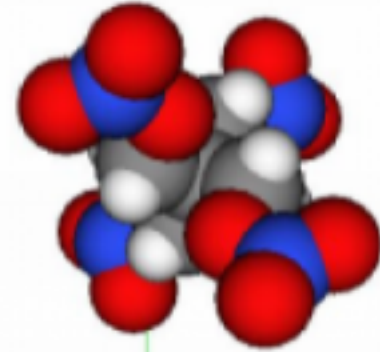


---

# GSAS-2



## GSAS-II Developers Documentation

*Release 0.2.0*

Robert B. Von Dreele and Brian H. Toby

October 14, 2015



<b>1</b>	<b><i>GSAS-II Main Module</i></b>	<b>1</b>
<b>2</b>	<b><i>GSASIIobj: Data objects</i></b>	<b>7</b>
2.1	Constraints Tree Item . . . . .	7
2.2	Covariance Tree Item . . . . .	8
2.3	Phase Tree Items . . . . .	8
2.4	Rigid Body Objects . . . . .	10
2.5	Space Group Objects . . . . .	11
2.6	Atom Records . . . . .	12
2.7	Drawing Atom Records . . . . .	12
2.8	Powder Diffraction Tree Items . . . . .	13
2.9	Powder Reflection Data Structure . . . . .	15
2.10	Single Crystal Tree Items . . . . .	15
2.11	Single Crystal Reflection Data Structure . . . . .	16
2.12	Image Data Structure . . . . .	16
2.13	Parameter Dictionary . . . . .	18
2.14	<i>Classes and routines</i> . . . . .	19
<b>3</b>	<b><i>GSAS-II Utility Modules</i></b>	<b>27</b>
3.1	<i>GSASIIdata: Data for computations</i> . . . . .	27
3.2	<i>GSASIIpath: locations &amp; updates</i> . . . . .	27
3.3	<i>GSASIIlog: Logging of “Actions”</i> . . . . .	30
3.4	<i>config_example.py: Configuration options</i> . . . . .	31
3.5	<i>GSASIIElem: functions for element types</i> . . . . .	32
3.6	<i>GSASIIlattice: Unit cells</i> . . . . .	35
3.7	<i>GSASIIspc: Space group module</i> . . . . .	41
3.8	<i>gltext: draw OpenGL text</i> . . . . .	46
3.9	<i>ElementTable: Periodic Table Data</i> . . . . .	48
3.10	<i>FormFactors: Scattering Data</i> . . . . .	49
3.11	<i>ImageCalibrants: Calibration Standards</i> . . . . .	49
3.12	<i>atmdata: Table of atomic data</i> . . . . .	49
3.13	<i>defaultIparms: Table of instrument parameters</i> . . . . .	50
<b>4</b>	<b><i>GSAS-II GUI Routines</i></b>	<b>51</b>
4.1	<i>GSASIIctrls: Custom GUI controls</i> . . . . .	51
4.2	<i>GSASIIgrid: Basic GUI routines</i> . . . . .	64
4.3	<i>GSASIIIO: Misc I/O routines</i> . . . . .	66
4.4	<i>ReadMarCCDFrame: Read Mar Files</i> . . . . .	73
4.5	<i>GSASIIpy3: Python 3.x Routines</i> . . . . .	74
<b>5</b>	<b><i>GSAS-II GUI Submodules</i></b>	<b>75</b>

5.1	<i>GSASIIphsGUI: Phase GUI</i>	75
5.2	<i>GSASIIddataGUI: Phase Diffraction Data GUI</i>	75
5.3	<i>GSASIIElemGUI: GUI to select and delete element lists</i>	76
5.4	<i>GSASIIconstrGUI: Constraint GUI routines</i>	76
5.5	<i>GSASIIimgGUI: Image GUI</i>	76
5.6	<i>GSASIIpwdGUI: Powder Pattern GUI routines</i>	77
5.7	<i>GSASIIrestrGUI: Restraint GUI routines</i>	78
5.8	<i>GSASIIexprGUI: Expression Handling</i>	78
<b>6</b>	<b><i>GSAS-II Structure Submodules</i></b>	<b>81</b>
6.1	<i>GSASIIstrMain: main structure routine</i>	81
6.2	<i>GSASIIstrMath - structure math routines</i>	82
6.3	<i>GSASIIstrIO: structure I/O routines</i>	85
<b>7</b>	<b><i>GSASIImapvars: Parameter constraints</i></b>	<b>89</b>
7.1	<i>External Routines</i>	90
7.2	<i>Global Variables</i>	91
7.3	<i>Routines</i>	91
<b>8</b>	<b><i>GSASIIimage: Image calc module</i></b>	<b>95</b>
<b>9</b>	<b><i>GSASIImath: computation module</i></b>	<b>97</b>
<b>10</b>	<b><i>GSASIIindex: Cell Indexing Module</i></b>	<b>113</b>
<b>11</b>	<b><i>GSASIIplot: plotting routines</i></b>	<b>115</b>
<b>12</b>	<b><i>GSASII powder calculation module</i></b>	<b>119</b>
<b>13</b>	<b><i>GSAS-II Small Angle Scattering Submodules</i></b>	<b>123</b>
13.1	<i>GSASII small angle calculation module</i>	123
13.2	<i>Substances: Define Materials</i>	126
<b>14</b>	<b><i>GSAS-II Scripts</i></b>	<b>129</b>
14.1	<i>testDeriv: Check derivative computation</i>	129
14.2	<i>GSASIItestplot: Plotting for testDeriv</i>	129
14.3	<i>scanCCD: reduce data from scanning CCD</i>	129
14.4	<i>makeMacApp: Create Mac Applet</i>	129
14.5	<i>unit_tests: Self-test Module</i>	130
<b>15</b>	<b><i>GSAS-II Export Modules</i></b>	<b>131</b>
15.1	<i>Module G2export_examples: Examples</i>	131
15.2	<i>Module G2export_csv: Spreadsheet export</i>	132
15.3	<i>Module G2export_PDB: Macromolecular export</i>	133
15.4	<i>Module G2export_image: 2D Image data export</i>	133
15.5	<i>Module G2export_map: Map export</i>	133
15.6	<i>Module G2export_shelx: Examples</i>	134
15.7	<i>Module G2export_CIF: CIF Exports</i>	134
15.8	<i>Module G2export_pwdr: Export powder input files</i>	137
15.9	<i>Module G2export_FIT2D: Fit2D “Chi” export</i>	137
<b>16</b>	<b><i>GSAS-II Import Modules</i></b>	<b>139</b>
16.1	<i>Writing an Import Routine</i>	139
16.2	<i>Phase Import Routines</i>	141
16.3	<i>Powder Data Import Routines</i>	142
16.4	<i>Single Crystal Data Import Routines</i>	144

16.5	Small Angle Scattering Data Import Routines . . . . .	145
16.6	Image Import Routines . . . . .	146
<b>17</b>	<b><i>Required packages</i></b>	<b>149</b>
	<b>Python Module Index</b>	<b>151</b>
	<b>Index</b>	<b>153</b>



## GSAS-II MAIN MODULE

Main routines for the GSAS-II program

**class** GSASII.**GSASII** (*parent*)

Define the main GSAS-II frame and its associated menu items

**Bind** (*eventtype, handler, \*args, \*\*kwargs*)

Override the Bind function so that we can wrap calls that will be logged.

N.B. This is a bit kludgy. Menu bindings with an id are wrapped and menu bindings with an object and no id are not.

**CheckNotebook** ()

Make sure the data tree has the minimally expected controls.

**class** **CopyDialog** (*parent, title, text, data*)

Creates a dialog for copying control settings between data tree items

GSASII.**EnableSeqRefineMenu** ()

Enable or disable the sequential refinement menu items based on the contents of the Controls 'Seq Data' item (if present)

GSASII.**ErrorDialog** (*title, message, parent=None, wtype=4*)

Display an error message

GSASII.**ExitMain** (*event*)

Called if the main window is closed

GSASII.**FillMainMenu** (*menubar*)

Define contents of the main GSAS-II menu for the (main) data tree window. For the mac, this is also called for the data item windows as well so that the main menu items are data menu as well.

GSASII.**GetFileList** (*fileType, skip=None*)

Appears unused. Note routine of same name in GSASIIpwdGUI

GSASII.**GetHKLFdatafromTree** (*HKLFname*)

Returns single crystal data from GSASII tree

**Parameters** **HKLFname** (*str*) – a single crystal histogram name as obtained from  
GSASIIstruct.GetHistogramNames ()

**Returns** HKLFdata = single crystal data list of reflections

GSASII.**GetHistogramNames** (*hType*)

Returns a list of histogram names found in the GSASII data tree Note routine  
GSASIIstrIO.GetHistogramNames () also exists to get same info from GPX file.

**Parameters** **hType** (*str*) – list of histogram types

**Returns** list of histogram names

**GSASII.GetImportFile** (*message*, *defaultDir=''*, *defaultFile=''*, *style=1*, *\*args*, *\*\*kwargs*)

Defines a customized dialog that gets files from the appropriate Import directory (unless overridden with *defaultDir*). The default import directory is found in *self.ImportDir*, which will be set to the location of the (first) file entered in this dialog.

**Returns** a list of files or an empty list

**GSASII.GetPWDRdatafromTree** (*PWDRname*)

Returns powder data from GSASII tree

**Parameters** **PWDRname** (*str*) – a powder histogram name as obtained from *GSASIIstruct.GetHistogramNames()*

**Returns** **PWDRdata** = powder data dictionary with Powder data arrays, Limits, Instrument Parameters, Sample Parameters

**GSASII.GetPhaseData** ()

Returns a dict with defined phases. Note routine *GSASIIstrIO.GetPhaseData()* also exists to get same info from GPX file.

**GSASII.GetPhaseInfofromTree** ()

Get the phase names and their rId values, also the histograms used in each phase.

**Returns**

(*phaseRidList*, *usedHistograms*) where

- *phaseRidList* is a list of random Id values for each phase
- *usedHistograms* is a dict where the keys are the phase names and the values for each key are a list of the histogram names used in each phase.

**GSASII.GetPhaseNames** ()

Returns a list of defined phases. Note routine *GSASIIstrIO.GetPhaseNames()* also exists to get same info from GPX file.

**GSASII.GetPowderIparm** (*rd*, *prevIparm*, *lastIparmfile*, *lastdatafile*)

Open and read an instrument parameter file for a data file Returns the list of parameters used in the data tree

**Parameters**

- **rd** (*obj*) – the raw data (histogram) data object.
- **prevIparm** (*str*) – not used
- **lastIparmfile** (*str*) – Name of last instrument parameter file that was read, or a empty string.
- **lastdatafile** (*str*) – Name of last data file that was read.

**Returns** a list of two dicts, the first containing instrument parameters and the second used for TOF lookup tables for profile coeff.

**GSASII.GetUsedHistogramsAndPhasesfromTree** ()

Returns all histograms that are found in any phase and any phase that uses a histogram. This also assigns numbers to used phases and histograms by the order they appear in the file. Note routine *GSASIIstrIO.GetUsedHistogramsAndPhasesfromTree()* also exists to get same info from GPX file.

**Returns**

(*Histograms*, *Phases*)

- **Histograms** = dictionary of histograms as {*name*:*data*,...}



- Phases = dictionary of phases that use histograms

**GSASII.MakeLSParmDict ()**

Load all parameters used for computation from the tree into a dict of paired values [value, refine flag]. Note that this is different than the parmDict used in the refinement, which only has values.

Note that similar things are done in `GSASIIIO.ExportBaseclass.loadParmDict ()` (from the tree) and `GSASIIstrMain.Refine ()` and `GSASIIstrMain.SeqRefine ()` (from a GPX file).

**Returns**

(parmDict,varyList) where:

- parmDict is a dict with values and refinement flags for each parameter and
- varyList is a list of variables (refined parameters).

**GSASII.MenuBinding (event)**

Called when a menu is clicked upon; looks up the binding in table

**GSASII.OnAddPhase (event)**

Add a new, empty phase to the tree. Called by Data/Add Phase menu

**GSASII.OnDataDelete (event)**

Delete one or more histograms from data tree. Called by the Data/DeleteData menu

**GSASII.OnDeletePhase (event)**

Delete a phase from the tree. Called by Data/Delete Phase menu

**GSASII.OnDummyPowder (event)**

Called in response to Import/Powder Data/Simulate menu item to create a Dummy powder diffraction data set.

Reads an instrument parameter file and then gets input from the user

**GSASII.OnFileClose (event)**

Clears the data tree in response to the File/New Project menu button. User is given option to save the project.

**GSASII.OnFileExit (event)**

Called in response to the File/Quit menu button

**GSASII.OnFileOpen (event, filename=None)**

Gets a GSAS-II .gpx project file in response to the File/Open Project menu button

**GSASII.OnFileSave (event)**

Save the current project in response to the File/Save Project menu button

**GSASII.OnFileSaveas (event)**

Save the current project in response to the File/Save as menu button

**GSASII.OnImageRead (event)**

Called to read in an image in any known format

**GSASII.OnImageSum (event)**

Sum together image data(?)

**GSASII.OnImportGeneric (reader, readerlist, label, multiple=False, usedRanIdList=[], Pre-view=True)**

Used to import Phases or datasets using reader objects subclassed from `GSASIIIO.ImportPhase`, `GSASIIIO.ImportStructFactor`, `GSASIIIO.ImportPowderData`, `GSASIIIO.ImportSmallAngleData` or `GSASIIIO.ImportImage`. If a specific reader is specified, only that method will be called, but if no reader is specified, every one that is potentially compatible (by file extension) will be tried on the file(s) selected in the Open File dialog.

### Parameters

- **reader** (*readerobject*) – This will be a reference to a particular object to be used to read a file or None, if every appropriate reader should be used.
- **readerlist** (*list*) – a list of reader objects appropriate for the current read attempt. At present, this will be either self.ImportPhaseReaderlist, self.ImportSfactReaderlist or self.ImportPowderReaderlist (defined in `_init_Imports` from the files found in the path), but in theory this list could be tailored. Used only when reader is None.
- **label** (*str*) – string to place on the open file dialog: Open *label* input file
- **multiple** (*bool*) – True if multiple files can be selected in the file dialog. False is default. At present True is used only for reading of powder data.
- **usedRanIdList** (*list*) – an optional list of random Ids that have been used and should not be reused
- **Preview** (*bool*) – indicates if a preview of the file should be shown. Default is True, but set to False for image files which are all binary.

**Returns** a list of reader objects (`rd_list`) that were able to read the specified file(s). This list may be empty.

#### GSASII.OnImportImage (*event*)

Called in response to an Import/Image/... menu item to read an image from a file. Like all the other imports, dict self.ImportMenuId is used to look up the specific reader item associated with the menu item, which will be None for the last menu item, which is the “guess” option where all appropriate formats will be tried.

A reader object is filled each time an image is read.

#### GSASII.OnImportPhase (*event*)

Called in response to an Import/Phase/... menu item to read phase information. dict self.ImportMenuId is used to look up the specific reader item associated with the menu item, which will be None for the last menu item, which is the “guess” option where all appropriate formats will be tried.

#### GSASII.OnImportPowder (*event*)

Called in response to an Import/Powder Data/... menu item to read a powder diffraction data set. dict self.ImportMenuId is used to look up the specific reader item associated with the menu item, which will be None for the last menu item, which is the “guess” option where all appropriate formats will be tried.

Also reads an instrument parameter file for each dataset.

#### GSASII.OnImportSfact (*event*)

Called in response to an Import/Structure Factor/... menu item to read single crystal datasets. dict self.ImportMenuId is used to look up the specific reader item associated with the menu item, which will be None for the last menu item, which is the “guess” option where all appropriate formats will be tried.

#### GSASII.OnImportSmallAngle (*event*)

Called in response to an Import/Small Angle Data/... menu item to read a small angle diffraction data set. dict self.ImportMenuId is used to look up the specific reader item associated with the menu item, which will be None for the last menu item, which is the “guess” option where all appropriate formats will be tried.

#### GSASII.OnMacroRecordStatus (*event, setvalue=None*)

Called when the record macro menu item is used which toggles the value. Alternately a value to be set can be provided. Note that this routine is made more complex because on the Mac there are lots of menu items (listed in self.MacroStatusList) and this loops over all of them.

#### GSASII.OnMakePDFs (*event*)

Calculates PDFs

- GSASII.OnPatternTreeItemActivated** (*event*)  
Called when a tree item is activated
- GSASII.OnPatternTreeItemCollapsed** (*event*)  
Called when a tree item is collapsed - all children will be collapsed
- GSASII.OnPatternTreeItemDelete** (*event*)  
Called when a tree item is deleted – not sure what this does
- GSASII.OnPatternTreeItemExpanded** (*event*)  
Called when a tree item is expanded
- GSASII.OnPatternTreeKeyDown** (*event*)  
Allows stepping through the tree with the up/down arrow keys
- GSASII.OnPatternTreeSelChanged** (*event*)  
Called when a data tree item is selected
- GSASII.OnPreferences** (*event*)  
Edit the GSAS-II configuration variables
- GSASII.OnPwdrSum** (*event*)  
Sum together powder data(?)
- GSASII.OnReadPowderPeaks** (*event*)  
Bound to menu Data/Read Powder Peaks
- GSASII.OnRefine** (*event*)  
Perform a refinement. Called from the Calculate/Refine menu.
- GSASII.OnRenameData** (*event*)  
Renames an existing phase. Called by Data/Rename Phase menu
- GSASII.OnSeqRefine** (*event*)  
Perform a sequential refinement. Called from the Calculate/Sequential refine menu.
- GSASII.OnSize** (*event*)  
Called when the main window is resized. Not sure why
- GSASII.OpenPowderInstprm** (*instfile*)  
Read a GSAS-II (new) instrument parameter file  
**Parameters** *instfile* (*str*) – name of instrument parameter file
- GSASII.PreviewFile** (*filename, fp*)  
confirm we have the right file
- GSASII.ReadPowderInstprm** (*instLines*)  
Read lines from a GSAS-II (new) instrument parameter file  
**Parameters** *instLines* (*list*) – strings from GSAS-II parameter file; can be concatenated with ‘;’
- GSASII.ReadPowderIparm** (*instfile, bank, databanks, rd*)  
Read a GSAS (old) instrument parameter file  
**Parameters**
- **instfile** (*str*) – name of instrument parameter file
  - **bank** (*int*) – the bank number read in the raw data file
  - **databanks** (*int*) – the number of banks in the raw data file. If the number of banks in the data and instrument parameter files agree, then the sets of banks are assumed to match up and bank is used to select the instrument parameter file. If not, the user is asked to make a selection.

- **rd** (*obj*) – the raw data (histogram) data object. This sets rd.instbank.

GSASII.**ShowLSParms** (*event*)

Displays a window showing all parameters in the refinement. Called from the Calculate/View LS Params menu.

GSASII.**StartProject** ()

Opens a GSAS-II project file & selects the 1st available data set to display (PWDR, HKLF or SASD)

**class** GSASII.**SumDialog** (*parent, title, text, dataType, data*)

Allows user to supply scale factor(s) when summing data

**class** GSASII.**GSASIImain** (*redirect=False, filename=None, useBestVisual=False, clearSigInt=True*)

Defines a wxApp for GSAS-II

Creates a wx frame (self.main) which contains the display of the data tree.

**OnInit** ()

Called automatically when the app is created.

GSASII.**main** ()

Start up the GSAS-II application

## GSASIIOBJ: DATA OBJECTS

This module defines and/or documents the data structures used in GSAS-II, as well as provides misc. support routines.

### 2.1 Constraints Tree Item

Constraints are stored in a dict, separated into groups. Note that parameter are named in the following pattern, p:h:<var>:n, where p is the phase number, h is the histogram number <var> is a variable name and n is the parameter number. If a parameter does not depend on a histogram or phase or is unnumbered, that number is omitted. Note that the contents of each dict item is a List where each element in the list is a *constraint definition objects*. The constraints in this form are converted in `GSASIIstrIO.ProcessConstraints()` to the form used in `GSASIImapvars`

The keys in the Constraints dict are:

key	explanation
Hist	This specifies a list of constraints on histogram-related parameters, which will be of form :h:<var>:n.
HAP	This specifies a list of constraints on parameters that are defined for every histogram in each phase and are of form p:h:<var>:n.
Phase	This specifies a list of constraints on phase parameters, which will be of form p::<var>:n.
Global	This specifies a list of constraints on parameters that are not tied to a histogram or phase and are of form ::<var>:n

Each constraint is defined as an item in a list. Each constraint is of form:

```
[[<mult1>, <var1>], [<mult2>, <var2>], ..., <fixedval>, <varyflag>, <constype>]
```

Where the variable pair list item containing two values [<mult>, <var>], where:

- <mult> is a multiplier for the constraint (float)
- <var> a `G2VarObj` object (previously a str variable name of form 'p:h:name[:at]')

Note that the last three items in the list play a special role:

- <fixedval> is the fixed value for a *constant equation* (`constype=c`) constraint or is None. For a *New variable* (`constype=f`) constraint, a variable name can be specified as a str (used for externally generated constraints)
- <varyflag> is True or False for *New variable* (`constype=f`) constraints or is None. This will be implemented in the future to indicate if these variables should be refined.
- <constype> is one of four letters, 'e', 'c', 'h', 'f' that determines the type of constraint:
  - 'e' defines a set of equivalent variables. Only the first variable is refined (if the appropriate refine flag is set) and all other equivalent variables in the list are generated from that variable, using the appropriate multipliers.

- 'c' defines a constraint equation of form,  $m_1 \times var_1 + m_2 \times var_2 + \dots = c$
- 'h' defines a variable to hold (not vary). Any variable on this list is not varied, even if its refinement flag is set. Only one [mult,var] pair is allowed in a hold constraint and the mult value is ignored. This is of particular value when needing to hold one or more variables where a single flag controls a set of variables such as, coordinates, the reciprocal metric tensor or anisotropic displacement parameter.
- 'f' defines a new variable (function) according to relationship  $newvar = m_1 \times var_1 + m_2 \times var_2 + \dots$

## 2.2 Covariance Tree Item

The Covariance tree item has results from the last least-squares run. They are stored in a dict with these keys:

key	sub-key	explanation
newCellDict		dict with lattice parameters computed by <code>GSASIIstrMath.GetNewCellParms()</code> (dict)
title		Name of gpx file(?) (str)
variables		Values for all N refined variables (list of float values, length N, ordered to match varyList)
sig		Uncertainty values for all N refined variables (list of float values, length N, ordered to match varyList)
varyList		List of directly refined variables (list of str values, length N)
newAtomDict		dict with atom position values computed in <code>GSASIIstrMath.ApplyXYZshifts()</code> (dict)
Rvals		R-factors, GOF, Marquardt value for last refinement cycle (dict)
	Nobs	Number of observed data points (int)
	Rwp	overall weighted profile R-factor (% , float)
	chisq	sum[w*(Iobs-Icalc)**2] for all data note this is not the reduced chi squared (float)
	lamMax	Marquardt value applied to Hessian diagonal (float)
	GOF	The goodness-of-fit, aka square root of the reduced chi squared. (float)
covMatrix		The (NxN) covVariance matrix (np.array)

## 2.3 Phase Tree Items

Phase information is stored in the GSAS-II data tree as children of the Phases item in a dict with keys:

key	sub-key	explanation
General		Overall information for the phase (dict)
	AtomPtrs	list of four locations to use to pull info from the atom records (list)
	F000X	x-ray F(000) intensity (float)
	F000N	neutron F(000) intensity (float)
	Mydir	directory of current .gpx file (str)
	MCSA controls	Monte Carlo-Simulated Annealing controls (dict)
	Cell	List with 8 items: cell refinement flag (bool) a, b, c, (Angstrom, float) alpha, beta & gamma (degrees, float) volume (A^3, float)
	Type	'nuclear' or 'macromolecular' for now (str)
	Map	dict of map parameters
	SH Texture	dict of spherical harmonic preferred orientation parameters
	Isotope	dict of isotopes for each atom type

Continued on next page

Table 2.1 – continued from previous page

key	sub-key	explanation
ranId pId Atoms  Drawing	Isotopes	dict of scattering lengths for each isotope combination for each element in phase
	Name	phase name (str)
	SGData	Space group details as a <i>space group (SGData) object</i> as defined in <code>GSASIIspc.SpcGroup()</code> .
	Pawley neg wt	Restraint value for negative Pawley intensities (float)
	Flip	dict of Charge flip controls
	Data plot type	data plot type ('Mustrain', 'Size' or 'Preferred orientation') for powder data (str)
	Mass	Mass of unit cell contents in g/mol
	POhkl	March-Dollase preferred orientation direction
	Z	dict of atomic numbers for each atom type
	vdWRadii	dict of van der Waals radii for each atom type
	Color	Colors for atoms (list of (r,b,g) triplets)
	AtomTypes	List of atom types
	AtomMass	List of masses for atoms
	doPawley	Flag for Pawley intensity extraction (bool)
	NoAtoms	Number of atoms per unit cell of each type (dict)
	Pawley dmin	maximum Q (as d-space) to use for Pawley extraction (float)
	BondRadii	Default radius for each atom used to compute interatomic distances (list of floats)
	AngleRadii	Default radius for each atom used to compute interatomic angles (list of floats)
	DisAglCtls	Dict with distance/angle search controls, which has keys 'Name', 'AtomTypes', 'BondRadii', 'AngleRadii' which are as above except are possibly edited. Also contains 'Factors', which is a 2 element list with a multiplier for bond and angle search range [typically (0.85,0.85)].
	ranId	unique random number Id for phase (int)
	pId	Phase Id number for current project (int).
	Atoms	Atoms in phase as a list of lists. The outer list is for each atom, the inner list contains varying items depending on the type of phase, see the <i>Atom Records</i> description. (list of lists)
	Drawing	Display parameters (dict)
	ballScale	Size of spheres in ball-and-stick display (float)
	bondList	dict with bonds
	contourLevel	map contour level in $e/A^3$ (float)
	showABC	Flag to show view point triplet (bool). True=show.
viewDir	cartesian viewing direction (np.array with three elements)	
Zclip	clipping distance in A (float)	
backColor	background for plot as and R,G,B triplet (default = [0, 0, 0], black). (list with three atoms)	
selectedAtoms	List of selected atoms (list of int values)	
showRigidBodies	Flag to highlight rigid body placement	
sizeH	Size ratio for H atoms (float)	
bondRadius	Size of binds in A (float)	
atomPtrs	positions of x, type, site sym, ADP flag in Draw Atoms (list)	
viewPoint	list of lists. First item in list is [x,y,z] in fractional coordinates for the center of the plot. Second item list of previous & current atom number viewed (may be [0,0])	
showHydrogen	Flag to control plotting of H atoms.	
unitCellBox	Flag to control display of the unit cell.	

Continued on next page

Table 2.1 – continued from previous page

key	sub-key	explanation
	ellipseProb	Probability limit for display of thermal ellipsoids in % (float).
	vdwScale	Multiplier of van der Waals radius for display of vdW spheres.
	Atoms	A list of lists with an entry for each atom that is plotted.
	Zstep	Step to de/increase Z-clip (float)
	Quaternion	Viewing quaternion (4 element np.array)
	radiusFactor	Distance ratio for searching for bonds. ? Bonds are located that are within $r(Ra+Rb)$ and $(Ra+Rb)/r$ where Ra and Rb are the atomic radii.
	oldxy	previous view point (list with two floats)
	cameraPos	Viewing position in A for plot (float)
	depthFog	True if use depthFog on plot - set currently as False (bool)
RBModels		Rigid body assignments (note Rigid body definitions are stored in their own main top-level tree entry.)
Pawley ref		Pawley reflections
Histograms		A dict of dicts. The key for the outer dict is the histograms tied to this phase. The inner dict contains the combined phase/histogram parameters for items such as scale factors, size and strain parameters. (dict)
MCSA		Monte-Carlo simulated annealing parameters (dict)

## 2.4 Rigid Body Objects

Rigid body descriptions are available for two types of rigid bodies: ‘Vector’ and ‘Residue’. Vector rigid bodies are developed by a sequence of translations each with a refinable magnitude and Residue rigid bodies are described as Cartesian coordinates with defined refinable torsion angles.



key	sub-key	explanation
Vector	RBId	vector rigid bodies (dict of dict)
	AtInfo	Drad, Color: atom drawing radius & color for each atom type (dict)
	RBname	Name assigned by user to rigid body (str)
	VectMag	vector magnitudes in A (list)
	rbXYZ	Cartesian coordinates for Vector rigid body (list of 3 float)
	rbRef	3 assigned reference atom nos. in rigid body for origin definition, use center of atoms flag (list of 3 int & 1 bool)
	VectRef	refinement flags for VectMag values (list of bool)
	rbTypes	Atom types for each atom in rigid body (list of str)
	rbVect	Cartesian vectors for each translation used to build rigid body (list of lists)
	useCount	Number of times rigid body is used in any structure (int)
Residue	RBId	residue rigid bodies (dict of dict)
	AtInfo	Drad, Color: atom drawing radius & color for each atom type(dict)
	RBname	Name assigned by user to rigid body (str)
	rbXYZ	Cartesian coordinates for Residue rigid body (list of 3 float)
	rbTypes	Atom types for each atom in rigid body (list of str)
	atNames	Names of each atom in rigid body (e.g. C1,N2...) (list of str)
	rbRef	3 assigned reference atom nos. in rigid body for origin definition, use center of atoms flag (list of 3 int & 1 bool)
	rbSeq	Orig,Piv,angle,Riding (list): definition of internal rigid body torsion; origin atom (int), pivot atom (int), torsion angle (float), riding atoms (list of int)
	SelSeq	[int,int] used by SeqSizer to identify objects
	useCount	Number of times rigid body is used in any structure (int)
RBIds		unique Ids generated upon creation of each rigid body (dict)
	Vector	Ids for each Vector rigid body (list)
	Residue	Ids for each Residue rigid body (list)

## 2.5 Space Group Objects

Space groups are interpreted by `GSASIIspc.SpcGroup()` and the information is placed in a SGdata object which is a dict with these keys:

key	explanation
SpGrp	space group symbol (str)
Laue	one of the following 14 Laue classes: -1, 2/m, mmm, 4/m, 4/mmm, 3R, 3mR, 3, 3m1, 31m, 6/m, 6/mmm, m3, m3m (str)
SGInv	True if centrosymmetric, False if not (bool)
SGLatt	Lattice centering type. Will be one of P, A, B, C, I, F, R (str)
SGUniq	unique axis if monoclinic. Will be a, b, or c for monoclinic space groups. Will be blank for non-monoclinic. (str)
SGCen	Symmetry cell centering vectors. A (n,3) np.array of centers. Will always have at least one row: <code>np.array([[0, 0, 0]])</code>
SGOps	symmetry operations as a list of form <code>[[M1, T1], [M2, T2], ...]</code> where $M_n$ is a 3x3 np.array and $T_n$ is a length 3 np.array. Atom coordinates are transformed where the Asymmetric unit coordinates [X is (x,y,z)] are transformed using $X' = M_n * X + T_n$
SGSys	symmetry unit cell: type one of 'triclinic', 'monoclinic', 'orthorhombic', 'tetragonal', 'rhombohedral', 'trigonal', 'hexagonal', 'cubic' (str)
SGPolax	Axes for space group polarity. Will be one of "", 'x', 'y', 'x y', 'z', 'x z', 'y z', 'xyz'. In the case where axes are arbitrary '111' is used (P 1, and ?).

