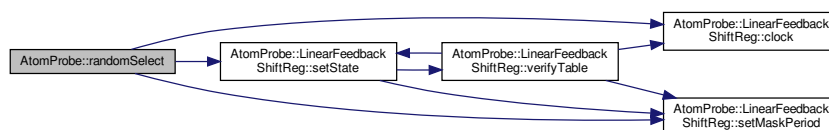
template<class T >
size_t AtomProbe::randomSelect (
    std::vector< T > & result,
    const std::vector< T > & source,
    size_t num,
    gsl_rng * rng )
  
```

Definition at line 40 of file sampling.h.

References AtomProbe::LinearFeedbackShiftReg::clock(), AtomProbe::LinearFeedbackShiftReg::setMaskPeriod(), and AtomProbe::LinearFeedbackShiftReg::setState().

Referenced by samplelons().

Here is the call graph for this function:



5.1.3.124 rangeOverlaps()

```
bool AtomProbe::rangeOverlaps (
    const pair< float, float > & r1,
    const pair< float, float > & r2 )
```

Definition at line 150 of file componentAnalysis.cpp.

Referenced by computeRangeAdjacency().

5.1.3.125 rangeToStr()

```
string AtomProbe::rangeToStr (
    const pair< float, float > & rng )
```

Definition at line 85 of file ranges.cpp.

References stream_cast().

Referenced by AtomProbe::RangeFile::isSelfConsistent().

Here is the call graph for this function:



5.1.3.126 readEposRecord()

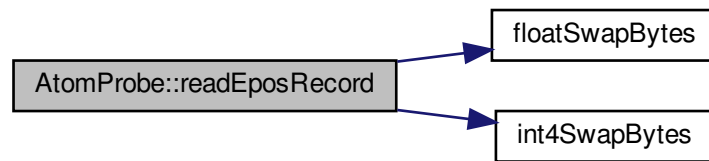
```
bool AtomProbe::readEposRecord (
    const char * src,
    const char * dest )
```

Definition at line 703 of file dataFiles.cpp.

References AtomProbe::EPOS_ENTRY::deltaPulse, floatSwapBytes(), AtomProbe::EPOS_ENTRY::hitMultiplicity, int4SwapBytes(), AtomProbe::EPOS_ENTRY::massToCharge, AtomProbe::EPOS_ENTRY::timeOfFlight, AtomProbe::EPOS_ENTRY::voltDC, AtomProbe::EPOS_ENTRY::voltPulse, AtomProbe::EPOS_ENTRY::x, AtomProbe::EPOS_ENTRY::xDetector, AtomProbe::EPOS_ENTRY::y, AtomProbe::EPOS_ENTRY::yDetector, and AtomProbe::EPOS_ENTRY::z.

Referenced by chunkLoadEposFile(), and loadEposFile().

Here is the call graph for this function:



5.1.3.127 readPosapOps()

```

unsigned int AtomProbe::readPosapOps (
    const char * file,
    THREEDAP_EXPERIMENT & data,
    unsigned int & badLine,
    unsigned int & progress,
    std::atomic< bool > & wantAbort,
    unsigned int nDelayLines = 2,
    bool strictMode = false )
  
```

Function to read POSAP "OPS" files.

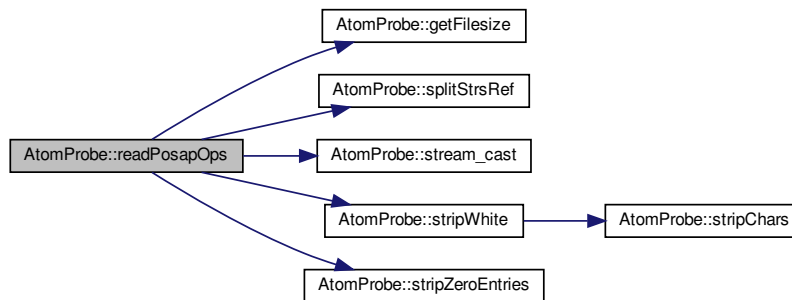
OPS files contain data on voltage, positioned hits and time of flights as well as some instrument parameters. This format is not standardised, so variants may exist

- `badLine` returns the line on which an error was encountered.
- `progress` gives a 0-100% progress value (counted in file size). Can be used to multithread monitor progress
- `nDelayLines` - The number of delay lines in the system. Usually 2, possibly 3.
- `strictMode` - enable to throw errors in cases where files appear to be badly formatted TODO: Can we work out how to obtain `nDelayLines` from parsing the file?

Definition at line 797 of file `dataFiles.cpp`.

References `AtomProbe::THREEDAP_DATA::alpha`, `ARRAYSIZE`, `ASSERT`, `AtomProbe::VOLTAGE_DATA::beta`, `AtomProbe::THREEDAP_DATA::beta`, `AtomProbe::THREEDAP_DATA::detectorChannels`, `AtomProbe::THREEDAP_DATA::detectorRadius`, `AtomProbe::THREEDAP_EXPERIMENT::eventData`, `AtomProbe::THREEDAP_EXPERIMENT::eventPulseNumber`, `AtomProbe::THREEDAP_DATA::flightPath`, `getFilesize()`, `AtomProbe::VOLTAGE_DATA::nextHitGroupOffset`, `OPSREADER_ABORT_ERR`, `OPSREADER_CHANNELS_DATA_ERR`, `OPSREADER_ENUM_END`, `OPSREADER_FORMAT_CHANNELS_ERR`, `OPSREADER_FORMAT_CLINE_ERR`, `OPSREADER_FORMAT_DETECTORLINE_ERR`, `OPSREADER_FORMAT_DOUBLEDASH`, `OPSREADER_FORMAT_DUPLICATE_DETECTORSIZE`, `OPSREADER_FORMAT_DUPLICATE_SYSDATA`, `OPSREADER_FORMAT_LINE_DASH_ERR`, `OPSREADER_FORMAT_LINE_VOLTAGE_DATA_ERR`, `OPSREADER_FORMAT_LINE_VOLTAGE_DATA_COUNT_ERR`, `OPSREADER_FORMAT_LINE_VOLTAGE_NOBETA`, `OPSREADER_FORMAT_LINE_EVENTTYPE_ERR`, `OPSREADER_FORMAT_SLINE_EVENTCOUNT_ERR`, `OPSREADER_FORMAT_SLINE_EVENTDATA_ERR`, `OPSREADER_FORMAT_SLINE_FORMAT_ERR`, `OPSREADER_FORMAT_SLINE_PREFIX_ERR`, `OPSREADER_FORMAT_TRAILING_DASH_ERR`, `OPSREADER_OPEN_ERR`, `OPSREADER_READ_ERR`, `AtomProbe::THREEDAP_EXPERIMENT::params`, `AtomProbe::VOLTAGE_DATA::pulseVolt`, `splitStrsRef()`, `stream_cast()`, `stripWhite()`, `stripZeroEntries()`, `AtomProbe::SINGLE_HIT::tof`, `AtomProbe::THREEDAP_DATA::tZero`, `AtomProbe::VOLTAGE_DATA::voltage`, and `AtomProbe::THREEDAP_EXPERIMENT::voltageData`.

Here is the call graph for this function:



5.1.3.128 reconstructTest()

```
bool AtomProbe::reconstructTest ( )
```

Referenced by `AtomProbe::ReconstructionSphereOnCone::reconstruct()`, and `runTests()`.

5.1.3.129 removeElements()

```
template<class T >
void AtomProbe::removeElements (
    const std::vector< size_t > & elems,
    std::vector< T > & vec )
```

Definition at line 64 of file `isoSurface.cpp`.

References `ASSERT`.

Referenced by `marchingCubes()`.

5.1.3.130 rotateMatch()

```
bool AtomProbe::rotateMatch (
    const unsigned int * a,
    const unsigned int * b,
    size_t n,
    bool directionForwards = true )
```

Definition at line 100 of file `mesh.cpp`.

Referenced by `AtomProbe::TRIANGLE::edgesMismatch()`.

5.1.3.131 sampleIons()

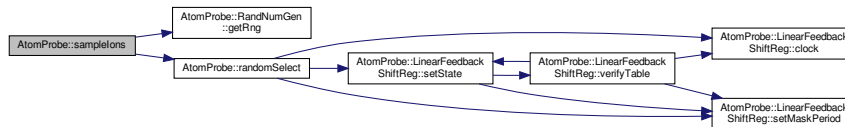
```
void AtomProbe::sampleIons (
    const std::vector< IonHit > & ions,
    float sampleFactor,
    std::vector< IonHit > & sampled,
    bool strongRandom = true )
```

Definition at line 30 of file sampling.cpp.

References AtomProbe::RandNumGen::getRng(), randGen, randomSelect(), and TEST.

Referenced by main().

Here is the call graph for this function:



5.1.3.132 savePosFile() [1/2]

```
unsigned int AtomProbe::savePosFile (
    const std::vector< Point3D > & points,
    float mass,
    const char * name,
    bool append = false )
```

Save a vector of Point3Ds into a pos file, using a fixed mass, return nonzero on error.

Create an pos file from a vector of points, and a given mass.

A nonzero return code can be decoded with getPosFileErrString, which will provide a human readable error message

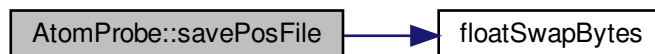
- if append is true, the the file will be appended to.

Definition at line 499 of file dataFiles.cpp.

References floatSwapBytes().

Referenced by getAtoErrString(), main(), and AtomProbe::ReconstructionSphereOnCone::reconstruct().

Here is the call graph for this function:



5.1.3.133 savePosFile() [2/2]

```
unsigned int AtomProbe::savePosFile (
    const std::vector< IonHit > & data,
    const char * name,
    bool append = false )
```

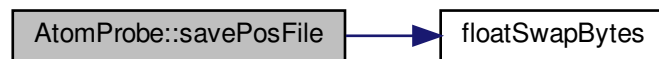
Save a vector of IonHits into a "pos" file, return nonzero on error.

Create an pos file from a vector of IonHits.

Definition at line 463 of file dataFiles.cpp.

References floatSwapBytes().

Here is the call graph for this function:



5.1.3.134 saveTapsimBin()

```
unsigned int AtomProbe::saveTapsimBin (
    std::ostream & f,
    std::vector< IonHit > & posIons )
```

Write a tapsim file from a bunch of IonHits.

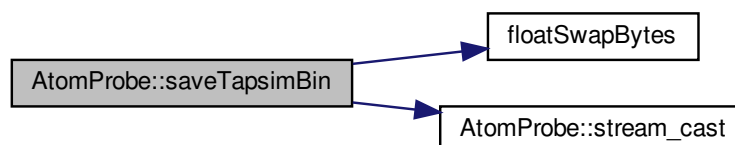
Note: there is some data loss here. Tapsim data basically writes a short header and then the ion data is little endian stored, not big-endian. Finally, m/c data is written as short, hence data loss

Definition at line 667 of file dataFiles.cpp.

References ASSERT, floatSwapBytes(), and stream_cast().

Referenced by getAtoErrString().

Here is the call graph for this function:



5.1.3.135 selectElements()

```
template<class T >
void AtomProbe::selectElements (
    const std::vector< T > & in,
    const std::vector< unsigned int > & indices,
    std::vector< T > & out )
```

Definition at line 60 of file misc.h.

Referenced by findPeaks().

5.1.3.136 setHardAssert()

```
void AtomProbe::setHardAssert (
    bool enabled )
```

Do assertions cause a straight up crash (enabled), or write a message to stderr (disabled)? By default, hard crash.

Definition at line 30 of file aptAssert.cpp.

5.1.3.137 signedDistanceToFacet()

```
float AtomProbe::signedDistanceToFacet (
    const Point3D & fA,
    const Point3D & fB,
    const Point3D & fC,
    const Point3D & normal,
    const Point3D & p )
```

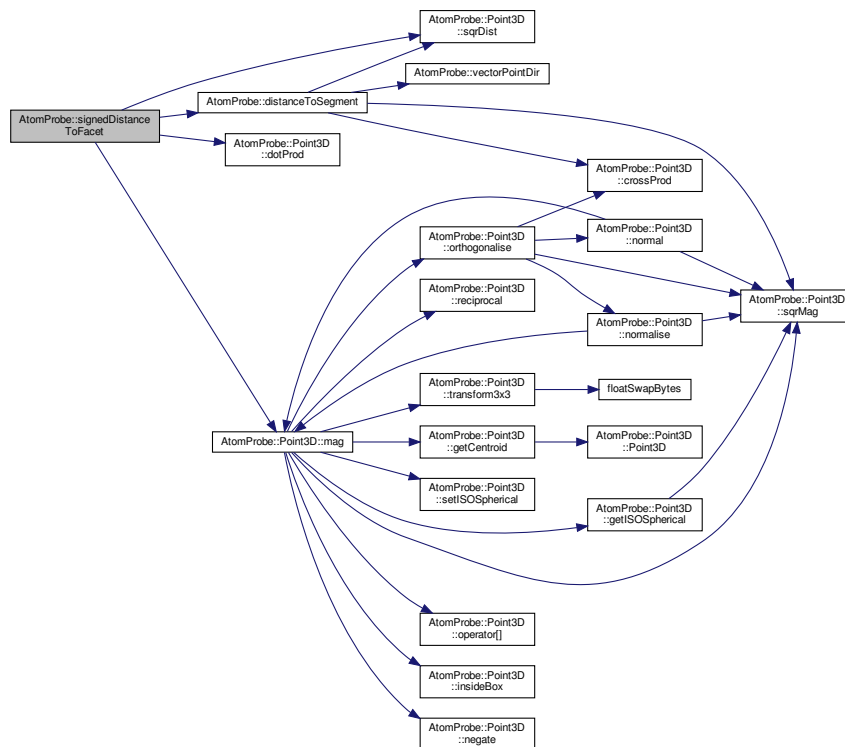
Find the distance between a point, and a triangular facet – may be positive or negative.

Definition at line 121 of file misc.cpp.

References ASSERT, distanceToSegment(), AtomProbe::Point3D::dotProd(), AtomProbe::Point3D::mag(), and AtomProbe::Point3D::sqrDist().

Referenced by distanceToFacet(), AtomProbe::Mesh::getNearestTri(), and meanAndStdev().

Here is the call graph for this function:



5.1.3.138 signVal()

```
float AtomProbe::signVal (
    unsigned int val )
```

Definition at line 147 of file mesh.cpp.

Referenced by Determinant().

5.1.3.139 solveLeastSquares()

```
bool AtomProbe::solveLeastSquares (
    gsl_matrix * m,
    gsl_vector * b,
    gsl_vector *& x )
```

Use an SVD based least-squares solver to solve $Mx=b$ (for x).

Definition at line 43 of file misc.cpp.

Referenced by leastSquaresDeconvolve().

5.1.3.140 SphericProjectionEqn()

```
double AtomProbe::SphericProjectionEqn (
    double eta,
    void * p )
```

Definition at line 25 of file projection.cpp.

References AtomProbe::SphericProjectionParams::focusDist, and AtomProbe::SphericProjectionParams::theta.

Referenced by AtomProbe::StereographicProjection::thetaToEta(), and AtomProbe::ModifiedFocusSphericProjection::thetaToEta().

5.1.3.141 splitStrsRef() [1/2]

```
void AtomProbe::splitStrsRef (
    const char * cpStr,
    const char delim,
    std::vector< std::string > & v )
```

Split string references using a single delimiter.

Definition at line 187 of file stringFuncs.cpp.

Referenced by AtomProbe::RangeFile::detectFileType(), AtomProbe::Mesh::loadGmshMesh(), loadTapsimBinFile(), loadTextData(), main(), AtomProbe::Point3D::parse(), parseCompositionData(), AtomProbe::RangeFile::rangeTypeString(), readPosapOps(), and strAppend().

5.1.3.142 splitStrsRef() [2/2]

```
void AtomProbe::splitStrsRef (
    const char * cpStr,
    const char * delim,
    std::vector< std::string > & v )
```

Split string references using any of a given string of delimiters.

Definition at line 221 of file stringFuncs.cpp.

5.1.3.143 stlStrToStlWStr()

```
std::wstring AtomProbe::stlStrToStlWStr (  
    const std::string & s ) [inline]
```

Definition at line 115 of file stringFuncs.h.

References nullifyMarker().

Here is the call graph for this function:



5.1.3.144 stlWStrToStlStr()

```
std::string AtomProbe::stlWStrToStlStr (  
    const std::wstring & s ) [inline]
```

Definition at line 108 of file stringFuncs.h.

5.1.3.145 strAppend()

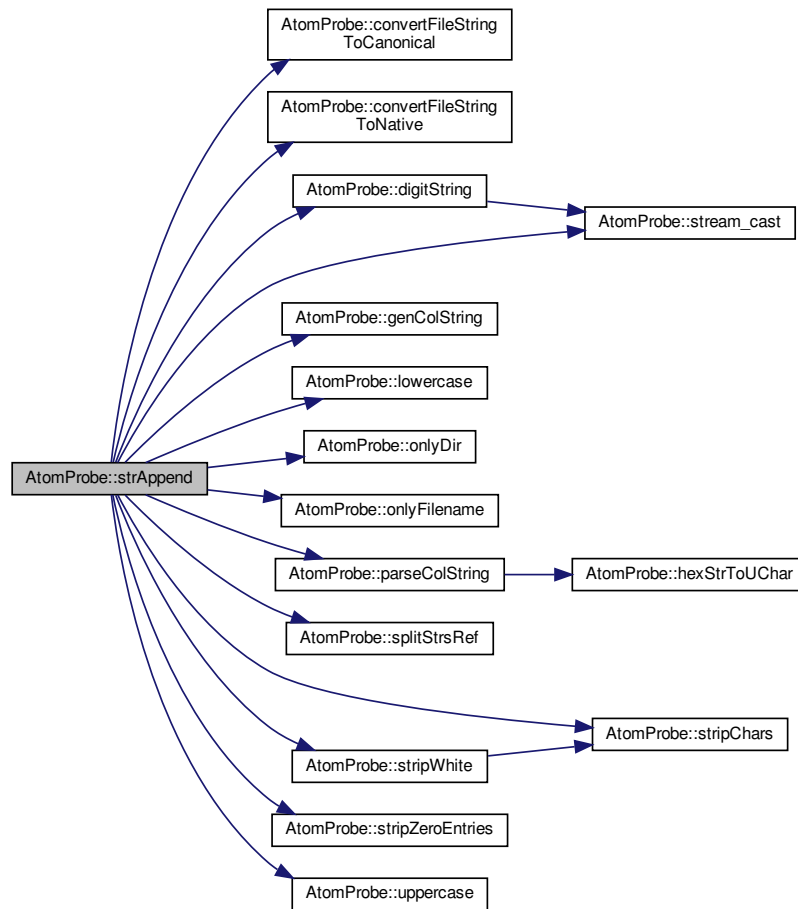
```
template<class T >  
void AtomProbe::strAppend (  
    std::string s,  
    const T & a )
```

Definition at line 41 of file stringFuncs.h.

References convertFileStringToCanonical(), convertFileStringToNative(), digitString(), genColString(), lowercase(), onlyDir(), onlyFilename(), parseColString(), splitStrsRef(), stream_cast(), stripChars(), stripWhite(), stripZeroEntries(), and uppercase().

Referenced by AtomProbe::RangeFile::isSelfConsistent().

Here is the call graph for this function:



5.1.3.146 stream_cast()

```

template<class T1 , class T2 >
bool AtomProbe::stream_cast (
    T1 & result,
    const T2 & obj )

```

Template function to cast and object to another by the stringstream.

Definition at line 32 of file stringFuncs.h.

Referenced by `AtomProbe::RangeFile::decomposeOnNames()`, `AtomProbe::RangeFile::detectFileType()`, `digitString()`, `AtomProbe::LibVersion::getVersionStr()`, `AtomProbe::Mesh::loadGmshMesh()`, `loadTapsimBinFile()`, `loadTextData()`, `main()`, `AtomProbe::MultiRange::open()`, `AtomProbe::RangeFile::open()`, `AtomProbe::Point3D::parse()`, `parseCompositionData()`, `rangeToStr()`, `AtomProbe::RangeFile::rangeTypeString()`, `readPosapOps()`, `saveTapsimBin()`, `strAppend()`, `XMLGetNextElemAttrib()`, and `XMLHelpGetProp()`.

5.1.3.147 strhas()

```
bool AtomProbe::strhas (
    const char * cpTest,
    const char * cpPossible )
```

Definition at line 1477 of file dataFiles.cpp.

Referenced by loadTextData().

5.1.3.148 stripChars()

```
std::string AtomProbe::stripChars (
    const std::string & Str,
    const char * chars )
```

Definition at line 137 of file stringFuncs.cpp.

Referenced by AtomProbe::RangeFile::rangeTypeString(), strAppend(), and stripWhite().

5.1.3.149 stripWhite()

```
std::string AtomProbe::stripWhite (
    const std::string & str )
```

Strip whitespace, (eg tab,space) from either side of a string.

Definition at line 132 of file stringFuncs.cpp.

References stripChars().

Referenced by AtomProbe::RangeFile::detectFileType(), AtomProbe::Point3D::parse(), parseCompositionData(), AtomProbe::RangeFile::rangeTypeString(), readPosapOps(), and strAppend().

Here is the call graph for this function:



5.1.3.150 stripZeroEntries()

```
void AtomProbe::stripZeroEntries (
    std::vector< std::string > & s )
```

Definition at line 155 of file stringFuncs.cpp.

Referenced by AtomProbe::RangeFile::detectFileType(), loadTextData(), main(), parseCompositionData(), AtomProbe::RangeFile::rangeTypeString(), readPosapOps(), and strAppend().

5.1.3.151 sumVoxels()

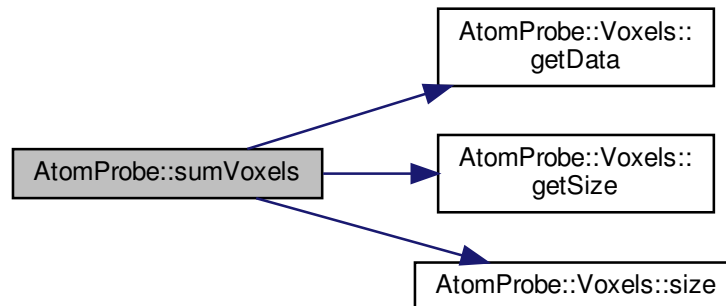
```
template<class T , class U >
void AtomProbe::sumVoxels (
    const Voxels< T > & src,
    U & counter )
```

Use one counting type to sum counts in a voxel of given type.

Definition at line 391 of file voxels.h.

References AtomProbe::Voxels< T >::getData(), AtomProbe::Voxels< T >::getSize(), and AtomProbe::Voxels< T >::size().

Here is the call graph for this function:



5.1.3.152 tabs()

```
std::string AtomProbe::tabs (
    unsigned int nTabs ) [inline]
```

Definition at line 99 of file stringFuncs.h.

Referenced by AtomProbe::MultiRange::write().

5.1.3.153 tolEqual()

```
template<class T >
bool AtomProbe::tolEqual (
    const T & a,
    const T & b,
    const T & f )
```

Definition at line 53 of file misc.h.

Referenced by AtomProbe::ModifiedFocusSphericProjection::scaleUp().

5.1.3.154 transposeVector()

```
template<class T >
void AtomProbe::transposeVector (
    std::vector< std::vector< T > > & v )
```

Definition at line 72 of file misc.h.

Referenced by main().

5.1.3.155 trilsDegenerate()

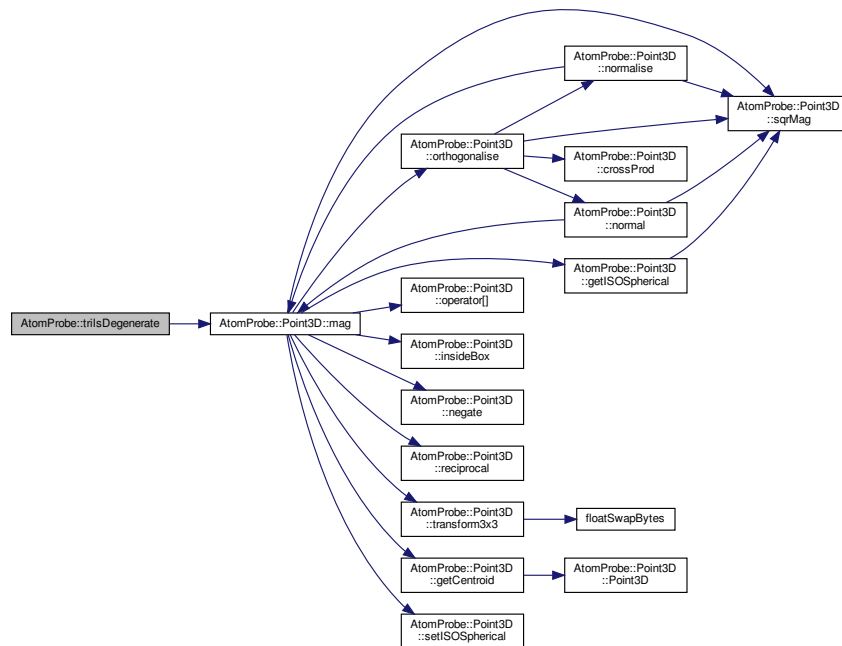
```
bool AtomProbe::triIsDegenerate (
    const Point3D & fA,
    const Point3D & fB,
    const Point3D & fC )
```

Definition at line 215 of file misc.cpp.

References AtomProbe::Point3D::mag().

Referenced by dotProduct(), and GetReducedHullPts().

Here is the call graph for this function:



5.1.3.156 ucharToHexStr()

```
void AtomProbe::ucharToHexStr (
    unsigned char c,
    std::string & s )
```

Definition at line 68 of file stringFuncs.cpp.

Referenced by genColString().

5.1.3.157 uppercase()

```
std::string AtomProbe::uppercase (
    std::string s )
```

Return a uppercase version for a given string.

Referenced by strAppend().

5.1.3.158 vectorMultiErase()

```
template<class T >
void AtomProbe::vectorMultiErase (
    std::vector< T > & vec,
    const std::vector< bool > & wantKill )
```

Definition at line 102 of file misc.h.

Referenced by `computelonDistAdjacency()`, `AtomProbe::RangeFile::decomposelonNames()`, `AtomProbe::RangeFile::eraselon()`, `filterBySolutionPPM()`, `filterPeakNeedBiggerObs()`, `findOverlaps()`, `AtomProbe::AbundanceData::generateIsotopeDist()`, `AtomProbe::RangeFile::makeSelfConsistent()`, and `FixedStack< T >::~~FixedStack()`.

5.1.3.159 vectorPointDir()

```
unsigned int AtomProbe::vectorPointDir (
    const Point3D & pA,
    const Point3D & pB,
    const Point3D & vC,
    const Point3D & vD )
```

Check which way vectors attached to two 3D points "point".

Two vectors may point "together", /__\ "apart" __/ or "In common" /__/ or __\

Definition at line 76 of file misc.cpp.

References `ASSERT`, `POINTDIR_APART`, `POINTDIR_IN_COMMON`, and `POINTDIR_TOGETHER`.

Referenced by `distanceToSegment()`, and `meanAndStdev()`.

5.1.3.160 weightedMean()

```
template<class T >
T AtomProbe::weightedMean (
    const std::vector< T > & values,
    const std::vector< T > & weight )
```

Definition at line 31 of file misc.h.

Referenced by `AtomProbe::AbundanceData::generateGroupedIsotopeDist()`.

5.1.3.161 XMLFreeDoc()

```
void AtomProbe::XMLFreeDoc (
    void * data )
```

Free a xmlDoc pointer. For use in conjunction with `std::unique_ptr` for auto-deallocation.

Definition at line 23 of file XMLHelper.cpp.

Referenced by `AtomProbe::MultiRange::open()`.

5.1.3.162 XMLGetNextElemAttrib()

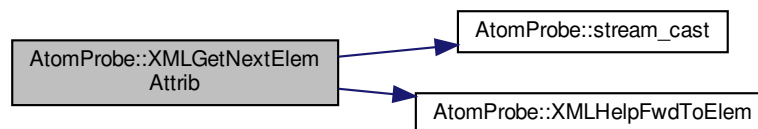
```
template<class T >
bool AtomProbe::XMLGetNextElemAttrib (
    xmlNodePtr & nodePtr,
    T & v,
    const char * nodeName,
    const char * attrib )
```

Grab the specified attribute from the next element, then `stream_cast()` it into the passed-in object. Returns true on success, false on error.

Definition at line 112 of file XMLHelper.h.

References `stream_cast()`, and `XMLHelpFwdToElem()`.

Here is the call graph for this function:

**5.1.3.163 XMLHelpFwdNotElem()**

```
unsigned int AtomProbe::XMLHelpFwdNotElem (
    xmlNodePtr & node,
    const char * nodeName )
```

Definition at line 53 of file XMLHelper.cpp.

5.1.3.164 XMLHelpFwdToElem()

```
unsigned int AtomProbe::XMLHelpFwdToElem (
    xmlNodePtr & node,
    const char * nodeName )
```

Definition at line 41 of file XMLHelper.cpp.

Referenced by AtomProbe::AbundanceData::open(), AtomProbe::MultiRange::open(), and XMLGetNextElem←Attrib().

5.1.3.165 XMLHelpFwdToList()

```
unsigned int AtomProbe::XMLHelpFwdToList (
    xmlNodePtr & node,
    const std::vector< std::string > & nodeList )
```

5.1.3.166 XMLHelpGetProp()

```
template<class T >
unsigned int AtomProbe::XMLHelpGetProp (
    T & prop,
    xmlNodePtr node,
    std::string propName )
```

Definition at line 87 of file XMLHelper.h.

References PROP_BAD_ATT, PROP_PARSE_ERR, and stream_cast().

Referenced by AtomProbe::AbundanceData::open(), and AtomProbe::MultiRange::open().

Here is the call graph for this function:



5.1.3.167 XMLHelpGetText() [1/2]

```
string AtomProbe::XMLHelpGetText (
    xmlNodePtr node )
```

Definition at line 64 of file XMLHelper.cpp.

References XMLHelpNextType().

Here is the call graph for this function:

**5.1.3.168 XMLHelpGetText()** [2/2]

```
std::string AtomProbe::XMLHelpGetText (
    xmlNodePtr & node )
```

5.1.3.169 XMLHelpNextType()

```
unsigned int AtomProbe::XMLHelpNextType (
    xmlNodePtr & node,
    int nodeType )
```

Definition at line 28 of file XMLHelper.cpp.

Referenced by XMLHelpGetText().

5.1.3.170 zechConfidenceLimits()

```
bool AtomProbe::zechConfidenceLimits (
    float lambdaBack,
    unsigned int observation,
    float alpha,
    float & estimate )
```

Provides a best estimate for true signal when true signal, background.

Nuclear Instruments and Methods in Physics Research A, 1989 608-610 "Upper limits in Experiments with Background or Measurement Errors" This is numerically unstable for large counts A similar problem is tackled by Bityukov, arxiv:hep-ex/0108020v1 Equation 3.5, but is less numerically stable

lambdaBack : background rate observation : Observed signal+background alpha : confidence parameter, (0,1) estimate : estimate for signal at confidence bound, specified by alpha Returns true if calculation succeeds, false otherwise. Note :this involves a root-finding approach, so may be slow

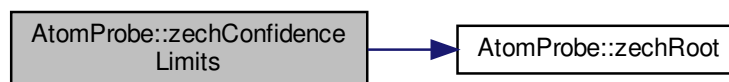
NOTE: may not be numerically stable for high lambdaback/observation, or lambdaback>>observation. This may cause GSL to error out.

Definition at line 391 of file confidence.cpp.

References AtomProbe::ZECH_ROOT::alpha, ASSERT, AtomProbe::ZECH_ROOT::lambdaBack, AtomProbe::ZECH_ROOT::observation, and zechRoot().

Referenced by poissonConfidenceObservation(), and zechCorrect().

Here is the call graph for this function:



5.1.3.171 zechRoot()

```
double AtomProbe::zechRoot (
    double sigGuess,
    void * params )
```

Definition at line 371 of file confidence.cpp.

References AtomProbe::ZECH_ROOT::alpha, AtomProbe::ZECH_ROOT::lambdaBack, and AtomProbe::ZECH_ROOT::observation.

Referenced by zechConfidenceLimits().

5.1.4 Variable Documentation

5.1.4.1 a2iTriangleConnectionTable

```
int AtomProbe::a2iTriangleConnectionTable[256][16]
```

Definition at line 194 of file isoSurface.cpp.

5.1.4.2 ABUNDANCE_ERROR

```
const char* AtomProbe::ABUNDANCE_ERROR[ ]
```

Initial value:

```
= { "Unable to read abundance data (opening file)",
    "Unable to create XML reader.",
    "Bad property found in XML file",
    "XML document did not match expected layout (DTD validation)",
    "Unable to find required node during parse",
    "Root node missing, expect <atomic-mass-table>!",
    "Found incorrect root node. Expected <atomic-mass-table>"
}
```

Definition at line 49 of file abundance.cpp.

5.1.4.3 aiCubeEdgeFlags

```
int AtomProbe::aiCubeEdgeFlags[256]
```

Initial value:

```
=
{
    0x000, 0x109, 0x203, 0x30a, 0x406, 0x50f, 0x605, 0x70c, 0x80c, 0x905, 0xa0f, 0xb06, 0xc0a, 0xd03,
    0xe09, 0xf00,
    0x190, 0x099, 0x393, 0x29a, 0x596, 0x49f, 0x795, 0x69c, 0x99c, 0x895, 0xb9f, 0xa96, 0xd9a, 0xc93,
    0xf99, 0xe90,
    0x230, 0x339, 0x033, 0x13a, 0x636, 0x73f, 0x435, 0x53c, 0xa3c, 0xb35, 0x83f, 0x936, 0xe3a, 0xf33,
    0xc39, 0xd30,
    0x3a0, 0x2a9, 0x1a3, 0x0aa, 0x7a6, 0x6af, 0x5a5, 0x4ac, 0xbac, 0xaa5, 0x9af, 0x8a6, 0xfaa, 0xea3,
    0xda9, 0xca0,
    0x460, 0x569, 0x663, 0x76a, 0x066, 0x16f, 0x265, 0x36c, 0xc6c, 0xd65, 0xe6f, 0xf66, 0x86a, 0x963,
    0xa69, 0xb60,
    0x5f0, 0x4f9, 0x7f3, 0x6fa, 0x1f6, 0x0ff, 0x3f5, 0x2fc, 0xdfc, 0xcf5, 0xfff, 0xef6, 0x9fa, 0x8f3,
    0xbf9, 0xaf0,
    0x650, 0x759, 0x453, 0x55a, 0x256, 0x35f, 0x055, 0x15c, 0xe5c, 0xf55, 0xc5f, 0xd56, 0xa5a, 0xb53,
    0x859, 0x950,
    0x7c0, 0x6c9, 0x5c3, 0x4ca, 0x3c6, 0x2cf, 0x1c5, 0x0cc, 0xfcc, 0xec5, 0xdcf, 0xcc6, 0xbca, 0xac3,
    0x9c9, 0x8c0,
    0x8c0, 0x9c9, 0xac3, 0xbca, 0xcc6, 0xdcf, 0xec5, 0xfcc, 0x0cc, 0x1c5, 0x2cf, 0x3c6, 0x4ca, 0x5c3,
    0x6c9, 0x7c0,
    0x950, 0x859, 0xb53, 0xa5a, 0xd56, 0xc5f, 0xf55, 0xe5c, 0x15c, 0x055, 0x35f, 0x256, 0x55a, 0x453,
    0x759, 0x650,
    0xaf0, 0xbf9, 0x8f3, 0x9fa, 0xef6, 0xfff, 0xcf5, 0xdfc, 0x2fc, 0x3f5, 0x0ff, 0x1f6, 0x6fa, 0x7f3,
    0x4f9, 0x5f0,
    0xb60, 0xa69, 0x963, 0x86a, 0xf66, 0xe6f, 0xd65, 0xc6c, 0x36c, 0x265, 0x16f, 0x066, 0x76a, 0x663,
    0x569, 0x460,
    0xca0, 0xda9, 0xea3, 0xfaa, 0x8a6, 0x9af, 0xaa5, 0xbac, 0x4ac, 0x5a5, 0x6af, 0x7a6, 0x0aa, 0x1a3,
    0x2a9, 0x3a0,
    0xd30, 0xc39, 0xf33, 0xe3a, 0x936, 0x83f, 0xb35, 0xa3c, 0x53c, 0x435, 0x73f, 0x636, 0x13a, 0x033,
    0x339, 0x230,
    0xe90, 0xf99, 0xc93, 0xd9a, 0xa96, 0xb9f, 0x895, 0x99c, 0x69c, 0x795, 0x49f, 0x596, 0x29a, 0x393,
    0x099, 0x190,
    0xf00, 0xe09, 0xd03, 0xc0a, 0xb06, 0xa0f, 0x905, 0x80c, 0x70c, 0x605, 0x50f, 0x406, 0x30a, 0x203,
    0x109, 0x000
}
```

Definition at line 166 of file isoSurface.cpp.

5.1.4.4 ATO_ERR_STRINGS

```
const char * AtomProbe::ATO_ERR_STRINGS
```

Initial value:

```
= { "",  
    "Error opening file",  
    "File is empty",  
    "Filesize does not match expected format",  
    "File version number not <4, as expected",  
    "Unable to allocate memory to store data",  
    "Unable to detect endian-ness in file"  
}
```

Human readable error messages for use with ATO reader return values.

Definition at line 73 of file dataFiles.cpp.

5.1.4.5 edgeRemap

```
int AtomProbe::edgeRemap[12]
```

Initial value:

```
={ 0,6,1,4,  
    2,7,3,5,  
    8,10,11,9}
```

Definition at line 488 of file isoSurface.cpp.

5.1.4.6 ELEM_FOUR_NODE_TETRAHEDRON

```
const unsigned int AtomProbe::ELEM_FOUR_NODE_TETRAHEDRON =4
```

Definition at line 38 of file mesh.h.

Referenced by AtomProbe::Mesh::loadGmshMesh(), and AtomProbe::Mesh::saveGmshMesh().

5.1.4.7 ELEM_SINGLE_NODE_POINT

```
const unsigned int AtomProbe::ELEM_SINGLE_NODE_POINT =15
```

Definition at line 35 of file mesh.h.

Referenced by AtomProbe::Mesh::loadGmshMesh(), and AtomProbe::Mesh::saveGmshMesh().

5.1.4.8 ELEM_THREE_NODE_TRIANGLE

```
const unsigned int AtomProbe::ELEM_THREE_NODE_TRIANGLE =2
```

Definition at line 37 of file mesh.h.

Referenced by AtomProbe::Mesh::loadGmshMesh(), and AtomProbe::Mesh::saveGmshMesh().

5.1.4.9 ELEM_TWO_NODE_LINE

```
const unsigned int AtomProbe::ELEM_TWO_NODE_LINE =1
```

Definition at line 36 of file mesh.h.

Referenced by AtomProbe::Mesh::loadGmshMesh(), and AtomProbe::Mesh::saveGmshMesh().

5.1.4.10 elementList

```
const char* AtomProbe::elementList[]
```

Initial value:

```
= {
"H", "He", "Li", "Be", "B", "C", "N", "O", "F", "Ne",
"Na", "Mg", "Al", "Si", "P", "S", "Cl", "Ar",
"K", "Ca", "Sc", "Ti", "V", "Cr", "Mn", "Fe", "Co", "Ni",
"Cu", "Zn", "Ga", "Ge", "As", "Se", "Br", "Kr",
"Rb", "Sr", "Y", "Zr", "Nb", "Mo", "Tc", "Ru", "Rh", "Pd",
"Ag", "Cd", "In", "Sn", "Sb", "Te", "I", "Xe",
"Cs", "Ba", "La", "Ce", "Pr", "Nd", "Pm", "Sm", "Eu", "Gd",
"Tb", "Dy", "Ho", "Er", "Tm", "Yb", "Lu", "Hf", "Ta",
"W", "Re", "Os", "Ir", "Pt", "Au", "Hg", "Tl", "Pb",
"Bi", "Po", "At", "Rn",
"Fr", "Ra", "Ac", "Th", "Pa", "U", "Np", "Pu", "Am", "Cm",
"Bk", "Cf", "Es", "Fm", "Md", "No", "Lr", "Rf", "Db",
"Sg", "Bh", "Hs", "Mt", "Ds", "Rg", "Cn", "Uut", "Fl",
"Uup", "Lv", "Uus", "Uuo", ""
}
```

Definition at line 62 of file ranges.cpp.

5.1.4.11 EPOS_RECORD_SIZE

```
const size_t AtomProbe::EPOS_RECORD_SIZE = 11*4
```

Definition at line 139 of file ionHit.h.

Referenced by chunkLoadEposFile(), and loadEposFile().

5.1.4.12 hardAssert

```
bool AtomProbe::hardAssert =true
```

Definition at line 23 of file aptAssert.cpp.

Referenced by getHardAssert().

5.1.4.13 HULL_GRAB_SIZE

```
const unsigned int AtomProbe::HULL_GRAB_SIZE =4096
```

Definition at line 42 of file convexHull.cpp.

5.1.4.14 libVersion

```
LibVersion AtomProbe::libVersion
```

Definition at line 28 of file atomprobe.cpp.

Referenced by AtomProbe::RandNumGen::getRng().

5.1.4.15 MAX_LINE_SIZE

```
const size_t AtomProbe::MAX_LINE_SIZE = 16536
```

Definition at line 50 of file ranges.cpp.

Referenced by AtomProbe::RangeFile::open(), and AtomProbe::RangeFile::rangeTypeString().

5.1.4.16 MAX_RANGEFILE_SIZE

```
const size_t AtomProbe::MAX_RANGEFILE_SIZE = 50*1024*1024
```

Definition at line 52 of file ranges.cpp.

5.1.4.17 maximumLinearTable

```
const size_t AtomProbe::maximumLinearTable[ ]
```

Definition at line 22 of file lfsr.cpp.

Referenced by AtomProbe::LinearFeedbackShiftReg::verifyTable().

5.1.4.18 MESH_LOAD_ERRS

```
const char * AtomProbe::MESH_LOAD_ERRS
```

Initial value:

```
= { "",  
  "Missing error message. This is a bug, please report it",  
  "Node count was different to number of present nodes",  
  "Element count was less than number of present elements",  
  "Mesh loaded, but failed to pass sanity checks"  
}
```

Definition at line 53 of file mesh.cpp.

5.1.4.19 MULTIRANGE_FORMAT_VERSION

```
const char AtomProbe::MULTIRANGE_FORMAT_VERSION[ ] = "0.0.1"
```

Definition at line 46 of file multiRange.cpp.

5.1.4.20 NUM_ELEMENTS

```
const unsigned int AtomProbe::NUM_ELEMENTS =119
```

Definition at line 34 of file ranges.h.

5.1.4.21 OPS_ENUM_ERRSTRINGS

```
const char * AtomProbe::OPS_ENUM_ERRSTRINGS
```

Initial value:

```
=
{
    "",
    "\"C\" line (exp. parameters) not in expected format",
    "\"I\" line (Detector parameters) not in expected format",
    "\"-\" line (Time-of-flights) not in expected format",
    "\"V\" line (voltage data) not in expected format",
    "Error interpreting \"V\" line (voltage data) data",
    "Missing beta value from \"V\" line - needs to be in voltage, or in system header (\"C\" line",
    "Incorrect number of \"V\" line (voltage data) data entries",
    "Unknown linetype in data file",
    "\"P\" line (detector channels) not in expected format",
    "Unable to interpret channels line",
    "Incorrect number of events in \"S\" (hit position) line",
    "Unable to interpret event data on \"S\" (hit position) line",
    "Unable to parse \"S\" (hit position) line event count (eg 8 in S8 ...)",
    "\"S\" (hit position) line data not preceded by TOF data, as it should have been",
    "Duplicate system data/experiment setup (\"C\") entry found -- there can only be one",
    "Duplicate detector (\"I\") data entry found -- there can only be one",
    "Trailing \"-\" line found -- should have be followed by a \"S\" line, but wasn't.",
    "Duplicate\"-\" line found -- should have be followed by a \"S\" line, but wasn't.",
    "Unable to open file",
    "unable to read file, after opening",
    "Abort requested"
}
```

Definition at line 768 of file dataFiles.cpp.

5.1.4.22 PROGRESS_REDUCE

```
const size_t AtomProbe::PROGRESS_REDUCE =500
```

Definition at line 61 of file mesh.cpp.

Referenced by AtomProbe::K3DTreeExact::build(), computeConvexHull(), and AtomProbe::Mesh::pointsInside().

5.1.4.23 qhullInited

```
bool AtomProbe::qhullInited =false
```

Definition at line 39 of file convexHull.cpp.

5.1.4.24 randGen

```
RandNumGen AtomProbe::randGen
```

Definition at line 29 of file atomprobe.cpp.

Referenced by AtomProbe::RandNumGen::getRng(), leastSquaresDeconvolve(), loadATOFfile(), loadPosFile(), main(), poissonConfidenceObservation(), and sampleIons().

5.1.4.25 RANGE_EXTS

```
const char* AtomProbe::RANGE_EXTS[ ]
```

Initial value:

```
= { "rng",
    "env",
    "rng",
    "rrng",
    "" }
```

Definition at line 54 of file ranges.cpp.

5.1.4.26 RECORDREAD_ERR_STRINGS

```
const char * AtomProbe::RECORDREAD_ERR_STRINGS
```

Initial value:

```
= { "",
    "Unable to determine filesize",
    "Filesize indicates that file contains a non-integer number of entries",
    "Unable to open file",
    "Unable to allocate memory for reading file contents",
    "Unable to perform read operation on file",
    "Read past end of file requested",
    "Entry in file appears to be invalid",
}
```

Definition at line 82 of file dataFiles.cpp.

Referenced by main().

5.1.4.27 VERTEX_OFFSET

```
const unsigned int AtomProbe::VERTEX_OFFSET[8][3]
```

Initial value:

```
=
{
    {0, 0, 0}, {1, 0, 0}, {1, 1, 0}, {0, 1, 0},
    {0, 0, 1}, {1, 0, 1}, {1, 1, 1}, {0, 1, 1}
}
```

Definition at line 462 of file isoSurface.cpp.

Chapter 6

Class Documentation

6.1 AtomProbe::AbundanceData Class Reference

Class to load abundance information for natural isotopes.

```
#include <abundance.h>
```

Public Member Functions

- `size_t open` (const char *file, bool strict=false)
Attempt to open the abundance data file, return 0 on success, nonzero on failure.
- `size_t numIsotopes` () const
Return the number of isotopes in the loaded table.
- `size_t numElements` () const
Return the number of elements in the loaded table.
- `size_t symbolIndex` (const char *symbol, bool caseSensitive=true) const
Return the element's position in table, starting from 0.
- `void getSymbolIndices` (const std::vector< std::string > &symbols, std::vector< size_t > &indices) const
Return a vector of symbol indices.
- `void getSymbolIndices` (const std::vector< std::pair< std::string, size_t > > &symbols, std::vector< std::pair< size_t, unsigned int > > &indices) const
Obtain the symbol indices for a string,payload type. Resultant pair has first item in index, second is payload.
- `std::string elementName` (size_t elemIdx) const
Obtain the name of an element from its index.
- `size_t symbolIdxFromAtomicNumber` (size_t atomicNumber) const
Return the symbol index (offset in vector) for the given atomic number, -1 if it cannot be found.
- `std::string elementNames` (size_t start, size_t end, char separator=',') const
Obtain a delimited list of element names, using the (optional) supplied separator, over the specified [start,end) range.
- `void isotopeIndex` (size_t elem, float mass, size_t &isotopIdx) const
Obtain the atom ID (first) and the isotope value(second).
- `void isotopeIndex` (size_t elem, float mass, float tolerance, size_t &isotopIdx) const
Return the first matching element in the table which has a given tolerance to the specified match.
- `unsigned int getAtomicNumber` (size_t elemIdx) const
Obtain the atomic number for the given element, by element index.
- `const std::vector< ISOTOPE_ENTRY > & isotopes` (size_t offset) const

Return the isotope at a given position in the table. Offset must exist.

- void [generateIsotopeDist](#) (const std::vector< size_t > &elementIdx, const std::vector< size_t > &frequency, std::vector< std::pair< float, float > > &massDist) const

Compute the mass-probability distribution for the grouped set of ions.

- void [generateGroupedIsotopeDist](#) (const std::vector< size_t > &elementIdx, const std::vector< size_t > &frequency, std::vector< std::pair< float, float > > &massDist, float massTolerance) const

As per generateIsotopeDist, however, this convenience groups the distribution to limit the effect of minute mass changes.

- void [generateSingleAtomDist](#) (size_t atomIdx, unsigned int repeatCount, std::vector< std::pair< float, float > > &massDist) const

Obtain the mass distribution from a single isotope that repeats. Output is not grouped.

- float [abundanceBetweenLimits](#) (const std::vector< size_t > &elemIdx, const std::vector< size_t > &frequency, float massStart, float massEnd) const

Obtain the fractional abundance between two limits for the given species [start,end)

- const [ISOTOPE_ENTRY](#) & [isotope](#) (size_t elementIdx, size_t isotopIdx) const

Obtain a reference to a particular isotope, using the element's index in the table, and the isotope index.

- size_t [getMajorIsotopeFromElemIdx](#) (size_t elementIdx) const

Obtain the most prominent isotope's index from the element index.

- float [getNominalMass](#) (size_t elementIdx) const

Obtain the isotope weighted mass for the given element.

- size_t [getNearestCharge](#) (const std::vector< std::pair< size_t, size_t > > &molecule, float targetMass, size_t maxCharge) const

Obtain the closest charge state for the given mass and species group.

Static Public Member Functions

- static const char * [getErrorText](#) (size_t errorCode)

Obtain a human-readable error message from the [open\(\)](#) call.

6.1.1 Detailed Description

Class to load abundance information for natural isotopes.

Definition at line 54 of file abundance.h.

6.1.2 Member Function Documentation

6.1.2.1 abundanceBetweenLimits()

```
float AtomProbe::AbundanceData::abundanceBetweenLimits (
    const std::vector< size_t > & elemIdx,
    const std::vector< size_t > & frequency,
    float massStart,
    float massEnd ) const
```

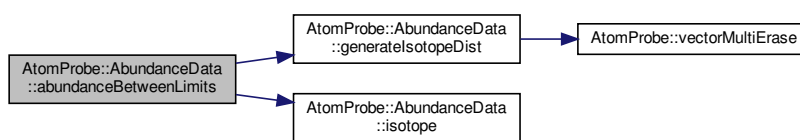
Obtain the fractional abundance between two limits for the given species [start,end)

Definition at line 455 of file abundance.cpp.

References ASSERT, generateIsotopeDist(), and isotope().

Referenced by AtomProbe::leastSquaresDeconvolve().

Here is the call graph for this function:



6.1.2.2 elementName()

```
std::string AtomProbe::AbundanceData::elementName (
    size_t elemIdx ) const [inline]
```

Obtain the name of an element from its index.

Definition at line 107 of file abundance.h.

References `AtomProbe::ISOTOPE_ENTRY::atomicNumber`, and `AtomProbe::ISOTOPE_ENTRY::mass`.

Referenced by `elementNames()`.

6.1.2.3 elementNames()

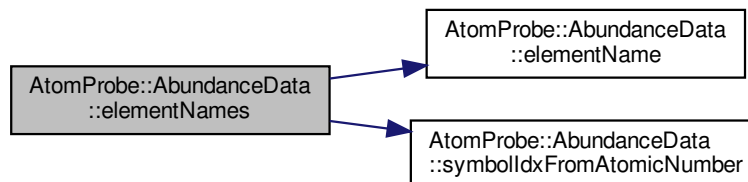
```
std::string AtomProbe::AbundanceData::elementNames (
    size_t start,
    size_t end,
    char separator = ',' ) const
```

Obtain a delimited list of element names, using the (optional) supplied separator, over the specified [start,end) range.

Definition at line 585 of file abundance.cpp.

References ASSERT, elementName(), and symbolIdxFromAtomicNumber().

Here is the call graph for this function:



6.1.2.4 generateGroupedIsotopeDist()

```
void AtomProbe::AbundanceData::generateGroupedIsotopeDist (
    const std::vector< size_t > & elementIdx,
    const std::vector< size_t > & frequency,
    std::vector< std::pair< float, float > > & massDist,
    float massTolerance ) const
```

As per `generatelsotopeDist`, however, this convenience groups the distribution to limit the effect of minute mass changes.

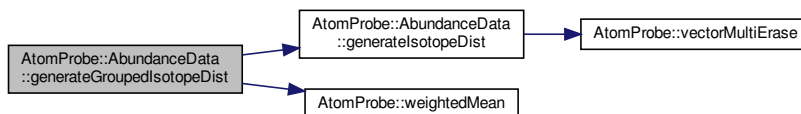
This groups the masses of nearby isotope combinations together. This is desirable due to species, such as Mo2, which have isotopes within <0.001 amu of one another. The mass centres are computed using a weighted average, and the intensities are summed

Definition at line 387 of file abundance.cpp.

References `generatelsotopeDist()`, and `AtomProbe::weightedMean()`.

Referenced by `AtomProbe::findOverlaps()`, `getMassDistributions()`, `getNearestCharge()`, and `AtomProbe::max←→ExplainedFraction()`.

Here is the call graph for this function:



6.1.2.5 generateIsotopeDist()

```

void AtomProbe::AbundanceData::generateIsotopeDist (
    const std::vector< size_t > & elementIdx,
    const std::vector< size_t > & frequency,
    std::vector< std::pair< float, float > > & massDist ) const
  
```

Compute the mass-probability distribution for the grouped set of ions.

E.g., for C2, this would be `elementIdx = elementIdx("C")`, and frequency 2. This would return probabilities for each unique mass {C-12 C-12},{C-12 C-13} and {C-13 C-13}.... note an internal "grouping" tolerance is used to group-together very close masses. This may be smaller than what is needed for your application, so if you wish to obtain a more specific distribution use "generateGroupedIsotopeDist"

Definition at line 277 of file abundance.cpp.

References ASSERT, and AtomProbe::vectorMultiErase().

Referenced by abundanceBetweenLimits(), AtomProbe::checkMassRangingCorrectness(), AtomProbe::computeIonDistAdjacency(), AtomProbe::filterPeakNeedBiggerObs(), generateGroupedIsotopeDist(), getNearestCharge(), and AtomProbe::RangeFile::guessChargeState().

Here is the call graph for this function:



6.1.2.6 generateSingleAtomDist()

```
void AtomProbe::AbundanceData::generateSingleAtomDist (
    size_t atomIdx,
    unsigned int repeatCount,
    std::vector< std::pair< float, float > > & massDist ) const
```

Obtain the mass distribution from a single isotope that repeats. Output is not grouped.

Definition at line 509 of file abundance.cpp.

Referenced by AtomProbe::MultiRange::copyDataFromRange().

6.1.2.7 getAtomicNumber()

```
unsigned int AtomProbe::AbundanceData::getAtomicNumber (
    size_t elemIdx ) const
```

Obtain the atomic number for the given element, by element index.

Definition at line 495 of file abundance.cpp.

References ASSERT.

Referenced by AtomProbe::computeRangeAdjacency(), AtomProbe::findOverlaps(), and AtomProbe::MultiRange↔::MultiRange().

6.1.2.8 getErrorText()

```
const char * AtomProbe::AbundanceData::getErrorText (
    size_t errorCode ) [static]
```

Obtain a human-readable error message from the [open\(\)](#) call.

Definition at line 59 of file abundance.cpp.

References ASSERT.

6.1.2.9 getMajorIsotopeFromElemIdx()

```
size_t AtomProbe::AbundanceData::getMajorIsotopeFromElemIdx (
    size_t elementIdx ) const
```

Obtain the most prominent isotope's index from the element index.

Definition at line 474 of file abundance.cpp.

References ASSERT.

Referenced by AtomProbe::maxExplainedFraction().

6.1.2.10 getNearestCharge()

```
size_t AtomProbe::AbundanceData::getNearestCharge (
    const std::vector< std::pair< size_t, size_t > > & molecule,
    float targetMass,
    size_t maxCharge ) const
```

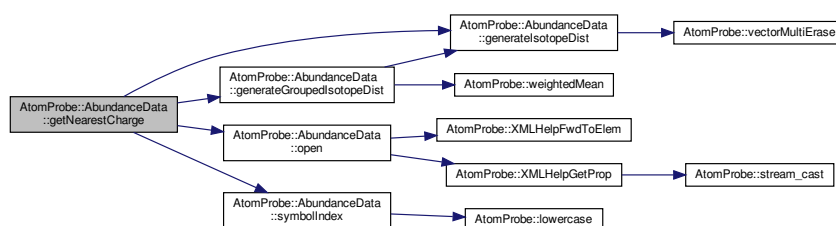
Obtain the closest charge state for the given mass and species group.

In the "molecule" parameter, the pair is <element index, count>, the target mass is given in Da, and the maximum charge is the maximum the function will search for

Definition at line 634 of file abundance.cpp.

References ASSERT, generateGroupedIsotopeDist(), generateIsotopeDist(), open(), symbolIndex(), and TEST.

Here is the call graph for this function:



6.1.2.11 getNominalMass()

```
float AtomProbe::AbundanceData::getNominalMass (
    size_t elementIdx ) const
```

Obtain the isotope weighted mass for the given element.

Definition at line 618 of file abundance.cpp.

Referenced by `AtomProbe::convertMassToMol()`, and `AtomProbe::convertMolToMass()`.

6.1.2.12 `getSymbolIndices()` [1/2]

```
void AtomProbe::AbundanceData::getSymbolIndices (
    const std::vector< std::string > & symbols,
    std::vector< size_t > & indices ) const
```

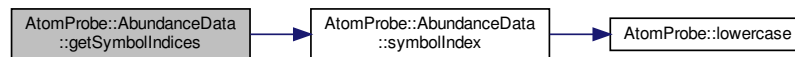
Return a vector of symbol indices.

Definition at line 551 of file abundance.cpp.

References `symbolIndex()`.

Referenced by `AtomProbe::MultiRange::copyDataFromRange()`, `AtomProbe::findOverlaps()`, `getMassDistributions()`, and `AtomProbe::MultiRange::MultiRange()`.

Here is the call graph for this function:



6.1.2.13 `getSymbolIndices()` [2/2]

```
void AtomProbe::AbundanceData::getSymbolIndices (
    const std::vector< std::pair< std::string, size_t > > & symbols,
    std::vector< std::pair< size_t, unsigned int > > & indices ) const
```

Obtain the symbol indices for a string,payload type. Resultant pair has first item in index, second is payload.

6.1.2.14 `isotope()`

```
const ISOTOPE_ENTRY & AtomProbe::AbundanceData::isotope (
    size_t elementIdx,
    size_t isotopeIdx ) const
```

Obtain a reference to a particular isotope, using the element's index in the table, and the isotope index.

Definition at line 271 of file abundance.cpp.

References `ASSERT`.

Referenced by `abundanceBetweenLimits()`, `AtomProbe::MultiRange::copyDataFromRange()`, `AtomProbe::filterBy←SolutionPPM()`, and `AtomProbe::maxExplainedFraction()`.

6.1.2.15 isotopeIndex() [1/2]

```
void AtomProbe::AbundanceData::isotopeIndex (
    size_t elem,
    float mass,
    size_t & isotopeIdx ) const
```

Obtain the atom ID (first) and the isotope value(second).

The mass and element ID must match exactly NOTE: The element and isotope indices are *NOT* its isotopic number it is the offset to that isotope int the abundance table

Definition at line 238 of file abundance.cpp.

References ASSERT.

6.1.2.16 isotopeIndex() [2/2]

```
void AtomProbe::AbundanceData::isotopeIndex (
    size_t elem,
    float mass,
    float tolerance,
    size_t & isotopeIdx ) const
```

Return the first matching element in the table which has a given tolerance to the specified match.

Definition at line 254 of file abundance.cpp.

References ASSERT.

6.1.2.17 isotopes()

```
const std::vector< ISOTOPE_ENTRY > & AtomProbe::AbundanceData::isotopes (
    size_t offset ) const
```

Return the isotope at a given position in the table. Offset *must* exist.

Definition at line 613 of file abundance.cpp.

References ASSERT.

Referenced by AtomProbe::leastSquaresDeconvolve().

6.1.2.18 numElements()

```
size_t AtomProbe::AbundanceData::numElements ( ) const
```

Return the number of elements in the loaded table.

Definition at line 74 of file abundance.cpp.

6.1.2.19 numIsotopes()

```
size_t AtomProbe::AbundanceData::numIsotopes ( ) const
```

Return the number of isotopes in the loaded table.

Definition at line 65 of file abundance.cpp.

6.1.2.20 open()

```
size_t AtomProbe::AbundanceData::open (
    const char * file,
    bool strict = false )
```

Attempt to open the abundance data file, return 0 on success, nonzero on failure.

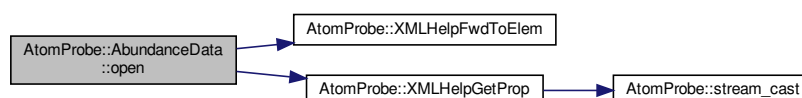
If nonzero, a human readable message can be obtained from `getErrorText(...)`

Definition at line 79 of file abundance.cpp.

References `AtomProbe::ISOTOPE_ENTRY::abundance`, `AtomProbe::ISOTOPE_ENTRY::abundanceError`, `AtomProbe::ISOTOPE_ENTRY::atomicNumber`, `AtomProbe::ISOTOPE_ENTRY::mass`, `AtomProbe::ISOTOPE_ENTRY::massError`, `AtomProbe::ISOTOPE_ENTRY::massNumber`, `AtomProbe::ISOTOPE_ENTRY::symbol`, `AtomProbe::XMLHelpFwdToElem()`, and `AtomProbe::XMLHelpGetProp()`.

Referenced by `AtomProbe::checkMassRangingCorrectness()`, `AtomProbe::computeRangeAdjacency()`, `AtomProbe::findOverlaps()`, `getNearestCharge()`, `AtomProbe::RangeFile::guessChargeState()`, `AtomProbe::leastSquaresDeconvolve()`, `main()`, `AtomProbe::maxExplainedFraction()`, `AtomProbe::parseCompositionData()`, and `runTests()`.

Here is the call graph for this function:



6.1.2.21 symbolIdxFromAtomicNumber()

```
size_t AtomProbe::AbundanceData::symbolIdxFromAtomicNumber (
    size_t atomicNumber ) const
```

Return the symbol index (offset in vector) for the given atomic number, -1 if it cannot be found.

Definition at line 601 of file abundance.cpp.

Referenced by AtomProbe::computeIonDistAdjacency(), elementNames(), and AtomProbe::leastSquaresDeconvolve().

6.1.2.22 symbolIndex()

```
size_t AtomProbe::AbundanceData::symbolIndex (
    const char * symbol,
    bool caseSensitive = true ) const
```

Return the element's position in table, starting from 0.

Notes

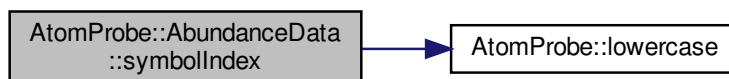
- Input is case insensitive.
- returns (unsigned) -1 if the symbol cannot be found

Definition at line 214 of file abundance.cpp.

References ASSERT, and AtomProbe::lowercase().

Referenced by AtomProbe::computeRangeAdjacency(), AtomProbe::MultiRange::copyDataFromRange(), getNearestCharge(), AtomProbe::getRangeMolecule(), getSymbolIndices(), AtomProbe::RangeFile::guessChargeState(), AtomProbe::leastSquaresDeconvolve(), AtomProbe::maxExplainedFraction(), and AtomProbe::parseCompositionData().

Here is the call graph for this function:



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

- include/atomprobe/isotopes/abundance.h
- src/isotopes/abundance.cpp

6.2 AtomProbe::ATO_ENTRY Struct Reference

```
#include <dataFiles.h>
```

Public Attributes

- float [x](#)
- float [y](#)
- float [z](#)
- float [mass](#)
- float [approxPulse](#)
- float [voltage](#)
- float [tof](#)
- float [detectorX](#)
- float [detectorY](#)
- float [pulseVoltage](#)

6.2.1 Detailed Description

Definition at line 67 of file dataFiles.h.

6.2.2 Member Data Documentation

6.2.2.1 approxPulse

```
float AtomProbe::ATO_ENTRY::approxPulse
```

Definition at line 72 of file dataFiles.h.

6.2.2.2 detectorX

```
float AtomProbe::ATO_ENTRY::detectorX
```

Definition at line 75 of file dataFiles.h.

6.2.2.3 detectorY

```
float AtomProbe::ATO_ENTRY::detectorY
```

Definition at line 75 of file dataFiles.h.

6.2.2.4 mass

```
float AtomProbe::ATO_ENTRY::mass
```

Definition at line 70 of file dataFiles.h.

Referenced by AtomProbe::getAtoErrString().

6.2.2.5 pulseVoltage

```
float AtomProbe::ATO_ENTRY::pulseVoltage
```

Definition at line 76 of file dataFiles.h.

6.2.2.6 tof

```
float AtomProbe::ATO_ENTRY::tof
```

Definition at line 74 of file dataFiles.h.

6.2.2.7 voltage

```
float AtomProbe::ATO_ENTRY::voltage
```

Definition at line 73 of file dataFiles.h.

6.2.2.8 x

```
float AtomProbe::ATO_ENTRY::x
```

Definition at line 69 of file dataFiles.h.

6.2.2.9 y

```
float AtomProbe::ATO_ENTRY::y
```

Definition at line 69 of file dataFiles.h.

6.2.2.10 z

```
float AtomProbe::ATO_ENTRY::z
```

Definition at line 69 of file dataFiles.h.

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

- [include/atomprobe/io/dataFiles.h](#)

6.3 AtomProbe::AxisCompare Class Reference

Functor allowing for sorting of points in 3D.

```
#include <K3DTree-approx.h>
```

Public Member Functions

- [AxisCompare](#) ()
- void [setAxis](#) (unsigned int Axis)
- bool [operator\(\)](#) (const [Point3D](#) &p1, const [Point3D](#) &p2) const

6.3.1 Detailed Description

Functor allowing for sorting of points in 3D.

Used by KD Tree to sort points based around which splitting axis is being used once the axis is set, points will be ranked based upon their relative value in that axis only

Definition at line 41 of file K3DTree-approx.h.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 AxisCompare()

```
AtomProbe::AxisCompare::AxisCompare ( )
```

Definition at line 40 of file K3DTree-approx.cpp.

6.3.3 Member Function Documentation

6.3.3.1 operator()

```
bool AtomProbe::AxisCompare::operator() (
    const Point3D & p1,
    const Point3D & p2 ) const [inline]
```

Definition at line 48 of file K3DTree-approx.h.

6.3.3.2 setAxis()

```
void AtomProbe::AxisCompare::setAxis (
    unsigned int Axis )
```

Definition at line 44 of file K3DTree-approx.cpp.

Referenced by AtomProbe::K3DTreeApprox::buildByRef().

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

- [include/atomprobe/algorithm/K3DTree-approx.h](#)
- [src/algorithm/K3DTree-approx.cpp](#)

6.4 AxisCompareExact Class Reference

Functor allowing for sorting of points in 3D.

Public Member Functions

- void [setAxis](#) (unsigned int Axis)
- bool [operator\(\)](#) (const std::pair< [Point3D](#), size_t > &p1, const std::pair< [Point3D](#), size_t > &p2) const

6.4.1 Detailed Description

Functor allowing for sorting of points in 3D.

Used by KD Tree to sort points based around which splitting axis is being used once the axis is set, points will be ranked based upon their relative value in that axis only

Definition at line 45 of file K3DTree-exact.cpp.

6.4.2 Member Function Documentation

6.4.2.1 operator()

```
bool AxisCompareExact::operator() (
    const std::pair< Point3D, size_t > & p1,
    const std::pair< Point3D, size_t > & p2 ) const [inline]
```

Definition at line 51 of file K3DTree-exact.cpp.

6.4.2.2 setAxis()

```
void AxisCompareExact::setAxis (
    unsigned int Axis ) [inline]
```

Definition at line 50 of file K3DTree-exact.cpp.

Referenced by AtomProbe::K3DTreeExact::build().

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

- [src/algorithm/K3DTree-exact.cpp](#)

6.5 AtomProbe::BACKGROUND_PARAMS Struct Reference

```
#include <processing.h>
```

Public Types

- enum {
[FIT_FAIL_MIN_REQ_BINS =1](#), [FIT_FAIL_AVG_COUNTS](#), [FIT_FAIL_INSUFF_DATA](#), [FIT_FAIL_DATA_N<←](#)
[ON_GAUSSIAN](#),
[FIT_FAIL_END](#) }

Public Attributes

- unsigned int [mode](#)
- float [massStart](#)
- float [massEnd](#)
- float [binWidth](#)
- float [intensity](#)
- float [stdev](#)

6.5.1 Detailed Description

Definition at line 28 of file processing.h.

6.5.2 Member Enumeration Documentation

6.5.2.1 anonymous enum

```
anonymous enum
```

Enumerator

FIT_FAIL_MIN_REQ_BINS	
FIT_FAIL_AVG_COUNTS	
FIT_FAIL_INSUFF_DATA	
FIT_FAIL_DATA_NON_GAUSSIAN	
FIT_FAIL_END	

Definition at line 30 of file processing.h.

6.5.3 Member Data Documentation

6.5.3.1 binWidth

```
float AtomProbe::BACKGROUND_PARAMS::binWidth
```

Definition at line 43 of file processing.h.

Referenced by AtomProbe::doFitBackground().

6.5.3.2 intensity

```
float AtomProbe::BACKGROUND_PARAMS::intensity
```

Definition at line 46 of file processing.h.

Referenced by AtomProbe::doFitBackground().

6.5.3.3 massEnd

```
float AtomProbe::BACKGROUND_PARAMS::massEnd
```

Definition at line 41 of file processing.h.

Referenced by AtomProbe::doFitBackground().

6.5.3.4 massStart

```
float AtomProbe::BACKGROUND_PARAMS::massStart
```

Definition at line 41 of file processing.h.

Referenced by AtomProbe::doFitBackground().

6.5.3.5 mode

```
unsigned int AtomProbe::BACKGROUND_PARAMS::mode
```

Definition at line 39 of file processing.h.

6.5.3.6 stdev

```
float AtomProbe::BACKGROUND_PARAMS::stdev
```

Definition at line 46 of file processing.h.

Referenced by AtomProbe::doFitBackground().

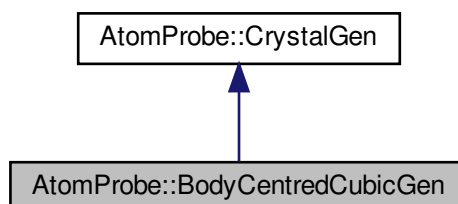
The documentation for this struct was generated from the following file:

- [include/atomprobe/spectrum/processing.h](#)

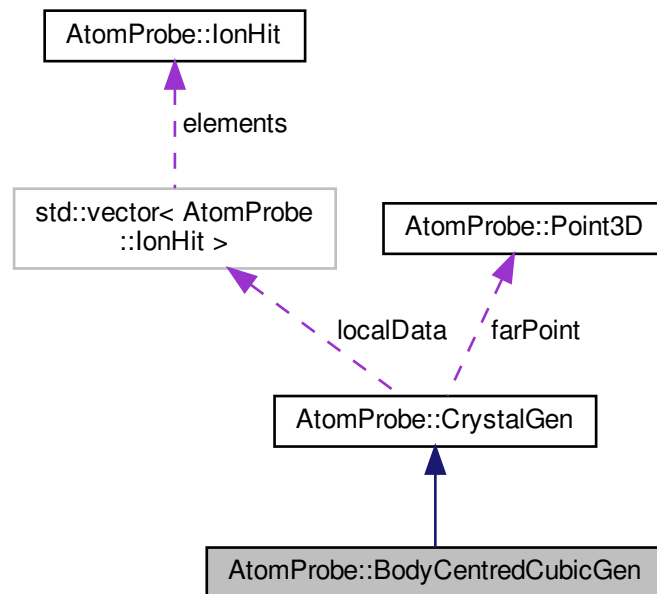
6.6 AtomProbe::BodyCentredCubicGen Class Reference

```
#include <generate.h>
```

Inheritance diagram for AtomProbe::BodyCentredCubicGen:



Collaboration diagram for AtomProbe::BodyCentredCubicGen:



Public Member Functions

- [BodyCentredCubicGen](#) (float latticeSpacing, float mToCCentre, const float mToCCorner, const [Point3D](#) &p)
- unsigned int [generateLattice](#) ()

Additional Inherited Members

6.6.1 Detailed Description

Definition at line 66 of file generate.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 BodyCentredCubicGen()

```

AtomProbe::BodyCentredCubicGen::BodyCentredCubicGen (
    float latticeSpacing,
    float mToCCentre,
    const float mToCCorner,
    const Point3D & p )
  
```

Definition at line 223 of file generate.cpp.

References [AtomProbe::CrystalGen::farPoint](#).

6.6.3 Member Function Documentation

6.6.3.1 generateLattice()

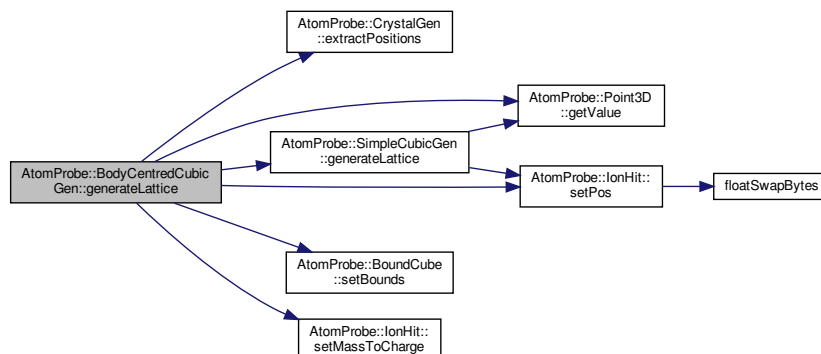
```
unsigned int AtomProbe::BodyCentredCubicGen::generateLattice ( ) [virtual]
```

Implements [AtomProbe::CrystalGen](#).

Definition at line 232 of file generate.cpp.

References [AtomProbe::CRYSTAL_BAD_UNIT_CELL](#), [AtomProbe::CrystalGen::extractPositions\(\)](#), [AtomProbe::CrystalGen::farPoint](#), [AtomProbe::SimpleCubicGen::generateLattice\(\)](#), [AtomProbe::Point3D::getValue\(\)](#), [AtomProbe::CrystalGen::localData](#), [AtomProbe::BoundCube::setBounds\(\)](#), [AtomProbe::IonHit::setMassToCharge\(\)](#), [AtomProbe::IonHit::setPos\(\)](#), and [TEST](#).

Here is the call graph for this function:



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

- [include/atomprobe/latticeplanes/generate.h](#)
- [src/lattice/generate.cpp](#)

6.7 AtomProbe::BoundCube Class Reference

Helper class to define a bounding cube.

```
#include <boundcube.h>
```

Public Member Functions

- [BoundCube](#) ()
- [BoundCube](#) (const std::vector< [Point3D](#) > &pts)
- [BoundCube](#) (const std::vector< [IonHit](#) > &pts)
- [BoundCube](#) (const [Point3D](#) &p1, const [Point3D](#) &p2)
- bool [operator==](#) (const [BoundCube](#) &other) const
- void [setBounds](#) (float xMin, float yMin, float zMin, float xMax, float yMax, float zMax)
 - Set the bounds by passing in minima and maxima of each dimension.*
- void [setBound](#) (unsigned int bound, unsigned int minMax, float value)
 - Set an individual bound.*
- void [setBounds](#) (const [BoundCube](#) &b)
 - Set the bounding cube to be the same as a different cube.*
- void [setBounds](#) (const [Point3D](#) *ptArray, unsigned int nPoints)
 - Obtain bounds from an array of Point3Ds.*
- void [setBounds](#) (const [Point3D](#) &p, const [Point3D](#) &q)
 - Use two points to set bounds – does not need to be high,low. This is worked out.*
- void [setBounds](#) (const std::vector< [Point3D](#) > &ptArray)
 - Obtain bounds from an array of Point3Ds.*
- void [setBounds](#) (const std::vector< [IonHit](#) > &ptArray)
 - Obtain bounds from an array of IonHits.*
- void [setBounds](#) (const [Point3D](#) &p, float radius)
 - Set the bounds using a single point and radius.*
- void [setInverseLimits](#) ()
 - Set the cube to be "inside out" at the limits of numeric results;.*
- float [getBound](#) (unsigned int bound, unsigned int minMax) const
 - Return the bound on the given dimension.*
- [Point3D](#) [getCentroid](#) () const
 - Return the centroid.*
- void [getBounds](#) ([Point3D](#) &low, [Point3D](#) &high) const
 - Get the bounds of the cube as two points, the lower left (minimum bound) and upper right corner (max bound)*
- [Point3D](#) [min](#) () const
- [Point3D](#) [max](#) () const
- float [getSize](#) (unsigned int dim) const
 - Return the size of the side parallel to the given dimension.*
- bool [isFlat](#) () const
 - Returns true if any bound is of null thickness.*
- bool [intersects](#) (const [Point3D](#) &pt, float sqrRad) const
 - Checks if a point intersects a sphere of centre Pt, radius² sqrRad.*
- bool [intersects](#) (const [BoundCube](#) &b) const
 - Check to see if this box intersects another.*
- bool [contains](#) (const [BoundCube](#) &b) const
 - Does this bounding box entirely contain another.*
- bool [containsPt](#) (const [Point3D](#) &pt) const
 - Check to see if the point is contained in, or part of the walls of the cube.*
- bool [containedInSphere](#) (const [Point3D](#) &origin, float radius) const
 - Returns true if this box is contained by the sphere, represented by the origin+radius.*
- [BoundCube](#) [makeUnion](#) (const [BoundCube](#) &b) const
 - Create the box containing a union of two bounding cubes.*
- [BoundCube](#) [makeIntersection](#) (const [BoundCube](#) &b) const
 - Create the box which is the intersection of two bounding cubes this is either null, or a box.*

- float `getMaxDistanceToBox` (const `Point3D` &pt) const
Returns maximum distance to box corners (which is an upper bound on max box distance).
- float `getLargestDim` () const
Get the largest dimension of the bound cube.
- float `getVolume` () const
Obtain the volume of the cube.
- void `setLimits` ()
Set min-max bounds to fp min/max.
- unsigned int `segmentTriple` (unsigned int dim, float slice) const
Return a triplet to indicate a spatial partition value lies in.
- `BoundCube` `operator=` (const `BoundCube` &)
- void `expand` (const `BoundCube` &b)
Expand (as needed) volume such that the argument bounding cube is enclosed by this one.
- void `expand` (const `Point3D` &p)
Expand such that point is contained in this volume. Existing volume must be valid.
- void `expand` (float f)
Expand the bounding cube by this value in each dimension (upper and lower)

Friends

- class `K3DTreeExact`
- class `K3DTreeApprox`
- `std::ostream` & `operator<<` (`std::ostream` &stream, const `BoundCube` &b)

6.7.1 Detailed Description

Helper class to define a bounding cube.

Definition at line 29 of file `boundcube.h`.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 `BoundCube`() [1/4]

```
AtomProbe::BoundCube::BoundCube ( ) [inline]
```

Definition at line 39 of file `boundcube.h`.

Referenced by `BoundCube()`.

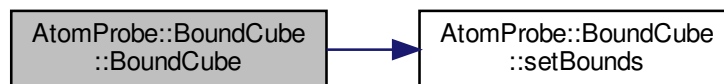
6.7.2.2 BoundCube() [2/4]

```
AtomProbe::BoundCube::BoundCube (
    const std::vector< Point3D > & pts ) [inline]
```

Definition at line 46 of file boundcube.h.

References `setBounds()`.

Here is the call graph for this function:



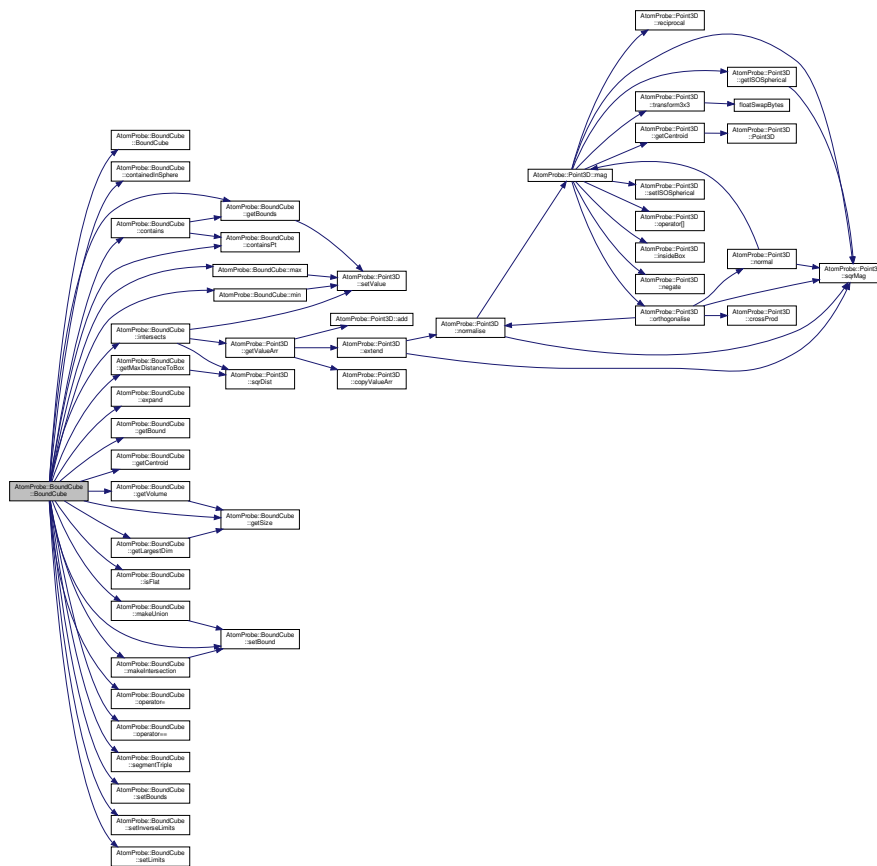
6.7.2.3 BoundCube() [3/4]

```
AtomProbe::BoundCube::BoundCube (
    const std::vector< IonHit > & pts ) [inline]
```

Definition at line 51 of file boundcube.h.

References `BoundCube()`, `containedInSphere()`, `contains()`, `containsPt()`, `expand()`, `getBound()`, `getBounds()`, `getCentroid()`, `getLargestDim()`, `getMaxDistanceToBox()`, `getSize()`, `getVolume()`, `intersects()`, `isFlat()`, `makeIntersection()`, `makeUnion()`, `max()`, `min()`, `operator<<`, `operator=()`, `operator==()`, `segmentTriple()`, `setBound()`, `setBounds()`, `setInverseLimits()`, and `setLimits()`.

Here is the call graph for this function:



6.7.2.4 BoundCube() [4 / 4]

```
AtomProbe::BoundCube::BoundCube (
    const Point3D & p1,
    const Point3D & p2 )
```

Definition at line 26 of file boundcube.cpp.

6.7.3 Member Function Documentation

6.7.3.1 containedInSphere()

```
bool AtomProbe::BoundCube::containedInSphere (
    const Point3D & origin,
    float radius ) const
```

Returns true if this box is contained by the sphere, represented by the origin+radius.

Definition at line 379 of file boundcube.cpp.

Referenced by BoundCube(), and AtomProbe::K3DTreeExact::ptsInSphere().

6.7.3.2 contains()

```
bool AtomProbe::BoundCube::contains (
    const BoundCube & b ) const
```

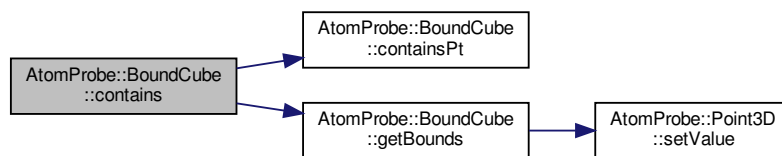
Does this bounding box entirely contain another.

Definition at line 360 of file boundcube.cpp.

References containsPt(), and getBounds().

Referenced by BoundCube().

Here is the call graph for this function:



6.7.3.3 containsPt()

```
bool AtomProbe::BoundCube::containsPt (
    const Point3D & pt ) const
```

Check to see if the point is contained in, or part of the walls of the cube.

Definition at line 367 of file boundcube.cpp.

References ASSERT.

Referenced by BoundCube(), contains(), AtomProbe::Mesh::getContainedNodes(), AtomProbe::Voxels< T >::getEdgeEnds(), AtomProbe::Voxels< T >::getInterpolatedData(), intersects(), AtomProbe::marchingCubes(), and AtomProbe::Mesh::pointsInside().

6.7.3.4 expand() [1/3]

```
void AtomProbe::BoundCube::expand (
    const BoundCube & b )
```

Expand (as needed) volume such that the argument bounding cube is enclosed by this one.

Definition at line 148 of file boundcube.cpp.

Referenced by BoundCube(), AtomProbe::Mesh::divideMeshSurface(), and AtomProbe::Voxels< T >::getEdgeEnds().

6.7.3.5 expand() [2/3]

```
void AtomProbe::BoundCube::expand (
    const Point3D & p )
```

Expand such that point is contained in this volume. Existing volume must be valid.

Definition at line 181 of file boundcube.cpp.

6.7.3.6 expand() [3/3]

```
void AtomProbe::BoundCube::expand (
    float f )
```

Expand the bounding cube by this value in each dimension (upper and lower)

Definition at line 197 of file boundcube.cpp.

6.7.3.7 getBound()

```
float AtomProbe::BoundCube::getBound (
    unsigned int bound,
    unsigned int minMax ) const
```

Return the bound on the given dimension.

bound - the dimension [x=0,y=1,z=2] minMax - 0 for lower bound, 1 for upper bound

Definition at line 86 of file boundcube.cpp.

References ASSERT.

Referenced by BoundCube().

6.7.3.8 getBounds()

```
void AtomProbe::BoundCube::getBounds (
    Point3D & low,
    Point3D & high ) const
```

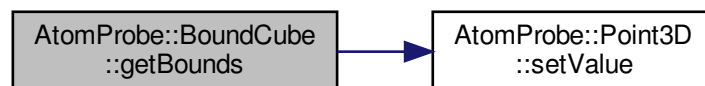
Get the bounds of the cube as two points, the lower left (minimum bound) and upper right corner (max bound)

Definition at line 302 of file boundcube.cpp.

References ASSERT, and AtomProbe::Point3D::setValue().

Referenced by BoundCube(), contains(), AtomProbe::Voxels< T >::init(), and intersects().

Here is the call graph for this function:



6.7.3.9 getCentroid()

```
Point3D AtomProbe::BoundCube::getCentroid ( ) const
```

Return the centroid.

Definition at line 486 of file boundcube.cpp.

References ASSERT.

Referenced by BoundCube().

6.7.3.10 getLargestDim()

```
float AtomProbe::BoundCube::getLargestDim ( ) const
```

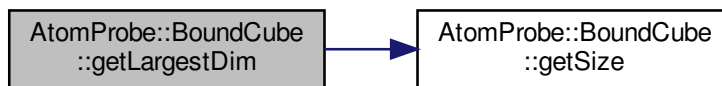
Get the largest dimension of the bound cube.

Definition at line 335 of file boundcube.cpp.

References getSize().

Referenced by BoundCube().

Here is the call graph for this function:



6.7.3.11 getMaxDistanceToBox()

```
float AtomProbe::BoundCube::getMaxDistanceToBox (
    const Point3D & pt ) const
```

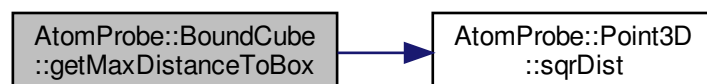
Returns maximum distance to box corners (which is an upper bound on max box distance).

Definition at line 499 of file boundcube.cpp.

References ASSERT, and AtomProbe::Point3D::sqrDist().

Referenced by BoundCube().

Here is the call graph for this function:



6.7.3.12 getSize()

```
float AtomProbe::BoundCube::getSize (
    unsigned int dim ) const
```

Return the size of the side parallel to the given dimension.

Definition at line 343 of file boundcube.cpp.

References ASSERT.

Referenced by BoundCube(), getLargestDim(), getVolume(), and AtomProbe::ReconstructionSphereOnCone←
::reconstruct().

6.7.3.13 getVolume()

```
float AtomProbe::BoundCube::getVolume ( ) const
```

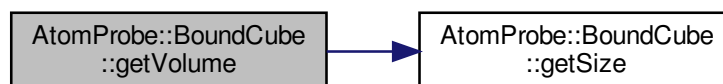
Obtain the volume of the cube.

Definition at line 355 of file boundcube.cpp.

References getSize().

Referenced by BoundCube(), and AtomProbe::ReconstructionSphereOnCone::reconstruct().

Here is the call graph for this function:



6.7.3.14 intersects() [1/2]

```
bool AtomProbe::BoundCube::intersects (
    const Point3D & pt,
    float sqrRad ) const
```

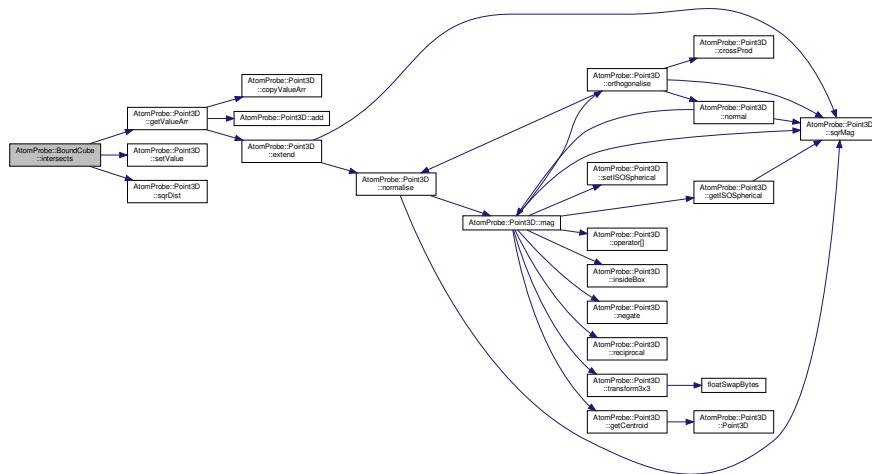
Checks if a point intersects a sphere of centre Pt, radius² sqrRad.

Definition at line 434 of file boundcube.cpp.

References AtomProbe::Point3D::getValueArr(), AtomProbe::Point3D::setValue(), and AtomProbe::Point3D::sqrDist().

Referenced by BoundCube(), AtomProbe::K3DTreeApprox::findNearest(), AtomProbe::K3DTreeExact::findNearestUntagged(), and AtomProbe::K3DTreeExact::ptsInSphere().

Here is the call graph for this function:



6.7.3.15 intersects() [2/2]

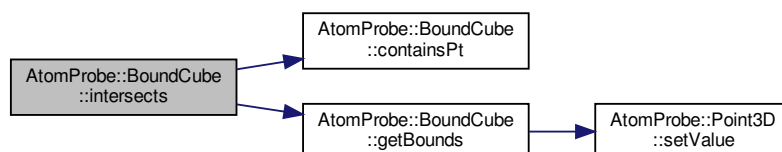
```
bool AtomProbe::BoundCube::intersects (
    const BoundCube & b ) const
```

Check to see if this box intersects another.

Definition at line 465 of file boundcube.cpp.

References `containsPt()`, and `getBounds()`.

Here is the call graph for this function:



6.7.3.16 isFlat()

```
bool AtomProbe::BoundCube::isFlat ( ) const
```

Returns true if any bound is of null thickness.

Definition at line 136 of file boundcube.cpp.

Referenced by BoundCube(), and main().

6.7.3.17 makeIntersection()

```
BoundCube AtomProbe::BoundCube::makeIntersection (
    const BoundCube & b ) const
```

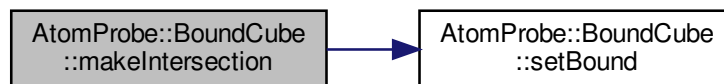
Create the box which is the intersection of two bounding cubes this is either null, or a box.

Definition at line 416 of file boundcube.cpp.

References setBound().

Referenced by BoundCube().

Here is the call graph for this function:



6.7.3.18 makeUnion()

```
BoundCube AtomProbe::BoundCube::makeUnion (
    const BoundCube & b ) const
```

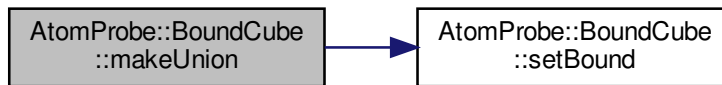
Create the box containing a union of two bounding cubes.

Definition at line 399 of file boundcube.cpp.

References setBound().

Referenced by BoundCube().

Here is the call graph for this function:



6.7.3.19 `max()`

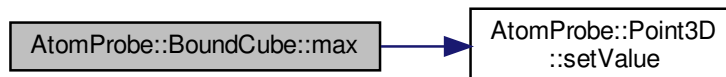
```
Point3D AtomProbe::BoundCube::max ( ) const
```

Definition at line 324 of file `boundcube.cpp`.

References `ASSERT`, and `AtomProbe::Point3D::setValue()`.

Referenced by `BoundCube()`, and `AtomProbe::Voxels< T >::setBounds()`.

Here is the call graph for this function:



6.7.3.20 `min()`

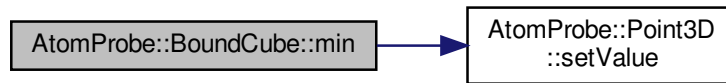
```
Point3D AtomProbe::BoundCube::min ( ) const
```

Definition at line 312 of file `boundcube.cpp`.

References `ASSERT`, and `AtomProbe::Point3D::setValue()`.

Referenced by `BoundCube()`, and `AtomProbe::Voxels< T >::setBounds()`.

Here is the call graph for this function:



6.7.3.21 operator=()

```
BoundCube AtomProbe::BoundCube::operator= (
    const BoundCube & b )
```

Definition at line 546 of file boundcube.cpp.

Referenced by BoundCube().

6.7.3.22 operator==()

```
bool AtomProbe::BoundCube::operator== (
    const BoundCube & other ) const
```

Definition at line 38 of file boundcube.cpp.

Referenced by BoundCube().

6.7.3.23 segmentTriple()

```
unsigned int AtomProbe::BoundCube::segmentTriple (
    unsigned int dim,
    float slice ) const
```

Return a triplet to indicate a spatial partition value lies in.

Returns which segment (lower:0, inside:1, higher:2) the given slice coordinate is at the specified dimension (eg, for a box [-1,1] in y, segmentTriple(1,-2) would return 0.

Definition at line 529 of file boundcube.cpp.

References ASSERT.

Referenced by BoundCube().

6.7.3.24 setBound()

```
void AtomProbe::BoundCube::setBound (
    unsigned int bound,
    unsigned int minMax,
    float value )
```

Set an individual bound.

bound : dimension of bound, 0=x,1=y,2=z *minMax* : 0 for lower bound, 1 for upper bound

Definition at line 76 of file boundcube.cpp.

References ASSERT.

Referenced by BoundCube(), makeIntersection(), and makeUnion().

6.7.3.25 setBounds() [1/7]

```
void AtomProbe::BoundCube::setBounds (
    float xMin,
    float yMin,
    float zMin,
    float xMax,
    float yMax,
    float zMax )
```

Set the bounds by passing in minima and maxima of each dimension.

Definition at line 49 of file boundcube.cpp.

Referenced by BoundCube(), AtomProbe::Mesh::divideMeshSurface(), AtomProbe::generate1DAxialDistHist(), AtomProbe::generate1DAxialDistHistSweep(), AtomProbe::BodyCentredCubicGen::generateLattice(), AtomProbe::IonHit::getBoundCube(), AtomProbe::K3DTreeExact::getBoundCube(), AtomProbe::Mesh::getBounds(), AtomProbe::Voxels< T >::getBounds(), AtomProbe::Voxels< T >::getEdgeEnds(), AtomProbe::marchingCubes(), AtomProbe::Mesh::pointsInside(), AtomProbe::Mesh::print(), AtomProbe::ReconstructionSphereOnCone::reconstruct(), AtomProbe::K3DTreeExact::resetPts(), and setBounds().

6.7.3.26 setBounds() [2/7]

```
void AtomProbe::BoundCube::setBounds (
    const BoundCube & b )
```

Set the bounding cube to be the same as a different cube.

Definition at line 60 of file boundcube.cpp.

6.7.3.27 setBounds() [3/7]

```
void AtomProbe::BoundCube::setBounds (
    const Point3D * ptArray,
    unsigned int nPoints )
```

Obtain bounds from an array of Point3Ds.

Definition at line 210 of file boundcube.cpp.

6.7.3.28 setBounds() [4/7]

```
void AtomProbe::BoundCube::setBounds (
    const Point3D & p,
    const Point3D & q )
```

Use two points to set bounds – does not need to be high,low. This is worked out.

Definition at line 244 of file boundcube.cpp.

6.7.3.29 setBounds() [5/7]

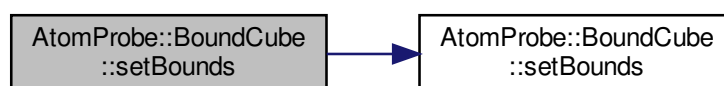
```
void AtomProbe::BoundCube::setBounds (
    const std::vector< Point3D > & ptArray )
```

Obtain bounds from an array of Point3Ds.

Definition at line 273 of file boundcube.cpp.

References setBounds().

Here is the call graph for this function:



6.7.3.30 setBounds() [6/7]

```
void AtomProbe::BoundCube::setBounds (
    const std::vector< IonHit > & ptArray )
```

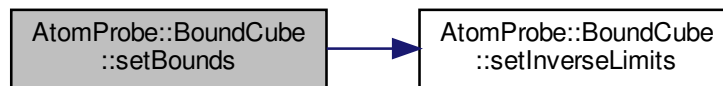
Obtain bounds from an array of IonHits.

Obtain bounds from an array of Point3Ds.

Definition at line 279 of file boundcube.cpp.

References setInverseLimits().

Here is the call graph for this function:

**6.7.3.31 setBounds()** [7/7]

```
void AtomProbe::BoundCube::setBounds (
    const Point3D & p,
    float radius )
```

Set the bounds using a single point and radius.

Bounds will be minimal sphere enclosing, where the sphere is defined by the (p,radius) pair

Definition at line 259 of file boundcube.cpp.

6.7.3.32 setInverseLimits()