Superspace groups [3+1] are interpreted by `GSASIIspc.SSpGroup()` and the information is placed in a `SSGdata` object which is a dict with these keys:

key	explanation
<code>SSpGrp</code>	superspace group symbol extension to space group symbol, accidental spaces removed
<code>SSGCen</code>	4D cell centering vectors [0,0,0,0] at least
<code>SSGOps</code>	4D symmetry operations as [M,T] so that $M*x+T = x'$

## 2.6 Atom Records

If `phasedict` points to the phase information in the data tree, then atoms are contained in a list of atom records (list) in `phasedict['Atoms']`. Also needed to read atom information are four pointers, `cx, ct, cs, cia = phasedict['General']['atomPtrs']`, which define locations in the atom record, as shown below. Items shown are always present; additional ones for macromolecular phases are marked 'mm'

location	explanation
<code>ct-4</code>	mm - residue number (str)
<code>ct-3</code>	mm - residue name (e.g. ALA) (str)
<code>ct-2</code>	mm - chain label (str)
<code>ct-1</code>	atom label (str)
<code>ct</code>	atom type (str)
<code>ct+1</code>	refinement flags; combination of 'F', 'X', 'U' (str)
<code>cx,cx+1,cx+2</code>	the x,y and z coordinates (3 floats)
<code>cs</code>	site symmetry (str)
<code>cs+1</code>	site multiplicity (int)
<code>cia</code>	ADP flag: Isotropic ('I') or Anisotropic ('A')
<code>cia+1</code>	Uiso (float)
<code>cia+2...cia+7</code>	U11, U22, U33, U12, U13, U23 (6 floats)
<code>atom[cia+8]</code>	unique atom identifier (int)

## 2.7 Drawing Atom Records

If `phasedict` points to the phase information in the data tree, then drawing atoms are contained in a list of drawing atom records (list) in `phasedict['Drawing']['Atoms']`. Also needed to read atom information are four pointers, `cx, ct, cs, ci = phasedict['Drawing']['AtomPtrs']`, which define locations in the atom record, as shown below. Items shown are always present; additional ones for macromolecular phases are marked 'mm'

location	explanation
ct-4	mm - residue number (str)
ct-3	mm - residue name (e.g. ALA) (str)
ct-2	mm - chain label (str)
ct-1	atom label (str)
ct	atom type (str)
cx,cx+1,cx+2	the x,y and z coordinates (3 floats)
cs-1	Sym Op symbol; sym. op number + unit cell id (e.g. '1,0,-1') (str)
cs	atom drawing style; e.g. 'balls & sticks' (str)
cs+1	atom label style (e.g. 'name') (str)
cs+2	atom color (RGB triplet) (int)
cs+3	ADP flag: Isotropic ('I') or Anisotropic ('A')
cs+4	Uiso (float)
cs+5...cs+11	U11, U22, U33, U12, U13, U23 (6 floats)
ci	unique atom identifier; matches source atom Id in Atom Records (int)

## 2.8 Powder Diffraction Tree Items

Every powder diffraction histogram is stored in the GSAS-II data tree with a top-level entry named beginning with the string "PWDR". The diffraction data for that information are directly associated with that tree item and there are a series of children to that item. The routines `GSASII.GSASII.GetUsedHistogramsAndPhasesfromTree()` and `GSASIIstrIO.GetUsedHistogramsAndPhases()` will load this information into a dictionary where the child tree name is used as a key, and the information in the main entry is assigned a key of `Data`, as outlined below.

key	sub-key	explanation
Comments		Text strings extracted from the original powder data header; can be changed by the user; it may be empty.
Limits		A list of two two element lists, as <code>[[Ld,Hd],[L,H]]</code> where <code>Ld</code> and <code>L</code> are the current and default lowest two-theta value to be used and <code>Hd</code> and <code>H</code> are the current and default highest two-theta value.
Reflection Lists		A dict with an entry for each phase in the histogram. Each dict item is a dict containing reflections, as described in the <a href="#">Reflections</a> description.
Instrument Parameters		A list containing two dicts where the possible keys in each dict are listed below. The value for each item is a list containing three values: the current value, the default value, and a refinement flag which can be True, False or 0 where 0 indicates a value that cannot be refined. The first and second values are floats unless otherwise noted. The first dict are noted as [1]
	Lam	Specifies a wavelength in Angstroms [1]
	Lam1	Specifies the primary wavelength in Angstrom, when an alternative source is used [1]
	Lam2 I(L2)/I(L1)	Specifies the secondary wavelength in Angstrom, when an alternative source is used [1] Ratio of Lam2 to Lam1 [1]
	Type	<p><b>Histogram type (str) [1]:</b></p> <ul style="list-style-type: none"> <li>'PXC' for constant wavelength x-ray</li> <li>'PNC' for constant wavelength neutron</li> <li>'PNT' for time of flight neutron</li> </ul>
	Zero	Two-theta zero correction in <i>degrees</i> [1]

Continued

Table 2.2 – continued from previous page

key	sub-key	explanation
Sample Parameters	Azimuth	Azimuthal setting angle for data recorded with differin [1]
	U, V, W	Cagliotti profile coefficients for Gaussian instrumen where the FWHM goes as $U \tan^2 \theta + V \tan \theta + W$ [1]
	X, Y SH/L	Cauchy (Lorentzian) instrumental broadening coefficient Variant of the Finger-Cox-Jephcoat asymmetric peak b Note that this is the average between S/L and H/L wh height, H is the slit height and L is the goniometer diam
	Polariz.	Polarization coefficient. [1]
	wtFactor	A weighting factor to increase or decrease the leverage o togram (float). A value of 1.0 weights the data with their tainties and a larger value increases the weighting of the to decreasing the uncertainties).
	Scale	Specifies a dict with parameters that describe how the lected, as listed below. Refinable parameters are a list co and a bool, where the second value specifies if the value erwise the value is a float unless otherwise noted.
	Absorption	The histogram scale factor (refinable) The sample absorption coefficient as $\mu r$ where r is the ra Only valid for Debye-Scherrer geometry.
	SurfaceRoughA	Surface roughness parameter A as defined by Surotti, 5,325-331, 1972.(refinable - only valid for Bragg-Brenta
	SurfaceRoughB	Surface roughness parameter B (refinable - only va Brentano geometry)
	DisplaceX, DisplaceY	Sample displacement from goniometer center where Y is direction and X is perpendicular. Units are $\mu m$ (refinable)
	Phi, Chi, Omega	Goniometer sample setting angles, in degrees.
	Gonio. radius	Radius of the diffractometer in mm
	InstrName	A name for the instrument, used in preparing a CIF (str).
	Force, Temperature, Humidity, Pressure, Voltage	Variables that describe how the measurement was perfor directly in any computations.
	hId	ranId
Type		Type of diffraction data, may be 'Debye-Scherrer' or 'B (str).
Diffuse		not in use?
Background	ranId	The number assigned to the histogram when the proje edited (can change)
	Background	A random number id for the histogram that does not cha The background is stored as a list with where the first is list and the second item is a dict. The list contains function and its coefficients; the dict contains Debye di background peaks. (TODO: this needs to be expanded.)
Data		The data consist of a list of 6 np.arrays containing in ord <ul style="list-style-type: none"> <li>0. the x-positions (two-theta in degrees),</li> <li>1. the intensity values (Yobs),</li> <li>2. the weights for each Yobs value</li> <li>3. the computed intensity values (Ycalc)</li> <li>4. the background values</li> <li>5. Yobs-Ycalc</li> </ul>

## 2.9 Powder Reflection Data Structure

For every phase in a histogram, the `Reflection Lists` value is a dict one element of which is `RefList`, which is a `np.array` containing reflections. The columns in that array are documented below.

index	explanation
0,1,2	h,k,l (float)
3	multiplicity
4	d-space, Angstrom
5	pos, two-theta
6	sig, Gaussian width
7	gam, Lorentzian width
8	$F_{obs}^2$
9	$F_{calc}^2$
10	reflection phase, in degrees
11	intensity correction for reflection, this times $F_{obs}^2$ or $F_{calc}^2$ gives $I_{obs}$ or $I_{calc}$

## 2.10 Single Crystal Tree Items

Every single crystal diffraction histogram is stored in the GSAS-II data tree with a top-level entry named beginning with the string "HKLFL". The diffraction data for that information are directly associated with that tree item and there are a series of children to that item. The routines `GSASII.GSASII.GetUsedHistogramsAndPhasesfromTree()` and `GSASIIstrIO.GetUsedHistogramsAndPhases()` will load this information into a dictionary where the child tree name is used as a key, and the information in the main entry is assigned a key of `Data`, as outlined below.

key	sub-key	explanation
Data		A dict that contains the reflection table, as described in the <i>Single Crystal Reflections</i> description.
Instrument Parameters		A list containing two dicts where the possible keys in each dict are listed below. The value for most items is a list containing two values: the initial value, the current value. The first and second values are floats unless otherwise noted.
	Lam Type	Specifies a wavelength in Angstroms (two floats)  <b>Histogram type (two str values):</b> <ul style="list-style-type: none"> <li>• 'SXC' for constant wavelength x-ray</li> <li>• 'SNC' for constant wavelength neutron</li> <li>• 'SNT' for time of flight neutron</li> </ul>
wtFactor	InstrName	A name for the instrument, used in preparing a CIF (str). A weighting factor to increase or decrease the leverage of data in the histogram (float). A value of 1.0 weights the data with their standard uncertainties and a larger value increases the weighting of the data (equivalent to decreasing the uncertainties).
hId		The number assigned to the histogram when the project is loaded or edited (can change)
ranId		A random number id for the histogram that does not change

## 2.11 Single Crystal Reflection Data Structure

For every single crystal a histogram, the 'Data' item contains the structure factors as an np.array in item 'RefList'. The columns in that array are documented below.

index	explanation
0,1,2	h,k,l (float)
3	multiplicity
4	d-space, Angstrom
5	$F_{obs}^2$
6	$\sigma(F_{obs}^2)$
7	$F_{calc}^2$
8	$F_{obs}^2 T$
9	$F_{calc}^2 T$
10	reflection phase, in degrees
11	intensity correction for reflection, this times $F_{obs}^2$ or $F_{calc}^2$ gives Iobs or Icalc

## 2.12 Image Data Structure

Every 2-dimensional image is stored in the GSAS-II data tree with a top-level entry named beginning with the string "IMG ". The image data are directly associated with that tree item and there are a series of children to that item. The routines `GSASII.GSASII.GetUsedHistogramsAndPhasesfromTree()` and `GSASIIstrIO.GetUsedHistogramsAndPhases()` will load this information into a dictionary where the child tree name is used as a key, and the information in the main entry is assigned a key of Data, as outlined below.

key	sub-key	explanation
Comments		Text strings extracted from the original image data header or a metafile. These cannot be changed by the user; it may be empty.
Image Controls	azmthOff	(float) The offset to be applied to an azimuthal value. Accomodates detector orientations other than with the detector X-axis horizontal.
	background image	(list:str,float) The name of a tree item ("IMG ...") that is to be subtracted during image integration multiplied by value. It must have the same size/shape as the integrated image. NB: value < 0 for subtraction.
	calibrant	(str) The material used for determining the position/orientation of the image. The data is obtained from <code>ImageCalibrants()</code> and <code>UserCalibrants.py</code> (supplied by user).
	calibdmin	(float) The minimum d-spacing used during the last calibration run.
	calibskip	(int) The number of expected diffraction lines skipped during the last calibration run.
	center	(list:floats) The [X,Y] point in detector coordinates (mm) where the direct beam strikes the detector plane as determined by calibration. This point does not have to be within the limits of the detector boundaries.
	centerAzm	(bool) If True then the azimuth reported for the integrated slice of the image is at the center line otherwise it is at the leading edge.
	color	(str) The name of the colormap used to display the image. Default = 'Paired'.
	cutoff	(float) The minimum value of I/Ib for a point selected in a diffraction ring for calibration calculations. See <code>pixLimit</code> for details as how point is found.
		Continued on next page

Table 2.3 – continued from previous page

key	sub-key	explanation
	DetDepth	(float) Coefficient for penetration correction to distance; accounts for diffraction ring offset at higher angles. Optionally determined by calibration.
	DetDepthRef	(bool) If True then refine DetDepth during calibration/recalibration calculation.
	distance	(float) The distance (mm) from sample to detector plane.
	ellipses	(list:lists) Each object in ellipses is a list [center,phi,radii,color] where center (list) is location (mm) of the ellipse center on the detector plane, phi is the rotation of the ellipse minor axis from the x-axis, and radii are the minor & major radii of the ellipse. If radii[0] is negative then parameters describe a hyperbola. Color is the selected drawing color (one of 'b', 'g', 'r') for the ellipse/hyperbola.
	edgemin	(float) Not used; parameter in EdgeFinder code.
	fullIntegrate	(bool) If True then integrate over full 360 deg azimuthal range.
	GonioAngles	(list:floats) The 'Omega','Chi','Phi' goniometer angles used for this image. Required for texture calculations.
	invert_x	(bool) If True display the image with the x-axis inverted.
	invert_y	(bool) If True display the image with the y-axis inverted.
	IOth	(list:floats) The minimum and maximum 2-theta values to be used for integration.
	LRazimuth	(list:floats) The minimum and maximum azimuth values to be used for integration.
	Oblique	(list:float,bool) If True apply a detector absorption correction using the value to the intensities obtained during integration.
	outAzimuths	(int) The number of azimuth pie slices.
	outChannels	(int) The number of 2-theta steps.
	pixelSize	(list:ints) The X,Y dimensions (microns) of each pixel.
	pixLimit	(int) A box in the image with 2*pixLimit+1 edges is searched to find the maximum. This value (I) along with the minimum (Ib) in the box is reported by <code>GSASIIimage.ImageLocalMax()</code> and subject to cut-off in <code>GSASIIimage.makeRing()</code> . Locations are used to construct rings of points for calibration calculations.
	PolaVal	(list:float,bool) If type='SASD' and if True, apply polarization correction to intensities from integration using value.
	rings	(list:lists) Each entry is [X,Y,dsp] where X & Y are lists of x,y coordinates around a diffraction ring with the same d-spacing (dsp)
	ring	(list) The x,y coordinates of the >5 points on an inner ring selected by the user,
	Range	(list) The minimum & maximum values of the image
	rotation	(float) The angle between the x-axis and the vector about which the detector is tilted. Constrained to -180 to 180 deg.
	SampleShape	(str) Currently only 'Cylinder'. Sample shape for Debye-Scherrer experiments; used for absorption calculations.
	SampleAbs	(list: float,bool) Value of absorption coefficient for Debye-Scherrer experiments, flag if True to cause correction to be applied.
	setDefault	(bool) If True the use the image controls values for all new images to be read. (might be removed)
	setRings	(bool) If True then display all the selected x,y ring positions (vida supra rings) used in the calibration.
	showLines	(bool) If True then isplay the integration limits to be used.
	size	(list:int) The number of pixels on the image x & y axes

Continued on next page

Table 2.3 – continued from previous page

key	sub-key	explanation
Masks	type	(str) One of ‘PWDR’, ‘SASD’ or ‘REFL’ for powder, small angle or reflectometry data, respectively.
	tilt	(float) The angle the detector normal makes with the incident beam; range -90 to 90.
	wavelength	(float) The radiation wavelength (Angstroms) as entered by the user (or someday obtained from the image header).
	Arcs	(list: lists) Each entry [2-theta,[azimuth[0],azimuth[1]],thickness] describes an arc mask to be excluded from integration
	Frames	(list:lists) Each entry describes the x,y points (3 or more - mm) that describe a frame outside of which is excluded from recalibration and integration. Only one frame is allowed.
	Points	(list:lists) Each entry [x,y,radius] (mm) describes an excluded spot on the image to be excluded from integration.
	Polygons	(list:lists) Each entry is a list of 3+ [x,y] points (mm) that describe a polygon on the image to be excluded from integration.
	Rings	(list: lists) Each entry [2-theta,thickness] describes a ring mask to be excluded from integration.
	Thresholds	(list:[tuple,list]) [(Imin,Imax],[Imin,Imax]] This gives lower and upper limits for points on the image to be included in integrsation. The tuple is the image intensity limits and the list are those set by the user.
Stress/Strain	Sample phi	(float) Sample rotation about vertical axis.
	Sample z	(float) Sample translation from the calibration sample position (for Sample phi = 0) These will be restricted by space group symmetry; result of strain fit refinement.
	Type	(str) ‘True’ or ‘Conventional’: The strain model used for the calculation.
	d-zero	(list:dict) Each item is for a diffraction ring on the image; all items are from the same phase and are used to determine the strain tensor. The dictionary items are: ‘Dset’: (float) True d-spacing for the diffraction ring; entered by the user. ‘Dcalc’: (float) Average calculated d-spacing determined from strain coeff. ‘Emat’: (list: float) The strain tensor elements e11, e12 & e22 (e21=e12, rest are 0) ‘Esig’: (list: float) Esds for Emat from fitting. ‘pixLimit’: (int) Search range to find highest point on ring for each data point ‘cutoff’: (float) I/Ib cutoff for searching. ‘ImxyObs’: (list: lists) [[X],[Y]] observed points to be used for strain calculations. ‘ImtaObs’: (list: lists) [[d],[azm]] transformed via detector calibration from ImxyObs. ‘ImtaCalc’: (list: lists) [[d],[azm]] calculated d-spacing & azimuth from fit.

## 2.13 Parameter Dictionary

The parameter dictionary contains all of the variable parameters for the refinement. The dictionary keys are the name of the parameter (<phase>:<hist>:<name>:<atom>). It is prepared in two ways. When loaded from the tree (in `GSASII.GSASII.MakeLSParmDict()` and `GSASIIIO.ExportBaseclass.loadParmDict()`), the values are lists with two elements: [value, refine flag]

When loaded from the GPX file (in `GSASIIstrMain.Refine()` and `GSASIIstrMain.SeqRefine()`), the value in the dict is the actual parameter value (usually a float, but sometimes a letter or string flag value (such as I or A for iso/anisotropic)).



## 2.14 Classes and routines

**GSASIIobj.AtomIdLookup = {}**

dict listing for each phase index as a str, the atom label and atom random Id, keyed by atom sequential index as a str; best to access this using `LookupAtomLabel()`

**GSASIIobj.AtomRanIdLookup = {}**

dict listing for each phase the atom sequential index keyed by atom random Id; best to access this using `LookupAtomId()`

**GSASIIobj.CompileVarDesc()**

Set the values in the variable description lookup table (`VarDesc`) into `reVarDesc`. This is called in `getDescr()` so the initialization is always done before use.

Note that keys may contain regular expressions, where '[xyz]' matches 'x' 'y' or 'z' (equivalently '[x-z]' describes this as range of values). '.' matches any string. For example:

```
'AUiso': 'Atomic isotropic displacement parameter',
```

will match variable 'p:AUiso:a'. If parentheses are used in the key, the contents of those parentheses can be used in the value, such as:

```
'AU([123][123]'):'Atomic anisotropic displacement parameter U\1',
```

will match AU11, AU23,.. and U11, U23 etc will be displayed in the value when used.

**GSASIIobj.DefaultControls = {'SeqPseudoVars': {}, 'Author': 'no name', 'Reverse Seq': False, 'Copy2Next': False, 'Seq'**

Values to be used as defaults for the initial contents of the `Controls` data tree item.

**class GSASIIobj.ExpressionCalcObj (exprObj)**

An object used to evaluate an expression from a `ExpressionObj` object.

**Parameters** `exprObj (ExpressionObj)` – a `ExpressionObj` expression object with an expression string and mappings for the parameter labels in that object.

**EvalExpression ()**

Evaluate an expression. Note that the expression and mapping are taken from the `ExpressionObj` expression object and the parameter values were specified in `SetupCalc()`. :returns: a single value for the expression. If parameter values are arrays (for example, from wild-carded variable names), the sum of the resulting expression is returned.

For example, if the expression is 'A\*B', where A is 2.0 and B maps to '1: :Afrac:\*', which evaluates to:

```
[0.5, 1, 0.5]
```

then the result will be 4.0.

**SetupCalc (parmDict)**

Do all preparations to use the expression for computation. Adds the free parameter values to the parameter dict (parmDict).

**UpdateDict (parmDict)**

Update the dict for the expression with values in a dict :param list parmDict: a dict of values some of which may be in use here

**UpdateVars (varList, valList)**

Update the dict for the expression with a set of values :param list varList: a list of variable names :param list valList: a list of corresponding values

**compiledExpr = None**

The expression as compiled byte-code

**eObj = None**

The expression and mappings; a `ExpressionObj` object

**exprDict = None**

dict that defines values for labels used in expression and packages referenced by functions

**fxnpkgdict = None**

a dict with references to packages needed to find functions referenced in the expression.

**lblLookup = None**

Lookup table that specifies the expression label name that is tied to a particular GSAS-II parameters in the `parmDict`.

**varLookup = None**

Lookup table that specifies the GSAS-II variable(s) indexed by the expression label name. (Used for only for diagnostics not evaluation of expression.)

**class** `GSASIIobj.ExpressionObj`

Defines an object with a user-defined expression, to be used for secondary fits or restraints. Object is created null, but is changed using `LoadExpression()`. This contains only the minimum information that needs to be stored to save and load the expression and how it is mapped to GSAS-II variables.

**CheckVars ()**

Check that the expression can be parsed, all functions are defined and that input loaded into the object is internally consistent. If not an Exception is raised.

**Returns** a dict with references to packages needed to find functions referenced in the expression.

**EditExpression** (*exprVarLst, varSelect, varName, varValue, varRefflag*)

Load the expression and associated settings from the object into arrays used for editing.

**Parameters**

- **exprVarLst** (*list*) – parameter labels found in the expression
- **varSelect** (*dict*) – this will be 0 for Free parameters and non-zero for expression labels linked to G2 variables.
- **varName** (*dict*) – Defines a name (str) associated with each free parameter
- **varValue** (*dict*) – Defines a value (float) associated with each free parameter
- **varRefflag** (*dict*) – Defines a refinement flag (bool) associated with each free parameter

**Returns** the expression as a str

**GetDepVar ()**

return the dependent variable, or None

**GetIndependentVars ()**

Returns the names of the required independent parameters used in expression

**GetVaried ()**

Returns the names of the free parameters that will be refined

**GetVariedVarVal ()**

Returns the names and values of the free parameters that will be refined

**LoadExpression** (*expr, exprVarLst, varSelect, varName, varValue, varRefflag*)

Load the expression and associated settings into the object. Raises an exception if the expression is not parsed, if not all functions are defined or if not all needed parameter labels in the expression are defined.

This will not test if the variable referenced in these definitions are actually in the parameter dictionary. This is checked when the computation for the expression is done in `SetupCalc()`.

### Parameters

- **expr** (*str*) – the expression
- **exprVarLst** (*list*) – parameter labels found in the expression
- **varSelect** (*dict*) – this will be 0 for Free parameters and non-zero for expression labels linked to G2 variables.
- **varName** (*dict*) – Defines a name (*str*) associated with each free parameter
- **varValue** (*dict*) – Defines a value (*float*) associated with each free parameter
- **varRefflag** (*dict*) – Defines a refinement flag (*bool*) associated with each free parameter

### ParseExpression (*expr*)

Parse an expression and return a dict of called functions and the variables used in the expression. Returns `None` in case an error is encountered. If packages are referenced in functions, they are loaded and the functions are looked up into the modules global workspace.

Note that no changes are made to the object other than saving an error message, so that this can be used for testing prior to the save.

**Returns** a list of used variables

### SetDepVar (*var*)

Set the dependent variable, if used

### UpdateVariedVars (*varyList, values*)

Updates values for the free parameters (after a refinement); only updates refined vars

### assgnVars = None

A dict where keys are label names in the expression mapping to a GSAS-II variable. The value a G2 variable name. Note that the G2 variable name may contain a wild-card and correspond to multiple values.

### expression = None

The expression as a text string

### freeVars = None

A dict where keys are label names in the expression mapping to a free parameter. The value is a list with:

- a name assigned to the parameter
- a value for to the parameter and
- a flag to determine if the variable is refined.

### lastError = None

Shows last encountered error in processing expression (list of 1-3 *str* values)

### class GSASIIobj.G2VarObj (\*args)

Defines a GSAS-II variable either using the phase/atom/histogram unique Id numbers or using a character string that specifies variables by phase/atom/histogram number (which can change). Note that `LoadID()` should be used to (re)load the current Ids before creating or later using the `G2VarObj` object.

This can store rigid body variables, but does not translate the residue # and body # to/from random Ids

A `G2VarObj` object can be created with a single parameter:

**Parameters** *varname* (*str/tuple*) –

a single value can be used to create a `G2VarObj` object. If a string, it must be of form “p:h:var” or “p:h:var:a”, where

- p is the phase number (which may be left blank or may be '\*' to indicate all phases);
- h is the histogram number (which may be left blank or may be '\*' to indicate all histograms);
- a is the atom number (which may be left blank in which case the third colon is omitted). The atom number can be specified as '\*' if a phase number is specified (not as '\*'). For rigid body variables, specify a will be a string of form "residue:body#"

Alternately a single tuple of form (Phase,Histogram,VarName,AtomID) can be used, where Phase, Histogram, and AtomID are None or are ranId values (or one can be '\*') and VarName is a string. Note that if Phase is '\*' then the AtomID is an atom number. For a rigid body variables, AtomID is a string of form "residue:body#".

If four positional arguments are supplied, they are:

#### Parameters

- **phasenum** (*str/int*) – The number for the phase (or None or '\*')
- **histnum** (*str/int*) – The number for the histogram (or None or '\*')
- **varname** (*str*) – a single value can be used to create a `G2VarObj`
- **atomnum** (*str/int*) – The number for the atom (or None or '\*')

**varname** ()

Formats the GSAS-II variable name as a "traditional" GSAS-II variable string (p:h:<var>:a) or (p:h:<var>)

**Returns** the variable name as a str

`GSASIIobj.GenWildcard` (*varlist*)

Generate wildcard versions of G2 variables. These introduce '\*' for a phase, histogram or atom number (but only for one of these fields) but only when there is more than one matching variable in the input variable list. So if the input is this:

```
varlist = ['0::AUiso:0', '0::AUiso:1', '1::AUiso:0']
```

then the output will be this:

```
wildList = ['*::AUiso:0', '0::AUiso:*']
```

**Parameters** *varlist* (*list*) – an input list of GSAS-II variable names (such as 0::AUiso:0)

**Returns** wildList, the generated list of wild card variable names.

`GSASIIobj.HistIdLookup` = {}

dict listing histogram name and random Id, keyed by sequential histogram index as a str; best to access this using `LookupHistName()`

`GSASIIobj.HistRanIdLookup` = {}

dict listing histogram sequential index keyed by histogram random Id; best to access this using `LookupHistId()`

`GSASIIobj.IndexAllIds` (*Histograms, Phases*)

Scan through the used phases & histograms and create an index to the random numbers of phases, histograms and atoms. While doing this, confirm that assigned random numbers are unique – just in case lightning strikes twice in the same place.

Note: this code assumes that the atom random Id (ranId) is the last element each atom record.

This is called in three places (only): `GSASIIstrIO.GetUsedHistogramsAndPhases()` (which loads the histograms and phases from a GPX file), `GetUsedHistogramsAndPhasesfromTree()` (which loads

the histograms and phases from the data tree.) and `GSASIIconstrGUI.UpdateConstraints()` (which displays & edits the constraints in a GUI)

TODO: do we need a lookup for rigid body variables?

`GSASIIobj.LookupAtomId (pId, ranId)`

Get the atom number from a phase and atom random Id

**Parameters**

- **pId** (*int/str*) – the sequential number of the phase
- **ranId** (*int*) – the random Id assigned to an atom

**Returns** the index number of the atom (str)

`GSASIIobj.LookupAtomLabel (pId, index)`

Get the atom label from a phase and atom index number

**Parameters**

- **pId** (*int/str*) – the sequential number of the phase
- **index** (*int*) – the index of the atom in the list of atoms

**Returns** the label for the atom (str) and the random Id of the atom (int)

`GSASIIobj.LookupHistId (ranId)`

Get the histogram number and name from a histogram random Id

**Parameters** **ranId** (*int*) – the random Id assigned to a histogram

**Returns** the sequential Id (hId) number for the histogram (str)

`GSASIIobj.LookupHistName (hId)`

Get the histogram number and name from a histogram Id

**Parameters** **hId** (*int/str*) – the sequential assigned to a histogram

**Returns** (hist,ranId) where hist is the name of the histogram (str) and ranId is the random # id for the histogram (int)

`GSASIIobj.LookupPhaseId (ranId)`

Get the phase number and name from a phase random Id

**Parameters** **ranId** (*int*) – the random Id assigned to a phase

**Returns** the sequential Id (pId) number for the phase (str)

`GSASIIobj.LookupPhaseName (pId)`

Get the phase number and name from a phase Id

**Parameters** **pId** (*int/str*) – the sequential assigned to a phase

**Returns** (phase,ranId) where phase is the name of the phase (str) and ranId is the random # id for the phase (int)

`GSASIIobj.LookupWildcard (varname, varlist)`

returns a list of variable names from list varname that match wildcard name in varname

**Parameters**

- **varname** (*str*) – a G2 variable name containing a wildcard (such as `*::var`)
- **varlist** (*list*) – the list of all variable names used in the current project

**Returns** a list of matching GSAS-II variables (may be empty)

`GSASIIobj.MakeUniqueLabel (lbl, labellist)`

Make sure that every a label is unique against a list by adding digits at the end until it is not found in list.

**Parameters**

- **lbl** (*str*) – the input label
- **labellist** (*list*) – the labels that have already been encountered

**Returns** *lbl* if not found in *labellist* or *lbl* with `_1–9` (or `_10–99`, etc.) appended at the end

`GSASIIobj.PhaseIdLookup = {}`

dict listing phase name and random Id keyed by sequential phase index as a str; best to access this using `LookupPhaseName ()`

`GSASIIobj.PhaseRanIdLookup = {}`

dict listing phase sequential index keyed by phase random Id; best to access this using `LookupPhaseId ()`

`GSASIIobj.ShortHistNames = {}`

a dict containing a possibly shortened and when non-unique numbered version of the histogram name. Keyed by the histogram sequential index.

`GSASIIobj.ShortPhaseNames = {}`

a dict containing a possibly shortened and when non-unique numbered version of the phase name. Keyed by the phase sequential index.

`GSASIIobj.VarDesc = {}`

This dictionary lists descriptions for GSAS-II variables, as set in `CompileVarDesc ()`. See that function for a description for how keys and values are written.

`GSASIIobj.VarDescr (varname)`

Return two strings with a more complete description for a GSAS-II variable

**Parameters** **name** (*str*) – A full G2 variable name with 2 or 3 or 4 colons (`<p><h>:name[:<a>]` or `<p>::RBname:<r>:<t>`)

**Returns** (*loc,meaning*) where *loc* describes what item the variable is mapped (phase, histogram, etc.) and *meaning* describes what the variable does.

`GSASIIobj.fmtVarDescr (varname)`

Return a string with a more complete description for a GSAS-II variable

**Parameters** **varname** (*str*) – A full G2 variable name with 2 or 3 or 4 colons (`<p><h>:name[:<a>]` or `<p>::RBname:<r>:<t>`)

**Returns** a string with the description

`GSASIIobj.getDescr (name)`

Return a short description for a GSAS-II variable

**Parameters** **name** (*str*) – The descriptive part of the variable name without colons (:)

**Returns** a short description or *None* if not found

`GSASIIobj.getVarDescr (varname)`

Return a short description for a GSAS-II variable

**Parameters** **name** (*str*) – A full G2 variable name with 2 or 3 or 4 colons (`<p><h>:name[:<a1>][:<a2>]`)

**Returns** a six element list as [*p*,*h*,*name*,*a1*,*a2*,*description*], where *p*, *h*, *a1*, *a2* are str values or *None*, for the phase number, the histogram number and the atom number; *name* will always be a str; and *description* is str or *None*. If the variable name is incorrectly formed (for example, wrong number of colons), *None* is returned instead of a list.

`GSASIIobj.reVarDesc = {}`

This dictionary lists descriptions for GSAS-II variables with the same values as `VarDesc` except that keys have been compiled as regular expressions. Initialized in `CompileVarDesc()`.





## GSAS-II UTILITY MODULES

### 3.1 GSASIIdata: Data for computations

At present this module defines one dict, `ramachandranDist`, which contains arrays for All and specific amino acids

### 3.2 GSASIIpath: locations & updates

Routines for dealing with file locations, etc.

Determines the location of the compiled (.pyd or .so) libraries.

Interfaces with subversion (svn): Determine the subversion release number by determining the highest version number where `SetVersionNumber()` is called (best done in every GSASII file). Other routines will update GSASII from the subversion server if svn can be found.

Accesses configuration options, as defined in `config.py`

`GSASIIpath.DoNothing()`

A routine that does nothing. This is called in place of `IPyBreak` and `pdbBreak` except when the debug option is set True in `config.py`

`GSASIIpath.GetConfigValue(key, default=None)`

Return the configuration file value for key or a default value if not present

#### Parameters

- **key** (*str*) – a value to be found in the configuration (`config.py`) file
- **default** – a value to be supplied is none is in the config file or the config file is not found. Defaults to None

**Returns** the value found or the default.

`GSASIIpath.GetVersionNumber()`

Return the maximum version number seen in `SetVersionNumber()`

`GSASIIpath.IPyBreak()`

A routine that does nothing. This is called in place of `IPyBreak` and `pdbBreak` except when the debug option is set True in `config.py`

`GSASIIpath.IPyBreak_base()`

A routine that invokes an IPython session at the calling location This routine is only used when `debug=True` is set in `config.py`

`GSASIIpath.InvokeDebugOpts ()`

Called in `GSASII.py` to set up debug options

`GSASIIpath.LoadConfigFile (filename)`

Read a GSAS-II configuration file. Comments (starting with “%”) are removed, as are empty lines

**Parameters** `filename` (*str*) – base file name (such as ‘file.dat’). Files with this name are located from the path and the contents of each are concatenated.

**Returns** a list containing each non-empty (after removal of comments) line found in every matching config file.

`GSASIIpath.SetConfigValue (parmdict)`

Set configuration variables from a dictionary where elements are lists First item in list is the default value and second is the value to use.

`GSASIIpath.SetVersionNumber (RevString)`

Set the subversion version number

**Parameters** `RevString` (*str*) – something like “\$Revision: 1971 \$” that is set by subversion when the file is retrieved from subversion.

Place `GSASIIpath.SetVersionNumber (“$Revision: 1971 $”)` in every python file.

`GSASIIpath.exceptHook (*args)`

A routine to be called when an exception occurs. It prints the traceback with fancy formatting and then calls an IPython shell with the environment of the exception location.

This routine is only used when `debug=True` is set in `config.py`

`GSASIIpath.pdbBreak ()`

A routine that does nothing. This is called in place of `IPyBreak` and `pdbBreak` except when the debug option is set `True` in `config.py`

`GSASIIpath.svnFindLocalChanges (fpath=’/Users/toby/software/G2/GSASII’)`

**Returns a list of files that were changed locally. If no files are changed, the list has length 0**

**Parameters** `fpath` – path to repository dictionary, defaults to directory where the current file is located

**Returns** None if there is a subversion error (likely because the path is not a repository or svn is not found)

`GSASIIpath.svnGetLog (fpath=’/Users/toby/software/G2/GSASII’, version=None)`

Get the revision log information for a specific version of the specified package

**Parameters**

- **fpath** (*str*) – path to repository dictionary, defaults to directory where the current file is located.
- **version** (*int*) – the version number to be looked up or None (default) for the latest version.

**Returns** a dictionary with keys (one hopes) ‘author’, ‘date’, ‘msg’, and ‘revision’

`GSASIIpath.svnGetRev (fpath=’/Users/toby/software/G2/GSASII’, local=True)`

Obtain the version number for the either the last update of the local version or contacts the subversion server to get the latest update version (# of Head).

**Parameters**

- **fpath** (*str*) – path to repository dictionary, defaults to directory where the current file is located

- **local** (*bool*) – determines the type of version number, where True (default): returns the latest installed update False: returns the version number of Head on the server

**Returns** the version number as an str or None if there is a subversion error (likely because the path is not a repository or svn is not found)

GSASIIpath.**svnInstallDir** (*URL, loadpath*)

Load a subversion tree into a specified directory

**Parameters**

- **rpath** (*str*) – path to locate files, relative to the GSAS-II installation path (defaults to path2GSAS2)
- **URL** (*str*) – the repository URL

GSASIIpath.**svnSwitchDir** (*rpath, filename, baseURL, loadpath=None*)

This performs a switch command to move files between subversion trees.

This is currently used for moving tutorial web pages and demo files into the GSAS-II source tree. Note that if the files were previously downloaded the switch command will update the files to the newest version.

**Parameters**

- **rpath** (*str*) – path to locate files, relative to the GSAS-II installation path (defaults to path2GSAS2)
- **URL** (*str*) – the repository URL
- **loadpath** (*str*) – the prefix for the path, if specified. Defaults to path2GSAS2

GSASIIpath.**svnUpdateDir** (*fpath='/Users/toby/software/G2/GSASII', version=None*)

This performs an update of the files in a local directory from a server.

**Parameters**

- **fpath** (*str*) – path to repository dictionary, defaults to directory where the current file is located
- **version** – the number of the version to be loaded. Used only cast as a string, but should be an integer or something that corresponds to a string representation of an integer value when cast. A value of None (default) causes the latest version on the server to be used.

GSASIIpath.**svnUpdateProcess** (*version=None, projectfile=None*)

perform an update of GSAS-II in a separate python process

GSASIIpath.**svnUpgrade** (*fpath='/Users/toby/software/G2/GSASII'*)

This reformats subversion files, which may be needed if an upgrade of subversion is done.

**Parameters** **fpath** (*str*) – path to repository dictionary, defaults to directory where the current file is located

GSASIIpath.**svnVersion** ()

Get the version number of the current subversion executable

**Returns** a string with a version number such as “1.6.6” or None if subversion is not found.

GSASIIpath.**svnVersionNumber** ()

Get the version number of the current subversion executable

**Returns** a fractional version number such as 1.6 or None if subversion is not found.

GSASIIpath.**whichsvn** ()

Returns a path to the subversion exe file, if any is found. Searches the current path as well as subdirectory “svn” and “svn/bin” in the location of the GSASII source files.

**Returns** None if svn is not found or an absolute path to the subversion executable file.

### 3.3 GSASIIlog: Logging of “Actions”

Module to provide logging services, e.g. track and replay “actions” such as menu item, tree item, button press, value change and so on.

**GSASIIlog.ButtonBindingLookup** = {}  
 Lookup table for button objects

**class GSASIIlog.ButtonLogEntry** (*locationcode, label*)  
 Object to track button press

**GSASIIlog.G2logList** = [None]  
 Contains a list of logged actions; first item is ignored

**GSASIIlog.InvokeMenuCommand** (*id, G2frame, event*)  
 Called when a menu item is used to log the action as well as call the routine “bind”ed to that menu item

**class GSASIIlog.LogEntry**  
 Base class to define logging objects. These store information on events in a manner that can be pickled and saved – direct references to wx objects is not allowed.

Each object must define:

- **\_\_init\_\_**: stores the information needed to log & later recreate the action
- **\_\_str\_\_**: shows a nice ASCII string for each action
- **Replay**: recreates the action when the log is played

optional:

- **Repaint**: redisplay the current window

**GSASIIlog.LogInfo** = {'Tree': None, 'Logging': False, 'LastPaintAction': None}  
 Contains values that are needed in the module for past actions & object location

**GSASIIlog.LogOff** ()  
 Turn Off logging of actions

**GSASIIlog.LogOn** ()  
 Turn On logging of actions

**GSASIIlog.LogVarChange** (*result, key*)  
 Called when a variable is changed to log that action

**GSASIIlog.MakeButtonLog** (*locationcode, label*)  
 Create a ButtonLogEntry action log

**GSASIIlog.MakeTabLog** (*title, tabname*)  
 Create a TabLogEntry action log

**GSASIIlog.MakeTreeLog** (*textlist*)  
 Create a TreeLogEntry action log

**GSASIIlog.MenuBindingLookup** = {}  
 Lookup table for Menu buttons

**class GSASIIlog.MenuLogEntry** (*menulabellist*)  
 object that tracks when a menu command is executed

**Replay ()**

Perform a Menu item action when read from the log

`GSASIIlog.OnReplayPress (event)`

execute one or more commands when the replay button is pressed

`GSASIIlog.ReplayLog (event)`

replay the logged actions

`GSASIIlog.SaveMenuCommand (id, G2frame, handler)`

Creates a table of menu items and their pseudo-bindings

`GSASIIlog.ShowLogStatus ()`

Return the logging status

**class** `GSASIIlog.TabLogEntry (title, tabname)`

Object to track when tabs are pressed in the DataFrame window

**Repaint ()**

Used to redraw a window created in response to a Tab press

**Replay ()**

Perform a Tab press action when read from the log

**class** `GSASIIlog.TreeLogEntry (itemlist)`

Object to track when tree items are pressed in the main window

**Repaint ()**

Used to redraw a window created in response to a click on a data tree item

**Replay ()**

Perform a Tree press action when read from the log

**class** `GSASIIlog.VarLogEntry (treeRefs, indexRefs, value)`

object that tracks changes to a variable

**Replay ()**

Perform a Variable Change action, when read from the log

**class** `GSASIIlog.dictLogged (obj, treeRefs, indexRefs=[])`

A version of a dict object that tracks the source of the object back to the location on the G2 tree. If a list (tuple) or dict are pulled from inside this object the source information is appended to the provenance tracking lists.

tuples are converted to lists.

**class** `GSASIIlog.listLogged (obj, treeRefs, indexRefs=[])`

A version of a list object that tracks the source of the object back to the location on the G2 tree. If a list (tuple) or dict are pulled from inside this object the source information is appended to the provenance tracking lists.

tuples are converted to lists.

### 3.4 *config\_example.py*: Configuration options

This file contains optional configuration options for GSAS-II. The variables in this file can be copied to file `config.py`, which is imported if present. Access these variables using `GSASIIpath.GetConfigValue()`, which returns `None` if the variable is not set. Note that a `config.py` file need not be present, but if in use it will typically be found with the GSAS-II source directory (`GSASIIpath.Path2GSAS2`) or a directory for local GSAS-II modifications (`~/G2local/` or `/Documents and Settings/<User>/G2local/`).

When defining new config variables for GSAS-II, define them here with a default value: use None or a string for strings, or use integers or real values. Include a doc string after each variable is defined to explain what it does. Use names ending in `_location` or `_directory` for items that will contain directory names.

For example:

```
test_int = 0
test_float = 0.0
test_string = None (or)
test_string = 'value'
```

`config_example.Enable_logging = False`

Set to True to enable use of command logging (under development.)

`config_example.Help_mode = 'browser'`

Set to “internal” to use a Python-based web viewer to display help documentation and tutorials. If set to the default (“browser”) the default web browser is used.

`config_example.Import_directory = None`

Specifies a default location for finding exercise files used for Tutorials.

`config_example.Starting_directory = None`

Specifies a default location for starting GSAS-II

`config_example.Transpose = False`

Set to True to cause images to be Transposed when read (for code development)

`config_example.Tutorial_location = None`

Change this to place tutorials by in a different spot. If None, this defaults to `<user>/My Documents/G2tutorials` (on windows) or `<user>/G2tutorials`. If you want to use a different location, this can be set here. To install into the location where GSAS-II is installed, use this:

```
Tutorial_location = GSASIIpath.path2GSAS2
```

As another example, to use `~/G2tutorials` do this:

```
Tutorial_location = os.path.expanduser('~/.G2tutorials')
```

Note that `os.path` and `GSASIIpath` are imported inside `config.py`; other imports will require manual editing of the file.

`config_example.debug = False`

Set to True to turn on debugging mode. This enables use of IPython on exceptions and on calls to `GSASIIpath.IPyBreak()`. Calls to `GSASIIpath.pdbBreak()` will invoke `pdb` at that location.

If `debug` is False, calls to `GSASIIpath.IPyBreak()` and `GSASIIpath.pdbBreak()` are ignored.

`config_example.logging_debug = False`

Set to True to enable debug for logging (under development.)

`config_example.wxInspector = False`

If set to True, the `wxInspector` widget is displayed when GSAS-II is started.

### 3.5 GSASIIElem: functions for element types

`GSASIIElem.CheckElement (El)`

Check if element `El` is in the periodic table

**Parameters** `El` (*str*) – One or two letter element symbol, capitalization ignored

**Returns** True if the element is found

GSASIIElem.**ComptonFac** (*El, SQ*)  
compute Compton scattering factor

**Parameters**

- **El** – element dictionary
- **SQ** –  $(\sin\text{-theta}/\lambda)^2$

**Returns** compton scattering factor

GSASIIElem.**FPcalc** (*Orbs, KEv*)  
Compute real & imaginary resonant X-ray scattering factors

**Parameters**

- **Orbs** – list of orbital dictionaries as defined in GetXsectionCoeff
- **KEv** – x-ray energy in keV

**Returns** C: (f',f'',mu): real, imaginary parts of resonant scattering & atomic absorption coeff.

GSASIIElem.**FixValence** (*El*)  
Returns the element symbol, even when a valence is present

GSASIIElem.**GetAtomInfo** (*El*)  
reads element information from atmdata.py

GSASIIElem.**GetBLtable** (*General*)  
returns a dictionary of neutron scattering length data for atom types & isotopes found in General

**Parameters** **General** (*dict*) – dictionary of phase info.; includes AtomTypes & Isotopes

**Returns** BLtable, dictionary of scattering length data; key is atom type

GSASIIElem.**GetFFC5** (*ElSym*)  
Get 5 term form factor and Compton scattering data

**Parameters** **ElSym** – str(1-2 character element symbol with proper case);

**Return** **El** dictionary with 5 term form factor & compton coefficients

GSASIIElem.**GetFFtable** (*atomTypes*)  
returns a dictionary of form factor data for atom types found in atomTypes

**Parameters** **atomTypes** (*list*) – list of atom types

**Returns** FFtable, dictionary of form factor data; key is atom type

GSASIIElem.**GetFormFactorCoeff** (*El*)  
Read X-ray form factor coefficients from *atomdata.py* file

**Parameters** **El** (*str*) – element 1-2 character symbol, case irrelevant

**Returns** *FormFactors*: list of form factor dictionaries

Each X-ray form factor dictionary is:

- *Symbol*: 4 character element symbol with valence (e.g. 'NI+2')
- *Z*: atomic number
- *fa*: 4 A coefficients
- *fb*: 4 B coefficients
- *fc*: C coefficient

GSASIIElem.**GetMagFormFacCoeff** (*El*)

Read magnetic form factor data from atomdata.asc file

**Parameters** *El* – 2 character element symbol

**Returns** MagFormFactors: list of all magnetic form factors dictionaries for element *El*.

each dictionary contains:

- ‘Symbol’: Symbol
- ‘Z’: Z
- ‘mfa’: 4 MA coefficients
- ‘nfa’: 4 NA coefficients
- ‘mfb’: 4 MB coefficients
- ‘nfb’: 4 NB coefficients
- ‘mfc’: MC coefficient
- ‘nfc’: NC coefficient

GSASIIElem.**GetXsectionCoeff** (*El*)

Read atom orbital scattering cross sections for fprime calculations via Cromer-Lieberman algorithm

**Parameters** *El* – 2 character element symbol

**Returns** Orbs: list of orbitals each a dictionary with detailed orbital information used by FPcalc

each dictionary is:

- ‘OrbName’: Orbital name read from file
- ‘IfBe’ 0/2 depending on orbital
- ‘BindEn’: binding energy
- ‘BB’: BindEn/0.02721
- ‘XSectIP’: 5 cross section inflection points
- ‘ElEterm’: energy correction term
- ‘SEdge’: absorption edge for orbital
- ‘Nval’: 10/11 depending on IfBe
- ‘LEner’: 10/11 values of log(energy)
- ‘LXSect’: 10/11 values of log(cross section)

GSASIIElem.**ScatFac** (*El, SQ*)

compute value of form factor

**Parameters**

- *El* – element dictionary defined in GetFormFactorCoeff
- *SQ* – (sin-theta/lambda)\*\*2

**Returns** real part of form factor

GSASIIElem.**getBLvalues** (*BLtables, ifList=False*)

Needs a doc string

GSASIIElem.**getFFvalues** (*FFtables, SQ, ifList=False*)

Needs a doc string



### 3.6 GSASIIlattice: Unit cells

Perform lattice-related computations

Note that  $g$  is the reciprocal lattice tensor, and  $G$  is its inverse,  $G = g^{-1}$ , where

$$G = \begin{pmatrix} a^2 & ab \cos \gamma & ac \cos \beta \\ ab \cos \gamma & b^2 & bc \cos \alpha \\ ac \cos \beta & bc \cos \alpha & c^2 \end{pmatrix}$$

The “A tensor” terms are defined as  $A = (G_{11} \ G_{22} \ G_{33} \ 2G_{12} \ 2G_{13} \ 2G_{23})$  and  $A$  can be used in this fashion:  $d^* = \sqrt{A_1 h^2 + A_2 k^2 + A_3 l^2 + A_4 hk + A_5 hl + A_6 kl}$ , where  $d$  is the d-spacing, and  $d^*$  is the reciprocal lattice spacing,  $Q = 2\pi d^* = 2\pi/d$

`GSASIIlattice.A2Gmat` ( $A$ , *inverse=True*)

Fill real & reciprocal metric tensor (G) from A.

**Parameters**

- **A** – reciprocal metric tensor elements as [G11,G22,G33,2\*G12,2\*G13,2\*G23]
- **inverse** (*bool*) – if True return both G and g; else just G

**Returns** reciprocal (G) & real (g) metric tensors (list of two numpy 3x3 arrays)

`GSASIIlattice.A2cell` ( $A$ )

Compute unit cell constants from A

**Parameters** **A** – [G11,G22,G33,2\*G12,2\*G13,2\*G23] G - reciprocal metric tensor

**Returns** a,b,c,alpha, beta, gamma (degrees) - lattice parameters

`GSASIIlattice.A2invcell` ( $A$ )

Compute reciprocal unit cell constants from A returns tuple with a\*,b\*,c\*,alpha\*, beta\*, gamma\* (degrees)

`GSASIIlattice.CellAbsorption` (*ElList*, *Volume*)

Compute unit cell absorption

**Parameters**

- **ElList** (*dict*) – dictionary of element contents including mu and number of atoms be cell
- **Volume** (*float*) – unit cell volume

**Returns** mu-total/Volume

`GSASIIlattice.CellBlock` (*nCells*)

Generate block of unit cells  $n*n*n$  on a side; [0,0,0] centered,  $n = 2*nCells+1$  currently only works for nCells = 0 or 1 (not >1)

`GSASIIlattice.CentCheck` (*Cent*, *H*)

needs doc string

`GSASIIlattice.CosAngle` ( $U$ ,  $V$ ,  $G$ )

calculate cos of angle between U & V in generalized coordinates defined by metric tensor G

**Parameters**

- **U** – 3-vectors assume numpy arrays, can be multiple reflections as (N,3) array
- **V** – 3-vectors assume numpy arrays, only as (3) vector
- **G** – metric tensor for U & V defined space assume numpy array

**Returns**  $\cos(\phi)$

`GSASIIlattice.CosSinAngle` (*U*, *V*, *G*)

calculate sin & cos of angle between *U* & *V* in generalized coordinates defined by metric tensor *G*

**Parameters**

- **U** – 3-vectors assume numpy arrays
- **V** – 3-vectors assume numpy arrays
- **G** – metric tensor for *U* & *V* defined space assume numpy array

**Returns**  $\cos(\phi)$  &  $\sin(\phi)$

`GSASIIlattice.CrsAng` (*H*, *cell*, *SGData*)

needs doc string

`GSASIIlattice.Dsp2pos` (*Inst*, *dsp*)

convert d-spacing to powder pattern position (2-theta or TOF, musec)

`GSASIIlattice.Flnh` (*Start*, *SHCoef*, *phi*, *beta*, *SGData*)

needs doc string

`GSASIIlattice.GenHBravais` (*dmin*, *Bravais*, *A*)

Generate the positionally unique powder diffraction reflections

**Parameters**

- **dmin** – minimum d-spacing in *A*
- **Bravais** – lattice type (see `GetBraviasNum`). Bravais is one of:: 0 F cubic 1 I cubic 2 P cubic 3 R hexagonal (trigonal not rhombohedral) 4 P hexagonal 5 I tetragonal 6 P tetragonal 7 F orthorhombic 8 I orthorhombic 9 C orthorhombic 10 P orthorhombic 11 C monoclinic 12 P monoclinic 13 P triclinic
- **A** – reciprocal metric tensor elements as [G11,G22,G33,2\*G12,2\*G13,2\*G23]

**Returns** HKL unique d list of [h,k,l,d,-1] sorted with largest d first

`GSASIIlattice.GenHLaue` (*dmin*, *SGData*, *A*)

Generate the crystallographically unique powder diffraction reflections for a lattice and Bravais type

**Parameters**

- **dmin** – minimum d-spacing
- **SGData** – space group dictionary with at least
  - ‘*SGLaue*’: Laue group symbol: one of ‘-1’, ‘2/m’, ‘mmm’, ‘4/m’, ‘6/m’, ‘4/mmm’, ‘6/mmm’, ‘3m1’, ‘31m’, ‘3’, ‘3R’, ‘3mR’, ‘m3’, ‘m3m’
  - ‘*SGLatt*’: lattice centering: one of ‘P’, ‘A’, ‘B’, ‘C’, ‘I’, ‘F’
  - ‘*SGUniq*’: code for unique monoclinic axis one of ‘a’, ‘b’, ‘c’ (only if ‘*SGLaue*’ is ‘2/m’) otherwise an empty string
- **A** – reciprocal metric tensor elements as [G11,G22,G33,2\*G12,2\*G13,2\*G23]

**Returns** HKL = list of [h,k,l,d] sorted with largest d first and is unique part of reciprocal space ignoring anomalous dispersion

`GSASIIlattice.GenPFHKLs` (*nMax*, *SGData*, *A*)

Generate the unique pole figure reflections for a lattice and Bravais type. Min d-spacing=1.0Å & no more than *nMax* returned

**Parameters**

- **nMax** – maximum number of hkls returned
- **SGData** – space group dictionary with at least
  - ‘SGLaue’: Laue group symbol: one of ‘-1’, ‘2/m’, ‘mmm’, ‘4/m’, ‘6/m’, ‘4/mmm’, ‘6/mmm’, ‘3m1’, ‘31m’, ‘3’, ‘3R’, ‘3mR’, ‘m3’, ‘m3m’
  - ‘SGLatt’: lattice centering: one of ‘P’, ‘A’, ‘B’, ‘C’, ‘I’, ‘F’
  - ‘SGUniq’: code for unique monoclinic axis one of ‘a’, ‘b’, ‘c’ (only if ‘SGLaue’ is ‘2/m’) otherwise an empty string
- **A** – reciprocal metric tensor elements as [G11,G22,G33,2\*G12,2\*G13,2\*G23]

**Returns** HKL = list of ‘h k l’ strings sorted with largest d first; no duplicate zones

GSASIIlattice.**GenSHCoeff** (*SGLaue, SamSym, L, IfLMN=True*)  
needs doc string

GSASIIlattice.**GenSSHLaue** (*dmin, SGData, SSGData, Vec, maxH, A*)  
needs a doc string

GSASIIlattice.**GetBraviasNum** (*center, system*)  
Determine the Bravais lattice number, as used in GenHBravais

**Parameters**

- **center** – one of: ‘P’, ‘C’, ‘I’, ‘F’, ‘R’ (see SGLatt from GSASIIspc.SpcGroup)
- **system** – one of ‘cubic’, ‘hexagonal’, ‘tetragonal’, ‘orthorhombic’, ‘trigonal’ (for R) ‘monoclinic’, ‘triclinic’ (see SGSys from GSASIIspc.SpcGroup)

**Returns** a number between 0 and 13 or throws a ValueError exception if the combination of center, system is not found (i.e. non-standard)

GSASIIlattice.**GetKcl** (*L, N, SGLaue, phi, beta*)  
needs doc string

GSASIIlattice.**GetKclKsl** (*L, N, SGLaue, psi, phi, beta*)

**This is used for spherical harmonics description of preferred orientation;** cylindrical symmetry only (M=0) and no sample angle derivatives returned

GSASIIlattice.**GetKsl** (*L, M, SamSym, psi, gam*)  
needs doc string

GSASIIlattice.**Glnh** (*Start, SHCoef, psi, gam, SamSym*)  
needs doc string

GSASIIlattice.**Gmat2A** (*G*)  
Extract A from reciprocal metric tensor (G)

**Parameters** **G** – reciprocal maetric tensor (3x3 numpy array

**Returns** A = [G11,G22,G33,2\*G12,2\*G13,2\*G23]

GSASIIlattice.**Gmat2AB** (*G*)  
Computes orthogonalization matrix from reciprocal metric tensor G

**Returns**

tuple of two 3x3 numpy arrays (A,B)

- A for crystal to Cartesian transformations  $A*x = np.inner(A,x) = X$
- B (= inverse of A) for Cartesian to crystal transformation  $B*X = np.inner(B,X) = x$

GSASIIlattice.**Gmat2cell** (*g*)

Compute real/reciprocal lattice parameters from real/reciprocal metric tensor (*g/G*) The math works the same either way.