```
void AtomProbe::BoundCube::setInverseLimits ( )
```

Set the cube to be "inside out" at the limits of numeric results;

Definition at line 94 of file boundcube.cpp.

Referenced by BoundCube(), AtomProbe::Mesh::divideMeshSurface(), AtomProbe::IonHit::getIonDataLimits(), and setBounds().

6.7.3.33 setLimits()

```
void AtomProbe::BoundCube::setLimits ( )
```

Set min-max bounds to fp min/max.

Definition at line 115 of file boundcube.cpp.

Referenced by BoundCube().

6.7.4 Friends And Related Function Documentation

6.7.4.1 K3DTreeApprox

```
friend class K3DTreeApprox [friend]
```

Definition at line 172 of file boundcube.h.

6.7.4.2 K3DTreeExact

```
friend class K3DTreeExact [friend]
```

Definition at line 171 of file boundcube.h.

6.7.4.3 operator<<

```
std::ostream& operator<< (
    std::ostream & stream,
    const BoundCube & b ) [friend]
```

Definition at line 563 of file boundcube.cpp.

Referenced by BoundCube().

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

- [include/atomprobe/primitives/boundcube.h](#)
- [src/primitives/boundcube.cpp](#)

6.8 AtomProbe::ComparePairFirst Class Reference

```
#include <helpFuncs.h>
```

Public Member Functions

- `template<class T1 , class T2 >`
`bool operator() (const std::pair< T1, T2 > &p1, const std::pair< T1, T2 > &p2) const`

6.8.1 Detailed Description

Definition at line 48 of file `helpFuncs.h`.

6.8.2 Member Function Documentation

6.8.2.1 operator()

```
template<class T1 , class T2 >  
bool AtomProbe::ComparePairFirst::operator() (  
    const std::pair< T1, T2 > & p1,  
    const std::pair< T1, T2 > & p2 ) const [inline]
```

Definition at line 52 of file `helpFuncs.h`.

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

- `src/helper/helpFuncs.h`

6.9 AtomProbe::ComparePairFirstReverse Class Reference

```
#include <helpFuncs.h>
```

Public Member Functions

- `template<class T1 , class T2 >`
`bool operator() (const std::pair< T1, T2 > &p1, const std::pair< T1, T2 > &p2) const`

6.9.1 Detailed Description

Definition at line 79 of file `helpFuncs.h`.

6.9.2 Member Function Documentation

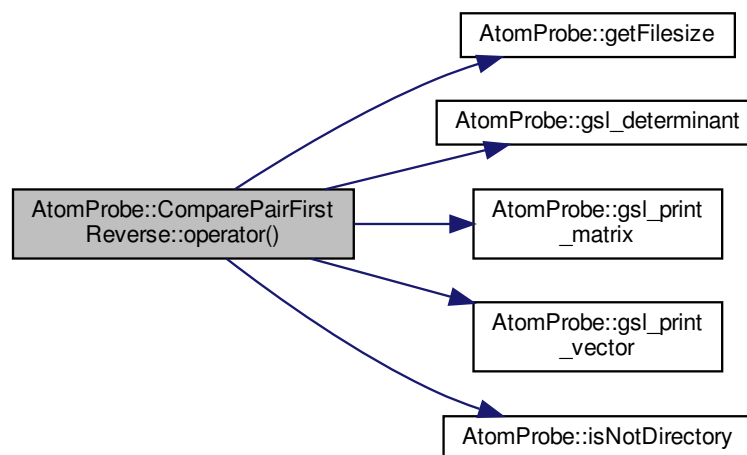
6.9.2.1 operator()

```
template<class T1 , class T2 >
bool AtomProbe::ComparePairFirstReverse::operator() (
    const std::pair< T1, T2 > & p1,
    const std::pair< T1, T2 > & p2 ) const [inline]
```

Definition at line 83 of file helpFuncs.h.

References `AtomProbe::getFileSize()`, `AtomProbe::gsl_determinant()`, `AtomProbe::gsl_print_matrix()`, `AtomProbe::gsl_print_vector()`, and `AtomProbe::isNotDirectory()`.

Here is the call graph for this function:



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

- [src/helper/helpFuncs.h](#)

6.10 AtomProbe::ComparePairSecond Class Reference

```
#include <helpFuncs.h>
```

Public Member Functions

- `template<class T1 , class T2 >`
`bool operator() (const std::pair< T1, T2 > &p1, const std::pair< T1, T2 > &p2) const`

6.10.1 Detailed Description

Definition at line 58 of file helpFuncs.h.

6.10.2 Member Function Documentation

6.10.2.1 operator()

```
template<class T1 , class T2 >
bool AtomProbe::ComparePairSecond::operator() (
    const std::pair< T1, T2 > & p1,
    const std::pair< T1, T2 > & p2 ) const [inline]
```

Definition at line 62 of file helpFuncs.h.

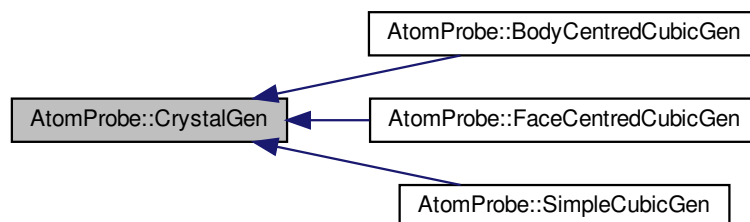
The documentation for this class was generated from the following file:

- [src/helper/helpFuncs.h](#)

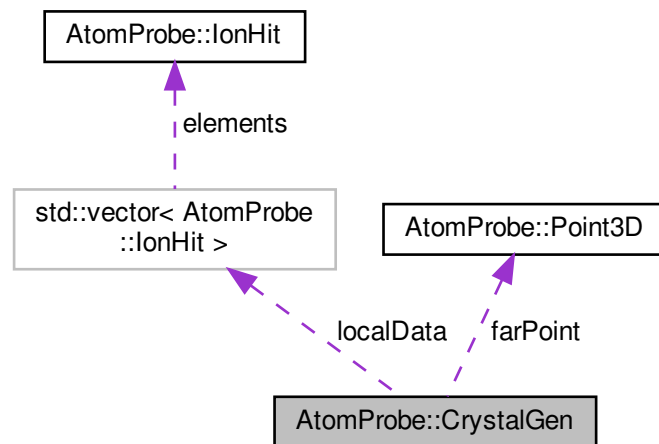
6.11 AtomProbe::CrystalGen Class Reference

```
#include <generate.h>
```

Inheritance diagram for AtomProbe::CrystalGen:



Collaboration diagram for AtomProbe::CrystalGen:



Public Member Functions

- [CrystalGen](#) ()
- virtual [~CrystalGen](#) ()
- virtual unsigned int [generateLattice](#) ()=0
- virtual void [swap](#) (std::vector< [lonHit](#) > &data)
- virtual void [extractPositions](#) (std::vector< [Point3D](#) > &data)
- virtual void [mirrorOut](#) ()

Protected Attributes

- std::vector< [lonHit](#) > [localData](#)
- [Point3D](#) [farPoint](#)

6.11.1 Detailed Description

Definition at line 14 of file generate.h.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 CrystalGen()

```
AtomProbe::CrystalGen::CrystalGen ( ) [inline]
```

Definition at line 25 of file generate.h.

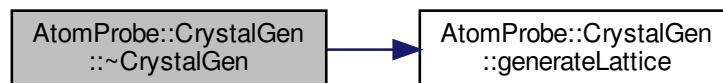
6.11.2.2 ~CrystalGen()

```
virtual AtomProbe::CrystalGen::~~CrystalGen ( ) [inline], [virtual]
```

Definition at line 26 of file generate.h.

References generateLattice().

Here is the call graph for this function:



6.11.3 Member Function Documentation

6.11.3.1 extractPositions()

```
void AtomProbe::CrystalGen::extractPositions (
    std::vector< Point3D > & data ) [virtual]
```

Definition at line 7 of file generate.cpp.

References localData.

Referenced by `AtomProbe::BodyCentredCubicGen::generateLattice()`, `main()`, and `swap()`.

6.11.3.2 generateLattice()

```
virtual unsigned int AtomProbe::CrystalGen::generateLattice ( ) [pure virtual]
```

Implemented in [AtomProbe::BodyCentredCubicGen](#), [AtomProbe::FaceCentredCubicGen](#), and [AtomProbe::SimpleCubicGen](#).

Referenced by `main()`, and `~CrystalGen()`.

6.11.3.3 mirrorOut()

```
void AtomProbe::CrystalGen::mirrorOut ( ) [virtual]
```

Definition at line 17 of file generate.cpp.

References localData.

Referenced by main(), and swap().

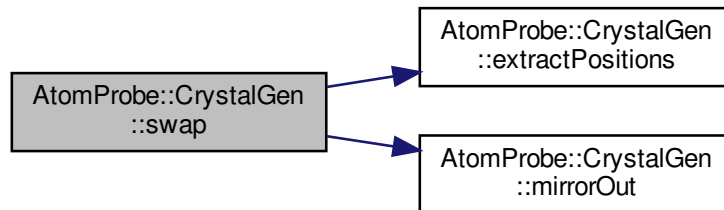
6.11.3.4 swap()

```
virtual void AtomProbe::CrystalGen::swap (
    std::vector< IonHit > & data ) [inline], [virtual]
```

Definition at line 32 of file generate.h.

References extractPositions(), and mirrorOut().

Here is the call graph for this function:



6.11.4 Member Data Documentation

6.11.4.1 farPoint

```
Point3D AtomProbe::CrystalGen::farPoint [protected]
```

Definition at line 22 of file generate.h.

Referenced by AtomProbe::BodyCentredCubicGen::BodyCentredCubicGen(), AtomProbe::FaceCentredCubicGen::FaceCentredCubicGen(), AtomProbe::SimpleCubicGen::generateLattice(), AtomProbe::FaceCentredCubicGen::generateLattice(), AtomProbe::BodyCentredCubicGen::generateLattice(), and AtomProbe::SimpleCubicGen::SimpleCubicGen().

6.11.4.2 localData

```
std::vector<IonHit> AtomProbe::CrystalGen::localData [protected]
```

Definition at line 18 of file generate.h.

Referenced by `extractPositions()`, `AtomProbe::SimpleCubicGen::generateLattice()`, `AtomProbe::FaceCentredCubicGen::generateLattice()`, `AtomProbe::BodyCentredCubicGen::generateLattice()`, and `mirrorOut()`.

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

- [include/atomprobe/latticeplanes/generate.h](#)
- [src/lattice/generate.cpp](#)

6.12 AtomProbe::EPOS_ENTRY Class Reference

```
#include <ionHit.h>
```

Public Member Functions

- [IonHit](#) `getIonHit ()` const
- void `getIonHit (IonHit &h)` const
- bool `operator== (const EPOS_ENTRY &obj)` const

Public Attributes

- float `x`
- float `y`
- float `z`
- float `massToCharge`
- float `timeOfFlight`
- float `voltDC`
- float `voltPulse`
- float `xDetector`
- float `yDetector`
- int32_t `deltaPulse`
- int32_t `hitMultiplicity`

6.12.1 Detailed Description

Definition at line 113 of file ionHit.h.

6.12.2 Member Function Documentation

6.12.2.1 getIonHit() [1/2]

```
IonHit AtomProbe::EPOS_ENTRY::getIonHit ( ) const
```

Definition at line 231 of file dataFiles.cpp.

6.12.2.2 getIonHit() [2/2]

```
void AtomProbe::EPOS_ENTRY::getIonHit (
    IonHit & h ) const
```

Definition at line 223 of file dataFiles.cpp.

References AtomProbe::IONHIT::massToCharge.

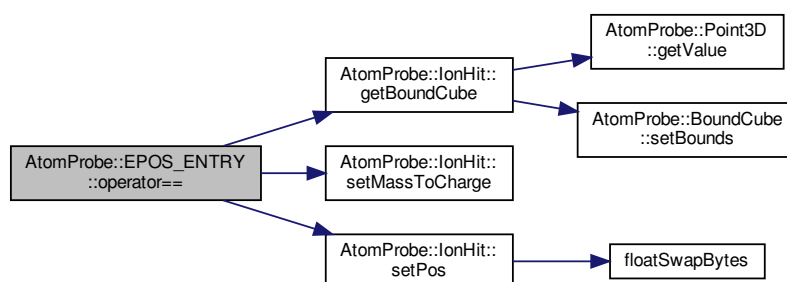
6.12.2.3 operator==()

```
bool AtomProbe::EPOS_ENTRY::operator== (
    const EPOS_ENTRY & obj ) const
```

Definition at line 271 of file ionhit.cpp.

References deltaPulse, AtomProbe::IonHit::getBoundCube(), hitMultiplicity, massToCharge, AtomProbe::IonHit::setMassToCharge(), AtomProbe::IonHit::setPos(), TEST, timeOfFlight, voltDC, voltPulse, x, xDetector, y, yDetector, and z.

Here is the call graph for this function:



6.12.3 Member Data Documentation

6.12.3.1 deltaPulse

```
int32_t AtomProbe::EPOS_ENTRY::deltaPulse
```

Definition at line 130 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

6.12.3.2 hitMultiplicity

```
int32_t AtomProbe::EPOS_ENTRY::hitMultiplicity
```

Definition at line 130 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

6.12.3.3 massToCharge

```
float AtomProbe::EPOS_ENTRY::massToCharge
```

Definition at line 118 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

6.12.3.4 timeOfFlight

```
float AtomProbe::EPOS_ENTRY::timeOfFlight
```

Definition at line 122 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

6.12.3.5 voltDC

```
float AtomProbe::EPOS_ENTRY::voltDC
```

Definition at line 122 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

6.12.3.6 voltPulse

```
float AtomProbe::EPOS_ENTRY::voltPulse
```

Definition at line 122 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

6.12.3.7 x

```
float AtomProbe::EPOS_ENTRY::x
```

Definition at line 118 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

6.12.3.8 xDetector

```
float AtomProbe::EPOS_ENTRY::xDetector
```

Definition at line 124 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

6.12.3.9 y

```
float AtomProbe::EPOS_ENTRY::y
```

Definition at line 118 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

6.12.3.10 yDetector

```
float AtomProbe::EPOS_ENTRY::yDetector
```

Definition at line 124 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

6.12.3.11 z

```
float AtomProbe::EPOS_ENTRY::z
```

Definition at line 118 of file ionHit.h.

Referenced by operator==(), and AtomProbe::readEposRecord().

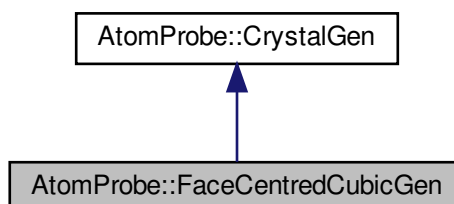
The documentation for this class was generated from the following files:

- [include/atomprobe/primitives/ionHit.h](#)
- [src/io/dataFiles.cpp](#)
- [src/primitives/ionhit.cpp](#)

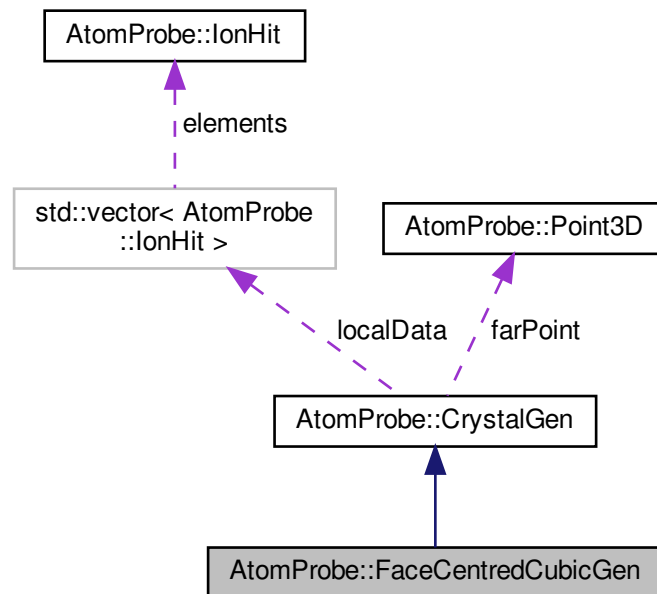
6.13 AtomProbe::FaceCentredCubicGen Class Reference

```
#include <generate.h>
```

Inheritance diagram for AtomProbe::FaceCentredCubicGen:



Collaboration diagram for AtomProbe::FaceCentredCubicGen:



Public Member Functions

- [FaceCentredCubicGen](#) (float latticeSpacing, float mToCCorner, const float *mToCFace, const [Point3D](#) &p)
- unsigned int [generateLattice](#) ()

Additional Inherited Members

6.13.1 Detailed Description

Definition at line 52 of file generate.h.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 FaceCentredCubicGen()

```

AtomProbe::FaceCentredCubicGen::FaceCentredCubicGen (
    float latticeSpacing,
    float mToCCorner,
    const float * mToCFace,
    const Point3D & p )
  
```

Definition at line 104 of file generate.cpp.

References [AtomProbe::CrystalGen::farPoint](#).

6.13.3 Member Function Documentation

6.13.3.1 generateLattice()

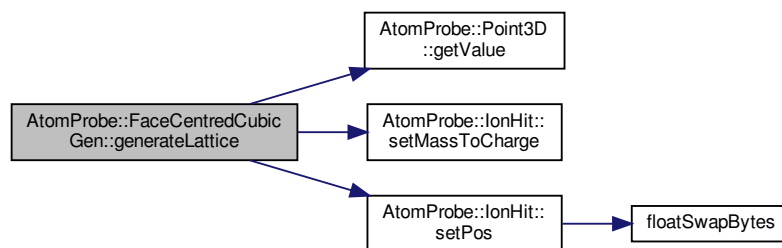
```
unsigned int AtomProbe::FaceCentredCubicGen::generateLattice ( ) [virtual]
```

Implements [AtomProbe::CrystalGen](#).

Definition at line 114 of file generate.cpp.

References [AtomProbe::CRYSTAL_BAD_UNIT_CELL](#), [AtomProbe::CrystalGen::farPoint](#), [AtomProbe::Point3D::getValue\(\)](#), [AtomProbe::CrystalGen::localData](#), [AtomProbe::IonHit::setMassToCharge\(\)](#), and [AtomProbe::IonHit::setPos\(\)](#).

Here is the call graph for this function:



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

- [include/atomprobe/latticeplanes/generate.h](#)
- [src/lattice/generate.cpp](#)

6.14 FixedStack< T > Class Template Reference

Public Member Functions

- [FixedStack](#) (size_t size)
- [FixedStack](#) (const [FixedStack](#) &f)
- [~FixedStack](#) ()
- [T & top](#) ()
- [void pop](#) ()
- [void push](#) (const T &t)
- [bool empty](#) () const

6.14.1 Detailed Description

```
template<class T>
class FixedStack< T >
```

Definition at line 51 of file massTool.cpp.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 FixedStack() [1/2]

```
template<class T >
FixedStack< T >::FixedStack (
    size_t size )
```

Definition at line 68 of file massTool.cpp.

6.14.2.2 FixedStack() [2/2]

```
template<class T>
FixedStack< T >::FixedStack (
    const FixedStack< T > & f )
```

6.14.2.3 ~FixedStack()

```
template<class T >
FixedStack< T >::~~FixedStack ( )
```

Definition at line 95 of file massTool.cpp.

References AtomProbe::vectorMultiErase().

Here is the call graph for this function:



6.14.3 Member Function Documentation

6.14.3.1 empty()

```
template<class T>
bool FixedStack< T >::empty ( ) const [inline]
```

Definition at line 64 of file massTool.cpp.

Referenced by AtomProbe::MassTool::bruteKnapsack().

6.14.3.2 pop()

```
template<class T >
void FixedStack< T >::pop ( )
```

Definition at line 82 of file massTool.cpp.

References ASSERT.

Referenced by AtomProbe::MassTool::bruteKnapsack().

6.14.3.3 push()

```
template<class T >
void FixedStack< T >::push (
    const T & t )
```

Definition at line 75 of file massTool.cpp.

Referenced by AtomProbe::MassTool::bruteKnapsack().

6.14.3.4 top()

```
template<class T >
T & FixedStack< T >::top ( )
```

Definition at line 89 of file massTool.cpp.

Referenced by AtomProbe::MassTool::bruteKnapsack().

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

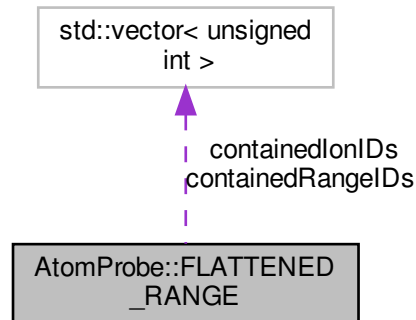
- src/algorithm/massTool.cpp

6.15 AtomProbe::FLATTENED_RANGE Struct Reference

Structure that allows for the multirange data to be mapped into.

```
#include <multiRange.h>
```

Collaboration diagram for AtomProbe::FLATTENED_RANGE:



Public Attributes

- float [startMass](#)
- float [endMass](#)
- `std::vector< unsigned int >` [containedIonIDs](#)
- `std::vector< unsigned int >` [containedRangeIDs](#)

6.15.1 Detailed Description

Structure that allows for the multirange data to be mapped into.

Definition at line 47 of file `multiRange.h`.

6.15.2 Member Data Documentation

6.15.2.1 containedIonIDs

```
std::vector<unsigned int> AtomProbe::FLATTENED_RANGE::containedIonIDs
```

Definition at line 52 of file `multiRange.h`.

6.15.2.2 containedRangeIDs

```
std::vector<unsigned int> AtomProbe::FLATTENED_RANGE::containedRangeIDs
```

Definition at line 52 of file multiRange.h.

6.15.2.3 endMass

```
float AtomProbe::FLATTENED_RANGE::endMass
```

Definition at line 50 of file multiRange.h.

6.15.2.4 startMass

```
float AtomProbe::FLATTENED_RANGE::startMass
```

Definition at line 50 of file multiRange.h.

Referenced by AtomProbe::MultiRange::flattenToMassAxis().

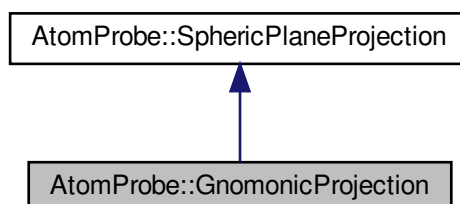
The documentation for this struct was generated from the following file:

- [include/atomprobe/io/multiRange.h](#)

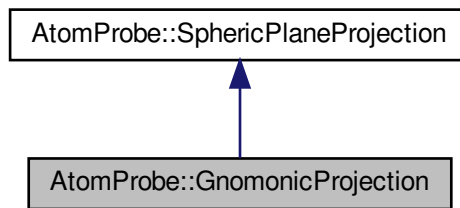
6.16 AtomProbe::GnomonicProjection Class Reference

```
#include <projection.h>
```

Inheritance diagram for AtomProbe::GnomonicProjection:



Collaboration diagram for AtomProbe::GnomonicProjection:



Public Member Functions

- virtual bool [toAzimuthal](#) (float fx, float fy, float &theta, float &phi) const
Convert from plane coordinates to spherical coords.
- virtual bool [toPlanar](#) (float theta, float phi, float &fx, float &fy) const
Convert from spherical coordinates to plane.
- virtual void [scaleDown](#) (float flightLength, float detX, float detY, float &scaledX, float &scaledY) const
Convert from actual detector position (eg. mm) and flight path to scaled-down transform.
- virtual void [scaleUp](#) (float flightLength, float scaledX, float scaledY, float &realX, float &realY) const
Convert from scaled-down radius to actual radius.

6.16.1 Detailed Description

Definition at line 44 of file projection.h.

6.16.2 Member Function Documentation

6.16.2.1 scaleDown()

```

void AtomProbe::GnomonicProjection::scaleDown (
    float flightLength,
    float detX,
    float detY,
    float & scaledX,
    float & scaledY ) const [virtual]
  
```

Convert from actual detector position (eg. mm) and flight path to scaled-down transform.

Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 84 of file projection.cpp.

References ASSERT.

6.16.2.2 scaleUp()

```
void AtomProbe::GnomonicProjection::scaleUp (
    float flightLength,
    float scaledX,
    float scaledY,
    float & realX,
    float & realY ) const [virtual]
```

Convert from scaled-down radius to actual radius.

Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 96 of file projection.cpp.

References ASSERT.

6.16.2.3 toAzimuthal()

```
bool AtomProbe::GnomonicProjection::toAzimuthal (
    float fx,
    float fy,
    float & theta,
    float & phi ) const [virtual]
```

Convert from plane coordinates to spherical coords.

Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 68 of file projection.cpp.

Referenced by [AtomProbe::ModifiedFocusSphericProjection::scaleUp\(\)](#).

6.16.2.4 toPlanar()

```
bool AtomProbe::GnomonicProjection::toPlanar (
    float theta,
    float phi,
    float & fx,
    float & fy ) const [virtual]
```

Convert from spherical coordinates to plane.

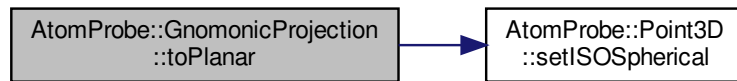
Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 34 of file projection.cpp.

References ASSERT, M_PI, and [AtomProbe::Point3D::setISOspherical\(\)](#).

Referenced by [AtomProbe::ModifiedFocusSphericProjection::scaleUp\(\)](#).

Here is the call graph for this function:



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

- [include/atomprobe/projection/projection.h](#)
- [src/projection/projection.cpp](#)

6.17 AtomProbe::IonHit Class Reference

This is a data holding class for POS file ions, from.

```
#include <ionHit.h>
```

Public Member Functions

- [IonHit](#) ()
- [IonHit](#) (float *)
- [IonHit](#) (const [IonHit](#) &)
- [IonHit](#) (const [Point3D](#) &p, float massToCharge)
- [IonHit](#) (float x, float y, float z, float mass)
- void [setHit](#) (float *arr)
- void [setMassToCharge](#) (float newMassToCharge)
- void [setPos](#) (const [Point3D](#) &pos)
- void [setPos](#) (unsigned int idx, float pos)
- void [setPos](#) (float fX, float fY, float fZ)
- [Point3D](#) [getPos](#) () const
- const [Point3D](#) & [getPosRef](#) () const
- bool [hasNaN](#) ()
- float [getMassToCharge](#) () const
- const [IonHit](#) & [operator=](#) (const [IonHit](#) &obj)
- bool [operator==](#) (const [IonHit](#) &obj) const
- float [operator\[\]](#) (unsigned int ui) const
- float & [operator\[\]](#) (unsigned int ui)
- [IonHit](#) [operator+](#) (const [Point3D](#) &obj)

Static Public Member Functions

- static void [getPoints](#) (const std::vector< [IonHit](#) > &ions, std::vector< [Point3D](#) > &pts)
- static void [getBoundCube](#) (const std::vector< [IonHit](#) > &p, [BoundCube](#) &b)
- static void [getCentroid](#) (const std::vector< [IonHit](#) > &points, [Point3D](#) ¢roid)
- static [BoundCube](#) [getIonDataLimits](#) (const std::vector< [IonHit](#) > &points)

Friends

- `std::ostream & operator<<` (`std::ostream &stream`, `const IonHit &`)
Output streaming operator. Users (x,y,z,m) as format for output.

6.17.1 Detailed Description

This is a data holding class for POS file ions, from.

Definition at line 36 of file `ionHit.h`.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 IonHit() [1/5]

```
AtomProbe::IonHit::IonHit ( )
```

Definition at line 38 of file `ionhit.cpp`.

6.17.2.2 IonHit() [2/5]

```
AtomProbe::IonHit::IonHit (
    float * buffer )
```

Definition at line 44 of file `ionhit.cpp`.

6.17.2.3 IonHit() [3/5]

```
AtomProbe::IonHit::IonHit (
    const IonHit & obj2 )
```

Definition at line 52 of file `ionhit.cpp`.

6.17.2.4 IonHit() [4/5]

```
AtomProbe::IonHit::IonHit (
    const Point3D & p,
    float massToCharge )
```

Definition at line 56 of file `ionhit.cpp`.

6.17.2.5 IonHit() [5/5]

```
AtomProbe::IonHit::IonHit (
    float x,
    float y,
    float z,
    float mass )
```

Definition at line 48 of file ionhit.cpp.

6.17.3 Member Function Documentation

6.17.3.1 getBoundCube()

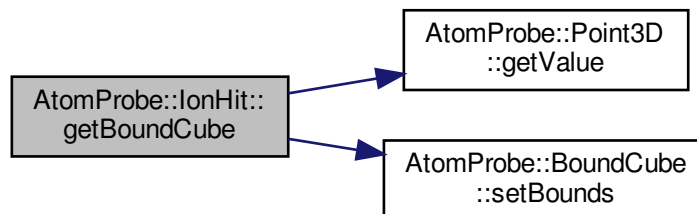
```
void AtomProbe::IonHit::getBoundCube (
    const std::vector< IonHit > & p,
    BoundCube & b ) [static]
```

Definition at line 153 of file ionhit.cpp.

References ASSERT, AtomProbe::Point3D::getValue(), and AtomProbe::BoundCube::setBounds().

Referenced by getPosRef(), and AtomProbe::EPOS_ENTRY::operator==().

Here is the call graph for this function:



6.17.3.2 getCentroid()

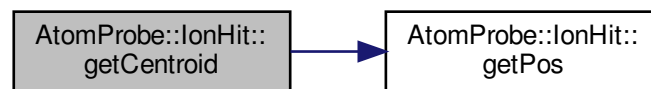
```
void AtomProbe::IonHit::getCentroid (
    const std::vector< IonHit > & points,
    Point3D & centroid ) [static]
```

Definition at line 128 of file ionhit.cpp.

References `getPos()`.

Referenced by `getPosRef()`.

Here is the call graph for this function:



6.17.3.3 getIonDataLimits()

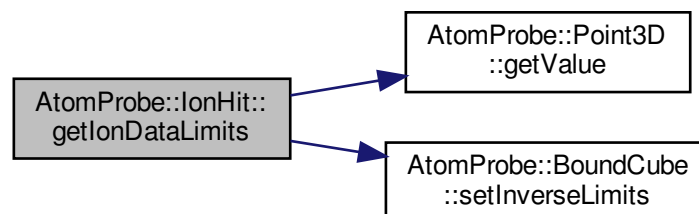
```
BoundCube AtomProbe::IonHit::getIonDataLimits (
    const std::vector< IonHit > & points ) [static]
```

Definition at line 188 of file ionhit.cpp.

References `ASSERT`, `AtomProbe::Point3D::getValue()`, and `AtomProbe::BoundCube::setInverseLimits()`.

Referenced by `getPosRef()`, and `AtomProbe::K3DTreeExact::resetPts()`.

Here is the call graph for this function:



6.17.3.4 getMassToCharge()

```
float AtomProbe::IonHit::getMassToCharge ( ) const
```

Definition at line 65 of file ionhit.cpp.

Referenced by AtomProbe::RangeFile::getIonID(), AtomProbe::RangeFile::getName(), AtomProbe::MultiRange::isRanged(), AtomProbe::RangeFile::isRanged(), main(), and AtomProbe::operator<<().

6.17.3.5 getPoints()

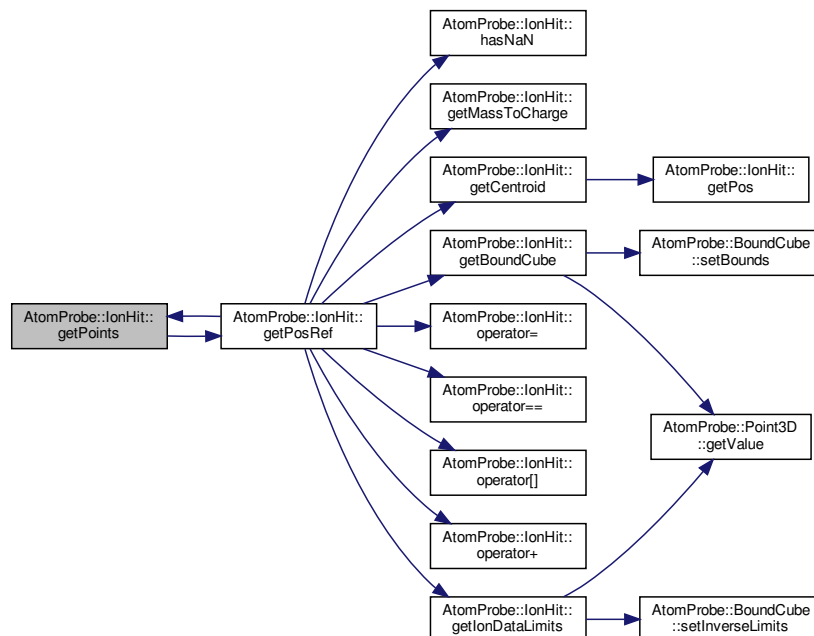
```
void AtomProbe::IonHit::getPoints (
    const std::vector< IonHit > & ions,
    std::vector< Point3D > & pts ) [static]
```

Definition at line 109 of file ionhit.cpp.

References getPosRef().

Referenced by getPosRef().

Here is the call graph for this function:



6.17.3.6 getPos()

```
Point3D AtomProbe::IonHit::getPos ( ) const
```

Definition at line 117 of file ionhit.cpp.

Referenced by getCentroid(), main(), AtomProbe::operator<<(), and setPos().

6.17.3.7 getPosRef()

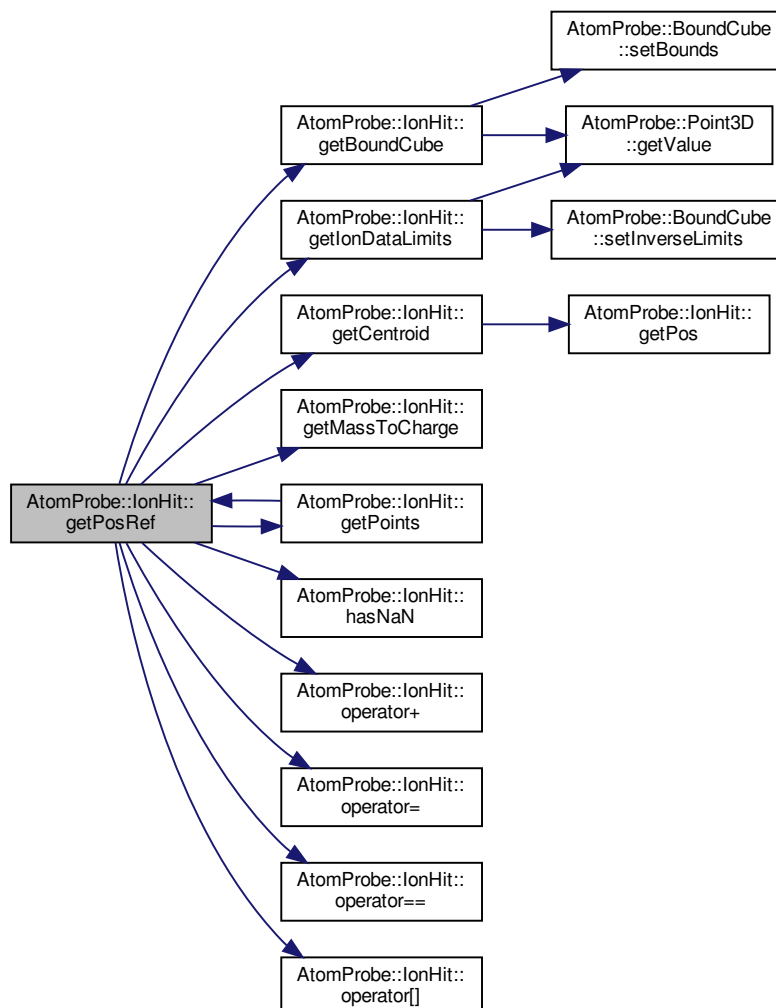
```
const Point3D& AtomProbe::IonHit::getPosRef ( ) const [inline]
```

Definition at line 65 of file ionHit.h.

References getBoundCube(), getCentroid(), getIonDataLimits(), getMassToCharge(), getPoints(), hasNaN(), operator+(), operator<<, operator=(), operator==(), and operator[]().

Referenced by getPoints().

Here is the call graph for this function:



6.17.3.8 hasNaN()

```
bool AtomProbe::IonHit::hasNaN ( )
```

Definition at line 122 of file ionhit.cpp.

Referenced by `getPosRef()`, and `AtomProbe::loadPosFile()`.

6.17.3.9 operator+()

```
IonHit AtomProbe::IonHit::operator+ (
    const Point3D & obj )
```

Definition at line 103 of file ionhit.cpp.

Referenced by `getPosRef()`.

6.17.3.10 operator=()

```
const IonHit & AtomProbe::IonHit::operator= (
    const IonHit & obj )
```

Definition at line 90 of file ionhit.cpp.

Referenced by `getPosRef()`.

6.17.3.11 operator==()

```
bool AtomProbe::IonHit::operator==(
    const IonHit & obj ) const
```

Definition at line 98 of file ionhit.cpp.

Referenced by `getPosRef()`.

6.17.3.12 operator[]() [1/2]

```
float AtomProbe::IonHit::operator[] (
    unsigned int ui ) const
```

Definition at line 251 of file ionhit.cpp.

References `ASSERT`.

Referenced by `getPosRef()`.

6.17.3.13 operator[]() [2/2]

```
float & AtomProbe::IonHit::operator[] (
    unsigned int ui )
```

Definition at line 260 of file ionhit.cpp.

References ASSERT.

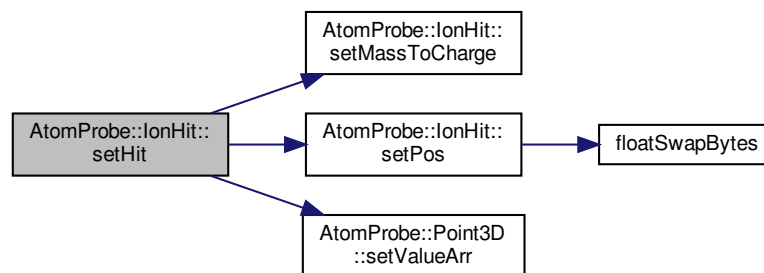
6.17.3.14 setHit()

```
void AtomProbe::IonHit::setHit (
    float * arr ) [inline]
```

Definition at line 50 of file ionHit.h.

References setMassToCharge(), setPos(), and AtomProbe::Point3D::setValueArr().

Here is the call graph for this function:



6.17.3.15 setMassToCharge()

```
void AtomProbe::IonHit::setMassToCharge (
    float newMassToCharge )
```

Definition at line 60 of file ionhit.cpp.

Referenced by `AtomProbe::FaceCentredCubicGen::generateLattice()`, `AtomProbe::BodyCentredCubicGen::generateLattice()`, `AtomProbe::loadPosFile()`, `AtomProbe::loadTapsimBinFile()`, `AtomProbe::EPOS_ENTRY::operator==()`, and `setHit()`.

6.17.3.16 setPos() [1/3]

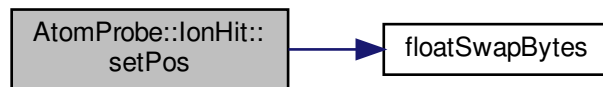
```
void AtomProbe::IonHit::setPos (
    const Point3D & pos )
```

Definition at line 76 of file ionhit.cpp.

References floatSwapBytes().

Referenced by AtomProbe::SimpleCubicGen::generateLattice(), AtomProbe::FaceCentredCubicGen::generateLattice(), AtomProbe::BodyCentredCubicGen::generateLattice(), AtomProbe::loadPosFile(), AtomProbe::loadTapsimBinFile(), AtomProbe::EPOS_ENTRY::operator==(), and setHit().

Here is the call graph for this function:

**6.17.3.17 setPos()** [2/3]

```
void AtomProbe::IonHit::setPos (
    unsigned int idx,
    float pos )
```

Definition at line 70 of file ionhit.cpp.

References ASSERT.

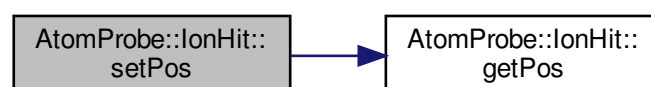
6.17.3.18 setPos() [3/3]

```
void AtomProbe::IonHit::setPos (
    float fX,
    float fY,
    float fZ ) [inline]
```

Definition at line 59 of file ionHit.h.

References getPos().

Here is the call graph for this function:



6.17.4 Friends And Related Function Documentation

6.17.4.1 operator<<

```
std::ostream& operator<< (  
    std::ostream & stream,  
    const IonHit & ion ) [friend]
```

Output streaming operator. Users (x,y,z,m) as format for output.

Definition at line 181 of file ionhit.cpp.

Referenced by getPosRef().

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

- include/atomprobe/primitives/ionHit.h
- src/primitives/ionhit.cpp

6.18 AtomProbe::IONHIT Struct Reference

Record as stored in a .POS file.

Public Attributes

- float [pos](#) [3]
- float [massToCharge](#)

6.18.1 Detailed Description

Record as stored in a .POS file.

Definition at line 45 of file dataFiles.cpp.

6.18.2 Member Data Documentation

6.18.2.1 massToCharge

```
float AtomProbe::IONHIT::massToCharge
```

Definition at line 48 of file dataFiles.cpp.

Referenced by AtomProbe::EPOS_ENTRY::getIonHit(), and AtomProbe::loadPosFile().

6.18.2.2 pos

```
float AtomProbe::IONHIT::pos[3]
```

Definition at line 47 of file dataFiles.cpp.

Referenced by AtomProbe::loadPosFile().

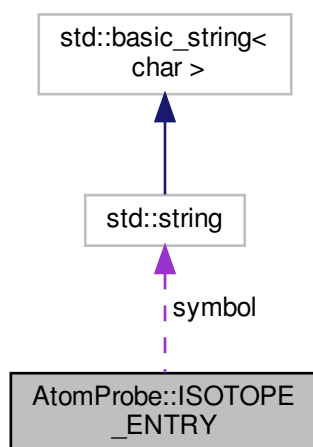
The documentation for this struct was generated from the following file:

- [src/io/dataFiles.cpp](#)

6.19 AtomProbe::ISOTOPE_ENTRY Struct Reference

```
#include <abundance.h>
```

Collaboration diagram for AtomProbe::ISOTOPE_ENTRY:



Public Attributes

- `std::string` [symbol](#)
- `size_t` [massNumber](#)
- `size_t` [atomicNumber](#)
- `float` [mass](#)
- `float` [massError](#)
- `float` [abundance](#)
- `float` [abundanceError](#)

6.19.1 Detailed Description

Definition at line 42 of file abundance.h.

6.19.2 Member Data Documentation

6.19.2.1 abundance

```
float AtomProbe::ISOTOPE_ENTRY::abundance
```

Definition at line 49 of file abundance.h.

Referenced by AtomProbe::filterBySolutionPPM(), and AtomProbe::AbundanceData::open().

6.19.2.2 abundanceError

```
float AtomProbe::ISOTOPE_ENTRY::abundanceError
```

Definition at line 50 of file abundance.h.

Referenced by AtomProbe::AbundanceData::open().

6.19.2.3 atomicNumber

```
size_t AtomProbe::ISOTOPE_ENTRY::atomicNumber
```

Definition at line 46 of file abundance.h.

Referenced by AtomProbe::MultiRange::copyDataFromRange(), AtomProbe::AbundanceData::elementName(), and AtomProbe::AbundanceData::open().

6.19.2.4 mass

```
float AtomProbe::ISOTOPE_ENTRY::mass
```

Definition at line 47 of file abundance.h.

Referenced by AtomProbe::AbundanceData::elementName(), AtomProbe::maxExplainedFraction(), and AtomProbe::AbundanceData::open().

6.19.2.5 massError

```
float AtomProbe::ISOTOPE_ENTRY::massError
```

Definition at line 48 of file abundance.h.

Referenced by AtomProbe::AbundanceData::open().

6.19.2.6 massNumber

```
size_t AtomProbe::ISOTOPE_ENTRY::massNumber
```

Definition at line 45 of file abundance.h.

Referenced by AtomProbe::AbundanceData::open().

6.19.2.7 symbol

```
std::string AtomProbe::ISOTOPE_ENTRY::symbol
```

Definition at line 44 of file abundance.h.

Referenced by AtomProbe::AbundanceData::open().

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

- [include/atomprobe/isotopes/abundance.h](#)

6.20 AtomProbe::K3DNodeApprox Class Reference

Node Class for storing point.

```
#include <K3DTree-approx.h>
```

Public Member Functions

- [Point3D](#) [getLoc](#) () const
Return the point data from the ndoe.
- const [Point3D](#) * [getLocRef](#) () const
Return a pointer to the point from the node.
- void [setLeft](#) ([K3DNodeApprox](#) *node)
Set the left child node.
- void [setRight](#) ([K3DNodeApprox](#) *node)
Set the right child node.
- void [setLoc](#) (const [Point3D](#) &)
Set the point data associated with this node.
- void [setAxis](#) (unsigned int newAxis)
Set the axis that this node operates on.
- unsigned int [getAxis](#) () const
Retrieve the axis that this node operates on.
- float [getAxisVal](#) () const
retrieve the value associated with this axis
- float [getLocVal](#) (unsigned int pos) const
Obtain the coordinates at dimension "pos".
- float [sqrDist](#) (const [Point3D](#) &pt) const
Obtain the distance between this node and another point.
- [K3DNodeApprox](#) * [left](#) () const
Obtain pointer to left child.
- [K3DNodeApprox](#) * [right](#) () const
Obtain pointer to right child.
- void [deleteChildren](#) ()
Recursively delete this node and all children.
- void [dump](#) (std::ostream &, unsigned int) const
print the node data out to a given stream

6.20.1 Detailed Description

Node Class for storing point.

Definition at line 53 of file [K3DTree-approx.h](#).

6.20.2 Member Function Documentation

6.20.2.1 [deleteChildren\(\)](#)

```
void AtomProbe::K3DNodeApprox::deleteChildren ( )
```

Recursively delete this node and all children.

Definition at line 63 of file [K3DTree-approx.cpp](#).

Referenced by [AtomProbe::K3DTreeApprox::kill\(\)](#).

6.20.2.2 dump()

```
void AtomProbe::K3DNodeApprox::dump (
    std::ostream & strm,
    unsigned int depth ) const
```

print the node data out to a given stream

Definition at line 83 of file K3DTree-approx.cpp.

Referenced by AtomProbe::K3DTreeApprox::dump().

6.20.2.3 getAxis()

```
unsigned int AtomProbe::K3DNodeApprox::getAxis ( ) const [inline]
```

Retrieve the axis that this node operates on.

Definition at line 79 of file K3DTree-approx.h.

6.20.2.4 getAxisVal()

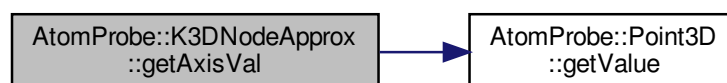
```
float AtomProbe::K3DNodeApprox::getAxisVal ( ) const [inline]
```

retrieve the value associated with this axis

Definition at line 81 of file K3DTree-approx.h.

References AtomProbe::Point3D::getValue().

Here is the call graph for this function:



6.20.2.5 getLoc()

```
Point3D AtomProbe::K3DNodeApprox::getLoc ( ) const
```

Return the point data from the ndoe.

Definition at line 58 of file K3DTree-approx.cpp.

6.20.2.6 getLocRef()

```
const Point3D* AtomProbe::K3DNodeApprox::getLocRef ( ) const [inline]
```

Return a pointer to the point from the node.

Definition at line 68 of file K3DTree-approx.h.

Referenced by AtomProbe::K3DTreeApprox::findNearest().

6.20.2.7 getLocVal()

```
float AtomProbe::K3DNodeApprox::getLocVal (
    unsigned int pos ) const [inline]
```

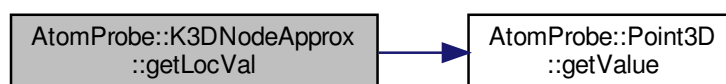
Obtain the coordinates at dimension "pos".

Definition at line 83 of file K3DTree-approx.h.

References AtomProbe::Point3D::getValue().

Referenced by AtomProbe::K3DTreeApprox::findNearest().

Here is the call graph for this function:



6.20.2.8 left()

```
K3DNodeApprox* AtomProbe::K3DNodeApprox::left ( ) const [inline]
```

Obtain pointer to left child.

Definition at line 87 of file K3DTree-approx.h.

Referenced by AtomProbe::K3DTreeApprox::findNearest().

6.20.2.9 right()

```
K3DNodeApprox* AtomProbe::K3DNodeApprox::right ( ) const [inline]
```

Obtain pointer to right child.

Definition at line 89 of file K3DTree-approx.h.

Referenced by AtomProbe::K3DTreeApprox::findNearest().

6.20.2.10 setAxis()

```
void AtomProbe::K3DNodeApprox::setAxis (
    unsigned int newAxis ) [inline]
```

Set the axis that this node operates on.

Definition at line 77 of file K3DTree-approx.h.

Referenced by AtomProbe::K3DTreeApprox::buildByRef().

6.20.2.11 setLeft()

```
void AtomProbe::K3DNodeApprox::setLeft (
    K3DNodeApprox * node ) [inline]
```

Set the left child node.

Definition at line 70 of file K3DTree-approx.h.

Referenced by AtomProbe::K3DTreeApprox::buildByRef().

6.20.2.12 setLoc()

```
void AtomProbe::K3DNodeApprox::setLoc (
    const Point3D & locNew )
```

Set the point data associated with this node.

Definition at line 53 of file K3DTree-approx.cpp.

Referenced by AtomProbe::K3DTreeApprox::buildByRef().

6.20.2.13 setRight()

```
void AtomProbe::K3DNodeApprox::setRight (
    K3DNodeApprox * node ) [inline]
```

Set the right child node.

Definition at line 72 of file K3DTree-approx.h.

Referenced by AtomProbe::K3DTreeApprox::buildByRef().

6.20.2.14 sqrDist()

```
float AtomProbe::K3DNodeApprox::sqrDist (
    const Point3D & pt ) const [inline]
```

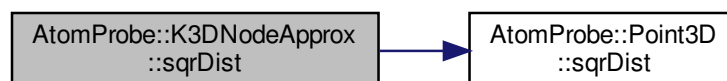
Obtain the distance between this node and another point.

Definition at line 85 of file K3DTree-approx.h.

References AtomProbe::Point3D::sqrDist().

Referenced by AtomProbe::K3DTreeApprox::findNearest().

Here is the call graph for this function:



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

- [include/atomprobe/algorithm/K3DTree-approx.h](#)
- [src/algorithm/K3DTree-approx.cpp](#)

6.21 AtomProbe::K3DNodeExact Class Reference

Node Class for storing point.

```
#include <K3DTree-exact.h>
```

Public Attributes

- `size_t` [childLeft](#)
- `size_t` [childRight](#)
- `bool` [tagged](#)

6.21.1 Detailed Description

Node Class for storing point.

Definition at line 41 of file `K3DTree-exact.h`.

6.21.2 Member Data Documentation

6.21.2.1 childLeft

```
size_t AtomProbe::K3DNodeExact::childLeft
```

Definition at line 45 of file `K3DTree-exact.h`.

6.21.2.2 childRight

```
size_t AtomProbe::K3DNodeExact::childRight
```

Definition at line 47 of file `K3DTree-exact.h`.

6.21.2.3 tagged

```
bool AtomProbe::K3DNodeExact::tagged
```

Definition at line 50 of file `K3DTree-exact.h`.

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

- `include/atomprobe/algorithm/K3DTree-exact.h`

6.22 AtomProbe::K3DTreeApprox Class Reference

3D specific KD tree

```
#include <K3DTree-approx.h>
```

Public Member Functions

- [K3DTreeApprox](#) ()
- [~K3DTreeApprox](#) ()
 - Cleans up tree, deallocates nodes.*
- void [build](#) (const std::vector< [Point3D](#) > &pts)
 - Builds a balanced KD tree from a list of points.*
- void [buildByRef](#) (std::vector< [Point3D](#) > &pts)
 - Builds a balanced KD tree from a list of points.*
- void [kill](#) ()
 - Destroy the tree contents.*
- const [Point3D](#) * [findNearest](#) (const [Point3D](#) &, const [BoundCube](#) &, float deadDistSqr) const
 - Find the nearest point to a given P that lies outside some dead zone.*
- void [findKNearest](#) (const [Point3D](#) &sourcePoint, const [BoundCube](#) &treeDomain, unsigned int numNNs, std::vector< const [Point3D](#) * > &results, float deadDistSqr=0.0f) const
 - Find the nearest N points.*
- void [dump](#) (std::ostream &) const
 - Textual output of tree. tabs are used to separate different levels of the tree.*
- unsigned int [nodeCount](#) () const
 - Print the number of nodes stored in the tree.*

6.22.1 Detailed Description

3D specific KD tree

Definition at line 98 of file K3DTree-approx.h.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 K3DTreeApprox()

```
AtomProbe::K3DTreeApprox::K3DTreeApprox ( )
```

Definition at line 103 of file K3DTree-approx.cpp.

6.22.2.2 ~K3DTreeApprox()

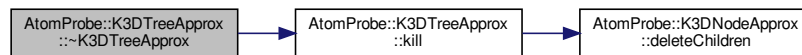
```
AtomProbe::K3DTreeApprox::~~K3DTreeApprox ( )
```

Cleans up tree, deallocates nodes.

Definition at line 109 of file K3DTree-approx.cpp.

References kill().

Here is the call graph for this function:



6.22.3 Member Function Documentation

6.22.3.1 build()

```
void AtomProbe::K3DTreeApprox::build (
    const std::vector< Point3D > & pts )
```

Builds a balanced KD tree from a list of points.

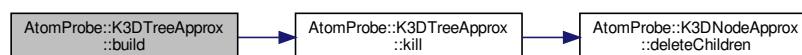
This call is const, and copies in order to prevent re-ordering of the points. If reordering points is unimportant, use buildByRef

Definition at line 127 of file K3DTree-approx.cpp.

References kill().

Referenced by findKNearest(), and testInexactKDTree().

Here is the call graph for this function:



6.22.3.2 buildByRef()

```
void AtomProbe::K3DTreeApprox::buildByRef (
    std::vector< Point3D > & pts )
```

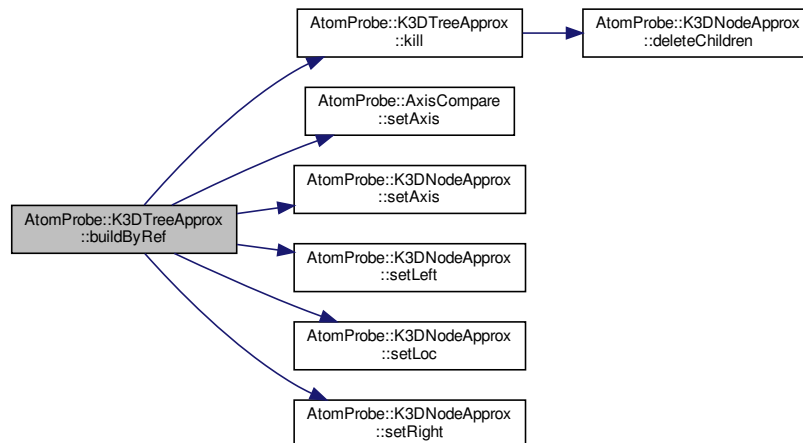
Builds a balanced KD tree from a list of points.

This uses a pass by ref where the points get scrambled (sorted). Resultant tree should be identical to that generated by [build\(\)](#)

Definition at line 149 of file K3DTree-approx.cpp.

References [kill\(\)](#), [AtomProbe::AxisCompare::setAxis\(\)](#), [AtomProbe::K3DNodeApprox::setAxis\(\)](#), [AtomProbe::K3DNodeApprox::setLeft\(\)](#), [AtomProbe::K3DNodeApprox::setLoc\(\)](#), and [AtomProbe::K3DNodeApprox::setRight\(\)](#).

Here is the call graph for this function:



6.22.3.3 dump()

```
void AtomProbe::K3DTreeApprox::dump (
    std::ostream & strm ) const
```

Textual output of tree. tabs are used to separate different levels of the tree.

The output from this function can be quite large for even modest trees. Recommended for debugging only

Definition at line 206 of file K3DTree-approx.cpp.

References [AtomProbe::K3DNodeApprox::dump\(\)](#).

Here is the call graph for this function:



6.22.3.4 findKNearest()

```

void AtomProbe::K3DTreeApprox::findKNearest (
    const Point3D & sourcePoint,
    const BoundCube & treeDomain,
    unsigned int numNNs,
    std::vector< const Point3D *> & results,
    float deadDistSqr = 0.0f ) const
  
```

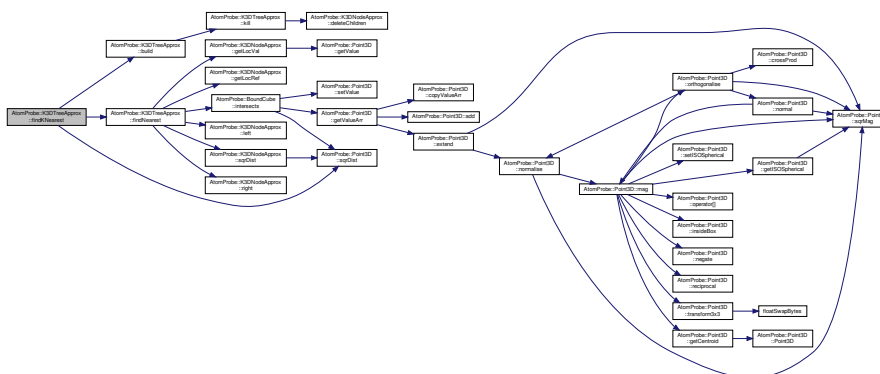
Find the nearest N points.

Finds the nearest N points, that lie outside a dead distance of deadDistSqr. k is the number to find

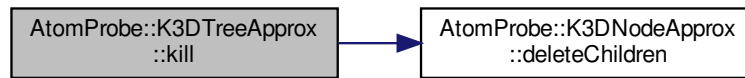
Definition at line 447 of file K3DTree-approx.cpp.

References [build\(\)](#), [findNearest\(\)](#), [AtomProbe::Point3D::sqrDist\(\)](#), and [TEST](#).

Here is the call graph for this function:



Here is the call graph for this function:



6.22.3.7 nodeCount()

```
unsigned int AtomProbe::K3DTreeApprox::nodeCount ( ) const [inline]
```

Print the number of nodes stored in the tree.

Definition at line 159 of file K3DTree-approx.h.

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

- [include/atomprobe/algorithm/K3DTree-approx.h](#)
- [src/algorithm/K3DTree-approx.cpp](#)

6.23 AtomProbe::K3DTreeExact Class Reference

3D specific KD tree

```
#include <K3DTree-exact.h>
```

Public Member Functions

- [K3DTreeExact \(\)](#)
KD Tree constructor. Tree is uninitialised at start.
- [~K3DTreeExact \(\)](#)
Cleans up tree, deallocates nodes.
- void [resetPts](#) (std::vector< [Point3D](#) > &pts, bool [clear](#)=true)
Supply points to KD tree. Output vector will be erased if clear=true.
- void [resetPts](#) (std::vector< [IonHit](#) > &pts, bool [clear](#)=true)
Supply points using IonHits to KD tree. Output vector will be erased if clear=true.
- bool [build](#) ()
Build the KD tree using the previously supplied points.
- void [getBoundCube](#) ([BoundCube](#) &b)
obtain the bounding rectangular prism volume for all elements in the KD tree
- void [dump](#) (std::ostream &, size_t depth=0, size_t offset=-1) const
Textual output of tree. Tabs are used to separate different levels of the tree.

- `size_t findNearestUntagged` (const `Point3D` &queryPt, const `BoundCube` &b, bool tag=true)
 - Find the nearest "untagged" point's internal index.*
- `void findUntaggedInRadius` (const `Point3D` &queryPt, const `BoundCube` &b, float radius, `std::vector< size_t >` &result)
 - Find untagged points within a given radius.*
- `void ptsInSphere` (const `Point3D` &origin, float radius, `std::vector< size_t >` &pts) const
- `const Point3D * getPt` (size_t index) const
- `const Point3D & getPtRef` (size_t index) const
- `void clearTags` (`std::vector< size_t >` &tagsToClear)
- `size_t getOrigIndex` (size_t treeIndex) const
- `void setCallback` (bool(*cb)(void))
- `void setProgressPointer` (unsigned int *p)
- `void clear` ()
- `void tag` (size_t treeIndex, bool tagVal=true)
- `void tag` (const `std::vector< size_t >` &treeIndices, bool tagVal=true)
- `bool getTag` (size_t treeIndex) const
- `size_t size` () const
- `size_t rootIdx` () const
- `size_t tagCount` () const
- `void clearAllTags` ()

6.23.1 Detailed Description

3D specific KD tree

Definition at line 54 of file `K3DTree-exact.h`.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 `K3DTreeExact()`

```
K3DTreeExact::K3DTreeExact ( )
```

KD Tree constructor. Tree is uninitialised at start.

Note : must set progress pointer and callback

Definition at line 65 of file `K3DTree-exact.cpp`.

6.23.2.2 `~K3DTreeExact()`

```
AtomProbe::K3DTreeExact::~~K3DTreeExact ( ) [inline]
```

Cleans up tree, deallocates nodes.

Definition at line 85 of file `K3DTree-exact.h`.

6.23.3.3 clearAllTags()

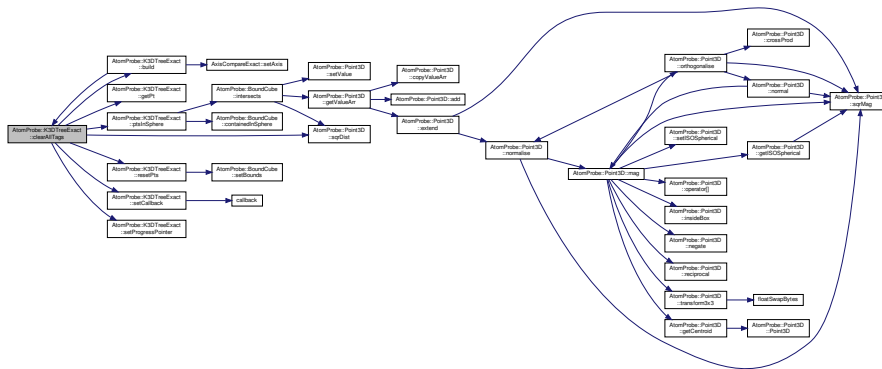
```
void K3DTreeExact::clearAllTags ( )
```

Definition at line 787 of file K3DTree-exact.cpp.

References ASSERT, build(), getPt(), ptsInSphere(), resetPts(), setCallback(), setProgressPointer(), AtomProbe::Point3D::sqrDist(), and TEST.

Referenced by build().

Here is the call graph for this function:



6.23.3.4 clearTags()

```
void K3DTreeExact::clearTags (
    std::vector< size_t > & tagsToClear )
```

Definition at line 780 of file K3DTree-exact.cpp.

6.23.3.5 dump()

```
void K3DTreeExact::dump (
    std::ostream & strm,
    size_t depth = 0,
    size_t offset = -1 ) const
```

Textual output of tree. Tabs are used to separate different levels of the tree.

The output from this function can be quite large for even modest trees. Recommended for debugging only

Definition at line 333 of file K3DTree-exact.cpp.

6.23.3.6 findNearestUntagged()

```
size_t K3DTreeExact::findNearestUntagged (
    const Point3D & queryPt,
    const BoundCube & b,
    bool tag = true )
```

Find the nearest "untagged" point's internal index.

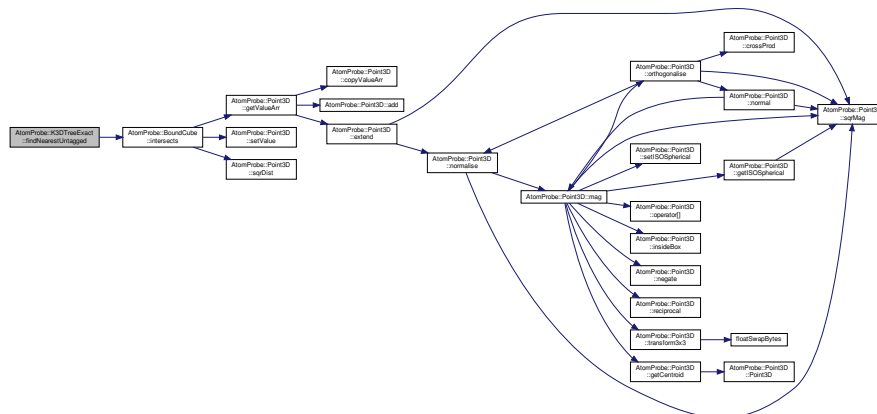
The point found is marked as "tagged" in the tree by default. -Returns -1 on failure (no untagged points), otherwise index of point

Definition at line 475 of file K3DTree-exact.cpp.

References ASSERT, and AtomProbe::BoundCube::intersects().

Referenced by findUntaggedInRadius(), and main().

Here is the call graph for this function:



6.23.3.7 findUntaggedInRadius()

```
void K3DTreeExact::findUntaggedInRadius (
    const Point3D & queryPt,
    const BoundCube & b,
    float radius,
    std::vector< size_t > & result )
```

Find untagged points within a given radius.

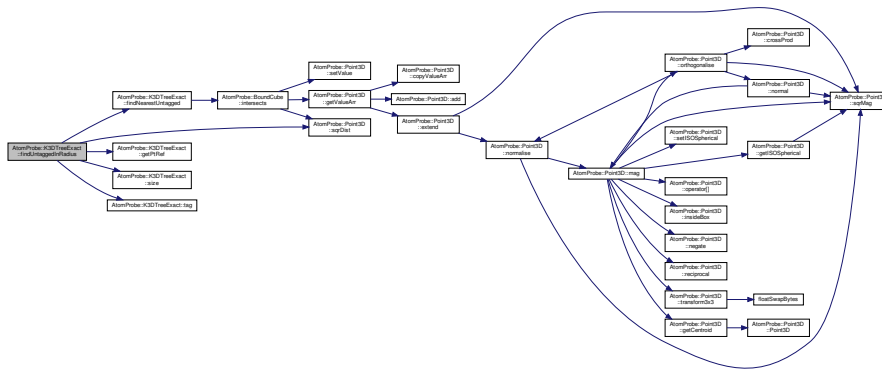
This is not thread safe - tags are read/written during operation

Definition at line 724 of file K3DTree-exact.cpp.

References ASSERT, findNearestUntagged(), getPtRef(), size(), AtomProbe::Point3D::sqrDist(), and tag().

Referenced by kdExactFuzz(), and main().

Here is the call graph for this function:



6.23.3.8 getBoundCube()

```
void K3DTreeExact::getBoundCube (
    BoundCube & b )
```

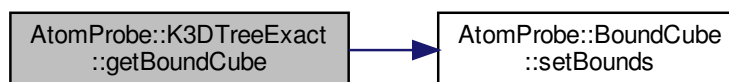
obtain the bounding rectangular prism volume for all elements in the KD tree

Definition at line 69 of file K3DTree-exact.cpp.

References ASSERT, and AtomProbe::BoundCube::setBounds().

Referenced by kdExactFuzz(), and main().

Here is the call graph for this function:



6.23.3.9 getOrigIndex()

```
size_t K3DTreeExact::getOrigIndex (
    size_t treeIndex ) const
```

Definition at line 87 of file K3DTree-exact.cpp.

References ASSERT.

Referenced by main().

6.23.3.10 getPt()

```
const Point3D * K3DTreeExact::getPt (
    size_t index ) const
```

Definition at line 75 of file K3DTree-exact.cpp.

References ASSERT.

Referenced by clearAllTags().

6.23.3.11 getPtRef()

```
const Point3D & K3DTreeExact::getPtRef (
    size_t index ) const
```

Definition at line 81 of file K3DTree-exact.cpp.

References ASSERT.

Referenced by findUntaggedInRadius(), AtomProbe::generate1DAxialDistHist(), AtomProbe::generate1DAxialDistHistSweep(), kdExactFuzz(), and main().

6.23.3.12 getTag()

```
bool K3DTreeExact::getTag (
    size_t treeIndex ) const
```

Definition at line 93 of file K3DTree-exact.cpp.

References ASSERT.

6.23.3.13 ptsInSphere()

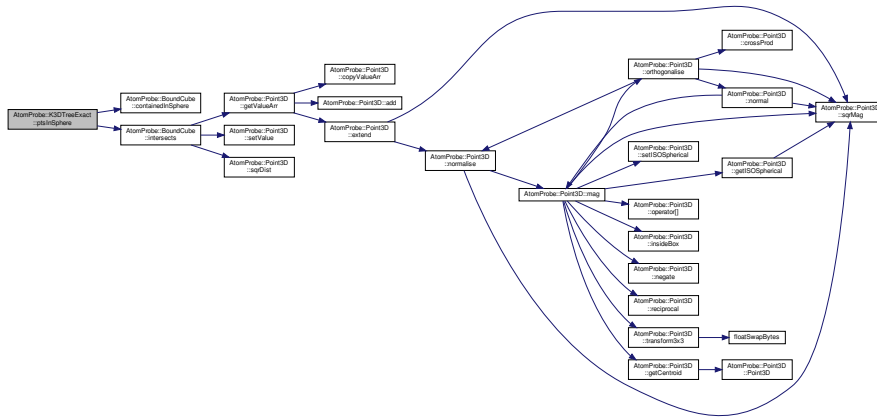
```
void K3DTreeExact::ptsInSphere (
    const Point3D & origin,
    float radius,
    std::vector< size_t > & pts ) const
```

Definition at line 379 of file K3DTree-exact.cpp.

References ASSERT, AtomProbe::BoundCube::containedInSphere(), and AtomProbe::BoundCube::intersects().

Referenced by clearAllTags(), AtomProbe::generate1DAxialDistHist(), AtomProbe::generate1DAxialDistHist←Sweep(), kdExactFuzz(), and main().

Here is the call graph for this function:



6.23.3.14 resetPts() [1/2]

```
void K3DTreeExact::resetPts (
    std::vector< Point3D > & pts,
    bool clear = true )
```

Supply points to KD tree. Ouput vector will be erased if clear=true.

Definition at line 119 of file K3DTree-exact.cpp.

References AtomProbe::BoundCube::setBounds().

Referenced by clearAllTags(), kdExactFuzz(), and main().

Here is the call graph for this function:



6.23.3.15 resetPts() [2/2]

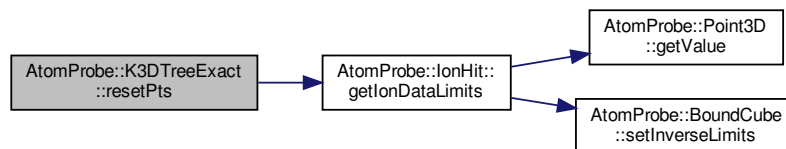
```
void K3DTreeExact::resetPts (
    std::vector< IonHit > & pts,
    bool clear = true )
```

Supply points using IonHits to KD tree. Ouput vector will be erased if clear=true.

Definition at line 138 of file K3DTree-exact.cpp.

References AtomProbe::IonHit::getIonDataLimits().

Here is the call graph for this function:



6.23.3.16 rootIdx()

```
size_t AtomProbe::K3DTreeExact::rootIdx ( ) const [inline]
```

Definition at line 163 of file K3DTree-exact.h.

6.23.3.17 setCallback()

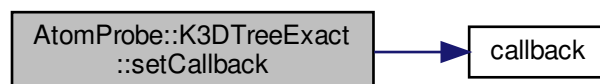
```
void AtomProbe::K3DTreeExact::setCallback (
    bool(*) (void) cb ) [inline]
```

Definition at line 143 of file K3DTree-exact.h.

References `callback()`.

Referenced by `clearAllTags()`, `kdExactFuzz()`, and `main()`.

Here is the call graph for this function:



6.23.3.18 setProgressPointer()

```
void AtomProbe::K3DTreeExact::setProgressPointer (
    unsigned int * p ) [inline]
```

Definition at line 146 of file K3DTree-exact.h.

Referenced by clearAllTags(), kdExactFuzz(), and main().

6.23.3.19 size()

```
size_t K3DTreeExact::size ( ) const
```

Definition at line 113 of file K3DTree-exact.cpp.

References ASSERT.

Referenced by findUntaggedInRadius(), AtomProbe::generate1DAxialDistHistSweep(), kdExactFuzz(), and main().

6.23.3.20 tag() [1/2]

```
void K3DTreeExact::tag (
    size_t treeIndex,
    bool tagVal = true )
```

Definition at line 99 of file K3DTree-exact.cpp.

References ASSERT.

Referenced by findUntaggedInRadius().

6.23.3.21 tag() [2/2]

```
void K3DTreeExact::tag (
    const std::vector< size_t > & treeIndices,
    bool tagVal = true )
```

Definition at line 105 of file K3DTree-exact.cpp.

References ASSERT.

6.23.3.22 tagCount()

```
size_t K3DTreeExact::tagCount ( ) const
```

Definition at line 765 of file K3DTree-exact.cpp.

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

- [include/atomprobe/algorithm/K3DTree-exact.h](#)
- [src/algorithm/K3DTree-exact.cpp](#)

6.24 AtomProbe::LibVersion Class Reference

Class to hold the library version data.

```
#include <version.h>
```

Public Member Functions

- [LibVersion](#) ()
- void [checkDebug](#) ()

Static Public Member Functions

- static unsigned int [getMajor](#) ()
- static unsigned int [getMinor](#) ()
- static unsigned int [getRevision](#) ()
- static std::string [getVersionStr](#) ()

Obtain the version of the program as a string.

6.24.1 Detailed Description

Class to hold the library version data.

Definition at line 19 of file version.h.

6.24.2 Constructor & Destructor Documentation

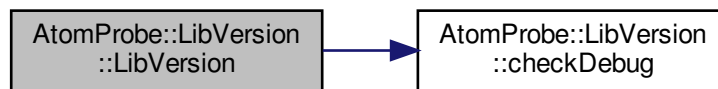
6.24.2.1 LibVersion()

```
AtomProbe::LibVersion::LibVersion ( ) [inline]
```

Definition at line 23 of file version.h.

References `checkDebug()`, `LIBATOMPROBE_MAJOR`, `LIBATOMPROBE_MINOR`, and `LIBATOMPROBE_REVISION`.

Here is the call graph for this function:



6.24.3 Member Function Documentation

6.24.3.1 checkDebug()

```
void AtomProbe::LibVersion::checkDebug ( )
```

Definition at line 39 of file atomprobe.cpp.

Referenced by `LibVersion()`.

6.24.3.2 getMajor()

```
static unsigned int AtomProbe::LibVersion::getMajor ( ) [inline], [static]
```

Definition at line 37 of file version.h.

References `LIBATOMPROBE_MAJOR`.

Referenced by `getVersionStr()`.

6.24.3.3 getMinor()

```
static unsigned int AtomProbe::LibVersion::getMinor ( ) [inline], [static]
```

Definition at line 38 of file version.h.

References LIBATOMPROBE_MINOR.

Referenced by getVersionStr().

6.24.3.4 getRevision()

```
static unsigned int AtomProbe::LibVersion::getRevision ( ) [inline], [static]
```

Definition at line 39 of file version.h.

References LIBATOMPROBE_REVISION.

Referenced by getVersionStr().

6.24.3.5 getVersionStr()

```
static std::string AtomProbe::LibVersion::getVersionStr ( ) [inline], [static]
```

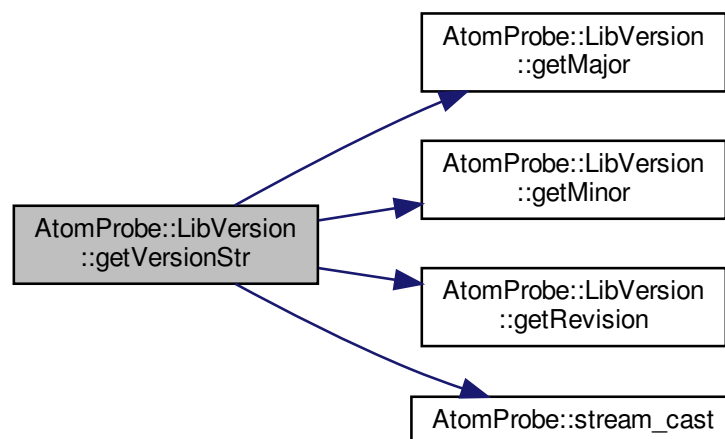
Obtain the version of the program as a string.

Definition at line 42 of file version.h.

References getMajor(), getMinor(), getRevision(), and AtomProbe::stream_cast().

Referenced by AtomProbe::MultiRange::write().

Here is the call graph for this function:



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

- [include/atomprobe/helper/version.h](#)
- [src/atomprobe.cpp](#)

6.25 AtomProbe::LINE Class Reference

```
#include <mesh.h>
```

Public Attributes

- unsigned int [p](#) [2]
- unsigned int [physGroup](#)

6.25.1 Detailed Description

Definition at line 66 of file mesh.h.

6.25.2 Member Data Documentation

6.25.2.1 p

```
unsigned int AtomProbe::LINE::p[2]
```

Definition at line 68 of file mesh.h.

Referenced by [AtomProbe::Mesh::loadGmshMesh\(\)](#).

6.25.2.2 physGroup

```
unsigned int AtomProbe::LINE::physGroup
```

Definition at line 69 of file mesh.h.

Referenced by [AtomProbe::Mesh::loadGmshMesh\(\)](#).

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

- [include/atomprobe/primitives/mesh.h](#)

6.26 AtomProbe::LinearFeedbackShiftReg Class Reference

This class implements a Linear Feedback Shift Register (in software)

```
#include <lfsr.h>
```


Public Member Functions

- `size_t clock ()`
Get a value from the shift register, and advance.
- `void setState (size_t newState)`
Set the internal lfsr state. Note 0 is the lock-up state.
- `void setMaskPeriod (unsigned int newMask)`
Set the mask to use such that the period is $2^n - 1$. 3 is minimum 60 is maximum.
- `bool verifyTable (size_t maxLen=0)`
Check the validity of the table.

6.26.1 Detailed Description

This class implements a Linear Feedback Shift Register (in software)

This is a mathematical construct based upon polynomials over closed natural numbers ($N \bmod p$). This will generate a weakly random digit string, but with guaranteed no duplicates, using $O(1)$ memory and $O(n)$ calls. The no duplicate guarantee is weak-ish, only guaranteeing no repetition in the shift register for $2^n - 1$ iterations. n can be set by `setMaskPeriod`. After this the sequence will repeat

Definition at line 30 of file `lfsr.h`.

6.26.2 Member Function Documentation

6.26.2.1 clock()

```
size_t AtomProbe::LinearFeedbackShiftReg::clock ( )
```

Get a value from the shift register, and advance.

Definition at line 135 of file `lfsr.cpp`.

Referenced by `AtomProbe::randomSelect()`, and `verifyTable()`.

6.26.2.2 setMaskPeriod()

```
void AtomProbe::LinearFeedbackShiftReg::setMaskPeriod (
    unsigned int newMask )
```

Set the mask to use such that the period is $2^n - 1$. 3 is minimum 60 is maximum.

Definition at line 81 of file `lfsr.cpp`.

References `ASSERT`.

Referenced by `AtomProbe::randomSelect()`, `setState()`, and `verifyTable()`.

6.26.2.3 setState()

```
void AtomProbe::LinearFeedbackShiftReg::setState (
    size_t newState ) [inline]
```

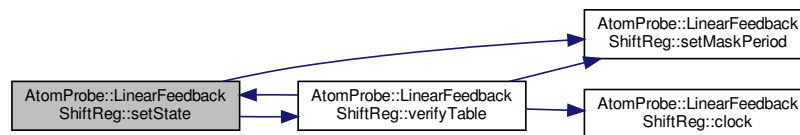
Set the internal lfsr state. Note 0 is the lock-up state.

Definition at line 39 of file lfsr.h.

References `setMaskPeriod()`, and `verifyTable()`.

Referenced by `AtomProbe::randomSelect()`, and `verifyTable()`.

Here is the call graph for this function:



6.26.2.4 verifyTable()

```
bool AtomProbe::LinearFeedbackShiftReg::verifyTable (
    size_t maxlen = 0 )
```

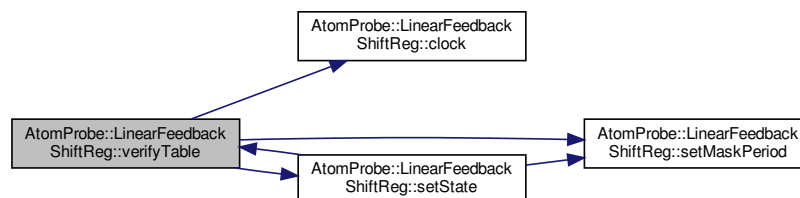
Check the validity of the table.

Definition at line 96 of file lfsr.cpp.

References `ASSERT`, `clock()`, `AtomProbe::maximumLinearTable`, `setMaskPeriod()`, and `setState()`.

Referenced by `setState()`.

Here is the call graph for this function:



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

- [include/atomprobe/helper/maths/lfsr.h](#)
- [src/helper/maths/lfsr.cpp](#)

6.27 AtomProbe::MassTool Class Reference

Class that brute-force solves the knapsack problem.

```
#include <massTool.h>
```

Static Public Member Functions

- static void [bruteKnapsack](#) (std::vector< [Weight](#) > &weights, float targetWeight, float tolerance, unsigned int maxObjects, std::vector< std::vector< [Weight](#) > > &solutions)
Depth-First-Search based brute force solve the knapsack problem,.
- static void [bruteKnapsack](#) (std::vector< [Weight](#) > &weights, const std::vector< float > &targetWeight, float tolerance, unsigned int maxObjects, std::vector< std::vector< [Weight](#) > > &solutions)
Multiple-target version of brute knapsack.

6.27.1 Detailed Description

Class that brute-force solves the knapsack problem.

Definition at line 43 of file massTool.h.

6.27.2 Member Function Documentation

6.27.2.1 [bruteKnapsack\(\)](#) [1/2]

```
void MassTool::bruteKnapsack (
    std::vector< Weight > & weights,
    float targetWeight,
    float tolerance,
    unsigned int maxObjects,
    std::vector< std::vector< Weight > > & solutions ) [static]
```

Depth-First-Search based brute force solve the knapsack problem,.

Record all solutions within "tolerance" of the value "targetWeight". Solutions are reported as std::vectors of weights. All solutions are ensured unique.

"weights" vector should contain the masses of the input object. Note that the elemIdx and isotopIdx values in the [Weight](#) structure are used for tracking purposes only. Elements may be removed from this input vector by [MassTool](#).

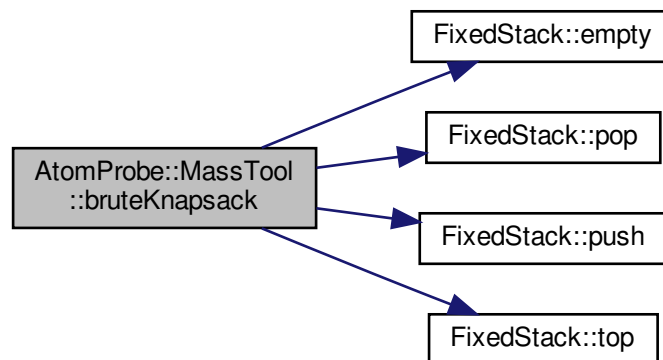
targetWeight is the total mass you want to find out what combination of weights matches. tolerance is the mass error from targetweight (+-tolerance) you are willing to accept maxObjects is the maximum number of weights you can combine to get the solution

Definition at line 117 of file massTool.cpp.

References WEIGHT_SEARCH_ENTRY::allowableWeights, WEIGHT_SEARCH_ENTRY::cumulativeWeight, WEIGHT_SEARCH_ENTRY::curWeights, FixedStack< T >::empty(), WEIGHT_SEARCH_ENTRY::offset, FixedStack< T >::pop(), FixedStack< T >::push(), and FixedStack< T >::top().

Referenced by main().

Here is the call graph for this function:



6.27.2.2 bruteKnapsack() [2/2]

```

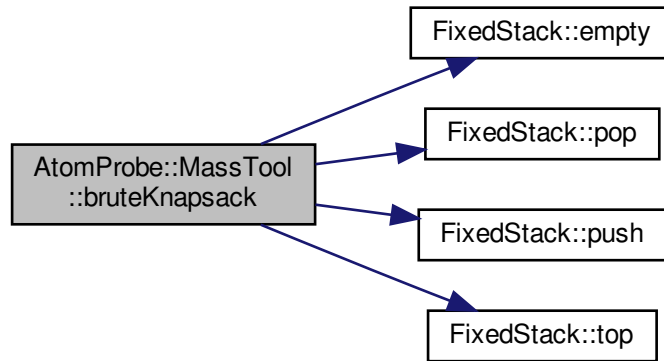
void MassTool::bruteKnapsack (
    std::vector< Weight > & weights,
    const std::vector< float > & targetWeight,
    float tolerance,
    unsigned int maxObjects,
    std::vector< std::vector< Weight > > & solutions ) [static]
  
```

Multiple-target version of brute knapsack.

Definition at line 220 of file massTool.cpp.

References WEIGHT_SEARCH_ENTRY::allowableWeights, WEIGHT_SEARCH_ENTRY::cumulativeWeight, WEIGHT_SEARCH_ENTRY::curWeights, FixedStack< T >::empty(), WEIGHT_SEARCH_ENTRY::offset, FixedStack< T >::pop(), FixedStack< T >::push(), TEST, and FixedStack< T >::top().

Here is the call graph for this function:



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

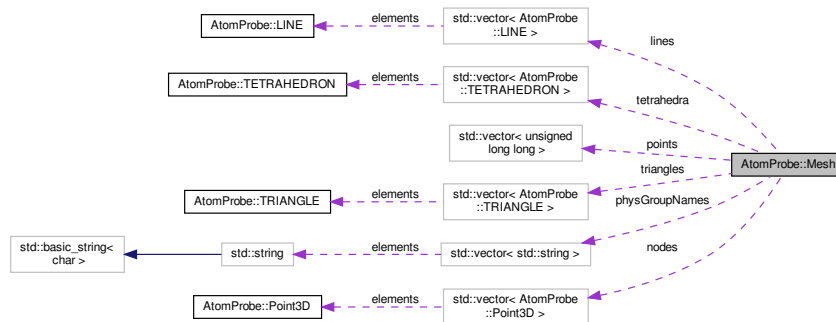
- [include/atomprobe/algorithm/massTool.h](#)
- [src/algorithm/massTool.cpp](#)

6.28 AtomProbe::Mesh Class Reference

Simple mesh class for storing and manipulating meshes consisting of 2 and 3D simplexes (triangles or tetrahedra)

```
#include <mesh.h>
```

Collaboration diagram for `AtomProbe::Mesh`:



Public Member Functions

- unsigned int [loadGmshMesh](#) (const char *meshfile, unsigned int &curLine, bool allowBadMeshes=true)
- unsigned int [saveGmshMesh](#) (const char *meshfile) const
- size_t [elementCount](#) () const
- void [setTriangleMesh](#) (const std::vector< float > &ptsA, const std::vector< float > &ptsB, const std::vector< float > &ptsC)
- void [reassignGroups](#) (unsigned int i)
reassign the physical groups to a single number
- void [removeDuplicateTris](#) ()
Remove exact duplicate triangles.
- void [removeStrayTris](#) ()
- void [mergeDuplicateVertices](#) (float tolerance)
- bool [isSane](#) (bool output=false, std::ostream &outStream=std::cerr) const
Perform various sanity tests on mesh. Should return true if your mesh is sane.
- void [getBounds](#) ([BoundCube](#) &b) const
Get the Axis aligned bounding box for this mesh.
- unsigned int [countTriNodes](#) () const
Count the number of unique nodes shared by triangles.
- void [translate](#) ()
Translate mesh around node centroid.
- void [translate](#) (const [Point3D](#) &origin)
Translate mesh to specified position.
- void [scale](#) (const [Point3D](#) &origin, float scaleFactor)
Scale the mesh around a specified origin.
- void [scale](#) (float scaleFactor)
Scale the mesh around origin.
- void [rotate](#) (const [Point3D](#) &axis, const [Point3D](#) &origin, float angle)
Rotate mesh.
- float [getVolume](#) () const
Obtain the volume of the triangulated space.
- void [resurface](#) (unsigned int newPhys)
place triangles over exposed tetrahedral faces
- void [clear](#) ()
Clear the mesh.
- void [getContainedNodes](#) (const [BoundCube](#) &b, std::vector< size_t > &nodes) const
Return all the nodes that are contained within specified bounding box.
- void [getIntersectingPrimitives](#) (std::vector< size_t > &searchNodes, std::vector< size_t > &lines, std::vector< size_t > &triangles, std::vector< size_t > &tetrahedra) const
Return all primitives that are WHOLLY contained withing bounding box.
- unsigned int [divideMeshSurface](#) (float divisionAngle, unsigned int newPhysGroupStart, const std::vector< size_t > &physGroupsToSplit)
- void [getCurPhysGroups](#) (std::vector< std::pair< unsigned int, size_t > > &curPhys) const
- void [erasePhysGroup](#) (unsigned int group, unsigned int typeMask)
- unsigned int [numDupVertices](#) (float tolerance) const
- unsigned int [numDupTris](#) () const
- void [print](#) (std::ostream &o) const
- bool [isOrientedCoherently](#) () const
- void [killOrphanNodes](#) ()
Kill specified orphan nodes within this dataset.
- void [pointsInside](#) (const std::vector< [Point3D](#) > &p, std::vector< bool > &meshResults, unsigned int &prog) const
- size_t [getNearestTri](#) (const [Point3D](#) &p, float &distance) const
- void [getTriNormal](#) (size_t tri, [Point3D](#) &normal) const

Public Attributes

- `std::vector< Point3D > nodes`
Point storage for 3D Data (nodes/coords/vertices..)
- `std::vector< std::string > physGroupNames`
Physical group storage.
- `std::vector< TETRAHEDRON > tetrahedra`
Storage for node connectivity in tetrahedral form.
- `std::vector< TRIANGLE > triangles`
Storage for node connectivity in triangle form (take in groups of 3)
- `std::vector< LINE > lines`
Storage for line segments. `.size()%2` should always==0.
- `std::vector< unsigned long long > points`

6.28.1 Detailed Description

Simple mesh class for storing and manipulating meshes consisting of 2 and 3D simplexes (triangles or tetrahedra)

Definition at line 73 of file mesh.h.

6.28.2 Member Function Documentation

6.28.2.1 `clear()`

```
void AtomProbe::Mesh::clear ( )
```

Clear the mesh.

Definition at line 922 of file mesh.cpp.

References `lines`, `nodes`, `physGroupNames`, `points`, `tetrahedra`, and `triangles`.

Referenced by `setTriangleMesh()`.

6.28.2.2 `countTriNodes()`

```
unsigned int AtomProbe::Mesh::countTriNodes ( ) const
```

Count the number of unique nodes shared by triangles.

Definition at line 1336 of file mesh.cpp.

References `triangles`.

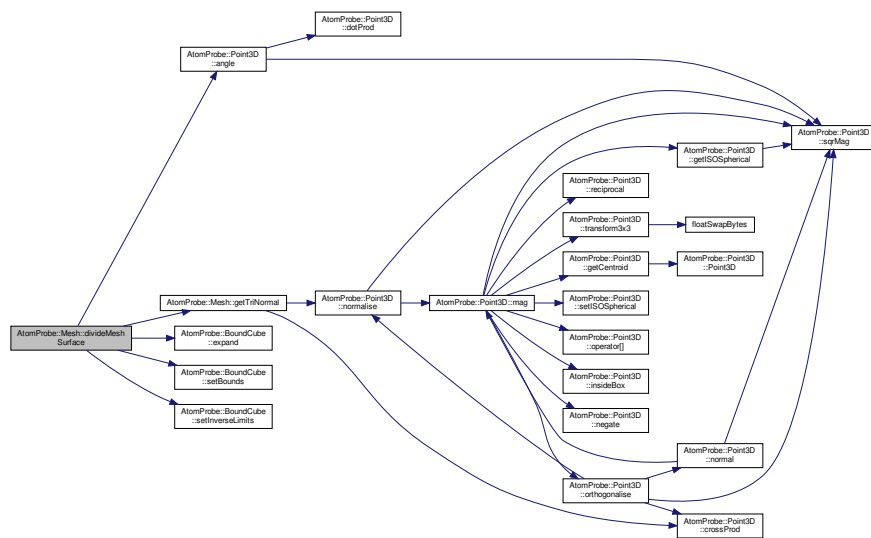
6.28.2.3 divideMeshSurface()

```
unsigned int AtomProbe::Mesh::divideMeshSurface (
    float divisionAngle,
    unsigned int newPhysGroupStart,
    const std::vector< size_t > & physGroupsToSplit )
```

Definition at line 1583 of file mesh.cpp.

References `AtomProbe::Point3D::angle()`, `ASSERT`, `AtomProbe::BoundCube::expand()`, `getTriNormal()`, `nodes`, `AtomProbe::BoundCube::setBounds()`, `AtomProbe::BoundCube::setInverseLimits()`, and `triangles`.

Here is the call graph for this function:



6.28.2.4 elementCount()

```
size_t AtomProbe::Mesh::elementCount ( ) const
```

Definition at line 1963 of file mesh.cpp.

References `lines`, `points`, `tetrahedra`, and `triangles`.

Referenced by `loadGmshMesh()`.

6.28.2.5 erasePhysGroup()

```
void AtomProbe::Mesh::erasePhysGroup (
    unsigned int group,
    unsigned int typeMask )
```

Definition at line 1866 of file mesh.cpp.

References `AtomProbe::ELEMENT_LINE`, `AtomProbe::ELEMENT_TETRAHEDRON`, `AtomProbe::ELEMENT_TRIANGLE`, `tetrahedra`, and `triangles`.

6.28.2.6 getBounds()

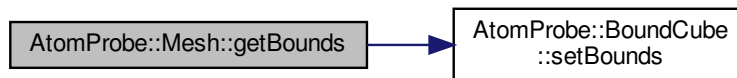
```
void AtomProbe::Mesh::getBounds (
    BoundCube & b ) const
```

Get the Axis aligned bounding box for this mesh.

Definition at line 1518 of file mesh.cpp.

References ASSERT, nodes, AtomProbe::BoundCube::setBounds(), and triangles.

Here is the call graph for this function:



6.28.2.7 getContainedNodes()

```
void AtomProbe::Mesh::getContainedNodes (
    const BoundCube & b,
    std::vector< size_t > & nodes ) const
```

Return all the nodes that are contained within specified bounding box.

Definition at line 1747 of file mesh.cpp.

References ASSERT, AtomProbe::BoundCube::containsPt(), and nodes.

Here is the call graph for this function:



6.28.2.8 getCurPhysGroups()

```
void AtomProbe::Mesh::getCurPhysGroups (
    std::vector< std::pair< unsigned int, size_t > > & curPhys ) const
```

Definition at line 1815 of file mesh.cpp.

References tetrahedra, and triangles.

6.28.2.9 getIntersectingPrimitives()

```
void AtomProbe::Mesh::getIntersectingPrimitives (
    std::vector< size_t > & searchNodes,
    std::vector< size_t > & lines,
    std::vector< size_t > & triangles,
    std::vector< size_t > & tetrahedra ) const
```

Return all primitives that are WHOLLY contained within bounding box.

Definition at line 1758 of file mesh.cpp.

References ASSERT, lines, tetrahedra, and triangles.

6.28.2.10 getNearestTri()

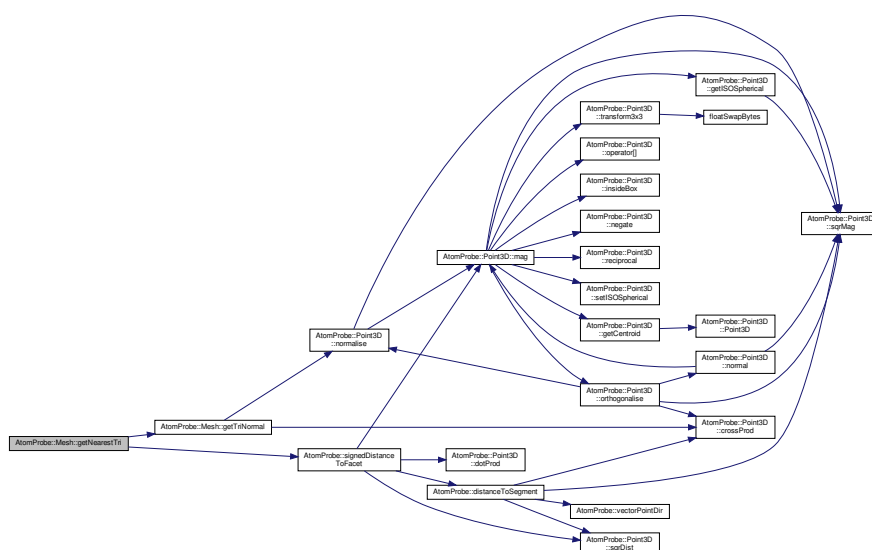
```
size_t AtomProbe::Mesh::getNearestTri (
    const Point3D & p,
    float & distance ) const
```

Definition at line 2105 of file mesh.cpp.

References getTriNormal(), nodes, AtomProbe::signedDistanceToFacet(), and triangles.

Referenced by AtomProbe::TRIANGLE::edgesMismatch(), and main().

Here is the call graph for this function:



6.28.2.11 getTriNormal()

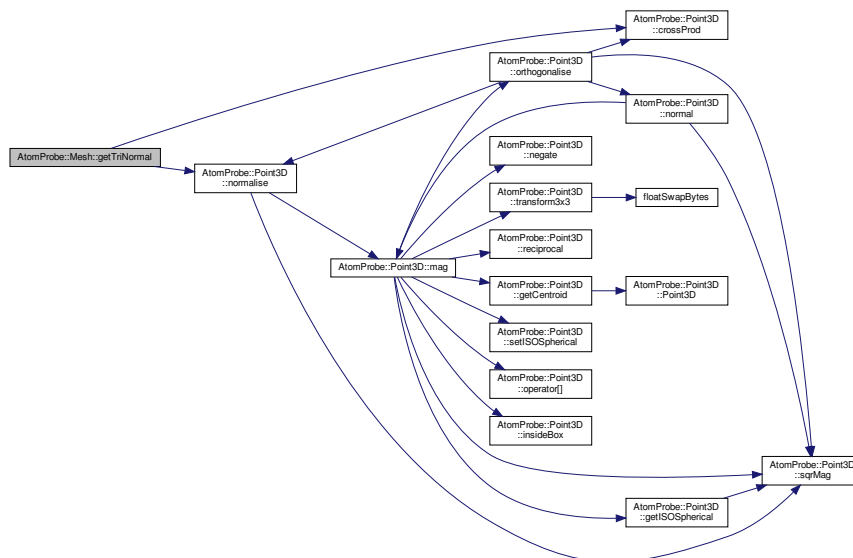
```
void AtomProbe::Mesh::getTriNormal (
    size_t tri,
    Point3D & normal ) const
```

Definition at line 1739 of file mesh.cpp.

References ASSERT, AtomProbe::Point3D::crossProd(), nodes, AtomProbe::Point3D::normalise(), and triangles.

Referenced by divideMeshSurface(), and getNearestTri().

Here is the call graph for this function:



6.28.2.12 getVolume()

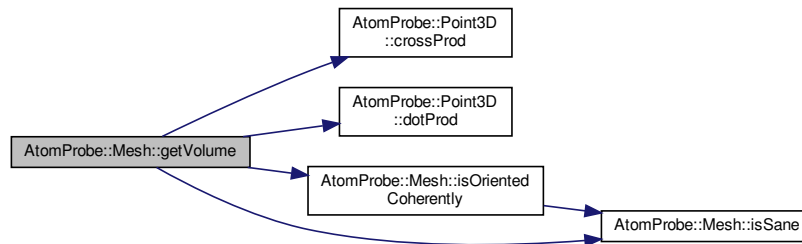
```
float AtomProbe::Mesh::getVolume ( ) const
```

Obtain the volume of the triangulated space.

Definition at line 1932 of file mesh.cpp.

References ASSERT, AtomProbe::Point3D::crossProd(), AtomProbe::Point3D::dotProd(), isOrientedCoherently(), isSane(), nodes, AtomProbe::TRIANGLE::p, and triangles.

Here is the call graph for this function:



6.28.2.13 isOrientedCoherently()

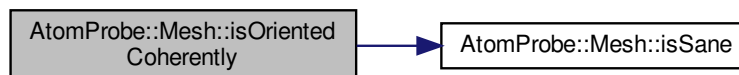
```
bool AtomProbe::Mesh::isOrientedCoherently ( ) const
```

Definition at line 2131 of file mesh.cpp.

References ASSERT, isSane(), and triangles.

Referenced by AtomProbe::TRIANGLE::edgesMismatch(), getVolume(), and pointsInside().

Here is the call graph for this function:



6.28.2.14 isSane()