**Parameters** (or **G**) (*g*) – real (or reciprocal) metric tensor 3x3 array

**Returns** a,b,c,alpha, beta, gamma (degrees) (or a\*,b\*,c\*,alpha\*,beta\*,gamma\* degrees)

GSASIIlattice.**Hx2Rh** (*Hx*)

needs doc string

GSASIIlattice.**MaxIndex** (*dmin, A*)

needs doc string

GSASIIlattice.**OdfChk** (*SGLaue, L, M*)

needs doc string

GSASIIlattice.**Pos2dsp** (*Inst, pos*)

convert powder pattern position (2-theta or TOF, musec) to d-spacing

GSASIIlattice.**Rh2Hx** (*Rh*)

needs doc string

GSASIIlattice.**SamAng** (*Tth, Gangls, Sangl, IFCoup*)

Compute sample orientation angles vs laboratory coord. system

**Parameters**

- **Tth** – Signed theta
- **Gangls** – Sample goniometer angles phi,chi,omega,azimuth
- **Sangl** – Sample angle zeros om-0, chi-0, phi-0
- **IFCoup** – True if omega & 2-theta coupled in CW scan

**Returns** psi,gam: Sample odf angles dPSdA,dGMdA: Angle zero derivatives

GSASIIlattice.**SwapIndx** (*Axis, H*)

needs doc string

GSASIIlattice.**TOF2dsp** (*Inst, Pos*)

convert powder pattern TOF, musec to d-spacing by successive approximation Pos can be numpy array

GSASIIlattice.**U6toUij** (*U6*)

Fill matrix (*Uij*) from *U6* = [U11,U22,U33,U12,U13,U23] NB: there is a non numpy version in GSASIIspc: U2Uij

**Parameters** **U6** (*list*) – 6 terms of u11,u22,...

**Returns** *Uij* - numpy [3][3] array of *uij*

GSASIIlattice.**Uij2Ueqv** (*Uij, GS, Amat*)

returns 1/3 trace of diagonalized U matrix

GSASIIlattice.**Uij2betaij** (*Uij, G*)

Convert *Uij* to beta-ij tensors – stub for eventual completion

**Parameters**

- **Uij** – numpy array [Uij]
- **G** – reciprocal metric tensor

**Returns** beta-ij - numpy array [beta-ij]

`GSASIIlattice.UijtoU6 (U)`  
 Fill vector [U11,U22,U33,U12,U13,U23] from Uij NB: there is a non numpy version in GSASIIspc: Uij2U

`GSASIIlattice.calc_v (A)`  
 Compute the real lattice volume (V) from A

`GSASIIlattice.calc_rDsqr (H, A)`  
 needs doc string

`GSASIIlattice.calc_rDsqr2 (H, G)`  
 needs doc string

`GSASIIlattice.calc_rDsqrSS (H, A, vec)`  
 needs doc string

`GSASIIlattice.calc_rDsqrT (H, A, Z, tof, difC)`  
 needs doc string

`GSASIIlattice.calc_rDsqrTSS (H, A, vec, Z, tof, difC)`  
 needs doc string

`GSASIIlattice.calc_rDsqrZ (H, A, Z, tth, lam)`  
 needs doc string

`GSASIIlattice.calc_rDsqrZSS (H, A, vec, Z, tth, lam)`  
 needs doc string

`GSASIIlattice.calc_rV (A)`  
 Compute the reciprocal lattice volume (V\*) from A

`GSASIIlattice.calc_rVsqr (A)`  
 Compute the square of the reciprocal lattice volume (1/V\*\*2) from A'

`GSASIIlattice.cell2A (cell)`  
 Obtain A = [G11,G22,G33,2\*G12,2\*G13,2\*G23] from lattice parameters  
**Parameters** `cell` – [a,b,c,alpha,beta,gamma] (degrees)  
**Returns** G reciprocal metric tensor as 3x3 numpy array

`GSASIIlattice.cell2AB (cell)`  
 Computes orthogonalization matrix from unit cell constants  
**Parameters** `cell (tuple)` – a,b,c, alpha, beta, gamma (degrees)  
**Returns** tuple of two 3x3 numpy arrays (A,B) A for crystal to Cartesian transformations  $A*x = np.inner(A,x) = X$  B (= inverse of A) for Cartesian to crystal transformation  $B*X = np.inner(B,X) = x$

`GSASIIlattice.cell2GS (cell)`  
 returns Uij to betaij conversion matrix

`GSASIIlattice.cell2Gmat (cell)`  
 Compute real and reciprocal lattice metric tensor from unit cell constants  
**Parameters** `cell` – tuple with a,b,c,alpha, beta, gamma (degrees)  
**Returns** reciprocal (G) & real (g) metric tensors (list of two numpy 3x3 arrays)

`GSASIIlattice.combinations (items, n)`  
 take n distinct items, order matters

`GSASIIlattice.criticalEllipse (prob)`  
 Calculate critical values for probability ellipsoids from probability

GSASIIlattice.**fillGmat** (*cell*)

Compute lattice metric tensor from unit cell constants

**Parameters** *cell* – tuple with a,b,c,alpha, beta, gamma (degrees)

**Returns** 3x3 numpy array

GSASIIlattice.**getHKLmax** (*dmin, SGData, A*)

finds maximum allowed hkl for given A within dmin

GSASIIlattice.**getPeakPos** (*dataType, parmdict, dsp*)

convert d-spacing to powder pattern position (2-theta or TOF, musec)

GSASIIlattice.**invcell2Gmat** (*invcell*)

**Compute real and reciprocal lattice metric tensor from reciprocal** unit cell constants

**Parameters** *invcell* – [a\*,b\*,c\*,alpha\*, beta\*, gamma\*] (degrees)

**Returns** reciprocal (G) & real (g) metric tensors (list of two 3x3 arrays)

GSASIIlattice.**invpolfcal** (*ODFln, SGData, phi, beta*)

needs doc string

GSASIIlattice.**permutations** (*items*)

take all items, order matters

GSASIIlattice.**polfcal** (*ODFln, SamSym, psi, gam*)

Perform a pole figure computation. Note that the the number of gam values must either be 1 or must match psi.  
Updated for numpy 1.8.0

GSASIIlattice.**rotMat** (*angle, axis=0*)

Prepare rotation matrix for angle in degrees about axis(=0,1,2)

**Parameters**

- **angle** – angle in degrees
- **axis** – axis (0,1,2 = x,y,z) about which for the rotation

**Returns** rotation matrix - 3x3 numpy array

GSASIIlattice.**rotMat4** (*angle, axis=0*)

Prepare rotation matrix for angle in degrees about axis(=0,1,2) with scaling for OpenGL

**Parameters**

- **angle** – angle in degrees
- **axis** – axis (0,1,2 = x,y,z) about which for the rotation

**Returns** rotation matrix - 4x4 numpy array (last row/column for openGL scaling)

GSASIIlattice.**sec2HMS** (*sec*)

Convert time in sec to H:M:S string

**Parameters** *sec* – time in seconds

**Returns** H:M:S string (to nearest 100th second)

GSASIIlattice.**selections** (*items, n*)

take n (not necessarily distinct) items, order matters

GSASIIlattice.**selftestlist** = [<function test0 at 0x108faaaa0>, <function test1 at 0x108faab18>, <function test2 at 0x108faab18>]

Defines a list of self-tests

`GSASIIlattice.sortHKLd` (*HKLd*, *ifreverse*, *ifdup*, *ifSS=False*)  
needs doc string

**Parameters**

- **HKLd** – a list of [h,k,l,d,...];
- **ifreverse** – True for largest d first
- **ifdup** – True if duplicate d-spacings allowed

`GSASIIlattice.test1` ()  
test cell2A and A2Gmat

`GSASIIlattice.test2` ()  
test Gmat2A, A2cell, A2Gmat, Gmat2cell

`GSASIIlattice.test3` ()  
test invcell2Gmat

`GSASIIlattice.test4` ()  
test calc\_rVsq, calc\_rV, calc\_V

`GSASIIlattice.test5` ()  
test A2invcell

`GSASIIlattice.test6` ()  
test cell2AB

`GSASIIlattice.test7` ()  
test GetBraviasNum(...) and GenHBravais(...)

`GSASIIlattice.test8` ()  
test GenHLaue

`GSASIIlattice.test9` ()  
test GenHLaue

`GSASIIlattice.textureIndex` (*SHCoef*)  
needs doc string

`GSASIIlattice.uniqueCombinations` (*items*, *n*)  
take n distinct items, order is irrelevant

### 3.7 GSASIIspc: Space group module

Space group interpretation routines. Note that space group information is stored in a *Space Group (SGData)* object.

`GSASIIspc.AllOps` (*SGData*)

Returns a list of all operators for a space group, including those for centering and a center of symmetry

**Parameters** *SGData* – from `SpcGroup()`

**Returns**

(*SGTextList*, *offsetList*, *symOpList*, *G2oprList*) where

- *SGTextList*: a list of strings with formatted and normalized symmetry operators.
- *offsetList*: a tuple of (dx,dy,dz) offsets that relate the GSAS-II symmetry operation to the operator in *SGTextList* and *symOpList*. these dx (etc.) values are added to the GSAS-II generated positions to provide the positions that are generated by the normalized symmetry operators.

- `symOpList`: a list of tuples with the normalized symmetry operations as (M,T) values (see `SGOps` in the *Space Group object*)
- `G2oprList`: The GSAS-II operations for each symmetry operation as a tuple with (center,mult,opnum), where center is (0,0,0), (0.5,0,0), (0.5,0.5,0.5),...; where mult is 1 or -1 for the center of symmetry and opnum is the number for the symmetry operation, in `SGOps` (starting with 0).

`GSASIIspc.ApplyStringOps (A, SGData, X, Uij=[ ])`

Needs a doc string

`GSASIIspc.ElemPosition (SGData)`

Under development. Object here is to return a list of symmetry element types and locations suitable for say drawing them. So far I have the element type... getting all possible locations without lookup may be impossible!

`GSASIIspc.GenAtom (XYZ, SGData, All=False, Uij=[ ], Move=True)`

Generates the equivalent positions for a specified coordinate and space group

#### Parameters

- **XYZ** – an array, tuple or list containing 3 elements: x, y & z
- **SGData** – from `SpcGroup()`
- **All** – True return all equivalent positions including duplicates; False return only unique positions
- **Uij** – [U11,U22,U33,U12,U13,U23] or [] if no Uij
- **Move** – True move generated atom positions to be inside cell False do not move atoms

#### Returns

[[XYZEquiv],Idup,[UijEquiv]]

- [XYZEquiv] is list of equivalent positions (XYZ is first entry)
- Idup = [-][C]SS where SS is the symmetry operator number (1-24), C (if not 0,0,0)
- is centering operator number (1-4) and - is for inversion Cell = unit cell translations needed to put new positions inside cell [UijEquiv] - equivalent Uij; absent if no Uij given

`GSASIIspc.GenHKLf (HKL, SGData)`

Uses old GSAS Fortran routine genhkl.for

#### Parameters

- **HKL** – [h,k,l] must be integral values for genhkl.for to work
- **SGData** – space group data obtained from `SpcGroup`

#### Returns

iabsnt,mulp,Uniq,phi

- iabsnt = True if reflection is forbidden by symmetry
- mulp = reflection multiplicity including Friedel pairs
- Uniq = numpy array of equivalent hkl in descending order of h,k,l
- phi = phase offset for each equivalent h,k,l

`GSASIIspc.GetCSuinel (siteSym)`

returns Uij terms, multipliers, GUI flags & Uiso2Uij multipliers



GSASIIspc.**GetCSxinel** (*siteSym*)

Needs a doc string

GSASIIspc.**GetKNsym** (*key*)

Needs a doc string

GSASIIspc.**GetNXUPQsym** (*siteSym*)

The codes XUPQ are for lookup of symmetry constraints for position(X), thermal parm(U) & magnetic moments (P&Q-not used in GSAS-II)

GSASIIspc.**GetOprPtrName** (*key*)

Needs a doc string

GSASIIspc.**HStrainNames** (*SGData*)

Needs a doc string

GSASIIspc.**Latt2text** (*Latt*)

From lattice type ('P', 'A', etc.) returns ';' delimited cell centering vectors

GSASIIspc.**MT2text** (*Opr*)

From space group matrix/translation operator returns text version

GSASIIspc.**MoveToUnitCell** (*xyz*)

Translates a set of coordinates so that all values are  $\geq 0$  and  $< 1$

**Parameters** xyz – a list or numpy array of fractional coordinates

**Returns** XYZ - numpy array of new coordinates now 0 or greater and less than 1

GSASIIspc.**Muiso2Shk1** (*muiso*, *SGData*, *cell*)

this is to convert isotropic mustrain to generalized Shkls

GSASIIspc.**MustrainCoeff** (*HKL*, *SGData*)

Needs a doc string

GSASIIspc.**MustrainNames** (*SGData*)

Needs a doc string

GSASIIspc.**Opposite** (*XYZ*, *toler=0.0002*)

**Gives opposite corner, edge or face of unit cell for position within tolerance.** Result may be just outside the cell within tolerance

**Parameters**

- **XYZ** –  $0 \leq \text{np.array}[x,y,z] < 1$  as by MoveToUnitCell
- **toler** – unit cell fraction tolerance making opposite

**Returns** XYZ: dict of opposite positions; key=unit cell & always contains XYZ

GSASIIspc.**SGErrors** (*IErr*)

Interprets the error message code from SpcGroup. Used in SpaceGroup.

**Parameters** IErr – see SGEror in SpcGroup ()

**Returns** ErrString - a string with the error message or "Unknown error"

GSASIIspc.**SGPrint** (*SGData*)

Print the output of SpcGroup in a nicely formatted way. Used in SpaceGroup

**Parameters** SGData – from SpcGroup ()

**Returns** SGText - list of strings with the space group details SGTable - list of strings for each of the operations

GSASIIspc.**SGProd** (*OpA, OpB*)

**Form space group operator product. OpA & OpB are [M,V] pairs;** both must be of same dimension (3 or 4). Returns [M,V] pair

GSASIIspc.**SGPtGroup** (*SGData*)

Determine point group of the space group - done after space group symbol has been evaluated by SpcGroup. Only short symbols are allowed

**Parameters** **SGData** – from :func SpcGroup

**Returns** SSGPtGrp & SSGKI (only defaults for Mono & Ortho)

GSASIIspc.**SGpolar** (*SGData*)

Determine identity of polar axes if any

GSASIIspc.**SSGModCheck** (*Vec, modSymb*)

Checks modulation vector compatibility with supersymmetry space group symbol. Superspace group symbol takes precedence & the vector will be modified accordingly

GSASIIspc.**SSGPrint** (*SGData, SSGData*)

Print the output of SSpCGroup in a nicely formatted way. Used in SSpaceGroup

**Parameters**

- **SGData** – space group data structure as defined in SpcGroup above.
- **SSGData** – from SSpCGroup ()

**Returns** SSGText - list of strings with the superspace group details SSGTable - list of strings for each of the operations

GSASIIspc.**SSLatt2text** (*SSGCen*)

Lattice centering vectors to text

GSASIIspc.**SSMT2text** (*Opr*)

From superspace group matrix/translation operator returns text version

GSASIIspc.**SSpaceGroup** (*SGSymbol, SSymbol*)

Print the output of SSpCGroup in a nicely formatted way.

**Parameters**

- **SGSymbol** – space group symbol with spaces between axial fields.
- **SSymbol** – superspace group symbol extension (string).

**Returns** nothing

GSASIIspc.**SSpCGroup** (*SGData, SSymbol*)

Determines supersymmetry information from superspace group name; currently only for (3+1) superlattices

**Parameters**

- **SGData** – space group data structure as defined in SpcGroup above (see *SGData*).
- **SSymbol** – superspace group symbol extension (string) defining modulation direction & generator info.

**Returns**

(SSGError,SSGData)

- SSGError = 0 for no errors; >0 for errors (see SSGErrors below for details)
- SSGData - is a dict (see *Superspace Group object*) with entries:

- ‘SSpGrp’: superspace group symbol extension to space group symbol, accidental spaces removed
- ‘SSGCen’: 4D cell centering vectors [0,0,0] at least
- ‘SSGOps’: 4D symmetry operations as [M,T] so that  $M*x+T = x$

GSASIIspc. **SpaceGroup** (*SGSymbol*)

Print the output of SpcGroup in a nicely formatted way.

**Parameters** **SGSymbol** – space group symbol (string) with spaces between axial fields

**Returns** nothing

GSASIIspc. **SpcGroup** (*SGSymbol*)

Determines cell and symmetry information from a short H-M space group name

**Parameters** **SGSymbol** – space group symbol (string) with spaces between axial fields

**Returns**

(SGError,SGData)

- SGError = 0 for no errors; >0 for errors (see SGErrors below for details)
- SGData - is a dict (see *Space Group object*) with entries:
  - ‘SpGrp’: space group symbol, slightly cleaned up
  - ‘SGLaue’: one of ‘-1’, ‘2/m’, ‘mmm’, ‘4/m’, ‘4/mmm’, ‘3R’, ‘3mR’, ‘3’, ‘3m1’, ‘31m’, ‘6/m’, ‘6/mmm’, ‘m3’, ‘m3m’
  - ‘SGInv’: boolean; True if centrosymmetric, False if not
  - ‘SGLatt’: one of ‘P’, ‘A’, ‘B’, ‘C’, ‘I’, ‘F’, ‘R’
  - ‘SGUniq’: one of ‘a’, ‘b’, ‘c’ if monoclinic, ‘’ otherwise
  - ‘SGCen’: cell centering vectors [0,0,0] at least
  - ‘SGOps’: symmetry operations as [M,T] so that  $M*x+T = x$
  - ‘SGSys’: one of ‘triclinic’, ‘monoclinic’, ‘orthorhombic’, ‘tetragonal’, ‘rhombohedral’, ‘trigonal’, ‘hexagonal’, ‘cubic’
  - ‘SGPolax’: one of ‘’, ‘x’, ‘y’, ‘x y’, ‘z’, ‘x z’, ‘y z’, ‘xyz’, ‘111’ for arbitrary axes
  - ‘SGPtGrp’: **one of 32 point group symbols (with some permutations), which is filled by SGPtGroup, is external (KE) part of supersymmetry point group**
  - ‘SSGKI’: **default internal (KI) part of supersymmetry point group; modified** in supersymmetry stuff depending on chosen modulation vector for Mono & Ortho

GSASIIspc. **StandardizeSpcName** (*spcgroup*)

Accept a spacegroup name where spaces may have not been used in the names according to the GSAS convention (spaces between symmetry for each axis) and return the space group name as used in GSAS

GSASIIspc. **StringOpsProd** (*A, B, SGData*)

Find  $A*B$  where A & B are in strings ‘-’ + ‘100\*c+n’ + ‘+ijk’ where ‘-’ indicates inversion, c(>0) is the cell centering operator, n is operator number from SgOps and ijk are unit cell translations (each may be <0). Should return resultant string - C. SGData - dictionary using entries:

- ‘SGCen’: cell centering vectors [0,0,0] at least
- ‘SGOps’: symmetry operations as [M,T] so that  $M*x+T = x$

GSASIIspc.**SytSym** (*XYZ*, *SGData*)

Generates the number of equivalent positions and a site symmetry code for a specified coordinate and space group

**Parameters**

- **XYZ** – an array, tuple or list containing 3 elements: x, y & z
- **SGData** – from SpcGroup

**Returns** a two element tuple:

- The 1st element is a code for the site symmetry (see GetKNsym)
- The 2nd element is the site multiplicity

GSASIIspc.**selftestlist** = [**<function test0 at 0x108fa1320>**, **<function test1 at 0x108fa1398>**, **<function test2 at 0x108fa13c0>**]

Defines a list of self-tests

GSASIIspc.**spglist** = {'**P6/mmm**': ('**P 3**', '**P 3 1**', '**P 3 2**', '**P -3**', '**P 3 1 2**', '**P 3 2 1**', '**P 3 1 2**', '**P 3 1 2**', '**P 3 2 1 2**', '**P 3 2 1 2**']}

A dictionary of space groups as ordered and named in the pre-2002 International Tables Volume A, except that spaces are used following the GSAS convention to separate the different crystallographic directions. Note that the symmetry codes here will recognize many non-standard space group symbols with different settings. They are ordered by Laue group

GSASIIspc.**splitSSsym** (*SSymbol*)

Splits supersymmetry symbol into two lists of strings

GSASIIspc.**ssdict** = {'**F m 2 m**': ['(**0b0**)', '(**0b0**)0ss', '(**0b0**)ss0', '(**0b0**)s0s', '(**0b1**)', '(**0b1**)s0s', '(**0b1**)0ss', '(**1b0**)', '(**1b0**)s0s']}

A dictionary of superspace group symbols allowed for each entry in spglist (except cubics). Monoclinics are all b-unique setting.

GSASIIspc.**test0** ()

self-test #0: exercise MoveToUnitCell

GSASIIspc.**test1** ()

self-test #1: SpcGroup against previous results

GSASIIspc.**test2** ()

self-test #2: SpcGroup against cctbx (sgtbx) computations

GSASIIspc.**test3** ()

self-test #3: exercise SytSym (includes GetOprPtrName, GenAtom, GetKNsym) for selected space groups against info in IT Volume A

### **3.8 gltext: draw OpenGL text**

Routines that render text on OpenGL without use of GLUT.

Code written by Christian Brugger & Stefan Hacker and distributed under GNU General Public License.

**class** gltext.**Text** (*text='Text', font=None, font\_size=8, foreground=wx.Colour(), centered=False*)

A simple class for using System Fonts to display text in an OpenGL scene. The Text adds a global Cache of already created text elements to TextElement's base functionality so you can save some memory and increase speed

**centered**

Display the text centered

**draw\_text** (*position=wx.Point(0, 0), scale=1.0, rotation=0*)

position (wx.Point) - x/y Position to draw in scene scale (float) - Scale rotation (int) - Rotation in degree

Draws the text to the scene

**font**  
Font of the object

**font\_size**  
Font size

**foreground**  
Color/Overlay bitmap of the text

**getTextElement ()**  
Returns the text element bound to the Text class

**getTexture ()**  
Returns the texture of the bound TextElement

**getTexture\_size ()**  
Returns a texture size tuple

**setCentered (value, reinit=True)**  
value (bool) - New centered value reinit (bool) - Create a new texture  
Sets a new value for 'centered'

**setFont (value, reinit=True)**  
value (bool) - New Font reinit (bool) - Create a new texture  
Sets a new font

**setFont\_size (value, reinit=True)**  
value (bool) - New font size reinit (bool) - Create a new texture  
Sets a new font size

**setForeground (value, reinit=True)**  
value (bool) - New centered value reinit (bool) - Create a new texture  
Sets a new value for 'centered'

**setText (value, reinit=True)**  
value (bool) - New Text reinit (bool) - Create a new texture  
Sets a new text

**text**  
Text of the object

**text\_element**  
TextElement bound to this class

**texture**  
Texture of bound TextElement

**texture\_size**  
Size of the used texture

**class gltext.TextElement (text='', font=None, foreground=wx.Colour(), centered=False)**  
A simple class for using system Fonts to display text in an OpenGL scene

**bind ()**  
Increase refcount

**centered**  
Is text centered

**createTexture ()**

Creates a texture from the settings saved in TextElement, to be able to use normal system fonts conveniently a wx.MemoryDC is used to draw on a wx.Bitmap. As wxwidgets device contexts don't support alpha at all it is necessary to apply a little hack to preserve antialiasing without sticking to a fixed background color:

We draw the bmp in b/w mode so we can use its data as a alpha channel for a solid color bitmap which after GL\_ALPHA\_TEST and GL\_BLEND will show a nicely antialiased text on any surface.

To access the raw pixel data the bmp gets converted to a wx.Image. Now we just have to merge our foreground color with the alpha data we just created and push it all into a OpenGL texture and we are DONE *inhalesdelpy*

DRAWBACK of the whole conversion thing is a really long time for creating the texture. If you see any optimizations that could save time PLEASE CREATE A PATCH!!!

**deleteTexture ()**

Deletes the OpenGL texture object

**draw\_text** (*position=wx.Point(0, 0), scale=1.0, rotation=0*)

position (wx.Point) - x/y Position to draw in scene scale (float) - Scale rotation (int) - Rotation in degree

Draws the text to the scene

**font**

Font of the object

**foreground**

Color of the text

**isBound ()**

Return refcount

**owner\_cnt**

Owner count

**release ()**

Decrease refcount

**text**

Text of the object

**texture**

Used texture

**texture\_size**

Size of the used texture

### 3.9 *ElementTable: Periodic Table Data*

Element table data for building periodic table with valences & JMOL colors. Need these in case we go back to this periodic table coloring scheme.

Defines list ElTable which contains all defined oxidation states for each element, the location in the table, an element name, a color, a size and a second color.

### 3.10 *FormFactors: Scattering Data*

Contains atomic scattering factors from “New Analytical Scattering Factor Functions for Free Atoms and Ions for Free Atoms and Ions”, D. Waasmaier & A. Kirfel, *Acta Cryst.* (1995). A51, 416-413.

Also, tabulated coefficients for calculation of Compton Cross Section as a function of  $\sin(\theta)/\lambda$  from “Analytic Approximations to Incoherently Scattered X-Ray Intensities”, H. H. M. Balyuzi, *Acta Cryst.* (1975). A31, 600.

### 3.11 *ImageCalibrants: Calibration Standards*

GSASII powder calibrants as a dictionary `ImageCalibrants.Calibrants` with substances commonly used for powder calibrations for image data.

Each entry in `ImageCalibrants` consists of:

```
'key': ([Bravais num, ], [space group, ], [(a,b,c,alpha,beta,gamma), ], no. lines skipped, (dmin,pixLimit,cut))
(The space group may be an empty string)
```

as an example:

```
'LaB6 SRM660a': ([2, ], ['' ], [(4.1569162, 4.1569162, 4.1569162, 90, 90, 90), ], 0, (1.0, 10, 10)),
```

or where “Bravais num” and “(a,b,...)” are repeated in the case of mixtures:

```
'LaB6 & CeO2': ([2, 0], ['' , '' ] [(4.1569, 4.1569, 4.1569, 90, 90, 90), (5.4117, 5.4117, 5.4117, 90, 90, 90)], 0, (
```

To expand this list with locally needed additions, do not modify this file, because you may lose these changes during a software update. Instead duplicate the format of this file in a file named *UserCalibrants.py* and there define the material(s) you want:

```
Calibrants={
  'LaB6 skip 2 lines': ([2, ], ['' ], [(4.1569162, 4.1569162, 4.1569162, 90, 90, 90), ], 2, (1.0, 10, 10)),
}
```

New key values will be added to the list of options. If a key is duplicated, the information in *UserCalibrants.py* will override the information in this file.

Note, some useful Bravais numbers are: F-cubic=0, I-cubic=1, P-cubic=2, R3/m (hex)=3, P6=4, P4mmm=6

### 3.12 *atmdata: Table of atomic data*

The entries here are:

XrayFF: a dict of form factor coefficients

AtmSize: atom Sizes, bond radii, angle radii, H-bond radii

AtmBlens: atom masses & neutron scattering length (b,b'), sig(incoh) @ 1A

MagFF: neutron magnetic form factor coeff: M for  $j < 0$  & N for  $j < 2$

Sources:

Exponential scattering factor curve coefficients, Cromer and Waber(1971) Int. Tables Vol. IV. Delta f' and delta f'' terms, Cromer(1971) Int. Tables Vol. IV Atomic weights from CRC 56th Edition.