```
bool AtomProbe::Mesh::isSane (
    bool output = false,
    std::ostream & outputStream = std::cerr ) const
```

Perform various sanity tests on mesh. Should return true if your mesh is sane.

Definition at line 572 of file mesh.cpp.

References lines, nodes, tetrahedra, and triangles.

Referenced by AtomProbe::TRIANGLE::edgesMismatch(), getVolume(), isOrientedCoherently(), killOrphanNodes(), loadGmshMesh(), mergeDuplicateVertices(), numDupTris(), print(), removeDuplicateTris(), saveGmshMesh(), and setTriangleMesh().

6.28.2.15 killOrphanNodes()

```
void AtomProbe::Mesh::killOrphanNodes ( )
```

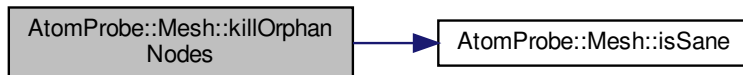
Kill specified orphan nodes within this dataset.

Definition at line 820 of file mesh.cpp.

References ASSERT, isSane(), lines, nodes, points, tetrahedra, and triangles.

Referenced by mergeDuplicateVertices(), and print().

Here is the call graph for this function:



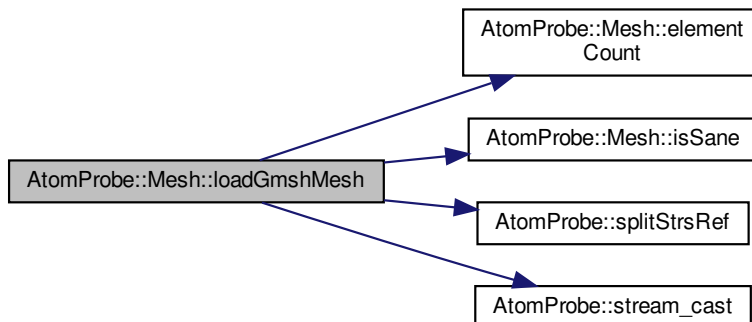
6.28.2.16 loadGmshMesh()

```
unsigned int AtomProbe::Mesh::loadGmshMesh (
    const char * meshfile,
    unsigned int & curLine,
    bool allowBadMeshes = true )
```

Definition at line 1025 of file mesh.cpp.

References ARRAYSIZE, AtomProbe::ELEM_FOUR_NODE_TETRAHEDRON, AtomProbe::ELEM_SINGLE_N↵
 ODE_POINT, AtomProbe::ELEM_THREE_NODE_TRIANGLE, AtomProbe::ELEM_TWO_NODE_LINE, element↵
 Count(), isSane(), lines, AtomProbe::MESH_LOAD_BAD_ELEMENTCOUNT, AtomProbe::MESH_LOAD_B↵
 AD_NODECOUNT, AtomProbe::MESH_LOAD_ENUM_END, AtomProbe::MESH_LOAD_IS_INSANE, nodes,
 AtomProbe::TETRAHEDRON::p, AtomProbe::TRIANGLE::p, AtomProbe::LINE::p, AtomProbe::TETRAHEDR↵
 ON::physGroup, AtomProbe::TRIANGLE::physGroup, AtomProbe::LINE::physGroup, points, AtomProbe::split↵
 StrsRef(), AtomProbe::stream_cast(), tetrahedra, and triangles.

Here is the call graph for this function:



6.28.2.17 mergeDuplicateVertices()

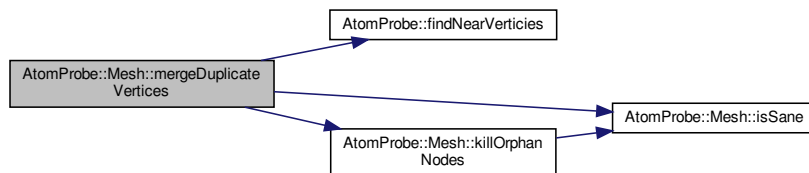
```
void AtomProbe::Mesh::mergeDuplicateVertices (
    float tolerance )
```

Definition at line 738 of file mesh.cpp.

References ASSERT, AtomProbe::findNearVertices(), isSane(), killOrphanNodes(), lines, nodes, points, tetrahedra, and triangles.

Referenced by main().

Here is the call graph for this function:



6.28.2.18 numDupTris()

```
unsigned int AtomProbe::Mesh::numDupTris ( ) const
```

Definition at line 854 of file mesh.cpp.

References ASSERT, isSane(), nodes, and triangles.

Here is the call graph for this function:



6.28.2.19 numDupVertices()

```
unsigned int AtomProbe::Mesh::numDupVertices (
    float tolerance ) const
```

Definition at line 897 of file mesh.cpp.

References nodes.

6.28.2.20 pointsInside()

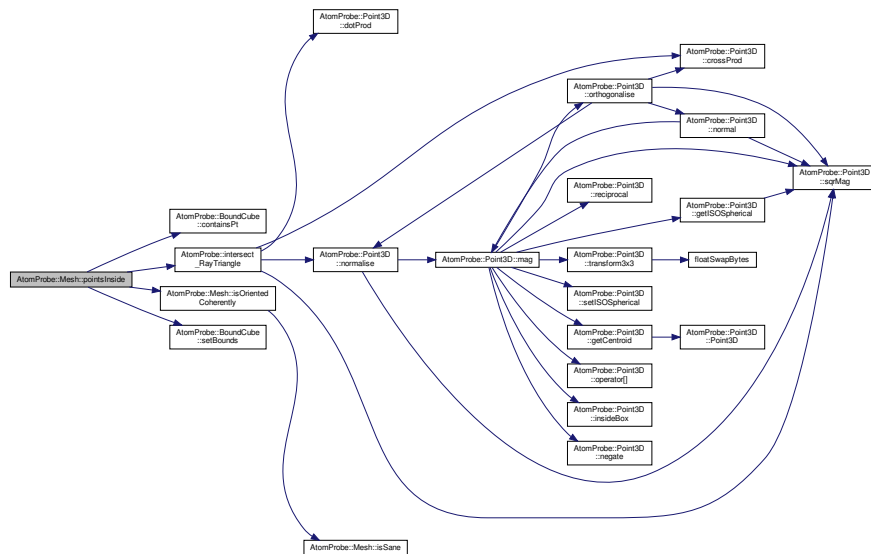
```
void AtomProbe::Mesh::pointsInside (
    const std::vector< Point3D > & p,
    std::vector< bool > & meshResults,
    unsigned int & prog ) const
```

Definition at line 1975 of file mesh.cpp.

References ASSERT, AtomProbe::BoundCube::containsPt(), AtomProbe::intersect_RayTriangle(), isOrientedCoherently(), nodes, AtomProbe::PROGRESS_REDUCE, AtomProbe::BoundCube::setBounds(), tetrahedra, and triangles.

Referenced by main().

Here is the call graph for this function:



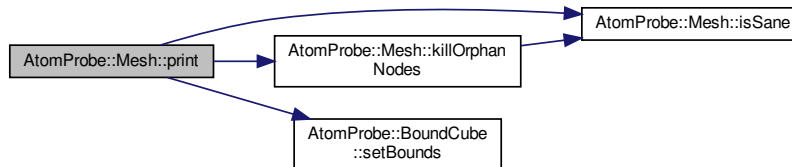
6.28.2.21 print()

```
void AtomProbe::Mesh::print (
    std::ostream & o ) const
```

Definition at line 445 of file mesh.cpp.

References ASSERT, isSane(), killOrphanNodes(), lines, nodes, AtomProbe::TRIANGLE::p, points, AtomProbe::BoundCube::setBounds(), tetrahedra, and triangles.

Here is the call graph for this function:



6.28.2.22 reassignGroups()

```
void AtomProbe::Mesh::reassignGroups (
    unsigned int i )
```

reassign the physical groups to a single number

Definition at line 1361 of file mesh.cpp.

References lines, tetrahedra, and triangles.

6.28.2.23 removeDuplicateTris()

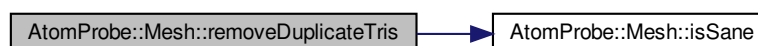
```
void AtomProbe::Mesh::removeDuplicateTris ( )
```

Remove exact duplicate triangles.

Definition at line 691 of file mesh.cpp.

References ASSERT, isSane(), nodes, and triangles.

Here is the call graph for this function:



6.28.2.24 removeStrayTris()

```
void AtomProbe::Mesh::removeStrayTris ( )
```

6.28.2.25 resurface()

```
void AtomProbe::Mesh::resurface (
    unsigned int newPhys )
```

place triangles over exposed tetrahedral faces

6.28.2.26 rotate()

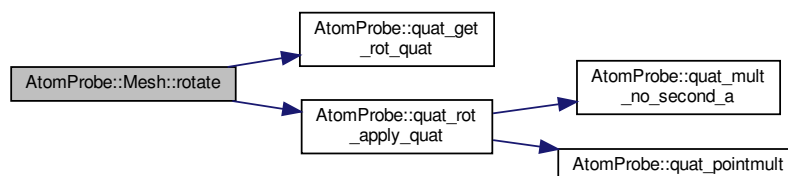
```
void AtomProbe::Mesh::rotate (
    const Point3D & axis,
    const Point3D & origin,
    float angle )
```

Rotate mesh.

Definition at line 1439 of file mesh.cpp.

References `AtomProbe::Point3f::fx`, `AtomProbe::Point3f::fy`, `AtomProbe::Point3f::fz`, `nodes`, `AtomProbe::quat_get_rot_quat()`, and `AtomProbe::quat_rot_apply_quat()`.

Here is the call graph for this function:



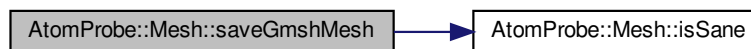
6.28.2.27 saveGmshMesh()

```
unsigned int AtomProbe::Mesh::saveGmshMesh (
    const char * meshfile ) const
```

Definition at line 1376 of file mesh.cpp.

References ASSERT, AtomProbe::ELEM_FOUR_NODE_TETRAHEDRON, AtomProbe::ELEM_SINGLE_NODE_POINT, AtomProbe::ELEM_THREE_NODE_TRIANGLE, AtomProbe::ELEM_TWO_NODE_LINE, isSane(), lines, nodes, points, tetrahedra, and triangles.

Here is the call graph for this function:



6.28.2.28 scale() [1/2]

```
void AtomProbe::Mesh::scale (
    const Point3D & origin,
    float scaleFactor )
```

Scale the mesh around a specified origin.

Definition at line 1500 of file mesh.cpp.

References nodes.

6.28.2.29 scale() [2/2]

```
void AtomProbe::Mesh::scale (
    float scaleFactor )
```

Scale the mesh around origin.

Definition at line 1511 of file mesh.cpp.

References nodes.

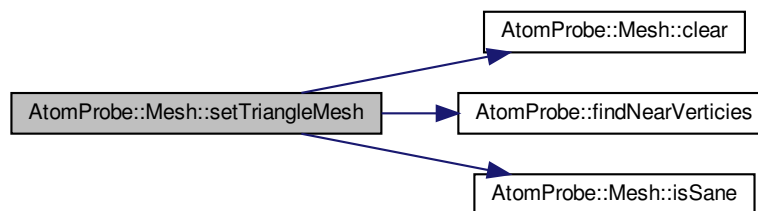
6.28.2.30 setTriangleMesh()

```
void AtomProbe::Mesh::setTriangleMesh (
    const std::vector< float > & ptsA,
    const std::vector< float > & ptsB,
    const std::vector< float > & ptsC )
```

Definition at line 932 of file mesh.cpp.

References ASSERT, clear(), AtomProbe::findNearVerticies(), isSane(), nodes, AtomProbe::TRIANGLE::p, and triangles.

Here is the call graph for this function:



6.28.2.31 translate() [1/2]

```
void AtomProbe::Mesh::translate ( )
```

Translate mesh around node centroid.

Definition at line 1482 of file mesh.cpp.

References nodes.

6.28.2.32 translate() [2/2]

```
void AtomProbe::Mesh::translate (
    const Point3D & origin )
```

Translate mesh to specified position.

Definition at line 1493 of file mesh.cpp.

References nodes.

6.28.3 Member Data Documentation

6.28.3.1 lines

```
std::vector<LINE> AtomProbe::Mesh::lines
```

Storage for line segments. `.size()%2` should always==0.

Definition at line 118 of file mesh.h.

Referenced by `clear()`, `elementCount()`, `getIntersectingPrimitives()`, `isSane()`, `killOrphanNodes()`, `loadGmshMesh()`, `mergeDuplicateVertices()`, `print()`, `reassignGroups()`, and `saveGmshMesh()`.

6.28.3.2 nodes

```
std::vector<Point3D> AtomProbe::Mesh::nodes
```

Point storage for 3D Data (nodes/coords/vertices..)

Definition at line 105 of file mesh.h.

Referenced by `clear()`, `divideMeshSurface()`, `AtomProbe::TRIANGLE::edgesMismatch()`, `getBounds()`, `get↔ContainedNodes()`, `getNearestTri()`, `getTriNormal()`, `getVolume()`, `isSane()`, `killOrphanNodes()`, `loadGmshMesh()`, `main()`, `mergeDuplicateVertices()`, `numDupTris()`, `numDupVertices()`, `pointsInside()`, `print()`, `removeDuplicateTris()`, `rotate()`, `saveGmshMesh()`, `scale()`, `setTriangleMesh()`, and `translate()`.

6.28.3.3 physGroupNames

```
std::vector<std::string> AtomProbe::Mesh::physGroupNames
```

Physical group storage.

Definition at line 108 of file mesh.h.

Referenced by `clear()`.

6.28.3.4 points

```
std::vector<unsigned long long> AtomProbe::Mesh::points
```

Definition at line 120 of file mesh.h.

Referenced by `clear()`, `elementCount()`, `killOrphanNodes()`, `loadGmshMesh()`, `mergeDuplicateVertices()`, `print()`, and `saveGmshMesh()`.

6.28.3.5 tetrahedra

```
std::vector<TETRAHEDRON> AtomProbe::Mesh::tetrahedra
```

Storage for node connectivity in tetrahedral form.

Definition at line 113 of file mesh.h.

Referenced by `clear()`, `elementCount()`, `erasePhysGroup()`, `getCurPhysGroups()`, `getIntersectingPrimitives()`, `isSane()`, `killOrphanNodes()`, `loadGmshMesh()`, `mergeDuplicateVertices()`, `pointsInside()`, `print()`, `reassignGroups()`, and `saveGmshMesh()`.

6.28.3.6 triangles

```
std::vector<TRIANGLE> AtomProbe::Mesh::triangles
```

Storage for node connectivity in triangle form (take in groups of 3)

Definition at line 116 of file mesh.h.

Referenced by `clear()`, `countTriNodes()`, `divideMeshSurface()`, `AtomProbe::TRIANGLE::edgesMismatch()`, `elementCount()`, `erasePhysGroup()`, `getBounds()`, `getCurPhysGroups()`, `getIntersectingPrimitives()`, `getNearestTri()`, `getTriNormal()`, `getVolume()`, `isOrientedCoherently()`, `isSane()`, `killOrphanNodes()`, `loadGmshMesh()`, `main()`, `mergeDuplicateVertices()`, `numDupTris()`, `pointsInside()`, `print()`, `reassignGroups()`, `removeDuplicateTris()`, `saveGmshMesh()`, and `setTriangleMesh()`.

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

- `include/atomprobe/primitives/mesh.h`
- `src/primitives/mesh.cpp`

6.29 AtomProbe::MILLER_TRIPLET Class Reference

```
#include <millerIndex.h>
```

Public Member Functions

- `MILLER_TRIPLET` (int a, int b, int c)
- `int h () const`
- `int k () const`
- `int l () const`
- `bool operator== (const MILLER_TRIPLET &m) const`
- `void simplify ()`
- `Point3D tripletToNormal` (unsigned int h, unsigned int k, unsigned int l)
- `Point3D tripletToNormal` (const MILLER_TRIPLET &m)

6.29.1 Detailed Description

Definition at line 20 of file millerIndex.h.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 MILLER_TRIPLET()

```
AtomProbe::MILLER_TRIPLET::MILLER_TRIPLET (
    int a,
    int b,
    int c )
```

Definition at line 39 of file millerIndex.cpp.

Referenced by operator==().

6.29.3 Member Function Documentation

6.29.3.1 h()

```
int AtomProbe::MILLER_TRIPLET::h ( ) const
```

Definition at line 46 of file millerIndex.cpp.

6.29.3.2 k()

```
int AtomProbe::MILLER_TRIPLET::k ( ) const
```

Definition at line 51 of file millerIndex.cpp.

6.29.3.3 l()

```
int AtomProbe::MILLER_TRIPLET::l ( ) const
```

Definition at line 56 of file millerIndex.cpp.

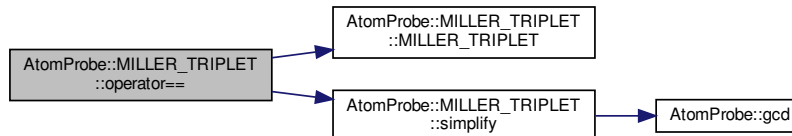
6.29.3.4 operator==()

```
bool AtomProbe::MILLER_TRIPLET::operator==(
    const MILLER_TRIPLET & m ) const
```

Definition at line 72 of file millerIndex.cpp.

References MILLER_TRIPLET(), simplify(), and TEST.

Here is the call graph for this function:



6.29.3.5 simplify()

```
void AtomProbe::MILLER_TRIPLET::simplify ( )
```

Definition at line 61 of file millerIndex.cpp.

References `AtomProbe::gcd()`.

Referenced by `operator==()`.

Here is the call graph for this function:



6.29.3.6 tripletToNormal() [1/2]

```
Point3D AtomProbe::MILLER_TRIPLET::tripletToNormal (
    unsigned int h,
    unsigned int k,
    unsigned int l )
```

6.29.3.7 tripletToNormal() [2/2]

```
Point3D AtomProbe::MILLER_TRIPLET::tripletToNormal (
    const MILLER_TRIPLET & m )
```

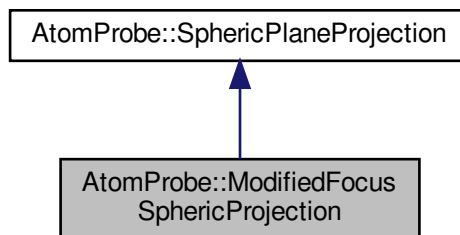
The documentation for this class was generated from the following files:

- [include/atomprobe/latticeplanes/millerIndex.h](#)
- [src/lattice/millerIndex.cpp](#)

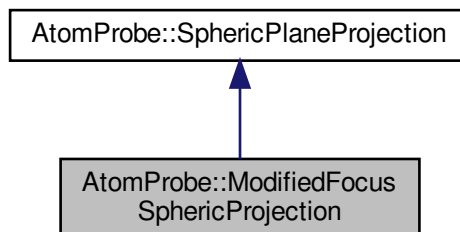
6.30 AtomProbe::ModifiedFocusSphericProjection Class Reference

```
#include <projection.h>
```

Inheritance diagram for AtomProbe::ModifiedFocusSphericProjection:



Collaboration diagram for AtomProbe::ModifiedFocusSphericProjection:



Public Member Functions

- [ModifiedFocusSphericProjection](#) (float focus)
Create a Spheric projection, with a focal point that is moved behind the sphere.
- void [setFocus](#) (float focus)
Set the focus position of the sphere. 0= sphere centre, 1=sphere end, > 1 is outside sphere.
- float [etaToTheta](#) (float eta) const
Convert the spherical angle to an azimuthal angle. See docs/figures for examples.
- float [thetaToEta](#) (float theta) const
Convert azimuthal angle to spherical. See etaToTheta for description.
- virtual bool [toAzimuthal](#) (float fx, float fy, float &theta, float &phi) const
Convert from plane coordinates to stereographic (NOT SPHERICAL) coords.
- virtual bool [toPlanar](#) (float theta, float phi, float &fx, float &fy) const
Convert from stereographic (NOT SPHERICAL) coordinates to plane.
- virtual void [scaleDown](#) (float flightLength, float detX, float detY, float &scaledX, float &scaledY) const
Convert from actual detector position (eg. mm) and flight path to scaled-down transform.
- virtual void [scaleUp](#) (float flightLength, float scaledX, float scaledY, float &realX, float &realY) const
Convert from scaled-down to actual dimension.
- float [getMaxFOV](#) () const
Obtain angle spherical angle, eta, (i.e. angle from point of sphere, sphere centre and x axis) of the line that is tangent to the sphere.
- float [getFOVRadius](#) (float eta) const
Obtain the radius in the XY plane of the sphere-angle (eta) that gives us a given FOV.

6.30.1 Detailed Description

Definition at line 95 of file projection.h.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 ModifiedFocusSphericProjection()

```
AtomProbe::ModifiedFocusSphericProjection::ModifiedFocusSphericProjection (
    float focus )
```

Create a Spheric projection, with a focal point that is moved behind the sphere.

Definition at line 202 of file projection.cpp.

6.30.3 Member Function Documentation

6.30.3.1 etaToTheta()

```
float AtomProbe::ModifiedFocusSphericProjection::etaToTheta (
    float eta ) const
```

Convert the spherical angle to an azimuthal angle. See docs/figures for examples.

Definition at line 216 of file projection.cpp.

References ASSERT, and AtomProbe::SphericProjectionParams::focusDist.

Referenced by main(), and scaleUp().

6.30.3.2 getFOVRadius()

```
float AtomProbe::ModifiedFocusSphericProjection::getFOVRadius (
    float eta ) const
```

Obtain the radius in the XY plane of the sphere-angle (eta) that gives us a given FOV.

Definition at line 288 of file projection.cpp.

References AtomProbe::SphericProjectionParams::focusDist, and AtomProbe::SphericProjectionParams::theta.

6.30.3.3 getMaxFOV()

```
float AtomProbe::ModifiedFocusSphericProjection::getMaxFOV ( ) const
```

Obtain angle spherical angle, eta, (i.e. angle from point of sphere, sphere centre and x axis) of the line that is tangent to the sphere.

Beyond this there is no intersection, and no solution.

Definition at line 270 of file projection.cpp.

References AtomProbe::SphericProjectionParams::focusDist.

Referenced by main(), and scaleUp().

6.30.3.4 scaleDown()

```
void AtomProbe::ModifiedFocusSphericProjection::scaleDown (
    float flightLength,
    float detX,
    float detY,
    float & scaledX,
    float & scaledY ) const [virtual]
```

Convert from actual detector position (eg. mm) and flight path to scaled-down transform.

Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 339 of file projection.cpp.

References ASSERT, and AtomProbe::SphericProjectionParams::focusDist.

6.30.3.5 scaleUp()

```
void AtomProbe::ModifiedFocusSphericProjection::scaleUp (
    float flightLength,
    float scaledX,
    float scaledY,
    float & realX,
    float & realY ) const [virtual]
```

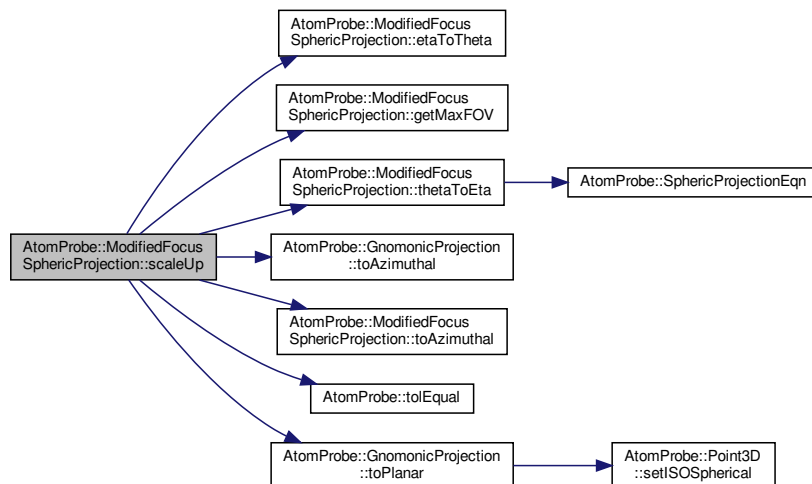
Convert from scaled-down to actual dimension.

Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 350 of file projection.cpp.

References ASSERT, etaToTheta(), AtomProbe::SphericProjectionParams::focusDist, getMaxFOV(), M_PI, T←EST, AtomProbe::SphericProjectionParams::theta, thetaToEta(), AtomProbe::GnomonicProjection::toAzimuthal(), toAzimuthal(), AtomProbe::tolEqual(), and AtomProbe::GnomonicProjection::toPlanar().

Here is the call graph for this function:



6.30.3.6 setFocus()

```
void AtomProbe::ModifiedFocusSphericProjection::setFocus (
    float focus )
```

Set the focus position of the sphere. 0= sphere centre, 1=sphere end, >1 is outside sphere.

Definition at line 207 of file projection.cpp.

References `AtomProbe::SphericProjectionParams::focusDist`.

6.30.3.7 thetaToEta()

```
float AtomProbe::ModifiedFocusSphericProjection::thetaToEta (
    float theta ) const
```

Convert azimuthal angle to spherical. See `etaToTheta` for description.

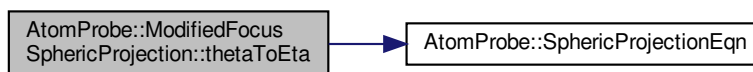
input range is limited to `[0, getMaxFOV()]`

Definition at line 225 of file projection.cpp.

References `AtomProbe::SphericProjectionParams::focusDist`, `M_PI`, `AtomProbe::SphericProjectionEqn()`, and `AtomProbe::SphericProjectionParams::theta`.

Referenced by `scaleUp()`.

Here is the call graph for this function:



6.30.3.8 toAzimuthal()

```
bool AtomProbe::ModifiedFocusSphericProjection::toAzimuthal (
    float fx,
    float fy,
    float & theta,
    float & phi ) const [virtual]
```

Convert from plane coordinates to stereographic (NOT SPHERICAL) coords.

Implements `AtomProbe::SphericPlaneProjection`.

Definition at line 324 of file projection.cpp.

References `AtomProbe::SphericProjectionParams::focusDist`.

Referenced by `scaleUp()`.

6.30.3.9 toPlanar()

```
bool AtomProbe::ModifiedFocusSphericProjection::toPlanar (
    float theta,
    float phi,
    float & fx,
    float & fy ) const [virtual]
```

Convert from stereographic (NOT SPHERICAL) coordinates to plane.

Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 310 of file projection.cpp.

Referenced by main().

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

- include/atomprobe/projection/[projection.h](#)
- src/projection/[projection.cpp](#)

6.31 AtomProbe::MultiRange Class Reference

Data storage and retrieval class for "ranging" a spectra, where overlapping ranges are permitted.

```
#include <multiRange.h>
```

Public Member Functions

- [MultiRange](#) ()
- [MultiRange](#) (const [RangeFile](#) &rng, const [AbundanceData](#) &natData)
 - Do our best to intialise from a rangefile.*
- unsigned int [addIon](#) (const std::set< [SIMPLE_SPECIES](#) > &molecule, const std::string &name, const [RGBf](#) &ionCol)
 - Add the ion to the database returns ion ID if successful, -1 otherwise.*
- unsigned int [addIon](#) (const [SIMPLE_SPECIES](#) &molecule, const std::string &name, const [RGBf](#) &ionCol)
 - Add a simple ion to the database - returns ion ID if successful, -1 otherwise.*
- unsigned int [addRange](#) (float start, float end, unsigned int ionID)
 - Add a range to the rangefile. Returns ID number of added range if adding successful, (unsigned int)-1 otherwise.*
- unsigned int [addRange](#) (const std::pair< float, float > &rng, unsigned int ionID)
 - Convenience wrapper for [addRange\(float,float, unsigned int\)](#)*
- void [setRangeGroups](#) (const std::vector< unsigned int > &groups)
 - Set the groupings for the ranges.*
- bool [write](#) (const char *fileName, unsigned int format=[MULTIRANGE_FORMAT_XML](#)) const
 - Save the structure to a file. format specifies output file format type.*
- bool [open](#) (const char *fileName, unsigned int format=-1)
 - read the contents of the file into class*
- unsigned int [getIonID](#) (unsigned int rangeld) const
 - Get the ion's ID from a specified mass.*
- void [setIonID](#) (unsigned int range, unsigned int newIonId)

- Set the ion ID for a given range.*

 - `RGBf getColour` (unsigned int ionID) const

Obtain the colour for a given ion.

 - void `setColour` (unsigned int, const `RGBf` &r)

Set the colour using the ion ID.

 - `std::string getErrString` () const

Retrieve the human-readable error associated with the current range file state.

 - `std::string getIonName` (unsigned int ionID) const

Get the name of a specified ionID.

 - unsigned int `getNumRanges` () const

Get the number of unique ranges.

 - unsigned int `getNumRanges` (unsigned int ionID) const

Get the number of ranges for a given ion ID.

 - unsigned int `getNumIons` () const

Get the number of unique ions.

 - `std::set< SIMPLE_SPECIES > getMolecule` (unsigned int ionID) const

Return the molecule that is associated with this ion.

 - `std::pair< float, float > getRange` (unsigned int rangeID) const

Retrieve the start and end of a given range as a pair(start,end)

 - `std::vector< unsigned int > getRanges` (float mass) const

Retrieve all of the ranges that specify the given mass.

 - `std::vector< unsigned int > getRangesFromIon` (unsigned int ionID) const

Retrieve all the ranges associated with a given ion.

 - bool `isRanged` (float mass) const

Returns true if a specified mass is ranged.

 - bool `isRanged` (const `IonHit` &) const

Returns true if an ion is ranged.

 - void `range` (`std::vector< IonHit >` &ionHits) const

Clips out ions that are not inside the rangefile's ranges.

 - bool `isSelfConsistent` () const

Check to see if data structure is internally consistent.

 - void `clear` ()

Erase the contents of the rangefile.

 - void `flattenToMassAxis` (`std::vector< FLATTENED_RANGE >` &ionMapping) const

Obtain a projection onto the mass axis of ranges that do not touch one another.

 - void `splitOverlapping` (`std::vector< MultiRange >` &decomposedRanges) const
 - void `copyDataFromRange` (const `MultiRange` &src, unsigned int srcRngID)

Copy range from a different multirange into this one, including all dependant data.

6.31.1 Detailed Description

Data storage and retrieval class for "ranging" a spectra, where overlapping ranges are permitted.

Definition at line 63 of file multiRange.h.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 MultiRange() [1/2]

```
AtomProbe::MultiRange::MultiRange ( )
```

Definition at line 136 of file multiRange.cpp.

6.31.2.2 MultiRange() [2/2]

```
AtomProbe::MultiRange::MultiRange (
    const RangeFile & rng,
    const AbundanceData & natData )
```

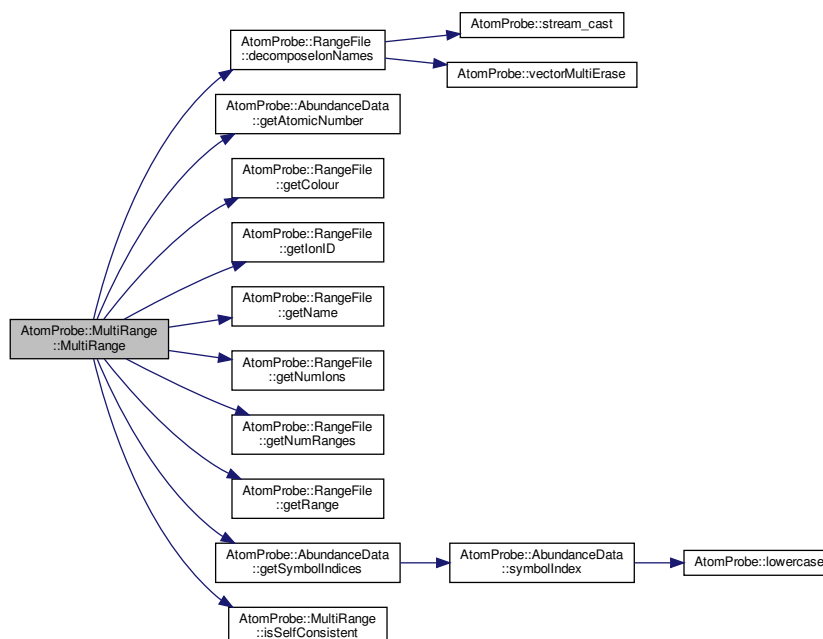
Do our best to initialise from a rangefile.

ions and their child ranges may be dropped if they cannot be identified

Definition at line 141 of file multiRange.cpp.

References ASSERT, AtomProbe::SIMPLE_SPECIES::atomicNumber, AtomProbe::SIMPLE_SPECIES::count, AtomProbe::RangeFile::decomposeIonNames(), AtomProbe::AbundanceData::getAtomicNumber(), AtomProbe::RangeFile::getColour(), AtomProbe::RangeFile::getIonID(), AtomProbe::RangeFile::getName(), AtomProbe::RangeFile::getNumIons(), AtomProbe::RangeFile::getNumRanges(), AtomProbe::RangeFile::getRange(), AtomProbe::AbundanceData::getSymbolIndices(), and isSelfConsistent().

Here is the call graph for this function:



6.31.3 Member Function Documentation

6.31.3.1 addlon() [1/2]

```
unsigned int AtomProbe::MultiRange::addIon (
    const std::set< SIMPLE_SPECIES > & molecule,
    const std::string & name,
    const RGBF & ionCol )
```

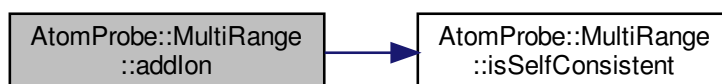
Add the ion to the database returns ion ID if successful, -1 otherwise.

Definition at line 250 of file multiRange.cpp.

References ASSERT, and isSelfConsistent().

Referenced by addlon(), AtomProbe::computeRangeAdjacency(), copyDataFromRange(), and AtomProbe::leastSquaresDeconvolve().

Here is the call graph for this function:



6.31.3.2 addlon() [2/2]

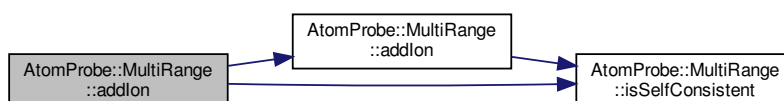
```
unsigned int AtomProbe::MultiRange::addIon (
    const SIMPLE_SPECIES & molecule,
    const std::string & name,
    const RGBF & ionCol )
```

Add a simple ion to the database - returns ion ID if successful, -1 otherwise.

Definition at line 272 of file multiRange.cpp.

References addlon(), ASSERT, and isSelfConsistent().

Here is the call graph for this function:



6.31.3.3 addRange() [1/2]

```
unsigned int AtomProbe::MultiRange::addRange (
    float start,
    float end,
    unsigned int ionID )
```

Add a range to the rangefile. Returns ID number of added range if adding successful, (unsigned int)-1 otherwise.

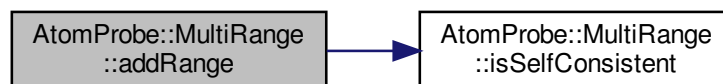
Note that ranges are, unlike [RangeFile](#), allowed to overlap, and still be self consistent.

Definition at line 282 of file multiRange.cpp.

References ASSERT, and isSelfConsistent().

Referenced by AtomProbe::computeRangeAdjacency(), copyDataFromRange(), and AtomProbe::leastSquares↔Deconvolve().

Here is the call graph for this function:



6.31.3.4 addRange() [2/2]

```
unsigned int AtomProbe::MultiRange::addRange (
    const std::pair< float, float > & rng,
    unsigned int ionID )
```

Convenience wrapper for [addRange\(float,float, unsigned int\)](#)

6.31.3.5 clear()

```
void AtomProbe::MultiRange::clear ( )
```

Erase the contents of the rangefile.

Definition at line 806 of file multiRange.cpp.

Referenced by open().

6.31.3.6 copyDataFromRange()

```
void AtomProbe::MultiRange::copyDataFromRange (
    const MultiRange & src,
    unsigned int srcRngId )
```

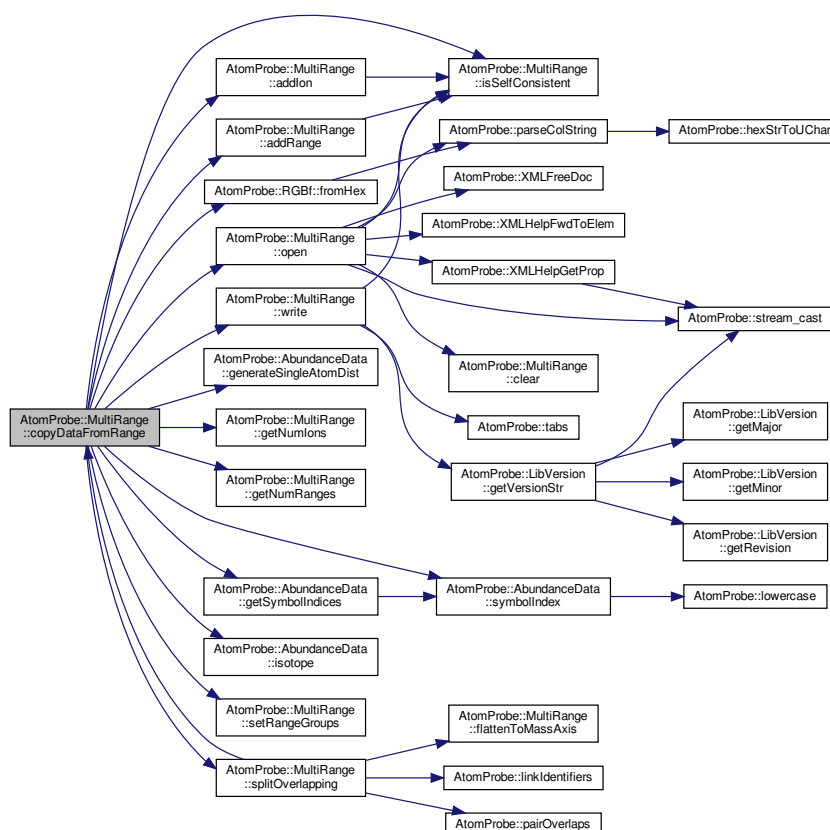
Copy range from a different multirange into this one, including all dependant data.

Definition at line 957 of file multiRange.cpp.

References `addlon()`, `addRange()`, `ASSERT`, `AtomProbe::SIMPLE_SPECIES::atomicNumber`, `AtomProbe::ISOT_OPE_ENTRY::atomicNumber`, `AtomProbe::RGBf::blue`, `AtomProbe::SIMPLE_SPECIES::count`, `AtomProbe::RGBf::fromHex()`, `AtomProbe::AbundanceData::generateSingleAtomDist()`, `getNumlons()`, `getNumRanges()`, `AtomProbe::AbundanceData::getSymbolIndices()`, `AtomProbe::RGBf::green`, `AtomProbe::AbundanceData::isotope()`, `isSelfConsistent()`, `MASS_TOL`, `open()`, `AtomProbe::RGBf::red`, `setRangeGroups()`, `splitOverlapping()`, `AtomProbe::AbundanceData::symbolIndex()`, `TEST`, and `write()`.

Referenced by `splitOverlapping()`.

Here is the call graph for this function:



6.31.3.7 flattenToMassAxis()

```
void AtomProbe::MultiRange::flattenToMassAxis (
    std::vector< FLATTENED_RANGE > & ionMapping ) const
```

Obtain a projection onto the mass axis of ranges that do not touch one another.

Definition at line 889 of file multiRange.cpp.

References ASSERT, and AtomProbe::FLATTENED_RANGE::startMass.

Referenced by splitOverlapping().

6.31.3.8 getColour()

```
RGBf AtomProbe::MultiRange::getColour (
    unsigned int ionID ) const
```

Obtain the colour for a given ion.

Definition at line 765 of file multiRange.cpp.

References ASSERT.

6.31.3.9 getErrString()

```
std::string AtomProbe::MultiRange::getErrString ( ) const
```

Retrieve the human-readable error associated with the current range file state.

Definition at line 671 of file multiRange.cpp.

References ASSERT, and AtomProbe::MULTIRANGE_ERR_ENUM_END.

6.31.3.10 getIonID()

```
unsigned int AtomProbe::MultiRange::getIonID (
    unsigned int rangeId ) const
```

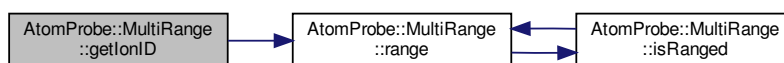
Get the ion's ID from a specified mass.

Returns the ions ID if there exists a range that contains this mass. Otherwise (unsigned int)-1 is returned

Definition at line 732 of file multiRange.cpp.

References ASSERT, and range().

Here is the call graph for this function:



6.31.3.11 `getIonName()`

```
std::string AtomProbe::MultiRange::getIonName (
    unsigned int ionID ) const
```

Get the name of a specified ionID.

Definition at line 713 of file multiRange.cpp.

References ASSERT.

Referenced by `AtomProbe::leastSquaresDeconvolve()`.

6.31.3.12 `getMolecule()`

```
set< SIMPLE_SPECIES > AtomProbe::MultiRange::getMolecule (
    unsigned int ionID ) const
```

Return the molecule that is associated with this ion.

Definition at line 707 of file multiRange.cpp.

References ASSERT.

Referenced by `AtomProbe::computeIonDistAdjacency()`.

6.31.3.13 `getNumIons()`

```
unsigned int AtomProbe::MultiRange::getNumIons ( ) const
```

Get the number of unique ions.

Definition at line 727 of file multiRange.cpp.

Referenced by `AtomProbe::computeIonDistAdjacency()`, `copyDataFromRange()`, and `AtomProbe::leastSquaresDeconvolve()`.

6.31.3.14 `getNumRanges()` [1/2]

```
unsigned int AtomProbe::MultiRange::getNumRanges ( ) const
```

Get the number of unique ranges.

Definition at line 700 of file multiRange.cpp.

Referenced by `AtomProbe::computeRangeAdjacency()`, `copyDataFromRange()`, and `AtomProbe::leastSquaresDeconvolve()`.

6.31.3.15 getNumRanges() [2/2]

```
unsigned int AtomProbe::MultiRange::getNumRanges (
    unsigned int ionID ) const
```

Get the number of ranges for a given ion ID.

Definition at line 719 of file multiRange.cpp.

References ASSERT.

6.31.3.16 getRange()

```
std::pair< float, float > AtomProbe::MultiRange::getRange (
    unsigned int rangeID ) const
```

Retrieve the start and end of a given range as a pair(start,end)

Definition at line 739 of file multiRange.cpp.

References ASSERT.

Referenced by AtomProbe::computeRangeAdjacency().

6.31.3.17 getRangeIdsFromIon()

```
std::vector<unsigned int> AtomProbe::MultiRange::getRangeIdsFromIon (
    unsigned int ionId ) const
```

Retrieve all the ranges associated with a given ion.

6.31.3.18 getRanges()

```
std::vector< unsigned int > AtomProbe::MultiRange::getRanges (
    float mass ) const
```

Retrieve all of the ranges that specify the given mass.

Definition at line 745 of file multiRange.cpp.

6.31.3.19 `isRanged()` [1/2]

```
bool AtomProbe::MultiRange::isRanged (
    float mass ) const
```

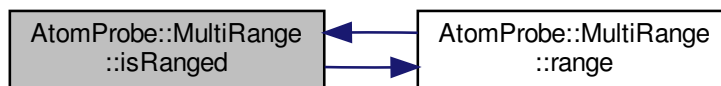
Returns true if a specified mass is ranged.

Definition at line 772 of file multiRange.cpp.

References `range()`.

Referenced by `isRanged()`, and `range()`.

Here is the call graph for this function:

6.31.3.20 `isRanged()` [2/2]

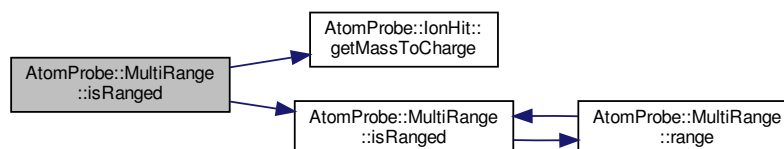
```
bool AtomProbe::MultiRange::isRanged (
    const IonHit & ion ) const
```

Returns true if an ion is ranged.

Definition at line 784 of file multiRange.cpp.

References `AtomProbe::IonHit::getMassToCharge()`, and `isRanged()`.

Here is the call graph for this function:



6.31.3.21 isSelfConsistent()

```
bool AtomProbe::MultiRange::isSelfConsistent ( ) const
```

Check to see if data structure is internally consistent.

Definition at line 209 of file multiRange.cpp.

Referenced by addlon(), addRange(), copyDataFromRange(), MultiRange(), open(), and write().

6.31.3.22 open()

```
bool AtomProbe::MultiRange::open (
    const char * fileName,
    unsigned int format = -1 )
```

read the contents of the file into class

returns true if successful, otherwise false

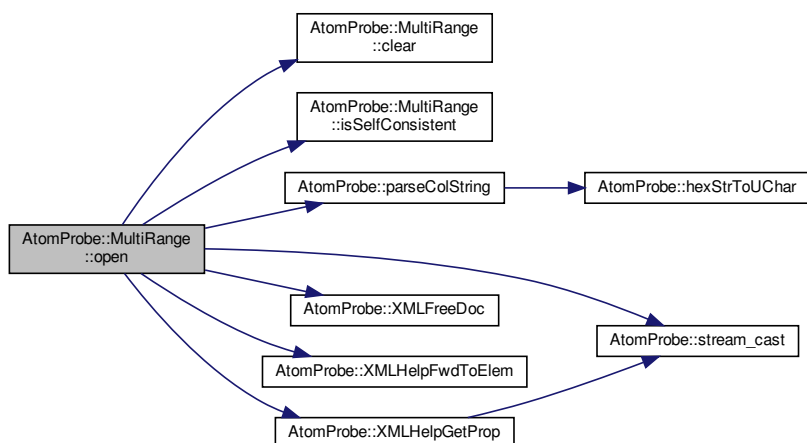
- if not successful then the errState will contain the error information, and the calling object (rangefile) will be in an invalid state

Definition at line 376 of file multiRange.cpp.

References AtomProbe::SIMPLE_SPECIES::atomicNumber, AtomProbe::RGBf::blue, clear(), AtomProbe::SIMPLE_SPECIES::count, AtomProbe::RGBf::green, isSelfConsistent(), AtomProbe::MULTIRANGE_ERR_BAD_GROUP, AtomProbe::MULTIRANGE_ERR_BAD_RANGE, AtomProbe::MULTIRANGE_ERR_BAD_RANGES, AtomProbe::MULTIRANGE_ERR_MOL_BAD_COLOUR, AtomProbe::MULTIRANGE_ERR_MOL_MISSING_COLOUR, AtomProbe::MULTIRANGE_ERR_MOL_MISSING_COMPONENTS, AtomProbe::MULTIRANGE_ERR_MOL_MISSING_ENTRY, AtomProbe::MULTIRANGE_ERR_MOL_MISSING_NAME, AtomProbe::MULTIRANGE_ERR_NO_CONTENT, AtomProbe::MULTIRANGE_ERR_NO_MOLECULE_ENTRIES, AtomProbe::MULTIRANGE_ERR_NO_MOLECULES, AtomProbe::MULTIRANGE_ERR_NO_NODEPTR, AtomProbe::MULTIRANGE_ERR_NO_PARSER, AtomProbe::MULTIRANGE_ERR_NO_RANGES, AtomProbe::MULTIRANGE_ERR_NO_TIMESTAMP, AtomProbe::MULTIRANGE_ERR_NO_XML_CONTEXT, AtomProbe::MULTIRANGE_ERR_NO_XML_DOC, AtomProbe::MULTIRANGE_ERR_RANGE_NOPARENT, AtomProbe::MULTIRANGE_ERR_SELF_INCONSISTENT, AtomProbe::MULTIRANGE_NOT_VALID_ROOTNODE, AtomProbe::parseColString(), AtomProbe::RGBf::red, AtomProbe::stream_cast(), AtomProbe::XMLFreeDoc(), AtomProbe::XMLHelpFwdToElem(), and AtomProbe::XMLHelpGetProp().

Referenced by copyDataFromRange().

Here is the call graph for this function:



6.31.3.23 range()

```
void AtomProbe::MultiRange::range (
    std::vector< IonHit > & ionHits ) const
```

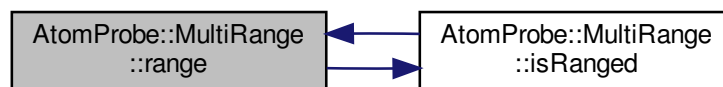
Clips out ions that are not inside the rangefile's ranges.

Definition at line 789 of file multiRange.cpp.

References isRanged().

Referenced by getIonID(), isRanged(), and setIonID().

Here is the call graph for this function:



6.31.3.24 setColour()

```
void AtomProbe::MultiRange::setColour (
    unsigned int,
    const RGBF & r )
```

Set the colour using the ion ID.

6.31.3.25 setIonID()

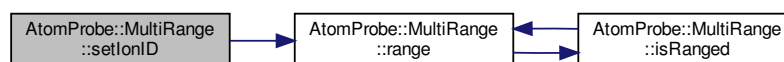
```
void AtomProbe::MultiRange::setIonID (
    unsigned int range,
    unsigned int newIonId )
```

Set the ion ID for a given range.

Definition at line 758 of file multiRange.cpp.

References ASSERT, and range().

Here is the call graph for this function:



6.31.3.26 setRangeGroups()

```
void AtomProbe::MultiRange::setRangeGroups (
    const std::vector< unsigned int > & groups )
```

Set the groupings for the ranges.

Definition at line 300 of file multiRange.cpp.

Referenced by copyDataFromRange().

6.31.3.27 splitOverlapping()

```
void AtomProbe::MultiRange::splitOverlapping (
    std::vector< MultiRange > & decomposedRanges ) const
```

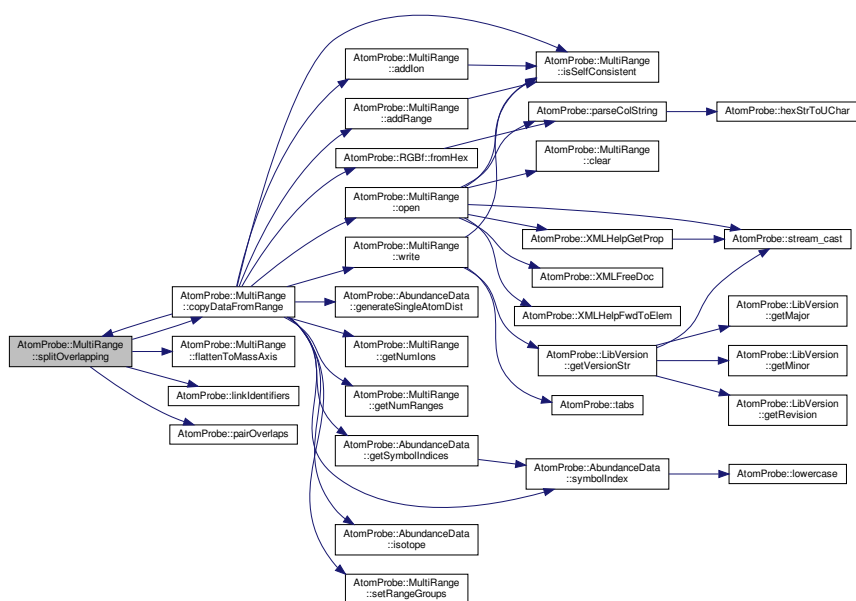
Separate out the non-interacting parts of the multi-ranges into their own multiRange entries The vector of maps converts the ioniDs of the child MultiRange to the parent

Definition at line 819 of file multiRange.cpp.

References ASSERT, copyDataFromRange(), flattenToMassAxis(), AtomProbe::linkIdentifiers(), and AtomProbe::pairOverlaps().

Referenced by copyDataFromRange().

Here is the call graph for this function:



6.31.3.28 write()

```
bool AtomProbe::MultiRange::write (
    const char * fileName,
    unsigned int format = MULTIRANGE_FORMAT_XML ) const
```

Save the structure to a file. format specifies output file format type.

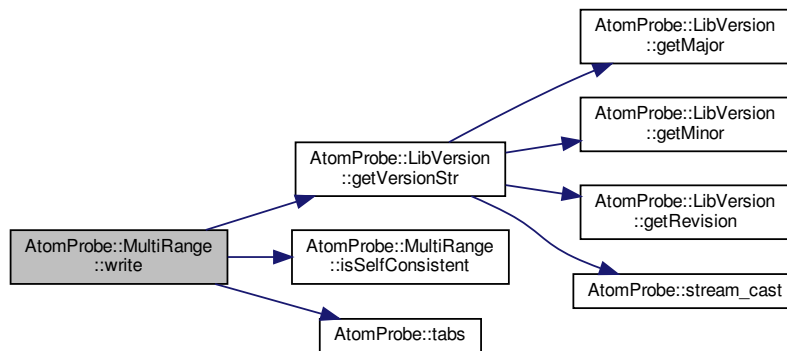
Returns true if successful, otherwise false

Definition at line 305 of file multiRange.cpp.

References ASSERT, AtomProbe::LibVersion::getVersionStr(), isSelfConsistent(), AtomProbe::MULTIRANGE_FORMAT_XML, and AtomProbe::tabs().

Referenced by copyDataFromRange().

Here is the call graph for this function:

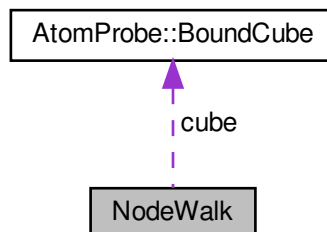


The documentation for this class was generated from the following files:

- [include/atomprobe/io/multiRange.h](#)
- [src/io/multiRange.cpp](#)

6.32 NodeWalk Class Reference

Collaboration diagram for NodeWalk:



Public Member Functions

- [NodeWalk](#) (unsigned int *idx*, const [BoundCube](#) &*bc*, unsigned int *dpth*)

Public Attributes

- size_t [index](#)
- [BoundCube](#) [cube](#)
- unsigned int [depth](#)

6.32.1 Detailed Description

Definition at line 55 of file `K3DTree-exact.cpp`.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 NodeWalk()

```
NodeWalk::NodeWalk (
    unsigned int idx,
    const BoundCube & bc,
    unsigned int dpth ) [inline]
```

Definition at line 61 of file `K3DTree-exact.cpp`.

6.32.3 Member Data Documentation

6.32.3.1 cube

[BoundCube](#) `NodeWalk::cube`

Definition at line 59 of file `K3DTree-exact.cpp`.

6.32.3.2 depth

unsigned int `NodeWalk::depth`

Definition at line 60 of file `K3DTree-exact.cpp`.

6.32.3.3 index

```
size_t NodeWalk::index
```

Definition at line 58 of file K3DTree-exact.cpp.

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

- [src/algorithm/K3DTree-exact.cpp](#)

6.33 AtomProbe::OVERLAP_PROBLEM_SETTINGS Struct Reference

```
#include <componentAnalysis.h>
```

Public Attributes

- float [massTolerance](#)
Tolerance in Da, above which we consider problems to be.
- float [intensityTolerance](#)
The minimal natural abundance for which we consider the ions.
- unsigned int [maxDefaultCharge](#)
Maximum charge state to consider.

6.33.1 Detailed Description

Definition at line 24 of file componentAnalysis.h.

6.33.2 Member Data Documentation

6.33.2.1 intensityTolerance

```
float AtomProbe::OVERLAP_PROBLEM_SETTINGS::intensityTolerance
```

The minimal natural abundance for which we consider the ions.

Definition at line 31 of file componentAnalysis.h.

Referenced by `AtomProbe::computeIonDistAdjacency()`.

6.33.2.2 massTolerance

```
float AtomProbe::OVERLAP_PROBLEM_SETTINGS::massTolerance
```

Tolerance in Da, above which we consider problems to be.

Definition at line 27 of file componentAnalysis.h.

Referenced by AtomProbe::computeIonDistAdjacency(), and AtomProbe::computeRangeAdjacency().

6.33.2.3 maxDefaultCharge

```
unsigned int AtomProbe::OVERLAP_PROBLEM_SETTINGS::maxDefaultCharge
```

Maximum charge state to consider.

Definition at line 34 of file componentAnalysis.h.

Referenced by AtomProbe::computeIonDistAdjacency(), and AtomProbe::computeRangeAdjacency().

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

- [include/atomprobe/algorithm/componentAnalysis.h](#)

6.34 AtomProbe::Point3D Class Reference

A 3D point data class storage.

```
#include <point3D.h>
```

Public Member Functions

- [Point3D](#) ()
Constructor with no initialisation.
- [Point3D](#) (const float *f)
Constructor initialising values from an array of length 4(sizeof(float))*
- [Point3D](#) (float x, float y, float z)
Constructor with initialising values, X, Y and Z.
- void [setValue](#) (unsigned int ui, float val)
Set value val of the ui-th dimension, ui in (0,1,2)
- void [setValue](#) (float fX, float fY, float fZ)
Set the XYZ value to the point.
- bool [parse](#) (const std::string &s)
Set from string representation.
- void [setValueArr](#) (const float *val)
Set value by pointer (X, Y, Z from array of len 3)
- float [getValue](#) (unsigned int ui) const

- Get value of ith dimension (0, 1, 2)*

 - const float * `getValueArr ()` const
- Obtain a pointer to internal array (3 floats)*

 - void `copyValueArr` (float *value) const
- Copy data from internal into target array of length 3 : Pointer MUST be allocated.*

 - void `add` (const `Point3D` &obj)
- Add a point to this, without generating a return value.*

 - void `extend` (float distance)
- Extend the vector by the specified distance.*

 - bool `hasNaN ()` const
- Returns true if any of the 3 data pts are NaN.*

 - bool `operator==` (const `Point3D` &pt) const
- Equality operator.*

 - const `Point3D` & `operator=` (const `Point3D` &pt)
- Assignment operator.*

 - const `Point3D` & `operator+=` (const `Point3D` &pt)
- += operator*

 - const `Point3D` & `operator-=` (const `Point3D` &pt)
- = operator*

 - `Point3D operator*=` (const float scale)
- *= operator, multiplies the whole `Point3D` by a scalar*

 - const `Point3D operator+` (const `Point3D` &pt) const
- Addition of `Point3D` objects (X1+X2, Y1+Y2, Z1+Z2)*

 - const `Point3D operator+` (float f) const
- Addition of a scalar to a `Point3D` object (X+f, Y+f, Z+f)*

 - const `Point3D operator*` (float scale) const
- Scalar multiplication (X*f, Y*f, Z*f)*

 - const `Point3D operator*` (const `Point3D` &pt) const
- Elemental multiplication of two `Point3D` objects (X1*X2, Y1*Y2, Z1*Z2)*

 - const `Point3D operator/` (float scale) const
- Scalar division of `Point3D` by scale (X/scale, Y/scale, Z/scale)*

 - const `Point3D operator/` (const `Point3D` &pt) const
- Elemental division of two `Point3D` objects (X1/X2, Y1/Y2, Z1/Z2)*

 - const `Point3D operator-` (const `Point3D` &pt) const
- Subtraction of two `Point3D` objects (X1-X2, Y1-Y2, Z1-Z2)*

 - const `Point3D operator-` () const
- Negation, Returns a `Point3D` object with the negative of the previous value.*

 - void `normalise ()`
- Make point unit magnitude, maintaining direction.*

 - `Point3D normal ()` const
- Return point unit magnitude, maintianing direction.*

 - float `sqrDist` (const `Point3D` &pt) const
- Returns the square of distance to another `Point3D`.*

 - float `dotProd` (const `Point3D` &pt) const
- Calculate the dot product of this and another point.*

 - `Point3D crossProd` (const `Point3D` &pt) const
- Calculate the cross product of this and another point.*

 - float `angle` (const `Point3D` &pt) const
- Calculate the angle between two position vectors in radians.*

 - float `sqrMag ()` const
- Returns magnitude², taking this as a position vector.*

- float [mag](#) () const
Magnitude of position vector.
- float [operator\[\]](#) (unsigned int ui) const
Array indexing operator, [Point3D\[1\]](#) returns the value of the ui-th dim.
- float & [operator\[\]](#) (unsigned int ui)
Array reference operator, [Point3D\[1\]](#).
- bool [insideBox](#) (const [Point3D](#) &farPt) const
Is this point inside a box bounded by origin and farPt?
- bool [insideBox](#) (const [Point3D](#) &lowPt, const [Point3D](#) &hiPt) const
Is this point inside a box bounded by lowPt and hiPt?
- void [negate](#) ()
Makes each value negative of old value (-X, -Y, -Z)
- void [reciprocal](#) ()
Makes each value its reciprocal (1/X, 1/Y, 1/Z)
- void [transform3x3](#) (const gsl_matrix *matrix)
Perform a 3x3 matrix transformation using a GSL matrix.
- bool [orthogonalise](#) (const [Point3D](#) &p)
Perform a cross-product based orthogonalisation with the specified vector.
- void [setISOSpherical](#) (float radius, float theta, float phi)
Assign the vector using spherical coordinates.
- void [getISOSpherical](#) (float &radius, float &theta, float &phi) const
Get the [Point3D](#) value as spherical coordinates (ISO 30-11)

Static Public Member Functions

- static void [getCentroid](#) (const std::vector< [Point3D](#) > &pts, [Point3D](#) &p)
find the centroid of a set of points

Friends

- std::ostream & [operator<<](#) (std::ostream &stream, const [Point3D](#) &)
Output streaming operator. Streams values in the text format (x,y,z)

6.34.1 Detailed Description

A 3D point data class storage.

Definition at line 43 of file point3D.h.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 Point3D() [1/3]

```
AtomProbe::Point3D::Point3D ( ) [inline]
```

Constructor with no initialisation.

Definition at line 50 of file point3D.h.

Referenced by getCentroid().

6.34.2.2 Point3D() [2/3]

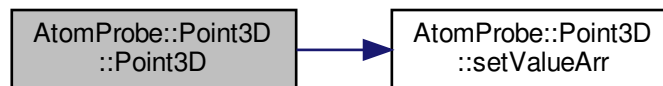
```
AtomProbe::Point3D::Point3D (
    const float * f ) [inline]
```

Constructor initialising values from an array of length 4*(sizeof(float))

Definition at line 53 of file point3D.h.

References setValueArr().

Here is the call graph for this function:



6.34.2.3 Point3D() [3/3]

```
AtomProbe::Point3D::Point3D (
    float x,
    float y,
    float z ) [inline]
```

Constructor with initialising values, X, Y and Z.

Definition at line 56 of file point3D.h.

6.34.3 Member Function Documentation

6.34.3.1 add()

```
void AtomProbe::Point3D::add (
    const Point3D & obj )
```

Add a point to this, without generating a return value.

Add a [Point3D](#) set of values to this [Point3D](#).

This is different to +=, because it generates no return value.

Definition at line 333 of file point3D.cpp.

Referenced by `getValueArr()`.

6.34.3.2 angle()

```
float AtomProbe::Point3D::angle (
    const Point3D & pt ) const
```

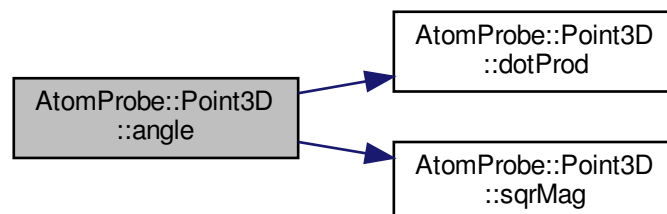
Calculate the angle between two position vectors in radians.

Definition at line 396 of file point3D.cpp.

References `dotProd()`, and `sqrMag()`.

Referenced by `AtomProbe::Mesh::divideMeshSurface()`, and `hasNaN()`.

Here is the call graph for this function:



6.34.3.3 copyValueArr()

```
void AtomProbe::Point3D::copyValueArr (
    float * value ) const
```

Copy data from internal into target array of length 3 : Pointer MUST be allocated.

As a quick example, you can copy into a buffer you own `Point3D p(1,2,3); float x[3]; p.copyValueArr(x[0]);`

Definition at line 147 of file point3D.cpp.

References ASSERT.

Referenced by `getValueArr()`.

6.34.3.4 crossProd()

```
Point3D AtomProbe::Point3D::crossProd (
    const Point3D & pt ) const
```

Calculate the cross product of this and another point.

Definition at line 293 of file point3D.cpp.

Referenced by `AtomProbe::TriangleWithVertexNorm::computeACWNormal()`, `AtomProbe::TriangleWithVertexNorm::computeArea()`, `AtomProbe::computeRotationMatrix()`, `AtomProbe::distanceToSegment()`, `AtomProbe::Mesh::getTriNormal()`, `AtomProbe::Mesh::getVolume()`, `hasNaN()`, `AtomProbe::intersect_RayTriangle()`, `main()`, `orthogonalise()`, and `AtomProbe::TriangleWithVertexNorm::safeComputeACWNormal()`.

6.34.3.5 dotProd()

```
float AtomProbe::Point3D::dotProd (
    const Point3D & pt ) const
```

Calculate the dot product of this and another point.

$(X1*X2 + Y1*Y2 \dots)$

Definition at line 287 of file point3D.cpp.

Referenced by `angle()`, `AtomProbe::generate1DAxialDistHistSweep()`, `AtomProbe::Mesh::getVolume()`, `hasNaN()`, `AtomProbe::intersect_RayTriangle()`, `main()`, and `AtomProbe::signedDistanceToFacet()`.

6.34.3.6 extend()

```
void AtomProbe::Point3D::extend (
    float distance )
```

Extend the vector by the specified distance.

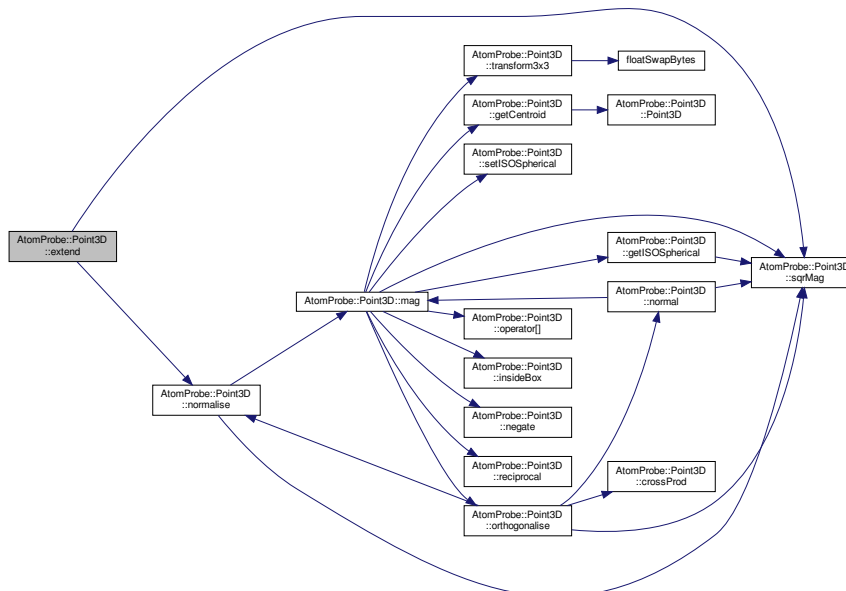
Assumes [Point3D](#) is a vector from the origin, and adds on a vector parallel to [Point3D](#) of length *distance*.

Definition at line 306 of file point3D.cpp.

References `ASSERT`, `normalise()`, and `sqrMag()`.

Referenced by `getValueArr()`.

Here is the call graph for this function:



6.34.3.7 getCentroid()

```
void AtomProbe::Point3D::getCentroid (
    const std::vector< Point3D > & pts,
    Point3D & p ) [static]
```

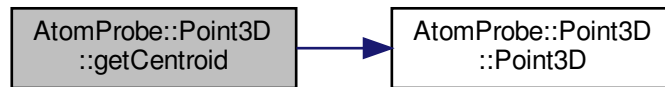
find the centroid of a set of points

Definition at line 401 of file point3D.cpp.

References `Point3D()`.

Referenced by `mag()`.

Here is the call graph for this function:



6.34.3.8 getISOSpherical()

```
void AtomProbe::Point3D::getISOSpherical (
    float & radius,
    float & theta,
    float & phi ) const
```

Get the [Point3D](#) value as spherical coordinates (ISO 30-11)

Internal cartesian representation converted to ISO 30-11 and returned in the pointers provided as input.

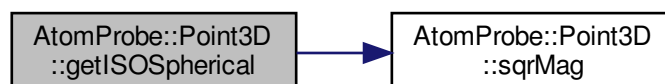
theta - inclination (angle from +z). phi - azimuth (angle from +x), r - radius

Definition at line 441 of file point3D.cpp.

References [sqrMag\(\)](#).

Referenced by [mag\(\)](#).

Here is the call graph for this function:



6.34.3.9 getValue()

```
float AtomProbe::Point3D::getValue (
    unsigned int ui ) const [inline]
```

Get value of ith dimension (0, 1, 2)

Definition at line 102 of file point3D.h.

Referenced by AtomProbe::SimpleCubicGen::generateLattice(), AtomProbe::FaceCentredCubicGen::generateLattice(), AtomProbe::BodyCentredCubicGen::generateLattice(), AtomProbe::K3DNodeApprox::getAxisVal(), AtomProbe::IonHit::getBoundCube(), AtomProbe::IonHit::getIonDataLimits(), and AtomProbe::K3DNodeApprox::getLocVal().

6.34.3.10 getValueArr()

```
const float* AtomProbe::Point3D::getValueArr ( ) const [inline]
```

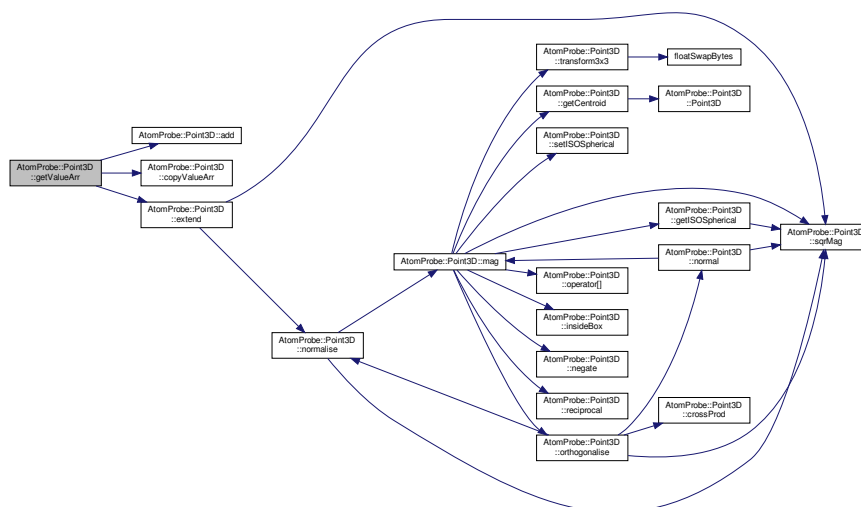
Obtain a pointer to internal array (3 floats)

Definition at line 105 of file point3D.h.

References add(), copyValueArr(), and extend().

Referenced by AtomProbe::BoundCube::intersects().

Here is the call graph for this function:



6.34.3.12 insideBox() [1/2]

```
bool AtomProbe::Point3D::insideBox (
    const Point3D & farPoint ) const
```

Is this point inside a box bounded by origin and *farPoint*?

Returns true if this point is located inside (0,0,0) -> *farPoint*

Assuming box shape (non zero edges return false)

farPoint must be positive in all dim

Definition at line 316 of file point3D.cpp.

Referenced by mag().

6.34.3.13 insideBox() [2/2]

```
bool AtomProbe::Point3D::insideBox (
    const Point3D & lowPt,
    const Point3D & hiPt ) const
```

Is this point inside a box bounded by *lowPt* and *hiPt*?

Returns true if this point is located inside *lowPt* -> *hiPt*

Assuming box shape.

Definition at line 324 of file point3D.cpp.

6.34.3.14 mag()

```
float AtomProbe::Point3D::mag ( ) const [inline]
```

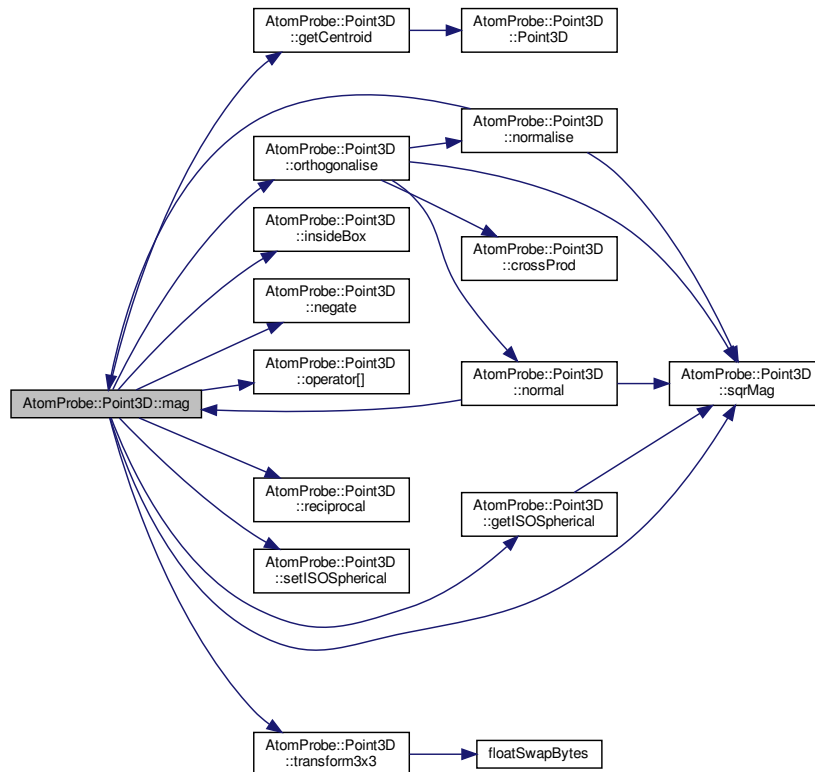
Magnitude of position vector.

Definition at line 201 of file point3D.h.

References getCentroid(), getISOSpherical(), insideBox(), negate(), operator[](), orthogonalise(), reciprocal(), set↔ISOSpherical(), sqrMag(), and transform3x3().

Referenced by normal(), normalise(), AtomProbe::signedDistanceToFacet(), and AtomProbe::trilsDegenerate().

Here is the call graph for this function:



6.34.3.15 negate()

```
void AtomProbe::Point3D::negate ( )
```

Makes each value negative of old value (-X, -Y, -Z)

Definition at line 366 of file point3D.cpp.

Referenced by mag().

6.34.3.16 normal()

```
Point3D AtomProbe::Point3D::normal ( ) const
```

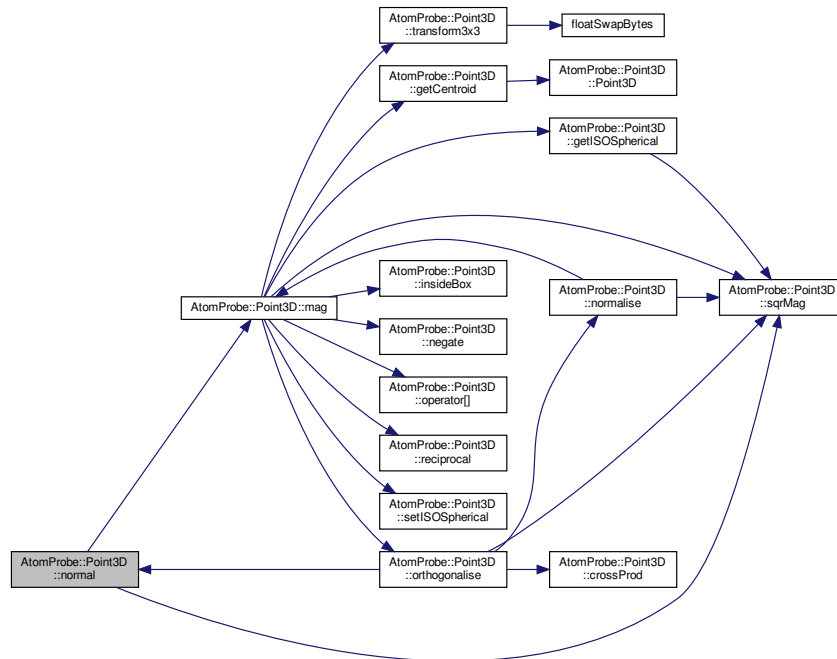
Return point unit magnitude, maintianing direction.

Definition at line 355 of file point3D.cpp.

References `mag()`, and `sqrMag()`.

Referenced by `hasNaN()`, and `orthogonalise()`.

Here is the call graph for this function:



6.34.3.17 normalise()

```
void AtomProbe::Point3D::normalise ( )
```

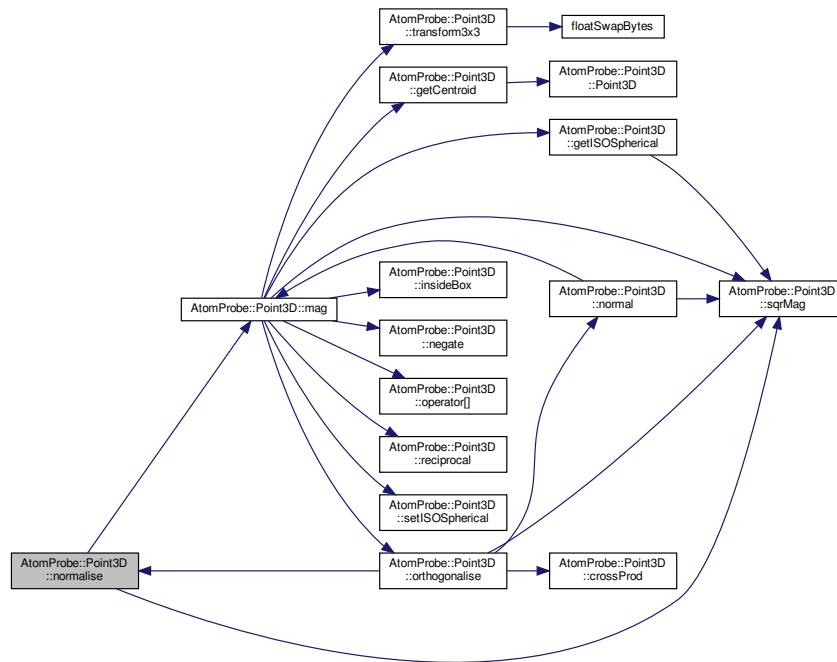
Make point unit magnitude, maintaining direction.

Definition at line 345 of file `point3D.cpp`.

References `mag()`, and `sqrMag()`.

Referenced by `AtomProbe::TriangleWithVertexNorm::computeACWNormal()`, `AtomProbe::computeRotationMatrix()`, `extend()`, `AtomProbe::Mesh::getTriNormal()`, `hasNaN()`, `AtomProbe::intersect_RayTriangle()`, `main()`, `orthogonalise()`, and `AtomProbe::TriangleWithVertexNorm::safeComputeACWNormal()`.

Here is the call graph for this function:



6.34.3.18 operator*() [1/2]

```
const Point3D AtomProbe::Point3D::operator* (
    float scale ) const
```

Scalar multiplication (X*f, Y*f, Z*f)

Definition at line 234 of file point3D.cpp.

Referenced by hasNaN().

6.34.3.19 operator*() [2/2]

```
const Point3D AtomProbe::Point3D::operator* (
    const Point3D & pt ) const
```

Elemental multiplication of two Point3D objects (X1*X2, Y1*Y2, Z1*Z2)

Definition at line 245 of file point3D.cpp.

6.34.3.20 operator*=()

```
Point3D AtomProbe::Point3D::operator*= (
    const float scale )
```

*= operator, multiplies the whole [Point3D](#) by a scalar

Definition at line 225 of file point3D.cpp.

Referenced by [hasNaN\(\)](#).

6.34.3.21 operator+() [1/2]

```
const Point3D AtomProbe::Point3D::operator+ (
    const Point3D & pt ) const
```

Addition of [Point3D](#) objects (X1+X2, Y1+Y2, Z1+Z2)

Definition at line 186 of file point3D.cpp.

Referenced by [hasNaN\(\)](#).

6.34.3.22 operator+() [2/2]

```
const Point3D AtomProbe::Point3D::operator+ (
    float f ) const
```

Addition of a scalar to a [Point3D](#) object (X+f, Y+f, Z+f)

Definition at line 196 of file point3D.cpp.

6.34.3.23 operator+=()

```
const Point3D & AtomProbe::Point3D::operator+= (
    const Point3D & pt )
```

+= operator

Definition at line 178 of file point3D.cpp.

Referenced by [hasNaN\(\)](#).

6.34.3.24 operator-() [1/2]

```
const Point3D AtomProbe::Point3D::operator- (
    const Point3D & pt ) const
```

Subtraction of two [Point3D](#) objects (X1-X2, Y1-Y2, Z1-Z2)

Definition at line 205 of file point3D.cpp.

6.34.3.25 operator-() [2/2]

```
const Point3D AtomProbe::Point3D::operator- ( ) const
```

Negation, Returns a [Point3D](#) object with the negative of the previous value.

Definition at line 215 of file point3D.cpp.

Referenced by [hasNaN\(\)](#).

6.34.3.26 operator-=()

```
const Point3D & AtomProbe::Point3D::operator-= (
    const Point3D & pt )
```

`-=` operator

Definition at line 157 of file point3D.cpp.

Referenced by [hasNaN\(\)](#).

6.34.3.27 operator/() [1/2]

```
const Point3D AtomProbe::Point3D::operator/ (
    float scale ) const
```

Scalar division of [Point3D](#) by *scale* (X/scale, Y/scale, Z/scale)

Definition at line 256 of file point3D.cpp.

Referenced by [hasNaN\(\)](#).

6.34.3.28 operator/() [2/2]

```
const Point3D AtomProbe::Point3D::operator/ (
    const Point3D & pt ) const
```

Elemental division of two [Point3D](#) objects (X1/X2, Y1/Y2, Z1/Z2)

Definition at line 268 of file point3D.cpp.

6.34.3.29 operator=()

```
const Point3D & AtomProbe::Point3D::operator= (
    const Point3D & pt )
```

Assignment operator.

Definition at line 170 of file point3D.cpp.

Referenced by [hasNaN\(\)](#).

6.34.3.30 operator==()

```
bool AtomProbe::Point3D::operator== (
    const Point3D & pt ) const
```

Equality operator.

Definition at line 165 of file point3D.cpp.

Referenced by [hasNaN\(\)](#).

6.34.3.31 operator[]() [1/2]

```
float AtomProbe::Point3D::operator[] (
    unsigned int ui ) const
```

Array indexing operator, [Point3D](#)[1] returns the value of the ui-th dim.

Definition at line 135 of file point3D.cpp.

References [ASSERT](#).

Referenced by [mag\(\)](#).

6.34.3.32 operator[]() [2/2]

```
float & AtomProbe::Point3D::operator[] (
    unsigned int ui )
```

Array reference operator, [Point3D\[1\]](#).

returns reference to data in ui-th dim

Definition at line 141 of file point3D.cpp.

References ASSERT.

6.34.3.33 orthogonalise()

```
bool AtomProbe::Point3D::orthogonalise (
    const Point3D & p )
```

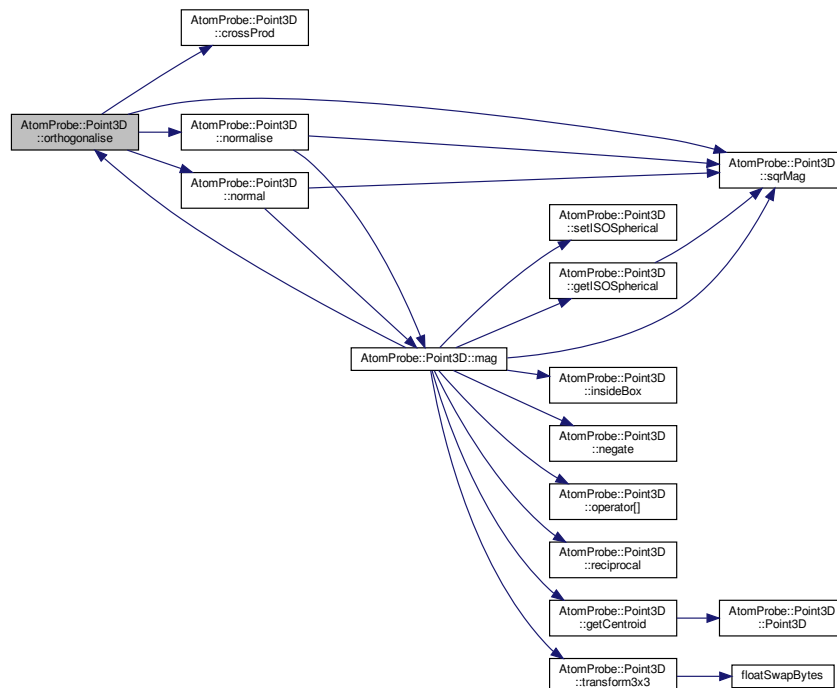
Perform a cross-product based orthogonalisation with the specified vector.

Definition at line 380 of file point3D.cpp.

References [crossProd\(\)](#), [normal\(\)](#), [normalise\(\)](#), and [sqrMag\(\)](#).

Referenced by [mag\(\)](#).

Here is the call graph for this function:



6.34.3.34 parse()

```
bool AtomProbe::Point3D::parse (
    const std::string & s )
```

Set from string representation.

From a string representation of a 3D point, parse and store the data in a [Point3D](#).

Must contain 3 entries.

Values can be separated by whitespace or any of `;,|_`

Polar notation is indicated by `< ... >`

Polar notation assumes `<radius,theta,phi>` in degrees (they will be converted to radians internally) and then converted into cartesian coordinates (X,Y,Z)

```
Point3D mypoint;
mypoint.parse("<1, 45, 90>");
cout << mypoint << endl;
```

Returns:

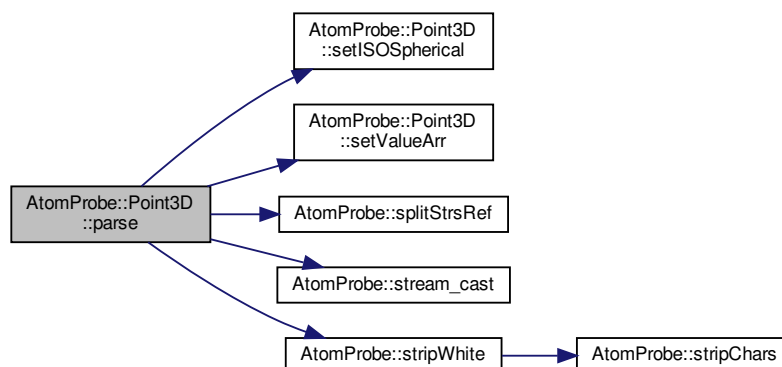
```
(-3.09086e-08, 0.707107, 0.707107)
```

Definition at line 47 of file point3D.cpp.

References `M_PI`, `setISOSpherical()`, `setValueArr()`, `AtomProbe::splitStrsRef()`, `AtomProbe::stream_cast()`, and `AtomProbe::stripWhite()`.

Referenced by `main()`, and `setValue()`.

Here is the call graph for this function:



6.34.3.35 reciprocal()

```
void AtomProbe::Point3D::reciprocal ( )
```

Makes each value its reciprocal (1/X, 1/Y, 1/Z)

Definition at line 373 of file point3D.cpp.

Referenced by mag().

6.34.3.36 setISOSpherical()

```
void AtomProbe::Point3D::setISOSpherical (
    float radius,
    float theta,
    float phi )
```

Assign the vector using spherical coordinates.

theta - inclination (angle from +z). phi - azimuth (angle from +x), r - radius

The value of the point is stored in cartesian coords.

Definition at line 433 of file point3D.cpp.

Referenced by AtomProbe::generate1DAxialDistHistSweep(), mag(), parse(), and AtomProbe::Gnomonic↔Projection::toPlanar().

6.34.3.37 setValue() [1/2]

```
void AtomProbe::Point3D::setValue (
    unsigned int ui,
    float val ) [inline]
```

Set value *val* of the *ui*-th dimension, *ui* in (0,1,2)

Definition at line 61 of file point3D.h.

Referenced by AtomProbe::BoundCube::getBounds(), AtomProbe::BoundCube::intersects(), AtomProbe::Bound↔Cube::max(), and AtomProbe::BoundCube::min().

6.34.3.38 setValue() [2/2]

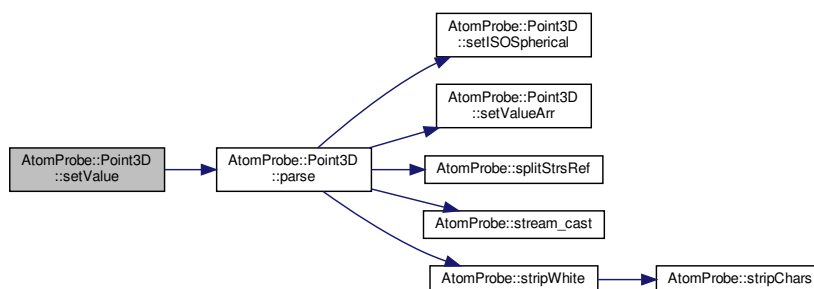
```
void AtomProbe::Point3D::setValue (
    float fX,
    float fY,
    float fZ ) [inline]
```

Set the XYZ value for the point.

Definition at line 64 of file point3D.h.

References parse().

Here is the call graph for this function:



6.34.3.39 setValueArr()

```
void AtomProbe::Point3D::setValueArr (
    const float * val ) [inline]
```

Set value by pointer (X, Y, Z from array of len 3)

Definition at line 94 of file point3D.h.

Referenced by parse(), Point3D(), and AtomProbe::IonHit::setHit().

6.34.3.40 sqrDist()

```
float AtomProbe::Point3D::sqrDist (
    const Point3D & pt ) const
```

Returns the square of distance to another [Point3D](#).

$(X1-X2)^2 + (Y1-Y2)^2 + \dots$

Definition at line 280 of file point3D.cpp.

Referenced by AtomProbe::K3DTreeExact::clearAllTags(), AtomProbe::computeConvexHull(), AtomProbe::computeRotationMatrix(), AtomProbe::distanceToSegment(), AtomProbe::K3DTreeApprox::findKNearest(), AtomProbe::K3DTreeExact::findUntaggedInRadius(), AtomProbe::generate1DAXialDistHist(), AtomProbe::generate1DAXialDistHistSweep(), AtomProbe::BoundCube::getMaxDistanceToBox(), hasNaN(), AtomProbe::incrementDataDistanceWeight(), AtomProbe::BoundCube::intersects(), main(), AtomProbe::signedDistanceToFacet(), AtomProbe::K3DNodeApprox::sqrDist(), and testInexactKDTree().

6.34.3.41 `sqrMag()`

```
float AtomProbe::Point3D::sqrMag ( ) const
```

Returns magnitude^2 , taking this as a position vector.

Definition at line 340 of file `point3D.cpp`.

Referenced by `angle()`, `AtomProbe::TriangleWithVertexNorm::computeArea()`, `AtomProbe::computeRotationMatrix()`, `AtomProbe::distanceToSegment()`, `extend()`, `AtomProbe::generate1DAXialDistHist()`, `getISOSpherical()`, `hasNaN()`, `AtomProbe::intersect_RayTriangle()`, `mag()`, `main()`, `normal()`, `normalise()`, `orthogonalise()`, and `AtomProbe::TriangleWithVertexNorm::safeComputeACWNNormal()`.

6.34.3.42 `transform3x3()`

```
void AtomProbe::Point3D::transform3x3 (
    const gsl_matrix * matrix )
```

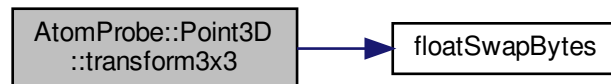
Perform a 3x3 matrix transformation using a GSL matrix.

Definition at line 453 of file `point3D.cpp`.

References `floatSwapBytes()`.

Referenced by `AtomProbe::computeRotationMatrix()`, and `mag()`.

Here is the call graph for this function:



6.34.4 Friends And Related Function Documentation

6.34.4.1 `operator<<`

```
std::ostream& operator<< (
    std::ostream & stream,
    const Point3D & pt ) [friend]
```

Output streaming operator. Streams values in the text format (x,y,z)

Definition at line 425 of file `point3D.cpp`.

Referenced by `hasNaN()`.

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

- `include/atomprobe/primitives/point3D.h`
- `src/primitives/point3D.cpp`

6.35 AtomProbe::Point3f Struct Reference

Data storage structure for points.

```
#include <quat.h>
```

Public Attributes

- float [fx](#)
- float [fy](#)
- float [fz](#)

6.35.1 Detailed Description

Data storage structure for points.

Definition at line 45 of file quat.h.

6.35.2 Member Data Documentation

6.35.2.1 fx

```
float AtomProbe::Point3f::fx
```

Definition at line 47 of file quat.h.

Referenced by [AtomProbe::quat_get_rot_quat\(\)](#), [AtomProbe::quat_pointmult\(\)](#), [AtomProbe::quat_rot\(\)](#), [AtomProbe::quat_rot_apply_quat\(\)](#), [AtomProbe::quat_rot_array\(\)](#), and [AtomProbe::Mesh::rotate\(\)](#).

6.35.2.2 fy

```
float AtomProbe::Point3f::fy
```

Definition at line 48 of file quat.h.

Referenced by [AtomProbe::quat_get_rot_quat\(\)](#), [AtomProbe::quat_pointmult\(\)](#), [AtomProbe::quat_rot\(\)](#), [AtomProbe::quat_rot_apply_quat\(\)](#), [AtomProbe::quat_rot_array\(\)](#), and [AtomProbe::Mesh::rotate\(\)](#).

6.35.2.3 fz

```
float AtomProbe::Point3f::fz
```

Definition at line 49 of file quat.h.

Referenced by `AtomProbe::quat_get_rot_quat()`, `AtomProbe::quat_pointmult()`, `AtomProbe::quat_rot()`, `AtomProbe::quat_rot_apply_quat()`, `AtomProbe::quat_rot_array()`, and `AtomProbe::Mesh::rotate()`.

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

- [include/atomprobe/helper/maths/quat.h](#)

6.36 AtomProbe::ProgressBar Class Reference

```
#include <progress.h>
```

Public Member Functions

- [ProgressBar](#) ()
- [~ProgressBar](#) ()
- void [setLength](#) (unsigned int l)
Set the number of markers in the progress bar.
- void [init](#) ()
Draw the initial progress bar.
- void [reset](#) ()
reset the progress bar internals, in case we want to re-use it
- void [update](#) (unsigned int newProgress)
Draw the progress bar as needed, using the given progress value [0,100].
- void [finish](#) ()
Finalise the progress bar. It is not necessary for the progress to be set to 100%, this is done for you.
- void [abort](#) ()
Abort drawing the progress.

6.36.1 Detailed Description

Definition at line 25 of file progress.h.

6.36.2 Constructor & Destructor Documentation

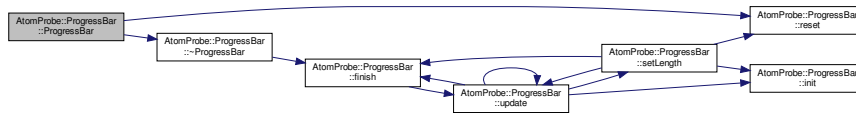
6.36.2.1 ProgressBar()

```
AtomProbe::ProgressBar::ProgressBar ( ) [inline]
```

Definition at line 37 of file progress.h.

References `reset()`, and `~ProgressBar()`.

Here is the call graph for this function:



6.36.2.2 ~ProgressBar()

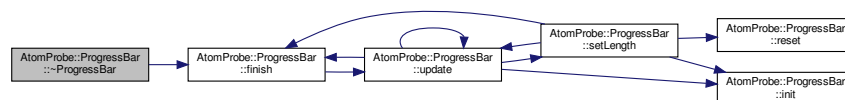
```
AtomProbe::ProgressBar::~ProgressBar ( )
```

Definition at line 29 of file progress.cpp.

References `finish()`.

Referenced by `ProgressBar()`.

Here is the call graph for this function:



6.36.3 Member Function Documentation

6.36.3.1 abort()

```
void AtomProbe::ProgressBar::abort ( ) [inline]
```

Abort drawing the progress.

Definition at line 55 of file progress.h.

6.36.3.2 finish()

```
void AtomProbe::ProgressBar::finish ( )
```

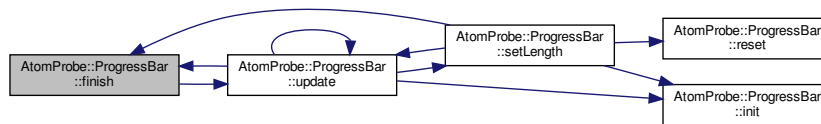
Finalise the progress bar. It is not necessary for the progress to be set to 100%, this is done for you.

Definition at line 52 of file progress.cpp.

References update().

Referenced by main(), setLength(), update(), and ~ProgressBar().

Here is the call graph for this function:



6.36.3.3 init()

```
void AtomProbe::ProgressBar::init ( )
```

Draw the initial progress bar.

Definition at line 35 of file progress.cpp.

Referenced by main(), setLength(), and update().

6.36.3.4 reset()

```
void AtomProbe::ProgressBar::reset ( )
```

reset the progress bar internals, in case we want to re-use it

Definition at line 43 of file progress.cpp.

Referenced by ProgressBar(), and setLength().

6.36.3.5 setLength()

```
void AtomProbe::ProgressBar::setLength (
    unsigned int l ) [inline]
```

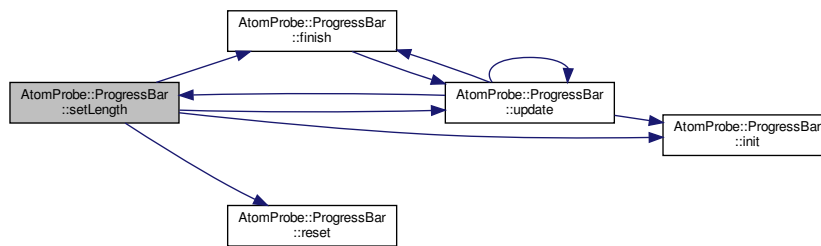
Set the number of markers in the progress bar.

Definition at line 40 of file progress.h.

References `finish()`, `init()`, `reset()`, and `update()`.

Referenced by `main()`, and `update()`.

Here is the call graph for this function:



6.36.3.6 update()

```
void AtomProbe::ProgressBar::update (
    unsigned int newProgress )
```

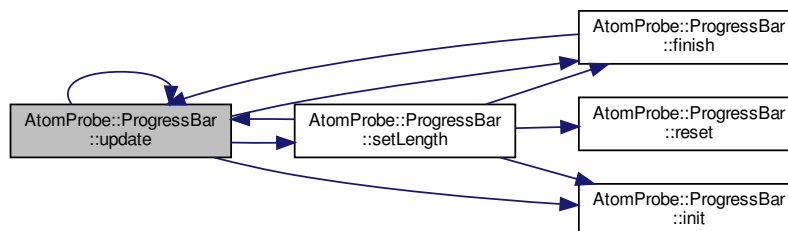
Draw the progress bar as needed, using the given progress value [0,100].

Definition at line 58 of file progress.cpp.

References `finish()`, `init()`, `progress`, `setLength()`, and `update()`.

Referenced by `callback()`, `finish()`, `AtomProbe::generate1DAXialDistHistSweep()`, `main()`, `setLength()`, and `update()`.

Here is the call graph for this function:



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

- [include/atomprobe/helper/progress.h](#)
- [src/helper/progress.cpp](#)

6.37 AtomProbe::Quaternion Struct Reference

Data storage structure for quaternions.

```
#include <quat.h>
```

Public Attributes

- float [a](#)
- float [b](#)
- float [c](#)
- float [d](#)

6.37.1 Detailed Description

Data storage structure for quaternions.

Definition at line 35 of file quat.h.

6.37.2 Member Data Documentation

6.37.2.1 a

```
float AtomProbe::Quaternion::a
```

Definition at line 37 of file quat.h.

Referenced by [AtomProbe::quat_get_rot_quat\(\)](#), [AtomProbe::quat_mult_no_second_a\(\)](#), [AtomProbe::quat_pointmult\(\)](#), [AtomProbe::quat_rot\(\)](#), and [AtomProbe::quat_rot_array\(\)](#).

6.37.2.2 b

```
float AtomProbe::Quaternion::b
```

Definition at line 38 of file quat.h.

Referenced by [AtomProbe::quat_get_rot_quat\(\)](#), [AtomProbe::quat_invert\(\)](#), [AtomProbe::quat_mult_no_second_a\(\)](#), [AtomProbe::quat_pointmult\(\)](#), [AtomProbe::quat_rot\(\)](#), [AtomProbe::quat_rot_apply_quat\(\)](#), and [AtomProbe::quat_rot_array\(\)](#).

6.37.2.3 c

```
float AtomProbe::Quaternion::c
```

Definition at line 39 of file quat.h.

Referenced by AtomProbe::quat_get_rot_quat(), AtomProbe::quat_invert(), AtomProbe::quat_mult_no_second_a(), AtomProbe::quat_pointmult(), AtomProbe::quat_rot(), AtomProbe::quat_rot_apply_quat(), and AtomProbe::quat_rot_array().

6.37.2.4 d

```
float AtomProbe::Quaternion::d
```

Definition at line 40 of file quat.h.

Referenced by AtomProbe::quat_get_rot_quat(), AtomProbe::quat_invert(), AtomProbe::quat_mult_no_second_a(), AtomProbe::quat_pointmult(), AtomProbe::quat_rot(), AtomProbe::quat_rot_apply_quat(), and AtomProbe::quat_rot_array().

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

- [include/atomprobe/helper/maths/quat.h](#)

6.38 AtomProbe::RandNumGen Class Reference

```
#include <atomprobe.h>
```

Public Member Functions

- [RandNumGen \(\)](#)
- [~RandNumGen \(\)](#)
- [gsl_rng * getRng \(\) const](#)
Obtain a GSL random number generator.

6.38.1 Detailed Description

Library-wide singleton random number source. Uses GSL as backend Defaults to marsenne twister

Definition at line 86 of file atomprobe.h.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 RandNumGen()

```
AtomProbe::RandNumGen::RandNumGen ( )
```

Definition at line 52 of file atomprobe.cpp.

6.38.2.2 ~RandNumGen()

```
AtomProbe::RandNumGen::~~RandNumGen ( )
```

Definition at line 63 of file atomprobe.cpp.

6.38.3 Member Function Documentation

6.38.3.1 getRng()

```
gsl_rng* AtomProbe::RandNumGen::getRng ( ) const [inline]
```

Obtain a GSL random number generator.

Definition at line 95 of file atomprobe.h.

References `AtomProbe::libVersion`, and `AtomProbe::randGen`.

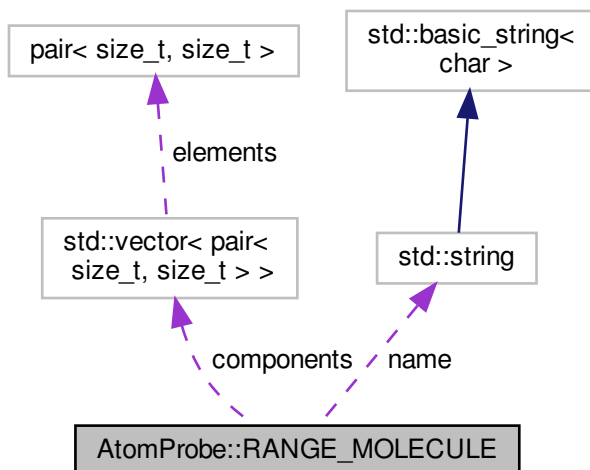
Referenced by `AtomProbe::leastSquaresDeconvolve()`, `AtomProbe::loadATOFile()`, `AtomProbe::loadPosFile()`, `main()`, `AtomProbe::poissonConfidenceObservation()`, and `AtomProbe::sampleIons()`.

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

- [include/atomprobe/atomprobe.h](#)
- [src/atomprobe.cpp](#)

6.39 AtomProbe::RANGE_MOLECULE Struct Reference

Collaboration diagram for AtomProbe::RANGE_MOLECULE:



Public Attributes

- bool `isOK`
- string `name`
- vector< pair< size_t, size_t > > `components`

6.39.1 Detailed Description

Definition at line 34 of file `rangeCheck.cpp`.

6.39.2 Member Data Documentation

6.39.2.1 components

```
vector<pair<size_t,size_t> > AtomProbe::RANGE_MOLECULE::components
```

Definition at line 41 of file `rangeCheck.cpp`.

Referenced by `AtomProbe::checkMassRangingCorrectness()`, and `AtomProbe::getRangeMolecule()`.

6.39.2.2 isOK

```
bool AtomProbe::RANGE_MOLECULE::isOK
```

Definition at line 36 of file rangeCheck.cpp.

Referenced by AtomProbe::checkMassRangingCorrectness(), and AtomProbe::getRangeMolecule().

6.39.2.3 name

```
string AtomProbe::RANGE_MOLECULE::name
```

Definition at line 38 of file rangeCheck.cpp.

Referenced by AtomProbe::getRangeMolecule().

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

- [src/algorithm/rangeCheck.cpp](#)

6.40 AtomProbe::RangeFile Class Reference

Data storage and retrieval class for various range files.

```
#include <ranges.h>
```

Public Member Functions

- [RangeFile](#) ()
- const [RangeFile](#) & [operator=](#) (const [RangeFile](#) &other)
- bool [open](#) (const char *rangeFile)
Open a specified range file - returns true on success.
- unsigned int [openFormat](#) (const char *rangeFile, unsigned int format)
Open a specified range file using a given file format. Returns nonzero on failure.
- bool [isSelfConsistent](#) () const
Performs checks for self consistency.
- bool [isSelfConsistent](#) (std::vector< std::string > &consistencyMessages) const
Performs checks for self consistency, whilst providing information on the inconsistent components.
- bool [makeSelfConsistent](#) ()
Modify range file to ensure self-consistency, by arbitrary heuristics.
- std::string [getErrString](#) () const
Retrieve the translated error associated with the current range file state.
- unsigned int [getErrState](#) () const
Obtain the error state last set for this rangefile.
- unsigned int [getNumRanges](#) () const
Get the number of unique ranges.
- unsigned int [getNumRanges](#) (unsigned int ionID) const

- Get the number of ranges for a given ion ID.*

 - unsigned int `getNumIons` () const
- Get the number of unique ions.*

 - std::pair< float, float > `getRange` (unsigned int) const

Retrieve the start and end of a given range as a pair(start,end)
- std::pair< float, float > & `getRangeByRef` (unsigned int)

Retrieve the start and end of a given range as a pair(start,end)
- bool `haveRangeVolumes` () const

Return true if we have range volume data, false otherwise.
- float `getRangeVolume` (unsigned int rangeld) const

Obtain an ions volume, if we have it. Zero may indicate no ion volume specified for that range.
- void `setRangeVolume` (unsigned int rangeld, float newVolume)

Set the ion volume for a given range. If ion volumes are not tracked, these will be created.
- RGBf `getColour` (unsigned int) const

Retrieve a given colour from the ion ID.
- void `setColour` (unsigned int, const RGBf &r)

Set the colour using the ion ID.
- unsigned int `getIonID` (float mass) const

Retrieve the colour from a given ion ID.
- unsigned int `getIonID` (const IonHit &hit) const

Get the ion's ID from a specified mass, using an IonHit.
- unsigned int `getIonID` (unsigned int range) const

Get the ion ID from a given range ID.
- unsigned int `getIonID` (const char *name, bool useShortName=true) const

Get the ion ID from its short or long name, returns -1 if name does not exist. Case must match.
- unsigned int `getIonID` (const std::string &name) const
- void `setIonID` (unsigned int range, unsigned int newIonId)

Set the ion ID for a given range.
- bool `isRanged` (float mass) const

Returns true if a specified mass is ranged.
- bool `isRanged` (const IonHit &) const

Returns true if an ion is ranged.
- void `range` (std::vector< IonHit > &ionHits) const

Clips out ions that are not inside the rangefile's ranges.
- void `rangeInvertable` (std::vector< IonHit > &ionHits, bool invert)

Clips out ions that are inside the rangefile's ranges, if invert is false. If true, then reverse the set.
- bool `range` (std::vector< IonHit > &ionHits, const std::string &shortIonName)

Clips out ions that don't match the specified ion name.
- bool `rangeByID` (std::vector< IonHit > &ionHits, unsigned int range)

Clips out ions that don't lie in the specified range number.
- void `rangeByRangeID` (std::vector< IonHit > &ionHits, unsigned int rangeID)

Range the given ions, allowing only ions in the specified range. Input data will be modified to return result.
- void `extractIons` (const std::vector< IonHit > &ionHits, const std::vector< unsigned int > &ionIDs, std::vector< IonHit > &hits) const

Extract ion hit events in the specified ion ID values, from the input ion ionhit sequence.
- std::string `getName` (unsigned int ionID, bool shortName=true) const

Get the short name or long name of a specified ionID.
- std::string `getName` (const IonHit &ion, bool shortName=true) const

Obtain the short name for a given IonHit. If not ranged, an empty string will be returned.
- void `setIonShortName` (unsigned int ionID, const std::string &newName)

set the short name for a given ion

- void [setIonLongName](#) (unsigned int ionID, const std::string &newName)
Set the long name for a given ion.
- bool [isRanged](#) (std::string shortName, bool caseSensitive=true)
Check to see if an atom is ranged.
- unsigned int [write](#) (std::ostream &o, size_t format=RANGE_FORMAT_ORNL) const
Write the rangefile to the specified output stream (default ORNL format)
- unsigned int [write](#) (const char *datafile, size_t format=RANGE_FORMAT_ORNL) const
Write the rangefile to a file (ORNL format)
- unsigned int [getRangeID](#) (float mass) const
Get a range ID from mass to charge.
- void [swap](#) ([RangeFile](#) &rng)
Swap a range file with this one.
- bool [moveRange](#) (unsigned int range, bool limit, float newMass)
Move a range's mass to a new location.
- bool [moveBothRanges](#) (unsigned int range, float newLow, float newHigh)
Move both of a range's masses to a new location.
- unsigned int [addRange](#) (float start, float end, unsigned int ionID)
Add a range to the rangefile. Returns ID number of added range.
- unsigned int [addIon](#) (const std::string &shortName, const std::string &longName, const [RGBf](#) &ionCol)
Add the ion to the database returns ion ID if successful, -1 otherwise.
- bool [setRangeStart](#) (unsigned int rangeID, float v)
Set the lower bound of the given range.
- bool [setRangeEnd](#) (unsigned int rangeID, float v)
Set the upper bound of the given range.
- void [eraseRange](#) (unsigned int rangeID)
Erase given range - this will reorder rangeIDs, so any ids you had previously will not longer be valid.
- void [eraselon](#) (unsigned int ionID)
erase given ions and associated ranges)
- bool [decomposelonById](#) (unsigned int ionID, std::vector< std::pair< std::string, size_t > > &fragments) const
Perform decomposelonNames(...) for a given ionID.
- bool [guessChargeState](#) (unsigned int rangeID, const [AtomProbe::AbundanceData](#) &massTable, unsigned int &charge, float tolerance=0.5f) const
Guess the charge state for the ion that corresponds to the given range's midpoint.

Static Public Member Functions

- static bool [extensionIsRange](#) (const char *ext)
is the extension string the same as that for a range file? I don't advocate this method, but it is convenient in a pinch.
- static void [getAllExts](#) (std::vector< std::string > &exts)
Grab a vector that contains all the extensions that are valid for range files.
- static unsigned int [detectFileType](#) (const char *file)
Attempt to detect the file format of an unknown rangefile.
- static std::string [rangeTypeString](#) (unsigned int rangeType)
Return the human readable name for the given RANGE_FORMAT value.
- static void [setEnforceConsistent](#) (bool shouldEnforce=true)
Set whether the class will attempt to enforce consistency when running.
- static bool [decomposelonNames](#) (const std::string &name, std::vector< std::pair< std::string, size_t > > &fragments)

6.40.1 Detailed Description

Data storage and retrieval class for various range files.

Definition at line 92 of file ranges.h.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 RangeFile()

```
AtomProbe::RangeFile::RangeFile ( )
```

Definition at line 370 of file ranges.cpp.

References `ARRAYSIZE`, and `AtomProbe::RANGE_FORMAT_END_OF_ENUM`.

6.40.3 Member Function Documentation

6.40.3.1 addIon()

```
unsigned int AtomProbe::RangeFile::addIon (
    const std::string & shortName,
    const std::string & longName,
    const RGBf & ionCol )
```

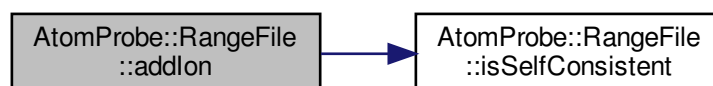
Add the ion to the database returns ion ID if successful, -1 otherwise.

Definition at line 3143 of file ranges.cpp.

References `ASSERT`, and `isSelfConsistent()`.

Referenced by `AtomProbe::checkMassRangingCorrectness()`, `AtomProbe::findOverlaps()`, `getIonID()`, `guessChargeState()`, and `main()`.

Here is the call graph for this function:



6.40.3.2 addRange()

```
unsigned int AtomProbe::RangeFile::addRange (
    float start,
    float end,
    unsigned int ionID )
```

Add a range to the rangefile. Returns ID number of added range.

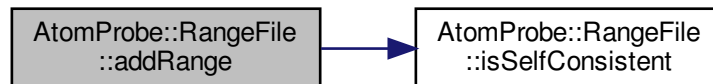
if adding successful, (unsigned int)-1 otherwise. If enforceConsistency is true, this will disallow addition of ranges that collide with existing ranges

Definition at line 3104 of file ranges.cpp.

References ASSERT, and isSelfConsistent().

Referenced by AtomProbe::checkMassRangingCorrectness(), AtomProbe::findOverlaps(), getIonID(), guess↔ChargeState(), and main().

Here is the call graph for this function:



6.40.3.3 decomposeIonById()

```
bool AtomProbe::RangeFile::decomposeIonById (
    unsigned int ionId,
    std::vector< std::pair< std::string, size_t > > & fragments ) const
```

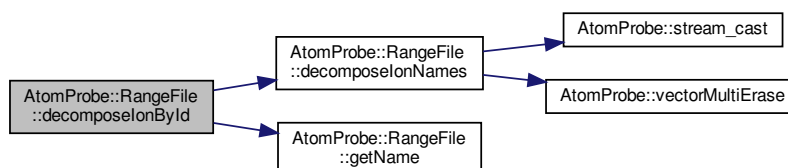
Perform `decomposeIonNames(...)` for a given ionID.

Definition at line 226 of file ranges.cpp.

References `decomposeIonNames()`, and `getName()`.

Referenced by `AtomProbe::findOverlaps()`, and `getIonID()`.

Here is the call graph for this function:



6.40.3.4 decomposeIonNames()

```
bool AtomProbe::RangeFile::decomposeIonNames (
    const std::string & name,
    std::vector< std::pair< std::string, size_t > > & fragments ) [static]
```

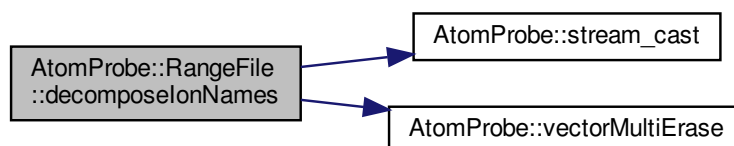
Break a given string down into a series of substring-count pairs depicting basic ionic components returns false if the name cannot be broken down. E.g. H2O will be converted to [{H,2}, {O,1}]

Definition at line 95 of file ranges.cpp.

References AtomProbe::stream_cast(), and AtomProbe::vectorMultiErase().

Referenced by decomposeIonById(), getIonID(), AtomProbe::getRangeMolecule(), guessChargeState(), AtomProbe::matchComposedName(), and AtomProbe::MultiRange::MultiRange().

Here is the call graph for this function:



6.40.3.5 detectFileType()

```
unsigned int AtomProbe::RangeFile::detectFileType (
    const char * file ) [static]
```

Attempt to detect the file format of an unknown rangefile.

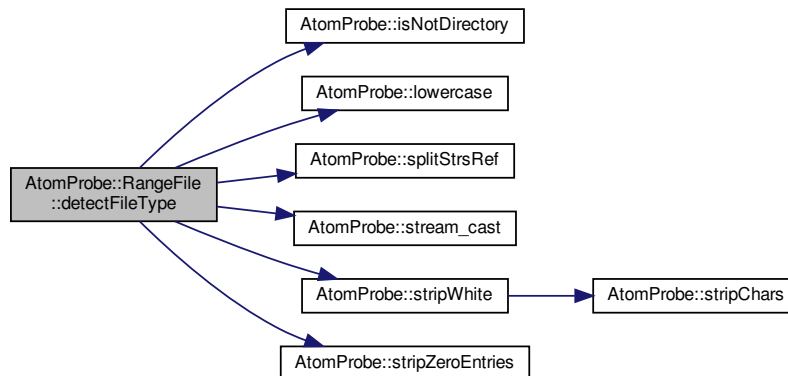
Returns one of RANGE_FORMAT_ enum value on success, or RANGE_FORMAT_END_OF_ENUM on failure

Definition at line 1074 of file ranges.cpp.

References AtomProbe::isNotDirectory(), AtomProbe::lowercase(), AtomProbe::RANGE_FORMAT_DBL_ORNL, AtomProbe::RANGE_FORMAT_END_OF_ENUM, AtomProbe::RANGE_FORMAT_ENV, AtomProbe::RANGE_FORMAT_ORNL, AtomProbe::RANGE_FORMAT_RRNG, AtomProbe::splitStrsRef(), AtomProbe::stream_cast(), AtomProbe::stripWhite(), and AtomProbe::stripZeroEntries().

Referenced by open().

Here is the call graph for this function:



6.40.3.6 eraselon()

```
void AtomProbe::RangeFile::eraseIon (
    unsigned int ionId )
```

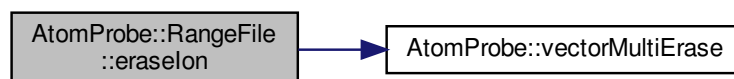
erase given ions and associated ranges)

Definition at line 3183 of file ranges.cpp.

References ASSERT, and `AtomProbe::vectorMultiErase()`.

Referenced by `getIonID()`.

Here is the call graph for this function:



6.40.3.7 eraseRange()

```
void AtomProbe::RangeFile::eraseRange (
    unsigned int rangeId )
```

Erase given range - this will reorder rangeIDs, so any ids you had previously will not longer be valid.

Definition at line 3167 of file ranges.cpp.

References ASSERT.

Referenced by getIonID().

6.40.3.8 extensionIsRange()

```
bool AtomProbe::RangeFile::extensionIsRange (
    const char * ext ) [static]
```

is the extension string the same as that for a range file? I don't advocate this method, but it is convenient in a pinch.

Definition at line 2329 of file ranges.cpp.

6.40.3.9 extractIons()

```
void AtomProbe::RangeFile::extractIons (
    const std::vector< IonHit > & ionHits,
    const std::vector< unsigned int > & ionIDs,
    std::vector< IonHit > & hits ) const
```

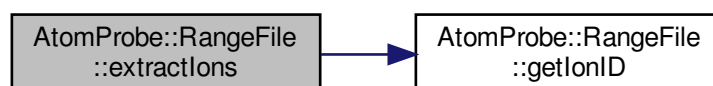
Extract ion hit events in the specified ion ID values, from the input ion ionhit sequence.

Definition at line 2871 of file ranges.cpp.

References getIonID().

Referenced by getIonID().

Here is the call graph for this function:



6.40.3.10 getAllExts()

```
void AtomProbe::RangeFile::getAllExts (
    std::vector< std::string > & exts ) [static]
```

Grab a vector that contains all the extensions that are valid for range files.

Definition at line 2347 of file ranges.cpp.

References ARRAYSIZE, and ASSERT.

6.40.3.11 getColour()

```
RGBf AtomProbe::RangeFile::getColour (
    unsigned int ui ) const
```

Retrieve a given colour from the ion ID.

Definition at line 2774 of file ranges.cpp.

References ASSERT.

Referenced by getErrState(), and AtomProbe::MultiRange::MultiRange().

6.40.3.12 getErrState()

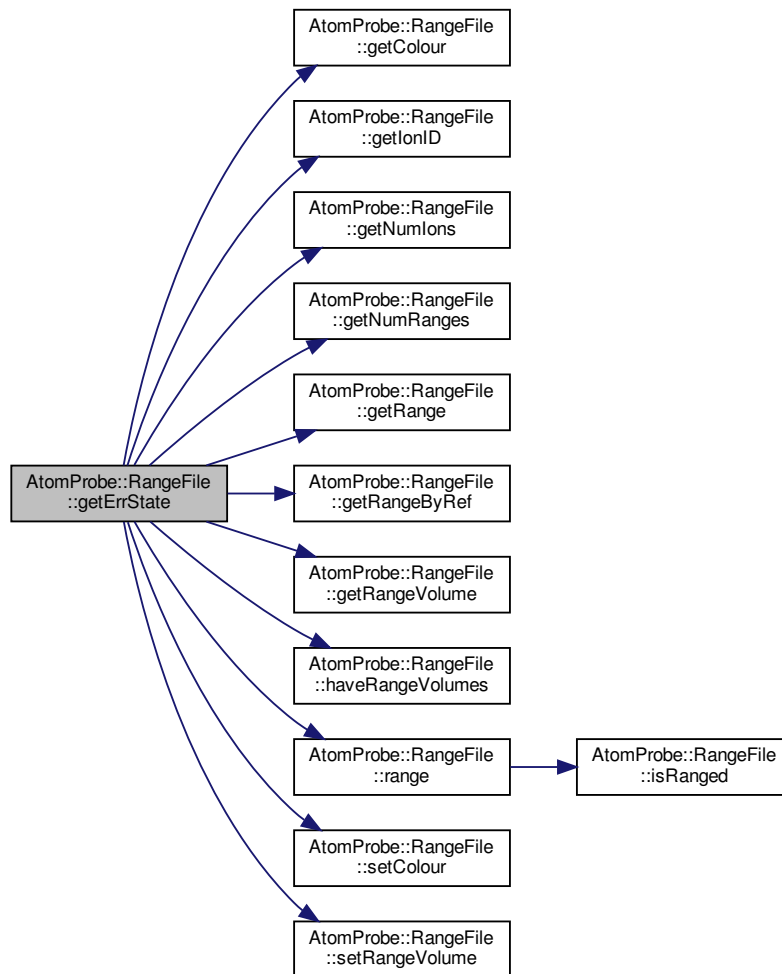
```
unsigned int AtomProbe::RangeFile::getErrState ( ) const [inline]
```

Obtain the error state last set for this rangefile.

Definition at line 198 of file ranges.h.

References getColour(), getIonID(), getNumIons(), getNumRanges(), getRange(), getRangeByRef(), getRangeVolume(), haveRangeVolumes(), range(), setColour(), and setRangeVolume().

Here is the call graph for this function:



6.40.3.13 getErrString()

```
std::string AtomProbe::RangeFile::getErrString ( ) const
```

Retrieve the translated error associated with the current range file state.

Definition at line 3219 of file ranges.cpp.

References `ARRAYSIZE`, and `AtomProbe::RANGE_ERR_ENUM_END`.

Referenced by `main()`, and `setEnforceConsistent()`.

6.40.3.14 `getIonID()` [1/5]

```
unsigned int AtomProbe::RangeFile::getIonID (  
    float mass ) const
```

Retrieve the colour from a given ion ID.

Get the ion's ID from a specified mass

Returns the ions ID if there exists a range that contains this mass. Otherwise (unsigned int)-1 is returned

Definition at line 2780 of file ranges.cpp.

Referenced by `extractIons()`, `getErrState()`, `getIonID()`, `getName()`, `getNumRanges()`, `AtomProbe::getRangeMolecule()`, `guessChargeState()`, `main()`, `AtomProbe::MultiRange::MultiRange()`, and `open()`.

6.40.3.15 `getIonID()` [2/5]

```
unsigned int AtomProbe::RangeFile::getIonID (  
    const IonHit & hit ) const
```

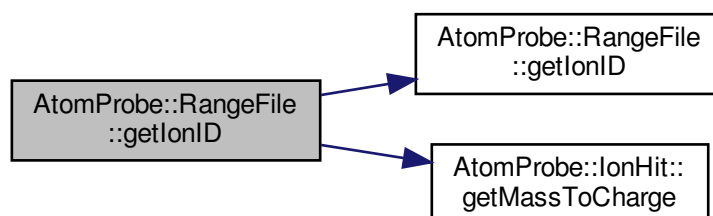
Get the ion's ID from a specified mass, using an [IonHit](#).

Returns the ions ID if there exists a range that contains this mass. Otherwise (unsigned int)-1 is returned

Definition at line 2794 of file ranges.cpp.

References `getIonID()`, and `AtomProbe::IonHit::getMassToCharge()`.

Here is the call graph for this function:



6.40.3.16 getIonID() [3/5]

```
unsigned int AtomProbe::RangeFile::getIonID (
    unsigned int range ) const
```

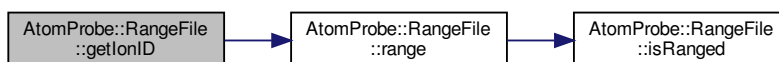
Get the ion ID from a given range ID.

No validation checks are performed outside debug mode. Ion range *must* exist

Definition at line 2814 of file ranges.cpp.

References ASSERT, and range().

Here is the call graph for this function:



6.40.3.17 getIonID() [4/5]

```
unsigned int AtomProbe::RangeFile::getIonID (
    const char * name,
    bool useShortName = true ) const
```

Get the ion ID from its short or long name, returns -1 if name does not exist. Case must match.

Definition at line 2821 of file ranges.cpp.

6.40.3.18 getIonID() [5/5]

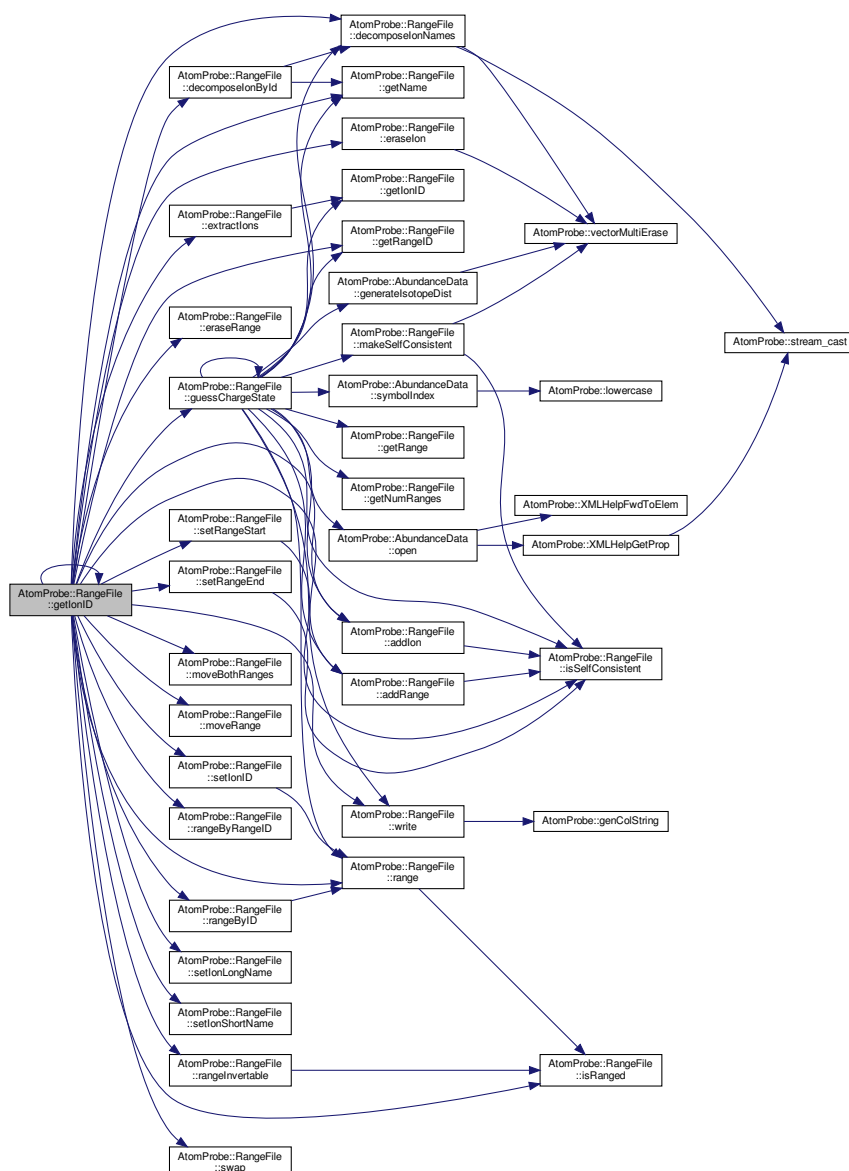
```
unsigned int AtomProbe::RangeFile::getIonID (
    const std::string & name ) const [inline]
```

Definition at line 248 of file ranges.h.

References `addIon()`, `addRange()`, `decomposeIonById()`, `decomposeIonNames()`, `eraseIon()`, `eraseRange()`, `extractIons()`, `getIonID()`, `getName()`, `getRangeID()`, `guessChargeState()`, `isRanged()`, `moveBothRanges()`, `moveRange()`, `range()`, `AtomProbe::RANGE_FORMAT_ORNL`, `rangeById()`, `rangeByRangeID()`, `rangeInvertable()`, `setIonID()`, `setIonLongName()`, `setIonShortName()`, `setRangeEnd()`, `setRangeStart()`, `swap()`, and `write()`.

Referenced by `getIonID()`.

Here is the call graph for this function:



6.40.3.19 getName() [1/2]

```
std::string AtomProbe::RangeFile::getName (
    unsigned int ionID,
    bool shortName = true ) const
```

Get the short name or long name of a specified ionID.

Pass shortname=false to retrieve the long name ionID passed in must exist. No checking outside debug mode

Definition at line 2845 of file ranges.cpp.

References ASSERT.

Referenced by `decomposeIonById()`, `dumpAutocorrelation()`, `getIonID()`, `AtomProbe::getRangeMolecule()`, `guessChargeState()`, `main()`, and `AtomProbe::MultiRange::MultiRange()`.

6.40.3.20 `getName()` [2/2]

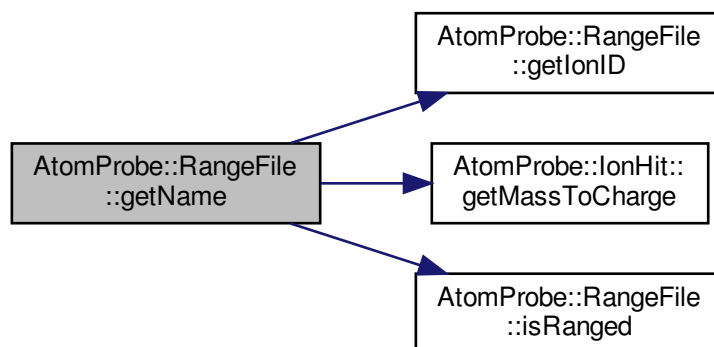
```
std::string AtomProbe::RangeFile::getName (
    const IonHit & ion,
    bool shortName = true ) const
```

Obtain the short name for a given `IonHit`. If not ranged, an empty string will be returned.

Definition at line 2854 of file `ranges.cpp`.

References `getIonID()`, `AtomProbe::IonHit::getMassToCharge()`, and `isRanged()`.

Here is the call graph for this function:



6.40.3.21 `getNumIons()`

```
unsigned int AtomProbe::RangeFile::getNumIons ( ) const
```

Get the number of unique ions.

Definition at line 2759 of file `ranges.cpp`.

Referenced by `dumpAutocorrelation()`, `AtomProbe::findOverlaps()`, `getErrState()`, `main()`, and `AtomProbe::MultiRange::MultiRange()`.

6.40.3.22 `getNumRanges()` [1/2]

```
unsigned int AtomProbe::RangeFile::getNumRanges ( ) const
```

Get the number of unique ranges.

Definition at line 2742 of file ranges.cpp.

Referenced by `AtomProbe::checkMassRangingCorrectness()`, `getErrState()`, `guessChargeState()`, `AtomProbe::MultiRange::MultiRange()`, and `open()`.

6.40.3.23 `getNumRanges()` [2/2]

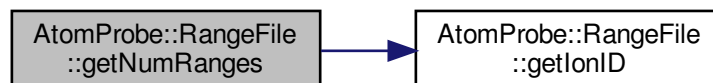
```
unsigned int AtomProbe::RangeFile::getNumRanges (
    unsigned int ionID ) const
```

Get the number of ranges for a given ion ID.

Definition at line 2747 of file ranges.cpp.

References `getIonID()`.

Here is the call graph for this function:



6.40.3.24 `getRange()`

```
pair< float, float > AtomProbe::RangeFile::getRange (
    unsigned int ui ) const
```

Retrieve the start and end of a given range as a pair(start,end)

Definition at line 2764 of file ranges.cpp.

Referenced by `AtomProbe::checkMassRangingCorrectness()`, `getErrState()`, `guessChargeState()`, `AtomProbe::MultiRange::MultiRange()`, and `open()`.

6.40.3.25 getRangeByRef()

```
pair< float, float > & AtomProbe::RangeFile::getRangeByRef (
    unsigned int ui )
```

Retrieve the start and end of a given range as a pair(start,end)

Definition at line 2769 of file ranges.cpp.

Referenced by getErrState().

6.40.3.26 getRangeID()

```
unsigned int AtomProbe::RangeFile::getRangeID (
    float mass ) const
```

Get a range ID from mass to charge.

Definition at line 2800 of file ranges.cpp.

Referenced by getIonID(), and guessChargeState().

6.40.3.27 getRangeVolume()

```
float AtomProbe::RangeFile::getRangeVolume (
    unsigned int rangeId ) const
```

Obtain an ions volume, if we have it. Zero may indicate no ion volume specified for that range.

Definition at line 3285 of file ranges.cpp.

References ASSERT.

Referenced by getErrState().

6.40.3.28 guessChargeState()

```
bool AtomProbe::RangeFile::guessChargeState (
    unsigned int rangeId,
    const AtomProbe::AbundanceData & massTable,
    unsigned int & charge,
    float tolerance = 0.5f ) const
```

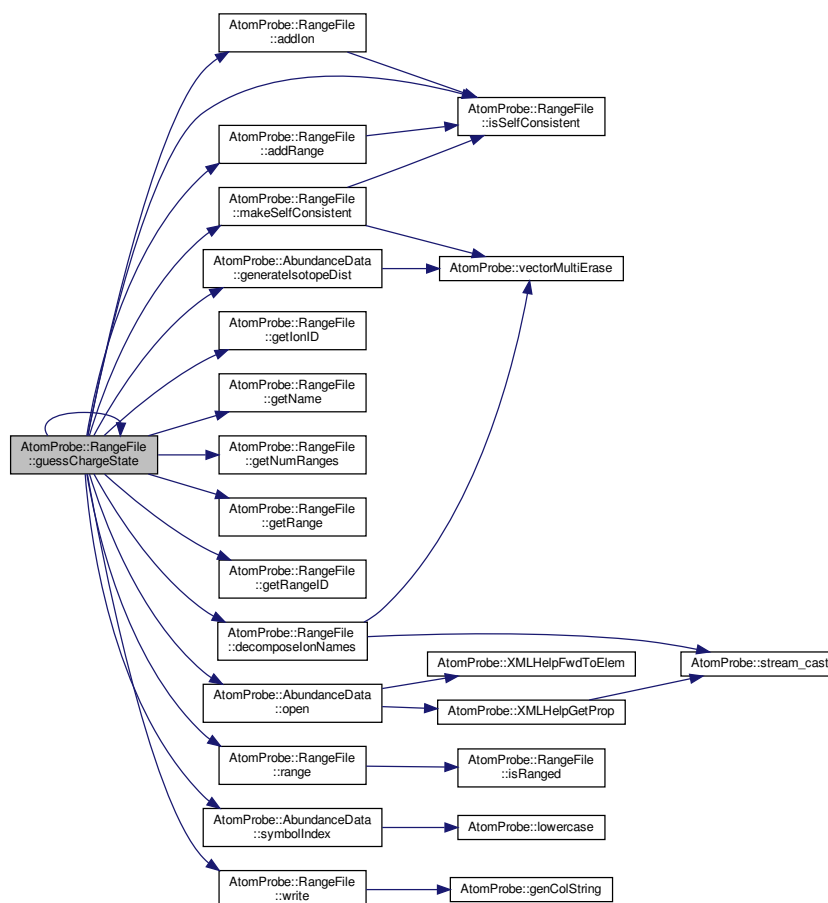
Guess the charge state for the ion that corresponds to the given range's midpoint.

Definition at line 3300 of file ranges.cpp.

References addlon(), addRange(), AtomProbe::RGBf::blue, decomposeIonNames(), AtomProbe::AbundanceData::generateIsotopeDist(), getIonID(), getName(), getNumRanges(), getRange(), getRangeID(), AtomProbe::RGBf::green, guessChargeState(), isSelfConsistent(), makeSelfConsistent(), AtomProbe::AbundanceData::open(), range(), AtomProbe::RANGE_FORMAT_DBL_ORNL, AtomProbe::RANGE_FORMAT_END_OF_ENUM, AtomProbe::RGBf::red, AtomProbe::AbundanceData::symbolIndex(), TEST, and write().

Referenced by getIonID(), guessChargeState(), and main().

Here is the call graph for this function:



6.40.3.29 haveRangeVolumes()

```
bool AtomProbe::RangeFile::haveRangeVolumes ( ) const
```

Return true if we have range volume data, false otherwise.

Definition at line 3295 of file ranges.cpp.

Referenced by getErrState().

6.40.3.30 isRanged() [1/3]

```
bool AtomProbe::RangeFile::isRanged (
    float mass ) const
```

Returns true if a specified mass is ranged.

Definition at line 2609 of file ranges.cpp.

Referenced by getIonID(), getName(), isRanged(), range(), and rangeInvertable().

6.40.3.31 isRanged() [2/3]

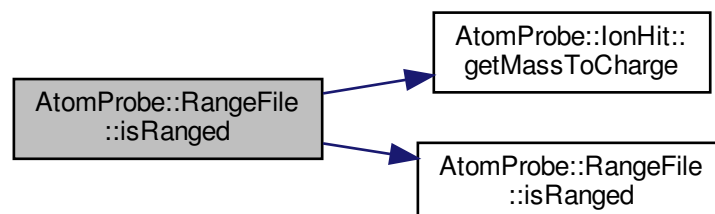
```
bool AtomProbe::RangeFile::isRanged (
    const IonHit & ion ) const
```

Returns true if an ion is ranged.

Definition at line 2623 of file ranges.cpp.

References AtomProbe::IonHit::getMassToCharge(), and isRanged().

Here is the call graph for this function:



6.40.3.32 isRanged() [3/3]

```
bool AtomProbe::RangeFile::isRanged (
    std::string shortName,
    bool caseSensitive = true )
```

Check to see if an atom is ranged.

Returns true if rangefile holds at least one range with shortname corresponding input value. Case sensitivity search is default

Definition at line 2894 of file ranges.cpp.

6.40.3.33 isSelfConsistent() [1/2]

```
bool AtomProbe::RangeFile::isSelfConsistent ( ) const
```

Performs checks for self consistency.

Definition at line 2360 of file ranges.cpp.

References ASSERT.

Referenced by addlon(), addRange(), guessChargeState(), main(), makeSelfConsistent(), open(), openFormat(), runTests(), setEnforceConsistent(), setRangeEnd(), and setRangeStart().

6.40.3.34 isSelfConsistent() [2/2]

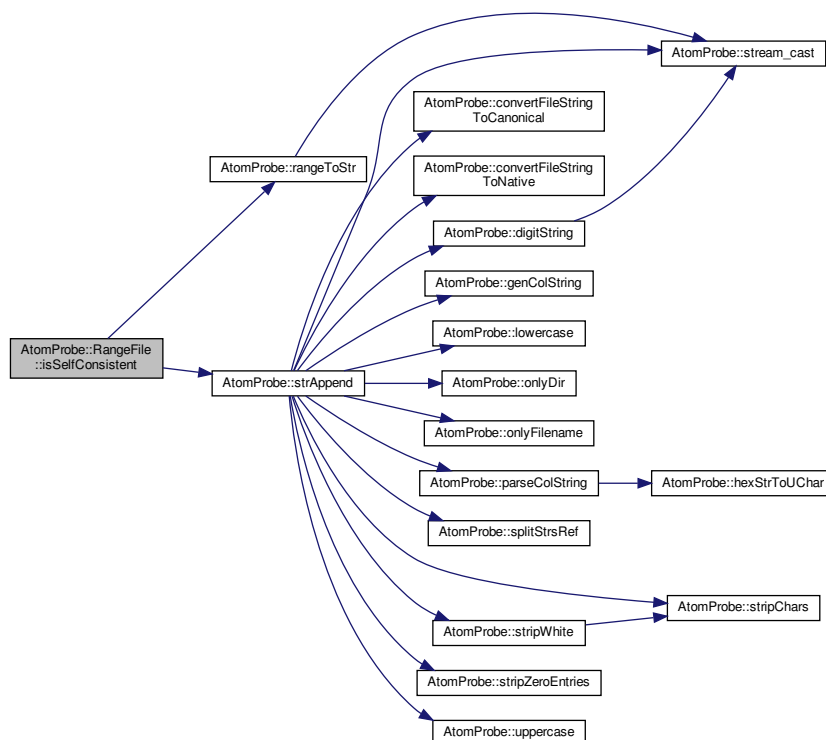
```
bool AtomProbe::RangeFile::isSelfConsistent (
    std::vector< std::string > & consistencyMessages ) const
```

Performs checks for self consistency, whilst providing information on the inconsistent components.

Definition at line 2487 of file ranges.cpp.

References AtomProbe::rangeToStr(), and AtomProbe::strAppend().

Here is the call graph for this function:



6.40.3.35 makeSelfConsistent()

```
bool AtomProbe::RangeFile::makeSelfConsistent ( )
```

Modify range file to ensure self-consistency, by arbitrary heuristics.

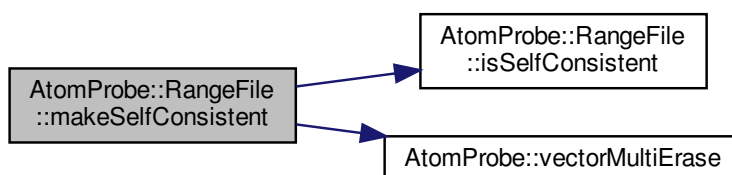
Returns true if self consistent after corrections. false otherwise

Definition at line 2432 of file ranges.cpp.

References `isSelfConsistent()`, and `AtomProbe::vectorMultiErase()`.

Referenced by `guessChargeState()`, and `setEnforceConsistent()`.

Here is the call graph for this function:



6.40.3.36 moveBothRanges()

```
bool AtomProbe::RangeFile::moveBothRanges (
    unsigned int range,
    float newLow,
    float newHigh )
```

Move both of a range's masses to a new location.

Definition at line 3066 of file ranges.cpp.

Referenced by AtomProbe::checkMassRangingCorrectness(), and getIonID().

6.40.3.37 moveRange()

```
bool AtomProbe::RangeFile::moveRange (
    unsigned int range,
    bool limit,
    float newMass )
```

Move a range's mass to a new location.

Definition at line 3005 of file ranges.cpp.

Referenced by getIonID().

6.40.3.38 open()

```
bool AtomProbe::RangeFile::open (
    const char * rangeFile )
```

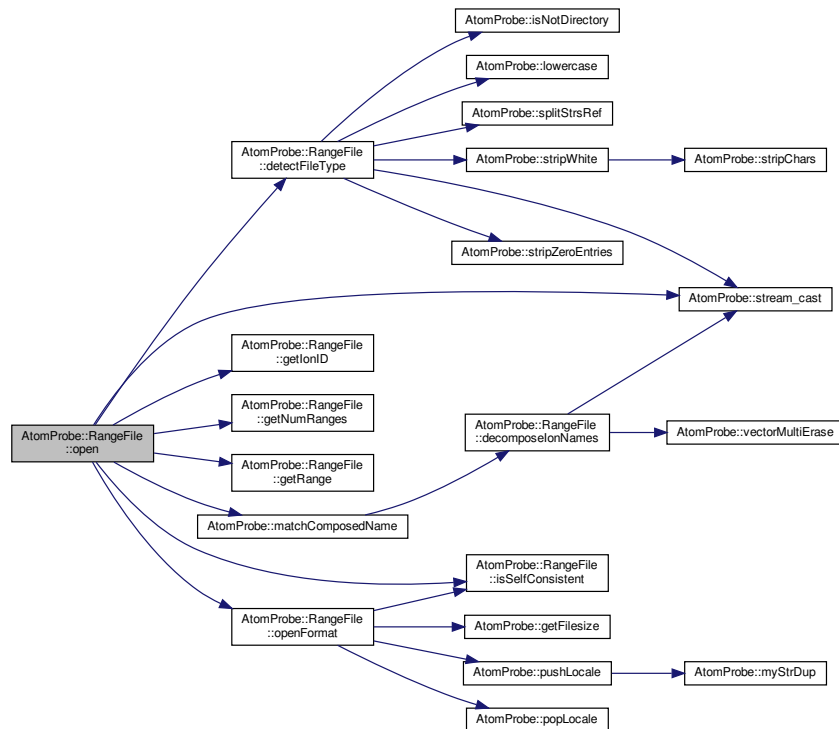
Open a specified range file - returns true on success.

Definition at line 637 of file ranges.cpp.

References ASSERT, AtomProbe::RGBf::blue, detectFileType(), getIonID(), getNumRanges(), getRange(), AtomProbe::RGBf::green, isSelfConsistent(), AtomProbe::matchComposedName(), AtomProbe::MAX_LINE_SIZE, openFormat(), AtomProbe::RANGE_ERR_DASHHEADER, AtomProbe::RANGE_ERR_DATA_NOMAPPED_IONNAME, AtomProbe::RANGE_ERR_FORMAT, AtomProbe::RANGE_ERR_FORMAT_EMPTY_RANGEROW, AtomProbe::RANGE_ERR_FORMAT_TABLESEPARATOR, AtomProbe::RANGE_ERR_NONUNIQUE_POLYATOMIC, AtomProbe::RANGE_FORMAT_END_OF_ENUM, AtomProbe::RGBf::red, and AtomProbe::stream_cast().

Referenced by main(), and runTests().

Here is the call graph for this function:



6.40.3.39 openFormat()

```

unsigned int AtomProbe::RangeFile::openFormat (
    const char * rangeFile,
    unsigned int format )
  
```

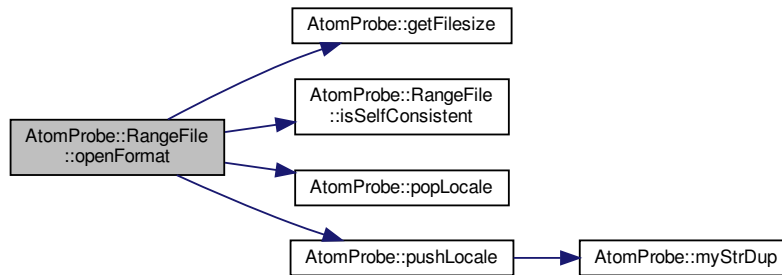
Open a specified range file using a given file format. Returns nonzero on failure.

Definition at line 568 of file ranges.cpp.

References ASSERT, AtomProbe::getFilesize(), isSelfConsistent(), AtomProbe::popLocale(), AtomProbe::pushLocale(), AtomProbe::RANGE_ERR_DATA_INCONSISTENT, AtomProbe::RANGE_ERR_FILESIZE, AtomProbe::RANGE_ERR_FORMAT, AtomProbe::RANGE_ERR_OPEN, AtomProbe::RANGE_FORMAT_DBL_ORNL, AtomProbe::RANGE_FORMAT_ENV, AtomProbe::RANGE_FORMAT_ORNL, and AtomProbe::RANGE_FORMAT_AT_RRNG.

Referenced by open().

Here is the call graph for this function:



6.40.3.40 operator=()

```
const RangeFile & AtomProbe::RangeFile::operator= (
    const RangeFile & other )
```

Definition at line 375 of file ranges.cpp.

6.40.3.41 range() [1/2]

```
void AtomProbe::RangeFile::range (
    std::vector< IonHit > & ionHits ) const
```

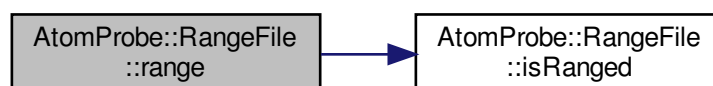
Clips out ions that are not inside the rangefile's ranges.

Definition at line 2679 of file ranges.cpp.

References `isRanged()`.

Referenced by `getErrState()`, `getIonID()`, `guessChargeState()`, `main()`, `rangeByID()`, and `setIonID()`.

Here is the call graph for this function:



6.40.3.42 range() [2/2]

```
bool AtomProbe::RangeFile::range (
    std::vector< IonHit > & ionHits,
    const std::string & shortIonName )
```

Clips out ions that don't match the specified ion name.

Returns false if the ion name given doesn't match any in the rangefile (case sensitive)

Definition at line 2628 of file ranges.cpp.

6.40.3.43 rangeByID()

```
bool AtomProbe::RangeFile::rangeByID (
    std::vector< IonHit > & ionHits,
    unsigned int range )
```

Clips out ions that don't lie in the specified range number.

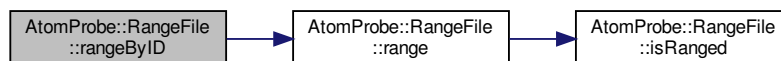
Returns false if the range does not exist any in the rangefile (case sensitive)

Definition at line 2865 of file ranges.cpp.

References range().

Referenced by getIonID().

Here is the call graph for this function:



6.40.3.44 rangeByRangeID()

```
void AtomProbe::RangeFile::rangeByRangeID (
    std::vector< IonHit > & ionHits,
    unsigned int rangeID )
```

Range the given ions, allowing only ions in the specified range. Input data will be modified to return result.

Definition at line 2718 of file ranges.cpp.

Referenced by getIonID().

6.40.3.45 rangeInvertable()

```
void AtomProbe::RangeFile::rangeInvertable (
    std::vector< IonHit > & ionHits,
    bool invert )
```

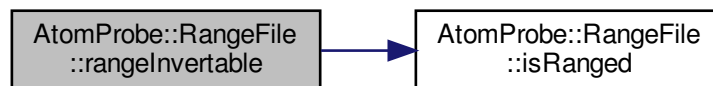
Clips out ions that are inside the rangefile's ranges, if invert is false. If true, then reverse the set.

Definition at line 2699 of file ranges.cpp.

References isRanged(), and XOR.

Referenced by getIonID().

Here is the call graph for this function:



6.40.3.46 rangeTypeString()

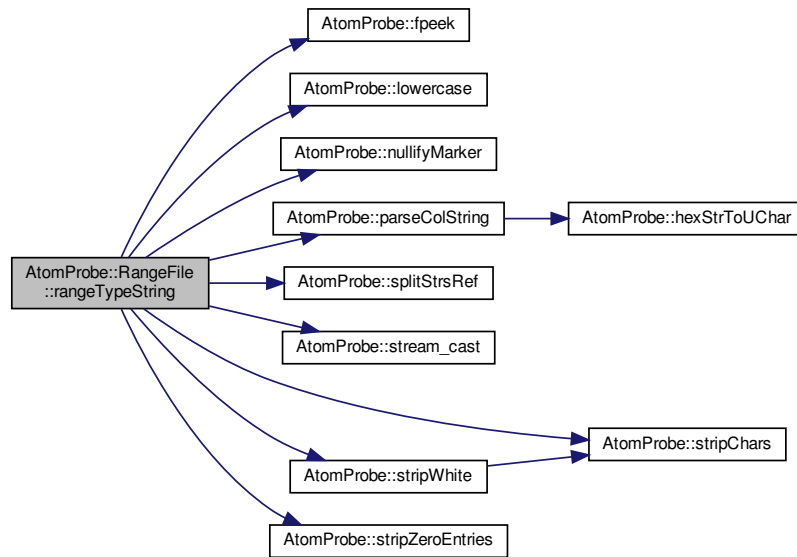
```
string AtomProbe::RangeFile::rangeTypeString (
    unsigned int rangeType ) [static]
```

Return the human readable name for the given RANGE_FORMAT value.

Definition at line 1298 of file ranges.cpp.

References ARRAYSIZE, ASSERT, AtomProbe::RGBf::blue, AtomProbe::fpeek(), AtomProbe::RGBf::green, AtomProbe::lowercase(), AtomProbe::MAX_LINE_SIZE, AtomProbe::nullifyMarker(), AtomProbe::parseColString(), AtomProbe::RANGE_ERR_BAD_MULTPLICITY, AtomProbe::RANGE_ERR_BADSTART, AtomProbe::RANGE_ERR_DATA_FLIPPED, AtomProbe::RANGE_ERR_DATA_TOO_MANY_USELESS_RANGES, AtomProbe::RANGE_ERR_DUPLICATE_NUMRANGES, AtomProbe::RANGE_ERR_EMPTY, AtomProbe::RANGE_ERR_FORMAT, AtomProbe::RANGE_ERR_FORMAT_COLOUR, AtomProbe::RANGE_ERR_FORMAT_HEADER, AtomProbe::RANGE_ERR_FORMAT_LONGNAME, AtomProbe::RANGE_ERR_FORMAT_MASS_PAIR, AtomProbe::RANGE_ERR_FORMAT_RANGETABLE, AtomProbe::RANGE_ERR_FORMAT_SHORTNAME, AtomProbe::RANGE_ERR_FORMAT_TABLE_ENTRY, AtomProbe::RANGE_ERR_FORMAT_TABLEHEADER_NUMIONS, AtomProbe::RANGE_ERR_FORMAT_TABLESEPARATOR, AtomProbe::RANGE_ERR_ION_BLOCK_NOT_PRESENT, AtomProbe::RANGE_ERR_ION_NOT_MAPPED, AtomProbe::RANGE_ERR_IONBLOCK_CONTENT, AtomProbe::RANGE_ERR_NUMIONS, AtomProbe::RANGE_ERR_NUMIONS_DUPLICATED, AtomProbe::RANGE_ERR_NUMRANGE_PARSE, AtomProbe::RANGE_ERR_RANGEBLOCK_FORMAT, AtomProbe::RANGE_ERR_RRNG_COLON_SEPARATOR, AtomProbe::RANGE_ERR_RRNG_IONS_TOOSHORT, AtomProbe::RANGE_ERR_TOO_MANYIONS, AtomProbe::RANGE_FORMAT_END_OF_ENUM, AtomProbe::RGBf::red, AtomProbe::splitStrsRef(), AtomProbe::stream_cast(), AtomProbe::stripChars(), AtomProbe::stripWhite(), and AtomProbe::stripZeroEntries().

Here is the call graph for this function:



6.40.3.47 setColour()

```
void AtomProbe::RangeFile::setColour (
    unsigned int id,
    const RGBF & r )
```

Set the colour using the ion ID.

Definition at line 2936 of file ranges.cpp.

References ASSERT.

Referenced by `getErrState()`.

6.40.3.48 setEnforceConsistent()

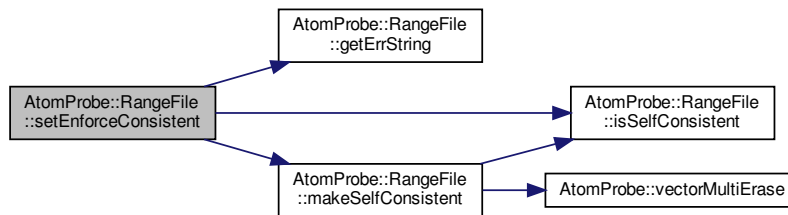
```
static void AtomProbe::RangeFile::setEnforceConsistent (
    bool shouldEnforce = true ) [inline], [static]
```

Set whether the class will attempt to enforce consistency when running.

Definition at line 181 of file ranges.h.

References `getErrString()`, `isSelfConsistent()`, and `makeSelfConsistent()`.

Here is the call graph for this function:



6.40.3.49 setIonID()

```
void AtomProbe::RangeFile::setIonID (
    unsigned int range,
    unsigned int newIonId )
```

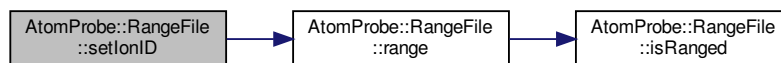
Set the ion ID for a given range.

Definition at line 3161 of file ranges.cpp.

References ASSERT, and range().

Referenced by getIonID().

Here is the call graph for this function:



6.40.3.50 setIonLongName()

```
void AtomProbe::RangeFile::setIonLongName (
    unsigned int ionID,
    const std::string & newName )
```

Set the long name for a given ion.

Definition at line 2948 of file ranges.cpp.

Referenced by getIonID().

6.40.3.51 setIonShortName()

```
void AtomProbe::RangeFile::setIonShortName (
    unsigned int ionID,
    const std::string & newName )
```

set the short name for a given ion

Definition at line 2943 of file ranges.cpp.

Referenced by getIonID().

6.40.3.52 setRangeEnd()

```
bool AtomProbe::RangeFile::setRangeEnd (
    unsigned int rangeID,
    float v )
```

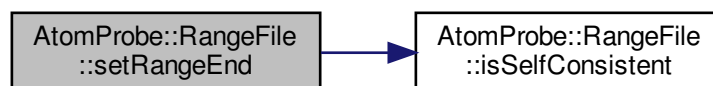
Set the upper bound of the given range.

Definition at line 2973 of file ranges.cpp.

References ASSERT, and isSelfConsistent().

Referenced by getIonID().

Here is the call graph for this function:



6.40.3.53 setRangeStart()

```
bool AtomProbe::RangeFile::setRangeStart (
    unsigned int rangeID,
    float v )
```

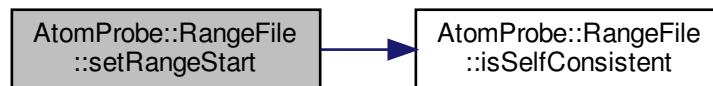
Set the lower bound of the given range.

Definition at line 2953 of file ranges.cpp.

References ASSERT, and isSelfConsistent().

Referenced by getIonID().

Here is the call graph for this function:

**6.40.3.54 setRangeVolume()**

```
void AtomProbe::RangeFile::setRangeVolume (
    unsigned int rangeId,
    float newVolume )
```

Set the ion volume for a given range. If ion volumes are not tracked, these will be created.

Definition at line 3274 of file ranges.cpp.

References ASSERT.

Referenced by getErrState().

6.40.3.55 swap()

```
void AtomProbe::RangeFile::swap (
    RangeFile & rng )
```

Swap a range file with this one.

Definition at line 2993 of file ranges.cpp.

Referenced by getIonID().

6.40.3.56 write() [1/2]

```
unsigned int AtomProbe::RangeFile::write (  
    std::ostream & o,  
    size_t format = RANGE_FORMAT_ORNL ) const
```

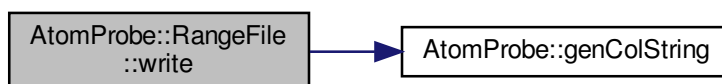
Write the rangefile to the specified output stream (default ORNL format)

Definition at line 396 of file ranges.cpp.

References ASSERT, AtomProbe::genColString(), AtomProbe::RANGE_FORMAT_ENV, AtomProbe::RANGE_FORMAT_ORNL, and AtomProbe::RANGE_FORMAT_RRNG.

Referenced by getlonID(), guessChargeState(), and write().

Here is the call graph for this function:



6.40.3.57 write() [2/2]

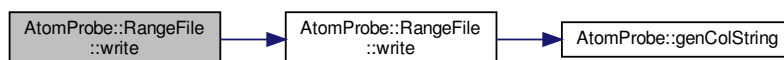
```
unsigned int AtomProbe::RangeFile::write (  
    const char * datafile,  
    size_t format = RANGE_FORMAT_ORNL ) const
```

Write the rangefile to a file (ORNL format)

Definition at line 547 of file ranges.cpp.

References write().

Here is the call graph for this function:



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

- [include/atomprobe/io/ranges.h](#)
- [src/io/ranges.cpp](#)

6.41 AtomProbe::ReconstructionSphereOnCone Class Reference

```
#include <reconstruction-simple.h>
```

Public Member Functions

- [ReconstructionSphereOnCone](#) ()
Default constructor.
- void [setProjModel](#) ([SphericPlaneProjection](#) *model)
Set a given projection model.
- void [setRadiusEvolutionMode](#) (unsigned int evolutionMode)
Set the method of evolution for the shank, as per EVOLUTION_MODE_ enum.
- void [setInitialRadius](#) (float rInit)
Set the initial tip radius.
- void [setShankAngle](#) (float angle)
Set initial angle, in radians.
- void [setDetectorEfficiency](#) (float detEff)
Set detector efficiency in range (0,1].
- void [setFlightPath](#) (float detEff)
Set flight path. Units must match detector units (eg mm)
- void [setReconFOV](#) (float angle)
Set the angle (between 0 and pi radians). This is.
- bool [reconstruct](#) (const std::vector< float > &detectorX, const std::vector< float > &detectorY, const std::vector< float > &timeOfFlight, const std::vector< float > &ionVolume, std::vector< [IonHit](#) > &outputPts)
const
Using the current reconstruction model, reconstruct a detector sequence.

6.41.1 Detailed Description

Definition at line 35 of file reconstruction-simple.h.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 ReconstructionSphereOnCone()

```
AtomProbe::ReconstructionSphereOnCone::ReconstructionSphereOnCone ( )
```

Default constructor.

Definition at line 14 of file reconstruction-simple.cpp.

References [AtomProbe::EVOLUTION_MODE_SHANKANGLE](#).

6.41.3 Member Function Documentation

6.41.3.1 reconstruct()

```
bool AtomProbe::ReconstructionSphereOnCone::reconstruct (
    const std::vector< float > & detectorX,
    const std::vector< float > & detectorY,
    const std::vector< float > & timeOfFlight,
    const std::vector< float > & ionVolume,
    std::vector< IonHit > & outputPts ) const
```

Using the current reconstruction model, reconstruct a detector sequence.

Uses the projection model given in `::setProjModel`, to perform the reconstruction

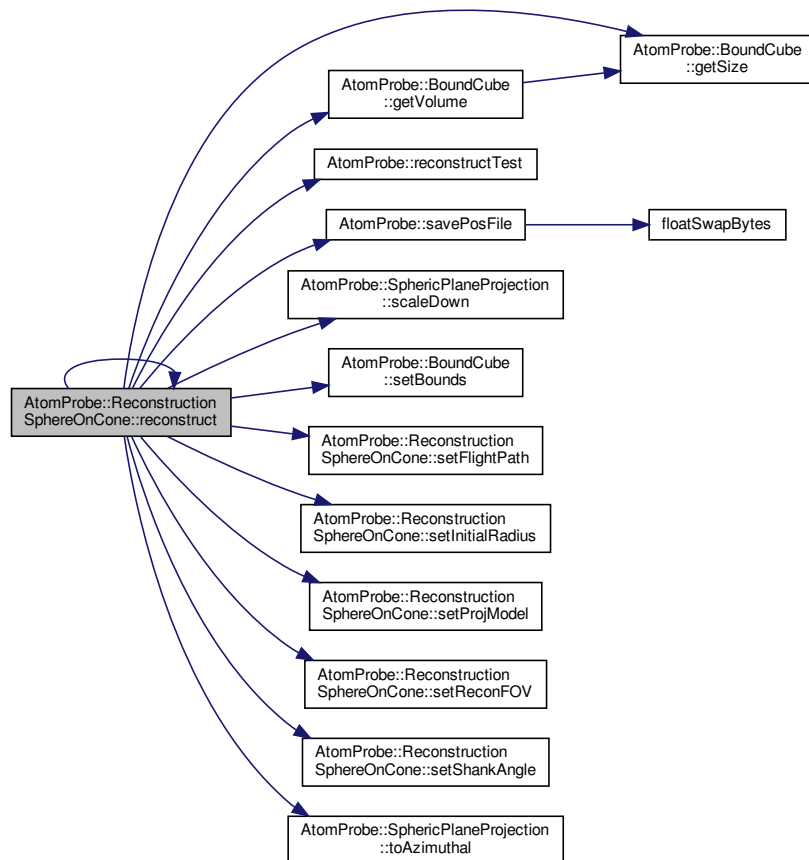
- This is a "shank" reconstruction
- Detector sequence input should typically be centered around 0,0 - automatic recentring may not be desired, so this will not be performed
- Sequence scale should match flight path (e.g. mm)
- Ions may not be reconstructed if outside FOV, so $|\text{outputPts}| < |\text{detector events}|$

Definition at line 62 of file reconstruction-simple.cpp.

References ASSERT, AtomProbe::EVOLUTION_MODE_SHANKANGLE, AtomProbe::BoundCube::getSize(), AtomProbe::BoundCube::getVolume(), M_PI, reconstruct(), AtomProbe::reconstructTest(), AtomProbe::savePos↔File(), AtomProbe::SphericPlaneProjection::scaleDown(), AtomProbe::BoundCube::setBounds(), setFlightPath(), setInitialRadius(), setProjModel(), setReconFOV(), setShankAngle(), TEST, and AtomProbe::SphericPlane↔Projection::toAzimuthal().

Referenced by reconstruct().

Here is the call graph for this function:



6.41.3.2 setDetectorEfficiency()

```
void AtomProbe::ReconstructionSphereOnCone::setDetectorEfficiency (
    float detEff )
```

Set detector efficiency in range (0,1].

Definition at line 31 of file reconstruction-simple.cpp.

References ASSERT.

6.41.3.3 setFlightPath()

```
void AtomProbe::ReconstructionSphereOnCone::setFlightPath (
    float detEff )
```

Set flight path. Units must match detector units (eg mm)

Definition at line 37 of file reconstruction-simple.cpp.

References ASSERT.

Referenced by reconstruct().

6.41.3.4 setInitialRadius()

```
void AtomProbe::ReconstructionSphereOnCone::setInitialRadius (
    float rInit )
```

Set the initial tip radius.

Definition at line 56 of file reconstruction-simple.cpp.

References ASSERT.

Referenced by reconstruct().

6.41.3.5 setProjModel()

```
void AtomProbe::ReconstructionSphereOnCone::setProjModel (
    SphericPlaneProjection * model )
```

Set a given projection model.

Definition at line 21 of file reconstruction-simple.cpp.

Referenced by reconstruct().

6.41.3.6 setRadiusEvolutionMode()

```
void AtomProbe::ReconstructionSphereOnCone::setRadiusEvolutionMode (
    unsigned int evolutionMode )
```

Set the method of evolution for the shank, as per EVOLUTION_MODE_ enum.

Definition at line 26 of file reconstruction-simple.cpp.

6.41.3.7 setReconFOV()

```
void AtomProbe::ReconstructionSphereOnCone::setReconFOV (
    float angle )
```

Set the angle (between 0 and pi radians). This is.

Definition at line 43 of file reconstruction-simple.cpp.

References ASSERT, and M_PI.

Referenced by reconstruct().

6.41.3.8 setShankAngle()

```
void AtomProbe::ReconstructionSphereOnCone::setShankAngle (
    float angle )
```

Set initial angle, in radians.

Definition at line 49 of file reconstruction-simple.cpp.

References ASSERT, and M_PI.

Referenced by reconstruct().

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

- include/atomprobe/reconstruction/[reconstruction-simple.h](#)
- src/reconstruction/[reconstruction-simple.cpp](#)

6.42 AtomProbe::RGBf Class Reference

Data holder for colour as float.

```
#include <misc.h>
```

Public Member Functions

- std::string [toHex](#) () const
Convert to hex-256 representation #rrggb.
- void [fromHex](#) (const std::string &s)

Public Attributes

- float [red](#)
- float [green](#)
- float [blue](#)

6.42.1 Detailed Description

Data holder for colour as float.

Definition at line 26 of file misc.h.

6.42.2 Member Function Documentation

6.42.2.1 fromHex()

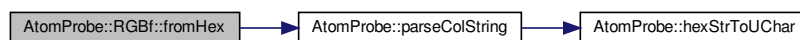
```
void AtomProbe::RGBf::fromHex (  
    const std::string & s ) [inline]
```

Definition at line 40 of file misc.h.

References AtomProbe::parseColString().

Referenced by AtomProbe::MultiRange::copyDataFromRange().

Here is the call graph for this function:



6.42.2.2 toHex()

```
std::string AtomProbe::RGBf::toHex ( ) const [inline]
```

Convert to hex-256 representation #rrggbb.

Definition at line 34 of file misc.h.

References AtomProbe::genColString().

Here is the call graph for this function:



6.42.3 Member Data Documentation

6.42.3.1 blue

```
float AtomProbe::RGBf::blue
```

Definition at line 31 of file misc.h.

Referenced by `AtomProbe::checkMassRangingCorrectness()`, `AtomProbe::computeRangeAdjacency()`, `AtomProbe::MultiRange::copyDataFromRange()`, `AtomProbe::findOverlaps()`, `AtomProbe::RangeFile::guessChargeState()`, `AtomProbe::leastSquaresDeconvolve()`, `main()`, `AtomProbe::MultiRange::open()`, `AtomProbe::RangeFile::open()`, and `AtomProbe::RangeFile::rangeTypeString()`.

6.42.3.2 green

```
float AtomProbe::RGBf::green
```

Definition at line 30 of file misc.h.

Referenced by `AtomProbe::checkMassRangingCorrectness()`, `AtomProbe::computeRangeAdjacency()`, `AtomProbe::MultiRange::copyDataFromRange()`, `AtomProbe::findOverlaps()`, `AtomProbe::RangeFile::guessChargeState()`, `AtomProbe::leastSquaresDeconvolve()`, `main()`, `AtomProbe::MultiRange::open()`, `AtomProbe::RangeFile::open()`, and `AtomProbe::RangeFile::rangeTypeString()`.

6.42.3.3 red

```
float AtomProbe::RGBf::red
```

Definition at line 29 of file misc.h.

Referenced by `AtomProbe::checkMassRangingCorrectness()`, `AtomProbe::computeRangeAdjacency()`, `AtomProbe::MultiRange::copyDataFromRange()`, `AtomProbe::findOverlaps()`, `AtomProbe::RangeFile::guessChargeState()`, `AtomProbe::leastSquaresDeconvolve()`, `main()`, `AtomProbe::MultiRange::open()`, `AtomProbe::RangeFile::open()`, and `AtomProbe::RangeFile::rangeTypeString()`.

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

- `include/atomprobe/helper/misc.h`

6.43 AtomProbe::SIMPLE_SPECIES Struct Reference

```
#include <multiRange.h>
```


Public Member Functions

- bool `operator==` (const `SIMPLE_SPECIES` &oth) const

Public Attributes

- unsigned int `atomicNumber`
Number of protons (element number)
- unsigned int `count`
Number of copies of this isotope.

6.43.1 Detailed Description

Definition at line 34 of file `multiRange.h`.

6.43.2 Member Function Documentation

6.43.2.1 `operator==()`

```
bool AtomProbe::SIMPLE_SPECIES::operator== (
    const SIMPLE_SPECIES & oth ) const
```

Definition at line 112 of file `multiRange.cpp`.

References `atomicNumber`, and `count`.

6.43.3 Member Data Documentation

6.43.3.1 `atomicNumber`

```
unsigned int AtomProbe::SIMPLE_SPECIES::atomicNumber
```

Number of protons (element number)

Definition at line 37 of file `multiRange.h`.

Referenced by `AtomProbe::computeRangeAdjacency()`, `AtomProbe::MultiRange::copyDataFromRange()`, `AtomProbe::leastSquaresDeconvolve()`, `AtomProbe::MultiRange::MultiRange()`, `AtomProbe::MultiRange::open()`, `AtomProbe::operator<()`, and `operator==()`.

6.43.3.2 count

```
unsigned int AtomProbe::SIMPLE_SPECIES::count
```

Number of copies of this isotope.

Definition at line 40 of file multiRange.h.

Referenced by AtomProbe::computeRangeAdjacency(), AtomProbe::MultiRange::copyDataFromRange(), AtomProbe::leastSquaresDeconvolve(), AtomProbe::MultiRange::MultiRange(), AtomProbe::MultiRange::open(), AtomProbe::operator<(), and operator==().

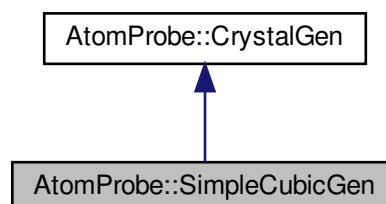
The documentation for this struct was generated from the following files:

- [include/atomprobe/io/multiRange.h](#)
- [src/io/multiRange.cpp](#)

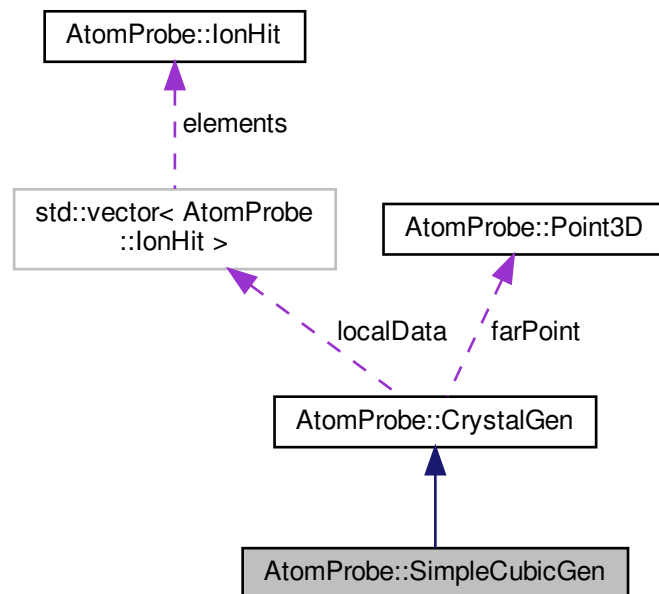
6.44 AtomProbe::SimpleCubicGen Class Reference

```
#include <generate.h>
```

Inheritance diagram for AtomProbe::SimpleCubicGen:



Collaboration diagram for AtomProbe::SimpleCubicGen:



Public Member Functions

- [SimpleCubicGen](#) (float latticeSpacing, float massToC, const [Point3D](#) &p)
- unsigned int [generateLattice](#) ()

Additional Inherited Members

6.44.1 Detailed Description

Definition at line 40 of file generate.h.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 SimpleCubicGen()

```

AtomProbe::SimpleCubicGen::SimpleCubicGen (
    float latticeSpacing,
    float massToC,
    const Point3D & p )
  
```

Definition at line 46 of file generate.cpp.

References [AtomProbe::CrystalGen::farPoint](#).

6.44.3 Member Function Documentation

6.44.3.1 generateLattice()

```
unsigned int AtomProbe::SimpleCubicGen::generateLattice ( ) [virtual]
```

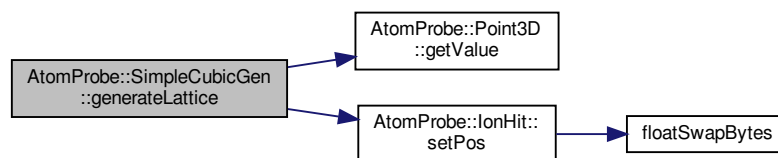
Implements [AtomProbe::CrystalGen](#).

Definition at line 53 of file generate.cpp.

References [AtomProbe::CRYSTAL_BAD_UNIT_CELL](#), [AtomProbe::CrystalGen::farPoint](#), [AtomProbe::Point3D::getValue\(\)](#), [AtomProbe::CrystalGen::localData](#), and [AtomProbe::IonHit::setPos\(\)](#).

Referenced by [AtomProbe::BodyCentredCubicGen::generateLattice\(\)](#).

Here is the call graph for this function:



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

- [include/atomprobe/latticeplanes/generate.h](#)
- [src/lattice/generate.cpp](#)

6.45 AtomProbe::SINGLE_HIT Struct Reference

Single 3DAP hit event.

```
#include <dataFiles.h>
```

Public Attributes

- [float x](#)
Position on the detector.
- [float y](#)
- [float tof](#)
Reported time of flight.

6.45.1 Detailed Description

Single 3DAP hit event.

Definition at line 197 of file dataFiles.h.

6.45.2 Member Data Documentation

6.45.2.1 tof

```
float AtomProbe::SINGLE_HIT::tof
```

Reported time of flight.

Definition at line 202 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

6.45.2.2 x

```
float AtomProbe::SINGLE_HIT::x
```

Position on the detector.

Definition at line 200 of file dataFiles.h.

6.45.2.3 y

```
float AtomProbe::SINGLE_HIT::y
```

Definition at line 200 of file dataFiles.h.

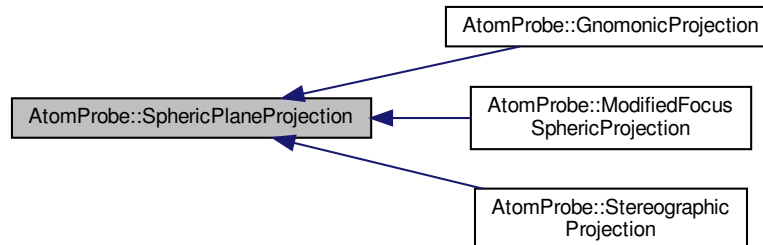
The documentation for this struct was generated from the following file:

- [include/atomprobe/io/dataFiles.h](#)

6.46 AtomProbe::SphericPlaneProjection Class Reference

```
#include <projection.h>
```

Inheritance diagram for AtomProbe::SphericPlaneProjection:



Public Member Functions

- virtual bool [toAzimuthal](#) (float fx, float fy, float &theta, float &phi) const =0
Convert from plane coordinates to projection coords.
- virtual bool [toPlanar](#) (float theta, float phi, float &fx, float &fy) const =0
Convert from spherical planar coordinates to projection coords.
- virtual void [scaleDown](#) (float flightLength, float detX, float detY, float &scaledX, float &scaledY) const =0
Convert from actual detector postion (eg. mm) and flight path to scaled-down transform.
- virtual void [scaleUp](#) (float flightLength, float scaledX, float scaledY, float &realX, float &realY) const =0
Convert from scaled-down coordinates to actual size.

6.46.1 Detailed Description

Definition at line 20 of file projection.h.

6.46.2 Member Function Documentation

6.46.2.1 scaleDown()

```
virtual void AtomProbe::SphericPlaneProjection::scaleDown (
    float flightLength,
    float detX,
    float detY,
    float & scaledX,
    float & scaledY ) const [pure virtual]
```

Convert from actual detector postion (eg. mm) and flight path to scaled-down transform.

Implemented in [AtomProbe::ModifiedFocusSphericProjection](#), [AtomProbe::StereographicProjection](#), and [AtomProbe::GnomonicProjection](#).

Referenced by [AtomProbe::ReconstructionSphereOnCone::reconstruct\(\)](#).

6.46.2.2 scaleUp()

```
virtual void AtomProbe::SphericPlaneProjection::scaleUp (
    float flightLength,
    float scaledX,
    float scaledY,
    float & realX,
    float & realY ) const [pure virtual]
```

Convert from scaled-down coordinates to actual size.

Implemented in [AtomProbe::ModifiedFocusSphericProjection](#), [AtomProbe::StereographicProjection](#), and [AtomProbe::GnomonicProjection](#).

6.46.2.3 toAzimuthal()

```
virtual bool AtomProbe::SphericPlaneProjection::toAzimuthal (
    float fx,
    float fy,
    float & theta,
    float & phi ) const [pure virtual]
```

Convert from plane coordinates to projection coords.

- Returns false if transform is not possible

Implemented in [AtomProbe::ModifiedFocusSphericProjection](#), [AtomProbe::StereographicProjection](#), and [AtomProbe::GnomonicProjection](#).

Referenced by [AtomProbe::ReconstructionSphereOnCone::reconstruct\(\)](#).

6.46.2.4 toPlanar()

```
virtual bool AtomProbe::SphericPlaneProjection::toPlanar (
    float theta,
    float phi,
    float & fx,
    float & fy ) const [pure virtual]
```

Convert from spherical planar coordinates to projection coords.

Implemented in [AtomProbe::ModifiedFocusSphericProjection](#), [AtomProbe::StereographicProjection](#), and [AtomProbe::GnomonicProjection](#).

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

- include/atomprobe/projection/[projection.h](#)

6.47 AtomProbe::SphericProjectionParams Struct Reference

Public Attributes

- double [theta](#)
- double [focusDist](#)

6.47.1 Detailed Description

Definition at line 19 of file projection.cpp.

6.47.2 Member Data Documentation

6.47.2.1 focusDist

```
double AtomProbe::SphericProjectionParams::focusDist
```

Definition at line 22 of file projection.cpp.

Referenced by [AtomProbe::ModifiedFocusSphericProjection::etaToTheta\(\)](#), [AtomProbe::ModifiedFocusSphericProjection::getFOVRadius\(\)](#), [AtomProbe::ModifiedFocusSphericProjection::getMaxFOV\(\)](#), [AtomProbe::ModifiedFocusSphericProjection::scaleDown\(\)](#), [AtomProbe::ModifiedFocusSphericProjection::scaleUp\(\)](#), [AtomProbe::ModifiedFocusSphericProjection::setFocus\(\)](#), [AtomProbe::SphericProjectionEqn\(\)](#), [AtomProbe::StereographicProjection::thetaToEta\(\)](#), [AtomProbe::ModifiedFocusSphericProjection::thetaToEta\(\)](#), and [AtomProbe::ModifiedFocusSphericProjection::toAzimuthal\(\)](#).

6.47.2.2 theta

```
double AtomProbe::SphericProjectionParams::theta
```

Definition at line 21 of file projection.cpp.

Referenced by [AtomProbe::ModifiedFocusSphericProjection::getFOVRadius\(\)](#), [AtomProbe::ModifiedFocusSphericProjection::scaleUp\(\)](#), [AtomProbe::SphericProjectionEqn\(\)](#), [AtomProbe::StereographicProjection::thetaToEta\(\)](#), and [AtomProbe::ModifiedFocusSphericProjection::thetaToEta\(\)](#).

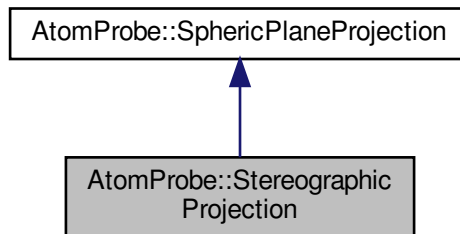
The documentation for this struct was generated from the following file:

- [src/projection/projection.cpp](#)

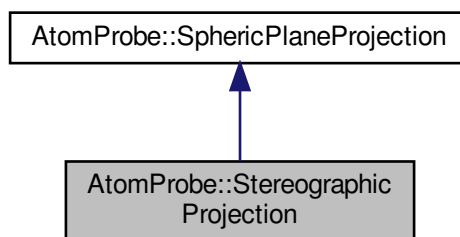
6.48 AtomProbe::StereographicProjection Class Reference

```
#include <projection.h>
```

Inheritance diagram for AtomProbe::StereographicProjection:



Collaboration diagram for AtomProbe::StereographicProjection:



Public Member Functions

- virtual bool [toAzimuthal](#) (float fx, float fy, float &theta, float &phi) const
Convert from plane coordinates to stereographic (NOT SPHERICAL) coords.
- virtual bool [toPlanar](#) (float theta, float phi, float &fx, float &fy) const
Convert from stereographic (NOT SPHERICAL) coordinates to plane.
- virtual void [scaleDown](#) (float flightLength, float detX, float detY, float &scaledX, float &scaledY) const
Convert from actual detector position (eg. mm) and flight path to scaled-down transform.
- virtual void [scaleUp](#) (float flightLength, float scaledX, float scaledY, float &realX, float &realY) const
Convert from scaled-down radius to actual radius.

Static Public Member Functions

- static float [thetaToEta](#) (float theta)
Convert azimuthal angle to spherical. See etaToTheta for description.

6.48.1 Detailed Description

Definition at line 68 of file projection.h.

6.48.2 Member Function Documentation

6.48.2.1 scaleDown()

```
void AtomProbe::StereographicProjection::scaleDown (
    float flightLength,
    float detX,
    float detY,
    float & scaledX,
    float & scaledY ) const [virtual]
```

Convert from actual detector position (eg. mm) and flight path to scaled-down transform.

Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 119 of file projection.cpp.

References ASSERT.

6.48.2.2 scaleUp()

```
void AtomProbe::StereographicProjection::scaleUp (
    float flightLength,
    float scaledX,
    float scaledY,
    float & realX,
    float & realY ) const [virtual]
```

Convert from scaled-down radius to actual radius.

Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 130 of file projection.cpp.

References ASSERT.

6.48.2.3 thetaToEta()

```
float AtomProbe::StereographicProjection::thetaToEta (
    float theta ) [static]
```

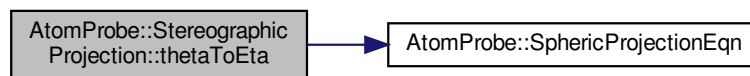
Convert azimuthal angle to spherical. See etaToTheta for description.

-input range is limited to [0,90)

Definition at line 158 of file projection.cpp.

References [AtomProbe::SphericProjectionParams::focusDist](#), [M_PI](#), [AtomProbe::SphericProjectionEqn\(\)](#), and [AtomProbe::SphericProjectionParams::theta](#).

Here is the call graph for this function:



6.48.2.4 toAzimuthal()

```
bool AtomProbe::StereographicProjection::toAzimuthal (
    float fx,
    float fy,
    float & theta,
    float & phi ) const [virtual]
```

Convert from plane coordinates to stereographic (NOT SPHERICAL) coords.

Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 141 of file projection.cpp.

6.48.2.5 toPlanar()

```
bool AtomProbe::StereographicProjection::toPlanar (
    float theta,
    float phi,
    float & fx,
    float & fy ) const [virtual]
```

Convert from stereographic (NOT SPHERICAL) coordinates to plane.

Implements [AtomProbe::SphericPlaneProjection](#).

Definition at line 107 of file projection.cpp.

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

- include/atomprobe/projection/[projection.h](#)
- src/projection/[projection.cpp](#)

6.49 AtomProbe::TETRAHEDRON Class Reference

```
#include <mesh.h>
```

Public Attributes

- unsigned int [p](#) [4]
- unsigned int [physGroup](#)

6.49.1 Detailed Description

Definition at line 51 of file mesh.h.

6.49.2 Member Data Documentation

6.49.2.1 p

```
unsigned int AtomProbe::TETRAHEDRON::p[4]
```

Definition at line 53 of file mesh.h.

Referenced by [AtomProbe::Mesh::loadGmshMesh\(\)](#).

6.49.2.2 physGroup

```
unsigned int AtomProbe::TETRAHEDRON::physGroup
```

Definition at line 54 of file mesh.h.

Referenced by [AtomProbe::Mesh::loadGmshMesh\(\)](#).

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

- [include/atomprobe/primitives/mesh.h](#)

6.50 AtomProbe::THREEDAP_DATA Struct Reference

experimental setup data

```
#include <dataFiles.h>
```

Public Attributes

- float [flightPath](#)
The flight path to the virtual (or real) projected tip image.
- float [alpha](#)
Pulse coupling coefficient (1 of 2)
- float [beta](#)
Pulse coupling coefficient (2 of 2)
- float [tZero](#)
Time offset for pulse, in nanoseconds.
- float [detectorRadius](#)
The size of the detector, in mm.
- unsigned int [detectorChannels](#)
The number of channels on the detector.

6.50.1 Detailed Description

experimental setup data

Definition at line 222 of file dataFiles.h.

6.50.2 Member Data Documentation

6.50.2.1 alpha

```
float AtomProbe::THREEDAP_DATA::alpha
```

Pulse coupling coefficient (1 of 2)

Definition at line 227 of file dataFiles.h.

Referenced by `AtomProbe::readPosapOps()`.

6.50.2.2 beta

```
float AtomProbe::THREEDAP_DATA::beta
```

Pulse coupling coefficient (2 of 2)

Definition at line 229 of file dataFiles.h.

Referenced by `AtomProbe::readPosapOps()`.

6.50.2.3 detectorChannels

```
unsigned int AtomProbe::THREEDAP_DATA::detectorChannels
```

The number of channels on the detector.

Definition at line 235 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

6.50.2.4 detectorRadius

```
float AtomProbe::THREEDAP_DATA::detectorRadius
```

The size of the detector, in mm.

Definition at line 233 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

6.50.2.5 flightPath

```
float AtomProbe::THREEDAP_DATA::flightPath
```

The flight path to the virtual (or real) projected tip image.

Definition at line 225 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

6.50.2.6 tZero

```
float AtomProbe::THREEDAP_DATA::tZero
```

Time offset for pulse, in nanoseconds.

Definition at line 231 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

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

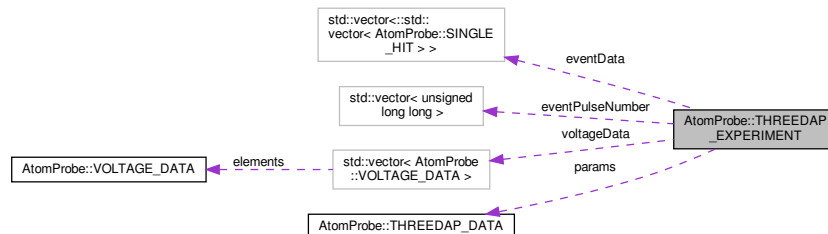
- [include/atomprobe/io/dataFiles.h](#)

6.51 AtomProbe::THREEDAP_EXPERIMENT Struct Reference

Data structure that contains the experiment information present in a 3Dap file.