Neutron scattering lengths & abs. cross sections from H. Rauch & W. Waschowski, Neutron Data Booklet, 2003. X-ray  $\langle j_0 \rangle$  &  $\langle j_2 \rangle$  coeff. from Intl. Tables for Cryst, Vol. C

Neutron anomalous coeff (LS) from fitting Lynn & Seeger, At. Data & Nuc. Data Tables, 44, 191-207(1990)

O2- x-ray scattering factor from Tokonami (1965) Acta Cryst 19, 486

At wts from 14th ed Nuclides & Isotopes, 1989 GE Co.

### **3.13 *defaultparms: Table of instrument parameters***

Defines some default instrument parameters. Format for each is a list of strings finished with a ‘ ‘. Begin with ‘#GSAS-II...’ as the reader routine checks this. Each line can be comprised of a block of ‘;’ delimited name:value pairs. All instrument parameters must be included; even those = 0. Use a GSAS-II instprm file as a source for the entries.

For a new entry:

Append a useful name to defaultIparms\_lbl.

Append the list of lines to defaultIparms.

defaultIparm\_lbl: defines a list of labels.

defaultIparms: defines a list of multiple strings with values for each set of defaults.



## GSAS-II GUI ROUTINES

### 4.1 GSASIIctrls: Custom GUI controls

A library of GUI controls for reuse throughout GSAS-II

(at present many are still in GSASIIgrid, but with time will be moved here)

**class** GSASIIctrls.**ASCIIValidator** (*result=None, key=None*)

A validator to be used with a TextCtrl to prevent entering characters other than ASCII characters.

**The value is checked for validity after every keystroke** If an invalid number is entered, the box is highlighted. If the number is valid, it is saved in result[key]

#### Parameters

- **result** (*dict/list*) – List or dict where value should be placed when valid
- **key** (*any*) – key to use for result (int for list)

#### Clone ()

Create a copy of the validator, a strange, but required component

#### OnChar (*event*)

Called each type a key is pressed ignores keys that are not allowed for int and float types

#### TestValid (*tc*)

Check if the value is valid by casting the input string into ASCII.

Save it in the dict/list where the initial value was stored

**Parameters** *tc* (*wx.TextCtrl*) – A reference to the TextCtrl that the validator is associated with.

#### TransferFromWindow ()

Needed by validator, strange, but required component

#### TransferToWindow ()

Needed by validator, strange, but required component

**class** GSASIIctrls.**AddHelp** (*frame, helpType, helpLbl=None, title=''*)

For the Mac: creates an entry to the help menu of type 'Help on <helpType>': where helpType is a reference to an HTML page to be opened.

NOTE: when appending this menu (menu.Append) be sure to set the title to '&Help' so that wx handles it correctly.

#### OnHelpById (*event*)

Called when Help on... is pressed in a menu. Brings up a web page for documentation.

`GSASIIctrls.CallScrolledMultiEditor` (*parent*, *dictlst*, *elemlst*, *prelbl*=[], *postlbl*=[], *title*='Edit items', *header*='', *size*=(300, 250), *CopyButton*=False, *\*\*kw*)

Shell routine to call a ScrolledMultiEditor dialog. See `ScrolledMultiEditor` for parameter definitions.

**Returns** True if the OK button is pressed; False if the window is closed with the system menu or the Cancel button.

**class** `GSASIIctrls.EnumSelector` (*parent*, *dct*, *item*, *choices*, *values*=None, *\*\*kw*)

A customized `wxpython.ComboBox` that selects items from a list of choices, but sets a dict (list) entry to the corresponding entry from the input list of values.

#### Parameters

- **parent** (*wx.Panel*) – the parent to the `ComboBox` (usually a frame or panel)
- **dct** (*dict*) – a dict (or list) to contain the value set for the `ComboBox`.
- **item** – the dict key (or list index) where `dct[item]` will be set to the value selected in the `ComboBox`. Also, `dct[item]` contains the starting value shown in the widget. If the value does not match an entry in *values*, the first value in *choices* is used as the default, but `dct[item]` is not changed.

- **choices** (*list*) – a list of choices to be displayed to the user such as

```
["default", "option 1", "option 2",]
```

Note that these options will correspond to the entries in *values* (if specified) item by item.

- **values** (*list*) – a list of values that correspond to the options in *choices*, such as

```
[0, 1, 2]
```

The default for *values* is to use the same list as specified for *choices*.

- **(other)** – additional keyword arguments accepted by `ComboBox` can be specified.

**class** `GSASIIctrls.G2CheckBox` (*parent*, *label*, *loc*, *key*, *OnChange*=None)

A customized version of a `CheckBox` that automatically initializes the control to a supplied list or dict entry and updates that entry as the widget is used.

#### Parameters

- **parent** (*wx.Panel*) – name of panel or frame that will be the parent to the widget. Can be None.
- **label** (*str*) – text to put on check button
- **loc** (*dict/list*) – the dict or list with the initial value to be placed in the `CheckBox`.
- **key** (*int/str*) – the dict key or the list index for the value to be edited by the `CheckBox`. The `loc[key]` element must exist. The `CheckBox` will be initialized from this value. If the value is anything other than True (or 1), it will be taken as False.
- **OnChange** (*function*) – specifies a function or method that will be called when the `CheckBox` is changed (Default is None). The called function is supplied with one argument, the calling event.

**class** `GSASIIctrls.G2ChoiceButton` (*parent*, *choiceList*, *indLoc*=None, *indKey*=None, *strLoc*=None, *strKey*=None, *onChoice*=None, *\*\*kwargs*)

A customized version of a `wx.Choice` that automatically initializes the control to match a supplied value and saves the choice directly into an array or list. Optionally a function can be called each time a choice is selected. The widget can be used with an array item that is set to to the choice by number (`indLoc[indKey]`) or by

string value (`strLoc[strKey]`) or both. The initial value is taken from `indLoc[indKey]` if not `None` or `strLoc[strKey]` if not `None`.

#### Parameters

- **parent** (*wx.Panel*) – name of panel or frame that will be the parent to the widget. Can be `None`.
- **choiceList** (*list*) – a list or tuple of choices to offer the user.
- **indLoc** (*dict/list*) – a dict or list with the initial value to be placed in the Choice button. If this is `None`, this is ignored.
- **indKey** (*int/str*) – the dict key or the list index for the value to be edited by the Choice button. If `indLoc` is not `None` then this must be specified and the `indLoc[indKey]` will be set. If the value for `indLoc[indKey]` is not `None`, it should be an integer in `range(len(choiceList))`. The Choice button will be initialized to the choice corresponding to the value in this element if not `None`.
- **strLoc** (*dict/list*) – a dict or list with the string value corresponding to `indLoc/indKey`. Default (`None`) means that this is not used.
- **strKey** (*int/str*) – the dict key or the list index for the string value. The `strLoc[strKey]` element must exist or `strLoc` must be `None` (default).
- **onChoice** (*function*) – name of a function to call when the choice is made.

```
class GSASIIctrls.G2ColumnIDDialoG(parent, title, header, Comments, ChoiceList, ColumnData,
                                  monoFont=False, **kw)
```

A dialog for matching column data to desired items; some columns may be ignored.

#### Parameters

- **ParentFrame** (*wx.Frame*) – reference to parent frame
- **title** (*str*) – heading above list of choices
- **header** (*str*) – Title to place on window frame
- **ChoiceList** (*list*) – a list of possible choices for the columns
- **ColumnData** (*list*) – lists of column data to be matched with `ChoiceList`
- **monoFont** (*bool*) – If `False` (default), use a variable-spaced font; if `True` use a equally-spaced font.
- **kw** – optional keyword parameters for the `wx.Dialog` may be included such as `size` [which defaults to `(320,310)`] and `style` (which defaults to `wx.DEFAULT_DIALOG_STYLE | wx.RESIZE_BORDER | wx.CENTRE | wx.OK | wx.CANCEL`); note that `wx.OK` and `wx.CANCEL` controls the presence of the eponymous buttons in the dialog.

**Returns** the name of the created dialog

#### GetSelection()

Returns the selected sample parm for each column

```
class GSASIIctrls.G2HistoDataDialog(parent, title, header, ParmList, ParmFmt, HistoList, Parm-
                                   Data, monoFont=False, **kw)
```

A dialog for editing histogram data globally.

#### Parameters

- **ParentFrame** (*wx.Frame*) – reference to parent frame
- **title** (*str*) – heading above list of choices

- **header** (*str*) – Title to place on window frame
- **ParmList** (*list*) – a list of names for the columns
- **ParmFmt** (*list*) – a list of formatting strings for the columns
- **list** – HistoList: a list of histogram names
- **ParmData** (*list*) – a list of lists of data matched to ParmList; one for each item in HistoList
- **monoFont** (*bool*) – If False (default), use a variable-spaced font; if True use a equally-spaced font.
- **kw** – optional keyword parameters for the wx.Dialog may be included such as size [which defaults to (320,310)] and style (which defaults to wx.DEFAULT\_DIALOG\_STYLE | wx.RESIZE\_BORDER | wx.CENTRE | wx.OK | wx.CANCEL); note that wx.OK and wx.CANCEL controls the presence of the eponymous buttons in the dialog.

**Returns** the modified ParmData

**GetData** ()

Returns the modified ParmData

**class** GSASIIctrls.**G2HtmlWindow** (*parent, \*args, \*\*kwargs*)

Displays help information in a primitive HTML browser type window

**class** GSASIIctrls.**G2LoggedButton** (*parent, id=-1, label='', locationcode='', handler=None, \*args, \*\*kwargs*)

A version of wx.Button that creates logging events. Bindings are saved in the object, and are looked up rather than directly set with a bind. An index to these buttons is saved as log.ButtonBindingLookup :param wx.Panel parent: parent widget :param int id: Id for button :param str label: label for button :param str locationcode: a label used internally to uniquely identify the button :param function handler: a routine to call when the button is pressed

**onPress** (*event*)

create log event and call handler

GSASIIctrls.**G2MessageBox** (*parent, msg, title='Error'*)

Simple code to display a error or warning message

**class** GSASIIctrls.**G2MultiChoiceDialog** (*parent, title, header, ChoiceList, toggle=True, monoFont=False, filterBox=True, \*\*kw*)

A dialog similar to MultiChoiceDialog except that buttons are added to set all choices and to toggle all choices.

#### Parameters

- **ParentFrame** (*wx.Frame*) – reference to parent frame
- **title** (*str*) – heading above list of choices
- **header** (*str*) – Title to place on window frame
- **ChoiceList** (*list*) – a list of choices where one will be selected
- **toggle** (*bool*) – If True (default) the toggle and select all buttons are displayed
- **monoFont** (*bool*) – If False (default), use a variable-spaced font; if True use a equally-spaced font.
- **filterBox** (*bool*) – If True (default) an input widget is placed on the window and only entries matching the entered text are shown.
- **kw** – optional keyword parameters for the wx.Dialog may be included such as size [which defaults to (320,310)] and style (which defaults to wx.DEFAULT\_DIALOG\_STYLE|wx.RESIZE\_BORDER|wx.CENTRE| wx.OK |

`wx.CANCEL`); note that `wx.OK` and `wx.CANCEL` controls the presence of the eponymous buttons in the dialog.

**Returns** the name of the created dialog

**Filter** (*event*)

Read text from filter control and select entries that match. Called by Timer after a delay with no input or if Enter is pressed.

**GetSelections** ()

Returns a list of the indices for the selected choices

**OnCheck** (*event*)

for CheckListBox events; if Set Range is in use, this sets/clears all entries in range between start and end according to the value in start. Repeated clicks on the start change the checkbox state, but do not trigger the range copy. The caption next to the button is updated on the first button press.

**SetRange** (*event*)

Respond to a press of the Set Range button. Set the range flag and the caption next to the button

**SetSelections** (*selList*)

Sets the selection indices in selList as selected. Resets any previous selections for compatibility with `wx.MultiChoiceDialog`. Note that the state for only the filtered items is shown.

**Parameters** `selList` (*list*) – indices of items to be selected. These indices are referenced to the order in `self.ChoiceList`

**onChar** (*event*)

Respond to keyboard events in the Filter box

**class** `GSASIIctrls.G2SingleChoiceDialog` (*parent, title, header, ChoiceList, monoFont=False, filterBox=True, \*\*kw*)

A dialog similar to `wx.SingleChoiceDialog` except that a filter can be added.

**Parameters**

- **ParentFrame** (*wx.Frame*) – reference to parent frame
- **title** (*str*) – heading above list of choices
- **header** (*str*) – Title to place on window frame
- **ChoiceList** (*list*) – a list of choices where one will be selected
- **monoFont** (*bool*) – If False (default), use a variable-spaced font; if True use an equally-spaced font.
- **filterBox** (*bool*) – If True (default) an input widget is placed on the window and only entries matching the entered text are shown.
- **kw** – optional keyword parameters for the `wx.Dialog` may be included such as `size` [which defaults to `(320,310)`] and `style` (which defaults to `wx.DEFAULT_DIALOG_STYLE | wx.RESIZE_BORDER | wx.CENTRE | wx.OK | wx.CANCEL`); note that `wx.OK` and `wx.CANCEL` controls the presence of the eponymous buttons in the dialog.

**Returns** the name of the created dialog

**GetSelection** ()

Returns the index of the selected choice

**class** `GSASIIctrls.G2TreeCtrl` (*parent=None, \*args, \*\*kwargs*)

Create a wrapper around the standard `TreeCtrl` so we can “wrap” various events.

This logs when a tree item is selected (in `onSelectionChanged()`)

This also wraps lists and dicts pulled out of the tree to track where they were retrieved from.

**Bind** (*eventtype, handler, \*args, \*\*kwargs*)

Override the Bind() function so that page change events can be trapped

**ConvertRelativeHistNum** (*histtype, histnum*)

Converts a histogram type and relative histogram number to a histogram name in the current project

**ConvertRelativePhaseNum** (*phasenum*)

Converts relative phase number to a phase name in the current project

**GetRelativeHistNum** (*histname*)

Returns list with a histogram type and a relative number for that histogram, or the original string if not a histogram

**GetRelativePhaseNum** (*phasename*)

Returns a phase number if the string matches a phase name or else returns the original string

**onSelectionChanged** (*event*)

Log each press on a tree item here.

**class** GSASIIctrls.**GSGrid** (*parent, name=''*)

Basic wx.Grid implementation

**InstallGridToolTip** (*rowcolhintcallback, colLblCallback=None, rowLblCallback=None*)

code to display a tooltip for each item on a grid from <http://wiki.wxpython.org/wxGrid%20ToolTips> (buggy!), expanded to column and row labels using hints from <https://groups.google.com/forum/#!topic/wxPython-users/bm8OARRVDCs>

#### Parameters

- **rowcolhintcallback** (*function*) – a routine that returns a text string depending on the selected row and column, to be used in explaining grid entries.
- **colLblCallback** (*function*) – a routine that returns a text string depending on the selected column, to be used in explaining grid columns (if None, the default), column labels do not get a tooltip.
- **rowLblCallback** (*function*) – a routine that returns a text string depending on the selected row, to be used in explaining grid rows (if None, the default), row labels do not get a tooltip.

**class** GSASIIctrls.**GSNoteBook** (*parent, name='', size=None*)

Notebook used in various locations; implemented with wx.aui extension

**Bind** (*eventtype, handler, \*args, \*\*kwargs*)

Override the Bind() function so that page change events can be trapped

GSASIIctrls.**GetItemOrder** (*parent, keylist, vallookup, posdict*)

Creates a panel where items can be ordered into columns

#### Parameters

- **keylist** (*list*) – is a list of keys for column assignments
- **vallookup** (*dict*) – is a dict keyed by names in keylist where each item is a dict. Each inner dict contains variable names as keys and their associated values
- **posdict** (*dict*) – is a dict keyed by names in keylist where each item is a dict. Each inner dict contains column numbers as keys and their associated variable name as a value. This is used for both input and output.

**class** GSASIIctrls.**GridFractionEditor** (*grid*)

A grid cell editor class that allows entry of values as fractions as well as sine and cosine values [as s() and c()]

**ApplyEdit** (*row, col, grid*)

Called only in wx >= 2.9 Save the value of the control into the grid if EndEdit() returns as True

**class** GSASIIctrls.**HelpButton** (*parent, msg*)

Create a help button that displays help information. The text is displayed in a modal message window.

TODO: it might be nice if it were non-modal: e.g. it stays around until the parent is deleted or the user closes it, but this did not work for me.

#### Parameters

- **parent** – the panel which will be the parent of the button
- **msg** (*str*) – the help text to be displayed

GSASIIctrls.**HorizontalLine** (*sizer, parent*)

Draws a horizontal line as wide as the window. This shows up on the Mac as a very thin line, no matter what I do

GSASIIctrls.**ItemSelector** (*ChoiceList, ParentFrame=None, title='Select an item', size=None, header='Item Selector', useCancel=True, multiple=False*)

Provide a wx dialog to select a single item or multiple items from list of choices

#### Parameters

- **ChoiceList** (*list*) – a list of choices where one will be selected
- **ParentFrame** (*wx.Frame*) – Name of parent frame (default None)
- **title** (*str*) – heading above list of choices (default 'Select an item')
- **size** (*wx.Size*) – Size for dialog to be created (default None – size as needed)
- **header** (*str*) – Title to place on window frame (default 'Item Selector')
- **useCancel** (*bool*) – If True (default) both the OK and Cancel buttons are offered
- **multiple** (*bool*) – If True then multiple items can be selected (default False)

**Returns** the selection index or None or a selection list if multiple is true

**class** GSASIIctrls.**MultiStringDialog** (*parent, title, prompts, values=[]*)

Dialog to obtain a multi string values from user

#### Parameters

- **parent** (*wx.Frame*) – name of parent frame
- **title** (*str*) – title string for dialog
- **prompts** (*str*) – strings to tell use what they are inputting
- **values** (*str*) – default input values, if any

**GetValues** ()

Use this method to get the value entered by the user :returns: string entered by user

**Show** ()

Use this method after creating the dialog to post it :returns: True if the user pressed OK; False if the User pressed Cancel

**class** GSASIIctrls.**MyHelp** (*frame, helpType=None, helpLbl=None, morehelpitems=[], title=''*)

A class that creates the contents of a help menu. The menu will start with two entries:

- 'Help on <helpType>': where helpType is a reference to an HTML page to be opened
- About: opens an About dialog using OnHelpAbout. N.B. on the Mac this gets moved to the App menu to be consistent with Apple style.

NOTE: for this to work properly with respect to system menus, the title for the menu must be &Help, or it will not be processed properly:

```
menu.Append(menu=MyHelp(self, ...), title("&Help"))
```

**OnCheckUpdates** (*event*)

Check if the GSAS-II repository has an update for the current source files and perform that update if requested.

**OnHelpAbout** (*event*)

Display an 'About GSAS-II' box

**OnHelpById** (*event*)

Called when Help on... is pressed in a menu. Brings up a web page for documentation.

**OnSelectVersion** (*event*)

Allow the user to select a specific version of GSAS-II

**class** GSASIIctrls.**MyHtmlPanel** (*frame, id*)

Defines a panel to display HTML help information, as an alternative to displaying help information in a web browser.

**class** GSASIIctrls.**NumberValidator** (*typ, positiveonly=False, min=None, max=None, result=None, key=None, OKcontrol=None, CIFinput=False*)

A validator to be used with a TextCtrl to prevent entering characters other than digits, signs, and for float input, a period and exponents.

**The value is checked for validity after every keystroke** If an invalid number is entered, the box is highlighted. If the number is valid, it is saved in result[key]

#### Parameters

- **typ** (*type*) – the base data type. Must be int or float.
- **positiveonly** (*bool*) – If True, negative integers are not allowed (default False). This prevents the + or - keys from being pressed. Used with typ=int; ignored for typ=float.
- **min** (*number*) – Minimum allowed value. If None (default) the lower limit is unbounded
- **max** (*number*) – Maximum allowed value. If None (default) the upper limit is unbounded
- **result** (*dict/list*) – List or dict where value should be placed when valid
- **key** (*any*) – key to use for result (int for list)
- **OKcontrol** (*function*) – function or class method to control an OK button for a window. Ignored if None (default)
- **CIFinput** (*bool*) – allows use of a single '?' or '.' character as valid input.

**CheckInput** (*previousInvalid*)

called to test every change to the TextCtrl for validity and to change the appearance of the TextCtrl

Anytime the input is invalid, call self.OKcontrol (if defined) because it is fast. If valid, check for any other invalid entries only when changing from invalid to valid, since that is slower.

**Parameters previousInvalid** (*bool*) – True if the TextCtrl contents were invalid prior to the current change.

**Clone** ()

Create a copy of the validator, a strange, but required component

**OnChar** (*event*)

Called each type a key is pressed ignores keys that are not allowed for int and float types



**ShowValidity** (*tc*)

Set the control colors to show invalid input

**Parameters** *tc* (*wx.TextCtrl*) – A reference to the TextCtrl that the validator is associated with.

**TestValid** (*tc*)

Check if the value is valid by casting the input string into the current type.

Set the invalid variable in the TextCtrl object accordingly.

If the value is valid, save it in the dict/list where the initial value was stored, if appropriate.

**Parameters** *tc* (*wx.TextCtrl*) – A reference to the TextCtrl that the validator is associated with.

**TransferFromWindow** ()

Needed by validator, strange, but required component

**TransferToWindow** ()

Needed by validator, strange, but required component

**class** GSASIIctrls.**OpenTutorial** (*parent=None*)

Open a tutorial, optionally copying it to the local disk. Always copy the data files locally.

For now tutorials will always be copied into the source code tree, but it might be better to have an option to copy them somewhere else, for people who don't have write access to the GSAS-II source code location.

**LoadExercise** (*tutorialname, fullpath='/Users/toby/G2tutorials', baseURL='https://subversion.xray.aps.anl.gov/pyGSAS'*)

Load Exercise file(s) for a Tutorial to the selected location

**LoadTutorial** (*tutorialname, fullpath='/Users/toby/G2tutorials', baseURL='https://subversion.xray.aps.anl.gov/pyGSAS'*)

Load a Tutorial to the selected location

**OnModeSelect** (*event*)

Respond when the mode is changed

**OnTutorialSelected** (*event*)

Respond when a tutorial is selected. Load tutorials and data locally, as needed and then display the page

**SelectDownloadLoc** (*event*)

Select a download location, Cancel resets to the default

**ShowTutorialPath** ()

Show the help and exercise directory names

**ValidateTutorialDir** (*fullpath='/Users/toby/G2tutorials', baseURL='https://subversion.xray.aps.anl.gov/pyGSAS'*)

Load help to new directory or make sure existing directory looks correctly set up throws an exception if there is a problem.

**class** GSASIIctrls.**OrderBox** (*parent, keylist, vallookup, posdict, \*arg, \*\*kw*)

Creates a panel with scrollbars where items can be ordered into columns

**Parameters**

- **keylist** (*list*) – is a list of keys for column assignments
- **vallookup** (*dict*) – is a dict keyed by names in keylist where each item is a dict. Each inner dict contains variable names as keys and their associated values
- **posdict** (*dict*) – is a dict keyed by names in keylist where each item is a dict. Each inner dict contains column numbers as keys and their associated variable name as a value. This is used for both input and output.

**OnChoice** (*event*)

Called when a column is assigned to a variable

**class** GSASIIctrls.**PickTwoDialog** (*parent, title, prompt, names, choices*)

This does not seem to be in use

**class** GSASIIctrls.**ScrolledMultiEditor** (*parent, dictlst, elemlst, prelbl=[ ], postlbl=[ ], title='Edit items', header='', size=(300, 250), CopyButton=False, minvals=[ ], maxvals=[ ], sizevals=[ ], checkdictlst=[ ], checkelemlst=[ ], checklabel=''*)

Define a window for editing a potentially large number of dict- or list-contained values with validation for each item. Edited values are automatically placed in their source location. If invalid entries are provided, the TextCtrl is turned yellow and the OK button is disabled.

The type for each TextCtrl validation is determined by the initial value of the entry (int, float or string). Float values can be entered in the TextCtrl as numbers or also as algebraic expressions using operators + - / \* () and \*\*, in addition pi, sind(), cosd(), tand(), and sqrt() can be used, as well as abbreviations s(), sin(), c(), cos(), t(), tan() and sq().

### Parameters

- **parent** (*wx.Frame*) – name of parent window, or may be None
- **dictlst** (*tuple*) – a list of dicts or lists containing values to edit
- **elemlst** (*tuple*) – a list of keys for each item in a dictlst. Must have the same length as dictlst.
- **parent** – name of parent window, or may be None
- **prelbl** (*tuple*) – a list of labels placed before the TextCtrl for each item (optional)
- **postlbl** (*tuple*) – a list of labels placed after the TextCtrl for each item (optional)
- **title** (*str*) – a title to place in the frame of the dialog
- **header** (*str*) – text to place at the top of the window. May contain new line characters.
- **size** (*wx.Size*) – a size parameter that dictates the size for the scrolled region of the dialog. The default is (300,250).
- **CopyButton** (*bool*) – if True adds a small button that copies the value for the current row to all fields below (default is False)
- **minvals** (*list*) – optional list of minimum values for validation of float or int values. Ignored if value is None.
- **maxvals** (*list*) – optional list of maximum values for validation of float or int values. Ignored if value is None.
- **sizevals** (*list*) – optional list of wx.Size values for each input widget. Ignored if value is None.
- **checkdictlst** (*tuple*) – an optional list of dicts or lists containing bool values (similar to dictlst).
- **checkelemlst** (*tuple*) – an optional list of dicts or lists containing bool key values (similar to elemlst). Must be used with checkdictlst.
- **checklabel** (*string*) – a string to use for each checkbutton

**Returns** the wx.Dialog created here. Use method .ShowModal() to display it.

*Example for use of ScrolledMultiEditor:*

```
dlg = <pkg>.ScrolledMultiEditor(frame,dictlst,elemlst,prelbl,postlbl,
                               header=header)
if dlg.ShowModal() == wx.ID_OK:
```

```
for d,k in zip(dictlst,elemlst):
    print d[k]
```

*Example definitions for dictlst and elemlst:*

```
dictlst = (dict1,list1,dict1,list1)
elemlst = ('a', 1, 2, 3)
```

This causes items `dict1['a']`, `list1[1]`, `dict1[2]` and `list1[3]` to be edited.

Note that these items must have int, float or str values assigned to them. The dialog will force these types to be retained. String values that are blank are marked as invalid.

**ControlOKButton** (*setvalue*)

Enable or Disable the OK button for the dialog. Note that this is passed into the ValidatedTxtCtrl for use by validators.

**Parameters** *setvalue* (*bool*) – if True, all entries in the dialog are checked for validity. if False then the OK button is disabled.

**class** GSASIIctrls.**SelectConfigSetting** (*parent=None*)

Dialog to select configuration variables and set associated values.

**GetVarDocs** (*fname*)

Read the module referenced in *fname* (often `<module>.__file__`) and return a dict with names of global variables as keys. For each global variable, the value contains four items:

- item 0: the default value
- item 1: the current value
- item 2: the initial value (starts same as item 1)
- item 3: the “docstring” that follows variable definition

**OnApplyChanges** (*event=None*)

Set config variables to match the current settings

**OnBoolSelect** (*event*)

Respond to a change in a True/False variable

**OnChange** (*event=None*)

Check if anything been changed. Turn the save button on/off.

**OnSave** (*event*)

Write the config variables to `config.py` and then set them as the current settings

**OnSelection** ()

show a selected variable

**onSelDir** (*event*)

Select a directory from a menu

GSASIIctrls.**SelectEdit1Var** (*G2frame, array, labelLst, elemKeysLst, dspLst, refFlgElem*)

Select a variable from a list, then edit it and select histograms to copy it to.

**Parameters**

- **G2frame** (*wx.Frame*) – main GSAS-II frame
- **array** (*dict*) – the array (dict or list) where values to be edited are kept
- **labelLst** (*list*) – labels for each data item

- **elemKeysLst** (*list*) – a list of lists of keys needed to be applied (see below) to obtain the value of each parameter
- **dspLst** (*list*) – list list of digits to be displayed (10,4) is 10 digits with 4 decimal places. Can be None.
- **refFlgElem** (*list*) – a list of lists of keys needed to be applied (see below) to obtain the refine flag for each parameter or None if the parameter does not have refine flag.

**Example::**

```
array = data labelLst = ['v1','v2'] elemKeysLst = [['v1'], ['v2',0]] refFlgElem = [None, ['v2',1]]
```

- The value for v1 will be in data['v1'] and this cannot be refined while,
- The value for v2 will be in data['v2'][0] and its refinement flag is data['v2'][1]

`GSASIIctrls.ShowHelp` (*helpType, frame*)

Called to bring up a web page for documentation.

`GSASIIctrls.ShowWebPage` (*URL, frame*)

Called to show a tutorial web page.

**class** `GSASIIctrls.SingleFloatDialog` (*parent, title, prompt, value, limits=[0.0, 1.0], format='%.5g'*)

Dialog to obtain a single float value from user

**class** `GSASIIctrls.SingleStringDialog` (*parent, title, prompt, value='', size=(200, -1)*)

Dialog to obtain a single string value from user

**Parameters**

- **parent** (*wx.Frame*) – name of parent frame
- **title** (*str*) – title string for dialog
- **prompt** (*str*) – string to tell use what they are inputting
- **value** (*str*) – default input value, if any

**GetValue** ()

Use this method to get the value entered by the user :returns: string entered by user

**Show** ()

Use this method after creating the dialog to post it :returns: True if the user pressed OK; False if the User pressed Cancel

`GSASIIctrls.StripIndents` (*msg*)

Strip indentation from multiline strings

**class** `GSASIIctrls.Table` (*data=[ ], rowLabels=None, colLabels=None, types=None*)

Basic data table for use with GSgrid

**class** `GSASIIctrls.ValidatedTxtCtrl` (*parent, loc, key, nDig=None, notBlank=True, min=None, max=None, OKcontrol=None, OnLeave=None, typeHint=None, CIFinput=False, OnLeaveArgs={}, \*\*kw*)

Create a TextCtrl widget that uses a validator to prevent the entry of inappropriate characters and changes color to highlight when invalid input is supplied. As valid values are typed, they are placed into the dict or list where the initial value came from. The type of the initial value must be int, float or str or None (see `key` and `typeHint`); this type (or the one in `typeHint`) is preserved.

Float values can be entered in the TextCtrl as numbers or also as algebraic expressions using operators + - / \* () and \*\*, in addition pi, sind(), cosd(), tand(), and sqrt() can be used, as well as abbreviations s, sin, c, cos, t, tan and sq.

### Parameters

- **parent** (*wx.Panel*) – name of panel or frame that will be the parent to the TextCtrl. Can be None.
- **loc** (*dict/list*) – the dict or list with the initial value to be placed in the TextCtrl.
- **key** (*int/str*) – the dict key or the list index for the value to be edited by the TextCtrl. The `loc[key]` element must exist, but may have value None. If None, the type for the element is taken from `typeHint` and the value for the control is set initially blank (and thus invalid.) This is a way to specify a field without a default value: a user must set a valid value. If the value is not None, it must have a base type of int, float, str or unicode; the TextCtrl will be initialized from this value.
- **nDig** (*list*) – number of digits & places (`[nDig,nPlc]`) after decimal to use for display of float. Alternately, None can be specified which causes numbers to be displayed with approximately 5 significant figures (Default=None).
- **notBlank** (*bool*) – if True (default) blank values are invalid for str inputs.
- **min** (*number*) – minimum allowed valid value. If None (default) the lower limit is unbounded.
- **max** (*number*) – maximum allowed valid value. If None (default) the upper limit is unbounded
- **OKcontrol** (*function*) – specifies a function or method that will be called when the input is validated. The called function is supplied with one argument which is False if the TextCtrl contains an invalid value and True if the value is valid. Note that this function should check all values in the dialog when True, since other entries might be invalid. The default for this is None, which indicates no function should be called.
- **OnLeave** (*function*) – specifies a function or method that will be called when the focus for the control is lost. The called function is supplied with (at present) three keyword arguments:
  - **invalid**: (*bool*) True if the value for the TextCtrl is invalid
  - **value**: (*int/float/str*) the value contained in the TextCtrl
  - **tc**: (*wx.TextCtrl*) the TextCtrl name

The number of keyword arguments may be increased in the future should needs arise, so it is best to code these functions with a `**kwargs` argument so they will continue to run without errors

The default for OnLeave is None, which indicates no function should be called.

- **typeHint** (*type*) – the value of typeHint is overrides the initial value for the dict/list element `loc[key]`, if set to int or float, which specifies the type for input to the TextCtrl. Defaults as None, which is ignored.
- **CIFinput** (*bool*) – for str input, indicates that only printable ASCII characters may be entered into the TextCtrl. Forces output to be ASCII rather than Unicode. For float and int input, allows use of a single '?' or '.' character as valid input.
- **OnLeaveArgs** (*dict*) – a dict with keyword args that are passed to the OnLeave function. Defaults to `{ }`

- **(other)** – other optional keyword parameters for the wx.TextCtrl widget such as size or style may be specified.

**OnKeyDown** (*event*)

Special callback for wx 2.9+ on Mac where backspace is not processed by validator

**ShowStringValidity** (*previousInvalid=True*)

Check if input is valid. Anytime the input is invalid, call self.OKcontrol (if defined) because it is fast. If valid, check for any other invalid entries only when changing from invalid to valid, since that is slower.

**Parameters previousInvalid** (*bool*) – True if the TextCtrl contents were invalid prior to the current change.

**class** GSASIIctrls.**downdate** (*parent=None*)

Dialog to allow a user to select a version of GSAS-II to install

**getVersion** ()

Get the version number in the dialog

## 4.2 GSASIIgrid: Basic GUI routines

**class** GSASIIgrid.**AddAtomDialog** (*parent, Neigh, phase*)

H atom addition dialog. After ShowModal () returns, the results are found in dict self.data, which is accessed using GetData ().

**Parameters**

- **parent** (*wx.Frame*) – reference to parent frame (or None)
- **Neigh** (*dict*) – a dict of atom names with list of atom name, dist pairs for neighboring atoms
- **phase** (*dict*) – a dict containing the phase as defined by *Phase Tree Item*

**Draw** (*Neigh, phase*)

Creates the contents of the dialog. Normally called by \_\_init\_\_ ().

**GetData** ()

Returns the values from the dialog

**OnOk** (*event*)

Called when the OK button is pressed

**class** GSASIIgrid.**DataFrame** (*parent, frame, data=None, name=None, size=None, pos=None*)

Create the data item window and all the entries in menus used in that window. For Linux and windows, the menu entries are created for the current data item window, but in the Mac the menu is accessed from all windows. This means that a different menu is posted depending on which data item is posted. On the Mac, all the menus contain the data tree menu items, but additional menus are added specific to the data item.

Note that while the menus are created here, the binding for the menus is done later in various GSASII\*GUI modules, where the functions to be called are defined.

**Bind** (*eventtype, handler, \*args, \*\*kwargs*)

Override the Bind() function: on the Mac the binding is to the main window, so that menus operate with any window on top. For other platforms, either wrap calls that will be logged or call the default wx.Frame Bind() to bind to the menu item directly.

Note that bindings can be made to objects by Id or by direct reference to the object. As a convention, when bindings are to objects, they are not logged but when bindings are by Id, they are logged.

**PostfillDataMenu** (*empty=False*)

Create the “standard” part of data frame menus. Note that on Linux and Windows, this is the standard help Menu. On Mac, this menu duplicates the tree menu, but adds an extra help command for the data item and a separator.

**PrefillDataMenu** (*menu, helpType, helpLbl=None, empty=False*)

Create the “standard” part of data frame menus. Note that on Linux and Windows nothing happens here. On Mac, this menu duplicates the tree menu, but adds an extra help command for the data item and a separator.

**class** GSASIIgrid.**DisAglDialog** (*parent, data, default, Reset=True*)

Distance/Angle Controls input dialog. After `ShowModal()` returns, the results are found in `dict self.data`, which is accessed using `GetData()`.

#### Parameters

- **parent** (*wx.Frame*) – reference to parent frame (or None)
- **data** (*dict*) – a dict containing the current search ranges or an empty dict, which causes default values to be used. Will be used to set element *DisAglCtrls* in *Phase Tree Item*
- **default** (*dict*) – A dict containing the default search ranges for each element.

**Draw** (*data*)

Creates the contents of the dialog. Normally called by `__init__()`.

**GetData** ()

Returns the values from the dialog

**OnOk** (*event*)

Called when the OK button is pressed

**OnReset** (*event*)

Called when the Reset button is pressed

GSASIIgrid.**GetPatternTreeDataNames** (*G2frame, dataTypes*)

Needs a doc string

GSASIIgrid.**GetPatternTreeItemId** (*G2frame, parentId, itemText*)

Needs a doc string

GSASIIgrid.**HowDidIgetHere** ()

Show a traceback with calls that brought us to the current location. Used for debugging.

GSASIIgrid.**MovePatternTreeToGrid** (*G2frame, item*)

Called from `GSASII.OnPatternTreeSelChanged` when a item is selected on the tree

**class** GSASIIgrid.**SGMessageBox** (*parent, title, text, table*)

Special version of `MessageBox` that displays space group & super space group text in two blocks

**Show** ()

Use this method after creating the dialog to post it

GSASIIgrid.**SetDataMenuBar** (*G2frame, menu=None*)

Set the menu for the data frame. On the Mac put this menu for the data tree window instead.

Note that data frame items do not have menus, for these (`menu=None`) display a blank menu or on the Mac display the standard menu for the data tree window.

**class** GSASIIgrid.**ShowLSParms** (*parent, title, parmDict, varyList, fullVaryList, size=(300, 430)*)

Create frame to show least-squares parameters

**class** GSASIIgrid.**SymOpDialog** (*parent, SGData, New=True, ForceUnit=False*)

Class to select a symmetry operator

GSASIIgrid.**UpdateControls** (*G2frame, data*)

Edit overall GSAS-II controls in main Controls data tree entry

GSASIIgrid.**UpdateNotebook** (*G2frame, data*)

Called when the data tree notebook entry is selected. Allows for editing of the text in that tree entry

GSASIIgrid.**UpdatePWHKPlot** (*G2frame, kind, item*)

Called when the histogram main tree entry is called. Displays the histogram weight factor, refinement statistics for the histogram and the range of data for a simulation.

Also invokes a plot of the histogram.

GSASIIgrid.**UpdateSeqResults** (*G2frame, data, prevSize=None*)

Called when the Sequential Results data tree entry is selected to show results from a sequential refinement.

#### Parameters

- **G2frame** (*wx.Frame*) – main GSAS-II data tree windows
- **data** (*dict*) – a dictionary containing the following items:
  - ‘histNames’ - list of histogram names in order as processed by Sequential Refinement
  - ‘varyList’ - list of variables - identical over all refinements in sequence note that this is the original list of variables, prior to processing constraints.
  - ‘variableLabels’ – a dict of labels to be applied to each parameter (this is created as an empty dict if not present in data).
  - keyed by histName - dictionaries for all data sets processed, which contains:
    - \* ‘variables’ - result[0] from leastsq call
    - \* ‘varyList’ - list of variables passed to leastsq call (not same as above)
    - \* ‘sig’ - esds for variables
    - \* ‘covMatrix’ - covariance matrix from individual refinement
    - \* ‘title’ - histogram name; same as dict item name
    - \* ‘newAtomDict’ - new atom parameters after shifts applied
    - \* ‘newCellDict’ - refined cell parameters after shifts to A0-A5 from Dij terms applied

### 4.3 GSASIIIO: Misc I/O routines

Module with miscellaneous routines for input and output. Many are GUI routines to interact with user.

Includes support for image reading.

Also includes base classes for data import routines.

GSASIIIO.**CheckImageFile** (*G2frame, imagefile*)

Try to locate an image file if the project and image have been moved together. If the image file cannot be found, request the location from the user.

#### Parameters

- **G2frame** (*wx.Frame*) – main GSAS-II Frame and data object
- **imagefile** (*str*) – name of image file

**Returns** imagefile, if it exists, or the name of a file that does exist or False if the user presses Cancel



**class** GSASIIIO.**ExportBaseclass** (*G2frame, formatName, extension, longFormatName=None*)

Defines a base class for the exporting of GSAS-II results.

This class is subclassed in the various exports/G2export\_\*.py files. Those files are imported in GSASII.GSASII.\_\_init\_\_Exports() which defines the appropriate menu items for each one and the .Exporter method is called directly from the menu item.

Routines may also define a .Writer method, which is used to write a single file without invoking any GUI objects.

**CloseFile** (*fp=None*)

Close a file opened in OpenFile

**Parameters** **fp** (*file*) – the file object to be closed. If None (default) file object self.fp is closed.

**ExportSelect** (*AskFile='ask'*)

Selects histograms or phases when needed. Sets a default file name when requested in self.filename; always sets a default directory in self.dirname.

**Parameters** **AskFile** (*bool*) – Determines how this routine processes getting a location to store the current export(s).

- if AskFile is 'ask' (default option), get the name of the file to be written; self.filename and self.dirname are always set. In the case where multiple files must be generated, the export routine should do this based on self.filename as a template.
- if AskFile is 'dir', get the name of the directory to be used; self.filename is not used, but self.dirname is always set. The export routine will always generate the file name.
- if AskFile is 'single', get only the name of the directory to be used when multiple items will be written (as multiple files) are used *or* a complete file name is requested when a single file name is selected. self.dirname is always set and self.filename used only when a single file is selected.
- if AskFile is 'default', creates a name of the file to be used from the name of the project (.gpx) file. If the project has not been saved, then the name of file is requested. self.filename and self.dirname are always set. In the case where multiple file names must be generated, the export routine should do this based on self.filename.
- if AskFile is 'default-dir', sets self.dirname from the project (.gpx) file. If the project has not been saved, then a directory is requested. self.filename is not used.

**Returns** True in case of an error

**GetAtoms** (*phasenam*)

Gets the atoms associated with a phase. Can be used with standard or macromolecular phases

**Parameters** **phasenam** (*str*) – the name for the selected phase

**Returns**

a list of items for eac atom where each item is a list containing: label, typ, mult, xyz, and td, where

- label and typ are the atom label and the scattering factor type (str)
- mult is the site multiplicity (int)
- xyz is contains a list with four pairs of numbers: x, y, z and fractional occupancy and their standard uncertainty (or a negative value)
- td is contains a list with either one or six pairs of numbers: if one number it is  $U_{iso}$  and with six numbers it is  $U_{11}$ ,  $U_{22}$ ,  $U_{33}$ ,  $U_{12}$ ,  $U_{13}$  &  $U_{23}$  paired with their standard uncertainty (or a negative value)

**GetCell** (*phasenam*)

Gets the unit cell parameters and their s.u.'s for a selected phase

**Parameters** *phasenam* (*str*) – the name for the selected phase

**Returns** *cellList, cellSig* where each is a 7 element list corresponding to a, b, c, alpha, beta, gamma, volume where *cellList* has the cell values and *cellSig* has their uncertainties.

**InitExport** (*event*)

Determines the type of menu that called the Exporter and misc initialization.

**MakePWDRfilename** (*hist*)

Make a filename root (no extension) from a PWDR histogram name

**Parameters** *hist* (*str*) – the histogram name in data tree (starts with “PWDR ”)

**OpenFile** (*fil=None, mode='w'*)

Open the output file

**Parameters** *fil* (*str*) – The name of the file to open. If None (default) the name defaults to `self.dirname + self.filename`. If an extension is supplied, it is not overridden, but if not, the default extension is used.

**Returns** the file object opened by the routine which is also saved as `self.fp`

**Write** (*line*)

write a line of output, attaching a line-end character

**Parameters** *line* (*str*) – the text to be written.

**askSaveDirectory** ()

Ask the user to supply a directory name. Path name is used as the starting point for the next export path search.

**Returns** a directory name (*str*) or None if Cancel is pressed

**askSaveFile** ()

Ask the user to supply a file name

**Returns** a file name (*str*) or None if Cancel is pressed

**dumpTree** (*mode='type'*)

Print out information on the data tree dicts loaded in `loadTree`

**loadParmDict** ()

Load the GSAS-II refinable parameters from the tree into a dict (`self.parmDict`). Update refined values to those from the last cycle and set the uncertainties for the refined parameters in another dict (`self.sigDict`).

Expands the parm & sig dicts to include values derived from constraints.

**loadTree** ()

Load the contents of the data tree into a set of dicts (`self.OverallParms`, `self.Phases` and `self.Histogram` as well as `self.powderDict` & `self.xtalDict`)

- The childrenless data tree items are overall parameters/controls for the entire project and are placed in `self.OverallParms`
- Phase items are placed in `self.Phases`
- Data items are placed in `self.Histogram`. The key for these data items begin with a keyword, such as PWDR, IMG, HKLF,... that identifies the data type.

**GSASIIIO.ExportPowder** (*G2frame, TreeName, fileroot, extension*)

Writes a single powder histogram using the Export routines

**Parameters**

- **G2frame** (*wx.Frame*) – the GSAS-II main data tree window
- **TreeName** (*str*) – the name of the histogram (PWDR ...) in the data tree
- **fileroot** (*str*) – name for file to be written, extension ignored
- **extension** (*str*) – extension for file to be written (start with '.'). Must match a powder export routine that has a Writer object.

GSASIIIO.**ExportPowderList** (*G2frame*)

Returns a list of extensions supported by GSASIIIO:ExportPowder ()

**Parameters** **G2frame** (*wx.Frame*) – the GSAS-II main data tree window

GSASIIIO.**ExtractFileFromZip** (*filename, selection=None, confirmread=True, confirmoverwrite=True, parent=None, multipleselect=False*)

If the filename is a zip file, extract a file from that archive.

**Parameters**

- **Selection** (*list*) – used to predefine the name of the file to be extracted. Filename case and zip directory name are ignored in selection; the first matching file is used.
- **confirmread** (*bool*) – if True asks the user to confirm before expanding the only file in a zip
- **confirmoverwrite** (*bool*) – if True asks the user to confirm before overwriting if the extracted file already exists
- **multipleselect** (*bool*) – if True allows more than one zip file to be extracted, a list of file(s) is returned. If only one file is present, do not ask which one, otherwise offer a list of choices (unless selection is used).

**Returns** the name of the file that has been created or a list of files (see multipleselect)

If the file is not a zipfile, return the name of the input file. If the zipfile is empty or no file has been selected, return None

GSASIIIO.**FileDlgFixExt** (*dlg, file*)

this is needed to fix a problem in linux wx.FileDialog

GSASIIIO.**GetEdfData** (*filename, imageOnly=False*)

Read European detector data edf file

GSASIIIO.**GetG2Image** (*filename*)

Read an image as a python pickle

GSASIIIO.**GetGESumData** (*filename, imageOnly=False*)

Read SUM file as produced at 1-ID from G.E. images

GSASIIIO.**GetImageData** (*G2frame, imagefile, imageOnly=False*)

Read an image with the file reader keyed by the file extension

**Parameters**

- **G2frame** (*wx.Frame*) – main GSAS-II Frame and data object.
- **imagefile** (*str*) – name of image file
- **imageOnly** (*bool*) – If True return only the image, otherwise (default) return more (see below)

**Returns** an image as a numpy array or a list of four items: Comments, Data, Npix and the Image, as selected by imageOnly

GSASIIIO.**GetImgData** (*filename, imageOnly=False*)

Read an ADSC image file

GSASIIIO.**GetMAR345Data** (*filename, imageOnly=False*)

Read a MAR-345 image plate image

GSASIIIO.**GetPNGData** (*filename, imageOnly=False*)

Read an image in a png format, assumes image is converted from CheMin tif file so default parameters are that machine.

GSASIIIO.**GetPowderPeaks** (*fileName*)

Read powder peaks from a file

GSASIIIO.**GetRigaku** (*filename, imageOnly=False*)

Read Rigaku R-Axis IV image file

GSASIIIO.**GetTifData** (*filename, imageOnly=False*)

Read an image in a pseudo-tif format, as produced by a wide variety of software, almost always incorrectly in some way.

**class** GSASIIIO.**ImportBaseclass** (*formatName, longFormatName=None, extensionlist=[ ], strictExtension=False*)

Defines a base class for the reading of input files (diffraction data, coordinates,...). See *Writing a Import Routine* for an explanation on how to use a subclass of this class.

**BlockSelector** (*ChoiceList, ParentFrame=None, title='Select a block', size=None, header='Block Selector', useCancel=True*)

Provide a wx dialog to select a block if the file contains more than one set of data and one must be selected

**CIFValidator** (*filepointer*)

A `ContentsValidator()` for use to validate CIF files.

**ContentsValidator** (*filepointer*)

This routine will attempt to determine if the file can be read with the current format. This will typically be overridden with a method that takes a quick scan of [some of] the file contents to do a “sanity” check if the file appears to match the selected format. Expected to be called via `self.Validator()`

**ExtensionValidator** (*filename*)

This methods checks if the file has the correct extension Return False if this filename will not be supported by this reader Return True if the extension matches the list supplied by the reader Return None if the reader allows un-registered extensions

**exception ImportError**

Defines an Exception that is used when an import routine hits an expected error, usually in `.Reader`.

Good practice is that the Reader should define a value in `self.errors` that tells the user some information about what is wrong with their file.

`ImportBaseclass`.**MultipleBlockSelector** (*ChoiceList, ParentFrame=None, title='Select a block', size=None, header='Block Selector'*)

Provide a wx dialog to select a block of data if the file contains more than one set of data and one must be selected.

**Returns** a list of the selected blocks

`ImportBaseclass`.**MultipleChoicesDialog** (*choicelist, headinglist, ParentFrame=None, \*\*kwargs*)

A modal dialog that offers a series of choices, each with a title and a wx.Choice widget. Typical input:

- choicelist=[ ('a','b','c'), ('test1','test2'),('no choice',)]
- headinglist = [ 'select a, b or c', 'select 1 of 2', 'No option here']

optional keyword parameters are: `head` (window title) and `title` returns a list of selected indices for each choice (or `None`)

`ImportBaseclass.ReInitialize()`  
 Reinitialize the Reader to initial settings

**class** `GSASIIIO.ImportImage` (*formatName*, *longFormatName=None*, *extensionlist=[]*, *strictExtension=False*)

Defines a base class for the reading of images

Structure factors are read with a call to `GSASII.GSASII.OnImportImage()` which in turn calls `GSASII.GSASII.OnImportGeneric()`, which calls methods `ExtensionValidator()`, `ContentsValidator()` and `Reader()`.

See *Writing a Import Routine* for an explanation on how to use import classes in general. The specifics for reading an image requires that the `Reader()` routine in the import class need to do only a few things...

(should load `RefDict` item 'RefList' with the reflection list, (and set `Parameters` with the instrument parameters (initialized with `InitParameters()` and set with `UpdateParameters()`).

**InitParameters()**  
 initialize the instrument parameters structure

**ReInitialize()**  
 Reinitialize the Reader to initial settings – not used at present

**class** `GSASIIIO.ImportPhase` (*formatName*, *longFormatName=None*, *extensionlist=[]*, *strictExtension=False*)

Defines a base class for the reading of files with coordinates

Objects constructed that subclass this (in `import/G2phase_*.py` etc.) will be used in `GSASII.GSASII.OnImportPhase()`. See *Writing a Import Routine* for an explanation on how to use this class.

**PhaseSelector** (*ChoiceList*, *ParentFrame=None*, *title='Select a phase'*, *size=None*, *header='Phase Selector'*)  
 Provide a wx dialog to select a phase if the file contains more than one phase

**class** `GSASIIIO.ImportPowderData` (*formatName*, *longFormatName=None*, *extensionlist=[]*, *strictExtension=False*)

Defines a base class for the reading of files with powder data.

Objects constructed that subclass this (in `import/G2pwd_*.py` etc.) will be used in `GSASII.GSASII.OnImportPowder()`. See *Writing a Import Routine* for an explanation on how to use this class.

**ReInitialize()**  
 Reinitialize the Reader to initial settings

**class** `GSASIIIO.ImportSmallAngleData` (*formatName*, *longFormatName=None*, *extensionlist=[]*, *strictExtension=False*)

Defines a base class for the reading of files with small angle data. See *Writing a Import Routine* for an explanation on how to use this class.

**ReInitialize()**  
 Reinitialize the Reader to initial settings

**class** `GSASIIIO.ImportStructFactor` (*formatName*, *longFormatName=None*, *extensionlist=[]*, *strictExtension=False*)

Defines a base class for the reading of files with tables of structure factors.

Structure factors are read with a call to `GSASII.GSASII.OnImportSfact()` which in turn calls `GSASII.GSASII.OnImportGeneric()`, which calls methods `ExtensionValidator()`, `ContentsValidator()` and `Reader()`.

See *Writing a Import Routine* for an explanation on how to use import classes in general. The specifics for reading a structure factor histogram require that the `Reader()` routine in the import class need to do only a few things: It should load `RefDict` item 'RefList' with the reflection list, and set `Parameters` with the instrument parameters (initialized with `InitParameters()` and set with `UpdateParameters()`).

**Banks = None**

`self.RefDict` is a dict containing the reflection information, as read from the file. Item 'RefList' contains the reflection information. See the *Single Crystal Reflection Data Structure* for the contents of each row. Dict element 'FF' contains the form factor values for each element type; if this entry is left as initialized (an empty list) it will be initialized as needed later.

**InitParameters ()**

initialize the instrument parameters structure

**Parameters = None**

`self.Parameters` is a list with two dicts for data parameter settings

**ReInitialize ()**

Reinitialize the Reader to initial settings

**UpdateParameters (Type=None, Wave=None)**

Revise the instrument parameters

**GSASIIIO.IndexPeakListSave (G2frame, peaks)**

Save powder peaks from the indexing list

**GSASIIIO.LoadImage (imagefile, G2frame, Comments, Data, Npix, Image)**

Load an image into the tree

**class GSASIIIO.MultipleChoicesDialog (choicelist, headinglist, head='Select options', title='Please select from options below', parent=None)**

A dialog that offers a series of choices, each with a title and a `wx.Choice` widget. Intended to be used Modally. typical input:

- choicelist=[ ('a','b','c'), ('test1','test2'),('no choice',)]
- headinglist = [ 'select a, b or c', 'select 1 of 2', 'No option here']

selections are placed in `self.chosen` when OK is pressed

**GSASIIIO.PDFSave (G2frame, exports)**

Save a PDF G(r) and S(Q) in column formats

**GSASIIIO.PeakListSave (G2frame, file, peaks)**

Save powder peaks to a data file

**GSASIIIO.ProjFileOpen (G2frame)**

Read a GSAS-II project file and load into the G2 data tree

**GSASIIIO.ProjFileSave (G2frame)**

Save a GSAS-II project file

**GSASIIIO.PutG2Image (filename, Comments, Data, Npix, image)**

Write an image as a python pickle - might be better as an .edf file?

**GSASIIIO.ReadCIF (URLorFile)**

Open a CIF, which may be specified as a file name or as a URL using `PyCifRW` (from James Hester). The open routine gets confused with DOS names that begin with a letter and colon "C:dir" so this routine will try to open the passed name as a file and if that fails, try it as a URL

**Parameters URLorFile (str)** – string containing a URL or a file name. Code will try first to open it as a file and then as a URL.

**Returns** a `PyCifRW` CIF object.

GSASIIIO.**ReadImageData** (*G2frame, imagefile, imageOnly=False*)  
 Read a single image with an image importer. replacement for GetImageData

**Parameters**

- **G2frame** (*wx.Frame*) – main GSAS-II Frame and data object.
- **imagefile** (*str*) – name of image file
- **imageOnly** (*bool*) – If True return only the image, otherwise (default) return more (see below)

**Returns** an image as a numpy array or a list of four items: Comments, Data, Npix and the Image, as selected by imageOnly

GSASIIIO.**ReadLoadImage** (*imagefile, G2frame*)  
 Read a GSAS-II image file and load it into the data tree

GSASIIIO.**SaveIntegration** (*G2frame, PickId, data*)  
 Save image integration results as powder pattern(s)

GSASIIIO.**SetNewPhase** (*Name='New Phase', SGData=None, cell=None, Super=None*)  
 Create a new phase dict with default values for various parameters

**Parameters**

- **Name** (*str*) – Name for new Phase
- **SGData** (*dict*) – space group data from GSASIIspc:SpCGroup(); defaults to data for P 1
- **cell** (*list*) – unit cell parameter list; defaults to [1.0,1.0,1.0,90.,90,90.,1.]

GSASIIIO.**sfloat** (*S*)  
 Convert a string to float. An empty field or a unconvertable value is treated as zero

GSASIIIO.**sint** (*S*)  
 Convert a string to int. An empty field is treated as zero

GSASIIIO.**trim** (*val*)  
 Simplify a string containing leading and trailing spaces as well as newlines, tabs, repeated spaces etc. into a shorter and more simple string, by replacing all ranges of whitespace characters with a single space.

**Parameters** *val* (*str*) – the string to be simplified

**Returns** the (usually) shortened version of the string

## 4.4 ReadMarCCDFrame: Read Mar Files

**class** ReadMarCCDFrame.**marFrame** (*File, byteOrd='<', IFD={}*)  
 A class to extract correct mar header and image info from a MarCCD file

**Parameters**

- **File** (*str*) – file object [from open()]
- **byteOrd** – '<' (default) or '>'
- **IFD** (*dict*) – ?

## 4.5 GSASIIpy3: Python 3.x Routines

Module to hold python 3-compatible code, to keep it separate from code that will break with `__future__` options.

`GSASIIpy3.FormatPadValue` (*val*, *maxdigits=None*)

Format a float to fit in `maxdigits[0]` spaces with `maxdigits[1]` after decimal.