```
#include <dataFiles.h>
```

Collaboration diagram for AtomProbe::THREEDAP_EXPERIMENT:



Public Attributes

- [THREEDAP_DATA params](#)
Experiment parameters.
- `std::vector<::std::vector< SINGLE_HIT > >` [eventData](#)
vector of event vectors. Each interior event vector describes a group of hits on one pulse
- `std::vector< VOLTAGE_DATA >` [voltageData](#)
Voltage information.
- `std::vector< unsigned long long >` [eventPulseNumber](#)
The pulse number associated with an event group.

6.51.1 Detailed Description

Data structure that contains the experiment information present in a 3Dap file.

Definition at line 239 of file dataFiles.h.

6.51.2 Member Data Documentation

6.51.2.1 eventData

```
std::vector<::std::vector<SINGLE\_HIT> > AtomProbe::THREEDAP_EXPERIMENT::eventData
```

vector of event vectors. Each interior event vector describes a group of hits on one pulse

Definition at line 244 of file dataFiles.h.

Referenced by `AtomProbe::readPosapOps()`.

6.51.2.2 eventPulseNumber

```
std::vector<unsigned long long> AtomProbe::THREEDAP_EXPERIMENT::eventPulseNumber
```

The pulse number associated with an event group.

Definition at line 248 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

6.51.2.3 params

```
THREEDAP_DATA AtomProbe::THREEDAP_EXPERIMENT::params
```

Experiment parameters.

Definition at line 242 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

6.51.2.4 voltageData

```
std::vector<VOLTAGE_DATA> AtomProbe::THREEDAP_EXPERIMENT::voltageData
```

Voltage information.

Definition at line 246 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

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

- [include/atomprobe/io/dataFiles.h](#)

6.52 AtomProbe::TRIANGLE Class Reference

```
#include <mesh.h>
```

Public Member Functions

- bool [isSane](#) (size_t pLimit=(size_t) -1) const
- bool [edgesMismatch](#) (const TRIANGLE &tOther) const

Public Attributes

- unsigned int [p](#) [3]
- unsigned int [physGroup](#)

6.52.1 Detailed Description

Definition at line 57 of file mesh.h.

6.52.2 Member Function Documentation

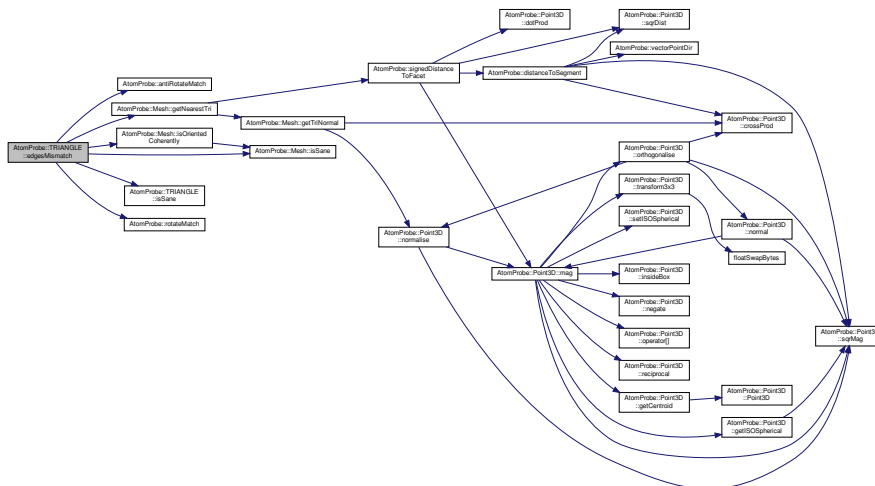
6.52.2.1 edgesMismatch()

```
bool AtomProbe::TRIANGLE::edgesMismatch (
    const TRIANGLE & tOther ) const
```

Definition at line 2201 of file mesh.cpp.

References [AtomProbe::antiRotateMatch\(\)](#), [ASSERT](#), [AtomProbe::Mesh::getNearestTri\(\)](#), [AtomProbe::Mesh::isOrientedCoherently\(\)](#), [isSane\(\)](#), [AtomProbe::Mesh::isSane\(\)](#), [AtomProbe::Mesh::nodes](#), [p](#), [AtomProbe::rotateMatch\(\)](#), [TEST](#), and [AtomProbe::Mesh::triangles](#).

Here is the call graph for this function:



6.52.2.2 isSane()

```
bool AtomProbe::TRIANGLE::isSane (
    size_t pLimit = (size_t)-1 ) const
```

Definition at line 2185 of file mesh.cpp.

Referenced by `edgesMismatch()`.

6.52.3 Member Data Documentation

6.52.3.1 p

```
unsigned int AtomProbe::TRIANGLE::p[3]
```

Definition at line 59 of file mesh.h.

Referenced by `edgesMismatch()`, `AtomProbe::Mesh::getVolume()`, `AtomProbe::Mesh::loadGmshMesh()`, `AtomProbe::marchingCubes()`, `AtomProbe::Mesh::print()`, and `AtomProbe::Mesh::setTriangleMesh()`.

6.52.3.2 physGroup

```
unsigned int AtomProbe::TRIANGLE::physGroup
```

Definition at line 60 of file mesh.h.

Referenced by `AtomProbe::Mesh::loadGmshMesh()`, and `AtomProbe::marchingCubes()`.

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

- [include/atomprobe/primitives/mesh.h](#)
- [src/primitives/mesh.cpp](#)

6.53 AtomProbe::TriangleWithIndexedVertices Struct Reference

```
#include <isoSurface.h>
```

Public Attributes

- `size_t p [3]`

6.53.1 Detailed Description

Definition at line 40 of file isoSurface.h.

6.53.2 Member Data Documentation

6.53.2.1 p

```
size_t AtomProbe::TriangleWithIndexedVertices::p[3]
```

Definition at line 42 of file isoSurface.h.

Referenced by AtomProbe::marchingCubes().

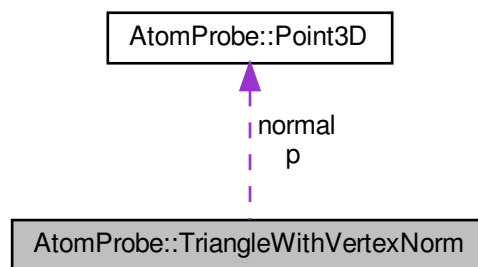
The documentation for this struct was generated from the following file:

- include/atomprobe/algorithm/isoSurface.h

6.54 AtomProbe::TriangleWithVertexNorm Class Reference

```
#include <isoSurface.h>
```

Collaboration diagram for AtomProbe::TriangleWithVertexNorm:



Public Member Functions

- void [getCentroid](#) (Point3D &p) const
- void [computeACWNormal](#) (Point3D &p) const
- void [safeComputeACWNormal](#) (Point3D &p) const
- float [computeArea](#) () const
- bool [isDegenerate](#) () const

Public Attributes

- [Point3D](#) p [3]
- [Point3D](#) normal [3]

6.54.1 Detailed Description

Definition at line 27 of file isoSurface.h.

6.54.2 Member Function Documentation

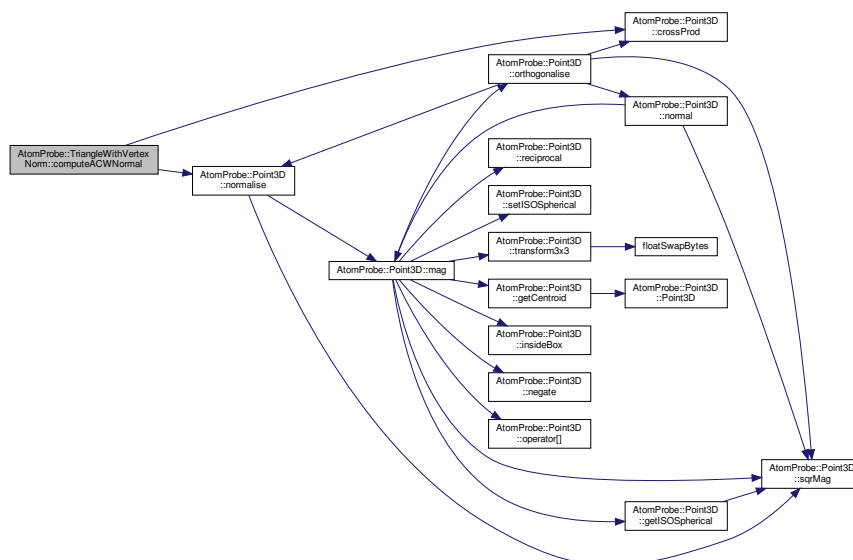
6.54.2.1 computeACWNormal()

```
void AtomProbe::TriangleWithVertexNorm::computeACWNormal (
    Point3D & p ) const
```

Definition at line 89 of file isoSurface.cpp.

References `AtomProbe::Point3D::crossProd()`, `AtomProbe::Point3D::normalise()`, and `p`.

Here is the call graph for this function:



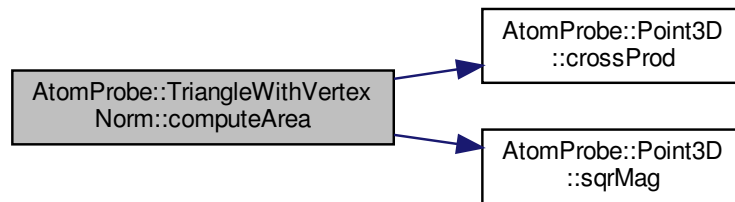
6.54.2.2 computeArea()

```
float AtomProbe::TriangleWithVertexNorm::computeArea ( ) const
```

Definition at line 117 of file isoSurface.cpp.

References `AtomProbe::Point3D::crossProd()`, `p`, and `AtomProbe::Point3D::sqrMag()`.

Here is the call graph for this function:



6.54.2.3 getCentroid()

```
void AtomProbe::TriangleWithVertexNorm::getCentroid (
    Point3D & p ) const
```

Definition at line 135 of file `isoSurface.cpp`.

References `p`.

6.54.2.4 isDegenerate()

```
bool AtomProbe::TriangleWithVertexNorm::isDegenerate ( ) const
```

Definition at line 128 of file `isoSurface.cpp`.

References `p`.

Referenced by `AtomProbe::marchingCubes()`.

6.54.2.5 safeComputeACWNormal()

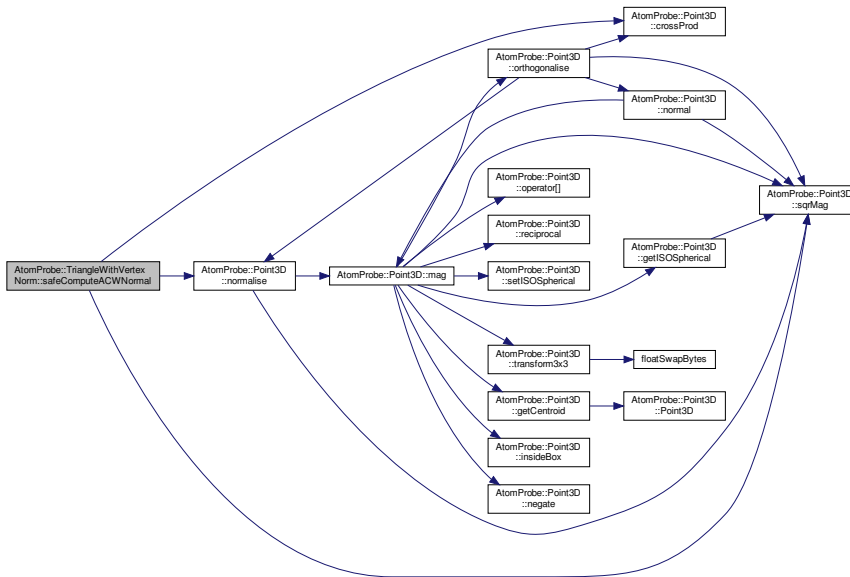
```
void AtomProbe::TriangleWithVertexNorm::safeComputeACWNormal (
    Point3D & p ) const
```

Definition at line 100 of file isoSurface.cpp.

References `AtomProbe::Point3D::crossProd()`, `AtomProbe::Point3D::normalise()`, `p`, and `AtomProbe::Point3D::sqrMag()`.

Referenced by `AtomProbe::marchingCubes()`.

Here is the call graph for this function:



6.54.3 Member Data Documentation

6.54.3.1 normal

`Point3D` `AtomProbe::TriangleWithVertexNorm::normal` [3]

Definition at line 31 of file isoSurface.h.

Referenced by `AtomProbe::marchingCubes()`.

6.54.3.2 p

`Point3D` AtomProbe::TriangleWithVertexNorm::p[3]

Definition at line 30 of file isoSurface.h.

Referenced by `computeACWNNormal()`, `computeArea()`, `getCentroid()`, `isDegenerate()`, `AtomProbe::marchingCubes()`, and `safeComputeACWNNormal()`.

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

- `include/atomprobe/algorithm/isoSurface.h`
- `src/algorithm/isoSurface.cpp`

6.55 AtomProbe::VOLTAGE_DATA Struct Reference

Voltage data structure – these updates occur periodically during the experiment.

```
#include <dataFiles.h>
```

Public Attributes

- `size_t` `nextHitGroupOffset`
FIXME: Document me.
- `float` `voltage`
Standing Voltage applied to specimen (V)
- `float` `pulseVolt`
Pulse voltage applied to specimen (V)
- `float` `beta`
Coupling coefficient to specimen.

6.55.1 Detailed Description

Voltage data structure – these updates occur periodically during the experiment.

Definition at line 206 of file dataFiles.h.

6.55.2 Member Data Documentation

6.55.2.1 beta

```
float AtomProbe::VOLTAGE_DATA::beta
```

Coupling coefficient to specimen.

Definition at line 216 of file dataFiles.h.

Referenced by `AtomProbe::readPosapOps()`.

6.55.2.2 nextHitGroupOffset

```
size_t AtomProbe::VOLTAGE_DATA::nextHitGroupOffset
```

FIXME: Document me.

Definition at line 209 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

6.55.2.3 pulseVolt

```
float AtomProbe::VOLTAGE_DATA::pulseVolt
```

Pulse voltage applied to specimen (V)

Definition at line 213 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

6.55.2.4 voltage

```
float AtomProbe::VOLTAGE_DATA::voltage
```

Standing Voltage applied to specimen (V)

Definition at line 211 of file dataFiles.h.

Referenced by AtomProbe::readPosapOps().

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

- [include/atomprobe/io/dataFiles.h](#)

6.56 AtomProbe::Voxels< T > Class Template Reference

Template class that stores 3D voxel data.

```
#include <voxels.h>
```


Public Member Functions

- [Voxels](#) ()
Constructor.
- [~Voxels](#) ()
Destructor.
- void [swap](#) ([Voxels](#)< T > &v)
Swap object contents with other voxel object.
- void [copy](#) ([Voxels](#)< T > &newCopy) const
Clone data into another object.
- void [setPoint](#) (const [AtomProbe::Point3D](#) &pt, const T &val)
Set the value of a point in the dataset.
- T [getPointData](#) (const [AtomProbe::Point3D](#) &pt) const
Retrieve the value of a datapoint, this is rounded to the nearest voxel.
- void [getIndex](#) (size_t &x, size_t &y, size_t &z, const [AtomProbe::Point3D](#) &p) const
Retrieve the XYZ voxel location associated with a given position.
- void [getIndexWithUpper](#) (size_t &x, size_t &y, size_t &z, const [AtomProbe::Point3D](#) &p) const
Retrieve the XYZ voxel location associated with a given position,.
- [AtomProbe::Point3D](#) [getPoint](#) (size_t x, size_t y, size_t z) const
Get the position associated with an XYZ voxel.
- T [getData](#) (size_t x, size_t y, size_t z) const
Retrieve the value of a specific voxel.
- T [getData](#) (size_t i) const
Retrieve value of the nth voxel.
- size_t [getOffset](#) (size_t x, size_t y, size_t z) const
Convert xyz to offset.
- void [setEntry](#) (size_t n, const T &val)
- void [setData](#) (size_t x, size_t y, size_t z, const T &val)
Retrieve a reference to the data ata given position.
- void [setData](#) (size_t n, const T &val)
Set the value of nth point in the dataset.
- void [getInterpolatedData](#) (const [AtomProbe::Point3D](#) &pt, T &v) const
- void [getInterpSlice](#) (size_t normal, float offset, T *p, size_t interpMode=[VOX_INTERP_NONE](#)) const
- void [getSlice](#) (size_t normal, size_t offset, T *p) const
- void [getSize](#) (size_t &x, size_t &y, size_t &z) const
Get the size of the data field.
- size_t [resize](#) (size_t newX, size_t newY, size_t newZ, const [AtomProbe::Point3D](#) &newMinBound=[AtomProbe::Point3D](#)(0.0f, 0.0f, 0.0f), const [AtomProbe::Point3D](#) &newMaxBound=[AtomProbe::Point3D](#)(1.0f, 1.0f, 1.0f))
Resize the data field.
- size_t [resize](#) (const [Voxels](#)< T > &v)
- size_t [resizeKeepData](#) (size_t newX, size_t newY, size_t newZ, unsigned int direction=[CLIP_LOWER_SOUTH_WEST](#), const [AtomProbe::Point3D](#) &newMinBound=[AtomProbe::Point3D](#)(0.0f, 0.0f, 0.0f), const [AtomProbe::Point3D](#) &newMaxBound=[AtomProbe::Point3D](#)(1.0f, 1.0f, 1.0f), const T &fill=T(0), bool doFill=false)
Resize the data field, maintaining data as best as possible.
- size_t [deprecatedGetEdgeUniqueIndex](#) (size_t x, size_t y, size_t z, unsigned int edge) const
DEPRECATED FUNCTION : Get a unique integer that corresponds to the edge index for the voxel; where edges are shared between voxels.
- size_t [getCellUniqueEdgeIndex](#) (size_t x, size_t y, size_t z, unsigned int edge) const
Get a unique integer that corresponds to an edge index for the voxel; where edges are shared between voxels.
- void [getEdgeEnds](#) (size_t edgeIndex, [AtomProbe::Point3D](#) &a, [AtomProbe::Point3D](#) &b) const
Convert the edge index (as generated by [getEdgeUniqueIndex](#)) into a centre position.

- void [getEdgeCell](#) (size_t edgeUniqId, size_t &x, size_t &y, size_t &z, size_t &axis) const
Convert edge index (only as generated by getCellUniqueEdgeIndex) into a cell & axis value.
- void [getEdgeEndApproxVals](#) (size_t edgeUniqId, T &a, T &b) const
Return the values that are associated with the edge ends, as returned by getEdgeEnds.
- void [rebin](#) (Voxels< T > &dest, size_t rate, size_t clipMode=[CLIP_NONE](#)) const
Rebin the data by a given rate.
- T [getSum](#) (const T &initialVal=T(0.0)) const
Get the total value of the data field.
- size_t [count](#) (const T &minIntensity) const
count the number of cells with at least this intensity
- void [fill](#) (const T &val)
Fill all voxels with a given value.
- [AtomProbe::Point3D getMinBounds](#) () const
Get the bounding box vertex (min/max)
- [AtomProbe::Point3D getMaxBounds](#) () const
- void [getAxisBounds](#) (size_t axis, float &minV, float &maxV) const
- [AtomProbe::Point3D getPitch](#) () const
! Get the spacing for a unit cell
- void [setBounds](#) (const [AtomProbe::Point3D](#) &pMin, const [AtomProbe::Point3D](#) &pMax)
Set the bounding size.
- void [setBounds](#) (const [BoundCube](#) &b)
- void [getBounds](#) ([AtomProbe::Point3D](#) &pMin, [AtomProbe::Point3D](#) &pMax) const
Get the bounding size.
- void [getBounds](#) ([BoundCube](#) &bc) const
- size_t [init](#) (size_t nX, size_t nY, size_t nZ, const [BoundCube](#) &bound)
Initialise the voxel storage.
- size_t [init](#) (size_t nX, size_t nY, size_t nZ)
Initialise the voxel storage.
- size_t [loadFile](#) (const char *cpFilename, size_t nX, size_t nY, size_t nZ, bool silent=false)
Load the voxels from file.
- size_t [writeFile](#) (const char *cpFilename) const
Write the voxel objects in column major written out to file.
- size_t [convolve](#) (const [Voxels](#)< T > &templateVec, [Voxels](#)< T > &result, size_t boundMode=[BOUND_CLIP](#)) const
Run convolution over this data, placing the correlation data into "result".
- size_t [separableConvolve](#) (const [Voxels](#)< T > &templateVec, [Voxels](#)< T > &result, size_t boundMode=[BOUND_CLIP](#)) const
Similar to convolve, but faster – only works with separable kernels.
- void [threshold](#) (const T &thresh, bool keepUpper, const T &newVal)
Threshold the voxels, keeping either the lower or upper values above this threshold.
- void [thresholdToBoolMask](#) (const T &thresh, bool keepUpper, [Voxels](#)< bool > &result) const
Generate a boolean voxel field stating whether the data is above or below the threshold specified.
- void [applyMask](#) (const [Voxels](#)< bool > &mask, const T &newVal, bool invert=false)
Apply a mask to this dataset, where the mask is true (if not inverting), replace with specified val (newVal)
- void [thresholdForPosition](#) (std::vector< [AtomProbe::Point3D](#) > &p, const T &thresh, bool lowerEq=false) const
Find the positions of the voxels that are above or below a given value.
- void [binarise](#) ([Voxels](#)< T > &result, const T &thresh, const T &onThresh, const T &offThresh) const
Binarise the data into a result vector.
- void [clear](#) ()
Empty the data.

- T `min` () const
Find minimum in dataset.
- T `max` () const
Find maximum in dataset.
- void `minMax` (T &`min`, T &`max`) const
Find both min and max in dataset in the same loop.
- int `countPoints` (const std::vector< `AtomProbe::Point3D` > &points, bool noWrap=true, bool doErase=false)
Generate a dataset that consists of the counts of points in a given voxel.
- T `trapezIntegral` () const
Integrate the dataset via the trapezoidal method.
- void `calculateDensity` ()
Convert voxel intensity into voxel density.
- float `getBinVolume` () const
- void `operator/=` (const `Voxels`< T > &v)
Element wise division.
- void `operator/=` (const T &v)
- bool `operator==` (const `Voxels`< T > &v) const
- size_t `size` () const

Static Public Member Functions

- static size_t `sizeofType` ()
Return the sizeof value for the T type.

6.56.1 Detailed Description

```
template<class T>
class AtomProbe::Voxels< T >
```

Template class that stores 3D voxel data.

To instantiate this class, objects must have basic mathematical operators, such as * + - and =

Definition at line 104 of file voxels.h.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 Voxels()

```
template<class T >
AtomProbe::Voxels< T >::Voxels ( )
```

Constructor.

Definition at line 407 of file voxels.h.

6.56.2.2 ~Voxels()

```
template<class T >
AtomProbe::Voxels< T >::~~Voxels ( )
```

Destructor.

Definition at line 412 of file voxels.h.

6.56.3 Member Function Documentation

6.56.3.1 applyMask()

```
template<class T >
void AtomProbe::Voxels< T >::applyMask (
    const Voxels< bool > & mask,
    const T & newVal,
    bool invert = false )
```

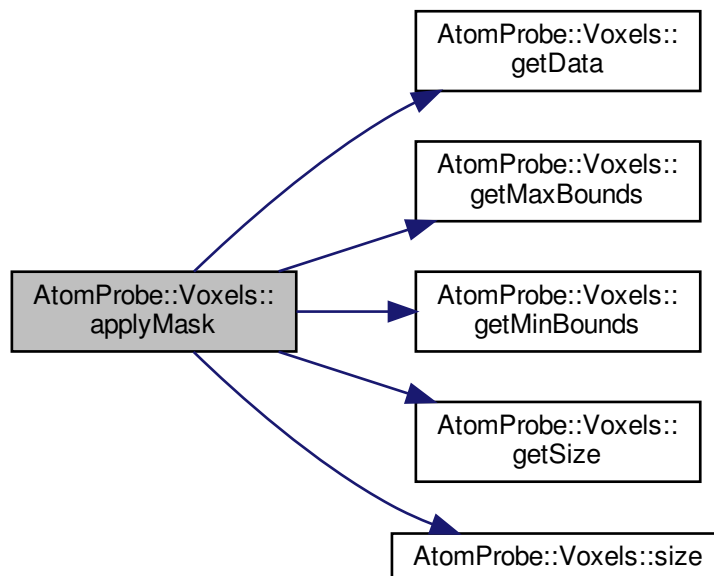
Apply a mask to this dataset, where the mask is true (if not inverting), replace with specified val (newVal)

Definition at line 1335 of file voxels.h.

References ASSERT, AtomProbe::Voxels< T >::getData(), AtomProbe::Voxels< T >::getMaxBounds(), AtomProbe::Voxels< T >::getMinBounds(), AtomProbe::Voxels< T >::getSize(), and AtomProbe::Voxels< T >::size().

Referenced by AtomProbe::Voxels< T >::getBounds().

Here is the call graph for this function:



6.56.3.2 binarise()

```
template<class T >
void AtomProbe::Voxels< T >::binarise (
    Voxels< T > & result,
    const T & thresh,
    const T & onThresh,
    const T & offThresh ) const
```

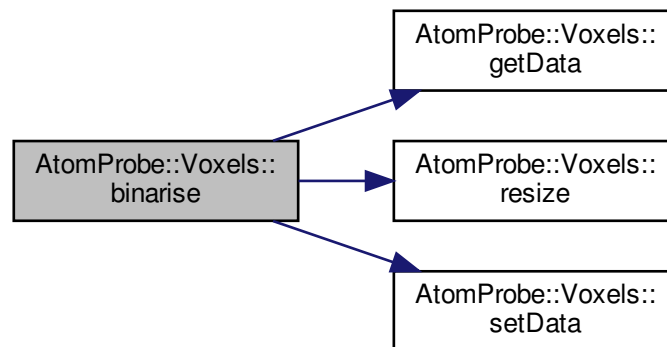
Binarise the data into a result vector.

Definition at line 1368 of file voxels.h.

References AtomProbe::Voxels< T >::getData(), AtomProbe::Voxels< T >::resize(), and AtomProbe::Voxels< T >::setData().

Referenced by AtomProbe::Voxels< T >::sizeofType().

Here is the call graph for this function:



6.56.3.3 calculateDensity()

```
template<class T >
void AtomProbe::Voxels< T >::calculateDensity ( )
```

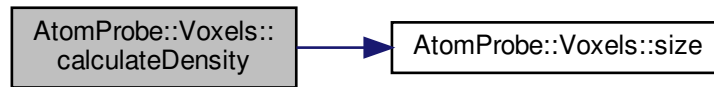
Convert voxel intensity into voxel density.

Definition at line 1458 of file voxels.h.

References AtomProbe::Voxels< T >::size().

Referenced by AtomProbe::Voxels< T >::clear().

Here is the call graph for this function:



6.56.3.4 clear()

```

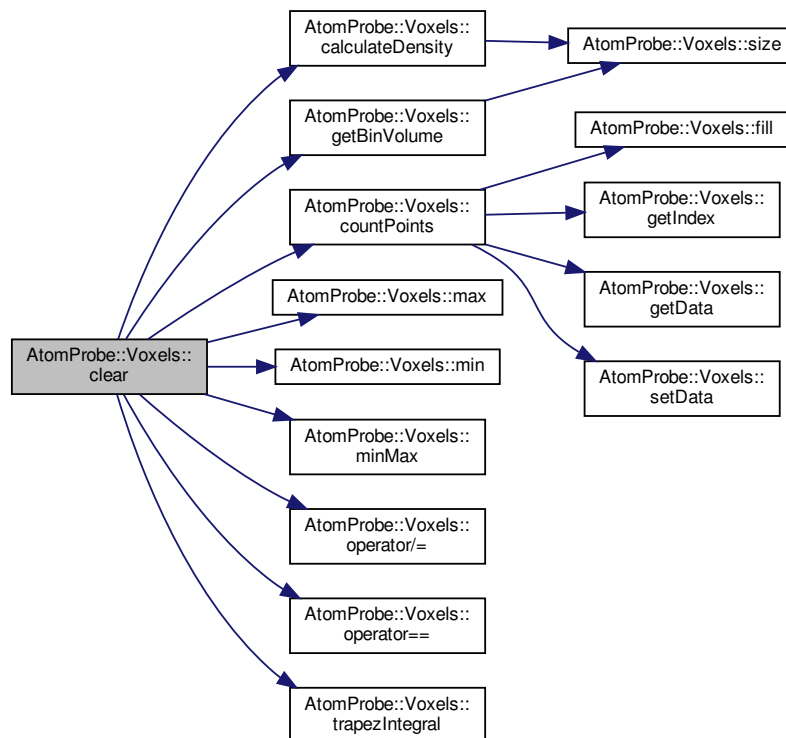
template<class T>
void AtomProbe::Voxels< T >::clear ( ) [inline]
  
```

Empty the data.

Definition at line 341 of file voxels.h.

References `AtomProbe::Voxels< T >::calculateDensity()`, `AtomProbe::Voxels< T >::countPoints()`, `AtomProbe::Voxels< T >::getBinVolume()`, `AtomProbe::Voxels< T >::max()`, `AtomProbe::Voxels< T >::min()`, `AtomProbe::Voxels< T >::minMax()`, `AtomProbe::Voxels< T >::operator/=()`, `AtomProbe::Voxels< T >::operator==()`, and `AtomProbe::Voxels< T >::trapezIntegral()`.

Here is the call graph for this function:



6.56.3.5 convolve()

```
template<class T>
size_t AtomProbe::Voxels< T >::convolve (
    const Voxels< T > & templateVec,
    Voxels< T > & result,
    size_t boundMode = BOUND_CLIP ) const
```

Run convolution over this data, placing the correlation data into "result".

Datasets MUST have the same pitch (spacing) for the result to be defined template type must have a T(0.0) constructor that initialises it to some "zero"

Referenced by AtomProbe::Voxels< T >::getBounds().

6.56.3.6 copy()

```
template<class T >
void AtomProbe::Voxels< T >::copy (
    Voxels< T > & newCopy ) const
```

Clone data into another object.

Definition at line 424 of file voxels.h.

6.56.3.7 count()

```
template<class T >
size_t AtomProbe::Voxels< T >::count (
    const T & minIntensity ) const
```

count the number of cells with at least this intensity

Definition at line 1194 of file voxels.h.

Referenced by AtomProbe::countDataDistanceWeight(), and AtomProbe::Voxels< T >::setEntry().

6.56.3.8 countPoints()

```
template<class T >
int AtomProbe::Voxels< T >::countPoints (
    const std::vector< AtomProbe::Point3D > & points,
    bool nowrap = true,
    bool doErase = false )
```

Generate a dataset that consists of the counts of points in a given voxel.

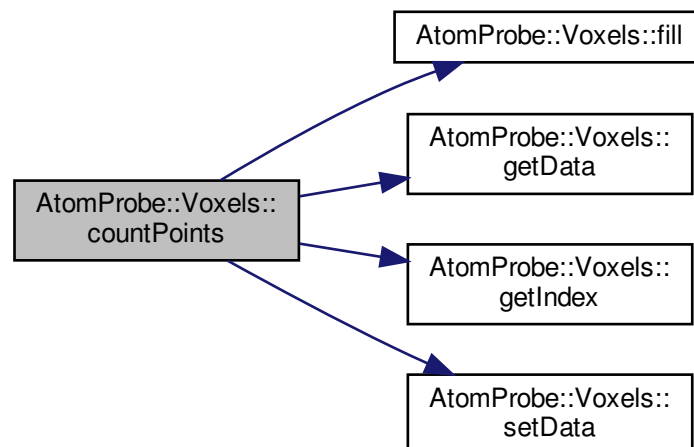
Ensure that the voxel scaling factors are set by calling `::setBounds()` with the appropriate parameters for your data. Disabling `nowrap` allows you to "saturate" your data field in the case of dense regions, rather than let wrap-around errors occur

Definition at line 1420 of file `voxels.h`.

References `AtomProbe::Voxels< T >::fill()`, `AtomProbe::Voxels< T >::getData()`, `AtomProbe::Voxels< T >::get←Index()`, `AtomProbe::Voxels< T >::setData()`, and `AtomProbe::VOXEL_ABORT_ERR`.

Referenced by `AtomProbe::Voxels< T >::clear()`.

Here is the call graph for this function:



6.56.3.9 deprecatedGetEdgeUniqueIndex()

```
template<class T >
size_t AtomProbe::Voxels< T >::deprecatedGetEdgeUniqueIndex (
    size_t x,
    size_t y,
    size_t z,
    unsigned int edge ) const
```


DEPRECATED FUNCTION : Get a unique integer that corresponds to the edge index for the voxel; where edges are shared between voxels.

Each voxel has 12 edges. These are shared (except voxels that on zero or positive boundary). Return a unique index that corresponds to a specified edge (0->11). Index *CANNOT* be inverted to yield cell

Definition at line 512 of file voxels.h.

Referenced by AtomProbe::marchingCubes(), and AtomProbe::Voxels< T >::setEntry().

6.56.3.10 fill()

```
template<class T >
void AtomProbe::Voxels< T >::fill (
    const T & val )
```

Fill all voxels with a given value.

Definition at line 1525 of file voxels.h.

Referenced by AtomProbe::countDataDistanceWeight(), AtomProbe::Voxels< T >::countPoints(), main(), AtomProbe::marchingCubes(), and AtomProbe::Voxels< T >::setEntry().

6.56.3.11 getAxisBounds()

```
template<class T >
void AtomProbe::Voxels< T >::getAxisBounds (
    size_t axis,
    float & minV,
    float & maxV ) const
```

Definition at line 843 of file voxels.h.

Referenced by AtomProbe::Voxels< T >::setEntry().

6.56.3.12 getBinVolume()

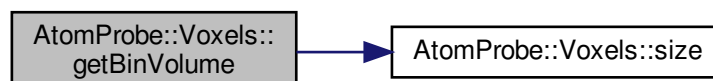
```
template<class T >
float AtomProbe::Voxels< T >::getBinVolume ( ) const
```

Definition at line 1474 of file voxels.h.

References AtomProbe::Voxels< T >::size().

Referenced by AtomProbe::Voxels< T >::clear().

Here is the call graph for this function:



6.56.3.13 `getBounds()` [1/2]

```
template<class T>
void AtomProbe::Voxels< T >::getBounds (
    AtomProbe::Point3D & pMin,
    AtomProbe::Point3D & pMax ) const [inline]
```

Get the bounding size.

Definition at line 260 of file voxels.h.

Referenced by `main()`, and `AtomProbe::marchingCubes()`.

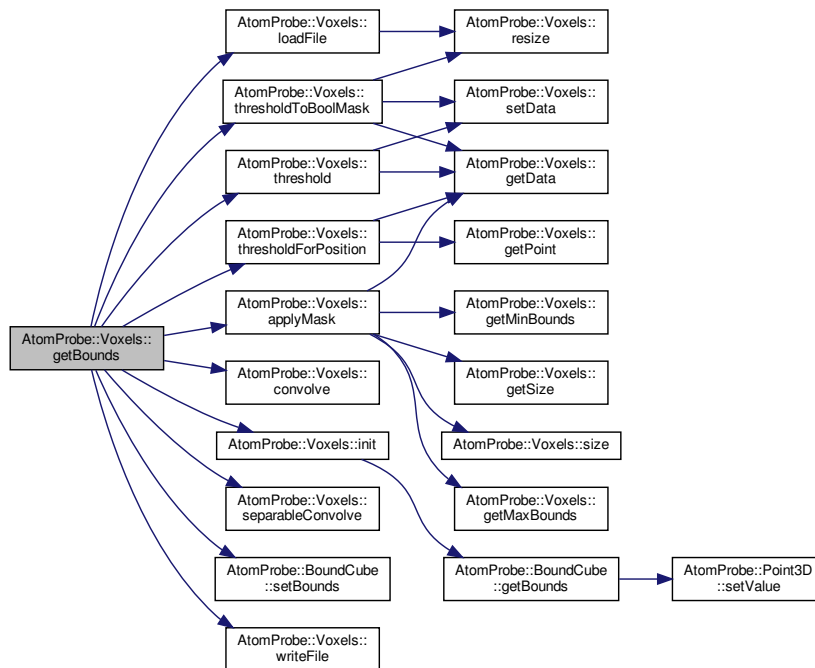
6.56.3.14 `getBounds()` [2/2]

```
template<class T>
void AtomProbe::Voxels< T >::getBounds (
    BoundCube & bc ) const [inline]
```

Definition at line 261 of file voxels.h.

References `AtomProbe::Voxels< T >::applyMask()`, `AtomProbe::BOUND_CLIP`, `AtomProbe::Voxels< T >::convolve()`, `AtomProbe::Voxels< T >::init()`, `AtomProbe::Voxels< T >::loadFile()`, `AtomProbe::Voxels< T >::separableConvolve()`, `AtomProbe::BoundCube::setBounds()`, `AtomProbe::Voxels< T >::threshold()`, `AtomProbe::Voxels< T >::thresholdForPosition()`, `AtomProbe::Voxels< T >::thresholdToBoolMask()`, and `AtomProbe::Voxels< T >::writeFile()`.

Here is the call graph for this function:



6.56.3.15 `getCellUniqueEdgeIndex()`

```
template<class T >
size_t AtomProbe::Voxels< T >::getCellUniqueEdgeIndex (
    size_t x,
    size_t y,
    size_t z,
    unsigned int edge ) const
```

Get a unique integer that corresponds to an edge index for the voxel; where edges are shared between voxels.

Each voxel has 12 edges. These are shared (except voxels that on zero or positive boundary). Return a NON-unique index that corresponds to a specified edge (0->11) Index can be inverted to yield cell

Definition at line 716 of file voxels.h.

Referenced by `AtomProbe::Voxels< T >::setEntry()`.

6.56.3.16 `getData()` [1/2]

```
template<class T >
T AtomProbe::Voxels< T >::getData (
    size_t x,
    size_t y,
    size_t z ) const [inline]
```

Retrieve the value of a specific voxel.

Definition at line 1224 of file voxels.h.

References ASSERT.

Referenced by `AtomProbe::Voxels< T >::applyMask()`, `AtomProbe::Voxels< T >::binarise()`, `AtomProbe::Voxels< T >::countPoints()`, `AtomProbe::Voxels< T >::getInterpolatedData()`, `AtomProbe::Voxels< T >::getSlice()`, `AtomProbe::incrementDataDistanceWeight()`, `AtomProbe::marchingCubes()`, `AtomProbe::Voxels< T >::resizeKeepData()`, `AtomProbe::sumVoxels()`, `AtomProbe::Voxels< T >::threshold()`, `AtomProbe::Voxels< T >::thresholdForPosition()`, and `AtomProbe::Voxels< T >::thresholdToBoolMask()`.

6.56.3.17 `getData()` [2/2]

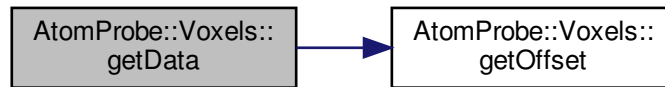
```
template<class T>
T AtomProbe::Voxels< T >::getData (
    size_t i ) const [inline]
```

Retrieve value of the nth voxel.

Definition at line 154 of file voxels.h.

References `AtomProbe::Voxels< T >::getOffset()`.

Here is the call graph for this function:



6.56.3.18 getEdgeCell()

```

template<class T >
void AtomProbe::Voxels< T >::getEdgeCell (
    size_t edgeUniqId,
    size_t & x,
    size_t & y,
    size_t & z,
    size_t & axis ) const
  
```

Convert edge index (only as generated by getCellUniqueEdgeIndex) into a cell & axis value.

Definition at line 766 of file voxels.h.

References ASSERT.

Referenced by AtomProbe::Voxels< T >::getEdgeEnds(), and AtomProbe::Voxels< T >::setEntry().

6.56.3.19 getEdgeEndApproxVals()

```

template<class T >
void AtomProbe::Voxels< T >::getEdgeEndApproxVals (
    size_t edgeUniqId,
    T & a,
    T & b ) const
  
```

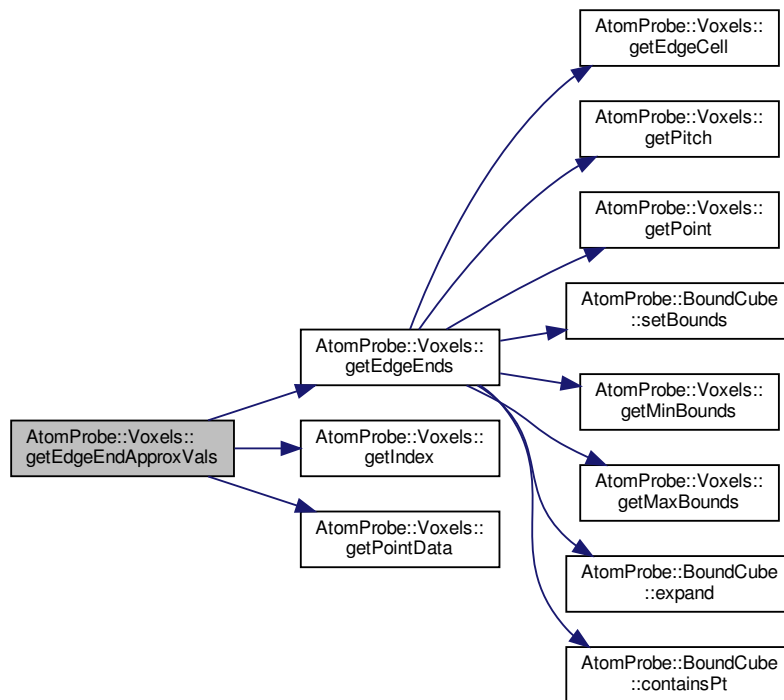
Return the values that are associated with the edge ends, as returned by getEdgeEnds.

Definition at line 828 of file voxels.h.

References AtomProbe::Voxels< T >::getEdgeEnds(), AtomProbe::Voxels< T >::getIndex(), and AtomProbe::Voxels< T >::getPointData().

Referenced by AtomProbe::marchingCubes(), and AtomProbe::Voxels< T >::setEntry().

Here is the call graph for this function:



6.56.3.20 getEdgeEnds()

```

template<class T >
void AtomProbe::Voxels< T >::getEdgeEnds (
    size_t edgeIndex,
    AtomProbe::Point3D & a,
    AtomProbe::Point3D & b ) const
  
```

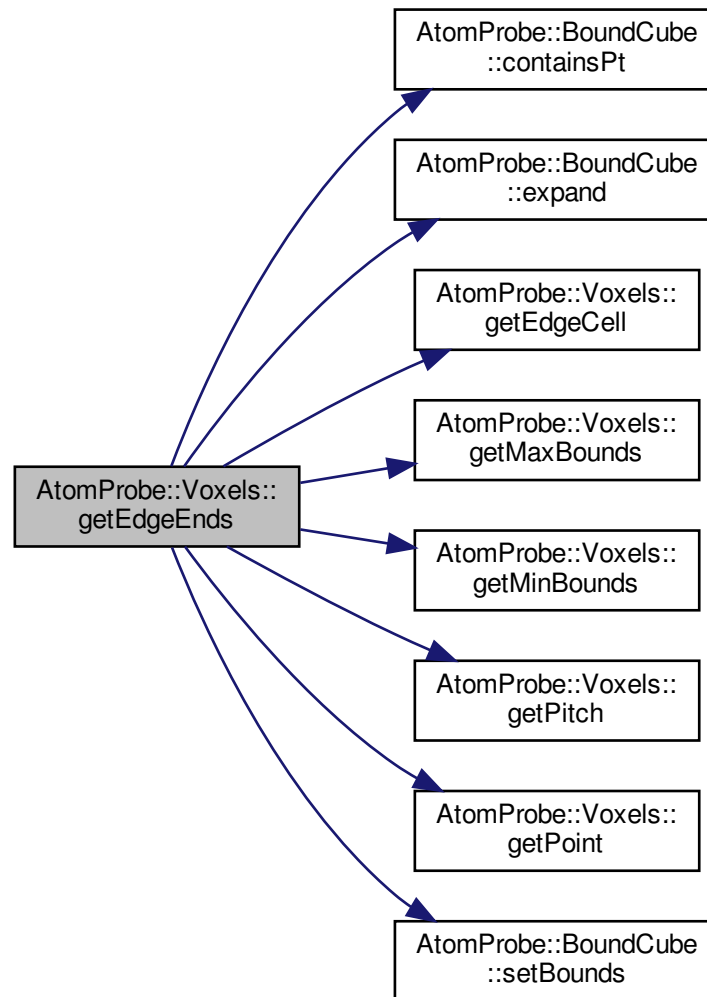
Convert the edge index (as generated by `getEdgeUniqueIndex`) into a centre position.

Definition at line 786 of file `voxels.h`.

References `ASSERT`, `AtomProbe::BoundCube::containsPt()`, `AtomProbe::BoundCube::expand()`, `AtomProbe::Voxels< T >::getEdgeCell()`, `AtomProbe::Voxels< T >::getMaxBounds()`, `AtomProbe::Voxels< T >::getMinBounds()`, `AtomProbe::Voxels< T >::getPitch()`, `AtomProbe::Voxels< T >::getPoint()`, and `AtomProbe::BoundCube::setBounds()`.

Referenced by `AtomProbe::Voxels< T >::getEdgeEndApproxVals()`, `AtomProbe::marchingCubes()`, and `AtomProbe::Voxels< T >::setEntry()`.

Here is the call graph for this function:



6.56.3.21 `getIndex()`

```

template<class T >
void AtomProbe::Voxels< T >::getIndex (
    size_t & x,
    size_t & y,
    size_t & z,
    const AtomProbe::Point3D & p ) const
  
```

Retrieve the XYZ voxel location associated with a given position.

Definition at line 1485 of file voxels.h.

References ASSERT.

Referenced by AtomProbe::Voxels< T >::countPoints(), AtomProbe::Voxels< T >::getEdgeEndApproxVals(), AtomProbe::Voxels< T >::getIndexWithUpper(), AtomProbe::Voxels< T >::getInterpolatedData(), and AtomProbe::incrementDataDistanceWeight().

6.56.3.22 getIndexWithUpper()

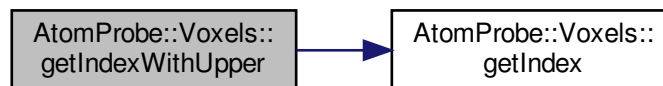
```
template<class T >
void AtomProbe::Voxels< T >::getIndexWithUpper (
    size_t & x,
    size_t & y,
    size_t & z,
    const AtomProbe::Point3D & p ) const
```

Retrieve the XYZ voxel location associated with a given position,.

Definition at line 1504 of file voxels.h.

References AtomProbe::Voxels< T >::getIndex().

Here is the call graph for this function:



6.56.3.23 getInterpolatedData()

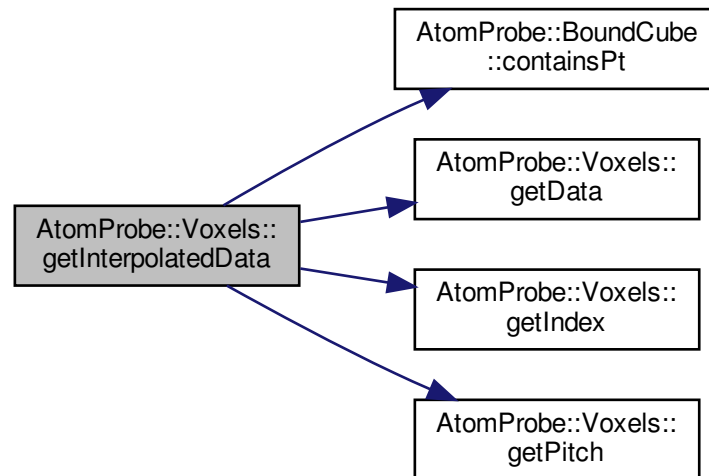
```
template<class T >
void AtomProbe::Voxels< T >::getInterpolatedData (
    const AtomProbe::Point3D & pt,
    T & v ) const
```

Definition at line 1667 of file voxels.h.

References ASSERT, AtomProbe::BoundCube::containsPt(), AtomProbe::Voxels< T >::getData(), AtomProbe::Voxels< T >::getIndex(), and AtomProbe::Voxels< T >::getPitch().

Referenced by AtomProbe::countDataDistanceWeight(), and AtomProbe::Voxels< T >::setEntry().

Here is the call graph for this function:



6.56.3.24 getInterpSlice()

```

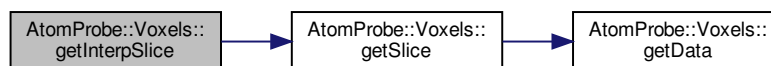
template<class T >
void AtomProbe::Voxels< T >::getInterpSlice (
    size_t normal,
    float offset,
    T * p,
    size_t interpMode = VOX_INTERP_NONE ) const
  
```

Definition at line 1603 of file voxels.h.

References `ASSERT`, `AtomProbe::Voxels< T >::getSlice()`, `AtomProbe::VOX_INTERP_LINEAR`, and `AtomProbe::VOX_INTERP_NONE`.

Referenced by `AtomProbe::Voxels< T >::setEntry()`.

Here is the call graph for this function:



6.56.3.25 getMaxBounds()

```
template<class T >
AtomProbe::Point3D AtomProbe::Voxels< T >::getMaxBounds ( ) const
```

Definition at line 990 of file voxels.h.

References ASSERT.

Referenced by AtomProbe::Voxels< T >::applyMask(), AtomProbe::Voxels< T >::getEdgeEnds(), AtomProbe::marchingCubes(), and AtomProbe::Voxels< T >::setEntry().

6.56.3.26 getMinBounds()

```
template<class T >
AtomProbe::Point3D AtomProbe::Voxels< T >::getMinBounds ( ) const
```

Get the bounding box vertex (min/max)

Definition at line 983 of file voxels.h.

References ASSERT.

Referenced by AtomProbe::Voxels< T >::applyMask(), AtomProbe::Voxels< T >::getEdgeEnds(), AtomProbe::marchingCubes(), and AtomProbe::Voxels< T >::setEntry().

6.56.3.27 getOffset()

```
template<class T>
size_t AtomProbe::Voxels< T >::getOffset (
    size_t x,
    size_t y,
    size_t z ) const
```

Convert xyz to offset.

Referenced by AtomProbe::Voxels< T >::getData().

6.56.3.28 getPitch()

```
template<class T >
AtomProbe::Point3D AtomProbe::Voxels< T >::getPitch ( ) const
```

! Get the spacing for a unit cell

Definition at line 495 of file voxels.h.

Referenced by AtomProbe::Voxels< T >::getEdgeEnds(), AtomProbe::Voxels< T >::getInterpolatedData(), AtomProbe::marchingCubes(), and AtomProbe::Voxels< T >::setEntry().

6.56.3.29 `getPoint()`

```
template<class T >
AtomProbe::Point3D AtomProbe::Voxels< T >::getPoint (
    size_t x,
    size_t y,
    size_t z ) const
```

Get the position associated with an XYZ voxel.

Definition at line 485 of file voxels.h.

Referenced by `AtomProbe::Voxels< T >::getEdgeEnds()`, `AtomProbe::incrementDataDistanceWeight()`, `main()`, and `AtomProbe::Voxels< T >::thresholdForPosition()`.

6.56.3.30 `getPointData()`

```
template<class T >
T AtomProbe::Voxels< T >::getPointData (
    const AtomProbe::Point3D & pt ) const
```

Retrieve the value of a datapoint, this is rounded to the nearest voxel.

Definition at line 469 of file voxels.h.

References ASSERT.

Referenced by `AtomProbe::Voxels< T >::getEdgeEndApproxVals()`.

6.56.3.31 `getSize()`

```
template<class T >
void AtomProbe::Voxels< T >::getSize (
    size_t & x,
    size_t & y,
    size_t & z ) const
```

Get the size of the data field.

Definition at line 503 of file voxels.h.

References ASSERT.

Referenced by `AtomProbe::Voxels< T >::applyMask()`, `AtomProbe::countDataDistanceWeight()`, `AtomProbe::incrementDataDistanceWeight()`, `AtomProbe::marchingCubes()`, `AtomProbe::Voxels< T >::setEntry()`, and `AtomProbe::sumVoxels()`.

6.56.3.32 `getSlice()`

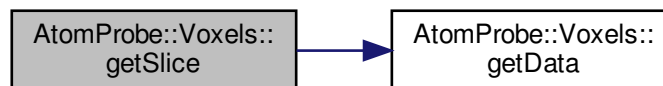
```
template<class T >
void AtomProbe::Voxels< T >::getSlice (
    size_t normal,
    size_t offset,
    T * p ) const
```

Definition at line 1534 of file voxels.h.

References ASSERT, and AtomProbe::Voxels< T >::getData().

Referenced by AtomProbe::Voxels< T >::getInterpSlice(), and AtomProbe::Voxels< T >::setEntry().

Here is the call graph for this function:

6.56.3.33 `getSum()`

```
template<class T >
T AtomProbe::Voxels< T >::getSum (
    const T & initialVal = T(0.0) ) const
```

Get the total value of the data field.

An "initial value" is provided to provide the definition of a zero value

Definition at line 1155 of file voxels.h.

References ASSERT.

Referenced by AtomProbe::countDataDistanceWeight(), and AtomProbe::Voxels< T >::setEntry().

6.56.3.34 `init()` [1/2]

```
template<class T >
size_t AtomProbe::Voxels< T >::init (
    size_t nX,
    size_t nY,
    size_t nZ,
    const BoundCube & bound )
```

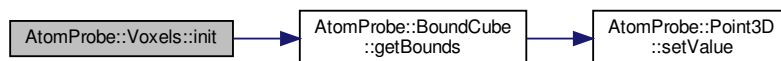
Initialise the voxel storage.

Definition at line 1018 of file voxels.h.

References `AtomProbe::BoundCube::getBounds()`.

Referenced by `AtomProbe::Voxels< T >::getBounds()`, and `AtomProbe::Voxels< T >::init()`.

Here is the call graph for this function:

6.56.3.35 `init()` [2/2]

```
template<class T >
size_t AtomProbe::Voxels< T >::init (
    size_t nX,
    size_t nY,
    size_t nZ )
```

Initialise the voxel storage.

Definition at line 1035 of file voxels.h.

References `AtomProbe::Voxels< T >::init()`.

Here is the call graph for this function:



6.56.3.36 loadFile()

```
template<class T >
size_t AtomProbe::Voxels< T >::loadFile (
    const char * cpFilename,
    size_t nX,
    size_t nY,
    size_t nZ,
    bool silent = false )
```

Load the voxels from file.

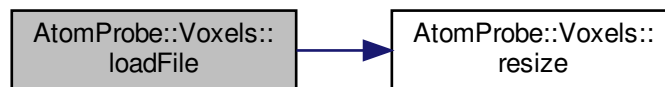
Format is flat size_ts in column major return codes: 1: File open error 2: Data size error

Definition at line 1044 of file voxels.h.

References AtomProbe::Voxels< T >::resize(), AtomProbe::VOXELS_BAD_FILE_OPEN, AtomProbe::VOXELS_BAD_FILE_READ, and AtomProbe::VOXELS_BAD_FILE_SIZE.

Referenced by AtomProbe::Voxels< T >::getBounds().

Here is the call graph for this function:



6.56.3.37 max()

```
template<class T >
T AtomProbe::Voxels< T >::max ( ) const
```

Find maximum in dataset.

Definition at line 1402 of file voxels.h.

References ASSERT.

Referenced by AtomProbe::Voxels< T >::clear(), and AtomProbe::countDataDistanceWeight().

6.56.3.38 min()

```
template<class T >
T AtomProbe::Voxels< T >::min ( ) const
```

Find minimum in dataset.

Definition at line 1395 of file voxels.h.

References ASSERT.

Referenced by AtomProbe::Voxels< T >::clear().

6.56.3.39 minMax()

```
template<class T >
void AtomProbe::Voxels< T >::minMax (
    T & min,
    T & max ) const
```

Find both min and max in dataset in the same loop.

Definition at line 1410 of file voxels.h.

References ASSERT.

Referenced by AtomProbe::Voxels< T >::clear().

6.56.3.40 operator/=() [1/2]

```
template<class T >
void AtomProbe::Voxels< T >::operator/= (
    const Voxels< T > & v )
```

Element wise division.

Definition at line 1725 of file voxels.h.

References ASSERT.

Referenced by AtomProbe::Voxels< T >::clear().

6.56.3.41 operator/=() [2/2]

```
template<class T >
void AtomProbe::Voxels< T >::operator/= (
    const T & v )
```

Definition at line 1745 of file voxels.h.

References ASSERT.

6.56.3.42 operator==()

```
template<class T >
bool AtomProbe::Voxels< T >::operator== (
    const Voxels< T > & v ) const
```

Definition at line 1764 of file voxels.h.

Referenced by AtomProbe::Voxels< T >::clear().

6.56.3.43 rebin()

```
template<class T>
void AtomProbe::Voxels< T >::rebin (
    Voxels< T > & dest,
    size_t rate,
    size_t clipMode = CLIP_NONE ) const
```

Rebin the data by a given rate.

This will perform a quick and dirty rebin operation, where groups of datablocks are binned into a single cell. Number of blocks binned is rate³. Field must be larger than rate in all directions. Currently only CLIP_NONE is supported.

Referenced by AtomProbe::Voxels< T >::setEntry().

6.56.3.44 resize() [1/2]

```
template<class T >
size_t AtomProbe::Voxels< T >::resize (
    size_t newX,
    size_t newY,
    size_t newZ,
    const AtomProbe::Point3D & newMinBound = AtomProbe::Point3D(0.0f,0.0f,0.0f),
    const AtomProbe::Point3D & newMaxBound = AtomProbe::Point3D(1.0f,1.0f,1.0f) )
```

Resize the data field.

This will destroy any data that was already in place. If the data needs to be preserved use "resizeKeepData". Data will *not* be initialised.

Definition at line 850 of file voxels.h.

Referenced by AtomProbe::Voxels< T >::binarise(), AtomProbe::countDataDistanceWeight(), AtomProbe::Voxels< T >::loadFile(), main(), AtomProbe::marchingCubes(), AtomProbe::Voxels< T >::resize(), AtomProbe::Voxels< T >::resizeKeepData(), AtomProbe::Voxels< T >::setEntry(), and AtomProbe::Voxels< T >::thresholdToBoolMask().

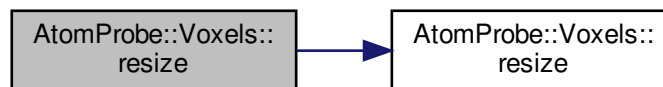
6.56.3.45 `resize()` [2/2]

```
template<class T >
size_t AtomProbe::Voxels< T >::resize (
    const Voxels< T > & v )
```

Definition at line 873 of file voxels.h.

References `AtomProbe::Voxels< T >::resize()`.

Here is the call graph for this function:



6.56.3.46 `resizeKeepData()`

```
template<class T >
size_t AtomProbe::Voxels< T >::resizeKeepData (
    size_t newX,
    size_t newY,
    size_t newZ,
    unsigned int direction = CLIP_LOWER_SOUTH_WEST,
    const AtomProbe::Point3D & newMinBound = AtomProbe::Point3D(0.0f, 0.0f, 0.0f),
    const AtomProbe::Point3D & newMaxBound = AtomProbe::Point3D(1.0f, 1.0f, 1.0f),
    const T & fill = T(0),
    bool doFill = false )
```

Resize the data field, maintaining data as best as possible.

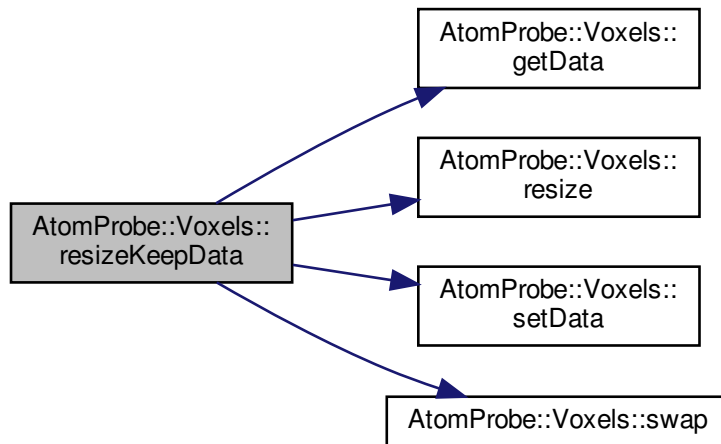
This will preserve data by resizing as much as possible about the origin. If the bounds are extended, the "fill" value is used by default iff `doFill` is set to true.

Definition at line 879 of file voxels.h.

References `ASSERT`, `AtomProbe::CLIP_LOWER_SOUTH_WEST`, `AtomProbe::Voxels< T >::getData()`, `AtomProbe::Voxels< T >::resize()`, `AtomProbe::Voxels< T >::setData()`, `AtomProbe::Voxels< T >::swap()`, and `AtomProbe::VOXEL_ABORT_ERR`.

Referenced by `AtomProbe::countDataDistanceWeight()`, and `AtomProbe::Voxels< T >::setEntry()`.

Here is the call graph for this function:



6.56.3.47 separableConvolve()

```

template<class T>
size_t AtomProbe::Voxels< T >::separableConvolve (
    const Voxels< T > & templateVec,
    Voxels< T > & result,
    size_t boundMode = BOUND_CLIP )
  
```

Similar to `convolve`, but faster – only works with separable kernels.

Datasets MUST have the same pitch (spacing) for the result to be defined template type must have a `T(0.0)` constructor that initialises it to some "zero"

Referenced by `AtomProbe::Voxels< T >::getBounds()`.

6.56.3.48 setBounds() [1/2]

```

template<class T >
void AtomProbe::Voxels< T >::setBounds (
    const AtomProbe::Point3D & pMin,
    const AtomProbe::Point3D & pMax )
  
```

Set the bounding size.

Definition at line 997 of file `voxels.h`.

References ASSERT.

Referenced by `AtomProbe::countDataDistanceWeight()`, `main()`, `AtomProbe::Voxels< T >::setBounds()`, and `AtomProbe::Voxels< T >::setEntry()`.

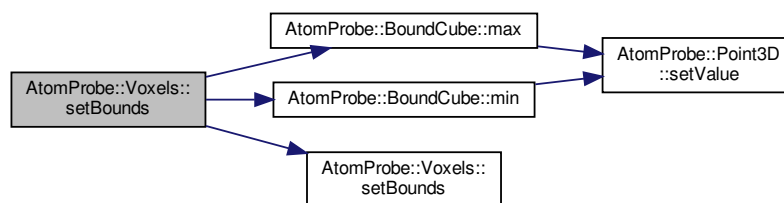
6.56.3.49 `setBounds()` [2/2]

```
template<class T >
void AtomProbe::Voxels< T >::setBounds (
    const BoundCube & b )
```

Definition at line 1012 of file voxels.h.

References `AtomProbe::BoundCube::max()`, `AtomProbe::BoundCube::min()`, and `AtomProbe::Voxels< T >::setBounds()`.

Here is the call graph for this function:

6.56.3.50 `setData()` [1/2]

```
template<class T >
void AtomProbe::Voxels< T >::setData (
    size_t x,
    size_t y,
    size_t z,
    const T & val )
```

Retrieve a reference to the data at a given position.

Set the value of a point in the dataset

Definition at line 450 of file voxels.h.

References ASSERT.

Referenced by `AtomProbe::Voxels< T >::binarise()`, `AtomProbe::countDataDistanceWeight()`, `AtomProbe::Voxels< T >::countPoints()`, `AtomProbe::incrementDataDistanceWeight()`, `main()`, `AtomProbe::marchingCubes()`, `AtomProbe::Voxels< T >::resizeKeepData()`, `AtomProbe::Voxels< T >::setEntry()`, `AtomProbe::Voxels< T >::threshold()`, and `AtomProbe::Voxels< T >::thresholdToBoolMask()`.

6.56.3.51 setData() [2/2]

```
template<class T >
void AtomProbe::Voxels< T >::setData (
    size_t n,
    const T & val ) [inline]
```

Set the value of nth point in the dataset.

Definition at line 460 of file voxels.h.

References ASSERT.

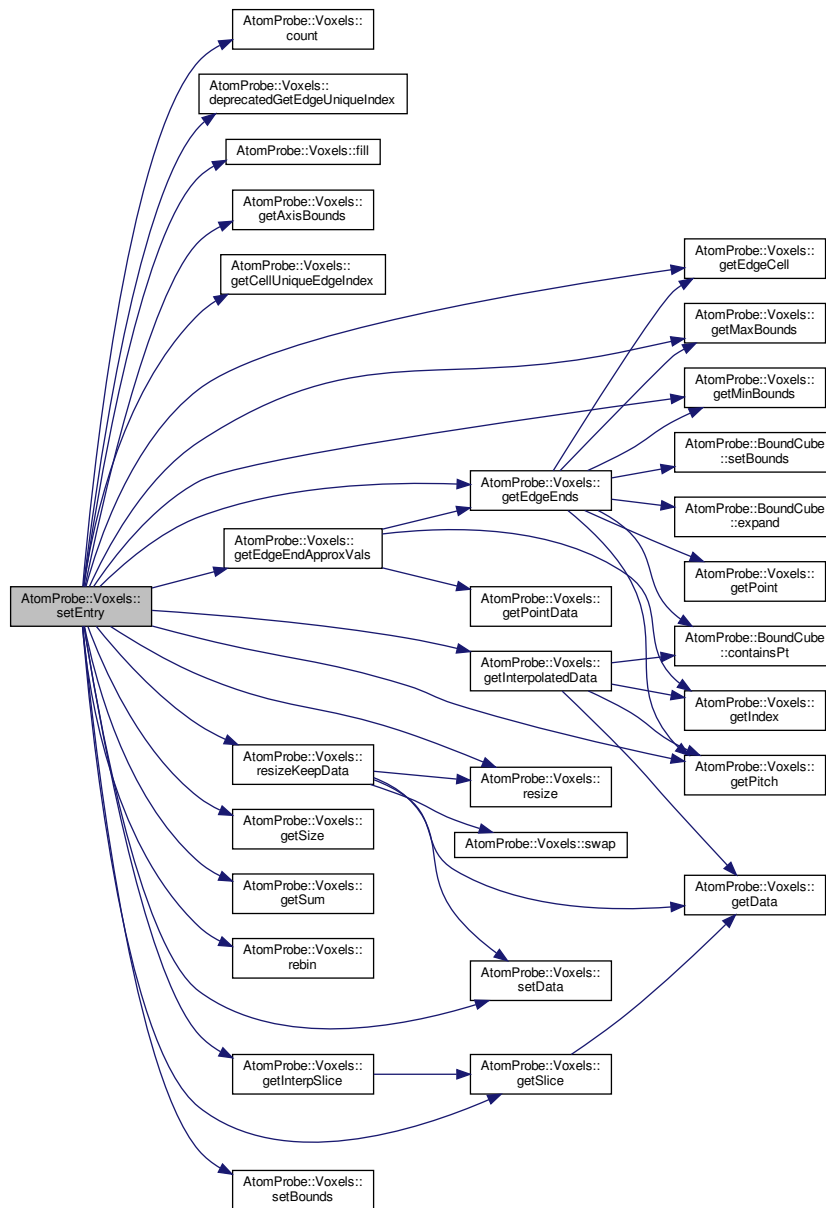
6.56.3.52 setEntry()

```
template<class T>
void AtomProbe::Voxels< T >::setEntry (
    size_t n,
    const T & val ) [inline]
```

Definition at line 159 of file voxels.h.

References AtomProbe::CLIP_LOWER_SOUTH_WEST, AtomProbe::CLIP_NONE, AtomProbe::Voxels< T >::count(), AtomProbe::Voxels< T >::deprecatedGetEdgeUniqueIndex(), AtomProbe::Voxels< T >::fill(), AtomProbe::Voxels< T >::getAxisBounds(), AtomProbe::Voxels< T >::getCellUniqueEdgeIndex(), AtomProbe::Voxels< T >::getEdgeCell(), AtomProbe::Voxels< T >::getEdgeEndApproxVals(), AtomProbe::Voxels< T >::getEdgeEnds(), AtomProbe::Voxels< T >::getInterpolatedData(), AtomProbe::Voxels< T >::getInterpSlice(), AtomProbe::Voxels< T >::getMaxBounds(), AtomProbe::Voxels< T >::getMinBounds(), AtomProbe::Voxels< T >::getPitch(), AtomProbe::Voxels< T >::getSize(), AtomProbe::Voxels< T >::getSlice(), AtomProbe::Voxels< T >::getSum(), AtomProbe::Voxels< T >::rebin(), AtomProbe::Voxels< T >::resize(), AtomProbe::Voxels< T >::resizeKeepData(), AtomProbe::Voxels< T >::setBounds(), AtomProbe::Voxels< T >::setData(), and AtomProbe::VOX_INTERP_NONE.

Here is the call graph for this function:



6.56.3.53 setPoint()

```

template<class T >
void AtomProbe::Voxels< T >::setPoint (
    const AtomProbe::Point3D & pt,
    const T & val )
  
```

Set the value of a point in the dataset.

Definition at line 438 of file voxels.h.

References ASSERT.

6.56.3.54 size()

```
template<class T>
size_t AtomProbe::Voxels< T >::size ( ) const [inline]
```

Definition at line 378 of file voxels.h.

Referenced by AtomProbe::Voxels< T >::applyMask(), AtomProbe::Voxels< T >::calculateDensity(), AtomProbe::Voxels< T >::getBinVolume(), and AtomProbe::sumVoxels().

6.56.3.55 sizeofType()

```
template<class T>
static size_t AtomProbe::Voxels< T >::sizeofType ( ) [inline], [static]
```

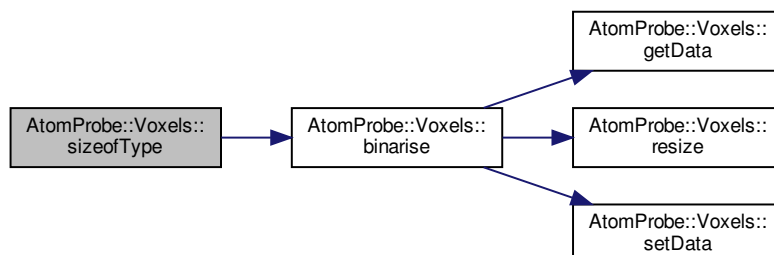
Return the sizeof value for the T type.

Maybe there is a better way to do this, I don't know

Definition at line 327 of file voxels.h.

References AtomProbe::Voxels< T >::binarise().

Here is the call graph for this function:



6.56.3.56 swap()

```
template<class T >
void AtomProbe::Voxels< T >::swap (
    Voxels< T > & v )
```

Swap object contents with other voxel object.

Definition at line 1211 of file voxels.h.

Referenced by AtomProbe::Voxels< T >::resizeKeepData().

6.56.3.57 threshold()

```
template<class T >
void AtomProbe::Voxels< T >::threshold (
    const T & thresh,
    bool keepUpper,
    const T & newVal )
```

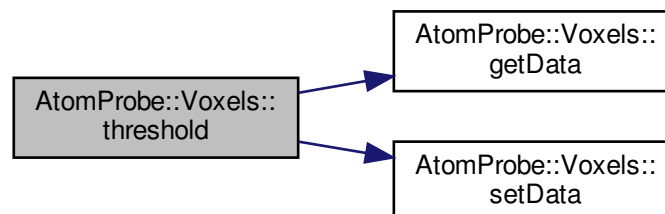
Threshold the voxels, keeping either the lower or upper values above this threshold.

Definition at line 1276 of file voxels.h.

References AtomProbe::Voxels< T >::getData(), and AtomProbe::Voxels< T >::setData().

Referenced by AtomProbe::Voxels< T >::getBounds().

Here is the call graph for this function:



6.56.3.58 thresholdForPosition()

```
template<class T >
void AtomProbe::Voxels< T >::thresholdForPosition (
    std::vector< AtomProbe::Point3D > & p,
    const T & thresh,
    bool lowerEq = false ) const
```

Find the positions of the voxels that are above or below a given value.

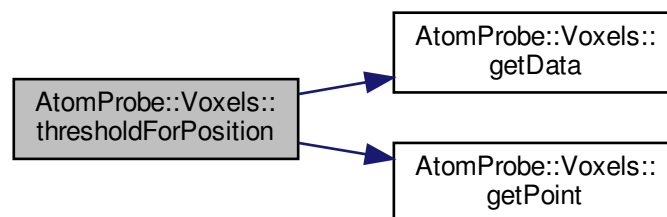
Returns the positions of the voxels' centroids for voxels that have, by default, a value greater than that of thresh. This behaviour can be reversed to "lesser than" by setting lowerEq to false

Definition at line 1231 of file voxels.h.

References AtomProbe::Voxels< T >::getData(), and AtomProbe::Voxels< T >::getPoint().

Referenced by AtomProbe::Voxels< T >::getBounds().

Here is the call graph for this function:



6.56.3.59 thresholdToBoolMask()

```
template<class T >
void AtomProbe::Voxels< T >::thresholdToBoolMask (
    const T & thresh,
    bool keepUpper,
    Voxels< bool > & result ) const
```

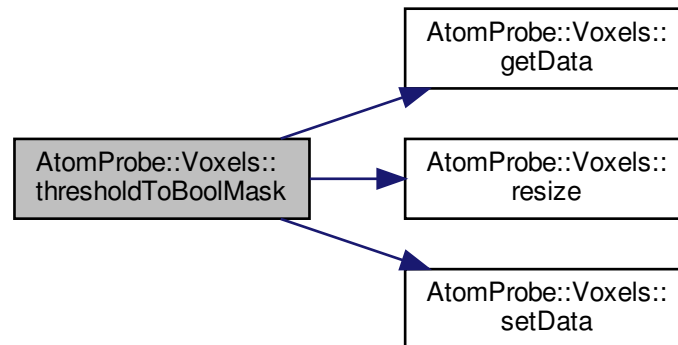
Generate a boolean voxel field stating whether the data is above or below the threshold specified.

Definition at line 1312 of file voxels.h.

References AtomProbe::Voxels< T >::getData(), AtomProbe::Voxels< T >::resize(), AtomProbe::Voxels< T >::setData(), and XOR.

Referenced by AtomProbe::Voxels< T >::getBounds().

Here is the call graph for this function:



6.56.3.60 trapezIntegral()

```
template<class T >
T AtomProbe::Voxels< T >::trapezIntegral ( ) const
```

Integrate the dataset via the trapezoidal method.

Definition at line 1168 of file voxels.h.

Referenced by `AtomProbe::Voxels< T >::clear()`.

6.56.3.61 writeFile()

```
template<class T >
size_t AtomProbe::Voxels< T >::writeFile (
    const char * cpFilename ) const
```

Write the voxel objects in column major written out to file.

Format is flat objects ("T"s) in column major format. Returns nonzero on failure

Definition at line 1131 of file voxels.h.

References ASSERT.

Referenced by `AtomProbe::Voxels< T >::getBounds()`.

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

- [include/atomprobe/helper/voxels.h](#)

6.57 AtomProbe::Weight Class Reference

Placeholder class for containing input weights for [MassTool](#).

```
#include <massTool.h>
```

Public Member Functions

- [Weight](#) ()
- [Weight](#) (float m, size_t uniqId=0)
- bool [operator==](#) (const [Weight](#) &b) const
- bool [operator<](#) (const [Weight](#) &b) const

Public Attributes

- float [mass](#)
- size_t [uniqueId](#)

6.57.1 Detailed Description

Placeholder class for containing input weights for [MassTool](#).

Definition at line 29 of file [massTool.h](#).

6.57.2 Constructor & Destructor Documentation

6.57.2.1 [Weight\(\)](#) [1/2]

```
AtomProbe::Weight::Weight ( ) [inline]
```

Definition at line 32 of file [massTool.h](#).

6.57.2.2 [Weight\(\)](#) [2/2]

```
AtomProbe::Weight::Weight (
    float m,
    size_t uniqId = 0 ) [inline]
```

Definition at line 33 of file [massTool.h](#).

References [mass](#), and [uniqueId](#).

6.57.3 Member Function Documentation

6.57.3.1 operator<()

```
bool AtomProbe::Weight::operator< (
    const Weight & b ) const [inline]
```

Definition at line 39 of file massTool.h.

References mass.

6.57.3.2 operator==(())

```
bool AtomProbe::Weight::operator==(
    const Weight & b ) const [inline]
```

Definition at line 38 of file massTool.h.

References mass, and uniqueId.

6.57.4 Member Data Documentation

6.57.4.1 mass

```
float AtomProbe::Weight::mass
```

Definition at line 35 of file massTool.h.

Referenced by operator<(), operator==(()), printSolution(), and Weight().

6.57.4.2 uniqueId

```
size_t AtomProbe::Weight::uniqueId
```

Definition at line 36 of file massTool.h.

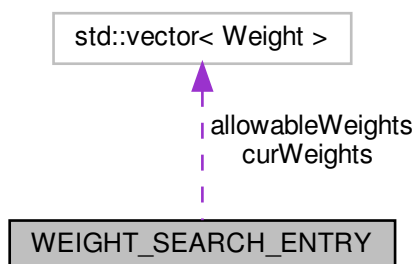
Referenced by operator==(()), printSolution(), and Weight().

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

- include/atomprobe/algorithm/massTool.h

6.58 WEIGHT_SEARCH_ENTRY Class Reference

Collaboration diagram for WEIGHT_SEARCH_ENTRY:



Public Attributes

- `vector< Weight > curWeights`
- `vector< Weight > allowableWeights`
- `size_t offset`
- `float cumulativeWeight`

6.58.1 Detailed Description

Definition at line 40 of file `massTool.cpp`.

6.58.2 Member Data Documentation

6.58.2.1 allowableWeights

```
vector<Weight> WEIGHT_SEARCH_ENTRY::allowableWeights
```

Definition at line 43 of file `massTool.cpp`.

Referenced by `AtomProbe::MassTool::bruteKnapsack()`.

6.58.2.2 cumulativeWeight

```
float WEIGHT_SEARCH_ENTRY::cumulativeWeight
```

Definition at line 45 of file massTool.cpp.

Referenced by AtomProbe::MassTool::bruteKnapsack().

6.58.2.3 curWeights

```
vector<Weight> WEIGHT_SEARCH_ENTRY::curWeights
```

Definition at line 43 of file massTool.cpp.

Referenced by AtomProbe::MassTool::bruteKnapsack().

6.58.2.4 offset

```
size_t WEIGHT_SEARCH_ENTRY::offset
```

Definition at line 44 of file massTool.cpp.

Referenced by AtomProbe::MassTool::bruteKnapsack().

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

- src/algorithm/[massTool.cpp](#)

6.59 AtomProbe::ZECH_ROOT Struct Reference

Public Attributes

- float [alpha](#)
- float [lambdaBack](#)
- float [observation](#)

6.59.1 Detailed Description

Definition at line 362 of file confidence.cpp.

6.59.2 Member Data Documentation

6.59.2.1 alpha

```
float AtomProbe::ZECH_ROOT::alpha
```

Definition at line 364 of file confidence.cpp.

Referenced by AtomProbe::zechConfidenceLimits(), and AtomProbe::zechRoot().

6.59.2.2 lambdaBack

```
float AtomProbe::ZECH_ROOT::lambdaBack
```

Definition at line 365 of file confidence.cpp.

Referenced by AtomProbe::zechConfidenceLimits(), and AtomProbe::zechRoot().

6.59.2.3 observation

```
float AtomProbe::ZECH_ROOT::observation
```

Definition at line 366 of file confidence.cpp.

Referenced by AtomProbe::zechConfidenceLimits(), and AtomProbe::zechRoot().

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

- [src/statistics/confidence.cpp](#)

Chapter 7

File Documentation

7.1 CMakeFiles/3.10.2/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- `#define COMPILER_ID ""`
- `#define STRINGIFY_HELPER(X) #X`
- `#define STRINGIFY(X) STRINGIFY_HELPER(X)`
- `#define PLATFORM_ID`
- `#define ARCHITECTURE_ID`
- `#define DEC(n)`
- `#define HEX(n)`
- `#define CXX_STD __cplusplus`

Functions

- `int main (int argc, char *argv[])`

Variables

- `char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"`
- `char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"`
- `char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"`
- `const char * info_language_dialect_default`

7.1.1 Macro Definition Documentation

7.1.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 453 of file CMakeCXXCompilerId.cpp.

7.1.1.2 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 273 of file CMakeCXXCompilerId.cpp.

7.1.1.3 CXX_STD

```
#define CXX_STD __cplusplus
```

Definition at line 536 of file CMakeCXXCompilerId.cpp.

7.1.1.4 DEC

```
#define DEC(  
    n )
```

Value:

```
('0' + ((n) / 10000000) % 10), \  
( '0' + ((n) / 1000000) % 10), \  
( '0' + ((n) / 100000) % 10), \  
( '0' + ((n) / 10000) % 10), \  
( '0' + ((n) / 1000) % 10), \  
( '0' + ((n) / 100) % 10), \  
( '0' + ((n) / 10) % 10), \  
( '0' + ((n) % 10)
```

Definition at line 457 of file CMakeCXXCompilerId.cpp.

7.1.1.5 HEX

```
#define HEX(  
    n )
```

Value:

```
('0' + ((n) >> 28 & 0xF)), \  
( '0' + ((n) >> 24 & 0xF)), \  
( '0' + ((n) >> 20 & 0xF)), \  
( '0' + ((n) >> 16 & 0xF)), \  
( '0' + ((n) >> 12 & 0xF)), \  
( '0' + ((n) >> 8 & 0xF)), \  
( '0' + ((n) >> 4 & 0xF)), \  
( '0' + ((n) & 0xF))
```

Definition at line 468 of file CMakeCXXCompilerId.cpp.

7.1.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 390 of file CMakeCXXCompilerId.cpp.

7.1.1.7 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY_HELPER(X)
```

Definition at line 294 of file CMakeCXXCompilerId.cpp.

7.1.1.8 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

Definition at line 293 of file CMakeCXXCompilerId.cpp.

7.1.2 Function Documentation

7.1.2.1 main()

```
int main (  
    int argc,  
    char * argv[] )
```

Definition at line 553 of file CMakeCXXCompilerId.cpp.

References `info_compiler`, `info_language_dialect_default`, and `info_platform`.

7.1.3 Variable Documentation

7.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 528 of file CMakeCXXCompilerId.cpp.

7.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 280 of file CMakeCXXCompilerId.cpp.

Referenced by main().

7.1.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
= "INFO" ":" "dialect_default["
```

```
"98"
```

```
"]"
```

Definition at line 539 of file CMakeCXXCompilerId.cpp.

Referenced by main().

7.1.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

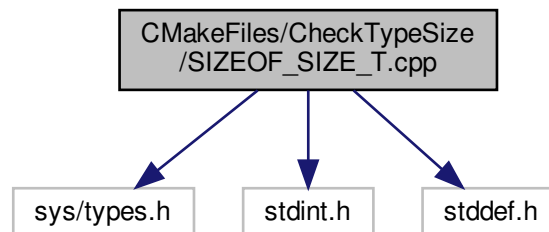
Definition at line 527 of file CMakeCXXCompilerId.cpp.

Referenced by main().

7.2 CMakeFiles/CheckTypeSize/SIZEOF_SIZE_T.cpp File Reference

```
#include <sys/types.h>  
#include <stdint.h>  
#include <stddef.h>
```

Include dependency graph for SIZEOF_SIZE_T.cpp:



Macros

- #define [SIZE](#) (sizeof(size_t))

Functions

- int [main](#) (int argc, char *argv[])

Variables

- char [info_size](#) []

7.2.1 Macro Definition Documentation

7.2.1.1 SIZE

```
#define SIZE (sizeof(size_t))
```

Definition at line 23 of file SIZEOF_SIZE_T.cpp.

7.2.2 Function Documentation

7.2.2.1 main()

```
int main (  
    int argc,  
    char * argv[] )
```

Definition at line 39 of file SIZEOF_SIZE_T.cpp.

References [info_size](#).

7.2.3 Variable Documentation

7.2.3.1 info_size

```
char info_size[]
```

Initial value:

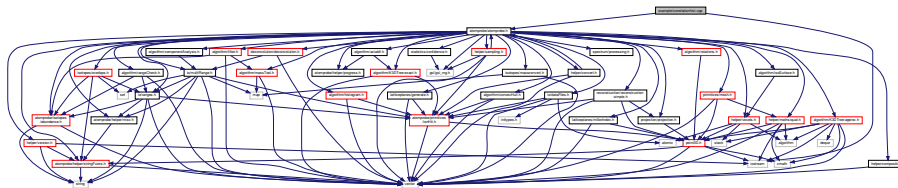
```
= { 'I', 'N', 'F', 'O', ':', 's', 'i', 'z', 'e', '[',
  ('0' + ((SIZE / 10000) % 10)),
  ('0' + ((SIZE / 1000) % 10)),
  ('0' + ((SIZE / 100) % 10)),
  ('0' + ((SIZE / 10) % 10)),
  ('0' + (SIZE % 10)),
  ']',
  '\0' }
```

Definition at line 24 of file SIZEOF_SIZE_T.cpp.

Referenced by main().

7.3 example/correlationhist.cpp File Reference

```
#include <iostream>
#include "atomprobe/atomprobe.h"
Include dependency graph for correlationhist.cpp:
```



Functions

- void [dumpHistogram](#) (std::vector< std::vector< unsigned int > > &hist, float step)
- int [main](#) (int argc, char *argv[])

7.3.1 Function Documentation

7.3.1.1 dumpHistogram()

```
void dumpHistogram (
    std::vector< std::vector< unsigned int > > & hist,
    float step )
```

Definition at line 31 of file correlationhist.cpp.

Referenced by main().

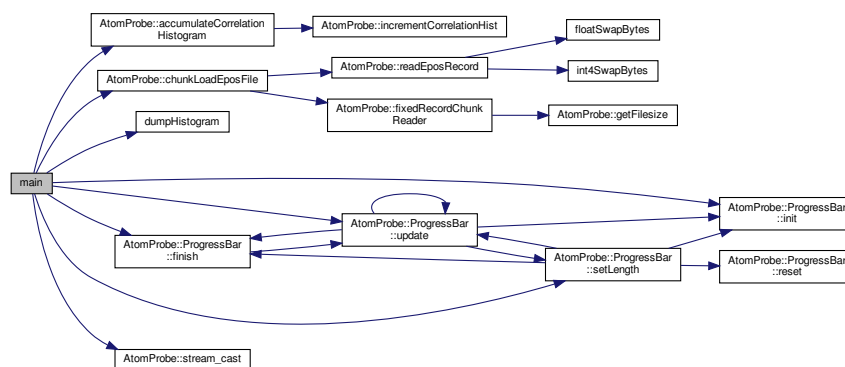
7.3.1.2 main()

```
int main (
    int argc,
    char * argv[] )
```

Definition at line 48 of file correlationhist.cpp.

References `AtomProbe::accumulateCorrelationHistogram()`, `AtomProbe::chunkLoadEposFile()`, `dumpHistogram()`, `AtomProbe::ProgressBar::finish()`, `AtomProbe::ProgressBar::init()`, `AtomProbe::ProgressBar::setLength()`, `AtomProbe::stream_cast()`, and `AtomProbe::ProgressBar::update()`.

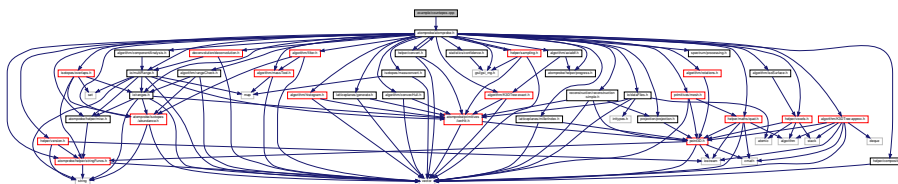
Here is the call graph for this function:



7.4 example/countepos.cpp File Reference

```
#include "atomprobe/atomprobe.h"
```

Include dependency graph for countepos.cpp:



Functions

- `int main` (`int argc`, `char *argv[]`)

7.4.1 Function Documentation

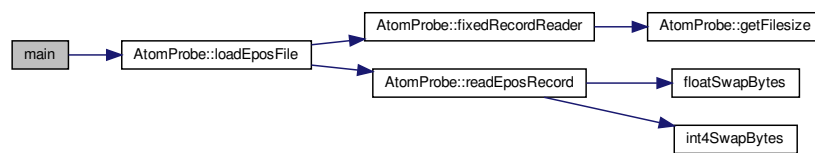
7.4.1.1 main()

```
int main (
    int argc,
    char * argv[] )
```

Definition at line 6 of file countepos.cpp.

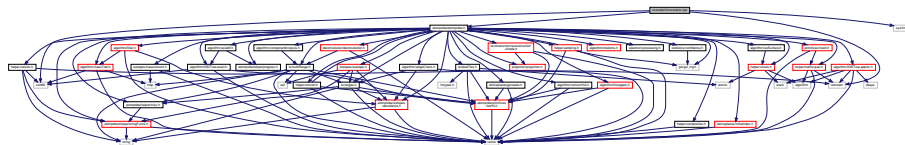
References AtomProbe::loadEposFile().

Here is the call graph for this function:



7.5 example/kd-example.cpp File Reference

```
#include <iostream>
#include <cstdlib>
#include <sys/time.h>
#include "atomprobe/atomprobe.h"
Include dependency graph for kd-example.cpp:
```



Macros

- #define [TIME_START\(\)](#) timeval TIME_DEBUG_t; gettimeofday(&TIME_DEBUG_t,NULL);
- #define [TIME_END\(\)](#)
- #define [M_PI](#) 3.141596254

Functions

- bool [callback](#) ()
- int [main](#) ()

Variables

- unsigned int [progress](#) =0

7.5.1 Macro Definition Documentation

7.5.1.1 M_PI

```
#define M_PI 3.141596254
```

Definition at line 34 of file kd-example.cpp.

Referenced by AtomProbe::gauss_ratio_pdf(), AtomProbe::generate1DAxialDistHistSweep(), main(), AtomProbe::Point3D::parse(), AtomProbe::ReconstructionSphereOnCone::reconstruct(), AtomProbe::ModifiedFocusSphericProjection::scaleUp(), AtomProbe::ReconstructionSphereOnCone::setReconFOV(), AtomProbe::ReconstructionSphereOnCone::setShankAngle(), AtomProbe::StereographicProjection::thetaToEta(), AtomProbe::ModifiedFocusSphericProjection::thetaToEta(), and AtomProbe::GnomonicProjection::toPlanar().

7.5.1.2 TIME_END

```
#define TIME_END( )
```

Value:

```
float TIME_DELTA; {timeval TIME_DEBUG_tend; gettimeofday(&TIME_DEBUG_tend,NULL); \
TIME_DELTA=(TIME_DEBUG_tend.tv_sec - TIME_DEBUG_t.tv_sec) + ((float)TIME_DEBUG_tend.tv_usec-(float) \
TIME_DEBUG_t.tv_usec)/1.0e6;}
```

Definition at line 29 of file kd-example.cpp.

Referenced by main().

7.5.1.3 TIME_START

```
#define TIME_START( ) timeval TIME_DEBUG_t; gettimeofday(&TIME_DEBUG_t,NULL);
```

Definition at line 28 of file kd-example.cpp.

Referenced by main().

7.5.2 Function Documentation

7.5.2.1 callback()

```
bool callback ( )
```

Definition at line 37 of file kd-example.cpp.

References progress.

Referenced by AtomProbe::computeConvexHull(), AtomProbe::GetReducedHullPts(), main(), and AtomProbe::K3DTreeExact::setCallback().

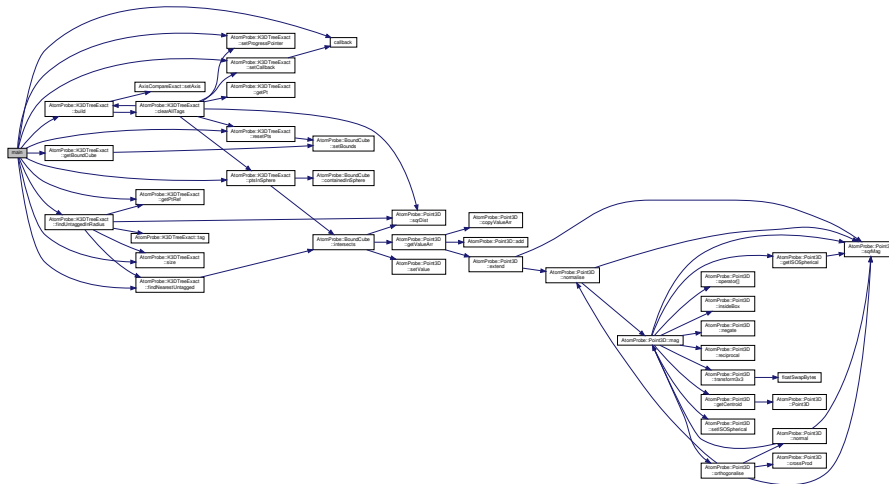
7.5.2.2 main()

```
int main ( )
```

Definition at line 49 of file kd-example.cpp.

References AtomProbe::K3DTreeExact::build(), callback(), AtomProbe::K3DTreeExact::findNearestUntagged(), AtomProbe::K3DTreeExact::findUntaggedInRadius(), AtomProbe::K3DTreeExact::getBoundCube(), AtomProbe::K3DTreeExact::getPtRef(), M_PI, NUM_IONS, progress, AtomProbe::K3DTreeExact::ptsInSphere(), AtomProbe::K3DTreeExact::resetPts(), SCALE, SEARCH_RAD, AtomProbe::K3DTreeExact::setCallback(), AtomProbe::K3DTreeExact::setProgressPointer(), AtomProbe::K3DTreeExact::size(), TIME_END, and TIME_START.

Here is the call graph for this function:



7.5.3 Variable Documentation

7.5.3.1 progress

```
unsigned int progress =0
```

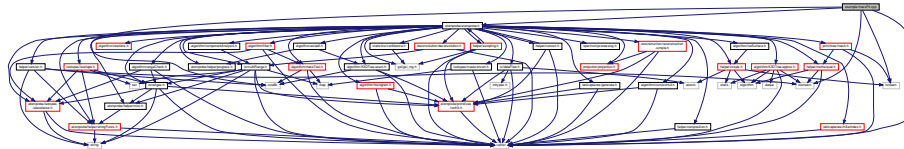
Definition at line 26 of file kd-example.cpp.

Referenced by callback(), main(), and AtomProbe::ProgressBar::update().

7.6 example/massFit.cpp File Reference

```
#include <atomprobe/atomprobe.h>
#include <iostream>
#include <cstdlib>
#include <vector>
#include <fstream>
```

Include dependency graph for massFit.cpp:



Functions

- void [printSolution](#) (const vector< [Weight](#) > &solnComponents, const vector< string > &labels)
- int [main](#) (int argc, const char *argv[])

7.6.1 Function Documentation

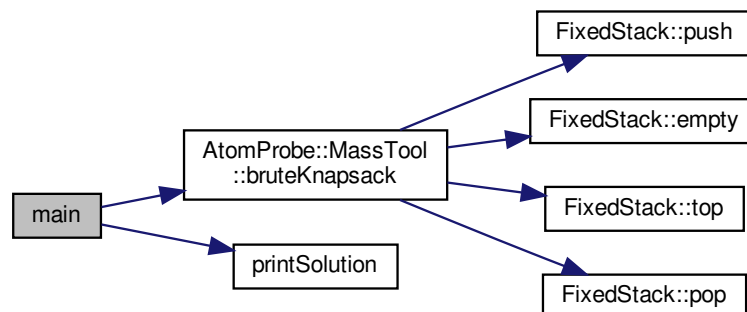
7.6.1.1 main()

```
int main (
    int argc,
    const char * argv[ ] )
```

Definition at line 52 of file massFit.cpp.

References [AtomProbe::MassTool::bruteKnapsack\(\)](#), and [printSolution\(\)](#).

Here is the call graph for this function:



7.6.1.2 printSolution()

```
void printSolution (
    const vector< Weight > & solnComponents,
    const vector< string > & labels )
```

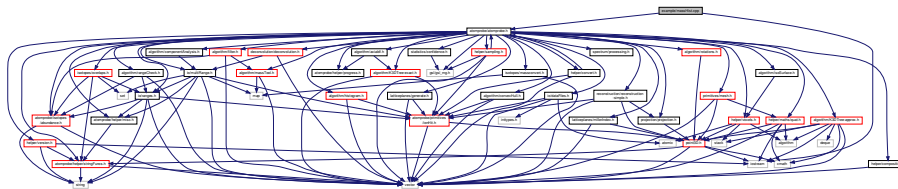
Definition at line 34 of file massFit.cpp.

References AtomProbe::Weight::mass, and AtomProbe::Weight::uniqueld.

Referenced by main().

7.7 example/massHist.cpp File Reference

```
#include <iostream>
#include "atomprobe/atomprobe.h"
Include dependency graph for massHist.cpp:
```



Functions

- int [main](#) (int argc, char *argv[])

7.7.1 Function Documentation

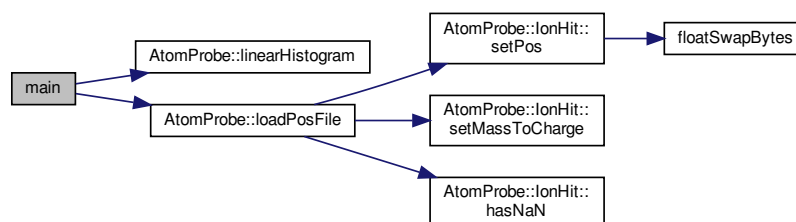
7.7.1.1 main()

```
int main (
    int argc,
    char * argv[] )
```

Definition at line 11 of file massHist.cpp.

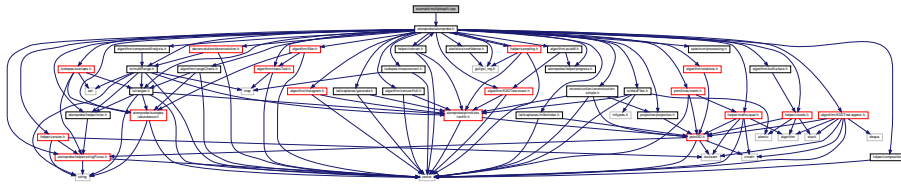
References AtomProbe::linearHistogram(), and AtomProbe::loadPosFile().

Here is the call graph for this function:



7.8 example/multiplesplit.cpp File Reference

```
#include "atomprobe/atomprobe.h"
Include dependency graph for multiplesplit.cpp:
```



Functions

- int [main](#) (int argc, char *argv[])

7.8.1 Function Documentation

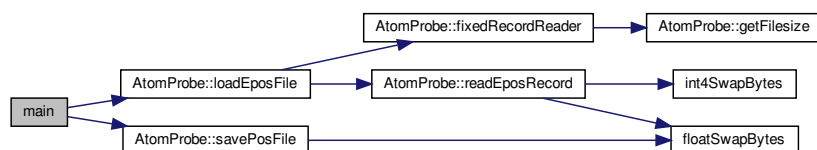
7.8.1.1 main()

```
int main (
    int argc,
    char * argv[] )
```

Definition at line 6 of file multiplesplit.cpp.

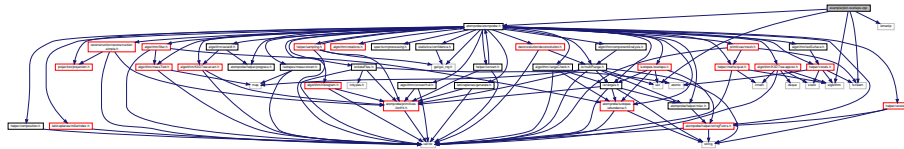
References [AtomProbe::loadEposFile\(\)](#), and [AtomProbe::savePosFile\(\)](#).

Here is the call graph for this function:



7.9 example/plot-overlaps.cpp File Reference

```
#include "atomprobe/atomprobe.h"  
#include <fstream>  
#include <set>  
#include <iomanip>  
#include <algorithm>  
Include dependency graph for plot-overlaps.cpp:
```



Functions

- int [main](#) (int argc, char *argv[])

7.9.1 Function Documentation

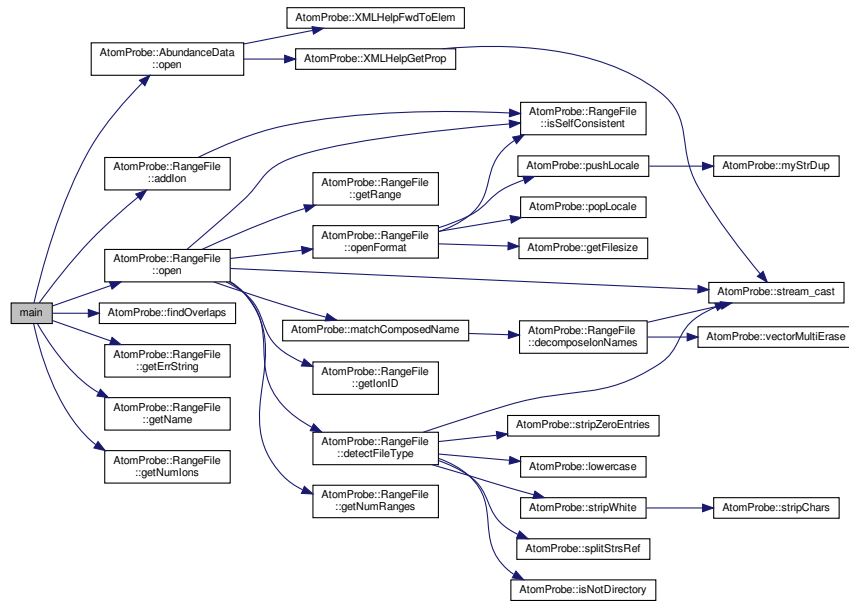
7.9.1.1 main()

```
int main (  
    int argc,  
    char * argv[ ] )
```

Definition at line 21 of file plot-overlaps.cpp.

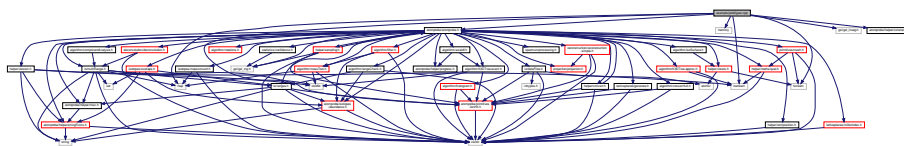
References [AtomProbe::RangeFile::addIon\(\)](#), [AtomProbe::findOverlaps\(\)](#), [AtomProbe::RangeFile::getErrString\(\)](#), [AtomProbe::RangeFile::getName\(\)](#), [AtomProbe::RangeFile::getNumIons\(\)](#), [AtomProbe::AbundanceData::open\(\)](#), and [AtomProbe::RangeFile::open\(\)](#).

Here is the call graph for this function:



7.10 example/polefigure.cpp File Reference

```
#include <iostream>
#include <cstdlib>
#include <fstream>
#include <memory>
#include <map>
#include "atomprobe/atomprobe.h"
#include <gsl/gsl_linalg.h>
#include "atomprobe/helper/constants.h"
Include dependency graph for polefigure.cpp:
```



Functions

- void [gsl_print_matrix](#) (gsl_matrix *m)
- int [main](#) (int argc, char *argv[])

7.10.1 Function Documentation

7.10.1.1 `gsl_print_matrix()`