### Parameters

- **val** (*float*) – number to be formatted.
- **maxdigits** (*list*) – the number of digits & places after decimal to be used for display of the number (defaults to [10,2]).

**Returns** a string with exactly `maxdigits[0]` characters (except under error conditions), but last character will always be a space

`GSASIIpy3.FormatSigFigs` (*val*, *maxdigits=10*, *sigfigs=5*, *treatAsZero=1e-20*)

Format a float to use `maxdigits` or fewer digits with `sigfigs` significant digits showing (if room allows).

### Parameters

- **val** (*float*) – number to be formatted.
- **maxdigits** (*int*) – the number of digits to be used for display of the number (defaults to 10).
- **sigfigs** (*int*) – the number of significant figures to use, if room allows
- **treatAsZero** (*float*) – numbers that are less than this in magnitude are treated as zero. Defaults to 1.0e-20, but this can be disabled if set to None.

**Returns** a string with `<= maxdigits` characters (I hope).

`GSASIIpy3.FormatValue` (*val*, *maxdigits=None*)

Format a float to fit in at most `maxdigits[0]` spaces with `maxdigits[1]` after decimal. Note that this code has been hacked from `FormatSigFigs` and may have unused sections.

### Parameters

- **val** (*float*) – number to be formatted.
- **maxdigits** (*list*) – the number of digits & places after decimal to be used for display of the number (defaults to [10,2]).

**Returns** a string with `<= maxdigits` characters (usually).

`GSASIIpy3.FormulaEval` (*string*)

Evaluates a algebraic formula into a float, if possible. Works properly on fractions e.g. `2/3` only with python 3.0+ division.

Expressions such as `2/3`, `3*pi`, `sin(45)/2`, `2*sqrt(2)`, `2**10` can all be evaluated.

**Parameters** **string** (*str*) – Character string containing a Python expression to be evaluated.

**Returns** the value for the expression as a float or None if the expression does not evaluate to a valid number.



## GSAS-II GUI SUBMODULES

### 5.1 GSASIIphsGUI: Phase GUI

Module to create the GUI for display of phase information in the data display window when a phase is selected. Phase information is stored in one or more *Phase Tree Item* objects. Note that there are functions that respond to some tabs in the phase GUI in other modules (such as GSASIIddata).

GSASIIphsGUI.**FindBondsDraw** (*data*)  
uses numpy & masks - very fast even for proteins!

GSASIIphsGUI.**UpdatePhaseData** (*G2frame, Item, data, oldPage*)  
Create the data display window contents when a phase is clicked on in the main (data tree) window. Called only from `GSASIIgrid.MovePatternTreeToGrid()`, which in turn is called from `GSASII.GSASII.OnPatternTreeSelChanged()` when a tree item is selected.

#### Parameters

- **G2frame** (*wx.frame*) – the main GSAS-II frame object
- **Item** (*wx.TreeItemId*) – the tree item that was selected
- **data** (*dict*) – all the information on the phase in a dictionary
- **oldPage** (*int*) – This sets a tab to select when moving from one phase to another, in which case the same tab is selected to display first. This is set only when the previous data tree selection is a phase, if not the value is None. The default action is to bring up the General tab.

### 5.2 GSASIIddataGUI: Phase Diffraction Data GUI

Module to create the GUI for display of diffraction data \* phase information that is shown in the data display window (when a phase is selected.)

GSASIIddataGUI.**UpdateDData** (*G2frame, DData, data, hist=''*)  
Display the Diffraction Data associated with a phase (items where there is a value for each histogram and phase)

#### Parameters

- **G2frame** (*wx.frame*) – the main GSAS-II frame object
- **DData** (*wx.ScrolledWindow*) – notebook page to be used for the display
- **data** (*dict*) – all the information on the phase in a dictionary

### 5.3 GSASIIElemGUI: GUI to select and delete element lists

Module to select elements from a periodic table and to delete an element from a list of selected elements.

**class** GSASIIElemGUI.**DeleteElement** (*parent, choice*)  
Delete element from selected set widget

**ElButton** (*id, name, pos*)  
Needs a doc string

**class** GSASIIElemGUI.**PickElement** (*parent, oneOnly=False, ifNone=False*)  
Makes periodic table widget for picking element - caller maintains element list

**ElButton** (*name, pos, tip, color*)  
Creates an element button widget

**class** GSASIIElemGUI.**PickElements** (*parent, list*)  
Makes periodic table widget for picking elements - caller maintains element list

### 5.4 GSASIIconstrGUI: Constraint GUI routines

Used to define constraints and rigid bodies.

**class** GSASIIconstrGUI.**ConstraintDialog** (*parent, title, text, data, separator='\*', varname='', varyflag=False*)  
Window to edit Constraint values

**class** GSASIIconstrGUI.**MultiIntegerDialog** (*parent, title, prompts, values*)  
Input a series of integers based on prompts

GSASIIconstrGUI.**UpdateConstraints** (*G2frame, data*)  
Called when Constraints tree item is selected. Displays the constraints in the data window

GSASIIconstrGUI.**UpdateRigidBodies** (*G2frame, data*)  
Called when Rigid bodies tree item is selected. Displays the rigid bodies in the data window

### 5.5 GSASIIimgGUI: Image GUI

Control image display and processing

GSASIIimgGUI.**CleanupMasks** (*data*)  
If a mask creation is not completed, an empty mask entry is created in the masks array. This cleans them out. It is called when the masks page is first loaded and before saving them or after reading them in. This should also probably be done before they are used for integration.

GSASIIimgGUI.**UpdateImageControls** (*G2frame, data, masks*)  
Shows and handles the controls on the "Image Controls" data tree entry

GSASIIimgGUI.**UpdateMasks** (*G2frame, data*)  
Shows and handles the controls on the "Masks" data tree entry

GSASIIimgGUI.**UpdateStressStrain** (*G2frame, data*)  
Shows and handles the controls on the "Stress/Strain" data tree entry

## 5.6 GSASIIpwdGUI: Powder Pattern GUI routines

Used to define GUI controls for the routines that interact with the powder histogram (PWDR) data tree items.

GSASIIpwdGUI.**CopyPlotCtrls** (*G2frame*)

Global copy: Copy plot controls from current histogram to others.

GSASIIpwdGUI.**CopySelectedHistItems** (*G2frame*)

Global copy: Copy items from current histogram to others.

GSASIIpwdGUI.**GetHistsLikeSelected** (*G2frame*)

Get the histograms that match the current selected one: The histogram prefix and data type (PXC etc.), the number of wavelengths and the instrument geometry (Debye-Scherrer etc.) must all match. The current histogram is not included in the list.

**Parameters** *G2frame* (*wx.Frame*) – pointer to main GSAS-II data tree

GSASIIpwdGUI.**IsHistogramInAnyPhase** (*G2frame*, *histoName*)

Needs a doc string

GSASIIpwdGUI.**SetCopyNames** (*histName*, *dataType*, *addNames*=[*]*)

Determine the items in the sample parameters that should be copied, depending on the histogram type and the instrument type.

GSASIIpwdGUI.**SetDefaultSASDModel** ()

Fills in default items for the SASD Models dictionary

GSASIIpwdGUI.**SetDefaultSample** ()

Fills in default items for the Sample dictionary

GSASIIpwdGUI.**SetDefaultSubstances** ()

Fills in default items for the SASD Substances dictionary

GSASIIpwdGUI.**SetupSampleLabels** (*histName*, *dataType*, *histType*)

Setup a list of labels and number formatting for use in labeling sample parameters. :param str histName: Name of histogram, ("PWDR ...") :param str dataType:

GSASIIpwdGUI.**UpdateBackground** (*G2frame*, *data*)

respond to selection of PWDR background data tree item.

GSASIIpwdGUI.**UpdateIndexPeaksGrid** (*G2frame*, *data*)

respond to selection of PWDR Index Peak List data tree item.

GSASIIpwdGUI.**UpdateInstrumentGrid** (*G2frame*, *data*)

respond to selection of PWDR/SASD Instrument Parameters data tree item.

GSASIIpwdGUI.**UpdateLimitsGrid** (*G2frame*, *data*, *plottype*)

respond to selection of PWDR Limits data tree item.

GSASIIpwdGUI.**UpdateModelsGrid** (*G2frame*, *data*)

respond to selection of SASD Models data tree item.

GSASIIpwdGUI.**UpdatePDFGrid** (*G2frame*, *data*)

respond to selection of PWDR PDF data tree item.

GSASIIpwdGUI.**UpdatePeakGrid** (*G2frame*, *data*)

respond to selection of PWDR powder peaks data tree item.

GSASIIpwdGUI.**UpdateReflectionGrid** (*G2frame*, *data*, *HKLF=False*, *Name=''*)

respond to selection of PWDR Reflections data tree item by displaying a table of reflections in the data window.

GSASIIpwdGUI.**UpdateSampleGrid** (*G2frame*, *data*)

respond to selection of PWDR/SASD Sample Parameters data tree item.

`GSASIIpwdGUI.UpdateSubstanceGrid (G2frame, data)`  
 respond to selection of SASD Substance data tree item.

`GSASIIpwdGUI.UpdateUnitCellsGrid (G2frame, data)`  
 respond to selection of PWDR Unit Cells data tree item.

## 5.7 GSASIIrestrGUI: Restraint GUI routines

Used to define restraints.

`GSASIIrestrGUI.UpdateRestraints (G2frame, data, Phases, phaseName)`  
 Respond to selection of the Restraints item on the data tree

## 5.8 GSASIIexprGUI: Expression Handling

This module defines a class for defining an expression in terms of values in a parameter dictionary via a `wx.Dialog`. The dialog creates a `GSASII.ExpressionObj` which is used to evaluate the expression against a supplied parameter dictionary.

The expression is parsed to find variables used in the expression and then the user is asked to assign parameters from the dictionary to each variable.

Default expressions are read from file `DefaultExpressions.txt` using `GSASIIpath.LoadConfigFile()`.

```
class GSASIIexprGUI.ExpressionDialog (parent, parmDict, exprObj=None, header='Enter restraint
expression here', wintitle='Expression Editor', fit=True,
VarLabel=None, depVarDict=None, ExtraButton=None,
usedVars=[])
```

A `wx.Dialog` that allows a user to input an arbitrary expression to be evaluated and possibly minimized.

To do this, the user assigns a new (free) or existing GSAS-II parameter to each parameter label used in the expression. The free parameters can optionally be designated to be refined. For example, is an expression is used such as:

```
'A*np.exp(-B/C)'
```

then A, B and C can each be assigned as Free parameter with a user-selected value or to any existing GSAS-II variable in the parameter dictionary. As the expression is entered it is checked for validity.

After the `ExpressionDialog` object is created, use `Show()` to run it and obtain a `GSASIIobj.ExpressionObj` object with the user input.

### Parameters

- **parent** (*wx.Frame*) – The parent of the Dialog. Can be None, but better is to provide the name of the Frame where the dialog is called.
- **parmDict** (*dict*) – a dict with defined parameters and their values. Each value may be a list with parameter values and a refine flag or may just contain the parameter value (non-float/int values in dict are ignored)
- **exprObj** (*str*) – a `GSASIIobj.ExpressionObj` object with an expression and label assignments or None (default)
- **wintitle** (*str*) – String placed on title bar of dialog; defaults to “Expression Editor”
- **header** (*str*) – String placed at top of dialog to tell the user what they will do here; default is “Enter restraint expression here”

- **fit** (*bool*) – determines if the expression will be used in fitting (default=True). If set to False, and refinement flags are not shown and Free parameters are not offered as an assignment option.
- **VarLabel** (*str*) – an optional variable label to include before the expression input. Ignored if None (default)
- **depVarDict** (*list*) – a dict of choices for the dependent variable to be fitted to the expression and their values. Ignored if None (default).
- **ExtraButton** (*list*) – a list with two terms that define [0]: the label for an extra button and [1] the callback routine to be used when the button is pressed. The button will only be enabled when the OK button can be used (meaning the equation/expression is valid). The default is None, meaning this will not be used.
- **usedVars** (*list*) – defines a list of previously used variable names. These names will not be reused as defaults for new free variables. (The default is an empty list).

**CheckVars** ()

Check that appropriate variables are defined for each symbol used in `self.expr`

**Returns** a text error message or None if all needed input is present

**GetDepVar** ()

Returns the name of the dependent variable, when `depVarDict` is used.

**OnChar** (*event*)

Called as each character is entered. Cancels any running timer and starts a new one. The timer causes a check of syntax after 2 seconds without input. Disables the OK button until a validity check is complete.

**OnChoice** (*event*)

Respond to a selection of a variable type for a label in an expression

**OnDepChoice** (*event*)

Respond to a selection of a variable type for a label in an expression

**OnValidate** (*event*)

Respond to a press of the Validate button or when a variable is associated with a label (in `OnChoice()`)

**Repaint** (*exprObj*)

Redisplay the variables and continue the validation

**RestartTimer** ()

Cancels any running timer and starts a new one. The timer causes a check of syntax after 2 seconds unless there is further input. Disables the OK button until a validity check is complete.

**SelectG2var** (*sel, var, parmList*)

Offer a selection of a GSAS-II variable.

**Parameters** **sel** (*int*) – Determines the type of variable to be selected. where 1 is used for Phase variables, and 2 for Histogram/Phase vars, 3 for Histogram vars and 4 for Global vars.

**Returns** a variable name or None (if Cancel is pressed)

**Show** (*mode=True*)

Call to use the dialog after it is created.

**Returns** None (On Cancel) or a new `ExpressionObj`

**defSize = None**

Starting size for dialog

**depVarDict = None**

dict for dependent variables

**dependentVar = None**

name for dependent variable selection, when depVarDict is specified

**expr = None**

Expression as a text string

**exprVarLst = None**

A list containing the variables utilized in the current expression. Placed into a `GSASIIobj.ExpressionObj` object when the dialog is closed with “OK”, saving any changes.

**parmDict = None**

A copy of the G2 parameter dict (parmDict) except only numerical values are included and only the value (not the vary flag, if present) is included.

**setEvalResult (msg)**

Show a string in the expression result area

**showError (msg1, msg2=' ', msg3='')**

Show an error message of 1 to 3 sections. The second section is shown in an equally-spaced font.

**Parameters**

- **msg1 (str)** – msg1 is shown in a the standard font
- **msg2 (str)** – msg2 is shown in a equally-spaced (`wx.MODERN`) font
- **msg3 (str)** – msg3 is shown in a the standard font

**usedVars = None**

variable names that have been used and should not be reused by default

**varName = None**

Name assigned to each variable

**varRefflag = None**

Refinement flag for a variable (Free parameters only)

**varSelect = None**

A dict that shows the variable type for each label found in the expression.

- If the value is None or is not defined, the value is not assigned.
- If the value is 0, then the variable is “free” – a new refineable parameter.
- Values above 1 determine what variables will be shown when the option is selected.

**varValue = None**

Value for a variable (Free parameters only)

`GSASIIexprGUI.IndexParmDict (parmDict, wildcard)`

Separate the parameters in parmDict into list of keys by parameter type.

**Parameters**

- **parmDict (dict)** – a dict with GSAS-II parameters
- **wildcard (bool)** – True if wildcard versions of parameters should be generated and added to the lists

**Returns** a dict of lists where key 1 is a list of phase parameters, 2 is histogram/phase parms, 3 is histogram parms and 4 are global parameters

`GSASIIexprGUI.LoadDefaultExpressions ()`

Read a configuration file with default expressions from all files named DefaultExpressions.txt found in the path. Duplicates are removed and expressions are sorted alphabetically

---

## GSAS-II STRUCTURE SUBMODULES

### 6.1 GSASIIstrMain: main structure routine

GSASIIstrMain.**BestPlane** (*PlaneData*)

Needs a doc string

GSASIIstrMain.**DisAglTor** (*DATData*)

Needs a doc string

GSASIIstrMain.**PrintDistAngle** (*DisAglCtrls*, *DisAglData*, *out=<open file '<stdout>', mode 'w' at 0x10028e150>*)

Print distances and angles

#### Parameters

- **DisAglCtrls** (*dict*) – contains distance/angle radii usually defined using `GSASIIgrid.DisAglDialog()`
- **DisAglData** (*dict*) – contains phase data: Items ‘OrigAtoms’ and ‘TargAtoms’ contain the atoms to be used for distance/angle origins and atoms to be used as targets. Item ‘SGData’ has the space group information (see *Space Group object*)
- **out** (*file*) – file object for output. Defaults to `sys.stdout`.

GSASIIstrMain.**Refine** (*GPXfile*, *dlg*)

Global refinement – refines to minimize against all histograms

GSASIIstrMain.**RefineCore** (*Controls*, *Histograms*, *Phases*, *restraintDict*, *rigidbodyDict*, *parmDict*, *varyList*, *calcControls*, *pawleyLookup*, *ifPrint*, *printFile*, *dlg*)

Core optimization routines, shared between SeqRefine and Refine

GSASIIstrMain.**RetDistAngle** (*DisAglCtrls*, *DisAglData*)

Compute and return distances and angles

#### Parameters

- **DisAglCtrls** (*dict*) – contains distance/angle radii usually defined using `GSASIIgrid.DisAglDialog()`
- **DisAglData** (*dict*) – contains phase data: Items ‘OrigAtoms’ and ‘TargAtoms’ contain the atoms to be used for distance/angle origins and atoms to be used as targets. Item ‘SGData’ has the space group information (see *Space Group object*)

#### Returns

AtomLabels,DistArray,AngArray where:

**AtomLabels** is a dict of atom labels, keys are the atom number

**DistArray** is a dict keyed by the origin atom number where the value is a list of distance entries. The value for each distance is a list containing:

0. the target atom number (int);
1. the unit cell offsets added to x,y & z (tuple of int values)
2. the symmetry operator number (which may be modified to indicate centering and center of symmetry)
3. an interatomic distance in A (float)
4. an uncertainty on the distance in A or 0.0 (float)

**AngArray** is a dict keyed by the origin (central) atom number where the value is a list of angle entries. The value for each angle entry consists of three values:

0. a distance item reference for one neighbor (int)
1. a distance item reference for a second neighbor (int)
2. a angle, uncertainty pair; the s.u. may be zero (tuple of two floats)

The AngArray distance reference items refer directly to the index of the items in the DistArray item for the list of distances for the central atom.

GSASIIstrMain.**SeqRefine** (*GPXfile, dlg*)

Perform a sequential refinement – cycles through all selected histograms, one at a time

GSASIIstrMain.**main** ()

Needs a doc string

## 6.2 GSASIIstrMath - structure math routines

GSASIIstrMath.**ApplyRBModelDerivs** (*dFdvDict, parmDict, rigidbodyDict, Phase*)

Needs a doc string

GSASIIstrMath.**ApplyRBModels** (*parmDict, Phases, rigidbodyDict, Update=False*)

Takes RB info from RBModels in Phase and RB data in rigidbodyDict along with current RB values in parmDict & modifies atom contents (xyz & Uij) of parmDict

GSASIIstrMath.**ApplyXYZshifts** (*parmDict, varyList*)

takes atom x,y,z shift and applies it to corresponding atom x,y,z value

### Parameters

- **parmDict** (*dict*) – parameter dictionary
- **varyList** (*list*) – list of variables (not used!)

**Returns** newAtomDict - dictionary of new atomic coordinate names & values; key is parameter shift name

GSASIIstrMath.**Dict2Values** (*parmdict, varylist*)

Use before call to leastsq to setup list of values for the parameters in parmdict, as selected by key in varylist

GSASIIstrMath.**GetAbsorb** (*refl, im, hfx, calcControls, parmDict*)

Needs a doc string

GSASIIstrMath.**GetAbsorbDerv** (*refl, im, hfx, calcControls, parmDict*)

Needs a doc string



- GSASIIstrMath.**GetAtomFXU** (*pfx, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetAtomSSFUXU** (*pfx, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetFobsSq** (*Histograms, Phases, parmDict, calcControls*)  
Needs a doc string
- GSASIIstrMath.**GetHStrainShift** (*refl, im, SGData, phfx, hfx, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetHStrainShiftDerv** (*refl, im, SGData, phfx, hfx, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetIntensityCorr** (*refl, im, uniq, G, g, pfx, phfx, hfx, SGData, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetIntensityDerv** (*refl, im, wave, uniq, G, g, pfx, phfx, hfx, SGData, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetNewCellParms** (*parmDict, varyList*)  
Needs a doc string
- GSASIIstrMath.**GetPrefOri** (*uniq, G, g, phfx, hfx, SGData, calcControls, parmDict*)  
March-Dollase preferred orientation correction
- GSASIIstrMath.**GetPrefOriDerv** (*refl, im, uniq, G, g, phfx, hfx, SGData, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetPwdrExt** (*refl, im, pfx, phfx, hfx, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetPwdrExtDerv** (*refl, im, pfx, phfx, hfx, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetRef1Pos** (*refl, im, wave, A, pfx, hfx, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetRef1PosDerv** (*refl, im, wave, A, pfx, hfx, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetSampleSigGam** (*refl, im, wave, G, GB, SGData, hfx, phfx, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**GetSampleSigGamDerv** (*refl, im, wave, G, GB, SGData, hfx, phfx, calcControls, parmDict*)  
Needs a doc string
- GSASIIstrMath.**HessRefine** (*values, HistoPhases, parmDict, varylist, calcControls, pawleyLookup, dlg*)  
Loop over histograms and compute derivatives of the fitting model (M) with respect to all parameters. For each histogram, the Jacobian matrix, dMdv, with dimensions (n by m) where n is the number of parameters and m is the number of data points *in the histogram*. The (n by n) Hessian is computed from each Jacobian and it is returned. This routine is used when refinement derivatives are selected as “analytic Hessian” in Controls.
- Returns** Vec,Hess where Vec is the least-squares vector and Hess is the Hessian
- GSASIIstrMath.**SCExtinction** (*ref, im, phfx, hfx, pfx, calcControls, parmDict, varyList*)  
Single crystal extinction function; returns extinction & derivative

GSASIIstrMath.**SHPOcal** (*refl, im, g, phfx, hfx, SGData, calcControls, parmDict*)  
 spherical harmonics preferred orientation (cylindrical symmetry only)

GSASIIstrMath.**SHPOcalDerv** (*refl, im, g, phfx, hfx, SGData, calcControls, parmDict*)  
 spherical harmonics preferred orientation derivatives (cylindrical symmetry only)

GSASIIstrMath.**SHTXcal** (*refl, im, g, pfx, hfx, SGData, calcControls, parmDict*)  
 Spherical harmonics texture

GSASIIstrMath.**SHTXcalDerv** (*refl, im, g, pfx, hfx, SGData, calcControls, parmDict*)  
 Spherical harmonics texture derivatives

GSASIIstrMath.**SStructureFactor2** (*refDict, im, G, hfx, pfx, SGData, SSGData, calcControls, parmDict*)

Compute super structure factors for all h,k,l,m for phase puts the result, F<sup>2</sup>, in each ref[8+im] in refList input:

**Parameters**

- **refDict** (*dict*) – where ‘RefList’ list where each ref = h,k,l,m,d,... ‘FF’ dict of form factors - filed in below
- **G** (*np.array*) – reciprocal metric tensor
- **pfx** (*str*) – phase id string
- **SGData** (*dict*) – space group info. dictionary output from SpcGroup
- **calcControls** (*dict*) –
- **ParmDict** (*dict*) –

GSASIIstrMath.**SStructureFactorDerv** (*refDict, im, G, hfx, pfx, SGData, SSGData, calcControls, parmDict*)

Needs a doc string

GSASIIstrMath.**StructureFactor2** (*refDict, G, hfx, pfx, SGData, calcControls, parmDict*)

Compute structure factors for all h,k,l for phase puts the result, F<sup>2</sup>, in each ref[8] in refList input:

**Parameters**

- **refDict** (*dict*) – where ‘RefList’ list where each ref = h,k,l,m,d,... ‘FF’ dict of form factors - filed in below
- **G** (*np.array*) – reciprocal metric tensor
- **pfx** (*str*) – phase id string
- **SGData** (*dict*) – space group info. dictionary output from SpcGroup
- **calcControls** (*dict*) –
- **ParmDict** (*dict*) –

GSASIIstrMath.**StructureFactorDerv** (*refDict, G, hfx, pfx, SGData, calcControls, parmDict*)

Needs a doc string

GSASIIstrMath.**Values2Dict** (*parmdict, varylist, values*)

Use after call to leastsq to update the parameter dictionary with values corresponding to keys in varylist

GSASIIstrMath.**dervHKLf** (*Histogram, Phase, calcControls, varylist, parmDict, rigidbodyDict*)

Loop over reflections in a HKLF histogram and compute derivatives of the fitting model (M) with respect to all parameters. Independent and dependant dM/dp arrays are returned to either dervRefine or HessRefine.

**Returns**

GSASIIstrMath.**dervRefine** (*values, HistoPhases, parmDict, varylist, calcControls, pawleyLookup, dlg*)

Loop over histograms and compute derivatives of the fitting model (M) with respect to all parameters. Results are returned in a Jacobian matrix (aka design matrix) of dimensions (n by m) where n is the number of parameters and m is the number of data points. This can exceed memory when m gets large. This routine is used when refinement derivatives are selected as “analytic Jacobian” in Controls.

**Returns** Jacobian numpy.array dMdv for all histograms concatenated

GSASIIstrMath.**errRefine** (*values, HistoPhases, parmDict, varylist, calcControls, pawleyLookup, dlg=None*)

Computes the point-by-point discrepancies between every data point in every histogram and the observed value

**Returns** an np array of differences between observed and computed diffraction values.

GSASIIstrMath.**getPowderProfile** (*parmDict, x, varylist, Histogram, Phases, calcControls, pawleyLookup*)

Needs a doc string

GSASIIstrMath.**getPowderProfileDerv** (*parmDict, x, varylist, Histogram, Phases, rigidbodyDict, calcControls, pawleyLookup*)

Needs a doc string

GSASIIstrMath.**penaltyDeriv** (*pNames, pVal, HistoPhases, calcControls, parmDict, varyList*)

Needs a doc string

GSASIIstrMath.**penaltyFxn** (*HistoPhases, calcControls, parmDict, varyList*)

Needs a doc string

## 6.3 GSASIIstrIO: structure I/O routines

GSASIIstrIO.**GPXBackup** (*GPXfile, makeBack=True*)

makes a backup of the current .gpx file (?)

### Parameters

- **GPXfile** (*str*) – full .gpx file name
- **makeBack** (*bool*) – if True (default), the backup is written to a new file; if False, the last backup is overwritten

**Returns** the name of the backup file that was written

GSASIIstrIO.**GetAllPhaseData** (*GPXfile, PhaseName*)

Returns the entire dictionary for PhaseName from GSASII gpx file

### Parameters

- **GPXfile** (*str*) – full .gpx file name
- **PhaseName** (*str*) – phase name

**Returns** phase dictionary

GSASIIstrIO.**GetConstraints** (*GPXfile*)

Read the constraints from the GPX file and interpret them

called in `ReadCheckConstraints()`, `GSASIIstrMain.Refine()` and `GSASIIstrMain.SeqRefine()`.

GSASIIstrIO.**GetControls** (*GPXfile*)

Returns dictionary of control items found in GSASII gpx file

**Parameters** **GPXfile** (*str*) – full .gpx file name

**Returns** dictionary of control items

GSASIIstrIO.**GetFprime** (*controlDict, Histograms*)

Needs a doc string

GSASIIstrIO.**GetHistogramData** (*Histograms, Print=True, pFile=None*)

needs a doc string

GSASIIstrIO.**GetHistogramNames** (*GPXfile, hType*)

Returns a list of histogram names found in GSASII gpx file

**Parameters**

- **GPXfile** (*str*) – full .gpx file name
- **hType** (*str*) – list of histogram types

**Returns** list of histogram names (types = PWDR & HKLF)

GSASIIstrIO.**GetHistogramPhaseData** (*Phases, Histograms, Print=True, pFile=None, resetRefList=True*)

Loads the HAP histogram/phase information into dicts

**Parameters**

- **Phases** (*dict*) – phase information
- **Histograms** (*dict*) – Histogram information
- **Print** (*bool*) – prints information as it is read
- **pFile** (*file*) – file object to print to (the default, None causes printing to the console)
- **resetRefList** (*bool*) – Should the contents of the Reflection List be initialized on loading. The default, True, initializes the Reflection List as it is loaded.

**Returns** (hapVary,hapDict,controlDict) \* hapVary: list of refined variables \* hapDict: dict with refined variables and their values \* controlDict: dict with fixed parameters

GSASIIstrIO.**GetHistograms** (*GPXfile, hNames*)

Returns a dictionary of histograms found in GSASII gpx file

**Parameters**

- **GPXfile** (*str*) – full .gpx file name
- **hNames** (*str*) – list of histogram names

**Returns** dictionary of histograms (types = PWDR & HKLF)

GSASIIstrIO.**GetPawleyConstr** (*SGLaue, PawleyRef, im, pawleyVary*)

needs a doc string

GSASIIstrIO.**GetPhaseData** (*PhaseData, RestraintDict={}, rIds={}, Print=True, pFile=None, seqRef=False*)

needs a doc string

GSASIIstrIO.**GetPhaseNames** (*GPXfile*)

Returns a list of phase names found under 'Phases' in GSASII gpx file

**Parameters** **GPXfile** (*str*) – full .gpx file name

**Returns** list of phase names

GSASIIstrIO.**GetRestrains** (*GPXfile*)

Read the restraints from the GPX file. Throws an exception if not found in the .GPX file

GSASIIstrIO.**GetRigidBodies** (*GPXfile*)

Read the rigid body models from the GPX file

GSASIIstrIO.**GetRigidBodyModels** (*rigidbodyDict, Print=True, pFile=None*)

needs a doc string

GSASIIstrIO.**GetUsedHistogramsAndPhases** (*GPXfile*)

Returns all histograms that are found in any phase and any phase that uses a histogram. This also assigns numbers to used phases and histograms by the order they appear in the file.