```
void gsl_print_matrix (
    gsl_matrix * m )
```

Definition at line 15 of file polefigure.cpp.

Referenced by `main()`.

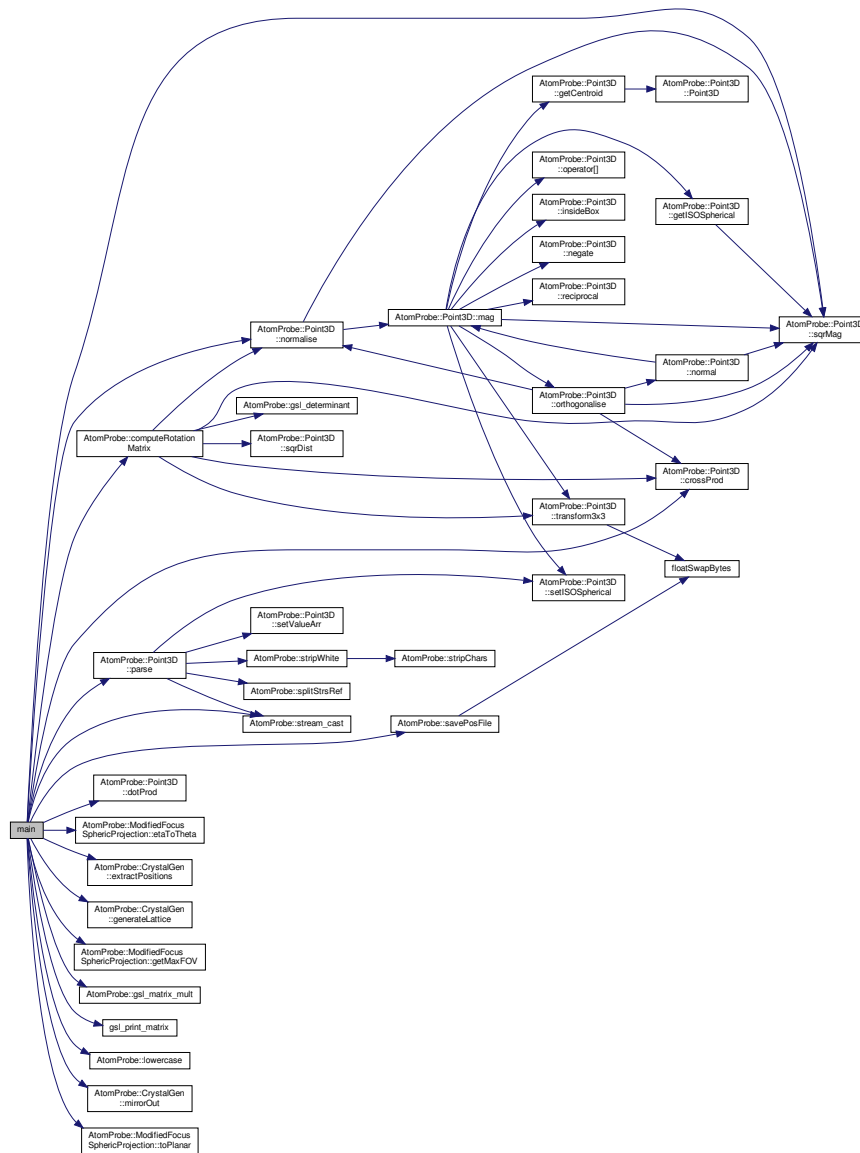
7.10.1.2 `main()`

```
int main (
    int argc,
    char * argv[] )
```

Definition at line 34 of file polefigure.cpp.

References `AtomProbe::computeRotationMatrix()`, `AtomProbe::Point3D::crossProd()`, `AtomProbe::Point3D::dotProd()`, `AtomProbe::ModifiedFocusSphericProjection::etaToTheta()`, `AtomProbe::CrystalGen::extractPositions()`, `AtomProbe::CrystalGen::generateLattice()`, `AtomProbe::ModifiedFocusSphericProjection::getMaxFOV()`, `AtomProbe::gsl_matrix_mult()`, `gsl_print_matrix()`, `AtomProbe::LATTICE_BCC`, `AtomProbe::LATTICE_FCC`, `AtomProbe::lowercase()`, `M_PI`, `AtomProbe::CrystalGen::mirrorOut()`, `AtomProbe::Point3D::normalise()`, `AtomProbe::Point3D::parse()`, `AtomProbe::savePosFile()`, `AtomProbe::Point3D::sqrMag()`, `AtomProbe::stream_cast()`, and `AtomProbe::ModifiedFocusSphericProjection::toPlanar()`.

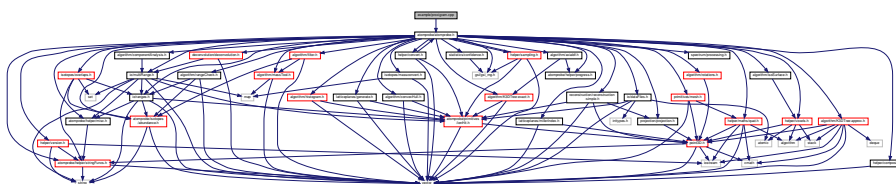
Here is the call graph for this function:



7.11 example/proxigram.cpp File Reference

```
#include "atomprobe/atomprobe.h"
```

Include dependency graph for proxigram.cpp:



Functions

- `int main (int argc, char *argv[])`

7.11.1 Function Documentation

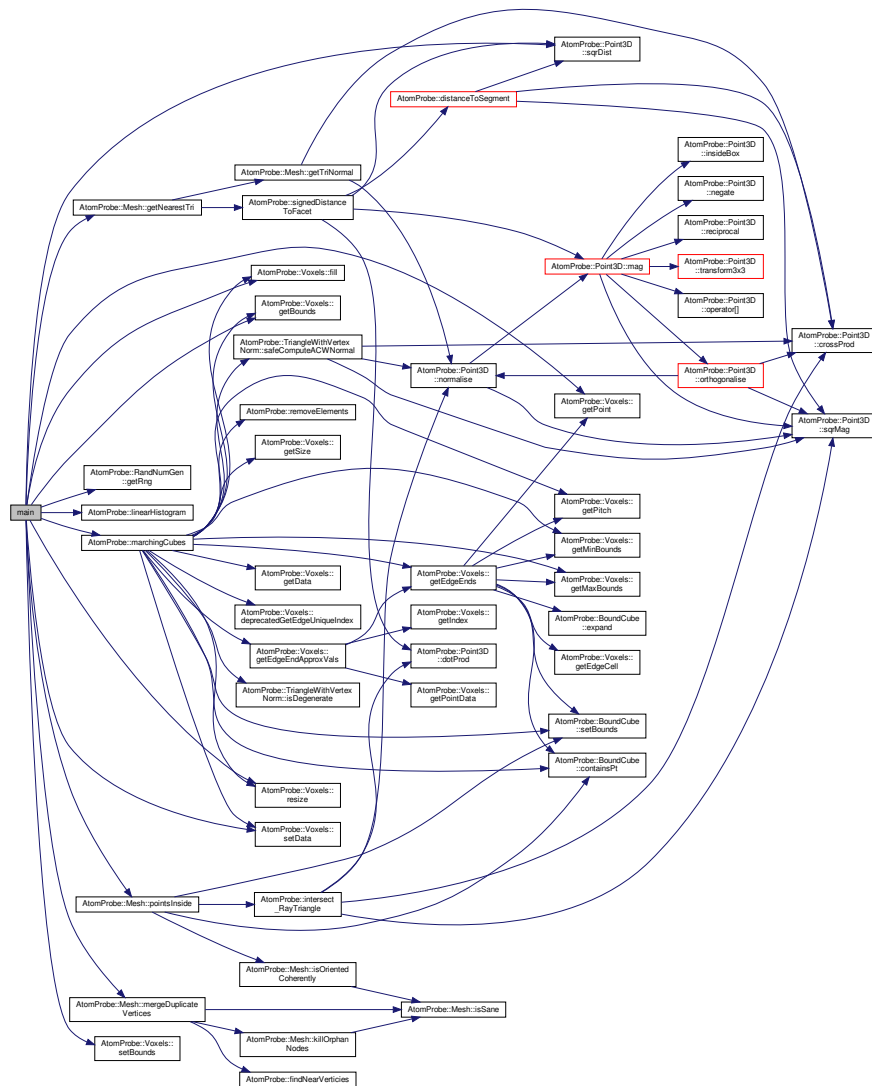
7.11.1.1 main()

```
int main (  
    int argc,  
    char * argv[] )
```

Definition at line 7 of file proxigram.cpp.

References `AtomProbe::Voxels< T >::fill()`, `AtomProbe::Voxels< T >::getBounds()`, `AtomProbe::Mesh::getNearestTri()`, `AtomProbe::Voxels< T >::getPoint()`, `AtomProbe::RandNumGen::getRng()`, `AtomProbe::linearHistogram()`, `AtomProbe::marchingCubes()`, `AtomProbe::Mesh::mergeDuplicateVertices()`, `AtomProbe::Mesh::nodes`, `AtomProbe::Mesh::pointsInside()`, `progress`, `AtomProbe::randGen`, `AtomProbe::Voxels< T >::resize()`, `AtomProbe::Voxels< T >::setBounds()`, `AtomProbe::Voxels< T >::setData()`, `AtomProbe::Point3D::sqrDist()`, and `AtomProbe::Mesh::triangles`.

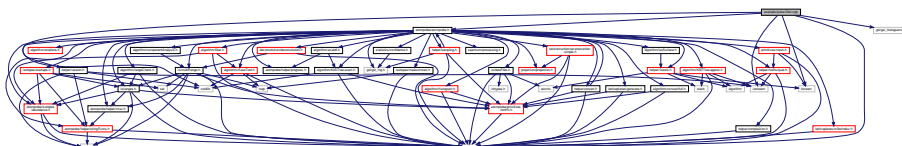
Here is the call graph for this function:



7.12 example/pulse-filter.cpp File Reference

```
#include <atomprobe/atomprobe.h>
#include <gsl/gsl_histogram2d.h>
#include <iostream>
#include <cstdlib>
#include <vector>
#include <fstream>
```

Include dependency graph for pulse-filter.cpp:



Enumerations

- enum { [ERR_BAD_ARGS](#), [ERR_BAD_EPOS_READ](#), [ERR_BAD_NUMSAMPLES](#), [ERR_NOMEM](#) }

Functions

- bool [convertToPos](#) (const vector< [EPOS_ENTRY](#) > &epos, vector< [IonHit](#) > &pos)
- bool [dumpHistogram](#) (gsl_histogram2d *h, const char *outFile)
- void [filterEposByPulse](#) (const vector< [EPOS_ENTRY](#) > &input, unsigned int filterCount, vector< [EPOS_ENTRY](#) > &output)
- int [main](#) (int argc, const char *argv[])

7.12.1 Enumeration Type Documentation

7.12.1.1 anonymous enum

anonymous enum

Enumerator

ERR_BAD_ARGS	
ERR_BAD_EPOS_READ	
ERR_BAD_NUMSAMPLES	
ERR_NOMEM	

Definition at line 33 of file pulse-filter.cpp.

7.12.2 Function Documentation

7.12.2.1 convertToPos()

```
bool convertToPos (
    const vector< EPOS\_ENTRY > & epos,
    vector< IonHit > & pos )
```

Definition at line 42 of file pulse-filter.cpp.

7.12.2.2 dumpHistogram()

```
bool dumpHistogram (
    gsl_histogram2d * h,
    const char * outFile )
```

Definition at line 64 of file pulse-filter.cpp.

7.12.2.3 filterEposByPulse()

```
void filterEposByPulse (
    const vector< EPOS_ENTRY > & input,
    unsigned int filterCount,
    vector< EPOS_ENTRY > & output )
```

Definition at line 84 of file pulse-filter.cpp.

Referenced by main().

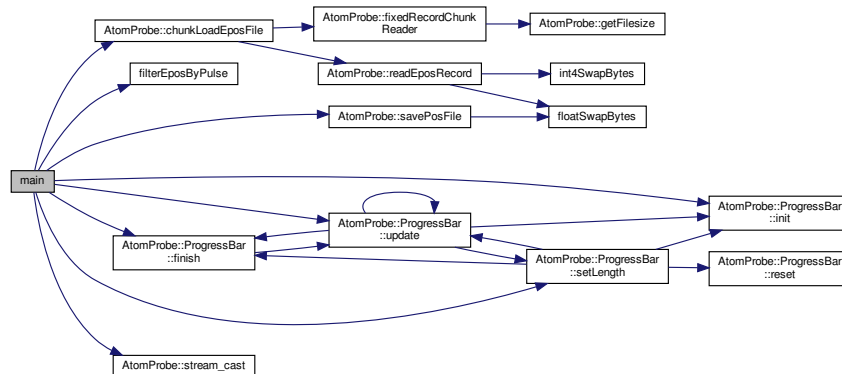
7.12.2.4 main()

```
int main (
    int argc,
    const char * argv[] )
```

Definition at line 102 of file pulse-filter.cpp.

References AtomProbe::chunkLoadEposFile(), ERR_BAD_ARGS, ERR_BAD_EPOS_READ, filterEposByPulse(), AtomProbe::ProgressBar::finish(), AtomProbe::ProgressBar::init(), AtomProbe::RECORDREAD_ERR_STRINGS, AtomProbe::savePosFile(), AtomProbe::ProgressBar::setLength(), AtomProbe::stream_cast(), and AtomProbe::ProgressBar::update().

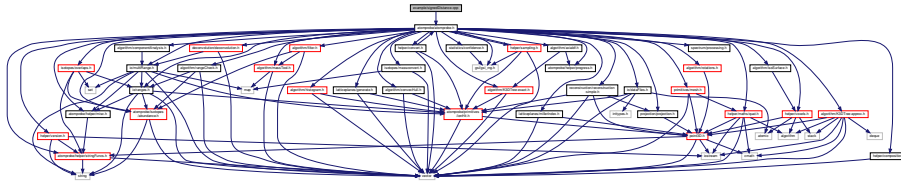
Here is the call graph for this function:



7.13 example/signedDistance.cpp File Reference

```
#include "atomprobe/atomprobe.h"
```

Include dependency graph for signedDistance.cpp:



Functions

- int [main](#) ()

7.13.1 Function Documentation

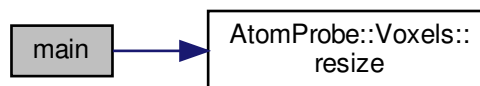
7.13.1.1 main()

```
int main ( )
```

Definition at line 8 of file signedDistance.cpp.

References [AtomProbe::Voxels< T >::resize\(\)](#).

Here is the call graph for this function:



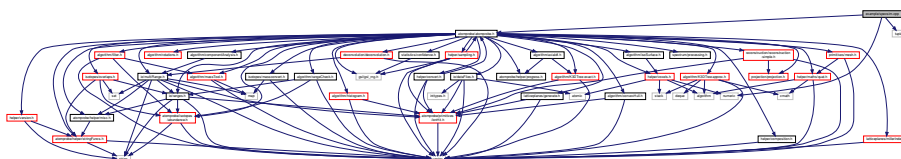
7.14 example/specsim.cpp File Reference

```
#include "atomprobe/atomprobe.h"
```

```
#include <tuple>
```

```
#include <numeric>
```

Include dependency graph for specsim.cpp:



Typedefs

- typedef vector< pair< string, size_t > > [FRAGMENT](#)

Functions

- void [getMassDistributions](#) (const vector< [FRAGMENT](#) > &fragmentVec, const vector< string > &parentSpecies, const [AbundanceData](#) &natTable, vector< vector< pair< float, float > > > &massDistVec)
- void [normaliseVec](#) (vector< float > &v)
- int [main](#) (int argc, char *argv[])

Variables

- const float [MASS_TOL](#) =0.05

7.14.1 Typedef Documentation

7.14.1.1 FRAGMENT

```
typedef vector<pair<string,size_t> > FRAGMENT
```

Definition at line 11 of file specsim.cpp.

7.14.2 Function Documentation

7.14.2.1 getMassDistributions()

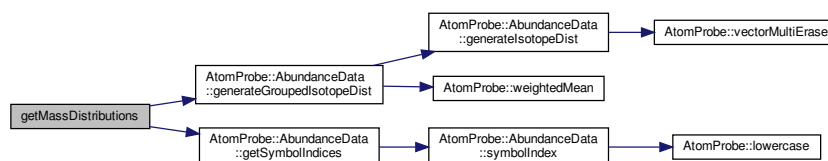
```
void getMassDistributions (
    const vector< FRAGMENT > & fragmentVec,
    const vector< string > & parentSpecies,
    const AbundanceData & natTable,
    vector< vector< pair< float, float > > > & massDistVec )
```

Definition at line 358 of file specsim.cpp.

References [AtomProbe::AbundanceData::generateGroupedIsotopeDist\(\)](#), [AtomProbe::AbundanceData::getSymbolIndices\(\)](#), and [MASS_TOL](#).

Referenced by [main\(\)](#).

Here is the call graph for this function:



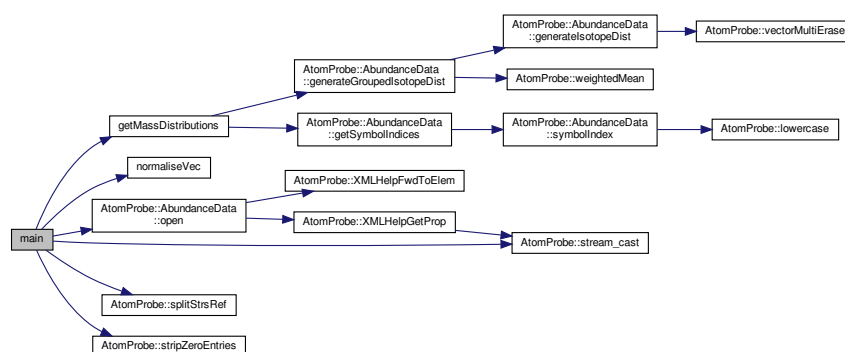
7.14.2.2 main()

```
int main (
    int argc,
    char * argv[] )
```

Definition at line 26 of file specsim.cpp.

References `getMassDistributions()`, `MASS_TOL`, `normaliseVec()`, `AtomProbe::AbundanceData::open()`, `AtomProbe::splitStrsRef()`, `AtomProbe::stream_cast()`, and `AtomProbe::stripZeroEntries()`.

Here is the call graph for this function:



7.14.2.3 normaliseVec()

```
void normaliseVec (
    vector< float > & v )
```

Definition at line 18 of file specsim.cpp.

Referenced by `main()`.

7.14.3 Variable Documentation

7.14.3.1 MASS_TOL

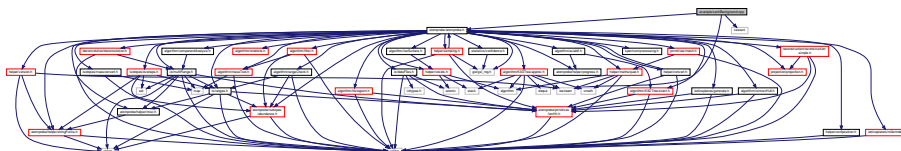
```
const float MASS_TOL =0.05
```

Definition at line 9 of file specsim.cpp.

Referenced by `AtomProbe::MultiRange::copyDataFromRange()`, `getMassDistributions()`, and `main()`.

7.15 example/zechBackground.cpp File Reference

```
#include <iostream>
#include "atomprobe/atomprobe.h"
#include <cassert>
Include dependency graph for zechBackground.cpp:
```



Macros

- #define [ASSERT](#)(f) assert(f)

Functions

- template<class T >
bool [dumpHistogramToFile](#) (std::vector< std::vector< T > > &hist, const char *filename)
- void [zechCorrect](#) (vector< float > &background, vector< float > &observation, float alpha, vector< float > &corrected)
- int [main](#) (int argc, char *argv[])

7.15.1 Macro Definition Documentation

7.15.1.1 ASSERT

```
#define ASSERT(  
    f ) assert(f)
```

Definition at line 26 of file zechBackground.cpp.

Referenced by [AtomProbe::AbundanceData::abundanceBetweenLimits\(\)](#), [AtomProbe::MultiRange::addlon\(\)](#), [AtomProbe::RangeFile::addlon\(\)](#), [AtomProbe::MultiRange::addRange\(\)](#), [AtomProbe::RangeFile::addRange\(\)](#), [AtomProbe::Voxels< T >::applyMask\(\)](#), [AtomProbe::K3DTreeExact::build\(\)](#), [AtomProbe::buildFrequencyTable\(\)](#), [AtomProbe::K3DTreeExact::clearAllTags\(\)](#), [AtomProbe::computeIonDistAdjacency\(\)](#), [AtomProbe::computeRangeAdjacency\(\)](#), [AtomProbe::computeRotationMatrix\(\)](#), [AtomProbe::BoundCube::containsPt\(\)](#), [AtomProbe::convertMolToMass\(\)](#), [AtomProbe::MultiRange::copyDataFromRange\(\)](#), [AtomProbe::Point3D::copyValueArr\(\)](#),

AtomProbe::countDataDistanceWeight(), AtomProbe::cumTrapezoid(), AtomProbe::Determinant(), AtomProbe::Mesh::divideMeshSurface(), AtomProbe::doFitBackground(), AtomProbe::edgIdx(), AtomProbe::TRIANGLE::edgesMismatch(), AtomProbe::AbundanceData::elementNames(), AtomProbe::RangeFile::eraselon(), AtomProbe::RangeFile::eraseRange(), AtomProbe::ModifiedFocusSphericProjection::etaToTheta(), AtomProbe::Point3D::extend(), AtomProbe::filterPeakNeedBiggerObs(), AtomProbe::findMaxLessThanOrEq(), AtomProbe::K3DTreeApprox::findNearest(), AtomProbe::K3DTreeExact::findNearestUntagged(), AtomProbe::findNearVertices(), AtomProbe::findOverlaps(), AtomProbe::findPeaks(), AtomProbe::K3DTreeExact::findUntaggedInRadius(), AtomProbe::MultiRange::flattenToMassAxis(), AtomProbe::generate1DAxialDistHist(), AtomProbe::generate1DAxialDistHistSweep(), AtomProbe::AbundanceData::generateIsotopeDist(), AtomProbe::RangeFile::getAllExts(), AtomProbe::AbundanceData::getAtomicNumber(), AtomProbe::BoundCube::getBound(), AtomProbe::IonHit::getBoundCube(), AtomProbe::K3DTreeExact::getBoundCube(), AtomProbe::BoundCube::getBounds(), AtomProbe::Mesh::getBounds(), AtomProbe::BoundCube::getCentroid(), AtomProbe::MultiRange::getColour(), AtomProbe::RangeFile::getColour(), AtomProbe::Mesh::getContainedNodes(), AtomProbe::Voxels< T >::getData(), AtomProbe::Voxels< T >::getEdgeCell(), AtomProbe::Voxels< T >::getEdgeEnds(), AtomProbe::AbundanceData::getErrorText(), AtomProbe::MultiRange::getErrString(), AtomProbe::getFitErrorMsg(), AtomProbe::Voxels< T >::getIndex(), AtomProbe::Voxels< T >::getInterpolatedData(), AtomProbe::Voxels< T >::getInterpSlice(), AtomProbe::Mesh::getIntersectingPrimitives(), AtomProbe::IonHit::getIonDataLimits(), AtomProbe::MultiRange::getIonID(), AtomProbe::RangeFile::getIonID(), AtomProbe::MultiRange::getIonName(), AtomProbe::AbundanceData::getMajorIsotopeFromElemIdx(), AtomProbe::Voxels< T >::getMaxBounds(), AtomProbe::BoundCube::getMaxDistanceToBox(), AtomProbe::Voxels< T >::getMinBounds(), AtomProbe::MultiRange::getMolecule(), AtomProbe::RangeFile::getName(), AtomProbe::AbundanceData::getNearestCharge(), AtomProbe::MultiRange::getNumRanges(), AtomProbe::K3DTreeExact::getOrigIndex(), AtomProbe::Voxels< T >::getPointData(), AtomProbe::K3DTreeExact::getPt(), AtomProbe::K3DTreeExact::getPtRef(), AtomProbe::MultiRange::getRange(), AtomProbe::RangeFile::getRangeVolume(), AtomProbe::GetReducedHullPts(), AtomProbe::BoundCube::getSize(), AtomProbe::Voxels< T >::getSize(), AtomProbe::Voxels< T >::getSlice(), AtomProbe::Voxels< T >::getSum(), AtomProbe::K3DTreeExact::getTag(), AtomProbe::Mesh::getTriNormal(), AtomProbe::Mesh::getVolume(), AtomProbe::gsl_determinant(), AtomProbe::hexStrToUChar(), AtomProbe::Mesh::isOrientedCoherently(), AtomProbe::AbundanceData::isotope(), AtomProbe::AbundanceData::isotopeIndex(), AtomProbe::AbundanceData::isotopes(), AtomProbe::RangeFile::isSelfConsistent(), AtomProbe::Mesh::killOrphanNodes(), AtomProbe::leastSquaresDeconvolve(), AtomProbe::linearHistogram(), AtomProbe::loadATOFFile(), AtomProbe::loadPosFile(), main(), AtomProbe::makeHistogram(), AtomProbe::marchingCubes(), AtomProbe::BoundCube::max(), AtomProbe::Voxels< T >::max(), AtomProbe::maxExplainedFraction(), AtomProbe::Mesh::mergeDuplicateVertices(), AtomProbe::BoundCube::min(), AtomProbe::Voxels< T >::min(), AtomProbe::Voxels< T >::minMax(), AtomProbe::MultiRange::MultiRange(), AtomProbe::Mesh::numDupTris(), AtomProbe::numericalEstimateGaussRatioConf(), AtomProbe::numericalEstimatePoissRatioConf(), AtomProbe::numericalEstimateSkellamConf(), AtomProbe::RangeFile::open(), AtomProbe::RangeFile::openFormat(), AtomProbe::Voxels< T >::operator/=(), AtomProbe::IonHit::operator[](), AtomProbe::Point3D::operator[](), AtomProbe::pairContains(), AtomProbe::pairOverlaps(), AtomProbe::Mesh::pointsInside(), AtomProbe::poissonConfidenceObservation(), FixedStack< T >::pop(), AtomProbe::Mesh::print(), AtomProbe::K3DTreeExact::ptsInSphere(), AtomProbe::pushLocale(), AtomProbe::pyramidVol(), AtomProbe::quat_get_rot_quat(), AtomProbe::quat_rot(), AtomProbe::quat_rot_array(), AtomProbe::RangeFile::rangeTypeString(), AtomProbe::readPosapOps(), AtomProbe::ReconstructionSphereOnCone::reconstruct(), AtomProbe::Mesh::removeDuplicateTris(), AtomProbe::removeElements(), AtomProbe::Voxels< T >::resizeKeepData(), AtomProbe::Mesh::saveGmshMesh(), AtomProbe::saveTapsimBin(), AtomProbe::GnomonicProjection::scaleDown(), AtomProbe::StereographicProjection::scaleDown(), AtomProbe::ModifiedFocusSphericProjection::scaleDown(), AtomProbe::GnomonicProjection::scaleUp(), AtomProbe::StereographicProjection::scaleUp(), AtomProbe::ModifiedFocusSphericProjection::scaleUp(), AtomProbe::BoundCube::segmentTriple(), AtomProbe::BoundCube::setBound(), AtomProbe::Voxels< T >::setBounds(), AtomProbe::RangeFile::setColour(), AtomProbe::Voxels< T >::setData(), AtomProbe::ReconstructionSphereOnCone::setDetectorEfficiency(), AtomProbe::ReconstructionSphereOnCone::setFlightPath(), AtomProbe::ReconstructionSphereOnCone::setInitialRadius(), AtomProbe::MultiRange::setIonID(), AtomProbe::RangeFile::setIonID(), AtomProbe::LinearFeedbackShiftReg::setMaskPeriod(), AtomProbe::Voxels< T >::setPoint(), AtomProbe::IonHit::setPos(), AtomProbe::RangeFile::setRangeEnd(), AtomProbe::RangeFile::setRangeStart(), AtomProbe::RangeFile::setRangeVolume(), AtomProbe::ReconstructionSphereOnCone::setReconFOV(), AtomProbe::ReconstructionSphereOnCone::setShankAngle(), AtomProbe::Mesh::setTriangleMesh(), AtomProbe::signedDistanceToFacet(), AtomProbe::K3DTreeExact::size(), AtomProbe::MultiRange::splitOverlapping(), AtomProbe::AbundanceData::symbolIndex(), AtomProbe::K3DTreeExact::tag(), AtomProbe::GnomonicProjection::toPlanar(), AtomProbe::vectorPointDir(), AtomProbe::LinearFeedbackShiftReg::verifyTable(), AtomProbe::MultiRange::write(), AtomProbe::RangeFile::write(), AtomProbe::Voxels< T >::writeFile(), Atom

Probe::zechConfidenceLimits(), and zechCorrect().

7.15.2 Function Documentation

7.15.2.1 dumpHistogramToFile()

```
template<class T >
bool dumpHistogramToFile (
    std::vector< std::vector< T > > & hist,
    const char * filename )
```

Definition at line 30 of file zechBackground.cpp.

Referenced by main().

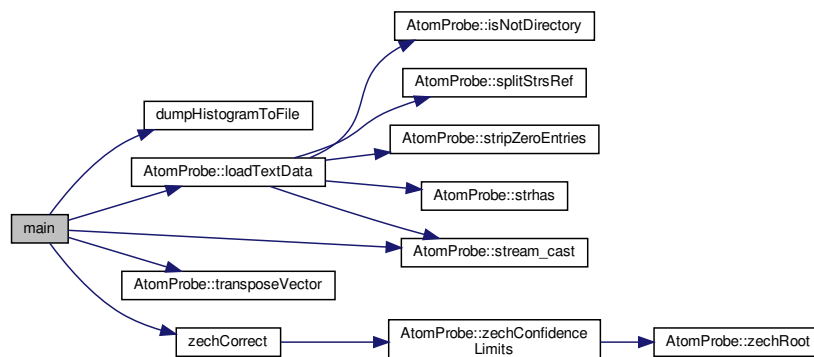
7.15.2.2 main()

```
int main (
    int argc,
    char * argv[] )
```

Definition at line 80 of file zechBackground.cpp.

References ASSERT, dumpHistogramToFile(), AtomProbe::loadTextData(), AtomProbe::stream_cast(), AtomProbe::transposeVector(), and zechCorrect().

Here is the call graph for this function:



7.15.2.3 zechCorrect()

```
void zechCorrect (
    vector< float > & background,
    vector< float > & observation,
    float alpha,
    vector< float > & corrected )
```

Definition at line 49 of file zechBackground.cpp.

References ASSERT, and AtomProbe::zechConfidenceLimits().

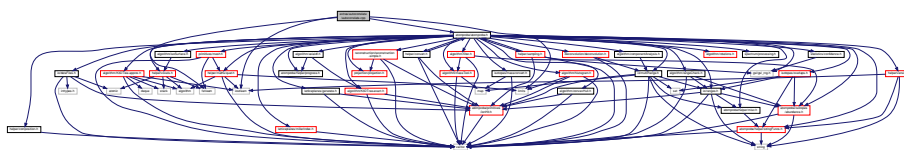
Referenced by main().

Here is the call graph for this function:



7.16 extras/autocorrelate/autocorrelate.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <limits>
#include "atomprobe/atomprobe.h"
Include dependency graph for autocorrelate.cpp:
```



Macros

- `#define ASSERT(f) {}`

Functions

- bool `callback ()`
- `template<class T >`
void `dumpAutocorrelation` (const vector< vector< T > > &autoCorrelateFunc, const `RangeFile` &rangeFile, float binStep, std::ostream &strm)
- int `main` (int argc, char *argv[])

Variables

- const unsigned int `PROGRESS_BAR_SIZE` =50
- `ProgressBar` * `treeProgressBar` =0
- unsigned int `treeProgressValue` =0

7.16.1 Macro Definition Documentation

7.16.1.1 ASSERT

```
#define ASSERT(  
    f ) {}
```

Definition at line 35 of file autocorrelate.cpp.

Referenced by `dumpAutocorrelation()`, and `main()`.

7.16.2 Function Documentation

7.16.2.1 callback()

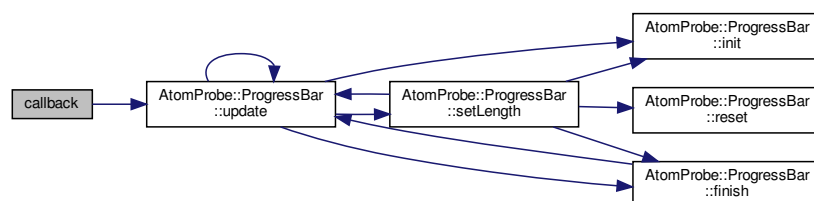
```
bool callback ( )
```

Definition at line 43 of file autocorrelate.cpp.

References `treeProgressValue`, and `AtomProbe::ProgressBar::update()`.

Referenced by `main()`.

Here is the call graph for this function:



7.16.2.2 dumpAutocorrelation()

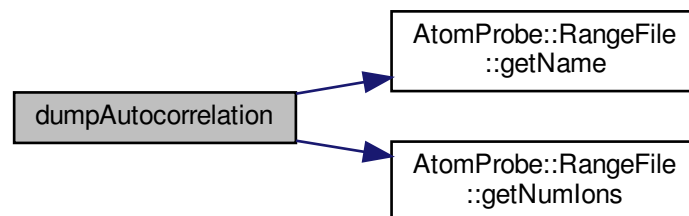
```
template<class T >
void dumpAutocorrelation (
    const vector< vector< T > > & autoCorrelateFunc,
    const RangeFile & rangeFile,
    float binStep,
    std::ostream & strm )
```

Definition at line 59 of file autocorrelate.cpp.

References ASSERT, AtomProbe::RangeFile::getName(), and AtomProbe::RangeFile::getNumIons().

Referenced by main().

Here is the call graph for this function:



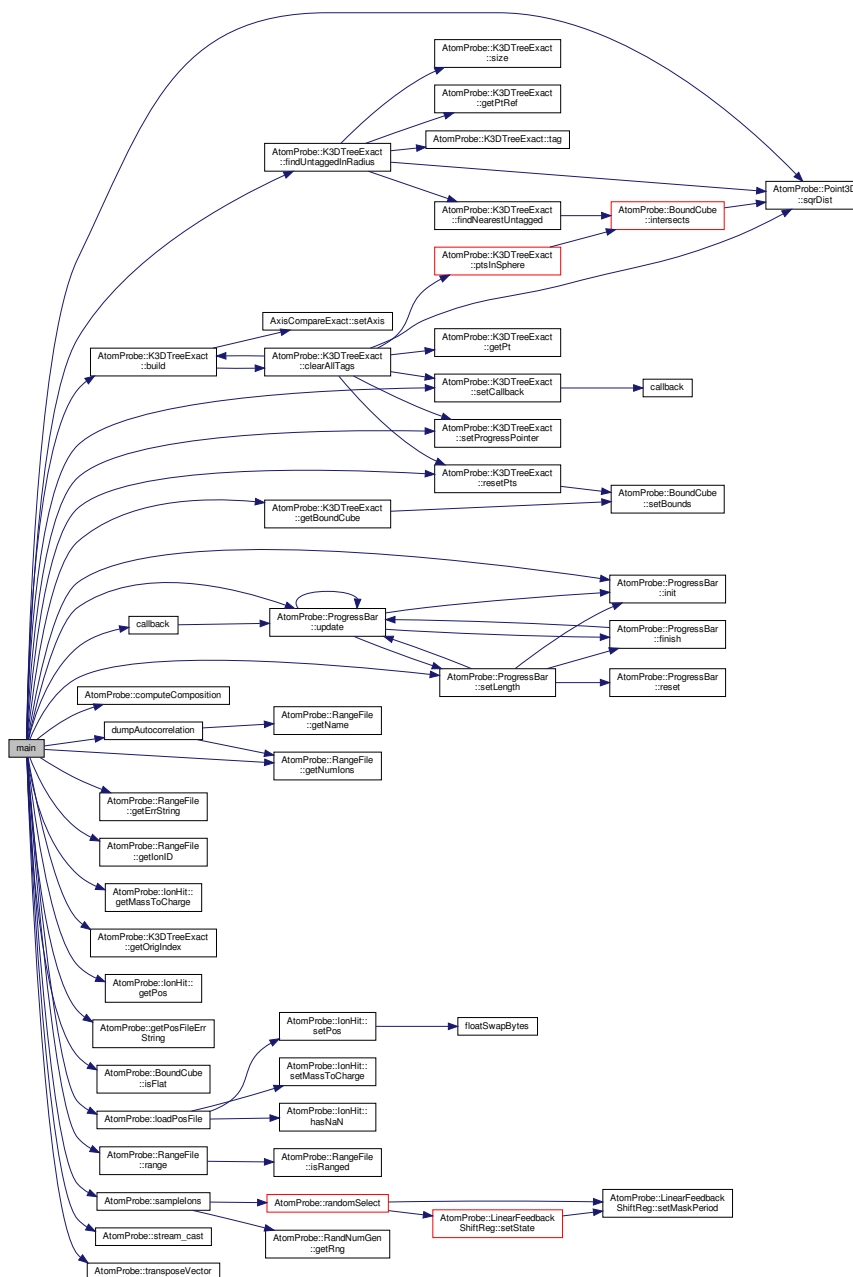
7.16.2.3 main()

```
int main (
    int argc,
    char * argv[] )
```

Definition at line 86 of file autocorrelate.cpp.

References ASSERT, AtomProbe::K3DTreeExact::build(), callback(), AtomProbe::computeComposition(), dumpAutocorrelation(), AtomProbe::K3DTreeExact::findUntaggedInRadius(), AtomProbe::K3DTreeExact::getBoundCube(), AtomProbe::RangeFile::getErrString(), AtomProbe::RangeFile::getIonID(), AtomProbe::IonHit::getMassToCharge(), AtomProbe::RangeFile::getNumIons(), AtomProbe::K3DTreeExact::getOrigIndex(), AtomProbe::IonHit::getPos(), AtomProbe::getPosFileErrString(), AtomProbe::ProgressBar::init(), AtomProbe::BoundCube::isFlat(), AtomProbe::loadPosFile(), PROGRESS_BAR_SIZE, AtomProbe::RangeFile::range(), AtomProbe::K3DTreeExact::resetPts(), AtomProbe::sampleIons(), AtomProbe::K3DTreeExact::setCallback(), AtomProbe::ProgressBar::setLength(), AtomProbe::K3DTreeExact::setProgressPointer(), AtomProbe::Point3D::sqrDist(), AtomProbe::stream_cast(), AtomProbe::transposeVector(), treeProgressBar, treeProgressValue, and AtomProbe::ProgressBar::update().

Here is the call graph for this function:



7.16.3 Variable Documentation

7.16.3.1 PROGRESS_BAR_SIZE

```
const unsigned int PROGRESS_BAR_SIZE =50
```

Definition at line 38 of file autocorrelate.cpp.

Referenced by main().

7.16.3.2 treeProgressBar

```
ProgressBar* treeProgressBar =0
```

Definition at line 40 of file autocorrelate.cpp.

Referenced by main().

7.16.3.3 treeProgressValue

```
unsigned int treeProgressValue =0
```

Definition at line 41 of file autocorrelate.cpp.

Referenced by callback(), and main().

7.17 extras/readpos_c/readpos.h File Reference

Functions

- unsigned int [readPos](#) (const char *filename, float **x, float **y, float **z, float **m, unsigned int *nPts)

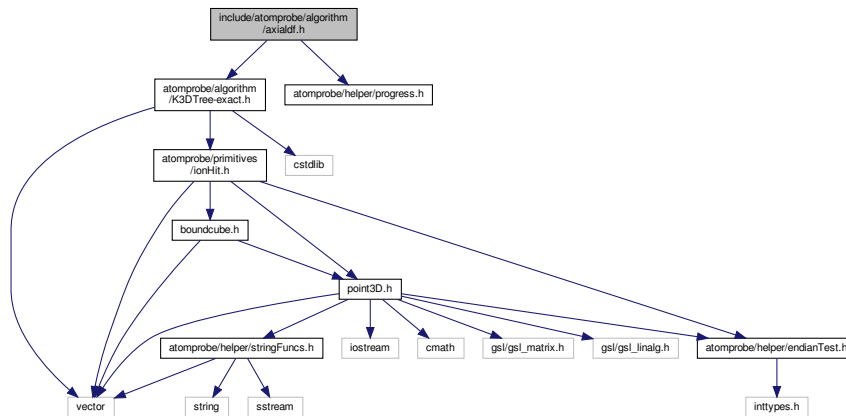
7.17.1 Function Documentation

7.17.1.1 readPos()

```
unsigned int readPos (  
    const char * filename,  
    float ** x,  
    float ** y,  
    float ** z,  
    float ** m,  
    unsigned int * nPts )
```

7.18 include/atomprobe/algorithm/axialdf.h File Reference

```
#include "atomprobe/algorithm/K3DTree-exact.h"
#include "atomprobe/helper/progress.h"
Include dependency graph for axialdf.h:
```



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



Namespaces

- [AtomProbe](#)

Functions

- unsigned int [AtomProbe::generate1DAxialDistHist](#) (const std::vector< Point3D > &pointList, K3DTreeExact &tree, const Point3D &axisDir, float distMax, std::vector< unsigned int > &histogram)

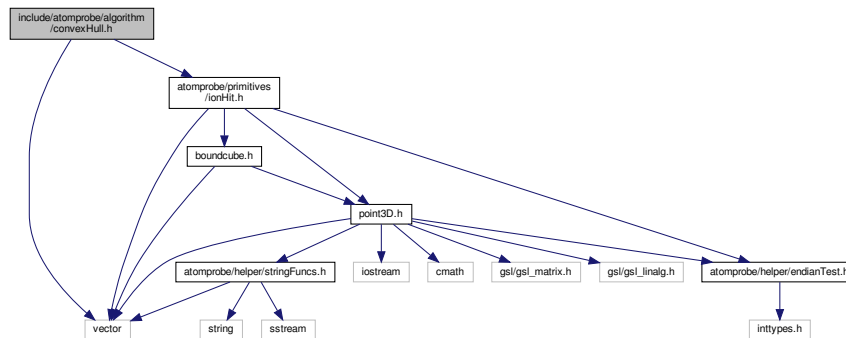
Generate a 1D axial distribution function,.

- unsigned int [AtomProbe::generate1DAxialDistHistSweep](#) (const std::vector< Point3D > &pointList, K3DTreeExact &tree, float distMax, float dTheta, float dPhi, ProgressBar &prog, std::vector< std::vector< std::vector< unsigned int > > > &histogram)

Generate a series of 1D distribution functions, one per pixel in a 2D grid of spherical coordinate directions.

7.20 include/atomprobe/algorithm/convexHull.h File Reference

```
#include <vector>
#include "atomprobe/primitives/ionHit.h"
Include dependency graph for convexHull.h:
```



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



Namespaces

- [AtomProbe](#)

Enumerations

- enum { [AtomProbe::HULL_SURFREDUCE_NEGATIVE_SCALE_FACT](#) =1, [AtomProbe::HULL_ERR_USE_ABORT](#), [AtomProbe::HULL_ERR_NO_MEM](#) }

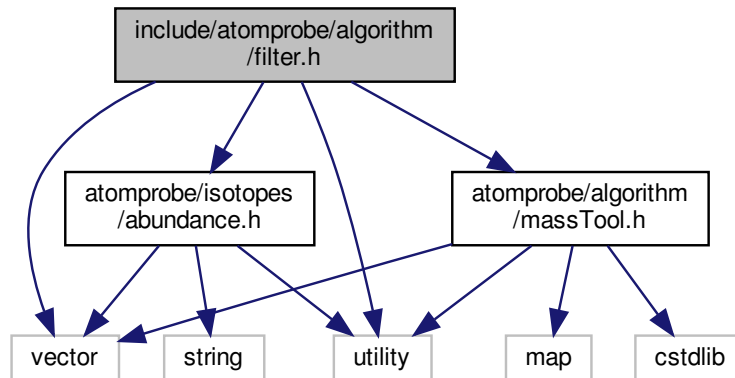
Functions

- unsigned int [AtomProbe::GetReducedHullPts](#) (const std::vector< [IonHit](#) > &points, float reductionDim, unsigned int *progress, bool(*callback)(bool), std::vector< [IonHit](#) > &pointResult)
Obtain a set of points that are a subset of points the convex hull that are at least "reductionDim" away from the hull itself.
- unsigned int [AtomProbe::computeConvexHull](#) (const std::vector< [IonHit](#) > &data, unsigned int *progress, bool(*callback)(bool), std::vector< [Point3D](#) > &curHull, bool freeHull)
Obtain the convex hull of a set of ions.

7.21 include/atomprobe/algorithm/filter.h File Reference

```
#include <vector>
#include <utility>
#include "atomprobe/isotopes/abundance.h"
#include "atomprobe/algorithm/massTool.h"
```

Include dependency graph for filter.h:



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



Namespaces

- [AtomProbe](#)

Functions

- void [AtomProbe::filterPeakNeedBiggerObs](#) (const AbundanceData &massTable, const std::vector< float > &peakData, float tolerance, size_t solutionCharge, std::vector< std::vector< ISOTOPE_ENTRY > > &solutions)
- void [AtomProbe::filterBySolutionPPM](#) (const AbundanceData &massTable, float minPpm, std::vector< std::vector< ISOTOPE_ENTRY > > &solutions)

Use the maximum possible PPM for each isotopic combination to filter possible solutions.

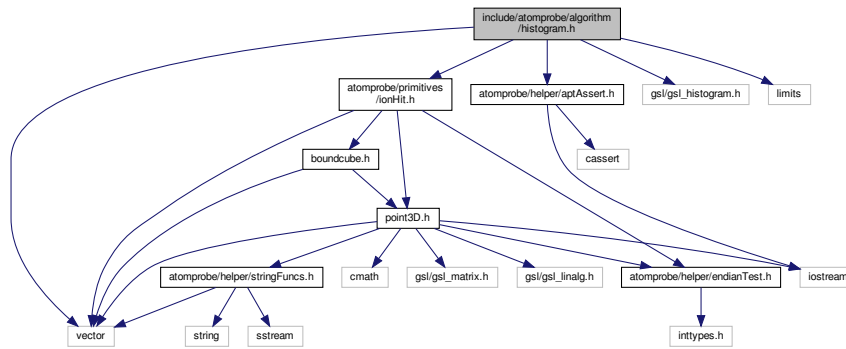
- std::vector< float > [AtomProbe::maxExplainedFraction](#) (const std::vector< std::pair< float, float > > &massData, float peakMass, float massWidth, const std::vector< std::vector< ISOTOPE_ENTRY > > &solutions, const AbundanceData &massTable, float massDistTol, unsigned int solutionCharge)

Compute the fraction of the data that has been explained, using the natural abundance information, and intensity data.

7.22 include/atomprobe/algorithm/histogram.h File Reference

```
#include <vector>
#include "atomprobe/primitives/ionHit.h"
#include "atomprobe/helper/aptAssert.h"
#include <gsl/gsl_histogram.h>
#include <limits>
```

Include dependency graph for histogram.h:



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



Namespaces

- [AtomProbe](#)

Functions

- bool [AtomProbe::correlationHistogram](#) (const std::vector< EPOS_ENTRY > &eposlons, std::vector< std::vector< unsigned int > > &histogram, float stepMass, float endMass, bool lowerTriangular=false)

Generates an ion "correlation histogram" from a vector of EPOS ions. The input vector MUST
- bool [AtomProbe::accumulateCorrelationHistogram](#) (const std::vector< EPOS_ENTRY > &eposlons, std::vector< std::vector< unsigned int > > &histogram, float stepMass, float endMass, bool lowerTriangular=true)

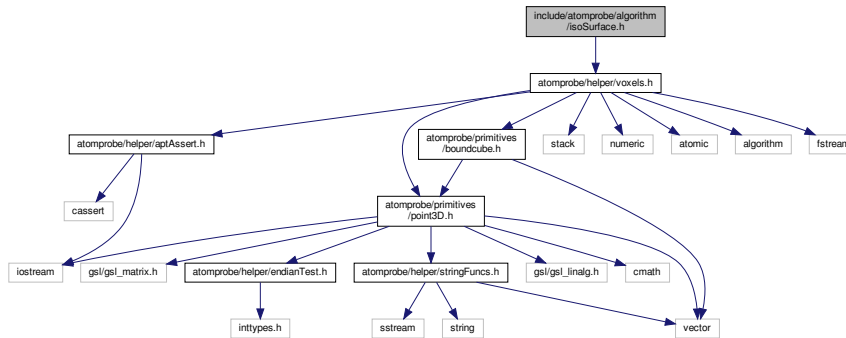
Increments a correlation histogram, as per correlationHistogram,.
- template<class T >

void [AtomProbe::linearHistogram](#) (const std::vector< T > &data, T start, T end, T step, std::vector< T > &histVals)

7.23 include/atomprobe/algorithm/isoSurface.h File Reference

```
#include "atomprobe/helper/voxels.h"
```

Include dependency graph for isoSurface.h:



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



Classes

- class [AtomProbe::TriangleWithVertexNorm](#)
- struct [AtomProbe::TriangleWithIndexedVertices](#)

Namespaces

- [AtomProbe](#)

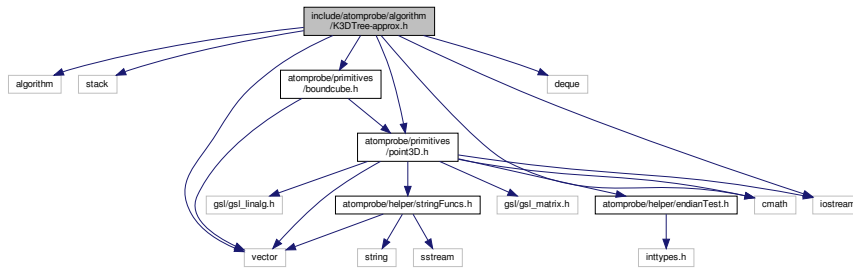
Functions

- void [AtomProbe::marchingCubes](#) (const Voxels< float > &v, float isoValue, std::vector< TriangleWithVertexNorm > &tVec)

7.24 include/atomprobe/algorithm/K3DTree-approx.h File Reference

```
#include <algorithm>
#include <stack>
#include <vector>
#include <deque>
#include <cmath>
#include <iostream>
#include "atomprobe/primitives/point3D.h"
```

```
#include "atomprobe/primitives/boundcube.h"  
Include dependency graph for K3DTree-approx.h:
```



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



Classes

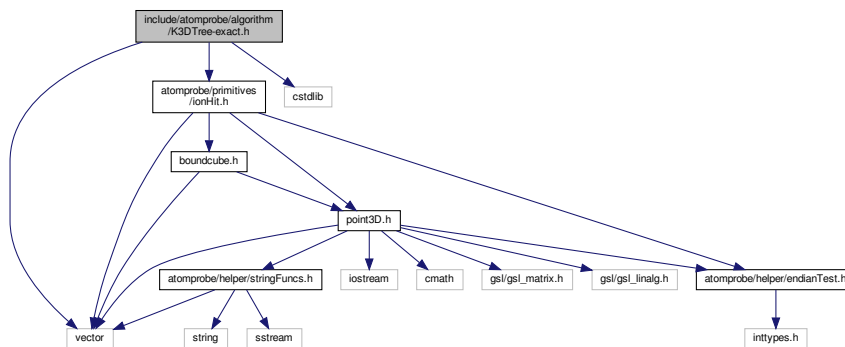
- class [AtomProbe::AxisCompare](#)
Functor allowing for sorting of points in 3D.
- class [AtomProbe::K3DNodeApprox](#)
Node Class for storing point.
- class [AtomProbe::K3DTreeApprox](#)
3D specific KD tree

Namespaces

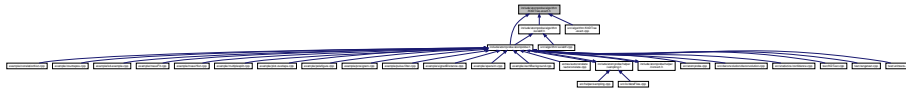
- [AtomProbe](#)

7.25 include/atomprobe/algorithm/K3DTree-exact.h File Reference

```
#include <vector>  
#include <cstdlib>  
#include "atomprobe/primitives/ionHit.h"  
Include dependency graph for K3DTree-exact.h:
```



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



Classes

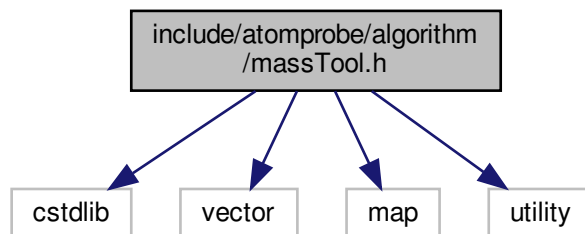
- class [AtomProbe::K3DNodeExact](#)
Node Class for storing point.
- class [AtomProbe::K3DTreeExact](#)
3D specific KD tree

Namespaces

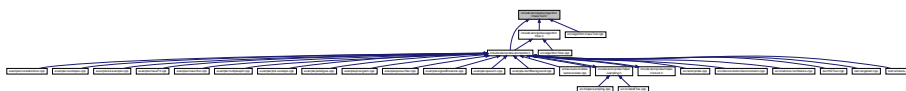
- [AtomProbe](#)

7.26 include/atomprobe/algorithm/massTool.h File Reference

```
#include <cstdlib>
#include <vector>
#include <map>
#include <utility>
Include dependency graph for massTool.h:
```



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



Classes

- class [AtomProbe::Weight](#)
Placeholder class for containing input weights for [MassTool](#).
- class [AtomProbe::MassTool](#)
Class that brute-force solves the knapsack problem.

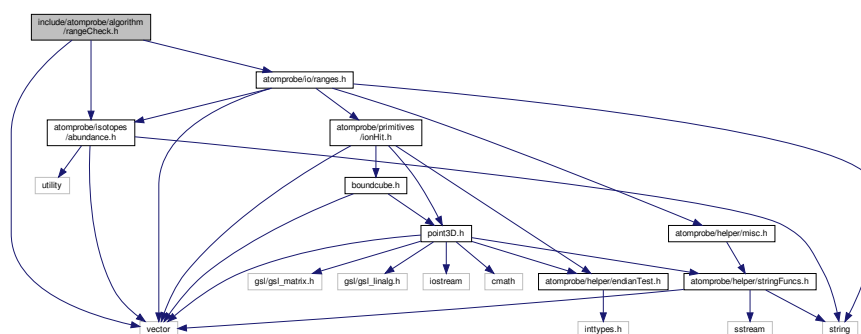
Namespaces

- [AtomProbe](#)

7.27 include/atomprobe/algorithm/rangeCheck.h File Reference

```
#include "atomprobe/isotopes/abundance.h"
#include "atomprobe/io/ranges.h"
#include <vector>
```

Include dependency graph for rangeCheck.h:



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



Namespaces

- [AtomProbe](#)

Functions

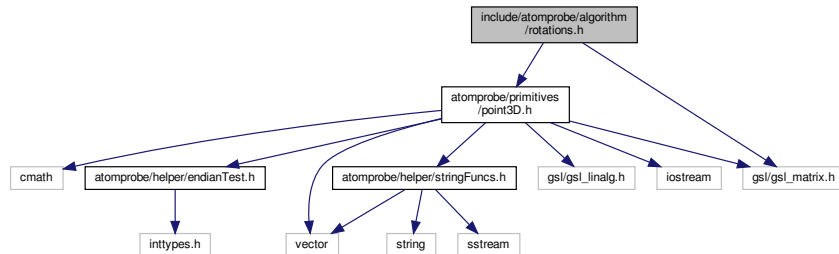
- void [AtomProbe::checkMassRangingCorrectness](#) (const [RangeFile](#) &rng, [AbundanceData](#) &massTable, float massTolerance, unsigned int maxChargeState, unsigned int maxComponents, std::vector< bool > &bad← Ranges)
Ensure that each mass given spans a peak that should exist.

7.28 include/atomprobe/algorithm/rotations.h File Reference

```
#include "atomprobe/primitives/point3D.h"
```

```
#include <gsl/gsl_matrix.h>
```

Include dependency graph for rotations.h:



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



Namespaces

- [AtomProbe](#)

Functions

- void [AtomProbe::computeRotationMatrix](#) (const Point3D &ur1, const Point3D &ur2, const Point3D &r1, const Point3D &r2, gsl_matrix *m)

7.29 include/atomprobe/atomprobe.h File Reference

```
#include "io/ranges.h"
#include "io/multiRange.h"
#include "io/dataFiles.h"
#include "algorithm/K3DTree-approx.h"
#include "algorithm/K3DTree-exact.h"
#include "algorithm/rotations.h"
#include "algorithm/convexHull.h"
#include "algorithm/axialdf.h"
#include "algorithm/histogram.h"
#include "algorithm/massTool.h"
#include "algorithm/filter.h"
#include "algorithm/rangeCheck.h"
#include "algorithm/componentAnalysis.h"
#include "helper/progress.h"
```

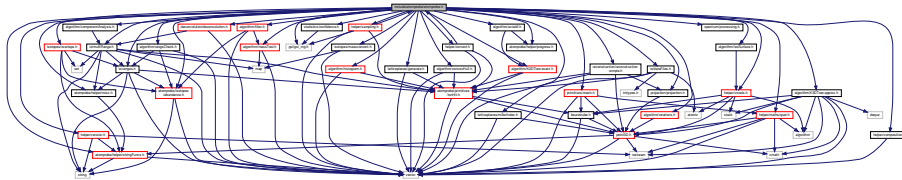


```

#include "helper/misc.h"
#include "helper/stringFuncs.h"
#include "helper/sampling.h"
#include "helper/composition.h"
#include "helper/convert.h"
#include "helper/maths/quat.h"
#include "helper/version.h"
#include "isotopes/abundance.h"
#include "isotopes/overlaps.h"
#include "isotopes/massconvert.h"
#include "projection/projection.h"
#include "latticeplanes/millerIndex.h"
#include "latticeplanes/generate.h"
#include "reconstruction/reconstruction-simple.h"
#include "spectrum/processing.h"
#include "primitives/mesh.h"
#include "statistics/confidence.h"
#include "deconvolution/deconvolution.h"
#include "helper/voxels.h"
#include "algorithm/isoSurface.h"
#include <gsl/gsl_rng.h>

```

Include dependency graph for atomprobe.h:



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



Classes

- class [AtomProbe::RandNumGen](#)

Namespaces

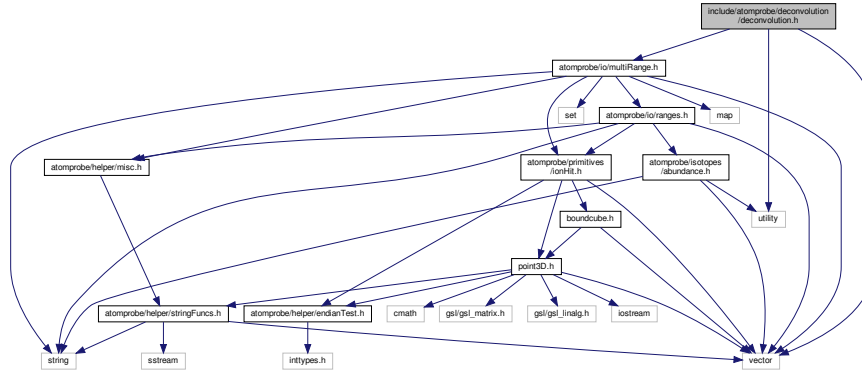
- [AtomProbe](#)

Variables

- LibVersion [AtomProbe::libVersion](#)
- RandNumGen [AtomProbe::randGen](#)

7.30 include/atomprobe/deconvolution/deconvolution.h File Reference

```
#include <vector>
#include <utility>
#include "atomprobe/io/multiRange.h"
Include dependency graph for deconvolution.h:
```



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

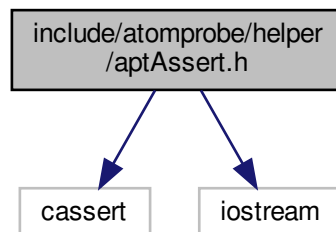


Namespaces

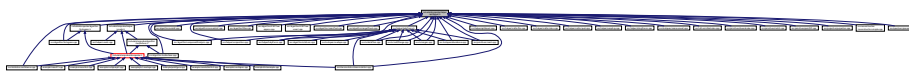
- [AtomProbe](#)

7.31 include/atomprobe/helper/aptAssert.h File Reference

```
#include <cassert>
#include <iostream>
Include dependency graph for aptAssert.h:
```



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



Namespaces

- [AtomProbe](#)

Macros

- `#define ASSERT(f)`
- `#define WARN(f, g)`
- `#define TEST(f, g)`

Functions

- void [AtomProbe::setHardAssert](#) (bool enabled)
Do assertions cause a straight up crash (enabled), or write a message to stderr (disabled)? By default, hard crash.
- void [AtomProbe::askAssert](#) (const char *, unsigned int)
Either abort program, or ask the user what to do for an assertion. depending on hardAssert setting.

7.31.1 Macro Definition Documentation

7.31.1.1 ASSERT

```
#define ASSERT(  
    f )
```

Definition at line 47 of file aptAssert.h.

7.31.1.2 TEST

```
#define TEST(  
    f,  
    g )
```

Definition at line 49 of file aptAssert.h.

Referenced by [AtomProbe::MassTool::bruteKnapsack\(\)](#), [AtomProbe::checkMassRangingCorrectness\(\)](#), [AtomProbe::K3DTreeExact::clearAllTags\(\)](#), [AtomProbe::computeRangeAdjacency\(\)](#), [AtomProbe::computeRotationMatrix\(\)](#), [AtomProbe::MultiRange::copyDataFromRange\(\)](#), [AtomProbe::countDataDistanceWeight\(\)](#), [AtomProbe::TRIANGLE::edgesMismatch\(\)](#), [AtomProbe::K3DTreeApprox::findKNearest\(\)](#), [AtomProbe::findOverlaps\(\)](#), [AtomProbe::BodyCentredCubicGen::generateLattice\(\)](#), [AtomProbe::AbundanceData::getNearestCharge\(\)](#), [AtomProbe::RangeFile::guessChargeState\(\)](#), [main\(\)](#), [AtomProbe::marchingCubes\(\)](#), [AtomProbe::maxExplainedFraction\(\)](#), [AtomProbe::MILLER_TRIPLET::operator==\(\)](#), [AtomProbe::EPOS_ENTRY::operator==\(\)](#), [AtomProbe::parseCompositionData\(\)](#), [AtomProbe::poissonConfidenceObservation\(\)](#), [AtomProbe::ReconstructionSphereOnCone::reconstruct\(\)](#), [runTests\(\)](#), [AtomProbe::sampleIons\(\)](#), and [AtomProbe::ModifiedFocusSphericProjection::scaleUp\(\)](#).

7.31.1.3 WARN

```
#define WARN(
    f,
    g )
```

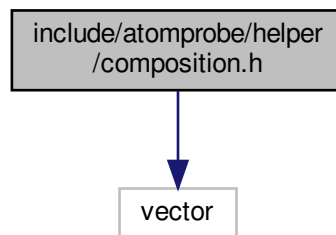
Definition at line 48 of file aptAssert.h.

Referenced by `AtomProbe::computeRangeAdjacency()`, and `AtomProbe::leastSquaresDeconvolve()`.

7.32 include/atomprobe/helper/composition.h File Reference

```
#include <vector>
```

Include dependency graph for composition.h:



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



Namespaces

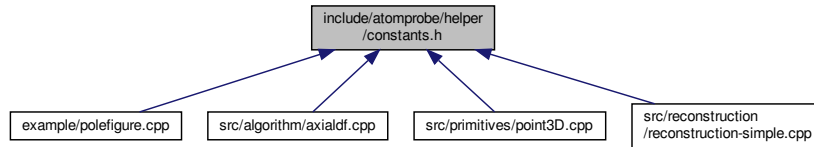
- [AtomProbe](#)

Functions

- void [AtomProbe::computeComposition](#) (const std::vector< unsigned int > &countData, std::vector< float > &compositionData)

7.33 include/atomprobe/helper/constants.h File Reference

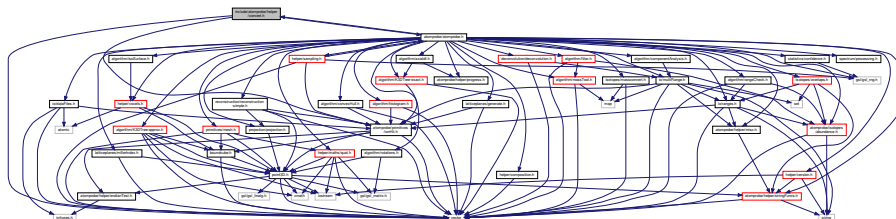
This graph shows which files directly or indirectly include this file:



7.34 include/atomprobe/helper/convert.h File Reference

```
#include <vector>
#include "atomprobe/primitives/ionHit.h"
#include "atomprobe/atomprobe.h"
```

Include dependency graph for convert.h:



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



Namespaces

- [AtomProbe](#)

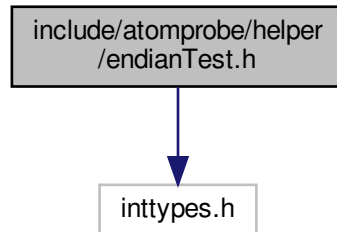
Functions

- void [AtomProbe::convertEPOSstoPos](#) (const std::vector< [EPOS_ENTRY](#) > &ePosEntry, std::vector< [IonHit](#) > &posFile)
 Convert an incoming entry of EPOS files to pos, and the append this to the pos vector given.

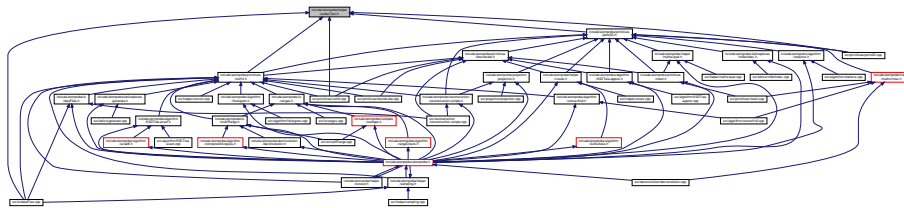
7.35 include/atomprobe/helper/endiannessTest.h File Reference

```
#include <inttypes.h>
```

Include dependency graph for endiannessTest.h:



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



Functions

- int [is_bigendian](#) ()
- int [is_littleendian](#) ()
- void [floatSwapBytes](#) (float *inFloat)
- void [int4SwapBytes](#) (int32_t *inInt)

Variables

- const int [ENDIAN_TEST](#) =1

7.35.1 Function Documentation

7.35.1.1 floatSwapBytes()

```
void floatSwapBytes (
    float * inFloat ) [inline]
```

Definition at line 58 of file endianTest.h.

Referenced by AtomProbe::loadAtoFile(), AtomProbe::readEposRecord(), AtomProbe::savePosFile(), AtomProbe::saveTapsimBin(), AtomProbe::IonHit::setPos(), and AtomProbe::Point3D::transform3x3().

7.35.1.2 int4SwapBytes()

```
void int4SwapBytes (
    int32_t * inInt ) [inline]
```

Definition at line 76 of file endianTest.h.

Referenced by AtomProbe::readEposRecord().

7.35.1.3 is_bigendian()

```
int is_bigendian ( ) [inline]
```

Definition at line 52 of file endianTest.h.

References ENDIAN_TEST.

7.35.1.4 is_littleendian()

```
int is_littleendian ( ) [inline]
```

Definition at line 54 of file endianTest.h.

References ENDIAN_TEST.

7.35.2 Variable Documentation

7.35.2.1 ENDIAN_TEST

```
const int ENDIAN_TEST =1
```

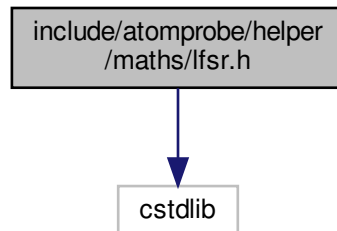
Definition at line 50 of file endianTest.h.

Referenced by is_bigendian(), and is_littleendian().

7.36 include/atomprobe/helper/maths/lfsr.h File Reference

```
#include <cstdlib>
```

Include dependency graph for lfsr.h:



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



Classes

- class [AtomProbe::LinearFeedbackShiftReg](#)

This class implements a Linear Feedback Shift Register (in software)

Namespaces

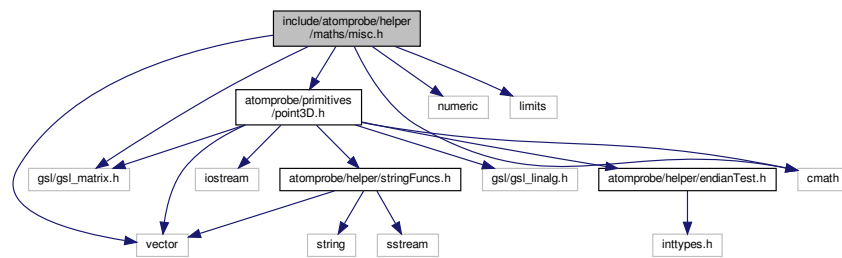
- [AtomProbe](#)

7.37 include/atomprobe/helper/maths/misc.h File Reference

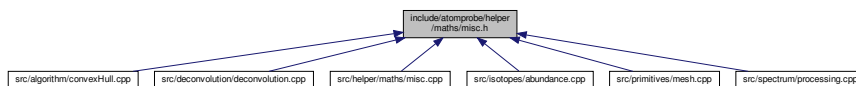
```
#include "gsl/gsl_matrix.h"
#include <numeric>
#include <vector>
#include <limits>
#include <cmath>
```



```
#include "atomprobe/primitives/point3D.h"
Include dependency graph for misc.h:
```



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



Namespaces

- [AtomProbe](#)

Enumerations

- enum [AtomProbe::PointDir](#) { [AtomProbe::POINTDIR_TOGETHER](#) =0, [AtomProbe::POINTDIR_IN_COMMON](#), [AtomProbe::POINTDIR_APART](#) }

Functions

- unsigned int [AtomProbe::estimateRank](#) (const [gsl_matrix](#) *m, float tolerance=sqrt(std::numeric_limits< float >::epsilon()))
Estimate the rank of the given matrix.
- bool [AtomProbe::solveLeastSquares](#) ([gsl_matrix](#) *m, [gsl_vector](#) *b, [gsl_vector](#) *&x)
Use an SVD based least-squares solver to solve $Mx=b$ (for x).
- template<class T >
T [AtomProbe::weightedMean](#) (const std::vector< T > &values, const std::vector< T > &weight)
- template<typename T >
void [AtomProbe::meanAndStdev](#) (const std::vector< T > &f, float &meanVal, float &stdevVal, bool normal← Correction=true)
- unsigned int [AtomProbe::vectorPointDir](#) (const [Point3D](#) &pA, const [Point3D](#) &pB, const [Point3D](#) &vC, const [Point3D](#) &vD)
Check which way vectors attached to two 3D points "point",.
- float [AtomProbe::distanceToSegment](#) (const [Point3D](#) &fA, const [Point3D](#) &fB, const [Point3D](#) &p)
- float [AtomProbe::signedDistanceToFacet](#) (const [Point3D](#) &fA, const [Point3D](#) &fB, const [Point3D](#) &fC, const [Point3D](#) &normal, const [Point3D](#) &p)

Find the distance between a point, and a triangular facet – may be positive or negative.

- float [AtomProbe::distanceToFacet](#) (const Point3D &fA, const Point3D &fB, const Point3D &fC, const Point3D &normal, const Point3D &p)
- float [AtomProbe::dotProduct](#) (float a1, float a2, float a3, float b1, float b2, float b3)

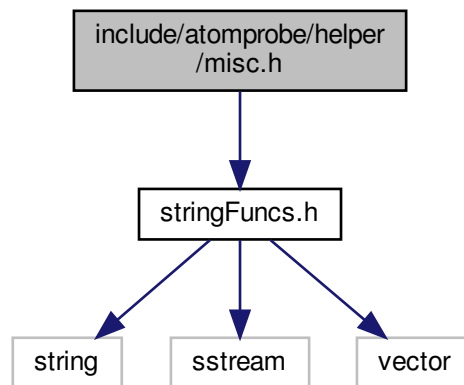
Inline func for calculating a(dot)b.

- double [AtomProbe::det3by3](#) (const double *ptArray)
- double [AtomProbe::pyramidVol](#) (const Point3D *planarPts, const Point3D &apex)
- bool [AtomProbe::trilsDegenerate](#) (const Point3D &fA, const Point3D &fB, const Point3D &fC)

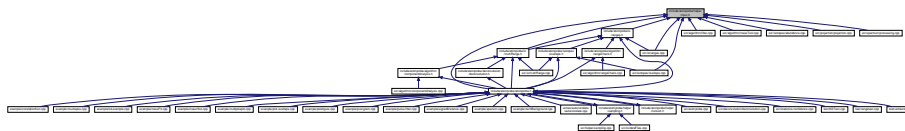
7.38 include/atomprobe/helper/misc.h File Reference

```
#include "stringFuncs.h"
```

Include dependency graph for misc.h:



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



Classes

- class [AtomProbe::RGBf](#)
Data holder for colour as float.

Namespaces

- [AtomProbe](#)

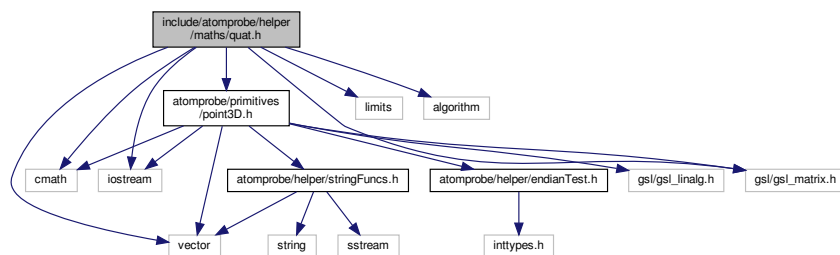
Functions

- template<class T >
bool [AtomProbe::tolEqual](#) (const T &a, const T &b, const T &f)
- template<class T >
void [AtomProbe::selectElements](#) (const std::vector< T > &in, const std::vector< unsigned int > &indices, std::vector< T > &out)
- template<class T >
void [AtomProbe::transposeVector](#) (std::vector< std::vector< T > > &v)
- template<class T >
void [AtomProbe::vectorMultiErase](#) (std::vector< T > &vec, const std::vector< bool > &wantKill)

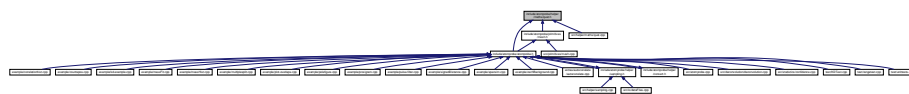
7.39 include/atomprobe/helper/maths/quat.h File Reference

```
#include <cmath>
#include <limits>
#include <iostream>
#include <vector>
#include <algorithm>
#include <gsl/gsl_matrix.h>
#include "atomprobe/primitives/point3D.h"
```

Include dependency graph for quat.h:



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



Classes

- struct [AtomProbe::Quaternion](#)
Data storage structure for quaternions.
- struct [AtomProbe::Point3f](#)
Data storage structure for points.

Namespaces

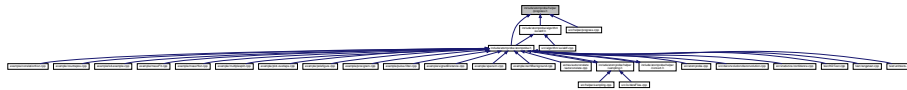
- [AtomProbe](#)

Functions

- void [AtomProbe::quat_rot](#) (Point3D &p, const Point3D &r, float angle)
Rotate a point around a given rotation axis by a specified angle.
- void [AtomProbe::quat_rot](#) (Point3f *point, const Point3f *rotVec, float angle)
Rotate a point around a given vector, with specified angle.
- void [AtomProbe::quat_rot_array](#) (Point3f *point, unsigned int n, const Point3f *rotVec, float angle)
Rotate each point in array of size n around a given vector, with specified angle.
- void [AtomProbe::quat_rot_array](#) (Point3D *point, unsigned int n, const Point3f *rotVec, float angle)
Rotate each point in array of size n around a given vector, with specified angle.
- void [AtomProbe::quat_get_rot_quat](#) (const Point3f *rotVec, float angle, Quaternion *rotQuat)
Compute the quaternion for specified rotation.
- void [AtomProbe::quat_rot_apply_quat](#) (Point3f *point, const Quaternion *rotQuat)
Use previously generated quats from quat_get_rot_quats to rotate a point.
- void [AtomProbe::applyQuaternionRotation](#) (std::vector< Point3D > &pts, const Quaternion &q)
Apply the given quaternion rotation to the input points.

7.40 include/atomprobe/helper/progress.h File Reference

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



Classes

- class [AtomProbe::ProgressBar](#)

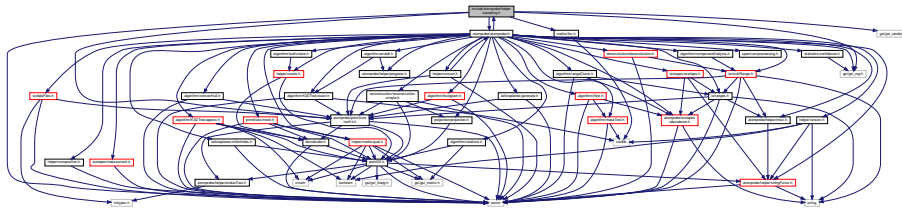
Namespaces

- [AtomProbe](#)

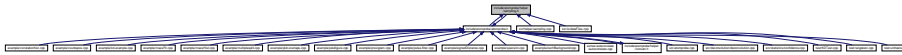
7.41 include/atomprobe/helper/sampling.h File Reference

```
#include <vector>
#include "atomprobe/primitives/ionHit.h"
#include "atomprobe/atomprobe.h"
#include "maths/lfsr.h"
#include <gsl/gsl_randist.h>
```

```
#include <gsl/gsl_rng.h>
Include dependency graph for sampling.h:
```



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



Namespaces

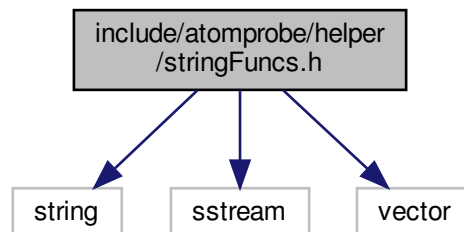
- [AtomProbe](#)

Functions

- void [AtomProbe::sampleIons](#) (const std::vector< IonHit > &ions, float sampleFactor, std::vector< IonHit > &sampled, bool strongRandom=true)
- template<class T >
size_t [AtomProbe::randomSelect](#) (std::vector< T > &result, const std::vector< T > &source, size_t num, gsl_rng *rng)
- template<class T >
void [AtomProbe::randomIndices](#) (std::vector< T > &res, size_t num, size_t nMax, gsl_rng *rng)

7.42 include/atomprobe/helper/stringFuncs.h File Reference

```
#include <string>
#include <sstream>
#include <vector>
Include dependency graph for stringFuncs.h:
```



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



Namespaces

- [AtomProbe](#)

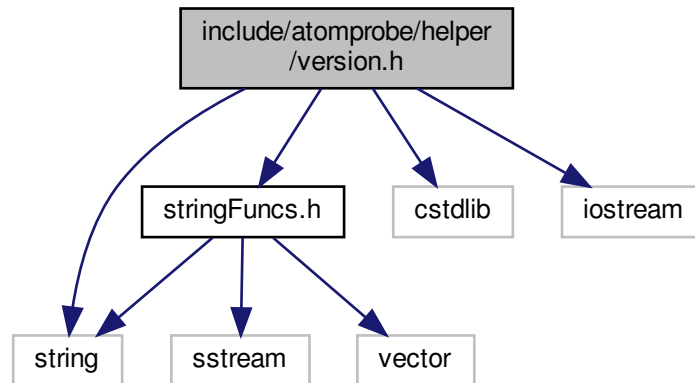
Functions

- `template<class T1 , class T2 >`
`bool AtomProbe::stream_cast (T1 &result, const T2 &obj)`
Template function to cast and object to another by the stringstream.
- `template<class T >`
`void AtomProbe::strAppend (std::string s, const T &a)`
- `void AtomProbe::genColString (unsigned char r, unsigned char g, unsigned char b, unsigned char a, std::string &s)`
- `void AtomProbe::genColString (unsigned char r, unsigned char g, unsigned char b, std::string &s)`
- `bool AtomProbe::parseColString (const std::string &str, unsigned char &r, unsigned char &g, unsigned char &b, unsigned char &a)`
Parse a colour string containing rgb[a]; hex for with leading #.
- `std::string AtomProbe::digitString (unsigned int thisDigit, unsigned int maxDigit)`
Generate a string with leading digits up to maxDigit (eg, if maxDigit is 424, and thisDigit is 1.
- `std::string AtomProbe::stripWhite (const std::string &str)`
Strip whitespace, (eg tab,space) from either side of a string.
- `std::string AtomProbe::stripChars (const std::string &Str, const char *chars)`
- `std::string AtomProbe::lowercase (std::string s)`
Return a lowercase version for a given string.
- `std::string AtomProbe::uppercase (std::string s)`
Return a uppercase version for a given string.
- `void AtomProbe::stripZeroEntries (std::vector< std::string > &s)`
- `void AtomProbe::splitStrsRef (const char *cpStr, const char delim, std::vector< std::string > &v)`
Split string references using a single delimiter.
- `void AtomProbe::splitStrsRef (const char *cpStr, const char *delim, std::vector< std::string > &v)`
Split string references using any of a given string of delimiters.
- `std::string AtomProbe::onlyFilename (const std::string &path)`
Return only the filename component.
- `std::string AtomProbe::onlyDir (const std::string &path)`
Return only the directory name component of the full path.
- `std::string AtomProbe::convertFileStringToNative (const std::string &s)`
Convert a path format into a native path from unix format.
- `std::string AtomProbe::convertFileStringToCanonical (const std::string &s)`
Convert a path format into a unix path from native format.
- `std::string AtomProbe::tabs (unsigned int nTabs)`
- `std::string AtomProbe::stlWStrToStlStr (const std::wstring &s)`
- `std::wstring AtomProbe::stlStrToStlWStr (const std::string &s)`
- `void AtomProbe::nullifyMarker (char *buffer, char marker)`

7.43 include/atomprobe/helper/version.h File Reference

```
#include "stringFuncs.h"
#include <string>
#include <cstdlib>
#include <iostream>
```

Include dependency graph for version.h:



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



Classes

- class [AtomProbe::LibVersion](#)
Class to hold the library version data.

Namespaces

- [AtomProbe](#)

Macros

- `#define` [LIBATOMPROBE_MAJOR](#) 0
- `#define` [LIBATOMPROBE_MINOR](#) 0
- `#define` [LIBATOMPROBE_REVISION](#) 1

7.43.1 Macro Definition Documentation

7.43.1.1 LIBATOMPROBE_MAJOR

```
#define LIBATOMPROBE_MAJOR 0
```

Definition at line 7 of file version.h.

Referenced by `AtomProbe::LibVersion::getMajor()`, and `AtomProbe::LibVersion::LibVersion()`.

7.43.1.2 LIBATOMPROBE_MINOR

```
#define LIBATOMPROBE_MINOR 0
```

Definition at line 8 of file version.h.

Referenced by `AtomProbe::LibVersion::getMinor()`, and `AtomProbe::LibVersion::LibVersion()`.

7.43.1.3 LIBATOMPROBE_REVISION

```
#define LIBATOMPROBE_REVISION 1
```

Definition at line 9 of file version.h.

Referenced by `AtomProbe::LibVersion::getRevision()`, and `AtomProbe::LibVersion::LibVersion()`.

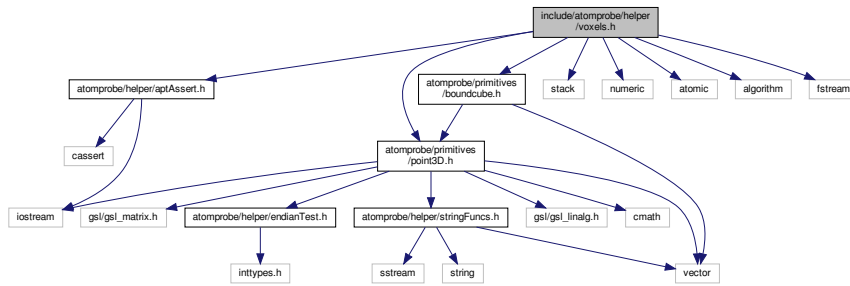
7.44 include/atomprobe/helper/voxels.h File Reference

```
#include "atomprobe/helper/aptAssert.h"  
#include "atomprobe/primitives/boundcube.h"  
#include "atomprobe/primitives/point3D.h"  
#include <stack>  
#include <numeric>  
#include <atomic>  
#include <algorithm>
```



```
#include <fstream>
```

Include dependency graph for voxels.h:



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



Classes

- class [AtomProbe::Voxels< T >](#)
Template class that stores 3D voxel data.

Namespaces

- [AtomProbe](#)

Macros

- `#define XOR(a, b) ((!(a)) ^ (!(b)))`

Enumerations

- enum { [AtomProbe::BOUND_CLIP =1](#), [AtomProbe::BOUND_HOLD](#), [AtomProbe::BOUND_DERIV_HOLD](#), [AtomProbe::BOUND_MIRROR](#), [AtomProbe::BOUND_ZERO](#), [AtomProbe::BOUND_ENUM_END](#) }
Boundary clipping mode.
- enum { [AtomProbe::VOX_INTERP_NONE](#), [AtomProbe::VOX_INTERP_LINEAR](#), [AtomProbe::VOX_INTERP_ENUM_END](#) }
- enum { [AtomProbe::ADJ_ALL](#), [AtomProbe::ADJ_PLUS](#) }
- enum { [AtomProbe::VOXELS_BAD_FILE_READ =1](#), [AtomProbe::VOXELS_BAD_FILE_OPEN](#), [AtomProbe::VOXELS_BAD_FILE_SIZE](#), [AtomProbe::VOXELS_OUT_OF_MEMORY](#) }
- enum { [AtomProbe::CLIP_NONE =0](#), [AtomProbe::CLIP_LOWER_SOUTH_WEST](#), [AtomProbe::CLIP_UPPER_NORTH_EAST](#) }
Clipping direction constants.
- enum { [AtomProbe::VOXEL_ABORT_ERR](#), [AtomProbe::VOXEL_MEMORY_ERR](#), [AtomProbe::VOXEL_BOUNDS_INVALID_ERR](#) }

Functions

- `template<class T, class U >`
void `AtomProbe::castVoxels` (const Voxels< T > &src, Voxels< U > &dest)
Convert one type of voxel into another by assignment operator.
- `template<class T, class U >`
void `AtomProbe::sumVoxels` (const Voxels< T > &src, U &counter)
Use one counting type to sum counts in a voxel of given type.

7.44.1 Macro Definition Documentation

7.44.1.1 XOR

```
#define XOR(  
    a,  
    b ) ((!(a)) ^ !(b))
```

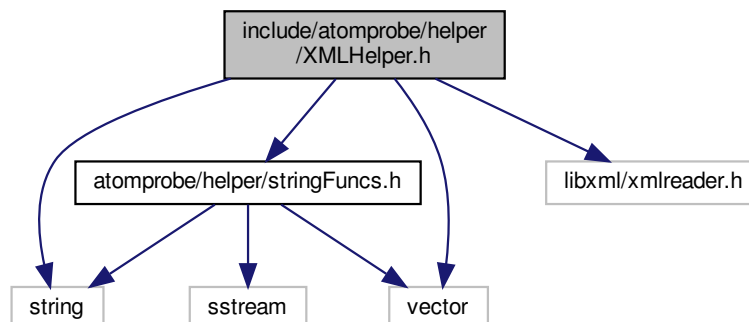
Definition at line 38 of file voxels.h.

Referenced by `AtomProbe::RangeFile::rangeInvertable()`, and `AtomProbe::Voxels< T >::thresholdToBoolMask()`.

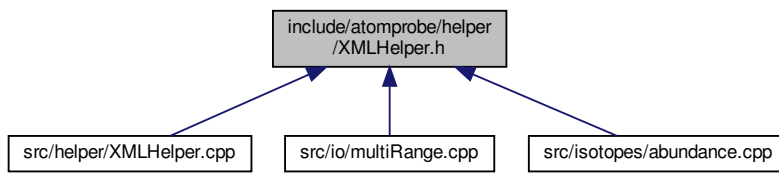
7.45 include/atomprobe/helper/XMLHelper.h File Reference

```
#include "atomprobe/helper/stringFuncs.h"  
#include <libxml/xmlreader.h>  
#include <string>  
#include <vector>
```

Include dependency graph for XMLHelper.h:



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



Namespaces

- [AtomProbe](#)

Enumerations

- enum { [AtomProbe::PROP_PARSE_ERR](#) = 1, [AtomProbe::PROP_BAD_ATT](#) }

Functions

- unsigned int [AtomProbe::XMLHelpNextType](#) (xmlNodePtr &node, int)
- unsigned int [AtomProbe::XMLHelpFwdToElem](#) (xmlNodePtr &node, const char *nodeName)
- unsigned int [AtomProbe::XMLHelpFwdNotElem](#) (xmlNodePtr &node, const char *nodeName)
- unsigned int [AtomProbe::XMLHelpFwdToList](#) (xmlNodePtr &node, const std::vector< std::string > &nodeList)
- std::string [AtomProbe::XMLHelpGetText](#) (xmlNodePtr &node)
- void [AtomProbe::XMLFreeDoc](#) (void *data)

Free a xmlDoc pointer. For use in conjunction with std::unique_ptr for auto-deallocation.

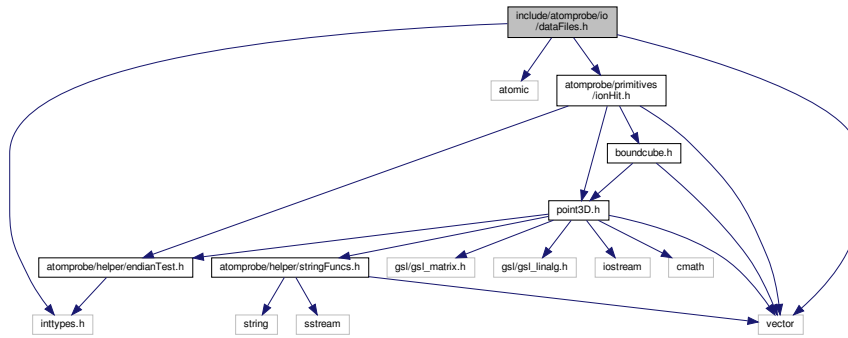
- template<class T >
unsigned int [AtomProbe::XMLHelpGetProp](#) (T &prop, xmlNodePtr node, std::string propName)
- template<class T >
bool [AtomProbe::XMLGetNextElemAttrib](#) (xmlNodePtr &nodePtr, T &v, const char *nodeName, const char *attrib)

Grab the specified attribute from the next element, then [stream_cast\(\)](#) it into the passed-in object. Returns true on success, false on error.

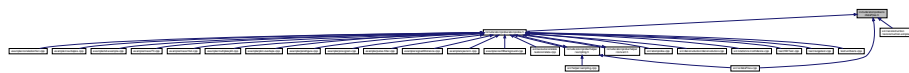
7.46 include/atomprobe/io/dataFiles.h File Reference

```
#include <vector>
#include <atomic>
#include <inttypes.h>
```

```
#include "atomprobe/primitives/ionHit.h"
Include dependency graph for dataFiles.h:
```



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



Classes

- struct [AtomProbe::ATO_ENTRY](#)
- struct [AtomProbe::SINGLE_HIT](#)
Single 3DAP hit event.
- struct [AtomProbe::VOLTAGE_DATA](#)
Voltage data structure – these updates occur periodically during the experiment.
- struct [AtomProbe::THREEDAP_DATA](#)
experimental setup data
- struct [AtomProbe::THREEDAP_EXPERIMENT](#)
Data structure that contains the experiment information present in a 3Dap file.

Namespaces

- [AtomProbe](#)

Typedefs

- typedef struct [AtomProbe::ATO_ENTRY](#) [AtomProbe::ATO_ENTRY](#)

Enumerations

- enum {
AtomProbe::RECORDREAD_ERR_GET_FILESIZE =1, AtomProbe::RECORDREAD_ERR_FILESIZE_M←
ODULO, AtomProbe::RECORDREAD_ERR_FILE_OPEN, AtomProbe::RECORDREAD_ERR_NOMEM,
AtomProbe::RECORDREAD_BAD_FILEREAD, AtomProbe::RECORDREAD_ERR_CHUNKOFFSET,
AtomProbe::RECORDREAD_BAD_RECORD }
- enum {
AtomProbe::ATO_OPEN_FAIL =1, AtomProbe::ATO_EMPTY_FAIL, AtomProbe::ATO_SIZE_ERR, Atom←
Probe::ATO_VERSIONCHECK_ERR,
AtomProbe::ATO_MEM_ERR, AtomProbe::ATO_BAD_ENDIAN_DETECT, AtomProbe::ATO_ENUM_END }
- enum { AtomProbe::TEXT_ERR_FILE_OPEN =1, AtomProbe::TEXT_ERR_FILE_NUM_FIELDS, Atom←
Probe::TEXT_ERR_FILE_FORMAT }
- enum {
AtomProbe::OPSREADER_FORMAT_CLINE_ERR =1, AtomProbe::OPSREADER_FORMAT_DETECTO←
RLINE_ERR, AtomProbe::OPSREADER_FORMAT_LINE_DASH_ERR, AtomProbe::OPSREADER_FOR←
MAT_LINE_VOLTAGE_ERR,
AtomProbe::OPSREADER_FORMAT_LINE_VOLTAGE_NOBETA, AtomProbe::OPSREADER_FORMAT←
_LINE_VOLTAGE_DATA_ERR, AtomProbe::OPSREADER_FORMAT_LINE_VOLTAGE_DATACOUNT←
ERR, AtomProbe::OPSREADER_FORMAT_LINETYPE_ERR,
AtomProbe::OPSREADER_FORMAT_CHANNELS_ERR, AtomProbe::OPSREADER_CHANNELS_DAT←
A_ERR, AtomProbe::OPSREADER_FORMAT_SLINE_EVENTCOUNT_ERR, AtomProbe::OPSREADER←
_FORMAT_SLINE_EVENTDATA_ERR,
AtomProbe::OPSREADER_FORMAT_SLINE_FORMAT_ERR, AtomProbe::OPSREADER_FORMAT_SL←
INE_PREFIX_ERR, AtomProbe::OPSREADER_FORMAT_DUPLICATE_SYSDATA, AtomProbe::OPSRE←
ADER_FORMAT_DUPLICATE_DETECTORSIZE,
AtomProbe::OPSREADER_FORMAT_TRAILING_DASH_ERR, AtomProbe::OPSREADER_FORMAT_D←
OUBLEDASH, AtomProbe::OPSREADER_OPEN_ERR, AtomProbe::OPSREADER_READ_ERR,
AtomProbe::OPSREADER_ABORT_ERR, AtomProbe::OPSREADER_ENUM_END }

Error codes for OPS file loading.

Functions

- const char * AtomProbe::getRecordReadErrString (unsigned int errCode)
- const char * AtomProbe::getAtoErrString (unsigned int errCode)
- unsigned int AtomProbe::loadPosFile (std::vector< IonHit > &posIons, const char *posFile)
Load a pos file directly into a single ion list.
- unsigned int AtomProbe::loadPosFile (std::vector< IonHit > &posIons, const char *posFile, unsigned int nSamplesMax)
As per loadPosFile, but with an additional setting for the maximum number of ions to load.
- const char * AtomProbe::getPosFileErrString (unsigned int errMesg)
- unsigned int AtomProbe::savePosFile (const std::vector< Point3D > &points, float mass, const char *name, bool append=false)
Save a vector of Point3Ds into a pos file, using a fixed mass, return nonzero on error.
- unsigned int AtomProbe::savePosFile (const std::vector< IonHit > &data, const char *name, bool append=false)
Save a vector of IonHits into a "pos" file, return nonzero on error.
- unsigned int AtomProbe::saveTapsimBin (std::ostream &f, std::vector< IonHit > &posIons)
Write a tapsim file from a bunch of IonHits.
- size_t AtomProbe::loadEposFile (std::vector< EPOS_ENTRY > &outData, const char *filename)
Load an entire "EPOS" File.
- size_t AtomProbe::chunkLoadEposFile (std::vector< EPOS_ENTRY > &outData, const char *filename, unsigned int chunkSize, unsigned int chunkOffset, unsigned int &nEntriesLeft)
Load an "EPOS" file, with a maximum chunk size.

- unsigned int [AtomProbe::loadTextData](#) (const char *cpFilename, std::vector< std::vector< float > > &data↔Vec, std::vector< std::string > &headerVec, const char *delim, bool allowNan=true)
Load a CSV, TSV or similar text file. Assumes "C" Locale for input (ie "." as decimal separator).
- unsigned int [AtomProbe::readPosapOps](#) (const char *file, THREEEDAP_EXPERIMENT &data, unsigned int &badLine, unsigned int &progress, std::atomic< bool > &wantAbort, unsigned int nDelayLines=2, bool strictMode=false)
Function to read POSAP "OPS" files.
- unsigned int [AtomProbe::loadATOFile](#) (const char *fileName, std::vector< ATO_ENTRY > &ions, unsigned int forceEndian=0)
Load a LAWATAP "ATO" file.

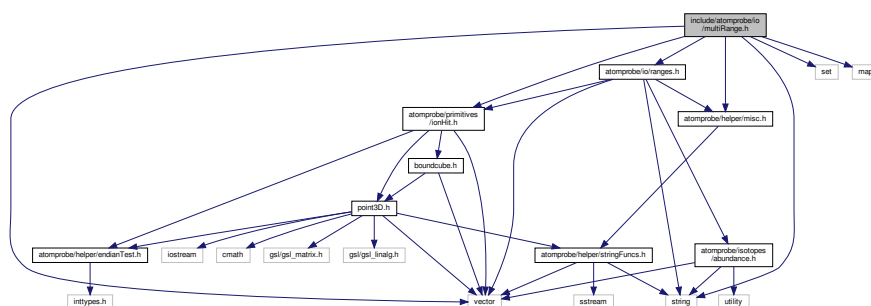
Variables

- const char * [AtomProbe::RECORDREAD_ERR_STRINGS](#) []
- const char * [AtomProbe::ATO_ERR_STRINGS](#) []
Human readable error messages for use with ATO reader return values.
- const char * [AtomProbe::OPS_ENUM_ERRSTRINGS](#) []

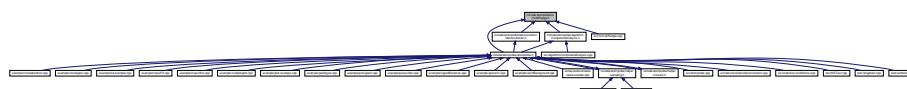
7.47 include/atomprobe/io/multiRange.h File Reference

```
#include <vector>
#include <set>
#include <string>
#include <map>
#include "atomprobe/io/ranges.h"
#include "atomprobe/helper/misc.h"
#include "atomprobe/primitives/ionHit.h"
```

Include dependency graph for multiRange.h:



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



Classes

- struct [AtomProbe::SIMPLE_SPECIES](#)
- struct [AtomProbe::FLATTENED_RANGE](#)
Structure that allows for the multirange data to be mapped into.
- class [AtomProbe::MultiRange](#)
Data storage and retrieval class for "ranging" a spectra, where overlapping ranges are permitted.

Namespaces

- [AtomProbe](#)

Enumerations

- enum { [AtomProbe::MULTIRANGE_FORMAT_XML](#) }

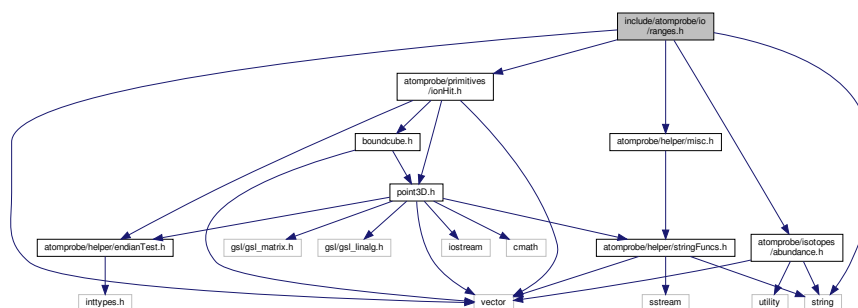
Functions

- bool [AtomProbe::operator<](#) (const SIMPLE_SPECIES &a, const SIMPLE_SPECIES &b)

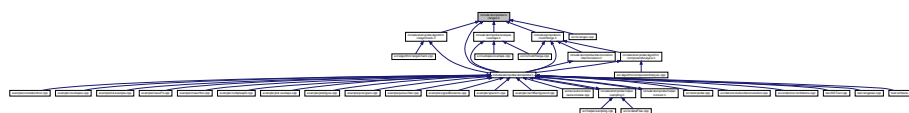
7.48 include/atomprobe/io/ranges.h File Reference

```
#include <vector>
#include <string>
#include "atomprobe/primitives/ionHit.h"
#include "atomprobe/isotopes/abundance.h"
#include "atomprobe/helper/misc.h"
```

Include dependency graph for ranges.h:



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



Classes

- class [AtomProbe::RangeFile](#)

Data storage and retrieval class for various range files.

Namespaces

- [AtomProbe](#)

Enumerations

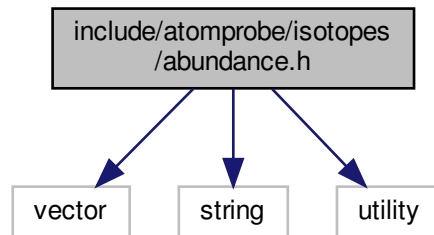
- enum {
[AtomProbe::RANGE_FORMAT_ORNL](#), [AtomProbe::RANGE_FORMAT_DBL_ORNL](#), [AtomProbe::RANGE_FORMAT_ENV](#), [AtomProbe::RANGE_FORMAT_RRNG](#),
[AtomProbe::RANGE_FORMAT_END_OF_ENUM](#) }
- enum {
[AtomProbe::RANGE_ERR_FORMAT =1](#), [AtomProbe::RANGE_ERR_FORMAT_HEADER](#), [AtomProbe::RANGE_ERR_FORMAT_EMPTY](#), [AtomProbe::RANGE_ERR_FORMAT_LONGNAME](#),
[AtomProbe::RANGE_ERR_FORMAT_SHORTNAME](#), [AtomProbe::RANGE_ERR_FORMAT_COLOUR](#),
[AtomProbe::RANGE_ERR_FORMAT_TABLESEPARATOR](#), [AtomProbe::RANGE_ERR_FORMAT_MASS_PAIR](#),
[AtomProbe::RANGE_ERR_FORMAT_TABLE_ENTRY](#), [AtomProbe::PARSE_RANGE_FORMAT_GENERIC_FAIL](#), [AtomProbe::RANGE_ERR_DATA_TOO_MANY_USELESS_RANGES](#), [AtomProbe::RANGE_ERR_DATA_FLIPPED](#),
[AtomProbe::RANGE_ERR_DATA_INCONSISTENT](#), [AtomProbe::RANGE_ERR_DATA_NOMAPPED_IONNAME](#), [AtomProbe::RANGE_ERR_OPEN](#), [AtomProbe::RANGE_ERR_FORMAT_RANGETABLE](#),
[AtomProbe::RANGE_ERR_FORMAT_TABLEHEADER_NUMIONS](#), [AtomProbe::RANGE_ERR_NONUNIQUE_POLYATOMIC](#), [AtomProbe::RANGE_ERR_TOO_LARGE](#), [AtomProbe::RANGE_ERR_DASHHEADER](#),
[AtomProbe::RANGE_ERR_IONBLOCK_CONTENT](#), [AtomProbe::RANGE_ERR_NUMIONS](#), [AtomProbe::RANGE_ERR_NUMIONS_DUPLICATED](#), [AtomProbe::RANGE_ERR_TOO_MANYIONS](#),
[AtomProbe::RANGE_ERR_ION_BLOCK_NOT_PRESENT](#), [AtomProbe::RANGE_ERR_DUPLICATE_NUMRANGES](#), [AtomProbe::RANGE_ERR_NUMRANGE_PARSE](#), [AtomProbe::RANGE_ERR_RRNG_IONS_TOO_SHORT](#),
[AtomProbe::RANGE_ERR_RRNG_COLON_SEPARATOR](#), [AtomProbe::RANGE_ERR_BADCOLOUR](#),
[AtomProbe::RANGE_ERR_ION_NOT_MAPPED](#), [AtomProbe::RANGE_ERR_BADSTART](#),
[AtomProbe::RANGE_ERR_BADEND](#), [AtomProbe::RANGE_ERR_NAME_EMPTY](#), [AtomProbe::RANGE_ERR_BAD_LINE_RANGEBLOCK](#), [AtomProbe::RANGE_ERR_MISSING_IONBLOCK](#),
[AtomProbe::RANGE_ERR_NO_RANGES](#), [AtomProbe::RANGE_ERR_NO_BASIC_IONS](#), [AtomProbe::RANGE_ERR_MISMATCHED_NUMRANGES](#), [AtomProbe::RANGE_ERR_RANGEBLOCK_FORMAT](#),
[AtomProbe::RANGE_ERR_BAD_MULTPLICITY](#), [AtomProbe::RANGE_ERR_FORMAT_EMPTY_RANGE](#),
[AtomProbe::RANGE_ERR_FILESIZE](#), [AtomProbe::RANGE_ERR_ENUM_END](#) }

Variables

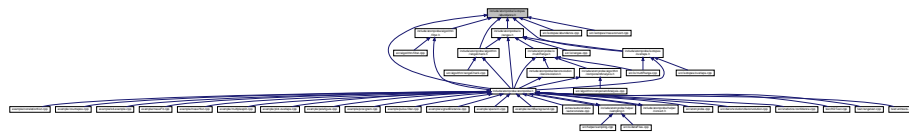
- const unsigned int [AtomProbe::NUM_ELEMENTS](#) =119

7.49 include/atomprobe/isotopes/abundance.h File Reference

```
#include <vector>
#include <string>
#include <utility>
Include dependency graph for abundance.h:
```



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



Classes

- struct [AtomProbe::ISOTOPE_ENTRY](#)
- class [AtomProbe::AbundanceData](#)

Class to load abundance information for natural isotopes.

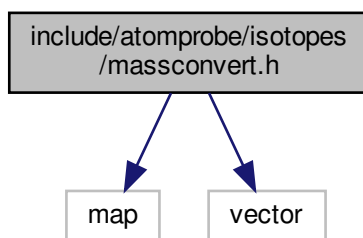
Namespaces

- [AtomProbe](#)

7.50 include/atomprobe/isotopes/massconvert.h File Reference

```
#include <map>
#include <vector>
```

Include dependency graph for massconvert.h:



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



Namespaces

- [AtomProbe](#)

Enumerations

- enum {
 - [AtomProbe::COMPPARSE_BAD_INPUT_FORMAT](#) =1, [AtomProbe::COMPPARSE_BAD_COMP_VALUE](#),
 - [AtomProbe::COMPPARSE_BAD_BAL_USAGE](#), [AtomProbe::COMPPARSE_BAD_SYMBOL](#),
 - [AtomProbe::COMPPARSE_DUPLICATE_SYMBOL](#), [AtomProbe::COMPPARSE_BAD_TOTAL_COMP](#) }*composition parsing error messages*

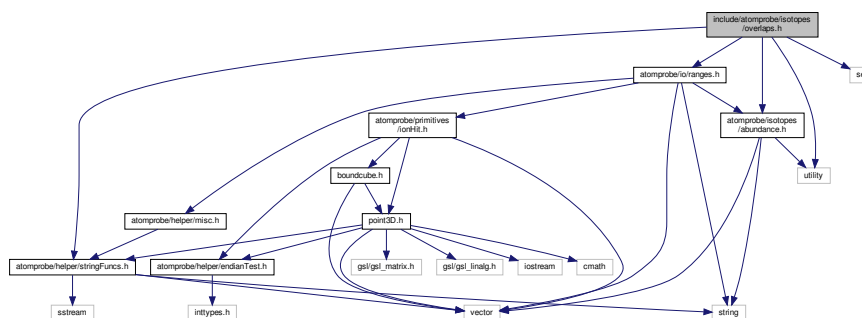
Functions

- unsigned int [AtomProbe::parseCompositionData](#) (const std::vector< std::string > &s, const AbundanceData &atomTable, std::map< unsigned int, double > &atomData)
 - Convert atomic string label data, in the form "Fe 0.81" to fraction map.*
- void [AtomProbe::convertMolToMass](#) (const AbundanceData &atomTable, const std::map< unsigned int, double > &compositionMols, std::map< unsigned int, double > &compositionMass)
 - Convert the given composition from molar fraction data to mass fraction.*
- void [AtomProbe::convertMassToMol](#) (const AbundanceData &atomTable, const std::map< unsigned int, double > &compositionMass, std::map< unsigned int, double > &compositionMols)
 - Convert the given composition from mass fraction data to molar fraction.*

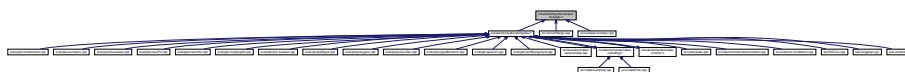
7.51 include/atomprobe/isotopes/overlaps.h File Reference

```
#include "atomprobe/isotopes/abundance.h"
#include "atomprobe/helper/stringFuncs.h"
#include "atomprobe/io/ranges.h"
#include <utility>
#include <set>
```

Include dependency graph for overlaps.h:



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



Namespaces

- [AtomProbe](#)

Functions

- void [AtomProbe::findOverlaps](#) (const AbundanceData &natData, const RangeFile &rng, float massDelta, unsigned int maxCharge, std::vector< std::pair< size_t, size_t > > &overlapIdx, std::vector< std::pair< float, float > > &overlapMasses)

Find the overlaps stemming from a given rangefile.

- void [AtomProbe::findOverlaps](#) (const AbundanceData &natData, const std::vector< std::string > &ions, float massDelta, unsigned int maxCharge, std::vector< std::pair< size_t, size_t > > &overlapIdx, std::vector< std::pair< float, float > > &overlapMasses)

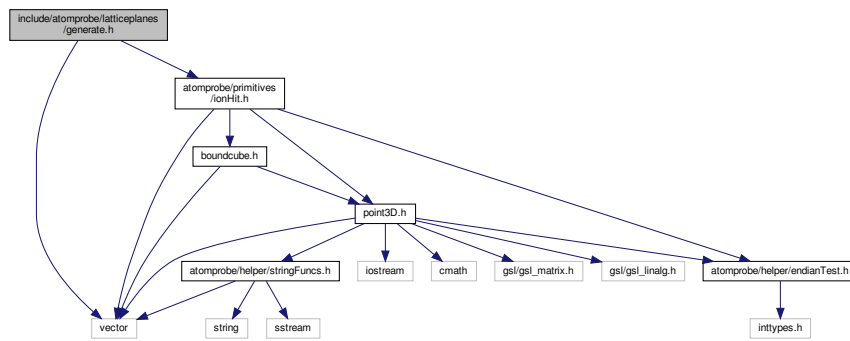
As per findOverlaps(...Rangefile ...) , but uses a vector of ions instead.

- void [AtomProbe::findOverlapsFromSpectra](#) (const std::vector< std::vector< std::pair< float, float > > > &massDistributions, float massDelta, const std::set< unsigned int > &skipIDs, std::vector< std::pair< size_t, size_t > > &overlapIdx, std::vector< std::pair< float, float > > &overlapMasses)

Find overlaps in the given mass distributions, as per findOverlaps(...Rangefile...)

7.52 include/atomprobe/latticeplanes/generate.h File Reference

```
#include <vector>
#include "atomprobe/primitives/ionHit.h"
Include dependency graph for generate.h:
```



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



Classes

- class [AtomProbe::CrystalGen](#)
- class [AtomProbe::SimpleCubicGen](#)
- class [AtomProbe::FaceCentredCubicGen](#)
- class [AtomProbe::BodyCentredCubicGen](#)

Namespaces

- [AtomProbe](#)

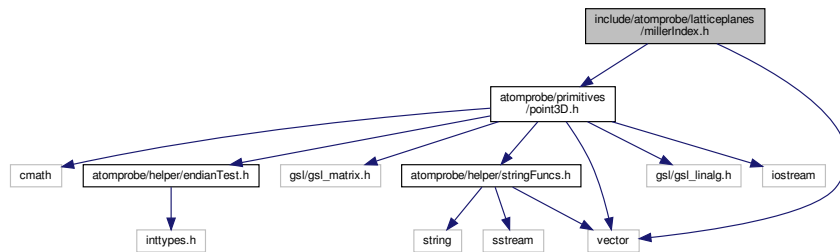
Enumerations

- enum { [AtomProbe::CRYSTAL_BAD_UNIT_CELL](#), [AtomProbe::CRYSTAL_ENUM_END](#) }

7.53 include/atomprobe/latticeplanes/millerIndex.h File Reference

```
#include "atomprobe/primitives/point3D.h"
#include <vector>
```

Include dependency graph for millerIndex.h:



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



Classes

- class [AtomProbe::MILLER_TRIPLET](#)

Namespaces

- [AtomProbe](#)

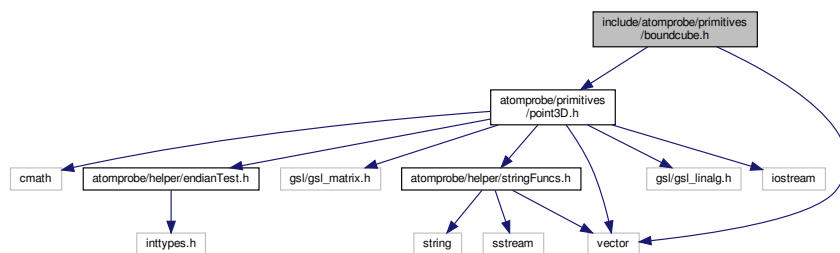
Enumerations

- enum { [AtomProbe::LATTICE_FCC](#), [AtomProbe::LATTICE_BCC](#), [AtomProbe::LATTICE_HCP](#), [AtomProbe::LATTICE_END_OF_ENUM](#) }

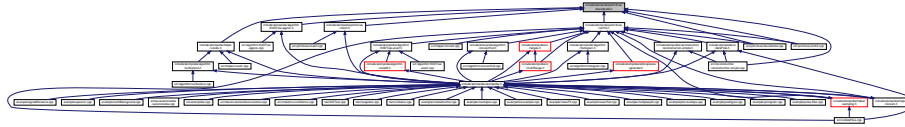
7.54 include/atomprobe/primitives/boundcube.h File Reference

```
#include "atomprobe/primitives/point3D.h"
#include <vector>
```

Include dependency graph for boundcube.h:



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



Classes

- class [AtomProbe::BoundCube](#)
Helper class to define a bounding cube.

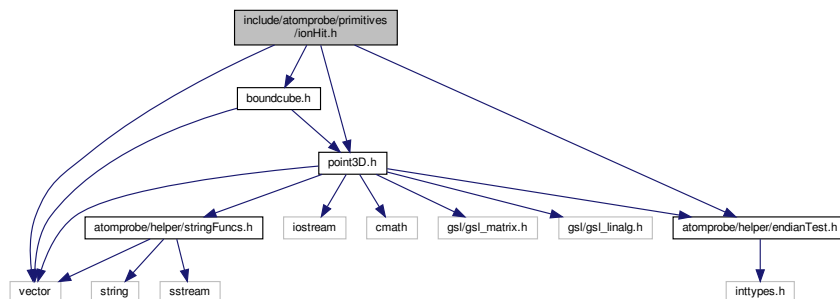
Namespaces

- [AtomProbe](#)

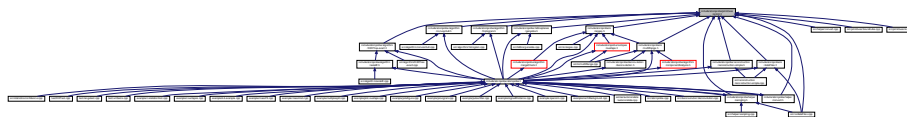
7.55 include/atomprobe/primitives/ionHit.h File Reference

```
#include <vector>
#include "atomprobe/helper/endianTest.h"
#include "point3D.h"
#include "boundcube.h"
```

Include dependency graph for ionHit.h:



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



Classes

- class [AtomProbe::IonHit](#)
This is a data holding class for POS file ions, from.
- class [AtomProbe::EPOS_ENTRY](#)

Namespaces

- [AtomProbe](#)

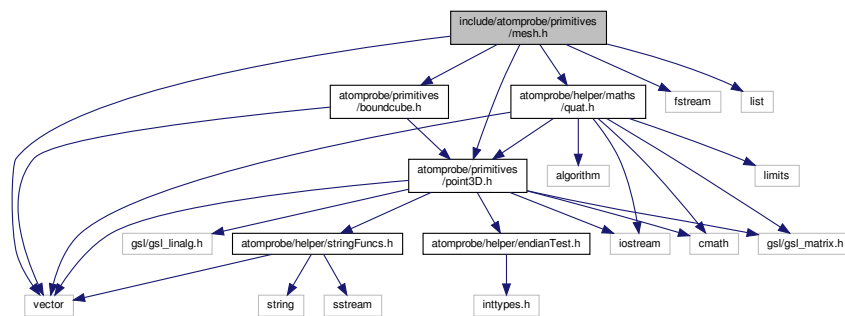
Variables

- const size_t [AtomProbe::EPOS_RECORD_SIZE](#) = 11*4

7.56 include/atomprobe/primitives/mesh.h File Reference

```
#include "atomprobe/primitives/point3D.h"
#include "atomprobe/primitives/boundcube.h"
#include "atomprobe/helper/math/quat.h"
#include <vector>
#include <fstream>
#include <list>
```

Include dependency graph for mesh.h:



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



Classes

- class [AtomProbe::TETRAHEDRON](#)
- class [AtomProbe::TRIANGLE](#)
- class [AtomProbe::LINE](#)
- class [AtomProbe::Mesh](#)

Simple mesh class for storing and manipulating meshes consisting of 2 and 3D simplexes (triangles or tetrahedra)

Namespaces

- [AtomProbe](#)

Enumerations

- enum { [AtomProbe::ELEMENT_TRIANGLE](#) =1, [AtomProbe::ELEMENT_TETRAHEDRON](#) =2, [AtomProbe::ELEMENT_LINE](#) =4 }

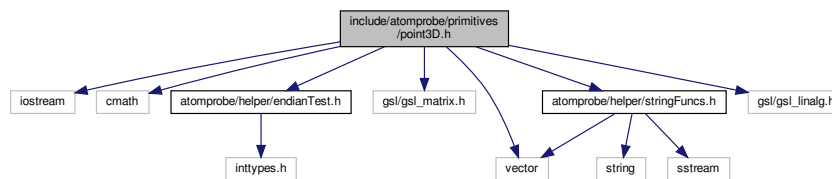
Variables

- const unsigned int [AtomProbe::ELEM_SINGLE_NODE_POINT](#) =15
- const unsigned int [AtomProbe::ELEM_TWO_NODE_LINE](#) =1
- const unsigned int [AtomProbe::ELEM_THREE_NODE_TRIANGLE](#) =2
- const unsigned int [AtomProbe::ELEM_FOUR_NODE_TETRAHEDRON](#) =4
- const char * [AtomProbe::MESH_LOAD_ERRS](#) []

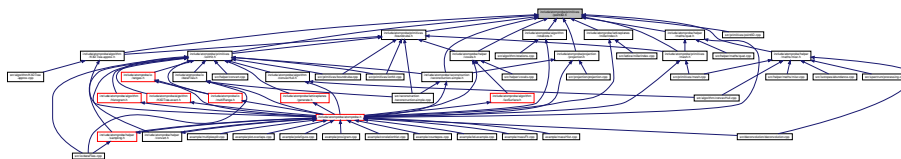
7.57 include/atomprobe/primitives/point3D.h File Reference

```
#include <iostream>
#include <cmath>
#include <vector>
#include <gsl/gsl_matrix.h>
#include "atomprobe/helper/EndianTest.h"
#include "atomprobe/helper/stringFuncs.h"
#include <gsl/gsl_linalg.h>
```

Include dependency graph for point3D.h:



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



Classes

- class [AtomProbe::Point3D](#)
A 3D point data class storage.

Namespaces

- [AtomProbe](#)

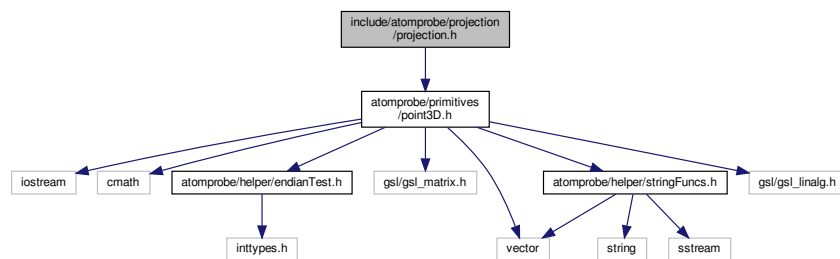
Functions

- void [AtomProbe::gsl_matrix_mult](#) (const gsl_matrix *a, const gsl_matrix *b, gsl_matrix *&res)

7.58 include/atomprobe/projection/projection.h File Reference

```
#include "atomprobe/primitives/point3D.h"
```

Include dependency graph for projection.h:



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



Classes

- class [AtomProbe::SphericPlaneProjection](#)
- class [AtomProbe::GnomonicProjection](#)
- class [AtomProbe::StereographicProjection](#)
- class [AtomProbe::ModifiedFocusSphericProjection](#)

Namespaces

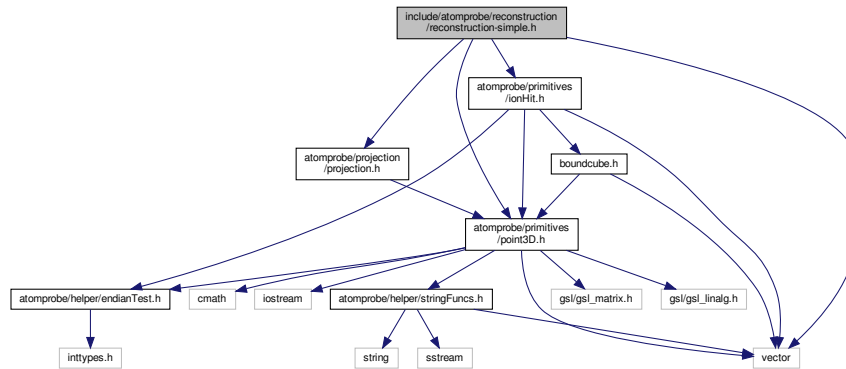
- [AtomProbe](#)

7.59 include/atomprobe/reconstruction/reconstruction-simple.h File Reference

```
#include "atomprobe/primitives/point3D.h"
#include "atomprobe/primitives/ionHit.h"
#include "atomprobe/projection/projection.h"
```

```
#include <vector>
```

Include dependency graph for reconstruction-simple.h:



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



Classes

- class [AtomProbe::ReconstructionSphereOnCone](#)

Namespaces

- [AtomProbe](#)

Enumerations

- enum { [AtomProbe::EVOLUTION_MODE_SHANKANGLE](#) }

Functions

- bool [AtomProbe::reconstructTest](#) ()

7.60 include/atomprobe/spectrum/processing.h File Reference

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



Classes

- struct [AtomProbe::BACKGROUND_PARAMS](#)

Namespaces

- [AtomProbe](#)

Enumerations

- enum { [AtomProbe::BACK_FIT_MODE_CONST_TOF](#), [AtomProbe::BACK_FIT_MODE_ENUM_END](#) }
Background fitting modes.

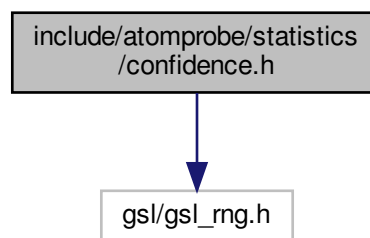
Functions

- unsigned int [AtomProbe::doFitBackground](#) (const std::vector< float > &massData, BACKGROUND_PARAMS ¶ms)
Perform a background fit, assuming constant TOF noise, from 0->inf time.
- std::string [AtomProbe::getFitErrorMsg](#) (unsigned int errCode)
- void [AtomProbe::createMassBackground](#) (float massStart, float massEnd, unsigned int nBinsMass, float tofBackIntensity, std::vector< float > &histogram)
Build a histogram of the background counts.
- void [AtomProbe::findPeaks](#) (const std::vector< float > &x0, std::vector< unsigned int > &peakInds, float sel, bool autoSel=true, bool includeEndpoints=true)
Simple peak-finding algorithm.

7.61 include/atomprobe/statistics/confidence.h File Reference

```
#include <gsl/gsl_rng.h>
```

Include dependency graph for confidence.h:



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



Namespaces

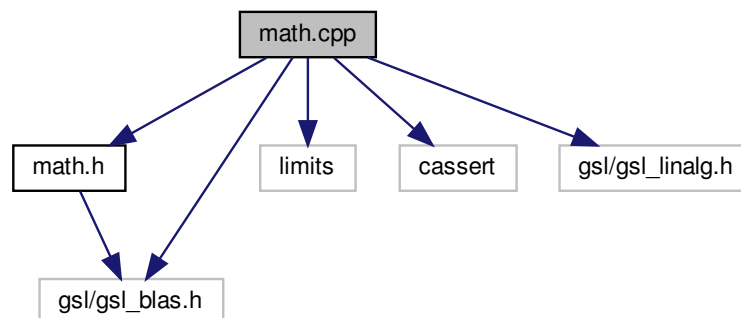
- [AtomProbe](#)

Functions

- void [AtomProbe::poissonConfidenceObservation](#) (float counts, float alpha, float &lBound, float &uBound)
Obtain poisson confidence limits for the mean of a poisson distribution.
- bool [AtomProbe::numericalEstimatePoissRatioConf](#) (float lambda1, float lambda2, float alpha, unsigned int nTrials, gsl_rng *r, float &lBound, float &uBound)
Brute-force poisson ratio confidence estimator. Returns the confidence interval in the estimate of the mean, at the given rates.
- bool [AtomProbe::numericalEstimateGaussRatioConf](#) (float mu1, float mu2, float var1, float var2, float alpha, unsigned int nTrials, gsl_rng *r, float &lBound, float &uBound)
Brute-force gaussian ratio confidence estimator.
- bool [AtomProbe::numericalEstimateSkellamConf](#) (float lambda1, float lambda2, float alpha, unsigned int nTrials, gsl_rng *r, float &lBound, float &uBound)
Brute-force the confidence bounds of a skellam distribution.
- bool [AtomProbe::zechConfidenceLimits](#) (float lambdaBack, unsigned int observation, float alpha, float &estimate)
Provides a best estimate for true signal when true signal, background.

7.62 math.cpp File Reference

```
#include "math.h"
#include <limits>
#include <cassert>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_linalg.h>
Include dependency graph for math.cpp:
```



Macros

- #define [ASSERT](#)(f) assert((f))

Functions

- void [psuedoInverseFromSVD](#) (const gsl_matrix *U, const gsl_vector *S, const gsl_matrix *V, gsl_matrix *P)
- void [gsl_matrix_mult](#) (const gsl_matrix *A, const gsl_matrix *B, gsl_matrix *&res, bool alloc)
- void [gsl_pseudo_inverse](#) (gsl_matrix *A, gsl_matrix *&res)

7.62.1 Macro Definition Documentation

7.62.1.1 ASSERT

```
#define ASSERT(  
    f ) assert((f))
```

Definition at line 6 of file math.cpp.

Referenced by [gsl_matrix_mult\(\)](#).

7.62.2 Function Documentation

7.62.2.1 gsl_matrix_mult()

```
void gsl_matrix_mult (  
    const gsl_matrix * A,  
    const gsl_matrix * B,  
    gsl_matrix *& res,  
    bool alloc )
```

Definition at line 16 of file math.cpp.

Referenced by [psuedoInverseFromSVD\(\)](#).

7.62.2.2 gsl_pseudo_inverse()

```
void gsl_pseudo_inverse (  
    gsl_matrix * A,  
    gsl_matrix *& res )
```

Definition at line 56 of file math.cpp.

7.62.2.3 psuedoInverseFromSVD()

```
void psuedoInverseFromSVD (
    const gsl_matrix * U,
    const gsl_vector * S,
    const gsl_matrix * V,
    gsl_matrix * P )
```

Definition at line 78 of file math.cpp.

References `gsl_matrix_mult()`.

Referenced by `gsl_pseudo_inverse()`.

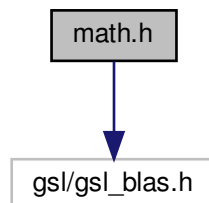
Here is the call graph for this function:



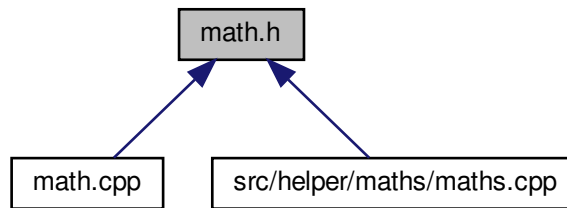
7.63 math.h File Reference

```
#include <gsl/gsl_blas.h>
```

Include dependency graph for math.h:



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



Functions

- void [gsl_pseudo_inverse](#) (gsl_matrix *A, gsl_matrix *&res)
- void [gsl_matrix_mult](#) (const gsl_matrix *A, const gsl_matrix *B, gsl_matrix *&res, bool alloc=true)
- template<class T >
void [computeMeanAndVariance](#) (const T *data, unsigned int n, T &mean, T &var)

7.63.1 Function Documentation

7.63.1.1 computeMeanAndVariance()

```
template<class T >  
void computeMeanAndVariance (  
    const T * data,  
    unsigned int n,  
    T & mean,  
    T & var )
```

Definition at line 16 of file math.h.

7.63.1.2 gsl_matrix_mult()

```
void gsl_matrix_mult (  
    const gsl_matrix * A,  
    const gsl_matrix * B,  
    gsl_matrix *& res,  
    bool alloc = true )
```

Definition at line 16 of file math.cpp.

References ASSERT.

Referenced by [psuedoInverseFromSVD\(\)](#).

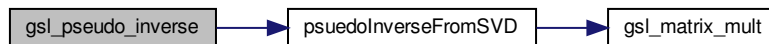
7.63.1.3 gsl_pseudo_inverse()

```
void gsl_pseudo_inverse (
    gsl_matrix * A,
    gsl_matrix *& res )
```

Definition at line 56 of file math.cpp.

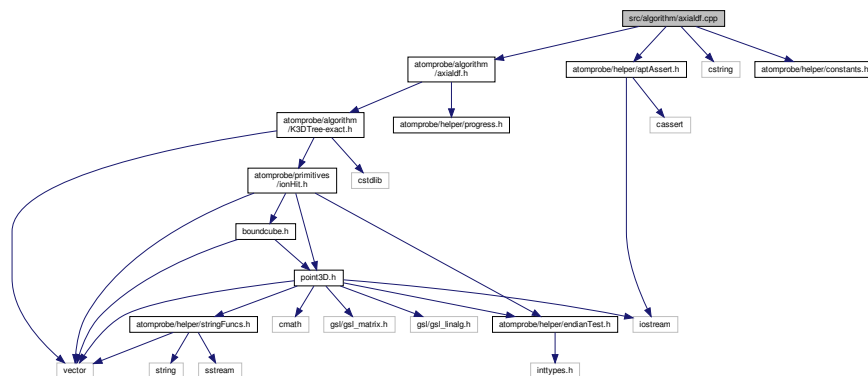
References psuedoInverseFromSVD().

Here is the call graph for this function:



7.64 src/algorithm/axialdf.cpp File Reference

```
#include "atomprobe/algorithm/axialdf.h"
#include "atomprobe/helper/aptAssert.h"
#include <cstring>
#include "atomprobe/helper/constants.h"
Include dependency graph for axialdf.cpp:
```



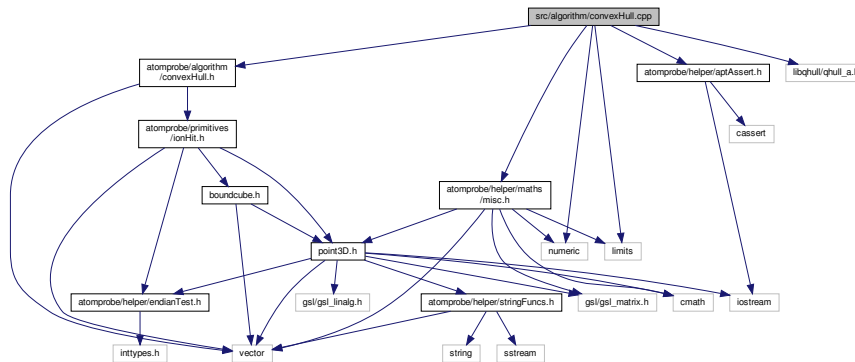
Namespaces

- [AtomProbe](#)

Functions

- unsigned int [AtomProbe::generate1DAxialDistHist](#) (const std::vector< Point3D > &pointList, K3DTreeExact &tree, const Point3D &axisDir, float distMax, std::vector< unsigned int > &histogram)
Generate a 1D axial distribution function,.
- unsigned int [AtomProbe::generate1DAxialDistHistSweep](#) (const std::vector< Point3D > &pointList, K3DTreeExact &tree, float distMax, float dTheta, float dPhi, ProgressBar &prog, std::vector< std::vector< std::vector< unsigned int > > > &histogram)
Generate a series of 1D distribution functions, one per pixel in a 2D grid of spherical coordinate directions.


```
#include "atomprobe/helper/aptAssert.h"
Include dependency graph for convexHull.cpp:
```



Namespaces

- [AtomProbe](#)

Functions

- void [AtomProbe::freeConvexHull](#) ()
- unsigned int [AtomProbe::doHull](#) (unsigned int bufferSize, double *buffer, vector< Point3D > &resHull, Point3D &midPoint, bool freeHullOnExit)
- unsigned int [AtomProbe::computeConvexHull](#) (const std::vector< IonHit > &data, unsigned int *progress, bool(*callback)(bool), std::vector< Point3D > &curHull, bool freeHull)

Obtain the convex hull of a set of ions.

- unsigned int [AtomProbe::GetReducedHullPts](#) (const std::vector< IonHit > &points, float reductionDim, unsigned int *progress, bool(*callback)(bool), std::vector< IonHit > &pointResult)

Obtain a set of points that are a subset of points the convex hull that are at least "reductionDim" away from the hull itself.

Variables

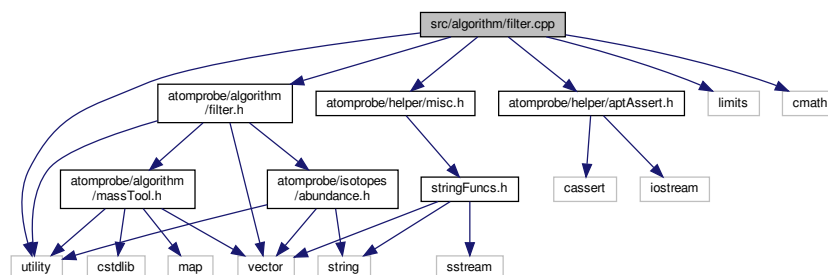
- bool [AtomProbe::qhullInited](#) =false
- const unsigned int [AtomProbe::HULL_GRAB_SIZE](#) =4096

7.67 src/algorithm/filter.cpp File Reference

```
#include "atomprobe/algorithm/filter.h"
#include "atomprobe/helper/aptAssert.h"
#include "atomprobe/helper/misc.h"
#include <limits>
#include <cmath>
```

```
#include <utility>
```

Include dependency graph for filter.cpp:



Namespaces

- [AtomProbe](#)

Functions

- unsigned int [AtomProbe::countIntensityEvents](#) (const vector< pair< float, float > > &data, float minV, float maxV)
- void [AtomProbe::buildFrequencyTable](#) (const vector< ISOTOPE_ENTRY > &solutionVec, vector< size_t > &solutionElements, vector< size_t > &solutionFrequency)
- void [AtomProbe::filterBySolutionPPM](#) (const AbundanceData &massTable, float minPpm, std::vector< std::vector< ISOTOPE_ENTRY > > &solutions)

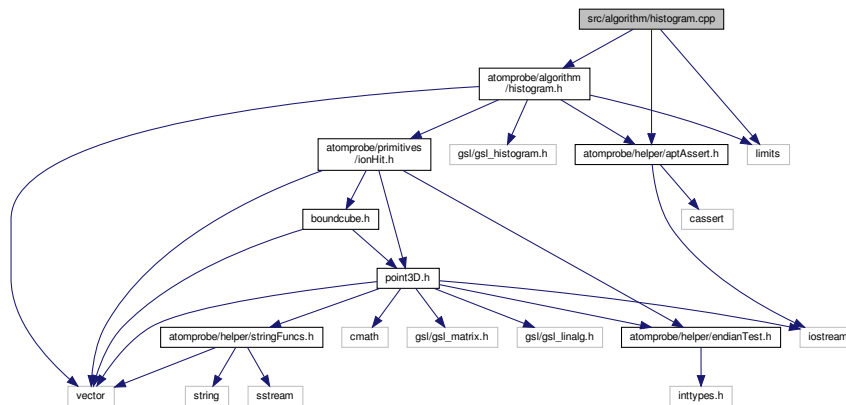
Use the maximum possible PPM for each isotopic combination to filter possible solutions.

- void [AtomProbe::filterPeakNeedBiggerObs](#) (const AbundanceData &massTable, const std::vector< float > &peakData, float tolerance, size_t solutionCharge, std::vector< std::vector< ISOTOPE_ENTRY > > &solutions)
- vector< float > [AtomProbe::maxExplainedFraction](#) (const vector< pair< float, float > > &intensityData, float peakMass, float massWidth, const vector< vector< ISOTOPE_ENTRY > > &solutions, const AbundanceData &massTable, float massDistTol, unsigned int solutionCharge)

7.68 src/algorithm/histogram.cpp File Reference

```
#include "atomprobe/algorithm/histogram.h"
#include "atomprobe/helper/aptAssert.h"
```

```
#include <limits>
Include dependency graph for histogram.cpp:
```



Namespaces

- [AtomProbe](#)

Functions

- bool [AtomProbe::incrementCorrelationHist](#) (const std::vector< EPOS_ENTRY > &eposlons, std::vector< std::vector< unsigned int > > &histogram, float stepMass, float endMass)
- bool [AtomProbe::correlationHistogram](#) (const std::vector< EPOS_ENTRY > &eposlons, std::vector< std::vector< unsigned int > > &histogram, float stepMass, float endMass, bool lowerTriangular=false)

Generates an ion "correlation histogram" from a vector of EPOS ions. The input vector MUST
- bool [AtomProbe::accumulateCorrelationHistogram](#) (const std::vector< EPOS_ENTRY > &eposlons, std::vector< std::vector< unsigned int > > &histogram, float stepMass, float endMass, bool lowerTriangular=true)

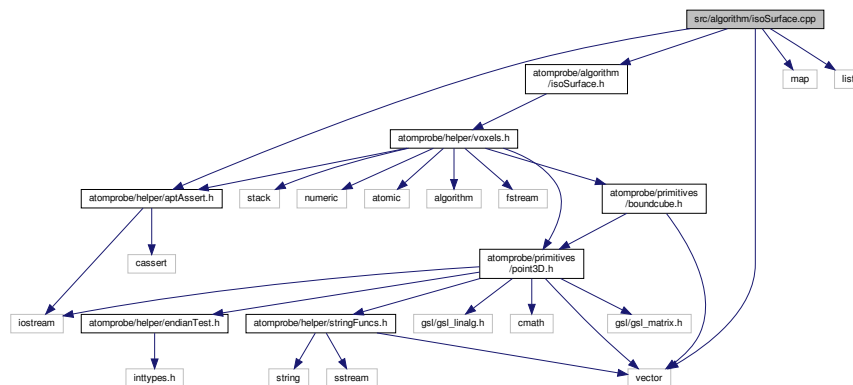
Increments a correlation histogram, as per correlationHistogram,.

7.69 src/algorithm/isoSurface.cpp File Reference

```
#include "atomprobe/algorithm/isoSurface.h"
#include "atomprobe/helper/aptAssert.h"
#include <map>
#include <list>
```

```
#include <vector>
```

Include dependency graph for isoSurface.cpp:



Namespaces

- [AtomProbe](#)

Functions

- `template<class T >`
void [AtomProbe::removeElements](#) (const std::vector< size_t > &elems, std::vector< T > &vec)
- void [AtomProbe::marchingCubes](#) (const Voxels< float > &v, float isoValue, std::vector< TriangleWithVertex↔ Norm > &tVec)

Variables

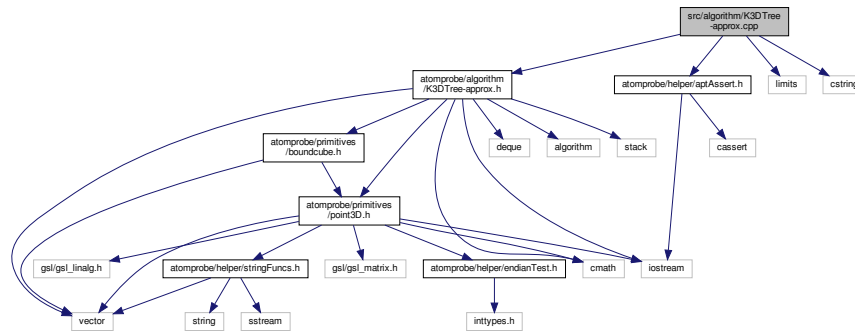
- int [AtomProbe::aiCubeEdgeFlags](#) [256]
- int [AtomProbe::a2iTriangleConnectionTable](#) [256][16]
- const unsigned int [AtomProbe::VERTEX_OFFSET](#) [8][3]
- int [AtomProbe::edgeRemap](#) [12]

7.70 src/algorithm/K3DTree-approx.cpp File Reference

```
#include "atomprobe/algorithm/K3DTree-approx.h"
#include "atomprobe/helper/aptAssert.h"
#include <limits>
```

```
#include <cstring>
```

Include dependency graph for K3DTree-approx.cpp:



Namespaces

- [AtomProbe](#)

7.71 src/algorithm/K3DTree-exact.cpp File Reference

```
#include "atomprobe/algorithm/K3DTree-exact.h"
```

```
#include <stack>
```

```
#include <queue>
```

```
#include <algorithm>
```

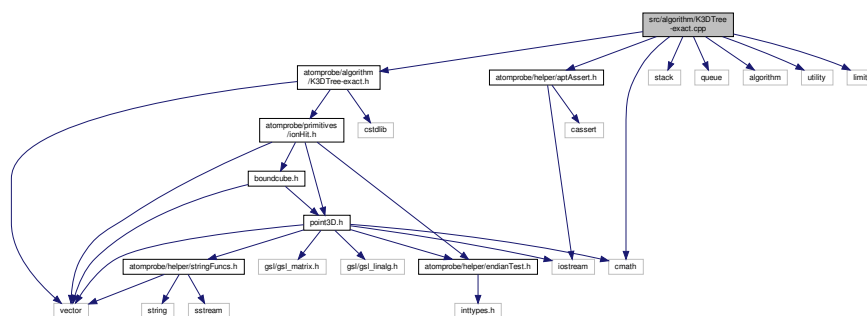
```
#include <cmath>
```

```
#include <utility>
```

```
#include <limits>
```

```
#include "atomprobe/helper/aptAssert.h"
```

Include dependency graph for K3DTree-exact.cpp:



Classes

- class [AxisCompareExact](#)
Functor allowing for sorting of points in 3D.
- class [NodeWalk](#)

Variables

- const int [PROGRESS_REDUCE](#) =5000

7.71.1 Variable Documentation

7.71.1.1 PROGRESS_REDUCE

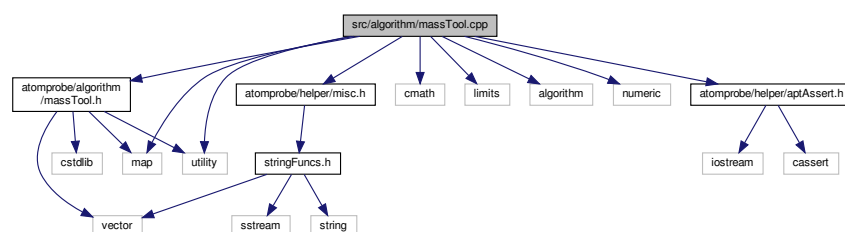
```
const int PROGRESS_REDUCE =5000
```

Definition at line 38 of file K3DTree-exact.cpp.

7.72 src/algorithm/massTool.cpp File Reference

```
#include "atomprobe/algorithm/massTool.h"
#include <cmath>
#include <limits>
#include <algorithm>
#include <numeric>
#include <utility>
#include <map>
#include "atomprobe/helper/misc.h"
#include "atomprobe/helper/aptAssert.h"
```

Include dependency graph for massTool.cpp:



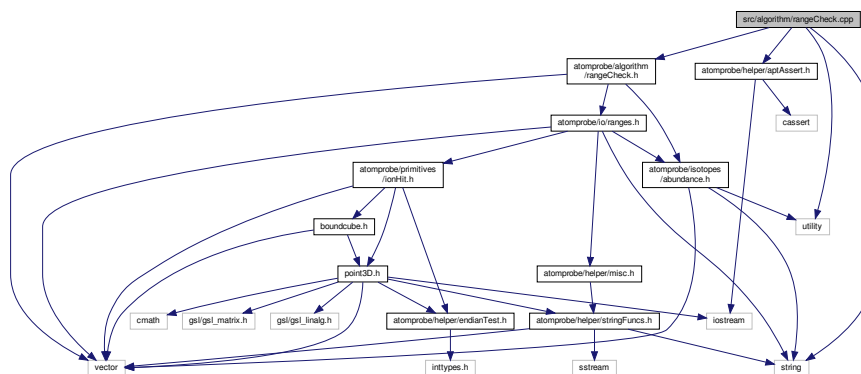
Classes

- class [WEIGHT_SEARCH_ENTRY](#)
- class [FixedStack< T >](#)

7.73 src/algorithm/rangeCheck.cpp File Reference

```
#include "atomprobe/algorithm/rangeCheck.h"
#include "atomprobe/helper/aptAssert.h"
#include <string>
#include <utility>
```

Include dependency graph for rangeCheck.cpp:



Classes

- struct [AtomProbe::RANGE_MOLECULE](#)

Namespaces

- [AtomProbe](#)

Functions

- bool [AtomProbe::pairContains](#) (const pair< float, float > &p, float m)
- RANGE_MOLECULE [AtomProbe::getRangeMolecule](#) (const AbundanceData &massTable, const RangeFile &rng, unsigned int rangeld)
- void [AtomProbe::checkMassRangingCorrectness](#) (const RangeFile &rng, AbundanceData &massTable, float massTolerance, unsigned int maxChargeState, unsigned int maxComponents, std::vector< bool > &bad← Ranges)

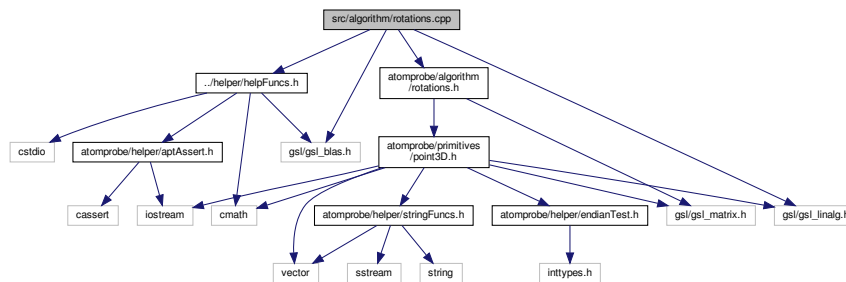
Ensure that each mass given spans a peak that should exist.

7.74 src/algorithm/rotations.cpp File Reference

```
#include "atomprobe/algorithm/rotations.h"
#include "../helper/helpFuncs.h"
#include <gsl/gsl_blas.h>
```



```
#include <gsl/gsl_linalg.h>
Include dependency graph for rotations.cpp:
```



Namespaces

- [AtomProbe](#)

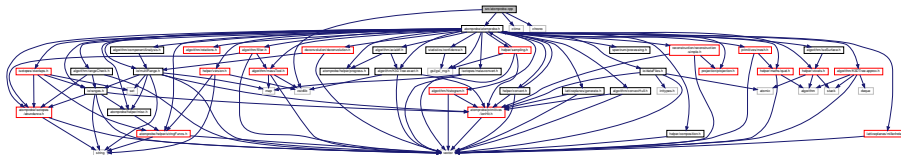
Functions

- void [AtomProbe::computeRotationMatrix](#) (const Point3D &r1, const Point3D &r2, const Point3D &r1, const Point3D &r2, gsl_matrix *m)

7.75 src/atomprobe.cpp File Reference

```
#include "atomprobe/atomprobe.h"
#include <cstdlib>
#include <ctime>
#include <chrono>
```

Include dependency graph for atomprobe.cpp:



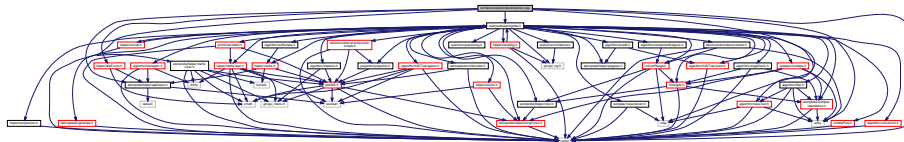
Namespaces

- [AtomProbe](#)

7.76 src/deconvolution/deconvolution.cpp File Reference

```
#include "atomprobe/atomprobe.h"
#include "atomprobe/helper/maths/misc.h"
#include "atomprobe/helper/aptAssert.h"
#include "helper/helpFuncs.h"
#include <utility>
#include <vector>
#include <map>
```

Include dependency graph for deconvolution.cpp:



Namespaces

- [AtomProbe](#)

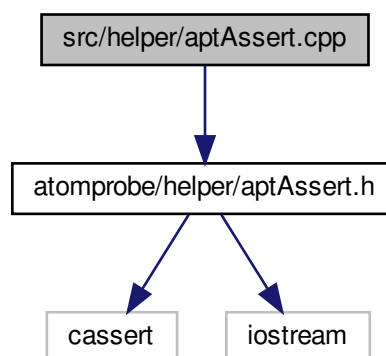
Functions

- float [AtomProbe::nullBackground](#) (float rangeStart, float rangeEnd)
- bool [AtomProbe::leastSquaresDeconvolve](#) (const MultiRange &rangeData, const AbundanceData &abundance, const float massTol, const vector< IonHit > &hits, float(*backgroundEstimator)(float, float), vector< pair< unsigned int, float > > &decomposedHits)

7.77 src/helper/aptAssert.cpp File Reference

```
#include <atomprobe/helper/aptAssert.h>
```

Include dependency graph for aptAssert.cpp:



Namespaces

- [AtomProbe](#)

Functions

- bool [AtomProbe::getHardAssert](#) ()
- void [AtomProbe::setHardAssert](#) (bool enabled)

Do assertions cause a straight up crash (enabled), or write a message to stderr (disabled)? By default, hard crash.
- void [AtomProbe::askAssert](#) (const char *, unsigned int)

Either abort program, or ask the user what to do for an assertion. depending on hardAssert setting.

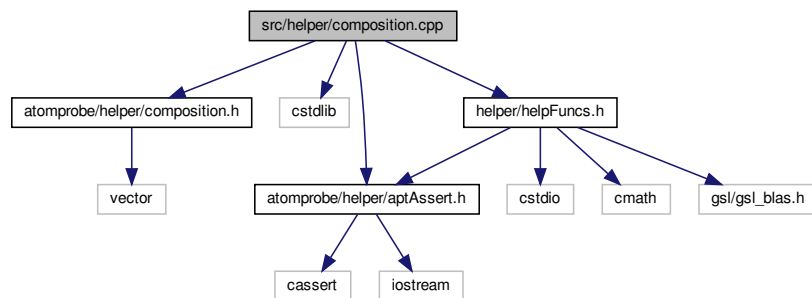
Variables

- bool [AtomProbe::hardAssert](#) =true

7.78 src/helper/composition.cpp File Reference

```
#include "atomprobe/helper/composition.h"
#include <cstdlib>
#include "atomprobe/helper/aptAssert.h"
#include "helper/helpFuncs.h"
```

Include dependency graph for composition.cpp:



Namespaces

- [AtomProbe](#)

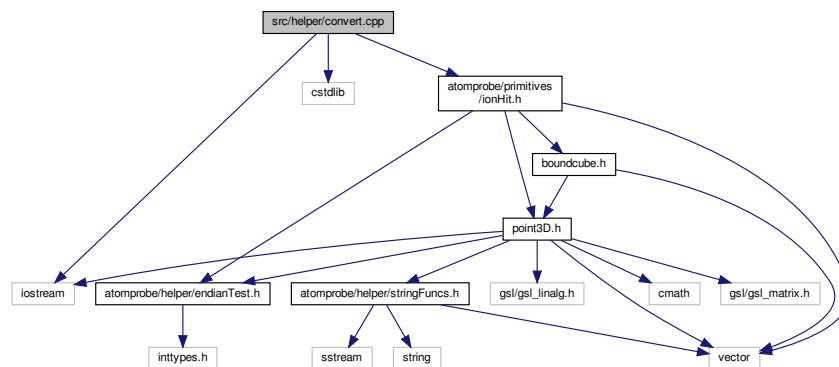
Functions

- void [AtomProbe::computeComposition](#) (const std::vector< unsigned int > &countData, std::vector< float > &compositionData)

7.79 src/helper/convert.cpp File Reference

```
#include <iostream>
#include <cstdlib>
#include "atomprobe/primitives/ionHit.h"
```

Include dependency graph for convert.cpp:



Functions

- void `convertEPOStoPos` (const std::vector< [EPOS_ENTRY](#) > &eposEntry, std::vector< [IonHit](#) > &posFile)
Convert an incoming entry of EPOS files to pos, and the append this to the pos vector given.

7.79.1 Function Documentation

7.79.1.1 convertEPOStoPos()

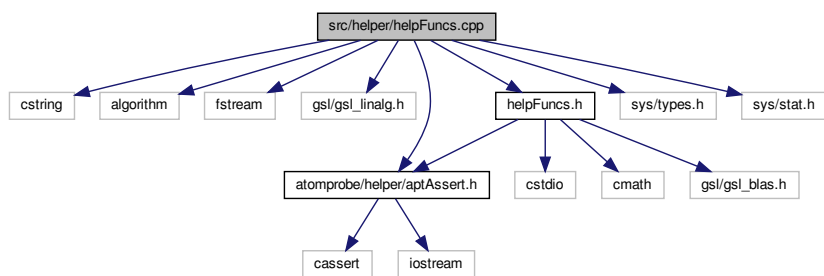
```
void convertEPOStoPos (
    const std::vector< EPOS\_ENTRY > & eposEntry,
    std::vector< IonHit > & posFile )
```

Convert an incoming entry of EPOS files to pos, and the append this to the pos vector given.

Definition at line 29 of file convert.cpp.

7.80 src/helper/helpFuncs.cpp File Reference

```
#include <cstring>
#include <algorithm>
#include <fstream>
#include <gsl/gsl_linalg.h>
#include "atomprobe/helper/aptAssert.h"
#include "helpFuncs.h"
#include <sys/types.h>
#include <sys/stat.h>
Include dependency graph for helpFuncs.cpp:
```



Namespaces

- [AtomProbe](#)

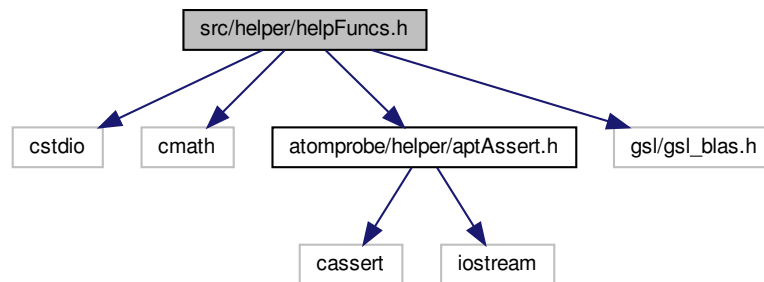
Functions

- char * [AtomProbe::myStrDup](#) (const char *s)
- void [AtomProbe::pushLocale](#) (const char *newLocale, int type)
- void [AtomProbe::popLocale](#) ()
- bool [AtomProbe::getFilesize](#) (const char *fname, size_t &size)
- void [AtomProbe::gsl_print_matrix](#) (const gsl_matrix *m)
- void [AtomProbe::gsl_print_vector](#) (const gsl_vector *v)
- float [AtomProbe::gsl_determinant](#) (const gsl_matrix *m)
- bool [AtomProbe::isNotDirectory](#) (const char *filename)

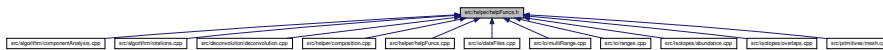
7.81 src/helper/helpFuncs.h File Reference

```
#include <cstdio>
#include <cmath>
#include "atomprobe/helper/aptAssert.h"
```

```
#include <gsl/gsl_blas.h>
Include dependency graph for helpFuncs.h:
```



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



Classes

- class [AtomProbe::ComparePairFirst](#)
- class [AtomProbe::ComparePairSecond](#)
- class [AtomProbe::ComparePairFirstReverse](#)

Namespaces

- [AtomProbe](#)

Macros

- `#define XOR(a, b) ((!(a)) ^ (!(b)))`
- `#define ARRAYSIZE(f) (sizeof (f) / sizeof(*f))`

Functions

- void [AtomProbe::pushLocale](#) (const char *newLocale, int type)
- void [AtomProbe::popLocale](#) ()
- int [AtomProbe::fpeek](#) (FILE *stream)
- bool [AtomProbe::getFilesize](#) (const char *fname, size_t &size)
- void [AtomProbe::gsl_print_matrix](#) (const gsl_matrix *m)
- void [AtomProbe::gsl_print_vector](#) (const gsl_vector *v)
- float [AtomProbe::gsl_determinant](#) (const gsl_matrix *m)
- bool [AtomProbe::isNotDirectory](#) (const char *filename)

7.81.1 Macro Definition Documentation

7.81.1.1 ARRAYSIZE

```
#define ARRAYSIZE(  
    f ) (sizeof (f) / sizeof(*f))
```

Definition at line 34 of file helpFuncs.h.

Referenced by AtomProbe::RangeFile::getAllExts(), AtomProbe::RangeFile::getErrString(), AtomProbe::getPos↵
FileErrString(), AtomProbe::Mesh::loadGmshMesh(), AtomProbe::RangeFile::RangeFile(), AtomProbe::Range↵
File::rangeTypeString(), and AtomProbe::readPosapOps().

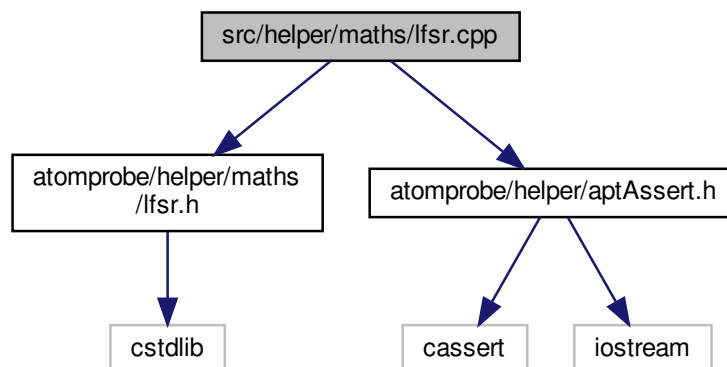
7.81.1.2 XOR

```
#define XOR(  
    a,  
    b ) ((!(a)) ^ !(b))
```

Definition at line 30 of file helpFuncs.h.

7.82 src/helper/mathslfsr.cpp File Reference

```
#include "atomprobe/helper/mathslfsr.h"  
#include "atomprobe/helper/aptAssert.h"  
Include dependency graph for lfsr.cpp:
```



Namespaces

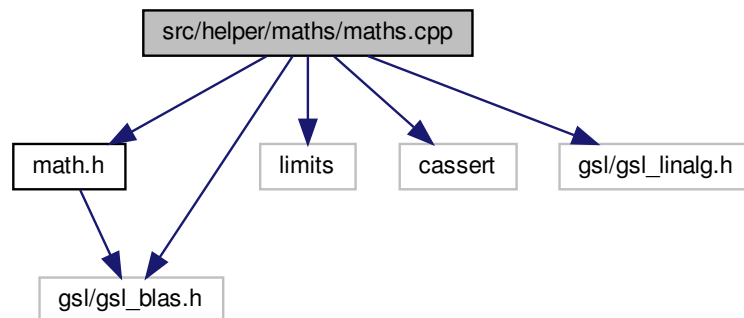
- [AtomProbe](#)

Variables

- `const size_t AtomProbe::maximumLinearTable []`

7.83 src/helper/maths/maths.cpp File Reference

```
#include "math.h"
#include <limits>
#include <cassert>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_linalg.h>
Include dependency graph for maths.cpp:
```



Macros

- `#define ASSERT(f) assert((f))`

Functions

- void [psuedoInverseFromSVD](#) (const `gsl_matrix *U`, const `gsl_vector *S`, const `gsl_matrix *V`, `gsl_matrix *P`)
- void [gsl_matrix_mult](#) (const `gsl_matrix *A`, const `gsl_matrix *B`, `gsl_matrix *&res`, bool alloc)
- void [gsl_pseudo_inverse](#) (`gsl_matrix *A`, `gsl_matrix *&res`)

7.83.1 Macro Definition Documentation

7.83.1.1 ASSERT

```
#define ASSERT(  
    f ) assert((f))
```

Definition at line 6 of file maths.cpp.

Referenced by `gsl_matrix_mult()`.

7.83.2 Function Documentation

7.83.2.1 `gsl_matrix_mult()`

```
void gsl_matrix_mult (  
    const gsl_matrix * A,  
    const gsl_matrix * B,  
    gsl_matrix *& res,  
    bool alloc )
```

Definition at line 16 of file maths.cpp.

References ASSERT.

Referenced by `psuedoInverseFromSVD()`.

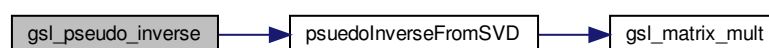
7.83.2.2 `gsl_pseudo_inverse()`

```
void gsl_pseudo_inverse (  
    gsl_matrix * A,  
    gsl_matrix *& res )
```

Definition at line 56 of file maths.cpp.

References `psuedoInverseFromSVD()`.

Here is the call graph for this function:



7.83.2.3 psuedoInverseFromSVD()

```
void psuedoInverseFromSVD (
    const gsl_matrix * U,
    const gsl_vector * S,
    const gsl_matrix * V,
    gsl_matrix * P )
```

Definition at line 78 of file maths.cpp.

References `gsl_matrix_mult()`.

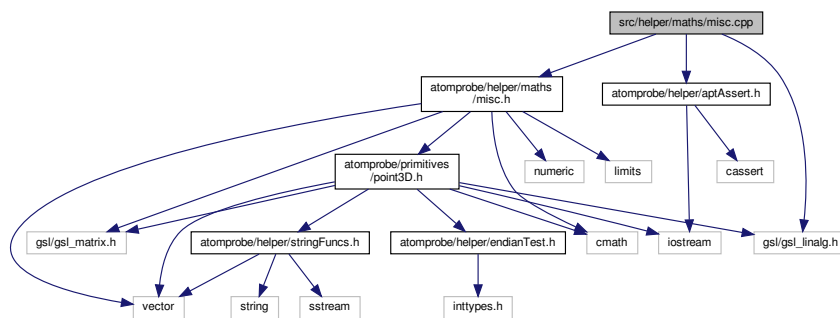
Referenced by `gsl_pseudo_inverse()`.

Here is the call graph for this function:



7.84 src/helper/maths/misc.cpp File Reference

```
#include "atomprobe/helper/maths/misc.h"
#include "gsl/gsl_linalg.h"
#include "atomprobe/helper/aptAssert.h"
Include dependency graph for misc.cpp:
```



Namespaces

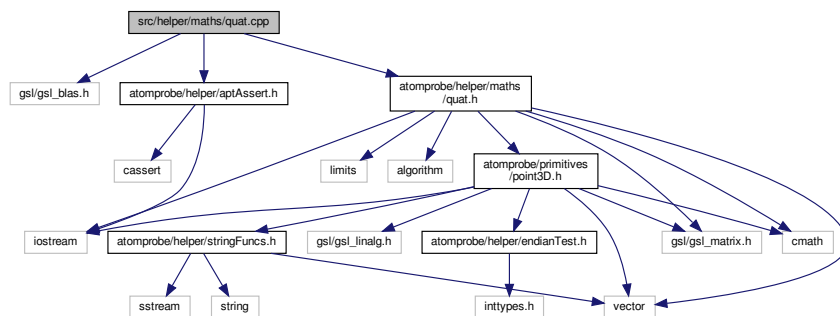
- [AtomProbe](#)

Functions

- unsigned int [AtomProbe::estimateRank](#) (const gsl_matrix *m, float tolerance=sqrt(std::numeric_limits< float >::epsilon()))
Estimate the rank of the given matrix.
- bool [AtomProbe::solveLeastSquares](#) (gsl_matrix *m, gsl_vector *b, gsl_vector *&x)
Use an SVD based least-squares solver to solve $Mx=b$ (for x).
- unsigned int [AtomProbe::vectorPointDir](#) (const Point3D &pA, const Point3D &pB, const Point3D &vC, const Point3D &vD)
Check which way vectors attached to two 3D points "point",.
- float [AtomProbe::distanceToSegment](#) (const Point3D &fA, const Point3D &fB, const Point3D &p)
- float [AtomProbe::signedDistanceToFacet](#) (const Point3D &fA, const Point3D &fB, const Point3D &fC, const Point3D &normal, const Point3D &p)
Find the distance between a point, and a triangular facet – may be positive or negative.
- float [AtomProbe::distanceToFacet](#) (const Point3D &fA, const Point3D &fB, const Point3D &fC, const Point3D &normal, const Point3D &p)
- double [AtomProbe::det3by3](#) (const double *ptArray)
- bool [AtomProbe::trilsDegenerate](#) (const Point3D &fA, const Point3D &fB, const Point3D &fC)
- double [AtomProbe::pyramidVol](#) (const Point3D *planarPts, const Point3D &apex)

7.85 src/helper/maths/quat.cpp File Reference

```
#include <gsl/gsl_blas.h>
#include <atomprobe/helper/maths/quat.h>
#include <atomprobe/helper/aptAssert.h>
Include dependency graph for quat.cpp:
```



Namespaces

- [AtomProbe](#)

Functions

- void [AtomProbe::quat_mult_no_second_a](#) (Quaternion *result, const Quaternion *q1, const Quaternion *q2)
- void [AtomProbe::quat_pointmult](#) (Point3f *result, const Quaternion *q1, const Quaternion *q2)
- void [AtomProbe::quat_rot](#) (Point3D &p, const Point3D &r, float angle)

Rotate a point around a given rotation axis by a specified angle.
- void [AtomProbe::quat_rot](#) (Point3f *point, const Point3f *rotVec, float angle)

Rotate a point around a given vector, with specified angle.
- void [AtomProbe::quat_rot_array](#) (Point3D *point, unsigned int n, const Point3f *rotVec, float angle)

Rotate each point in array of size n around a given vector, with specified angle.
- void [AtomProbe::quat_rot_array](#) (Point3f *point, unsigned int n, const Point3f *rotVec, float angle)

Rotate each point in array of size n around a given vector, with specified angle.
- void [AtomProbe::quat_get_rot_quat](#) (const Point3f *rotVec, float angle, Quaternion *rotQuat)

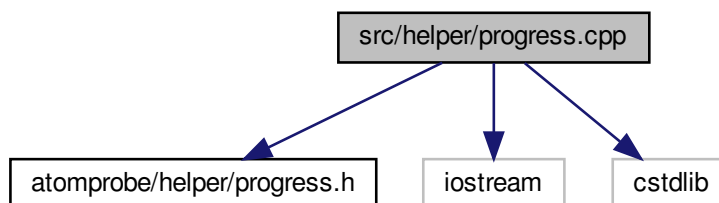
Compute the quaternion for specified rotation.
- void [AtomProbe::quat_rot_apply_quat](#) (Point3f *point, const Quaternion *rotQuat)

Use previously generated quats from quat_get_rot_quats to rotate a point.
- void [AtomProbe::quat_invert](#) (Quaternion *quat)

7.86 src/helper/progress.cpp File Reference

```
#include "atomprobe/helper/progress.h"
#include <iostream>
#include <cstdlib>
```

Include dependency graph for progress.cpp:

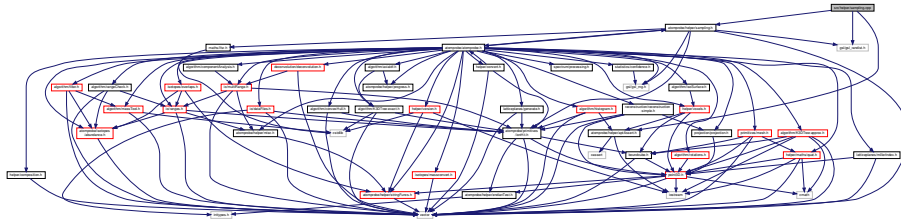


Namespaces

- [AtomProbe](#)

7.87 src/helper/sampling.cpp File Reference

```
#include "atomprobe/helper/aptAssert.h"
#include "atomprobe/helper/sampling.h"
#include "gsl/gsl_randist.h"
Include dependency graph for sampling.cpp:
```



Namespaces

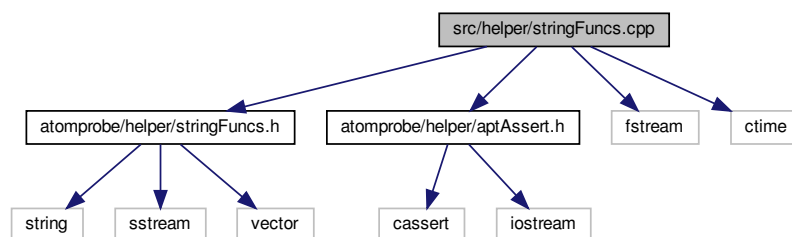
- [AtomProbe](#)

Functions

- void [AtomProbe::sampleIons](#) (const std::vector< IonHit > &ions, float sampleFactor, std::vector< IonHit > &sampled, bool strongRandom=true)

7.88 src/helper/stringFuncs.cpp File Reference

```
#include "atomprobe/helper/stringFuncs.h"
#include "atomprobe/helper/aptAssert.h"
#include <fstream>
#include <ctime>
Include dependency graph for stringFuncs.cpp:
```



Namespaces

- [AtomProbe](#)

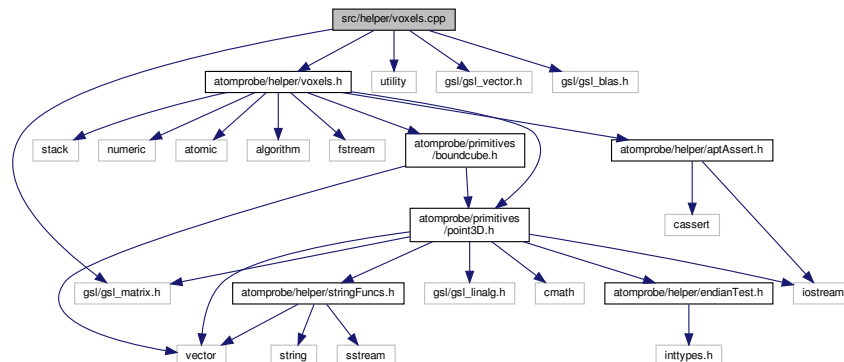
Functions

- `std::string AtomProbe::onlyFilename` (const `std::string` &path)
Return only the filename component.
- `std::string AtomProbe::onlyDir` (const `std::string` &path)
Return only the directory name component of the full path.
- void `AtomProbe::nullifyMarker` (char *buffer, char marker)
- void `AtomProbe::ucharToHexStr` (unsigned char c, `std::string` &s)
- void `AtomProbe::hexStrToUChar` (const `std::string` &s, unsigned char &c)
- `std::string AtomProbe::digitString` (unsigned int thisDigit, unsigned int maxDigit)
Generate a string with leading digits up to maxDigit (eg, if maxDigit is 424, and thisDigit is 1.
- `std::string AtomProbe::stripWhite` (const `std::string` &str)
Strip whitespace, (eg tab,space) from either side of a string.
- `std::string AtomProbe::stripChars` (const `std::string` &Str, const char *chars)
- void `AtomProbe::stripZeroEntries` (`std::vector`< `std::string` > &s)
- `std::string AtomProbe::lowercase` (`std::string` s)
Return a lowercase version for a given string.
- void `AtomProbe::splitStrsRef` (const char *cpStr, const char delim, `std::vector`< `std::string` > &v)
Split string references using a single delimiter.
- void `AtomProbe::splitStrsRef` (const char *cpStr, const char *delim, `std::vector`< `std::string` > &v)
Split string references using any of a given string of delimiters.
- bool `AtomProbe::parseColString` (const `std::string` &str, unsigned char &r, unsigned char &g, unsigned char &b, unsigned char &a)
Parse a colour string containing rgb[a]; hex for with leading #.
- void `AtomProbe::genColString` (unsigned char r, unsigned char g, unsigned char b, `std::string` &s)

7.89 src/helper/voxels.cpp File Reference

```
#include "atomprobe/helper/voxels.h"
#include <utility>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_blas.h>
```

Include dependency graph for voxels.cpp:



Namespaces

- [AtomProbe](#)

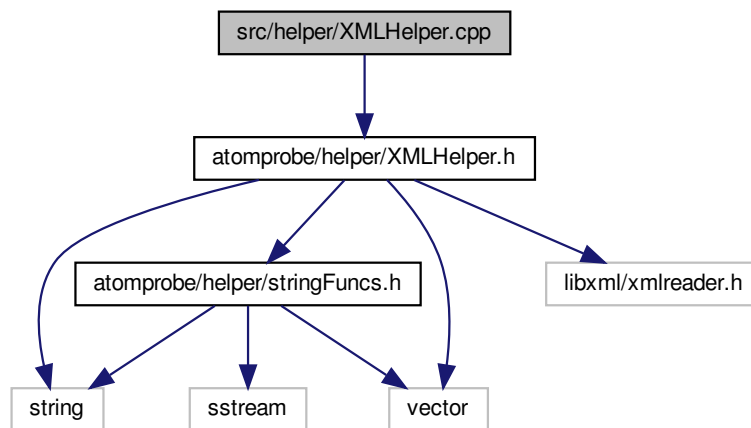
Functions

- void [AtomProbe::incrementDataDistanceWeight](#) (const Point3D &p, Voxels< float > &v)
- void [AtomProbe::countDataDistanceWeight](#) (const vector< Point3D > &pts, Voxels< float > &v)

7.90 src/helper/XMLHelper.cpp File Reference

```
#include "atomprobe/helper/XMLHelper.h"
```

Include dependency graph for XMLHelper.cpp:



Namespaces

- [AtomProbe](#)

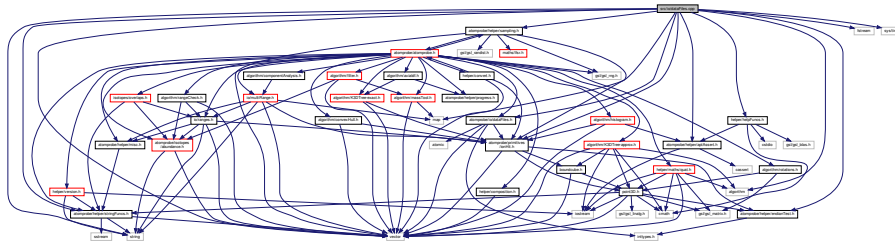
Functions

- void [AtomProbe::XMLFreeDoc](#) (void *data)
Free a xmlDoc pointer. For use in conjunction with `std::unique_ptr` for auto-deallocation.
- unsigned int [AtomProbe::XMLHelpNextType](#) (xmlNodePtr &node, int)
- unsigned int [AtomProbe::XMLHelpFwdToElem](#) (xmlNodePtr &node, const char *nodeName)
- unsigned int [AtomProbe::XMLHelpFwdNotElem](#) (xmlNodePtr &node, const char *nodeName)
- string [AtomProbe::XMLHelpGetText](#) (xmlNodePtr node)

7.91 src/io/dataFiles.cpp File Reference

```
#include "atomprobe/io/dataFiles.h"
#include "atomprobe/primitives/ionHit.h"
#include "atomprobe/helper/stringFuncs.h"
#include "atomprobe/helper/endianTest.h"
#include "atomprobe/helper/sampling.h"
#include "helper/helpFuncs.h"
#include "atomprobe/helper/aptAssert.h"
#include <vector>
#include <string>
#include <fstream>
#include <map>
#include <algorithm>
#include <sys/time.h>
```

Include dependency graph for dataFiles.cpp:



Classes

- struct [AtomProbe::IONHIT](#)
Record as stored in a .POS file.

Namespaces

- [AtomProbe](#)

Typedefs

- typedef struct [AtomProbe::IONHIT](#) [AtomProbe::IONHIT](#)
Record as stored in a .POS file.

Enumerations

- enum {
 [AtomProbe::POS_ALLOC_FAIL](#) =1, [AtomProbe::POS_OPEN_FAIL](#), [AtomProbe::POS_SIZE_MODULUS_←](#)
 [_ERR](#), [AtomProbe::POS_SIZE_EMPTY_ERR](#),
 [AtomProbe::POS_READ_FAIL](#), [AtomProbe::POS_NAN_LOAD_ERROR](#), [AtomProbe::POS_FILE_ENUM_←](#)
 [_END](#) }
- enum { [AtomProbe::TAPSIM_FILE_FORMAT_FAIL](#) =1, [AtomProbe::TAPSIM_OPEN_FAIL](#) }

Functions

- `template<class T >`
`unsigned int AtomProbe::fixedRecordReader (const char *filename, bool(*recordReader)(const char *buf←
Read, const char *destBuf), size_t recordSize, std::vector< T > &outputData)`
- `template<class T >`
`unsigned int AtomProbe::fixedRecordChunkReader (const char *filename, bool(*recordReader)(const char
*bufRead, const char *destBuf), size_t recordSize, std::vector< T > &outputData, unsigned int chunkSize,
unsigned int chunkOffset, unsigned int &nEntriesLeft)`
- `const char * AtomProbe::getPosFileErrString (unsigned int errMsg)`
- `unsigned int AtomProbe::loadPosFile (std::vector< IonHit > &posIons, const char *posFile)`
Load a pos file directly into a single ion list.
- `unsigned int AtomProbe::loadPosFile (std::vector< IonHit > &posIons, const char *posFile, unsigned int
nSamplesMax)`
As per loadPosFile, but with an additional setting for the maximum number of ions to load.
- `unsigned int AtomProbe::savePosFile (const std::vector< IonHit > &data, const char *name, bool ap-
pend=false)`
Save a vector of IonHits into a "pos" file, return nonzero on error.
- `unsigned int AtomProbe::savePosFile (const std::vector< Point3D > &points, float mass, const char *name,
bool append=false)`
Save a vector of Point3Ds into a pos file, using a fixed mass, return nonzero on error.
- `unsigned int AtomProbe::loadTapsimBinFile (vector< IonHit > &posIons, const char *posfile)`
- `unsigned int AtomProbe::saveTapsimBin (std::ostream &f, std::vector< IonHit > &posIons)`
Write a tapsim file from a bunch of IonHits.
- `bool AtomProbe::readEposRecord (const char *src, const char *dest)`
- `size_t AtomProbe::loadEposFile (std::vector< EPOS_ENTRY > &outData, const char *filename)`
Load an entire "EPOS" File.
- `size_t AtomProbe::chunkLoadEposFile (std::vector< EPOS_ENTRY > &outData, const char *filename, un-
signed int chunkSize, unsigned int chunkOffset, unsigned int &nEntriesLeft)`
Load an "EPOS" file, with a maximum chunk size.
- `unsigned int AtomProbe::readPosapOps (const char *file, THREEDAP_EXPERIMENT &data, unsigned int
&badLine, unsigned int &progress, std::atomic< bool > &wantAbort, unsigned int nDelayLines=2, bool
strictMode=false)`
Function to read POSAP "OPS" files.
- `unsigned int AtomProbe::loadATOFile (const char *fileName, std::vector< ATO_ENTRY > &ions, unsigned
int forceEndian=0)`
Load a LAWATAP "ATO" file.
- `bool AtomProbe::strhas (const char *cpTest, const char *cpPossible)`
- `unsigned int AtomProbe::loadTextData (const char *cpFilename, std::vector< std::vector< float > > &data←
Vec, std::vector< std::string > &headerVec, const char *delim, bool allowNan=true)`
Load a CSV, TSV or similar text file. Assumes "C" Locale for input (ie "." as decimal separator).

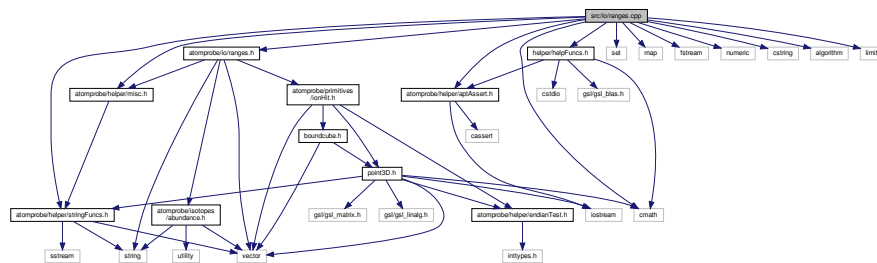
7.92 src/io/multiRange.cpp File Reference

```
#include "atomprobe/io/multiRange.h"
#include "atomprobe/helper/version.h"
#include "atomprobe/helper/XMLHelper.h"
#include "atomprobe/isotopes/overlaps.h"
#include "atomprobe/helper/aptAssert.h"
#include "helper/helpFuncs.h"
#include <algorithm>
#include <fstream>
```


7.93 src/io/ranges.cpp File Reference

```
#include "atomprobe/io/ranges.h"
#include "atomprobe/helper/stringFuncs.h"
#include "atomprobe/helper/misc.h"
#include "atomprobe/helper/aptAssert.h"
#include "helper/helpFuncs.h"
#include <set>
#include <map>
#include <fstream>
#include <numeric>
#include <cstring>
#include <algorithm>
#include <limits>
#include <cmath>
```

Include dependency graph for ranges.cpp:



Namespaces

- [AtomProbe](#)

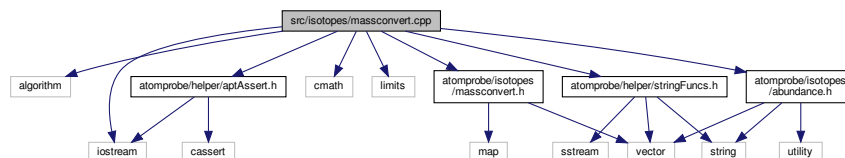
Functions

- string [AtomProbe::rangeToStr](#) (const pair< float, float > &rng)
- bool [AtomProbe::matchComposedName](#) (const std::map< string, size_t > &composedNames, const vector< pair< string, size_t > > &namesToFind, size_t &matchOffset)

Variables

- const size_t [AtomProbe::MAX_LINE_SIZE](#) = 16536
- const size_t [AtomProbe::MAX_RANGEFILE_SIZE](#) = 50*1024*1024
- const char * [AtomProbe::RANGE_EXTS](#) []
- const char * [AtomProbe::elementList](#) []


```
#include "atomprobe/isotopes/massconvert.h"
Include dependency graph for massconvert.cpp:
```



Namespaces

- [AtomProbe](#)

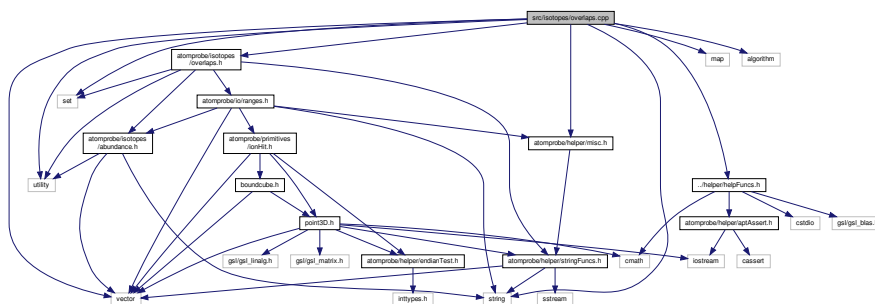
Functions

- void [AtomProbe::convertMolToMass](#) (const AbundanceData &atomTable, const std::map< unsigned int, double > &compositionMols, std::map< unsigned int, double > &compositionMass)
Convert the given composition from molar fraction data to mass fraction.
- void [AtomProbe::convertMassToMol](#) (const AbundanceData &atomTable, const std::map< unsigned int, double > &compositionMass, std::map< unsigned int, double > &compositionMols)
Convert the given composition from mass fraction data to molar fraction.
- unsigned int [AtomProbe::parseCompositionData](#) (const std::vector< std::string > &s, const AbundanceData &atomTable, std::map< unsigned int, double > &atomData)
Convert atomic string label data, in the form "Fe 0.81" to fraction map.

7.96 src/isotopes/overlaps.cpp File Reference

```
#include "atomprobe/isotopes/overlaps.h"
#include "atomprobe/helper/misc.h"
#include "../helper/helpFuncs.h"
#include <vector>
#include <map>
#include <utility>
#include <string>
#include <algorithm>
#include <set>
```

Include dependency graph for overlaps.cpp:



Namespaces

- [AtomProbe](#)

Macros

- `#define EQ_TOLV(f, g, h) (fabs((f) - (g)) < (h))`

Functions

- void [AtomProbe::findOverlapsFromSpectra](#) (const vector< vector< pair< float, float > > > &massDistributions, float massDelta, const set< unsigned int > &skipIDs, vector< pair< size_t, size_t > > &overlapIdx, vector< pair< float, float > > &overlapMasses)
- void [AtomProbe::findOverlaps](#) (const AbundanceData &natData, const RangeFile &rng, float massDelta, unsigned int maxCharge, vector< pair< size_t, size_t > > &overlapIdx, vector< pair< float, float > > &overlapMasses)
- void [AtomProbe::findOverlaps](#) (const AbundanceData &natData, const vector< string > &ionNames, float massDelta, unsigned int maxCharge, vector< pair< size_t, size_t > > &overlapIdx, vector< pair< float, float > > &overlapMasses)

7.96.1 Macro Definition Documentation

7.96.1.1 EQ_TOLV

```
#define EQ_TOLV(
    f,
    g,
    h ) (fabs( (f) - (g) ) < (h))
```

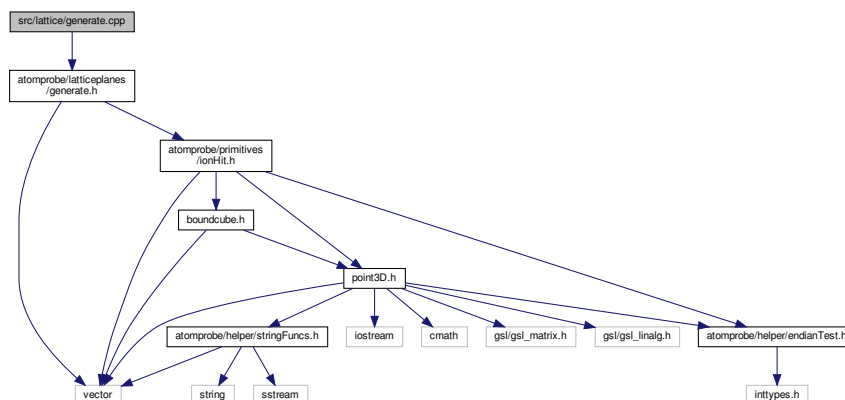
Definition at line 12 of file overlaps.cpp.

Referenced by `AtomProbe::findOverlapsFromSpectra()`.

7.97 src/lattice/generate.cpp File Reference

```
#include "atomprobe/latticeplanes/generate.h"
```

Include dependency graph for generate.cpp:



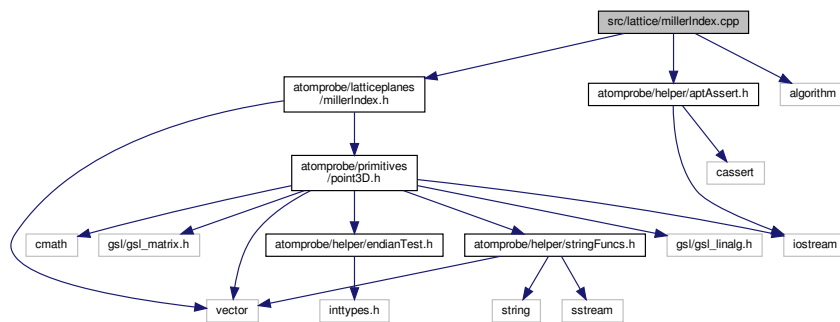
Namespaces

- [AtomProbe](#)

7.98 src/lattice/millerIndex.cpp File Reference

```
#include "atomprobe/latticeplanes/millerIndex.h"
#include "atomprobe/helper/aptAssert.h"
#include <algorithm>
```

Include dependency graph for millerIndex.cpp:



Namespaces

- [AtomProbe](#)

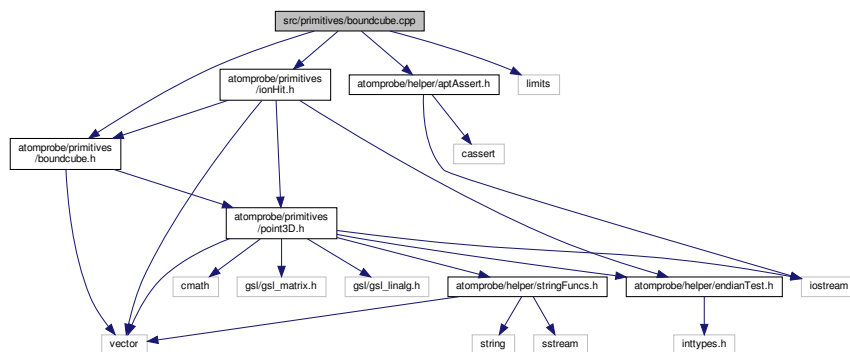
Functions

- int [AtomProbe::gcd](#) (int a, int b)

7.99 src/primitives/boundcube.cpp File Reference

```
#include "atomprobe/primitives/boundcube.h"
#include "atomprobe/primitives/ionHit.h"
#include "atomprobe/helper/aptAssert.h"
#include <limits>
```

Include dependency graph for boundcube.cpp:



Namespaces

- [AtomProbe](#)

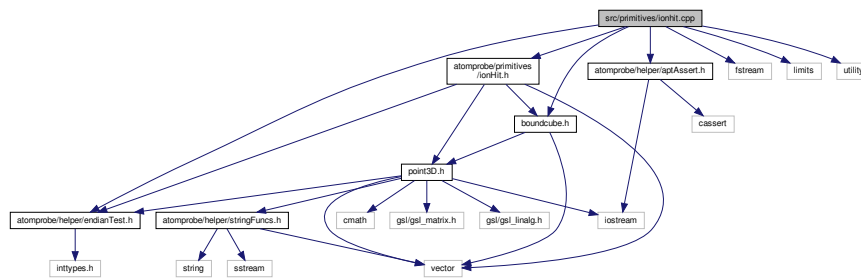
Functions

- `std::ostream & AtomProbe::operator<< (std::ostream &stream, const BoundCube &b)`

7.100 `src/primitives/ionhit.cpp` File Reference

```
#include "atomprobe/primitives/ionHit.h"
#include "atomprobe/primitives/boundcube.h"
#include "atomprobe/helper/aptAssert.h"
#include "atomprobe/helper/endianTest.h"
#include <fstream>
#include <limits>
#include <utility>
```

Include dependency graph for `ionhit.cpp`:



Namespaces

- [AtomProbe](#)

Functions

- `std::ostream & AtomProbe::operator<< (std::ostream &stream, const IonHit &ion)`

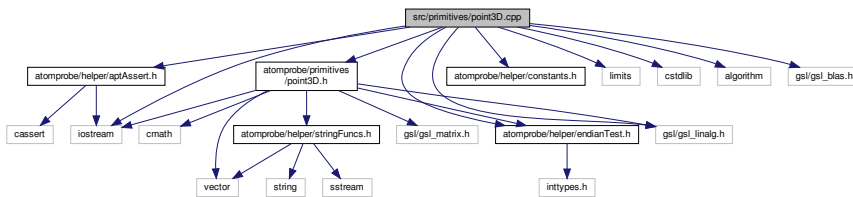
7.101 `src/primitives/mesh.cpp` File Reference

```
#include <atomprobe/primitives/mesh.h>
#include <atomprobe/helper/aptAssert.h>
#include <helper/helpFuncs.h>
#include "atomprobe/helper/math/misc.h"
#include <string>
#include <algorithm>
#include <limits>
```


7.102 src/primitives/point3D.cpp File Reference

```
#include "atomprobe/primitives/point3D.h"
#include "atomprobe/helper/EndianTest.h"
#include "atomprobe/helper/aptAssert.h"
#include "atomprobe/helper/constants.h"
#include <limits>
#include <cstdlib>
#include <iostream>
#include <algorithm>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_blas.h>
```

Include dependency graph for point3D.cpp:



Namespaces

- [AtomProbe](#)

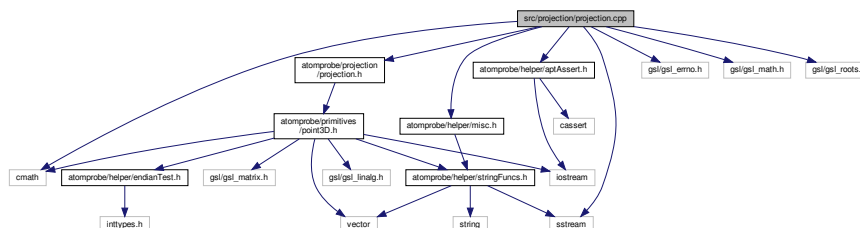
Functions

- void [AtomProbe::gsl_matrix_mult](#) (const [gsl_matrix](#) *a, const [gsl_matrix](#) *b, [gsl_matrix](#) *&res)
- [std::ostream](#) & [AtomProbe::operator<<](#) ([std::ostream](#) &stream, const [Point3D](#) &pt)

7.103 src/projection/projection.cpp File Reference

```
#include "atomprobe/projection/projection.h"
#include "atomprobe/helper/misc.h"
#include "atomprobe/helper/aptAssert.h"
#include <cmath>
#include <sstream>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_roots.h>
```

Include dependency graph for projection.cpp:



Classes

- struct [AtomProbe::SphericProjectionParams](#)

Namespaces

- [AtomProbe](#)

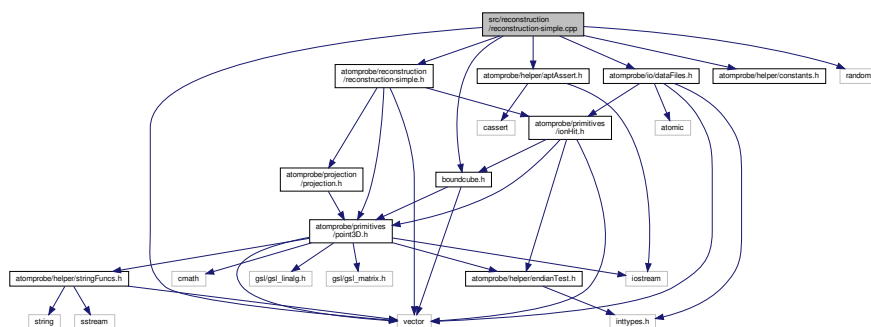
Functions

- double [AtomProbe::SphericProjectionEqn](#) (double eta, void *p)

7.104 src/reconstruction/reconstruction-simple.cpp File Reference

```
#include "atomprobe/reconstruction/reconstruction-simple.h"
#include "atomprobe/helper/aptAssert.h"
#include "atomprobe/primitives/boundcube.h"
#include "atomprobe/io/dataFiles.h"
#include "atomprobe/helper/constants.h"
#include <vector>
#include <random>
```

Include dependency graph for reconstruction-simple.cpp:



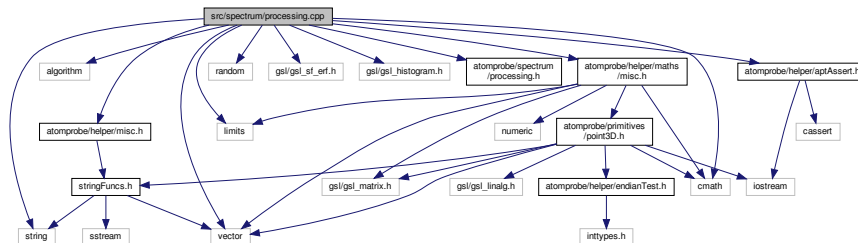
Namespaces

- [AtomProbe](#)

7.105 src/spectrum/processing.cpp File Reference

```
#include <vector>
#include <algorithm>
#include <cmath>
#include <limits>
#include <string>
#include <random>
#include <gsl/gsl_sf_erf.h>
#include <gsl/gsl_histogram.h>
#include "atomprobe/spectrum/processing.h"
#include "atomprobe/helper/misc.h"
#include "atomprobe/helper/maths/misc.h"
#include "atomprobe/helper/aptAssert.h"
```

Include dependency graph for processing.cpp:



Namespaces

- [AtomProbe](#)

Macros

- `#define USE_CENTRAL`

Functions

- `template<class T >`
`bool AtomProbe::andersonDarlingStatistic (std::vector< T > vals, float &meanV, float &stdevVal, float &statis-`
`tic, size_t &undefCount, bool computeMeanAndStdev=true)`
- `void AtomProbe::makeHistogram (const vector< float > &data, float start, float end, float step, vector< float`
`> &histVals)`
- `std::string AtomProbe::getFitErrorMsg (unsigned int errCode)`
- `unsigned int AtomProbe::doFitBackground (const std::vector< float > &massData, BACKGROUND_PAR↵`
`AMS ¶ms)`
Perform a background fit, assuming constant TOF noise, from 0->inf time.
- `void AtomProbe::createMassBackground (float massStart, float massEnd, unsigned int nBinsMass, float tof↵`
`BackIntensity, std::vector< float > &histogram)`
Build a histogram of the background counts.
- `void AtomProbe::diff (const vector< float > &in, vector< float > &out)`
- `void AtomProbe::findPeaks (const std::vector< float > &x0, std::vector< unsigned int > &peakInds, float sel,`
`bool autoSel=true, bool includeEndpoints=true)`
Simple peak-finding algorithm.

7.105.1 Macro Definition Documentation

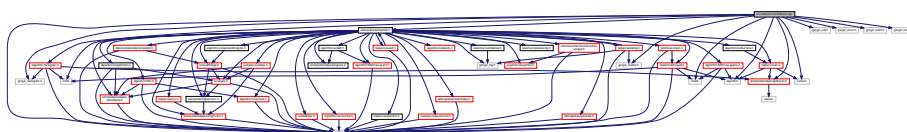
7.105.1.1 USE_CENTRAL

```
#define USE_CENTRAL
```

7.106 src/statistics/confidence.cpp File Reference

```
#include <algorithm>
#include <limits>
#include <cmath>
#include <vector>
#include <numeric>
#include "atomprobe/atomprobe.h"
#include "atomprobe/helper/aptAssert.h"
#include <gsl/gsl_cdf.h>
#include <gsl/gsl_randist.h>
#include <gsl/gsl_histogram.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_roots.h>
```

Include dependency graph for confidence.cpp:



Classes

- struct [AtomProbe::ZECH_ROOT](#)

Namespaces

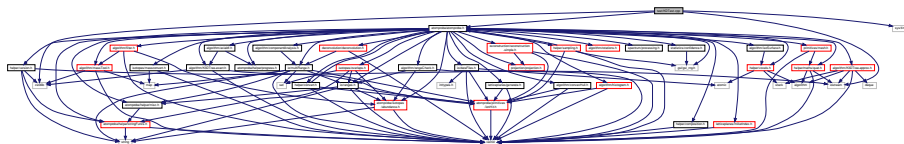
- [AtomProbe](#)

Functions

- `template<class T, class T2 >`
`std::iterator_traits< T >::value_type` `AtomProbe::kahansum` (T begin, T end, T2 other)
- `bool` `AtomProbe::numericalEstimatePoissRatioConf` (float lambda1, float lambda2, float alpha, unsigned int nTrials, `gsl_rng *r`, float &IBound, float &uBound)
Brute-force poisson ratio confidence estimator. Returns the confidence interval in the estimate of the mean, at the given rates.
- `bool` `AtomProbe::numericalEstimateSkellamConf` (float lambda1, float lambda2, float alpha, unsigned int nTrials, `gsl_rng *r`, float &IBound, float &uBound)
Brute-force the confidence bounds of a skellam distribution.
- `bool` `AtomProbe::numericalEstimateGaussRatioConf` (float mu1, float mu2, float var1, float var2, float alpha, unsigned int nTrials, `gsl_rng *r`, float &IBound, float &uBound)
Brute-force gaussian ratio confidence estimator.
- `double` `AtomProbe::zechRoot` (double sigGuess, void *params)
- `bool` `AtomProbe::zechConfidenceLimits` (float lambdaBack, unsigned int observation, float alpha, float &estimate)
Provides a best estimate for true signal when true signal, background.
- `template<class T >`
`void` `AtomProbe::cumTrapezoid` (const vector< T > &x, const vector< T > &vals, vector< T > &res)
- `double` `AtomProbe::gauss_ratio_pdf` (double x, double muX, double muY, double varX, double varY)
- `void` `AtomProbe::poissonConfidenceObservation` (float counts, float alpha, float &IBound, float &uBound)
Obtain poisson confidence limits for the mean of a poisson distribution.

7.107 test/KDTest.cpp File Reference

```
#include <iostream>
#include <cstdlib>
#include <sys/time.h>
#include "atomprobe/atomprobe.h"
Include dependency graph for KDTest.cpp:
```



Functions

- `bool` `callback` ()
- `void` `kdExactFuzz` ()
- `bool` `testInexactKDTree` ()
- `int` `main` ()

Variables

- `const unsigned int` `NUM_IONS` =1000
- `const float` `SCALE` =100
- `const float` `SEARCH_RAD` = 5
- `unsigned int` `progress` =0

7.107.1 Function Documentation

7.107.1.1 callback()

```
bool callback ( )
```

Definition at line 32 of file KDTest.cpp.

Referenced by kdExactFuzz().

7.107.1.2 kdExactFuzz()

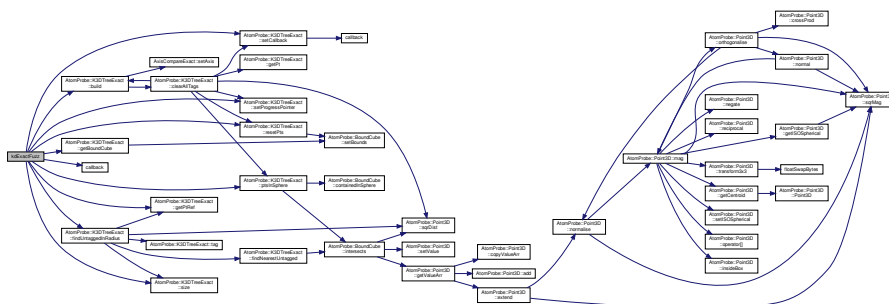
```
void kdExactFuzz ( )
```

Definition at line 37 of file KDTest.cpp.

References `AtomProbe::K3DTreeExact::build()`, `callback()`, `AtomProbe::K3DTreeExact::findUntaggedInRadius()`, `AtomProbe::K3DTreeExact::getBoundCube()`, `AtomProbe::K3DTreeExact::getPtRef()`, `NUM_IONS`, `progress`, `AtomProbe::K3DTreeExact::ptsInSphere()`, `AtomProbe::K3DTreeExact::resetPts()`, `SCALE`, `SEARCH_RAD`, `AtomProbe::K3DTreeExact::setCallback()`, `AtomProbe::K3DTreeExact::setProgressPointer()`, and `AtomProbe::K3DTreeExact::size()`.

Referenced by `main()`.

Here is the call graph for this function:



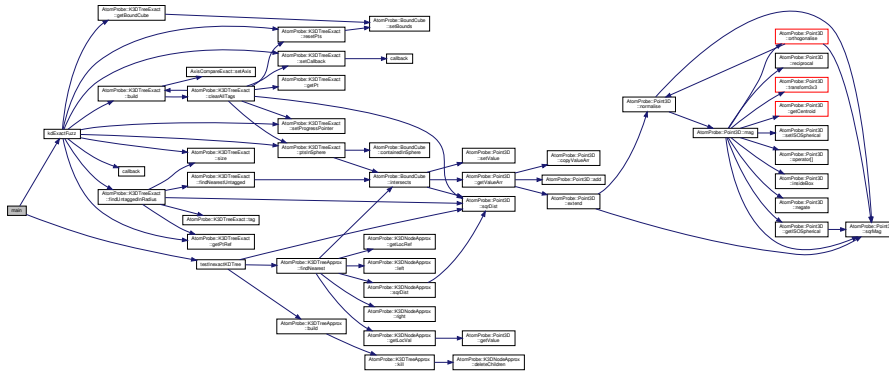
7.107.1.3 main()

```
int main ( )
```

Definition at line 125 of file KDTest.cpp.

References kdExactFuzz(), NUM_IONS, SCALE, SEARCH_RAD, and testInexactKDTree().

Here is the call graph for this function:



7.107.1.4 testInexactKDTree()

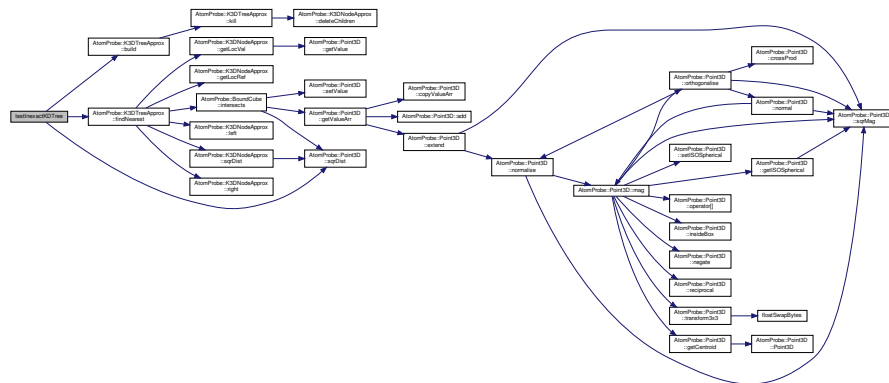
```
bool testInexactKDTree ( )
```

Definition at line 98 of file KDTest.cpp.

References AtomProbe::K3DTreeApprox::build(), AtomProbe::K3DTreeApprox::findNearest(), and AtomProbe::Point3D::sqrDist().

Referenced by main().

Here is the call graph for this function:



7.107.2 Variable Documentation

7.107.2.1 NUM_IONS

```
const unsigned int NUM_IONS =1000
```

Definition at line 26 of file KDTest.cpp.

Referenced by AtomProbe::findPeaks(), AtomProbe::GetReducedHullPts(), kdExactFuzz(), and main().

7.107.2.2 progress

```
unsigned int progress =0
```

Definition at line 30 of file KDTest.cpp.

Referenced by kdExactFuzz().

7.107.2.3 SCALE

```
const float SCALE =100
```

Definition at line 27 of file KDTest.cpp.

Referenced by AtomProbe::GetReducedHullPts(), kdExactFuzz(), and main().

7.107.2.4 SEARCH_RAD

```
const float SEARCH_RAD = 5
```

Definition at line 28 of file KDTest.cpp.

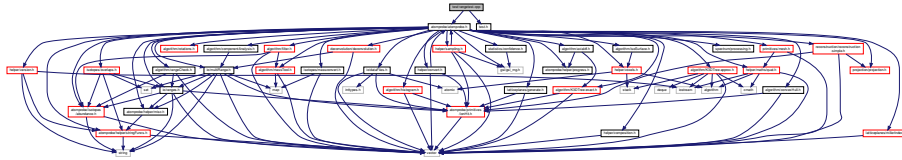
Referenced by kdExactFuzz(), and main().

7.108 test/rangetest.cpp File Reference

```
#include "atomprobe/atomprobe.h"
```

```
#include "test.h"
```

Include dependency graph for rangetest.cpp:



Functions

- int [main](#) ()

7.108.1 Function Documentation

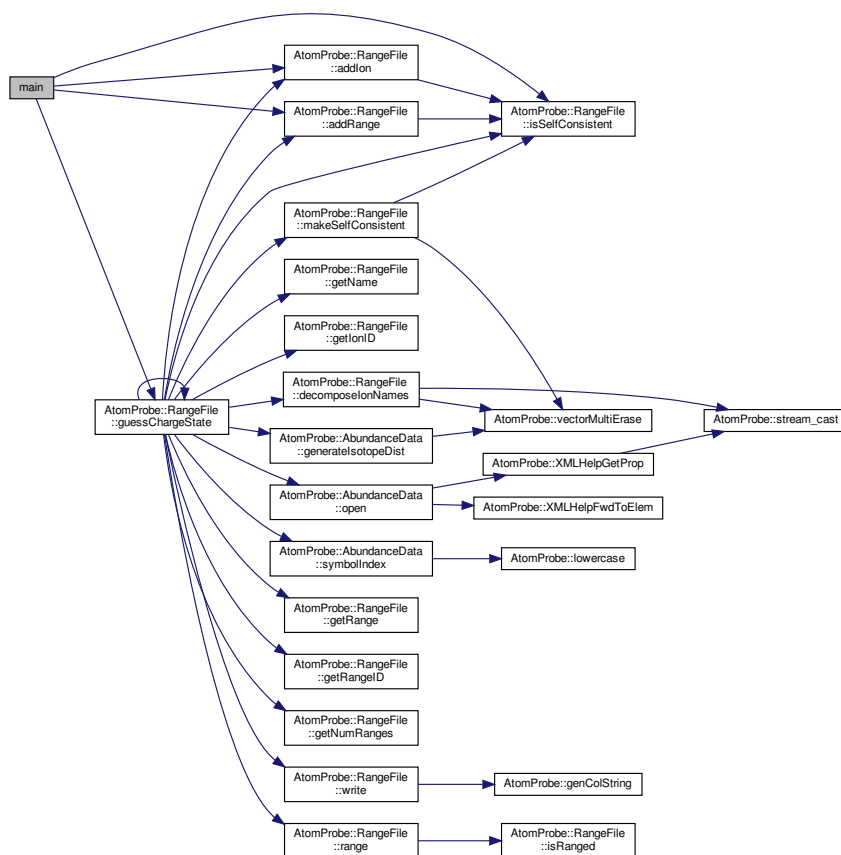
7.108.1.1 main()

```
int main ( )
```

Definition at line 9 of file rangetest.cpp.

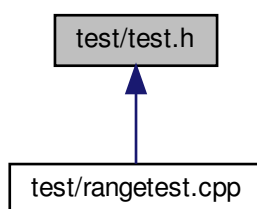
References [AtomProbe::RangeFile::addlon\(\)](#), [AtomProbe::RangeFile::addRange\(\)](#), [AtomProbe::RGBf::blue](#), [AtomProbe::RGBf::green](#), [AtomProbe::RangeFile::guessChargeState\(\)](#), [AtomProbe::RangeFile::isSelfConsistent\(\)](#), [AtomProbe::RGBf::red](#), and [TEST](#).

Here is the call graph for this function:



7.109 test/test.h File Reference

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



Macros

- `#define TEST(f, g) { if(!(f)) { cerr << "Test failed " << __FILE__ << ":" << __LINE__ << g << endl ; } }`

7.109.1 Macro Definition Documentation

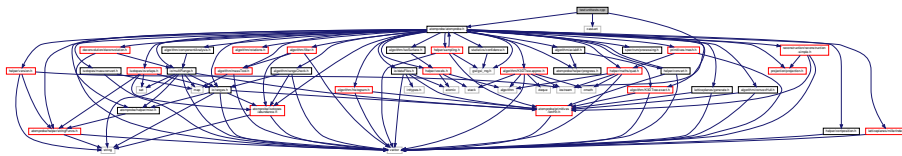
7.109.1.1 TEST

```
#define TEST(
    f,
    g ) { if(!(f)) { cerr << "Test failed " << __FILE__ << ":" << __LINE__ << g
<< endl ; } } ;
```

Definition at line 5 of file test.h.

7.110 test/unittests.cpp File Reference

```
#include <iostream>
#include <cassert>
#include "atomprobe/atomprobe.h"
Include dependency graph for unittests.cpp:
```



Macros

- #define [ASSERT](#)(f) assert((f));

Functions

- vector< string > [rangePaths](#) ()
- bool [runTests](#) ()
- int [main](#) ()

7.110.1 Macro Definition Documentation

7.110.1.1 ASSERT

```
#define ASSERT(
    f ) assert( (f) );
```

Definition at line 23 of file unittests.cpp.

Referenced by [rangePaths](#)().

7.110.2 Function Documentation

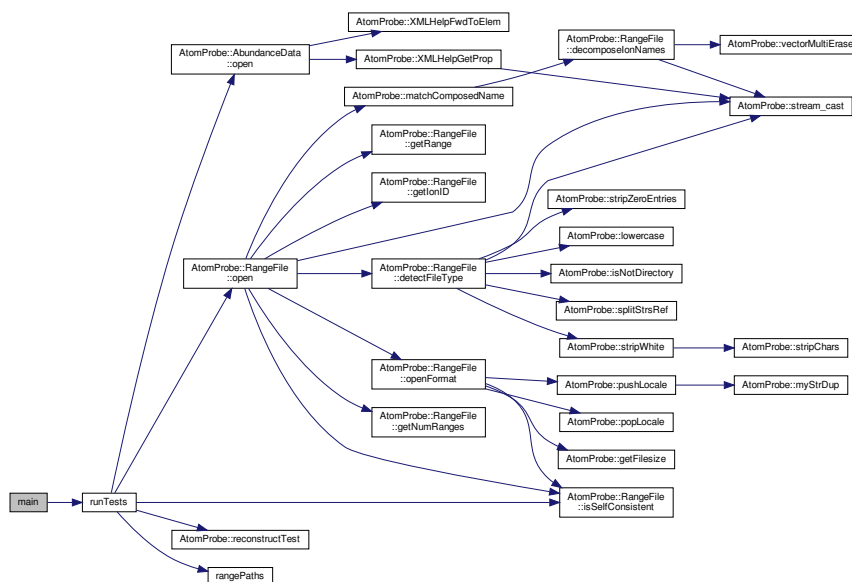
7.110.2.1 main()

```
int main ( )
```

Definition at line 203 of file unittests.cpp.

References `runTests()`.

Here is the call graph for this function:



7.110.2.2 rangePaths()

```
vector<string> rangePaths ( )
```

Definition at line 34 of file unittests.cpp.

References `ASSERT`.

Referenced by `runTests()`.

7.110.2.3 runTests()

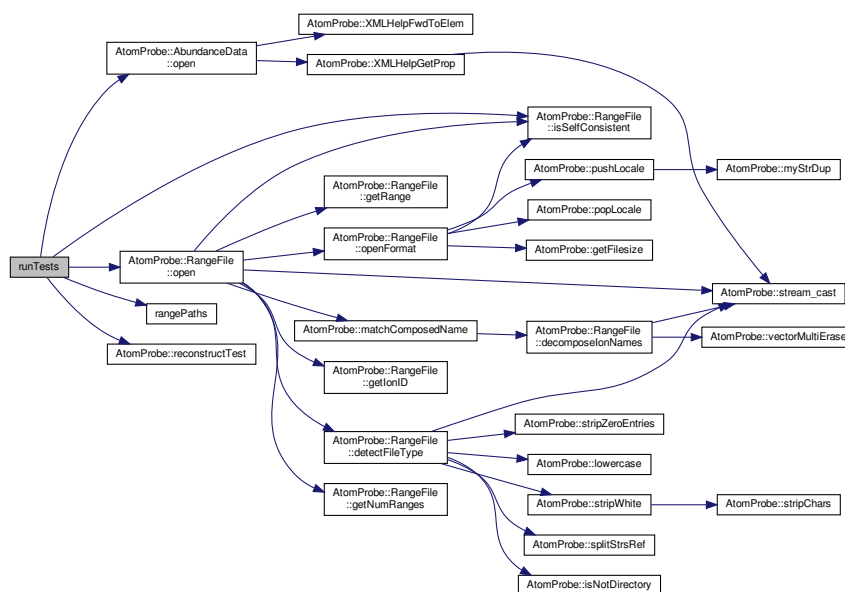
```
bool runTests ( )
```

Definition at line 70 of file unittests.cpp.

References `AtomProbe::RangeFile::isSelfConsistent()`, `AtomProbe::AbundanceData::open()`, `AtomProbe::RangeFile::open()`, `rangePaths()`, `AtomProbe::reconstructTest()`, and `TEST`.

Referenced by `main()`.

Here is the call graph for this function:



Index

- ~CrystalGen
 - AtomProbe::CrystalGen, [157](#)
- ~FixedStack
 - FixedStack, [167](#)
- ~K3DTreeApprox
 - AtomProbe::K3DTreeApprox, [192](#)
- ~K3DTreeExact
 - AtomProbe::K3DTreeExact, [198](#)
- ~ProgressBar
 - AtomProbe::ProgressBar, [279](#)
- ~RandNumGen
 - AtomProbe::RandNumGen, [284](#)
- ~Voxels
 - AtomProbe::Voxels, [349](#)
- a
 - AtomProbe::Quaternion, [282](#)
- a2iTriangleConnectionTable
 - AtomProbe, [110](#)
- ABUNDANCE_ERROR
 - AtomProbe, [110](#)
- ARCHITECTURE_ID
 - CMakeCXXCompilerId.cpp, [385](#)
- ARRAYSIZE
 - helpFuncs.h, [481](#)
- ASSERT
 - aptAssert.h, [429](#)
 - autocorrelate.cpp, [413](#)
 - math.cpp, [463](#)
 - maths.cpp, [482](#)
 - unittests.cpp, [510](#)
 - zechBackground.cpp, [409](#)
- ATO_ENTRY
 - AtomProbe, [21](#)
- ATO_ERR_STRINGS
 - AtomProbe, [110](#)
- abort
 - AtomProbe::ProgressBar, [279](#)
- abundance
 - AtomProbe::ISOTOPE_ENTRY, [184](#)
- abundanceBetweenLimits
 - AtomProbe::AbundanceData, [118](#)
- abundanceError
 - AtomProbe::ISOTOPE_ENTRY, [184](#)
- accumulateCorrelationHistogram
 - AtomProbe, [31](#)
- add
 - AtomProbe::Point3D, [258](#)
- addlon
 - AtomProbe::MultiRange, [241](#), [242](#)
 - AtomProbe::RangeFile, [289](#)
- addRange
 - AtomProbe::MultiRange, [242](#), [243](#)
 - AtomProbe::RangeFile, [289](#)
- aiCubeEdgeFlags
 - AtomProbe, [110](#)
- allowableWeights
 - WEIGHT_SEARCH_ENTRY, [381](#)
- alpha
 - AtomProbe::THREEDAP_DATA, [335](#)
 - AtomProbe::ZECH_ROOT, [382](#)
- andersonDarlingStatistic
 - AtomProbe, [32](#)
- angle
 - AtomProbe::Point3D, [259](#)
- antiRotateMatch
 - AtomProbe, [33](#)
- applyMask
 - AtomProbe::Voxels, [350](#)
- applyQuaternionRotation
 - AtomProbe, [33](#)
- approxPulse
 - AtomProbe::ATO_ENTRY, [128](#)
- aptAssert.h
 - ASSERT, [429](#)
 - TEST, [429](#)
 - WARN, [429](#)
- askAssert
 - AtomProbe, [33](#)
- AtomProbe, [11](#)
 - a2iTriangleConnectionTable, [110](#)
 - ABUNDANCE_ERROR, [110](#)
 - ATO_ENTRY, [21](#)
 - ATO_ERR_STRINGS, [110](#)
 - accumulateCorrelationHistogram, [31](#)
 - aiCubeEdgeFlags, [110](#)
 - andersonDarlingStatistic, [32](#)
 - antiRotateMatch, [33](#)
 - applyQuaternionRotation, [33](#)
 - askAssert, [33](#)
 - buildFrequencyTable, [33](#)
 - castVoxels, [33](#)
 - checkMassRangingCorrectness, [34](#)
 - chunkLoadEposFile, [34](#)
 - computeComposition, [35](#)
 - computeConvexHull, [35](#)
 - computeLonDistAdjacency, [36](#)
 - computeRangeAdjacency, [37](#)
 - computeRotationMatrix, [38](#)

- convertEPOStoPos, 39
- convertFileStringToCanonical, 39
- convertFileStringToNative, 39
- convertMassToMol, 40
- convertMolToMass, 40
- correlationHistogram, 41
- countDataDistanceWeight, 42
- countIntensityEvents, 43
- createMassBackground, 43
- cumTrapezoid, 44
- det3by3, 44
- Determinant, 44
- diff, 45
- digitString, 45
- distanceToFacet, 46
- distanceToSegment, 46
- doFitBackground, 47
- doHull, 48
- dotProduct, 48
- ELEM_FOUR_NODE_TETRAHEDRON, 111
- ELEM_SINGLE_NODE_POINT, 111
- ELEM_THREE_NODE_TRIANGLE, 111
- ELEM_TWO_NODE_LINE, 112
- EPOS_RECORD_SIZE, 112
- edgeldx, 49
- edgeRemap, 111
- elementList, 112
- estimateRank, 49
- filterBySolutionPPM, 50
- filterPeakNeedBiggerObs, 50
- findMaxLessThanOrEq, 51
- findNearVerticies, 51, 52
- findOverlaps, 52, 53
- findOverlapsFromSpectra, 54
- findPeaks, 54
- fixedRecordChunkReader, 55
- fixedRecordReader, 56
- fourDeterminant, 57
- fpeek, 57
- freeConvexHull, 57
- gauss_ratio_pdf, 58
- gcd, 58
- genColString, 58
- generate1DAxialDistHist, 59
- generate1DAxialDistHistSweep, 60
- getAtoErrString, 61
- getFileSize, 61
- getFitErrorMsg, 62
- getHardAssert, 62
- getPosFileErrString, 62
- getRangeMolecule, 62
- getRecordReadErrString, 63
- GetReducedHullPts, 63
- gsl_determinant, 64
- gsl_matrix_mult, 64
- gsl_print_matrix, 64
- gsl_print_vector, 65
- HULL_GRAB_SIZE, 113
- hardAssert, 112
- hexStrToUChar, 65
- IONHIT, 21
- incrementCorrelationHist, 65
- incrementDataDistanceWeight, 65
- intersect_RayTriangle, 66
- isNotDirectory, 67
- kahansum, 67
- leastSquaresDeconvolve, 68
- libVersion, 113
- linearHistogram, 68
- linkIdentifiers, 69
- loadAtoFile, 69
- loadEposFile, 70
- loadPosFile, 70, 71
- loadTapsimBinFile, 72
- loadTextData, 73
- lowercase, 73
- MAX_LINE_SIZE, 113
- MAX_RANGEFILE_SIZE, 113
- MESH_LOAD_ERRS, 114
- MULTIRANGE_FORMAT_VERSION, 114
- makeHistogram, 73
- marchingCubes, 74
- matchComposedName, 75
- maxExplainedFraction, 75, 76
- maximumLinearTable, 113
- meanAndStdev, 77
- myStrDup, 77
- NUM_ELEMENTS, 114
- nullBackground, 78
- nullifyMarker, 78
- numericalEstimateGaussRatioConf, 78
- numericalEstimatePoissRatioConf, 79
- numericalEstimateSkellamConf, 79
- OPS_ENUM_ERRSTRINGS, 114
- onlyDir, 80
- onlyFilename, 80
- operator<, 80
- operator<<, 81
- PROGRESS_REDUCE, 115
- pairContains, 82
- pairOverlaps, 82
- parseColString, 82
- parseCompositionData, 83
- PointDir, 31
- poissonConfidenceObservation, 84
- popLocale, 84
- pushLocale, 85
- pyramidVol, 85
- qhullInited, 115
- quat_get_rot_quat, 85
- quat_invert, 86
- quat_mult_no_second_a, 86
- quat_pointmult, 86
- quat_rot, 87
- quat_rot_apply_quat, 88
- quat_rot_array, 88, 89

- RANGE_EXTS, 115
- RECORDREAD_ERR_STRINGS, 116
- randGen, 115
- randomIndices, 90
- randomSelect, 90
- rangeOverlaps, 90
- rangeToStr, 91
- readEposRecord, 91
- readPosapOps, 92
- reconstructTest, 93
- removeElements, 93
- rotateMatch, 93
- sampleIons, 93
- savePosFile, 94
- saveTapsimBin, 95
- selectElements, 95
- setHardAssert, 96
- signVal, 97
- signedDistanceToFacet, 96
- solveLeastSquares, 97
- SphericProjectionEqn, 97
- splitStrsRef, 98
- stlStrToStlWStr, 98
- stlWStrToStlStr, 99
- strAppend, 99
- stream_cast, 100
- strhas, 100
- stripChars, 101
- stripWhite, 101
- stripZeroEntries, 101
- sumVoxels, 102
- tabs, 102
- toEqual, 102
- transposeVector, 103
- trilsDegenerate, 103
- ucharToHexStr, 104
- uppercase, 104
- VERTEX_OFFSET, 116
- vectorMultiErase, 104
- vectorPointDir, 105
- weightedMean, 105
- XMLFreeDoc, 105
- XMLGetNextElemAttrib, 106
- XMLHelpFwdNotElem, 106
- XMLHelpFwdToElem, 106
- XMLHelpFwdToList, 107
- XMLHelpGetProp, 107
- XMLHelpGetText, 107, 108
- XMLHelpNextType, 108
- zechConfidenceLimits, 108
- zechRoot, 109
- AtomProbe::ATO_ENTRY, 128
 - approxPulse, 128
 - detectorX, 128
 - detectorY, 128
 - mass, 128
 - pulseVoltage, 129
 - tof, 129
 - voltage, 129
 - x, 129
 - y, 129
 - z, 129
- AtomProbe::AbundanceData, 117
 - abundanceBetweenLimits, 118
 - elementName, 119
 - elementNames, 119
 - generateGroupedIsotopeDist, 120
 - generateIsotopeDist, 121
 - generateSingleAtomDist, 121
 - getAtomicNumber, 122
 - getErrorText, 122
 - getMajorIsotopeFromElemIdx, 122
 - getNearestCharge, 122
 - getNominalMass, 123
 - getSymbolIndices, 123, 124
 - isotope, 124
 - isotopeIndex, 124, 125
 - isotopes, 125
 - numElements, 125
 - numIsotopes, 126
 - open, 126
 - symbolIdxFromAtomicNumber, 126
 - symbolIndex, 127
- AtomProbe::AxisCompare, 130
 - AxisCompare, 130
 - operator(), 130
 - setAxis, 131
- AtomProbe::BACKGROUND_PARAMS, 132
 - binWidth, 133
 - intensity, 133
 - massEnd, 133
 - massStart, 133
 - mode, 134
 - stdev, 134
- AtomProbe::BodyCentredCubicGen, 134
 - BodyCentredCubicGen, 135
 - generateLattice, 136
- AtomProbe::BoundCube, 136
 - BoundCube, 138–140
 - containedInSphere, 140
 - contains, 140
 - containsPt, 141
 - expand, 141, 142
 - getBound, 142
 - getBounds, 142
 - getCentroid, 143
 - getLargestDim, 143
 - getMaxDistanceToBox, 144
 - getSize, 144
 - getVolume, 145
 - intersects, 145, 146
 - isFlat, 146
 - K3DTreeApprox, 153
 - K3DTreeExact, 153
 - makeIntersection, 147
 - makeUnion, 147

- max, 148
- min, 148
- operator<<, 153
- operator=, 149
- operator==, 149
- segmentTriple, 149
- setBound, 149
- setBounds, 150–152
- setInverseLimits, 152
- setLimits, 152
- AtomProbe::ComparePairFirst, 153
 - operator(), 154
- AtomProbe::ComparePairFirstReverse, 154
 - operator(), 154
- AtomProbe::ComparePairSecond, 155
 - operator(), 156
- AtomProbe::CrystalGen, 156
 - ~CrystalGen, 157
 - CrystalGen, 157
 - extractPositions, 158
 - farPoint, 159
 - generateLattice, 158
 - localData, 159
 - mirrorOut, 158
 - swap, 159
- AtomProbe::EPOS_ENTRY, 160
 - deltaPulse, 161
 - getIonHit, 160, 161
 - hitMultiplicity, 162
 - massToCharge, 162
 - operator==, 161
 - timeOfFlight, 162
 - voltDC, 162
 - voltPulse, 162
 - x, 163
 - xDetector, 163
 - y, 163
 - yDetector, 163
 - z, 163
- AtomProbe::FLATTENED_RANGE, 169
 - containedIonIDs, 169
 - containedRangeIDs, 169
 - endMass, 170
 - startMass, 170
- AtomProbe::FaceCentredCubicGen, 164
 - FaceCentredCubicGen, 165
 - generateLattice, 166
- AtomProbe::GnomonicProjection, 170
 - scaleDown, 171
 - scaleUp, 171
 - toAzimuthal, 172
 - toPlanar, 172
- AtomProbe::IONHIT, 182
 - massToCharge, 182
 - pos, 182
- AtomProbe::ISOTOPE_ENTRY, 183
 - abundance, 184
 - abundanceError, 184
 - atomicNumber, 184
 - mass, 184
 - massError, 184
 - massNumber, 185
 - symbol, 185
- AtomProbe::IonHit, 173
 - getBoundCube, 175
 - getCentroid, 175
 - getIonDataLimits, 176
 - getMassToCharge, 176
 - getPoints, 177
 - getPos, 177
 - getPosRef, 178
 - hasNaN, 178
 - IonHit, 174
 - operator<<, 182
 - operator+, 179
 - operator=, 179
 - operator==, 179
 - operator[], 179
 - setHit, 180
 - setMassToCharge, 180
 - setPos, 180, 181
- AtomProbe::K3DNodeApprox, 185
 - deleteChildren, 186
 - dump, 186
 - getAxis, 187
 - getAxisVal, 187
 - getLoc, 187
 - getLocRef, 188
 - getLocVal, 188
 - left, 188
 - right, 189
 - setAxis, 189
 - setLeft, 189
 - setLoc, 189
 - setRight, 190
 - sqrDist, 190
- AtomProbe::K3DNodeExact, 191
 - childLeft, 191
 - childRight, 191
 - tagged, 191
- AtomProbe::K3DTreeApprox, 192
 - ~K3DTreeApprox, 192
 - build, 193
 - buildByRef, 193
 - dump, 194
 - findKNearest, 195
 - findNearest, 195
 - K3DTreeApprox, 192
 - kill, 196
 - nodeCount, 197
- AtomProbe::K3DTreeExact, 197
 - ~K3DTreeExact, 198
 - build, 199
 - clear, 199
 - clearAllTags, 199
 - clearTags, 200

- dump, [200](#)
- findNearestUntagged, [200](#)
- findUntaggedInRadius, [201](#)
- getBoundCube, [202](#)
- getOrigIndex, [202](#)
- getPt, [202](#)
- getPtRef, [203](#)
- getTag, [203](#)
- K3DTreeExact, [198](#)
- ptsInSphere, [203](#)
- resetPts, [204](#)
- rootIdx, [205](#)
- setCallback, [205](#)
- setProgressPointer, [205](#)
- size, [206](#)
- tag, [206](#)
- tagCount, [206](#)
- AtomProbe::LINE, [210](#)
 - p, [210](#)
 - physGroup, [210](#)
- AtomProbe::LibVersion, [207](#)
 - checkDebug, [208](#)
 - getMajor, [208](#)
 - getMinor, [208](#)
 - getRevision, [209](#)
 - getVersionStr, [209](#)
 - LibVersion, [207](#)
- AtomProbe::LinearFeedbackShiftReg, [210](#)
 - clock, [211](#)
 - setMaskPeriod, [211](#)
 - setState, [211](#)
 - verifyTable, [212](#)
- AtomProbe::MILLER_TRIPLET, [231](#)
 - h, [232](#)
 - k, [232](#)
 - l, [232](#)
 - MILLER_TRIPLET, [232](#)
 - operator==, [232](#)
 - simplify, [233](#)
 - tripletToNormal, [233](#)
- AtomProbe::MassTool, [213](#)
 - bruteKnapsack, [213](#), [214](#)
- AtomProbe::Mesh, [215](#)
 - clear, [217](#)
 - countTriNodes, [217](#)
 - divideMeshSurface, [217](#)
 - elementCount, [218](#)
 - erasePhysGroup, [218](#)
 - getBounds, [218](#)
 - getContainedNodes, [219](#)
 - getCurPhysGroups, [219](#)
 - getIntersectingPrimitives, [220](#)
 - getNearestTri, [220](#)
 - getTriNormal, [220](#)
 - getVolume, [221](#)
 - isOrientedCoherently, [222](#)
 - isSane, [222](#)
 - killOrphanNodes, [222](#)
 - lines, [230](#)
 - loadGmshMesh, [223](#)
 - mergeDuplicateVertices, [224](#)
 - nodes, [230](#)
 - numDupTris, [224](#)
 - numDupVertices, [224](#)
 - physGroupNames, [230](#)
 - points, [230](#)
 - pointsInside, [225](#)
 - print, [225](#)
 - reassignGroups, [226](#)
 - removeDuplicateTris, [226](#)
 - removeStrayTris, [226](#)
 - resurface, [227](#)
 - rotate, [227](#)
 - saveGmshMesh, [227](#)
 - scale, [228](#)
 - setTriangleMesh, [228](#)
 - tetrahedra, [230](#)
 - translate, [229](#)
 - triangles, [231](#)
- AtomProbe::ModifiedFocusSphericProjection, [234](#)
 - etaToTheta, [235](#)
 - getFOVRadius, [236](#)
 - getMaxFOV, [236](#)
 - ModifiedFocusSphericProjection, [235](#)
 - scaleDown, [236](#)
 - scaleUp, [237](#)
 - setFocus, [237](#)
 - thetaToEta, [238](#)
 - toAzimuthal, [238](#)
 - toPlanar, [238](#)
- AtomProbe::MultiRange, [239](#)
 - addlon, [241](#), [242](#)
 - addRange, [242](#), [243](#)
 - clear, [243](#)
 - copyDataFromRange, [243](#)
 - flattenToMassAxis, [244](#)
 - getColour, [245](#)
 - getErrString, [245](#)
 - getlonID, [245](#)
 - getlonName, [245](#)
 - getMolecule, [246](#)
 - getNumIons, [246](#)
 - getNumRanges, [246](#)
 - getRange, [247](#)
 - getRangeldsFromlon, [247](#)
 - getRanges, [247](#)
 - isRanged, [247](#), [248](#)
 - isSelfConsistent, [248](#)
 - MultiRange, [240](#), [241](#)
 - open, [249](#)
 - range, [250](#)
 - setColour, [250](#)
 - setlonID, [250](#)
 - setRangeGroups, [250](#)
 - splitOverlapping, [251](#)
 - write, [251](#)

- AtomProbe::OVERLAP_PROBLEM_SETTINGS, 254
 - intensityTolerance, 254
 - massTolerance, 254
 - maxDefaultCharge, 255
- AtomProbe::Point3D, 255
 - add, 258
 - angle, 259
 - copyValueArr, 259
 - crossProd, 260
 - dotProd, 260
 - extend, 260
 - getCentroid, 261
 - getISOSpherical, 262
 - getValue, 262
 - getValueArr, 263
 - hasNaN, 263
 - insideBox, 264, 265
 - mag, 265
 - negate, 266
 - normal, 266
 - normalise, 267
 - operator<<, 276
 - operator*, 268
 - operator*=:, 268
 - operator+, 269
 - operator+=, 269
 - operator-, 269, 270
 - operator-=, 270
 - operator/, 270
 - operator=, 271
 - operator==, 271
 - operator[], 271
 - orthogonalise, 272
 - parse, 272
 - Point3D, 257, 258
 - reciprocal, 273
 - setISOSpherical, 274
 - setValue, 274
 - setValueArr, 275
 - sqrDist, 275
 - sqrMag, 275
 - transform3x3, 276
- AtomProbe::Point3f, 277
 - fx, 277
 - fy, 277
 - fz, 277
- AtomProbe::ProgressBar, 278
 - ~ProgressBar, 279
 - abort, 279
 - finish, 279
 - init, 280
 - ProgressBar, 278
 - reset, 280
 - setLength, 280
 - update, 281
- AtomProbe::Quaternion, 282
 - a, 282
 - b, 282
 - c, 282
 - d, 283
- AtomProbe::RANGE_MOLECULE, 285
 - components, 285
 - isOK, 285
 - name, 286
- AtomProbe::RGBf, 320
 - blue, 322
 - fromHex, 321
 - green, 322
 - red, 322
 - toHex, 321
- AtomProbe::RandNumGen, 283
 - ~RandNumGen, 284
 - getRng, 284
 - RandNumGen, 283
- AtomProbe::RangeFile, 286
 - addlon, 289
 - addRange, 289
 - decomposeLonById, 290
 - decomposeLonNames, 290
 - detectFileType, 291
 - eraselon, 292
 - eraseRange, 292
 - extensionIsRange, 293
 - extractlons, 293
 - getAllExts, 293
 - getColour, 294
 - getErrState, 294
 - getErrString, 295
 - getlonID, 295–297
 - getName, 298, 299
 - getNumlons, 299
 - getNumRanges, 299, 300
 - getRange, 300
 - getRangeByRef, 300
 - getRangeID, 301
 - getRangeVolume, 301
 - guessChargeState, 301
 - haveRangeVolumes, 302
 - isRanged, 303
 - isSelfConsistent, 304
 - makeSelfConsistent, 305
 - moveBothRanges, 306
 - moveRange, 306
 - open, 306
 - openFormat, 307
 - operator=, 308
 - range, 308
 - rangeByID, 309
 - rangeByRangeID, 309
 - RangeFile, 289
 - rangeInvertable, 309
 - rangeTypeString, 310
 - setColour, 311
 - setEnforceConsistent, 311
 - setlonID, 312
 - setlonLongName, 312

- setIonShortName, 312
- setRangeEnd, 313
- setRangeStart, 313
- setRangeVolume, 314
- swap, 314
- write, 314, 315
- AtomProbe::ReconstructionSphereOnCone, 316
 - reconstruct, 317
 - ReconstructionSphereOnCone, 316
 - setDetectorEfficiency, 318
 - setFlightPath, 318
 - setInitialRadius, 319
 - setProjModel, 319
 - setRadiusEvolutionMode, 319
 - setReconFOV, 319
 - setShankAngle, 320
- AtomProbe::SIMPLE_SPECIES, 322
 - atomicNumber, 323
 - count, 323
 - operator==, 323
- AtomProbe::SINGLE_HIT, 326
 - tof, 327
 - x, 327
 - y, 327
- AtomProbe::SimpleCubicGen, 324
 - generateLattice, 326
 - SimpleCubicGen, 325
- AtomProbe::SphericPlaneProjection, 328
 - scaleDown, 328
 - scaleUp, 328
 - toAzimuthal, 329
 - toPlanar, 329
- AtomProbe::SphericProjectionParams, 330
 - focusDist, 330
 - theta, 330
- AtomProbe::StereographicProjection, 331
 - scaleDown, 332
 - scaleUp, 332
 - thetaToEta, 332
 - toAzimuthal, 333
 - toPlanar, 333
- AtomProbe::TETRAHEDRON, 334
 - p, 334
 - physGroup, 334
- AtomProbe::THREEDAP_DATA, 334
 - alpha, 335
 - beta, 335
 - detectorChannels, 335
 - detectorRadius, 336
 - flightPath, 336
 - tZero, 336
- AtomProbe::THREEDAP_EXPERIMENT, 337
 - eventData, 337
 - eventPulseNumber, 337
 - params, 338
 - voltageData, 338
- AtomProbe::TRIANGLE, 338
 - edgesMismatch, 339
 - isSane, 339
 - p, 340
 - physGroup, 340
- AtomProbe::TriangleWithIndexedVertices, 340
 - p, 340
- AtomProbe::TriangleWithVertexNorm, 341
 - computeACWNormal, 342
 - computeArea, 342
 - getCentroid, 343
 - isDegenerate, 343
 - normal, 344
 - p, 344
 - safeComputeACWNormal, 343
- AtomProbe::VOLTAGE_DATA, 345
 - beta, 345
 - nextHitGroupOffset, 345
 - pulseVolt, 346
 - voltage, 346
- AtomProbe::Voxels
 - ~Voxels, 349
 - applyMask, 350
 - binarise, 350
 - calculateDensity, 351
 - clear, 352
 - convolve, 353
 - copy, 353
 - count, 353
 - countPoints, 353
 - deprecatedGetEdgeUniqueIndex, 354
 - fill, 355
 - getAxisBounds, 355
 - getBinVolume, 355
 - getBounds, 355, 356
 - getCellUniqueEdgeIndex, 356
 - getData, 357
 - getEdgeCell, 358
 - getEdgeEndApproxVals, 358
 - getEdgeEnds, 359
 - getIndex, 360
 - getIndexWithUpper, 361
 - getInterpSlice, 362
 - getInterpolatedData, 361
 - getMaxBounds, 362
 - getMinBounds, 363
 - getOffset, 363
 - getPitch, 363
 - getPoint, 363
 - getPointData, 364
 - getSize, 364
 - getSlice, 364
 - getSum, 365
 - init, 365, 366
 - loadFile, 366
 - max, 367
 - min, 367
 - minMax, 368
 - operator/=: 368
 - operator==, 369

- rebin, 369
- resize, 369
- resizeKeepData, 370
- separableConvolve, 371
- setBounds, 371
- setData, 372
- setEntry, 373
- setPoint, 374
- size, 375
- sizeofType, 375
- swap, 375
- threshold, 376
- thresholdForPosition, 376
- thresholdToBoolMask, 377
- trapezIntegral, 378
- Voxels, 349
- writeFile, 378
- AtomProbe::Voxels< T >, 346
- AtomProbe::Weight, 379
 - mass, 380
 - operator<, 380
 - operator==, 380
 - uniqueId, 380
 - Weight, 379
- AtomProbe::ZECH_ROOT, 382
 - alpha, 382
 - lambdaBack, 383
 - observation, 383
- atomicNumber
 - AtomProbe::ISOTOPE_ENTRY, 184
 - AtomProbe::SIMPLE_SPECIES, 323
- autocorrelate.cpp
 - ASSERT, 413
 - callback, 413
 - dumpAutocorrelation, 413
 - main, 414
 - PROGRESS_BAR_SIZE, 415
 - treeProgressBar, 415
 - treeProgressValue, 416
- AxisCompare
 - AtomProbe::AxisCompare, 130
- AxisCompareExact, 131
 - operator(), 131
 - setAxis, 132
- b
 - AtomProbe::Quaternion, 282
- beta
 - AtomProbe::THREEDAP_DATA, 335
 - AtomProbe::VOLTAGE_DATA, 345
- binWidth
 - AtomProbe::BACKGROUND_PARAMS, 133
- binarise
 - AtomProbe::Voxels, 350
- blue
 - AtomProbe::RGBf, 322
- BodyCentredCubicGen
 - AtomProbe::BodyCentredCubicGen, 135
- BoundCube
 - AtomProbe::BoundCube, 138–140
- bruteKnapsack
 - AtomProbe::MassTool, 213, 214
- build
 - AtomProbe::K3DTreeApprox, 193
 - AtomProbe::K3DTreeExact, 199
- buildByRef
 - AtomProbe::K3DTreeApprox, 193
- buildFrequencyTable
 - AtomProbe, 33
- c
 - AtomProbe::Quaternion, 282
- CMakeCXXCompilerId.cpp
 - ARCHITECTURE_ID, 385
 - COMPILER_ID, 385
 - CXX_STD, 386
 - DEC, 386
 - HEX, 386
 - info_arch, 387
 - info_compiler, 387
 - info_language_dialect_default, 388
 - info_platform, 388
 - main, 387
 - PLATFORM_ID, 386
 - STRINGIFY_HELPER, 387
 - STRINGIFY, 387
- CMakeFiles/3.10.2/CompilerIdCXX/CMakeCXXCompilerId.cpp, 385
- CMakeFiles/CheckTypeSize/SIZEOF_SIZE_T.cpp, 388
- COMPILER_ID
 - CMakeCXXCompilerId.cpp, 385
- CXX_STD
 - CMakeCXXCompilerId.cpp, 386
- calculateDensity
 - AtomProbe::Voxels, 351
- callback
 - autocorrelate.cpp, 413
 - KDTest.cpp, 505
 - kd-example.cpp, 393
- castVoxels
 - AtomProbe, 33
- checkDebug
 - AtomProbe::LibVersion, 208
- checkMassRangingCorrectness
 - AtomProbe, 34
- childLeft
 - AtomProbe::K3DNodeExact, 191
- childRight
 - AtomProbe::K3DNodeExact, 191
- chunkLoadEposFile
 - AtomProbe, 34
- clear
 - AtomProbe::K3DTreeExact, 199
 - AtomProbe::Mesh, 217
 - AtomProbe::MultiRange, 243
 - AtomProbe::Voxels, 352
- clearAllTags
 - AtomProbe::K3DTreeExact, 199

- clearTags
 - AtomProbe::K3DTreeExact, 200
- clock
 - AtomProbe::LinearFeedbackShiftReg, 211
- components
 - AtomProbe::RANGE_MOLECULE, 285
- computeACWNormal
 - AtomProbe::TriangleWithVertexNorm, 342
- computeArea
 - AtomProbe::TriangleWithVertexNorm, 342
- computeComposition
 - AtomProbe, 35
- computeConvexHull
 - AtomProbe, 35
- computeLonDistAdjacency
 - AtomProbe, 36
- computeMeanAndVariance
 - math.h, 465
- computeRangeAdjacency
 - AtomProbe, 37
- computeRotationMatrix
 - AtomProbe, 38
- containedInSphere
 - AtomProbe::BoundCube, 140
- containedLonIDs
 - AtomProbe::FLATTENED_RANGE, 169
- containedRangeIDs
 - AtomProbe::FLATTENED_RANGE, 169
- contains
 - AtomProbe::BoundCube, 140
- containsPt
 - AtomProbe::BoundCube, 141
- convert.cpp
 - convertEPOStoPos, 478
- convertEPOStoPos
 - AtomProbe, 39
 - convert.cpp, 478
- convertFileStringToCanonical
 - AtomProbe, 39
- convertFileStringToNative
 - AtomProbe, 39
- convertMassToMol
 - AtomProbe, 40
- convertMolToMass
 - AtomProbe, 40
- convertToPos
 - pulse-filter.cpp, 404
- convolve
 - AtomProbe::Voxels, 353
- copy
 - AtomProbe::Voxels, 353
- copyDataFromRange
 - AtomProbe::MultiRange, 243
- copyValueArr
 - AtomProbe::Point3D, 259
- correlationHistogram
 - AtomProbe, 41
- correlationhist.cpp
 - dumpHistogram, 390
 - main, 390
- count
 - AtomProbe::SIMPLE_SPECIES, 323
 - AtomProbe::Voxels, 353
- countDataDistanceWeight
 - AtomProbe, 42
- countIntensityEvents
 - AtomProbe, 43
- countPoints
 - AtomProbe::Voxels, 353
- countTriNodes
 - AtomProbe::Mesh, 217
- countepos.cpp
 - main, 391
- createMassBackground
 - AtomProbe, 43
- crossProd
 - AtomProbe::Point3D, 260
- CrystalGen
 - AtomProbe::CrystalGen, 157
- cube
 - NodeWalk, 253
- cumTrapezoid
 - AtomProbe, 44
- cumulativeWeight
 - WEIGHT_SEARCH_ENTRY, 381
- curWeights
 - WEIGHT_SEARCH_ENTRY, 382
- d
 - AtomProbe::Quaternion, 283
- DEC
 - CMakeCXXCompilerId.cpp, 386
- decomposeById
 - AtomProbe::RangeFile, 290
- decomposeByNames
 - AtomProbe::RangeFile, 290
- deleteChildren
 - AtomProbe::K3DNodeApprox, 186
- deltaPulse
 - AtomProbe::EPOS_ENTRY, 161
- deprecatedGetEdgeUniqueIndex
 - AtomProbe::Voxels, 354
- depth
 - NodeWalk, 253
- det3by3
 - AtomProbe, 44
- detectFileType
 - AtomProbe::RangeFile, 291
- detectorChannels
 - AtomProbe::THREEDAP_DATA, 335
- detectorRadius
 - AtomProbe::THREEDAP_DATA, 336
- detectorX
 - AtomProbe::ATO_ENTRY, 128
- detectorY
 - AtomProbe::ATO_ENTRY, 128
- Determinant

- AtomProbe, 44
- diff
 - AtomProbe, 45
- digitString
 - AtomProbe, 45
- distanceToFacet
 - AtomProbe, 46
- distanceToSegment
 - AtomProbe, 46
- divideMeshSurface
 - AtomProbe::Mesh, 217
- doFitBackground
 - AtomProbe, 47
- doHull
 - AtomProbe, 48
- dotProd
 - AtomProbe::Point3D, 260
- dotProduct
 - AtomProbe, 48
- dump
 - AtomProbe::K3DNodeApprox, 186
 - AtomProbe::K3DTreeApprox, 194
 - AtomProbe::K3DTreeExact, 200
- dumpAutocorrelation
 - autocorrelate.cpp, 413
- dumpHistogram
 - correlationhist.cpp, 390
 - pulse-filter.cpp, 404
- dumpHistogramToFile
 - zechBackground.cpp, 411
- ELEM_FOUR_NODE_TETRAHEDRON
 - AtomProbe, 111
- ELEM_SINGLE_NODE_POINT
 - AtomProbe, 111
- ELEM_THREE_NODE_TRIANGLE
 - AtomProbe, 111
- ELEM_TWO_NODE_LINE
 - AtomProbe, 112
- ENDIAN_TEST
 - endianTest.h, 433
- EPOS_RECORD_SIZE
 - AtomProbe, 112
- EQ_TOLV
 - overlaps.cpp, 496
- edgeldx
 - AtomProbe, 49
- edgeRemap
 - AtomProbe, 111
- edgesMismatch
 - AtomProbe::TRIANGLE, 339
- elementCount
 - AtomProbe::Mesh, 218
- elementList
 - AtomProbe, 112
- elementName
 - AtomProbe::AbundanceData, 119
- elementNames
 - AtomProbe::AbundanceData, 119
- empty
 - FixedStack, 168
- endMass
 - AtomProbe::FLATTENED_RANGE, 170
- endianTest.h
 - ENDIAN_TEST, 433
 - floatSwapBytes, 432
 - int4SwapBytes, 433
 - is_bigendian, 433
 - is_littleendian, 433
- eraselon
 - AtomProbe::RangeFile, 292
- erasePhysGroup
 - AtomProbe::Mesh, 218
- eraseRange
 - AtomProbe::RangeFile, 292
- estimateRank
 - AtomProbe, 49
- etaToTheta
 - AtomProbe::ModifiedFocusSphericProjection, 235
- eventData
 - AtomProbe::THREEDAP_EXPERIMENT, 337
- eventPulseNumber
 - AtomProbe::THREEDAP_EXPERIMENT, 337
- example/correlationhist.cpp, 390
- example/countepos.cpp, 391
- example/kd-example.cpp, 392
- example/massFit.cpp, 395
- example/massHist.cpp, 396
- example/multiplesplit.cpp, 397
- example/plot-overlaps.cpp, 398
- example/polefigure.cpp, 399
- example/proxigram.cpp, 401
- example/pulse-filter.cpp, 403
- example/signedDistance.cpp, 406
- example/specsim.cpp, 406
- example/zechBackground.cpp, 409
- expand
 - AtomProbe::BoundCube, 141, 142
- extend
 - AtomProbe::Point3D, 260
- extensionIsRange
 - AtomProbe::RangeFile, 293
- extractIons
 - AtomProbe::RangeFile, 293
- extractPositions
 - AtomProbe::CrystalGen, 158
- extras/autocorrelate/autocorrelate.cpp, 412
- extras/readpos_c/readpos.h, 416
- FRAGMENT
 - specsim.cpp, 407
- FaceCentredCubicGen
 - AtomProbe::FaceCentredCubicGen, 165
- farPoint
 - AtomProbe::CrystalGen, 159
- fill
 - AtomProbe::Voxels, 355
- filterBySolutionPPM

- AtomProbe, 50
- filterEposByPulse
 - pulse-filter.cpp, 405
- filterPeakNeedBiggerObs
 - AtomProbe, 50
- findKNearest
 - AtomProbe::K3DTreeApprox, 195
- findMaxLessThanOrEq
 - AtomProbe, 51
- findNearVerticies
 - AtomProbe, 51, 52
- findNearest
 - AtomProbe::K3DTreeApprox, 195
- findNearestUntagged
 - AtomProbe::K3DTreeExact, 200
- findOverlaps
 - AtomProbe, 52, 53
- findOverlapsFromSpectra
 - AtomProbe, 54
- findPeaks
 - AtomProbe, 54
- findUntaggedInRadius
 - AtomProbe::K3DTreeExact, 201
- finish
 - AtomProbe::ProgressBar, 279
- fixedRecordChunkReader
 - AtomProbe, 55
- fixedRecordReader
 - AtomProbe, 56
- FixedStack
 - ~FixedStack, 167
 - empty, 168
 - FixedStack, 167
 - pop, 168
 - push, 168
 - top, 168
- FixedStack< T >, 166
- flattenToMassAxis
 - AtomProbe::MultiRange, 244
- flightPath
 - AtomProbe::THREEDAP_DATA, 336
- floatSwapBytes
 - endianTest.h, 432
- focusDist
 - AtomProbe::SphericProjectionParams, 330
- fourDeterminant
 - AtomProbe, 57
- fpeek
 - AtomProbe, 57
- freeConvexHull
 - AtomProbe, 57
- fromHex
 - AtomProbe::RGBf, 321
- fx
 - AtomProbe::Point3f, 277
- fy
 - AtomProbe::Point3f, 277
- fz
 - AtomProbe::Point3f, 277
- gauss_ratio_pdf
 - AtomProbe, 58
- gcd
 - AtomProbe, 58
- genColString
 - AtomProbe, 58
- generate1DAxialDistHist
 - AtomProbe, 59
- generate1DAxialDistHistSweep
 - AtomProbe, 60
- generateGroupedIsotopeDist
 - AtomProbe::AbundanceData, 120
- generateIsotopeDist
 - AtomProbe::AbundanceData, 121
- generateLattice
 - AtomProbe::BodyCentredCubicGen, 136
 - AtomProbe::CrystalGen, 158
 - AtomProbe::FaceCentredCubicGen, 166
 - AtomProbe::SimpleCubicGen, 326
- generateSingleAtomDist
 - AtomProbe::AbundanceData, 121
- getAllExts
 - AtomProbe::RangeFile, 293
- getAtoErrString
 - AtomProbe, 61
- getAtomicNumber
 - AtomProbe::AbundanceData, 122
- getAxis
 - AtomProbe::K3DNodeApprox, 187
- getAxisBounds
 - AtomProbe::Voxels, 355
- getAxisVal
 - AtomProbe::K3DNodeApprox, 187
- getBinVolume
 - AtomProbe::Voxels, 355
- getBound
 - AtomProbe::BoundCube, 142
- getBoundCube
 - AtomProbe::IonHit, 175
 - AtomProbe::K3DTreeExact, 202
- getBounds
 - AtomProbe::BoundCube, 142
 - AtomProbe::Mesh, 218
 - AtomProbe::Voxels, 355, 356
- getCellUniqueEdgeIndex
 - AtomProbe::Voxels, 356
- getCentroid
 - AtomProbe::BoundCube, 143
 - AtomProbe::IonHit, 175
 - AtomProbe::Point3D, 261
 - AtomProbe::TriangleWithVertexNorm, 343
- getColour
 - AtomProbe::MultiRange, 245
 - AtomProbe::RangeFile, 294
- getContainedNodes
 - AtomProbe::Mesh, 219
- getCurPhysGroups

- AtomProbe::Mesh, 219
- getData
 - AtomProbe::Voxels, 357
- getEdgeCell
 - AtomProbe::Voxels, 358
- getEdgeEndApproxVals
 - AtomProbe::Voxels, 358
- getEdgeEnds
 - AtomProbe::Voxels, 359
- getErrState
 - AtomProbe::RangeFile, 294
- getErrString
 - AtomProbe::MultiRange, 245
 - AtomProbe::RangeFile, 295
- getErrorText
 - AtomProbe::AbundanceData, 122
- getFOVRadius
 - AtomProbe::ModifiedFocusSphericProjection, 236
- getFileSize
 - AtomProbe, 61
- getFitErrorMsg
 - AtomProbe, 62
- getHardAssert
 - AtomProbe, 62
- getISOSpherical
 - AtomProbe::Point3D, 262
- getIndex
 - AtomProbe::Voxels, 360
- getIndexWithUpper
 - AtomProbe::Voxels, 361
- getInterpSlice
 - AtomProbe::Voxels, 362
- getInterpolatedData
 - AtomProbe::Voxels, 361
- getIntersectingPrimitives
 - AtomProbe::Mesh, 220
- getIonDataLimits
 - AtomProbe::IonHit, 176
- getIonHit
 - AtomProbe::EPOS_ENTRY, 160, 161
- getIonID
 - AtomProbe::MultiRange, 245
 - AtomProbe::RangeFile, 295–297
- getIonName
 - AtomProbe::MultiRange, 245
- getLargestDim
 - AtomProbe::BoundCube, 143
- getLoc
 - AtomProbe::K3DNodeApprox, 187
- getLocRef
 - AtomProbe::K3DNodeApprox, 188
- getLocVal
 - AtomProbe::K3DNodeApprox, 188
- getMajor
 - AtomProbe::LibVersion, 208
- getMajorIsotopeFromElemIdx
 - AtomProbe::AbundanceData, 122
- getMassDistributions
 - specsimpl.cpp, 407
- getMassToCharge
 - AtomProbe::IonHit, 176
- getMaxBounds
 - AtomProbe::Voxels, 362
- getMaxDistanceToBox
 - AtomProbe::BoundCube, 144
- getMaxFOV
 - AtomProbe::ModifiedFocusSphericProjection, 236
- getMinBounds
 - AtomProbe::Voxels, 363
- getMinor
 - AtomProbe::LibVersion, 208
- getMolecule
 - AtomProbe::MultiRange, 246
- getName
 - AtomProbe::RangeFile, 298, 299
- getNearestCharge
 - AtomProbe::AbundanceData, 122
- getNearestTri
 - AtomProbe::Mesh, 220
- getNominalMass
 - AtomProbe::AbundanceData, 123
- getNumIons
 - AtomProbe::MultiRange, 246
 - AtomProbe::RangeFile, 299
- getNumRanges
 - AtomProbe::MultiRange, 246
 - AtomProbe::RangeFile, 299, 300
- getOffset
 - AtomProbe::Voxels, 363
- getOrigIndex
 - AtomProbe::K3DTreeExact, 202
- getPitch
 - AtomProbe::Voxels, 363
- getPoint
 - AtomProbe::Voxels, 363
- getPointData
 - AtomProbe::Voxels, 364
- getPoints
 - AtomProbe::IonHit, 177
- getPos
 - AtomProbe::IonHit, 177
- getPosFileErrString
 - AtomProbe, 62
- getPosRef
 - AtomProbe::IonHit, 178
- getPt
 - AtomProbe::K3DTreeExact, 202
- getPtRef
 - AtomProbe::K3DTreeExact, 203
- getRange
 - AtomProbe::MultiRange, 247
 - AtomProbe::RangeFile, 300
- getRangeByRef
 - AtomProbe::RangeFile, 300
- getRangeID
 - AtomProbe::RangeFile, 301

- getRangeIdsFromIon
 - AtomProbe::MultiRange, 247
- getRangeMolecule
 - AtomProbe, 62
- getRangeVolume
 - AtomProbe::RangeFile, 301
- getRanges
 - AtomProbe::MultiRange, 247
- getRecordReadErrString
 - AtomProbe, 63
- GetReducedHullPts
 - AtomProbe, 63
- getRevision
 - AtomProbe::LibVersion, 209
- getRng
 - AtomProbe::RandNumGen, 284
- getSize
 - AtomProbe::BoundCube, 144
 - AtomProbe::Voxels, 364
- getSlice
 - AtomProbe::Voxels, 364
- getSum
 - AtomProbe::Voxels, 365
- getSymbolIndices
 - AtomProbe::AbundanceData, 123, 124
- getTag
 - AtomProbe::K3DTreeExact, 203
- getTriNormal
 - AtomProbe::Mesh, 220
- getValue
 - AtomProbe::Point3D, 262
- getValueArr
 - AtomProbe::Point3D, 263
- getVersionStr
 - AtomProbe::LibVersion, 209
- getVolume
 - AtomProbe::BoundCube, 145
 - AtomProbe::Mesh, 221
- green
 - AtomProbe::RGBf, 322
- gsl_determinant
 - AtomProbe, 64
- gsl_matrix_mult
 - AtomProbe, 64
 - math.cpp, 463
 - math.h, 465
 - maths.cpp, 483
- gsl_print_matrix
 - AtomProbe, 64
 - polefigure.cpp, 399
- gsl_print_vector
 - AtomProbe, 65
- gsl_pseudo_inverse
 - math.cpp, 463
 - math.h, 465
 - maths.cpp, 483
- guessChargeState
 - AtomProbe::RangeFile, 301
- h
 - AtomProbe::MILLER_TRIPLET, 232
- HEX
 - CMakeCXXCompilerId.cpp, 386
- HULL_GRAB_SIZE
 - AtomProbe, 113
- hardAssert
 - AtomProbe, 112
- hasNaN
 - AtomProbe::IonHit, 178
 - AtomProbe::Point3D, 263
- haveRangeVolumes
 - AtomProbe::RangeFile, 302
- helpFuncs.h
 - ARRAYSIZE, 481
 - XOR, 481
- hexStrToUChar
 - AtomProbe, 65
- hitMultiplicity
 - AtomProbe::EPOS_ENTRY, 162
- IONHIT
 - AtomProbe, 21
- include/atomprobe/algorithm/K3DTree-approx.h, 422
- include/atomprobe/algorithm/K3DTree-exact.h, 423
- include/atomprobe/algorithm/axialdf.h, 417
- include/atomprobe/algorithm/componentAnalysis.h, 418
- include/atomprobe/algorithm/convexHull.h, 419
- include/atomprobe/algorithm/filter.h, 420
- include/atomprobe/algorithm/histogram.h, 421
- include/atomprobe/algorithm/isoSurface.h, 422
- include/atomprobe/algorithm/massTool.h, 424
- include/atomprobe/algorithm/rangeCheck.h, 425
- include/atomprobe/algorithm/rotations.h, 426
- include/atomprobe/atomprobe.h, 426
- include/atomprobe/deconvolution/deconvolution.h, 428
- include/atomprobe/helper/XMLHelper.h, 444
- include/atomprobe/helper/aptAssert.h, 428
- include/atomprobe/helper/composition.h, 430
- include/atomprobe/helper/constants.h, 431
- include/atomprobe/helper/convert.h, 431
- include/atomprobe/helper/endianTest.h, 432
- include/atomprobe/helper/mathsf/fsr.h, 434
- include/atomprobe/helper/mathsf/misc.h, 434
- include/atomprobe/helper/mathsf/quat.h, 437
- include/atomprobe/helper/misc.h, 436
- include/atomprobe/helper/progress.h, 438
- include/atomprobe/helper/sampling.h, 438
- include/atomprobe/helper/stringFuncs.h, 439
- include/atomprobe/helper/version.h, 441
- include/atomprobe/helper/voxels.h, 442
- include/atomprobe/io/dataFiles.h, 445
- include/atomprobe/io/multiRange.h, 448
- include/atomprobe/io/ranges.h, 449
- include/atomprobe/isotopes/abundance.h, 451
- include/atomprobe/isotopes/massconvert.h, 451
- include/atomprobe/isotopes/overlaps.h, 453
- include/atomprobe/latticeplanes/generate.h, 454
- include/atomprobe/latticeplanes/millerIndex.h, 455

- include/atomprobe/primitives/boundcube.h, 455
- include/atomprobe/primitives/ionHit.h, 456
- include/atomprobe/primitives/mesh.h, 457
- include/atomprobe/primitives/point3D.h, 458
- include/atomprobe/projection/projection.h, 459
- include/atomprobe/reconstruction/reconstruction-simple.h, 459
- include/atomprobe/spectrum/processing.h, 460
- include/atomprobe/statistics/confidence.h, 461
- incrementCorrelationHist
 - AtomProbe, 65
- incrementDataDistanceWeight
 - AtomProbe, 65
- index
 - NodeWalk, 253
- info_arch
 - CMakeCXXCompilerId.cpp, 387
- info_compiler
 - CMakeCXXCompilerId.cpp, 387
- info_language_dialect_default
 - CMakeCXXCompilerId.cpp, 388
- info_platform
 - CMakeCXXCompilerId.cpp, 388
- info_size
 - SIZEOF_SIZE_T.cpp, 389
- init
 - AtomProbe::ProgressBar, 280
 - AtomProbe::Voxels, 365, 366
- insideBox
 - AtomProbe::Point3D, 264, 265
- int4SwapBytes
 - endianTest.h, 433
- intensity
 - AtomProbe::BACKGROUND_PARAMS, 133
- intensityTolerance
 - AtomProbe::OVERLAP_PROBLEM_SETTINGS, 254
- intersect_RayTriangle
 - AtomProbe, 66
- intersects
 - AtomProbe::BoundCube, 145, 146
- IonHit
 - AtomProbe::IonHit, 174
- is_bigendian
 - endianTest.h, 433
- is_littleendian
 - endianTest.h, 433
- isDegenerate
 - AtomProbe::TriangleWithVertexNorm, 343
- isFlat
 - AtomProbe::BoundCube, 146
- isNotDirectory
 - AtomProbe, 67
- isOK
 - AtomProbe::RANGE_MOLECULE, 285
- isOrientedCoherently
 - AtomProbe::Mesh, 222
- isRanged
 - AtomProbe::MultiRange, 247, 248
 - AtomProbe::RangeFile, 303
- isSane
 - AtomProbe::Mesh, 222
 - AtomProbe::TRIANGLE, 339
- isSelfConsistent
 - AtomProbe::MultiRange, 248
 - AtomProbe::RangeFile, 304
- isotope
 - AtomProbe::AbundanceData, 124
- isotopeIndex
 - AtomProbe::AbundanceData, 124, 125
- isotopes
 - AtomProbe::AbundanceData, 125
- k
 - AtomProbe::MILLER_TRIPLET, 232
- K3DTree-exact.cpp
 - PROGRESS_REDUCE, 473
- K3DTreeApprox
 - AtomProbe::BoundCube, 153
 - AtomProbe::K3DTreeApprox, 192
- K3DTreeExact
 - AtomProbe::BoundCube, 153
 - AtomProbe::K3DTreeExact, 198
- KDTest.cpp
 - callback, 505
 - kdExactFuzz, 505
 - main, 505
 - NUM_IONS, 507
 - progress, 507
 - SCALE, 507
 - SEARCH_RAD, 507
 - testInexactKDTree, 506
- kahansum
 - AtomProbe, 67
- kd-example.cpp
 - callback, 393
 - M_PI, 393
 - main, 394
 - progress, 394
 - TIME_END, 393
 - TIME_START, 393
- kdExactFuzz
 - KDTest.cpp, 505
- kill
 - AtomProbe::K3DTreeApprox, 196
- killOrphanNodes
 - AtomProbe::Mesh, 222
- l
 - AtomProbe::MILLER_TRIPLET, 232
- LIBATOMPROBE_MAJOR
 - version.h, 442
- LIBATOMPROBE_MINOR
 - version.h, 442
- LIBATOMPROBE_REVISION
 - version.h, 442
- lambdaBack

- AtomProbe::ZECH_ROOT, 383
- leastSquaresDeconvolve
 - AtomProbe, 68
- left
 - AtomProbe::K3DNodeApprox, 188
- LibVersion
 - AtomProbe::LibVersion, 207
- libVersion
 - AtomProbe, 113
- linearHistogram
 - AtomProbe, 68
- lines
 - AtomProbe::Mesh, 230
- linkIdentifiers
 - AtomProbe, 69
- loadATOFile
 - AtomProbe, 69
- loadEposFile
 - AtomProbe, 70
- loadFile
 - AtomProbe::Voxels, 366
- loadGmshMesh
 - AtomProbe::Mesh, 223
- loadPosFile
 - AtomProbe, 70, 71
- loadTapsimBinFile
 - AtomProbe, 72
- loadTextData
 - AtomProbe, 73
- localData
 - AtomProbe::CrystalGen, 159
- lowercase
 - AtomProbe, 73
- M_PI
 - kd-example.cpp, 393
- MASS_TOL
 - specsim.cpp, 408
- MAX_LINE_SIZE
 - AtomProbe, 113
- MAX_RANGEFILE_SIZE
 - AtomProbe, 113
- MESH_LOAD_ERRS
 - AtomProbe, 114
- MILLER_TRIPLET
 - AtomProbe::MILLER_TRIPLET, 232
- MULTIRANGE_FORMAT_VERSION
 - AtomProbe, 114
- mag
 - AtomProbe::Point3D, 265
- main
 - autocorrelate.cpp, 414
 - CMakeCXXCompilerId.cpp, 387
 - correlationhist.cpp, 390
 - countepos.cpp, 391
 - KDTest.cpp, 505
 - kd-example.cpp, 394
 - massFit.cpp, 395
 - massHist.cpp, 396
 - multiplesplit.cpp, 397
 - plot-overlaps.cpp, 398
 - polefigure.cpp, 400
 - proxigram.cpp, 402
 - pulse-filter.cpp, 405
 - rangetest.cpp, 508
 - SIZEOF_SIZE_T.cpp, 389
 - signedDistance.cpp, 406
 - specsim.cpp, 407
 - unittests.cpp, 511
 - zechBackground.cpp, 411
- makeHistogram
 - AtomProbe, 73
- makeIntersection
 - AtomProbe::BoundCube, 147
- makeSelfConsistent
 - AtomProbe::RangeFile, 305
- makeUnion
 - AtomProbe::BoundCube, 147
- marchingCubes
 - AtomProbe, 74
- mass
 - AtomProbe::ATO_ENTRY, 128
 - AtomProbe::ISOTOPE_ENTRY, 184
 - AtomProbe::Weight, 380
- massEnd
 - AtomProbe::BACKGROUND_PARAMS, 133
- massError
 - AtomProbe::ISOTOPE_ENTRY, 184
- massFit.cpp
 - main, 395
 - printSolution, 395
- massHist.cpp
 - main, 396
- massNumber
 - AtomProbe::ISOTOPE_ENTRY, 185
- massStart
 - AtomProbe::BACKGROUND_PARAMS, 133
- massToCharge
 - AtomProbe::EPOS_ENTRY, 162
 - AtomProbe::IONHIT, 182
- massTolerance
 - AtomProbe::OVERLAP_PROBLEM_SETTINGS, 254
- matchComposedName
 - AtomProbe, 75
- math.cpp, 462
 - ASSERT, 463
 - gsl_matrix_mult, 463
 - gsl_pseudo_inverse, 463
 - psuedoInverseFromSVD, 463
- math.h, 464
 - computeMeanAndVariance, 465
 - gsl_matrix_mult, 465
 - gsl_pseudo_inverse, 465
- maths.cpp
 - ASSERT, 482
 - gsl_matrix_mult, 483

- gsl_pseudo_inverse, 483
- psuedoInverseFromSVD, 483
- max
 - AtomProbe::BoundCube, 148
 - AtomProbe::Voxels, 367
- maxDefaultCharge
 - AtomProbe::OVERLAP_PROBLEM_SETTINGS, 255
- maxExplainedFraction
 - AtomProbe, 75, 76
- maximumLinearTable
 - AtomProbe, 113
- meanAndStdev
 - AtomProbe, 77
- mergeDuplicateVertices
 - AtomProbe::Mesh, 224
- min
 - AtomProbe::BoundCube, 148
 - AtomProbe::Voxels, 367
- minMax
 - AtomProbe::Voxels, 368
- mirrorOut
 - AtomProbe::CrystalGen, 158
- mode
 - AtomProbe::BACKGROUND_PARAMS, 134
- ModifiedFocusSphericProjection
 - AtomProbe::ModifiedFocusSphericProjection, 235
- moveBothRanges
 - AtomProbe::RangeFile, 306
- moveRange
 - AtomProbe::RangeFile, 306
- MultiRange
 - AtomProbe::MultiRange, 240, 241
- multiplesplit.cpp
 - main, 397
- myStrDup
 - AtomProbe, 77
- NUM_ELEMENTS
 - AtomProbe, 114
- NUM_IONS
 - KDTest.cpp, 507
- name
 - AtomProbe::RANGE_MOLECULE, 286
- negate
 - AtomProbe::Point3D, 266
- nextHitGroupOffset
 - AtomProbe::VOLTAGE_DATA, 345
- nodeCount
 - AtomProbe::K3DTreeApprox, 197
- NodeWalk, 252
 - cube, 253
 - depth, 253
 - index, 253
 - NodeWalk, 253
- nodes
 - AtomProbe::Mesh, 230
- normal
 - AtomProbe::Point3D, 266
- AtomProbe::TriangleWithVertexNorm, 344
- normalise
 - AtomProbe::Point3D, 267
- normaliseVec
 - specsim.cpp, 408
- nullBackground
 - AtomProbe, 78
- nullifyMarker
 - AtomProbe, 78
- numDupTris
 - AtomProbe::Mesh, 224
- numDupVertices
 - AtomProbe::Mesh, 224
- numElements
 - AtomProbe::AbundanceData, 125
- numIsotopes
 - AtomProbe::AbundanceData, 126
- numericalEstimateGaussRatioConf
 - AtomProbe, 78
- numericalEstimatePoissRatioConf
 - AtomProbe, 79
- numericalEstimateSkellamConf
 - AtomProbe, 79
- OPS_ENUM_ERRSTRINGS
 - AtomProbe, 114
- observation
 - AtomProbe::ZECH_ROOT, 383
- offset
 - WEIGHT_SEARCH_ENTRY, 382
- onlyDir
 - AtomProbe, 80
- onlyFilename
 - AtomProbe, 80
- open
 - AtomProbe::AbundanceData, 126
 - AtomProbe::MultiRange, 249
 - AtomProbe::RangeFile, 306
- openFormat
 - AtomProbe::RangeFile, 307
- operator<
 - AtomProbe, 80
 - AtomProbe::Weight, 380
- operator<<
 - AtomProbe, 81
 - AtomProbe::BoundCube, 153
 - AtomProbe::IonHit, 182
 - AtomProbe::Point3D, 276
- operator*
 - AtomProbe::Point3D, 268
- operator*=
 - AtomProbe::Point3D, 268
- operator()
 - AtomProbe::AxisCompare, 130
 - AtomProbe::ComparePairFirst, 154
 - AtomProbe::ComparePairFirstReverse, 154
 - AtomProbe::ComparePairSecond, 156
 - AxisCompareExact, 131
- operator+

- AtomProbe::IonHit, 179
- AtomProbe::Point3D, 269
- operator+=
 - AtomProbe::Point3D, 269
- operator-
 - AtomProbe::Point3D, 269, 270
- operator-=
 - AtomProbe::Point3D, 270
- operator/
 - AtomProbe::Point3D, 270
- operator/=
 - AtomProbe::Voxels, 368
- operator=
 - AtomProbe::BoundCube, 149
 - AtomProbe::IonHit, 179
 - AtomProbe::Point3D, 271
 - AtomProbe::RangeFile, 308
- operator==
 - AtomProbe::BoundCube, 149
 - AtomProbe::EPOS_ENTRY, 161
 - AtomProbe::IonHit, 179
 - AtomProbe::MILLER_TRIPLET, 232
 - AtomProbe::Point3D, 271
 - AtomProbe::SIMPLE_SPECIES, 323
 - AtomProbe::Voxels, 369
 - AtomProbe::Weight, 380
- operator[]
 - AtomProbe::IonHit, 179
 - AtomProbe::Point3D, 271
- orthogonalise
 - AtomProbe::Point3D, 272
- overlaps.cpp
 - EQ_TOLV, 496
- p
 - AtomProbe::LINE, 210
 - AtomProbe::TETRAHEDRON, 334
 - AtomProbe::TRIANGLE, 340
 - AtomProbe::TriangleWithIndexedVertices, 340
 - AtomProbe::TriangleWithVertexNorm, 344
- PLATFORM_ID
 - CMakeCXXCompilerId.cpp, 386
- PROGRESS_BAR_SIZE
 - autocorrelate.cpp, 415
- PROGRESS_REDUCE
 - AtomProbe, 115
 - K3DTree-exact.cpp, 473
- pairContains
 - AtomProbe, 82
- pairOverlaps
 - AtomProbe, 82
- params
 - AtomProbe::THREEDAP_EXPERIMENT, 338
- parse
 - AtomProbe::Point3D, 272
- parseColString
 - AtomProbe, 82
- parseCompositionData
 - AtomProbe, 83
- physGroup
 - AtomProbe::LINE, 210
 - AtomProbe::TETRAHEDRON, 334
 - AtomProbe::TRIANGLE, 340
- physGroupNames
 - AtomProbe::Mesh, 230
- plot-overlaps.cpp
 - main, 398
- Point3D
 - AtomProbe::Point3D, 257, 258
- PointDir
 - AtomProbe, 31
- points
 - AtomProbe::Mesh, 230
- pointsInside
 - AtomProbe::Mesh, 225
- poissonConfidenceObservation
 - AtomProbe, 84
- polefigure.cpp
 - gsl_print_matrix, 399
 - main, 400
- pop
 - FixedStack, 168
- popLocale
 - AtomProbe, 84
- pos
 - AtomProbe::IONHIT, 182
- print
 - AtomProbe::Mesh, 225
- printSolution
 - massFit.cpp, 395
- processing.cpp
 - USE_CENTRAL, 503
- progress
 - KDTest.cpp, 507
 - kd-example.cpp, 394
- ProgressBar
 - AtomProbe::ProgressBar, 278
- proxigram.cpp
 - main, 402
- psuedoInverseFromSVD
 - math.cpp, 463
 - maths.cpp, 483
- ptsInSphere
 - AtomProbe::K3DTreeExact, 203
- pulse-filter.cpp
 - convertToPos, 404
 - dumpHistogram, 404
 - filterEposByPulse, 405
 - main, 405
- pulseVolt
 - AtomProbe::VOLTAGE_DATA, 346
- pulseVoltage
 - AtomProbe::ATO_ENTRY, 129
- push
 - FixedStack, 168
- pushLocale
 - AtomProbe, 85

- pyramidVol
 - AtomProbe, 85
- qhullInitd
 - AtomProbe, 115
- quat_get_rot_quat
 - AtomProbe, 85
- quat_invert
 - AtomProbe, 86
- quat_mult_no_second_a
 - AtomProbe, 86
- quat_pointmult
 - AtomProbe, 86
- quat_rot
 - AtomProbe, 87
- quat_rot_apply_quat
 - AtomProbe, 88
- quat_rot_array
 - AtomProbe, 88, 89
- RANGE_EXTS
 - AtomProbe, 115
- RECORDREAD_ERR_STRINGS
 - AtomProbe, 116
- randGen
 - AtomProbe, 115
- RandNumGen
 - AtomProbe::RandNumGen, 283
- randomIndices
 - AtomProbe, 90
- randomSelect
 - AtomProbe, 90
- range
 - AtomProbe::MultiRange, 250
 - AtomProbe::RangeFile, 308
- rangeByID
 - AtomProbe::RangeFile, 309
- rangeByRangeID
 - AtomProbe::RangeFile, 309
- RangeFile
 - AtomProbe::RangeFile, 289
- rangeInvertable
 - AtomProbe::RangeFile, 309
- rangeOverlaps
 - AtomProbe, 90
- rangePaths
 - unittests.cpp, 511
- rangeToStr
 - AtomProbe, 91
- rangeTypeString
 - AtomProbe::RangeFile, 310
- rangetest.cpp
 - main, 508
- readEposRecord
 - AtomProbe, 91
- readPos
 - readpos.h, 416
- readPosapOps
 - AtomProbe, 92
- readpos.h
 - readPos, 416
- reassignGroups
 - AtomProbe::Mesh, 226
- rebin
 - AtomProbe::Voxels, 369
- reciprocal
 - AtomProbe::Point3D, 273
- reconstruct
 - AtomProbe::ReconstructionSphereOnCone, 317
- reconstructTest
 - AtomProbe, 93
- ReconstructionSphereOnCone
 - AtomProbe::ReconstructionSphereOnCone, 316
- red
 - AtomProbe::RGBf, 322
- removeDuplicateTris
 - AtomProbe::Mesh, 226
- removeElements
 - AtomProbe, 93
- removeStrayTris
 - AtomProbe::Mesh, 226
- reset
 - AtomProbe::ProgressBar, 280
- resetPts
 - AtomProbe::K3DTreeExact, 204
- resize
 - AtomProbe::Voxels, 369
- resizeKeepData
 - AtomProbe::Voxels, 370
- resurface
 - AtomProbe::Mesh, 227
- right
 - AtomProbe::K3DNodeApprox, 189
- rootIdx
 - AtomProbe::K3DTreeExact, 205
- rotate
 - AtomProbe::Mesh, 227
- rotateMatch
 - AtomProbe, 93
- runTests
 - unittests.cpp, 511
- SCALE
 - KDTest.cpp, 507
- SEARCH_RAD
 - KDTest.cpp, 507
- SIZEOF_SIZE_T.cpp
 - info_size, 389
 - main, 389
 - SIZE, 389
- SIZE
 - SIZEOF_SIZE_T.cpp, 389
- STRINGIFY_HELPER
 - CMakeCXXCompilerId.cpp, 387
- STRINGIFY
 - CMakeCXXCompilerId.cpp, 387
- safeComputeACWNORMAL
 - AtomProbe::TriangleWithVertexNorm, 343

- sampleIons
 - AtomProbe, 93
- saveGmshMesh
 - AtomProbe::Mesh, 227
- savePosFile
 - AtomProbe, 94
- saveTapsimBin
 - AtomProbe, 95
- scale
 - AtomProbe::Mesh, 228
- scaleDown
 - AtomProbe::GnomonicProjection, 171
 - AtomProbe::ModifiedFocusSphericProjection, 236
 - AtomProbe::SphericPlaneProjection, 328
 - AtomProbe::StereographicProjection, 332
- scaleUp
 - AtomProbe::GnomonicProjection, 171
 - AtomProbe::ModifiedFocusSphericProjection, 237
 - AtomProbe::SphericPlaneProjection, 328
 - AtomProbe::StereographicProjection, 332
- segmentTriple
 - AtomProbe::BoundCube, 149
- selectElements
 - AtomProbe, 95
- separableConvolve
 - AtomProbe::Voxels, 371
- setAxis
 - AtomProbe::AxisCompare, 131
 - AtomProbe::K3DNodeApprox, 189
 - AxisCompareExact, 132
- setBound
 - AtomProbe::BoundCube, 149
- setBounds
 - AtomProbe::BoundCube, 150–152
 - AtomProbe::Voxels, 371
- setCallback
 - AtomProbe::K3DTreeExact, 205
- setColour
 - AtomProbe::MultiRange, 250
 - AtomProbe::RangeFile, 311
- setData
 - AtomProbe::Voxels, 372
- setDetectorEfficiency
 - AtomProbe::ReconstructionSphereOnCone, 318
- setEnforceConsistent
 - AtomProbe::RangeFile, 311
- setEntry
 - AtomProbe::Voxels, 373
- setFlightPath
 - AtomProbe::ReconstructionSphereOnCone, 318
- setFocus
 - AtomProbe::ModifiedFocusSphericProjection, 237
- setHardAssert
 - AtomProbe, 96
- setHit
 - AtomProbe::IonHit, 180
- setISOSpherical
 - AtomProbe::Point3D, 274
- setInitialRadius
 - AtomProbe::ReconstructionSphereOnCone, 319
- setInverseLimits
 - AtomProbe::BoundCube, 152
- setIonID
 - AtomProbe::MultiRange, 250
 - AtomProbe::RangeFile, 312
- setIonLongName
 - AtomProbe::RangeFile, 312
- setIonShortName
 - AtomProbe::RangeFile, 312
- setLeft
 - AtomProbe::K3DNodeApprox, 189
- setLength
 - AtomProbe::ProgressBar, 280
- setLimits
 - AtomProbe::BoundCube, 152
- setLoc
 - AtomProbe::K3DNodeApprox, 189
- setMaskPeriod
 - AtomProbe::LinearFeedbackShiftReg, 211
- setMassToCharge
 - AtomProbe::IonHit, 180
- setPoint
 - AtomProbe::Voxels, 374
- setPos
 - AtomProbe::IonHit, 180, 181
- setProgressPointer
 - AtomProbe::K3DTreeExact, 205
- setProjModel
 - AtomProbe::ReconstructionSphereOnCone, 319
- setRadiusEvolutionMode
 - AtomProbe::ReconstructionSphereOnCone, 319
- setRangeEnd
 - AtomProbe::RangeFile, 313
- setRangeGroups
 - AtomProbe::MultiRange, 250
- setRangeStart
 - AtomProbe::RangeFile, 313
- setRangeVolume
 - AtomProbe::RangeFile, 314
- setReconFOV
 - AtomProbe::ReconstructionSphereOnCone, 319
- setRight
 - AtomProbe::K3DNodeApprox, 190
- setShankAngle
 - AtomProbe::ReconstructionSphereOnCone, 320
- setState
 - AtomProbe::LinearFeedbackShiftReg, 211
- setTriangleMesh
 - AtomProbe::Mesh, 228
- setValue
 - AtomProbe::Point3D, 274
- setValueArr
 - AtomProbe::Point3D, 275
- signVal
 - AtomProbe, 97
- signedDistance.cpp

- main, 406
- signedDistanceToFacet
 - AtomProbe, 96
- SimpleCubicGen
 - AtomProbe::SimpleCubicGen, 325
- simplify
 - AtomProbe::MILLER_TRIPLET, 233
- size
 - AtomProbe::K3DTreeExact, 206
 - AtomProbe::Voxels, 375
- sizeofType
 - AtomProbe::Voxels, 375
- solveLeastSquares
 - AtomProbe, 97
- specsim.cpp
 - FRAGMENT, 407
 - getMassDistributions, 407
 - MASS_TOL, 408
 - main, 407
 - normaliseVec, 408
- SphericProjectionEqn
 - AtomProbe, 97
- splitOverlapping
 - AtomProbe::MultiRange, 251
- splitStrsRef
 - AtomProbe, 98
- sqrDist
 - AtomProbe::K3DNodeApprox, 190
 - AtomProbe::Point3D, 275
- sqrMag
 - AtomProbe::Point3D, 275
- src/algorithm/K3DTree-approx.cpp, 471
- src/algorithm/K3DTree-exact.cpp, 472
- src/algorithm/axialdf.cpp, 466
- src/algorithm/componentAnalysis.cpp, 467
- src/algorithm/convexHull.cpp, 467
- src/algorithm/filter.cpp, 468
- src/algorithm/histogram.cpp, 469
- src/algorithm/isoSurface.cpp, 470
- src/algorithm/massTool.cpp, 473
- src/algorithm/rangeCheck.cpp, 474
- src/algorithm/rotations.cpp, 474
- src/atomprobe.cpp, 475
- src/deconvolution/deconvolution.cpp, 476
- src/helper/XMLHelper.cpp, 489
- src/helper/aptAssert.cpp, 476
- src/helper/composition.cpp, 477
- src/helper/convert.cpp, 478
- src/helper/helpFuncs.cpp, 479
- src/helper/helpFuncs.h, 479
- src/helper/mathsf/fsr.cpp, 481
- src/helper/mathsf/mathsf.cpp, 482
- src/helper/mathsf/misc.cpp, 484
- src/helper/mathsf/quat.cpp, 485
- src/helper/progress.cpp, 486
- src/helper/sampling.cpp, 487
- src/helper/stringFuncs.cpp, 487
- src/helper/voxels.cpp, 488
- src/io/dataFiles.cpp, 490
- src/io/multiRange.cpp, 491
- src/io/ranges.cpp, 493
- src/isotopes/abundance.cpp, 494
- src/isotopes/massconvert.cpp, 494
- src/isotopes/overlaps.cpp, 495
- src/lattice/generate.cpp, 496
- src/lattice/millerIndex.cpp, 497
- src/primitives/boundcube.cpp, 497
- src/primitives/ionhit.cpp, 498
- src/primitives/mesh.cpp, 498
- src/primitives/point3D.cpp, 500
- src/projection/projection.cpp, 500
- src/reconstruction/reconstruction-simple.cpp, 501
- src/spectrum/processing.cpp, 502
- src/statistics/confidence.cpp, 503
- startMass
 - AtomProbe::FLATTENED_RANGE, 170
- stdev
 - AtomProbe::BACKGROUND_PARAMS, 134
- stlStrToStlWStr
 - AtomProbe, 98
- stlWStrToStlStr
 - AtomProbe, 99
- strAppend
 - AtomProbe, 99
- stream_cast
 - AtomProbe, 100
- strhas
 - AtomProbe, 100
- stripChars
 - AtomProbe, 101
- stripWhite
 - AtomProbe, 101
- stripZeroEntries
 - AtomProbe, 101
- sumVoxels
 - AtomProbe, 102
- swap
 - AtomProbe::CrystalGen, 159
 - AtomProbe::RangeFile, 314
 - AtomProbe::Voxels, 375
- symbol
 - AtomProbe::ISOTOPE_ENTRY, 185
- symbolIdxFromAtomicNumber
 - AtomProbe::AbundanceData, 126
- symbolIndex
 - AtomProbe::AbundanceData, 127
- TEST
 - aptAssert.h, 429
 - test.h, 510
- TIME_END
 - kd-example.cpp, 393
- TIME_START
 - kd-example.cpp, 393
- tZero
 - AtomProbe::THREEDAP_DATA, 336
- tabs

- AtomProbe, 102
- tag
 - AtomProbe::K3DTreeExact, 206
- tagCount
 - AtomProbe::K3DTreeExact, 206
- tagged
 - AtomProbe::K3DNodeExact, 191
- test.h
 - TEST, 510
- test/KDTest.cpp, 504
- test/rangetest.cpp, 508
- test/test.h, 509
- test/unittests.cpp, 510
- testInexactKDTree
 - KDTest.cpp, 506
- tetrahedra
 - AtomProbe::Mesh, 230
- theta
 - AtomProbe::SphericProjectionParams, 330
- thetaToEta
 - AtomProbe::ModifiedFocusSphericProjection, 238
 - AtomProbe::StereographicProjection, 332
- threshold
 - AtomProbe::Voxels, 376
- thresholdForPosition
 - AtomProbe::Voxels, 376
- thresholdToBoolMask
 - AtomProbe::Voxels, 377
- timeOfFlight
 - AtomProbe::EPOS_ENTRY, 162
- toAzimuthal
 - AtomProbe::GnomonicProjection, 172
 - AtomProbe::ModifiedFocusSphericProjection, 238
 - AtomProbe::SphericPlaneProjection, 329
 - AtomProbe::StereographicProjection, 333
- toHex
 - AtomProbe::RGBf, 321
- toPlanar
 - AtomProbe::GnomonicProjection, 172
 - AtomProbe::ModifiedFocusSphericProjection, 238
 - AtomProbe::SphericPlaneProjection, 329
 - AtomProbe::StereographicProjection, 333
- tof
 - AtomProbe::ATO_ENTRY, 129
 - AtomProbe::SINGLE_HIT, 327
- tolEqual
 - AtomProbe, 102
- top
 - FixedStack, 168
- transform3x3
 - AtomProbe::Point3D, 276
- translate
 - AtomProbe::Mesh, 229
- transposeVector
 - AtomProbe, 103
- trapezIntegral
 - AtomProbe::Voxels, 378
- treeProgressBar
 - autocorrelate.cpp, 415
- treeProgressValue
 - autocorrelate.cpp, 416
- trilsDegenerate
 - AtomProbe, 103
- triangles
 - AtomProbe::Mesh, 231
- tripletToNormal
 - AtomProbe::MILLER_TRIPLET, 233
- USE_CENTRAL
 - processing.cpp, 503
- ucharToHexStr
 - AtomProbe, 104
- uniqueId
 - AtomProbe::Weight, 380
- unittests.cpp
 - ASSERT, 510
 - main, 511
 - rangePaths, 511
 - runTests, 511
- update
 - AtomProbe::ProgressBar, 281
- uppercase
 - AtomProbe, 104
- VERTEX_OFFSET
 - AtomProbe, 116
- vectorMultiErase
 - AtomProbe, 104
- vectorPointDir
 - AtomProbe, 105
- verifyTable
 - AtomProbe::LinearFeedbackShiftReg, 212
- version.h
 - LIBATOMPROBE_MAJOR, 442
 - LIBATOMPROBE_MINOR, 442
 - LIBATOMPROBE_REVISION, 442
- voltDC
 - AtomProbe::EPOS_ENTRY, 162
- voltPulse
 - AtomProbe::EPOS_ENTRY, 162
- voltage
 - AtomProbe::ATO_ENTRY, 129
 - AtomProbe::VOLTAGE_DATA, 346
- voltageData
 - AtomProbe::THREEDAP_EXPERIMENT, 338
- Voxels
 - AtomProbe::Voxels, 349
- voxels.h
 - XOR, 444
- WARN
 - aptAssert.h, 429
- WEIGHT_SEARCH_ENTRY, 381
 - allowableWeights, 381
 - cumulativeWeight, 381
 - curWeights, 382
 - offset, 382

Weight

- AtomProbe::Weight, [379](#)

weightedMean

- AtomProbe, [105](#)

write

- AtomProbe::MultiRange, [251](#)

- AtomProbe::RangeFile, [314](#), [315](#)

writeFile

- AtomProbe::Voxels, [378](#)

x

- AtomProbe::ATO_ENTRY, [129](#)

- AtomProbe::EPOS_ENTRY, [163](#)

- AtomProbe::SINGLE_HIT, [327](#)

xDetector

- AtomProbe::EPOS_ENTRY, [163](#)

XMLFreeDoc

- AtomProbe, [105](#)

XMLGetNextElemAttrib

- AtomProbe, [106](#)

XMLHelpFwdNotElem

- AtomProbe, [106](#)

XMLHelpFwdToElem

- AtomProbe, [106](#)

XMLHelpFwdToList

- AtomProbe, [107](#)

XMLHelpGetProp

- AtomProbe, [107](#)

XMLHelpGetText

- AtomProbe, [107](#), [108](#)

XMLHelpNextType

- AtomProbe, [108](#)

XOR

- helpFuncs.h, [481](#)

- voxels.h, [444](#)

y

- AtomProbe::ATO_ENTRY, [129](#)

- AtomProbe::EPOS_ENTRY, [163](#)

- AtomProbe::SINGLE_HIT, [327](#)

yDetector

- AtomProbe::EPOS_ENTRY, [163](#)

z

- AtomProbe::ATO_ENTRY, [129](#)

- AtomProbe::EPOS_ENTRY, [163](#)

zechBackground.cpp

- ASSERT, [409](#)

- dumpHistogramToFile, [411](#)

- main, [411](#)

- zechCorrect, [411](#)

zechConfidenceLimits

- AtomProbe, [108](#)

zechCorrect

- zechBackground.cpp, [411](#)

zechRoot

- AtomProbe, [109](#)