**Parameters** *GPXfile* (*str*) – full .gpx file name

**Returns**

(Histograms,Phases)

- Histograms = dictionary of histograms as {name:data,...}
- Phases = dictionary of phases that use histograms

GSASIIstrIO.**PrintRestrains** (*cell, SGData, AtPtrs, Atoms, AtLookup, textureData, phaseRest, pFile*)

needs a doc string

GSASIIstrIO.**ProcessConstraints** (*constList*)

Interpret the constraints in the constList input into a dictionary, etc. All GSASIIobj.G2VarObj objects are mapped to the appropriate phase/hist/atoms based on the object internals (random Ids). If this can't be done (if a phase has been deleted, etc.), the variable is ignored. If the constraint cannot be used due to too many dropped variables, it is counted as ignored.

**Parameters** *constList* (*list*) – a list of lists where each item in the outer list specifies a constraint of some form, as described in the GSASIIobj *Constraint definition*.

**Returns**

a tuple of (constDict,fixedList,ignored) where:

- constDict (list of dicts) contains the constraint relationships
- fixedList (list) contains the fixed values for each type of constraint.
- ignored (int) counts the number of invalid constraint items (should always be zero!)

GSASIIstrIO.**ReadCheckConstraints** (*GPXfile*)

Load constraints and related info and return any error or warning messages

GSASIIstrIO.**SetHistogramData** (*parmDict, sigDict, Histograms, FFtables, Print=True, pFile=None*)

needs a doc string

GSASIIstrIO.**SetHistogramPhaseData** (*parmDict, sigDict, Phases, Histograms, FFtables, Print=True, pFile=None*)

needs a doc string

GSASIIstrIO.**SetPhaseData** (*parmDict, sigDict, Phases, RBIds, covData, RestraintDict=None, pFile=None*)

needs a doc string

GSASIIstrIO.**SetRigidBodyModels** (*parmDict, sigDict, rigidbodyDict, pFile=None*)

needs a doc string

GSASIIstrIO.**SetSeqResult** (*GPXfile, Histograms, SeqResult*)

Needs doc string

**Parameters** *GPXfile* (*str*) – full .gpx file name

GSASIIstrIO.**SetUsedHistogramsAndPhases** (*GPXfile, Histograms, Phases, RigidBodies, CovData, makeBack=True*)

Updates gpxfile from all histograms that are found in any phase and any phase that used a histogram. Also updates rigid body definitions.

**Parameters**

- **GPXfile** (*str*) – full .gpx file name
- **Histograms** (*dict*) – dictionary of histograms as {name:data,...}
- **Phases** (*dict*) – dictionary of phases that use histograms
- **RigidBodies** (*dict*) – dictionary of rigid bodies
- **CovData** (*dict*) – dictionary of refined variables, varyList, & covariance matrix
- **makeBack** (*bool*) – True if new backup of .gpx file is to be made; else use the last one made

GSASIIstrIO.**ShowBanner** (*pFile=None*)

Print authorship, copyright and citation notice

GSASIIstrIO.**ShowControls** (*Controls, pFile=None, SeqRef=False*)

Print controls information

GSASIIstrIO.**cellFill** (*pfx, SGData, parmDict, sigDict*)

Returns the filled-out reciprocal cell (A) terms and their uncertainties from the parameter and sig dictionaries.

**Parameters**

- **pfx** (*str*) – parameter prefix (“n:”, where n is a phase number)
- **SGdata** (*dict*) – a symmetry object
- **parmDict** (*dict*) – a dictionary of parameters
- **sigDict** (*dict*) – a dictionary of uncertainties on parameters

**Returns** A,sigA where each is a list of six terms with the A terms

GSASIIstrIO.**cellVary** (*pfx, SGData*)

needs a doc string

GSASIIstrIO.**getBackupName** (*GPXfile, makeBack*)

Get the name for the backup .gpx file name

**Parameters**

- **GPXfile** (*str*) – full .gpx file name
- **makeBack** (*bool*) – if True the name of a new file is returned, if False the name of the last file that exists is returned

**Returns** the name of a backup file

GSASIIstrIO.**getCellEsd** (*pfx, SGData, A, covData*)

needs a doc string

## ***GSASIIMAPVARS: PARAMETER CONSTRAINTS***

Module to implements algebraic constraints, parameter redefinition and parameter simplification constraints.

Parameter redefinition (new vars) is done by creating one or more relationships between a set of parameters

```
Mx1 * Px + My1 * Py + ...  
Mx2 * Px + Mz2 * Pz + ...
```

where  $P_j$  is a parameter name and  $M_{jk}$  is a constant.

Constant constraint Relations can also be supplied in the form of an equation:

```
nx1 * Px + ny1 * Py + ... = C1
```

where  $C_n$  is a constant. These equations define an algebraic constraint.

Parameters can also be “fixed” (held), which prevents them from being refined.

All of the above three cases are input using routines `GroupConstraints` and `GenerateConstraints`. The input consists of a list of relationship dictionaries:

```
constrDict = [  
    {'0:12:Scale': 2.0, '0:14:Scale': 4.0, '0:13:Scale': 3.0, '0:0:Scale': 0.5},  
    {'2::C(10,6,1)': 1.0, '1::C(10,6,1)': 1.0},  
    {'0::A0': 0.0}]  
fixedList = ['5.0', None, '0']
```

Where the dictionary defines the first part of an expression and the corresponding `fixedList` item is either `None` (for parameter redefinition) or the constant value, for a constant constraint equation. A dictionary that contains a single term defines a variable that will be fixed (held). The multiplier and the `fixedList` value in this case are ignored.

Parameters can also be equivalenced or “slaved” to another parameter, such that one (independent) parameter is equated to several (now dependent) parameters. In algebraic form this is:

```
P0 = M1 * P1 = M2 * P2 = ...
```

Thus parameters  $P_0, P_1$  and  $P_2, \dots$  are linearly equivalent. Routine `StoreEquivalence` is used to specify these equivalences.

Parameter redefinition (new vars) describes a new, independent, parameter, which is defined in terms of dependent parameters that are defined in the Model, while fixed constrained relations effectively reduce the complexity of the Model by removing a degree of freedom. It is possible for a parameter to appear in both a parameter redefinition expression and a fixed constraint equation, but a parameter cannot be used a parameter equivalence cannot be used elsewhere (not fixed, constrained or redefined). Likewise a fixed parameter cannot be used elsewhere (not equivalenced, constrained or redefined).

Relationships are grouped so that a set of dependent parameters appear in only one group (done in routine `GroupConstraints`.) Note that if a group contains relationships/equations that involve  $N$  dependent parameters, there must exist  $N-C$  new parameters, where  $C$  is the number of constraint equations in the group. Routine `GenerateConstraints`

takes the output from `GroupConstraints` and generates the “missing” relationships and saves that information in the module’s global variables. Each generated parameter is named sequentially using `paramPrefix`.

A list of parameters that will be varied is specified as input to `GenerateConstraints` (`varyList`). A fixed parameter will simply be removed from this list preventing that parameter from being varied. Note that all parameters in a constraint relationship must be specified as varied (appear in `varyList`) or none can be varied. This is checked in `GenerateConstraints`. Likewise, if all parameters in a constraint are not referenced in a refinement, the constraint is ignored, but if some parameters in a constraint group are not referenced in a refinement, but others are this constitutes an error.

- When a new variable is created, the variable is assigned the name associated in the constraint definition or it is assigned a default name of form `: : constr<n>` (see `paramPrefix`). The vary setting for variables used in the constraint are ignored. Note that any generated “missing” relations are not varied. Only the input relations can be varied.
- If all parameters in a fixed constraint equation are varied, the generated “missing” relations in the group are all varied. This provides the N-C degrees of freedom.

## 7.1 External Routines

To define a set of constrained and unconstrained relations, one defines a list of dictionary defining constraint parameters and their values, a list of fixed values for each constraint and a list of parameters to be varied. In addition, one uses `StoreEquivalence()` to define parameters that are equivalent. One can then use `CheckConstraints()` to check that the input is internally consistent and finally `GroupConstraints()` and `GenerateConstraints()` to generate the internally used tables. Routines `Map2Dict()` is used to initialize the parameter dictionary and `Dict2Map()`, `Dict2Deriv()`, and `ComputeDepESD()` are used to apply constraints. Routine `VarRemapShow()` is used to print out the constraint information, as stored by `GenerateConstraints()`.

**InitVars()** This is optionally used to clear out all defined previously defined constraint information

**StoreEquivalence()** To implement parameter redefinition, one calls `StoreEquivalence`. This should be called for every set of equivalence relationships. There is no harm in using `StoreEquivalence` with the same independent variable:

```
StoreEquivalence('x', ('y',))
StoreEquivalence('x', ('z',))
```

or equivalently

```
StoreEquivalence('x', ('y', 'z'))
```

The latter will run more efficiently. Note that mixing independent and dependent variables is problematic. This is not allowed:

```
StoreEquivalence('x', ('y',))
StoreEquivalence('y', ('z',))
```

Use `StoreEquivalence` before calling `GenerateConstraints` or `CheckConstraints`

**CheckConstraints()** To check that input is internally consistent, use `CheckConstraints`

**Map2Dict()** To determine values for the parameters created in this module, one calls `Map2Dict`. This will not apply constraints.

**Dict2Map()** To take values from the new independent parameters and constraints, one calls `Dict2Map`. This will apply constraints.

**Dict2Deriv()** Use `Dict2Deriv` to determine derivatives on independent parameters from those on dependent ones



**ComputeDepESD ()** Use ComputeDepESD to compute uncertainties on dependent variables

**VarRemapShow ()** To show a summary of the parameter remapping, one calls VarRemapShow

## 7.2 Global Variables

**dependentParmList:** contains a list by group of lists of parameters used in the group. Note that parameters listed in dependentParmList should not be refined as they will not affect the model

**indParmList:** a list of groups of Independent parameters defined in each group. This contains both parameters used in parameter redefinitions as well as names of generated new parameters.

**fixedVarList:** a list of variables that have been ‘fixed’ by defining them as equal to a constant (`::var: = 0`). Note that the constant value is ignored at present. These variables are later removed from varyList which prevents them from being refined. Unlikely to be used externally.

**arrayList:** a list by group of relationship matrices to relate parameters in dependentParmList to those in indParmList. Unlikely to be used externally.

**invarrayList:** a list by group of relationship matrices to relate parameters in indParmList to those in dependentParmList. Unlikely to be used externally.

**fixedDict:** a dictionary containing the fixed values corresponding to parameter equations. The dict key is an ascii string, but the dict value is a float. Unlikely to be used externally.

## 7.3 Routines

Note that parameter names in GSAS-II are strings of form `<ph>:<hst>:<nam>`

`GSASIImapvars.CheckConstraints (varyList, constrDict, fixedList)`

Takes a list of relationship entries comprising a group of constraints and checks for inconsistencies such as conflicts in parameter/variable definitions and or inconsistently varied parameters.

### Parameters

- **varyList** (*list*) – a list of parameters names that will be varied
- **constrDict** (*dict*) – a list of dicts defining relationships/constraints (as created in `GSASIIstrIO.ProcessConstraints()` and documented in `GroupConstraints()`)
- **fixedList** (*list*) – a list of values specifying a fixed value for each dict in constrDict. Values are either strings that can be converted to floats or `None` if the constraint defines a new parameter rather than a constant.

### Returns

two strings:

- the first lists conflicts internal to the specified constraints
- the second lists conflicts where the varyList specifies some parameters in a constraint, but not all

If there are no errors, both strings will be empty

`GSASIImapvars.ComputeDepESD (covMatrix, varyList, parmDict)`

Compute uncertainties for dependent parameters from independent ones returns a dictionary containing the esd values for dependent parameters

GSASIImapvars.**Dict2Deriv** (*varyList*, *derivDict*, *dMdv*)

Compute derivatives for Independent Parameters from the derivatives for the original parameters

**Parameters**

- **varyList** (*list*) – a list of parameters names that will be varied
- **derivDict** (*dict*) – a dict containing derivatives for parameter values keyed by the parameter names.
- **dMdv** (*list*) – a Jacobian, as a list of np.array containing derivatives for dependent parameter computed from derivDict

GSASIImapvars.**Dict2Map** (*parmDict*, *varyList*)

Applies the constraints defined using `StoreEquivalence()`, `GroupConstraints()` and `GenerateConstraints()` by changing values in a dict containing the parameters. This should be done before the parameters are used for any computations

**Parameters**

- **parmDict** (*dict*) – a dict containing parameter values keyed by the parameter names. This will contain updated values for both dependent and independent parameters after Dict2Map is called. It will also contain some unexpected entries of every constant value {'0':0.0} & {'1.0':1.0}, which do not cause any problems.
- **varyList** (*list*) – a list of parameters names that will be varied

GSASIImapvars.**GenerateConstraints** (*groups*, *parmlist*, *varyList*, *constrDict*, *fixedList*, *parmDict=None*, *SeqHist=None*)

Takes a list of relationship entries comprising a group of constraints and builds the relationship lists and their inverse and stores them in global variables Also checks for internal conflicts or inconsistencies in parameter/variable definitions.

**Parameters**

- **groups** (*list*) – a list of grouped constraints where each constraint grouped contains a list of indices for constraint constrDict entries, created in `GroupConstraints()` (returned as 1st value)
- **parmlist** (*list*) – a list containing lists of parameter names contained in each group, created in `GroupConstraints()` (returned as 2nd value)
- **varyList** (*list*) – a list of parameters names (strings of form <ph>:<hst>:<nam>) that will be varied. Note that this is changed here.
- **constrDict** (*dict*) – a list of dicts defining relationships/constraints (as defined in `GroupConstraints()`)
- **fixedList** (*list*) – a list of values specifying a fixed value for each dict in constrDict. Values are either strings that can be converted to floats, float values or None if the constraint defines a new parameter.
- **parmDict** (*dict*) – a dict containing all parameters defined in current refinement.
- **SeqHist** (*int*) – number of current histogram, when used in a sequential refinement. None (default) otherwise. Wildcard variable names are set to the current histogram, when found if not None.

GSASIImapvars.**GetDependentVars** ()

Return a list of dependent variables: e.g. variables that are constrained in terms of other variables

**Returns** a list of variable names

GSASIImapvars.**GetIndependentVars** ()

Return a list of independent variables: e.g. variables that are created by constraints of other variables

**Returns** a list of variable names

GSASIImapvars.**GramSchmidtOrtho** (*a*, *nkeep=0*)

Use the Gram-Schmidt process (<http://en.wikipedia.org/wiki/Gram-Schmidt>) to find orthonormal unit vectors relative to first row.

If *nkeep* is non-zero, the first *nkeep* rows in the array are not changed

**input:** *arrayin*: a 2-D non-singular square array

**returns:** a orthonormal set of unit vectors as a square array

GSASIImapvars.**GroupConstraints** (*constrDict*)

divide the constraints into groups that share no parameters.

**Parameters** *constrDict* (*dict*) – a list of dicts defining relationships/constraints

*constrDict* = [{<constr1>}, {<constr2>}, ...]

where {<constr1>} is {‘var1’: mult1, ‘var2’: mult2,... }

**Returns**

two lists of lists:

- a list of grouped constraints where each constraint grouped contains a list of indices for constraint *constrDict* entries
- a list containing lists of parameter names contained in each group

GSASIImapvars.**InitVars** ()

Initializes all constraint information

GSASIImapvars.**Map2Dict** (*parmDict*, *varyList*)

Create (or update) the Independent Parameters from the original set of Parameters

Removes dependent variables from the *varyList*

This should be done once, after the constraints have been defined using `StoreEquivalence()`, `GroupConstraints()` and `GenerateConstraints()` and before any variable refinement is done. This completes the parameter dictionary by defining independent parameters and it satisfies the constraint equations in the initial parameters

**Parameters**

- **parmDict** (*dict*) – a dict containing parameter values keyed by the parameter names. This will contain updated values for both dependent and independent parameters after `Dict2Map` is called. It will also contain some unexpected entries of every constant value {‘0’:0.0} & {‘1.0’:1.0}, which do not cause any problems.
- **varyList** (*list*) – a list of parameters names that will be varied

GSASIImapvars.**PrintIndependentVars** (*parmDict*, *varyList*, *sigDict*, *PrintAll=False*, *pFile=None*)

Print the values and uncertainties on the independent variables

GSASIImapvars.**StoreEquivalence** (*independentVar*, *dependentList*)

Takes a list of dependent parameter(s) and stores their relationship to a single independent parameter (*independentVar*)

**Parameters**

- **independentVar** (*str*) – name of master parameter that will be used to determine the value to set the dependent variables

- **dependentList** (*list*) – a list of parameters that will set from independentVar. Each item in the list can be a string with the parameter name or a tuple containing a name and multiplier:  
`['parm1', ('parm2', .5), ]`

GSASIImapvars.**VarKeys** (*constr*)

Finds the keys in a constraint that represent variables e.g. eliminates any that start with '\_'

**Parameters** *constr* (*dict*) – a single constraint entry of form:

```
{'var1': mult1, 'var2': mult2, ... '_notVar': val, ...}
```

(see `GroupConstraints()`)

**Returns** a list of keys where any keys beginning with '\_' are removed.

GSASIImapvars.**VarRemapShow** (*varyList*, *inputOnly=False*)

List out the saved relationships. This should be done after the constraints have been defined using `StoreEquivalence()`, `GroupConstraints()` and `GenerateConstraints()`.

**Returns** a string containing the details of the constraint relationships

## GSASIIIMAGE: IMAGE CALC MODULE

Ellipse fitting & image integration

GSASIIimage.**EdgeFinder** (*image, data*)

this makes list of all x,y where I>edgeMin suitable for an ellipse search? Not currently used but might be useful in future?

GSASIIimage.**Fill2ThetaAzimuthMap** (*masks, TA, tam, image*)

Needs a doc string

GSASIIimage.**FitDetector** (*rings, varyList, parmDict, Print=True*)

Needs a doc string

GSASIIimage.**FitStrSta** (*Image, StrSta, Controls*)

Needs a doc string

GSASIIimage.**FitStrain** (*rings, p0, dset, wave, phi, StaType*)

Needs a doc string

GSASIIimage.**GetAzm** (*x, y, data*)

Give azimuth value for detector x,y position; calibration info in data

GSASIIimage.**GetDetXYfromThAzm** (*Th, Azm, data*)

Needs a doc string

GSASIIimage.**GetDetectorXY** (*dsp, azm, data*)

Needs a doc string

GSASIIimage.**GetDsp** (*x, y, data*)

Give d-spacing value for detector x,y position; calibration info in data

GSASIIimage.**GetEllipse** (*dsp, data*)

uses Dandelin spheres to find ellipse or hyperbola parameters from detector geometry as given in image controls dictionary (data) and a d-spacing (dsp)

GSASIIimage.**GetEllipse2** (*tth, dxy, dist, cent, tilt, phi*)

uses Dandelin spheres to find ellipse or hyperbola parameters from detector geometry on output radii[0] (b-minor axis) set < 0. for hyperbola

GSASIIimage.**GetTth** (*x, y, data*)

Give 2-theta value for detector x,y position; calibration info in data

GSASIIimage.**GetTthAzm** (*x, y, data*)

Give 2-theta, azimuth values for detector x,y position; calibration info in data

GSASIIimage.**GetTthAzmDsp** (*x, y, data*)

Needs a doc string - checked OK for ellipses & hyperbola

GSASIIimage.**GetTthAzmG** (*x, y, data*)

Give 2-theta, azimuth & geometric corr. values for detector x,y position; calibration info in data - only used in integration

GSASIIimage.**ImageCalibrate** (*self, data*)

Needs a doc string

GSASIIimage.**ImageCompress** (*image, scale*)

Needs a doc string

GSASIIimage.**ImageIntegrate** (*image, data, masks, blkSize=128, dlg=None, returnN=False*)

Needs a doc string

GSASIIimage.**ImageLocalMax** (*image, w, Xpix, Ypix*)

Needs a doc string

GSASIIimage.**ImageRecalibrate** (*self, data, masks*)

Needs a doc string

GSASIIimage.**Make2ThetaAzimuthMap** (*data, masks, iLim, jLim, times*)

Needs a doc string

GSASIIimage.**calcFij** (*omg, phi, azm, th*)

Does something...

Uses parameters as defined by Bob He & Kingsley Smith, Adv. in X-Ray Anal. 41, 501 (1997)

#### Parameters

- **omg** – his omega = sample omega rotation; 0 when incident beam || sample surface, 90 when perp. to sample surface
- **phi** – his phi = sample phi rotation; usually = 0, axis rotates with omg.
- **azm** – his chi = azimuth around incident beam
- **th** – his theta = theta

GSASIIimage.**checkEllipse** (*Zsum, distSum, xSum, ySum, dist, x, y*)

Needs a doc string

GSASIIimage.**makeMat** (*Angle, Axis*)

Make rotation matrix from Angle and Axis

#### Parameters

- **Angle** (*float*) – in degrees
- **Axis** (*int*) – 0 for rotation about x, 1 for about y, etc.

GSASIIimage.**makeRing** (*dsp, ellipse, pix, reject, scalex, scaley, image*)

Needs a doc string

GSASIIimage.**peneCorr** (*tth, dep, tilt=0.0, azm=0.0*)

Needs a doc string

GSASIIimage.**pointInPolygon** (*pXY, xy*)

Needs a doc string

---

**GSASII MATH: COMPUTATION MODULE**


---

Routines for least-squares minimization and other stuff

GSASIImath.**AV2Q** (*A*, *V*)

convert angle (radians) & vector to quaternion  $q=r+ai+bj+ck$

GSASIImath.**AVdeg2Q** (*A*, *V*)

convert angle (degrees) & vector to quaternion  $q=r+ai+bj+ck$

GSASIImath.**ApplyModulation** (*data*, *tau*)

Applies modulation to drawing atom positions & Uij's for given tau

GSASIImath.**AtomTLS2UIJ** (*atomData*, *atPtrs*, *Amat*, *rbObj*)

default doc string

**Parameters** *name* (*type*) – description

**Returns** type name: description

GSASIImath.**AtomUij2TLS** (*atomData*, *atPtrs*, *Amat*, *Bmat*, *rbObj*)

default doc string

**Parameters** *name* (*type*) – description

**Returns** type name: description

GSASIImath.**BessIn** (*nmax*, *x*)

compute modified Bessel function  $I(n,x)$  from scipy routines & recurrence relation returns sequence of  $I(n,x)$  for  $n$  in range  $[-nmax...0...nmax]$

**Parameters**

- **nmax** (*integer*) – maximul order for  $I_n(x)$
- **x** (*float*) – argument for  $I_n(x)$

**Returns** numpy array  $[I(-nmax,x)...I(0,x)...I(nmax,x)]$

GSASIImath.**BessJn** (*nmax*, *x*)

compute Bessel function  $J(n,x)$  from scipy routine & recurrence relation returns sequence of  $J(n,x)$  for  $n$  in range  $[-nmax...0...nmax]$

**Parameters**

- **nmax** (*integer*) – maximul order for  $J_n(x)$
- **x** (*float*) – argument for  $J_n(x)$

**Returns** numpy array  $[J(-nmax,x)...J(0,x)...J(nmax,x)]$

GSASIImath.**ChargeFlip** (*data*, *reflDict*, *pgbar*)

default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**Den2Vol** (*Elements, density*)  
converts density to molecular volume

**Parameters**

- **Elements** (*dict*) – elements in molecular formula; each must contain Num: number of atoms in formula Mass: at. wt.
- **density** (*float*) – material density in gm/cm<sup>3</sup>

**Returns** float volume: molecular volume in A<sup>3</sup>

GSASIImath.**E12EstVol** (*Elements*)  
Estimate volume from molecular formula; assumes atom volume = 10A<sup>3</sup>

**Parameters** **Elements** (*dict*) – elements in molecular formula; each must contain Num: number of atoms in formula

**Returns** float volume: estimate of molecular volume in A<sup>3</sup>

GSASIImath.**E12Mass** (*Elements*)  
compute molecular weight from Elements

**Parameters** **Elements** (*dict*) – elements in molecular formula; each must contain Num: number of atoms in formula Mass: at. wt.

**Returns** float mass: molecular weight.

GSASIImath.**FillAtomLookUp** (*atomData, indx*)  
create a dictionary of atom indexes with atom IDs as keys

**Parameters** **atomData** (*list*) – Atom table to be used

**Returns** dict atomLookUp: dictionary of atom indexes with atom IDs as keys

GSASIImath.**FindAtomIndexByIDs** (*atomData, loc, IDs, Draw=True*)  
finds the set of atom array indices for a list of atom IDs. Will search either the Atom table or the drawAtom table.

**Parameters**

- **atomData** (*list*) – Atom or drawAtom table containing coordinates, etc.
- **loc** (*int*) – location of atom id in atomData record
- **IDs** (*list*) – atom IDs to be found
- **Draw** (*bool*) – True if drawAtom table to be searched; False if Atom table is searched

**Returns** list indx: atom (or drawAtom) indices

GSASIImath.**Fourier4DMap** (*data, reflDict*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**FourierMap** (*data, reflDict*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description



GSASIImath.**GetAngleSig** (*Oatoms, Atoms, Amat, SGData, covData={}*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**GetAtomCoordsByID** (*pId, parmDict, AtLookup, indx*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**GetAtomItemsById** (*atomData, atomLookUp, IdList, itemLoc, numItems=1*)  
gets atom parameters for atoms using atom IDs

**Parameters**

- **atomData** (*list*) – Atom table to be used
- **atomLookUp** (*dict*) – dictionary of atom indexes with atom IDs as keys
- **IdList** (*list*) – atom IDs to be found
- **itemLoc** (*int*) – pointer to desired 1st item in an atom table entry
- **numItems** (*int*) – number of items to be retrieved

**Returns** type name: description

GSASIImath.**GetAtomsById** (*atomData, atomLookUp, IdList*)  
gets a list of atoms from Atom table that match a set of atom IDs

**Parameters**

- **atomData** (*list*) – Atom table to be used
- **atomLookUp** (*dict*) – dictionary of atom indexes with atom IDs as keys
- **IdList** (*list*) – atom IDs to be found

**Returns** list atoms: list of atoms found

GSASIImath.**GetDATSig** (*Oatoms, Atoms, Amat, SGData, covData={}*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**GetDistSig** (*Oatoms, Atoms, Amat, SGData, covData={}*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**GetSHCoeff** (*pId, parmDict, SHkeys*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**GetTorsionSig** (*Oatoms, Atoms, Amat, SGData, covData={}*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**GetXYZDist** (*xyz, XYZ, Amat*)

**gets distance from position xyz to all XYZ, xyz & XYZ are np.array** and are in crystal coordinates; Amat is crystal to Cart matrix

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**HessianLSQ** (*func, x0, Hess, args=(), ftol=1.49012e-08, xtol=1.49012e-08, maxcyc=0, Print=False*)

Minimize the sum of squares of a function (*f*) evaluated on a series of values (*y*):  $\sum_{y=0}^{N_{obs}} f(y, args)$

$$x = \arg \min_{y=0}^{N_{obs}} (\text{sum}(\text{func}(y) ** 2, \text{axis}=0))$$

### Parameters

- **func** (*function*) – callable method or function should take at least one (possibly length N vector) argument and returns M floating point numbers.
- **x0** (*np.ndarray*) – The starting estimate for the minimization of length N
- **Hess** (*function*) – callable method or function A required function or method to compute the weighted vector and Hessian for func. It must be a symmetric NxN array
- **args** (*tuple*) – Any extra arguments to func are placed in this tuple.
- **ftol** (*float*) – Relative error desired in the sum of squares.
- **xtol** (*float*) – Relative error desired in the approximate solution.
- **maxcyc** (*int*) – The maximum number of cycles of refinement to execute, if -1 refine until other limits are met (ftol, xtol)
- **Print** (*bool*) – True for printing results (residuals & times) by cycle

### Returns

(*x, cov\_x, infodict*) where

- **x** : ndarray The solution (or the result of the last iteration for an unsuccessful call).
- **cov\_x** : ndarray Uses the fjac and ipvt optional outputs to construct an estimate of the jacobian around the solution. *None* if a singular matrix encountered (indicates very flat curvature in some direction). This matrix must be multiplied by the residual standard deviation to get the covariance of the parameter estimates – see *curve\_fit*.
- **infodict** : dict a dictionary of optional outputs with the keys:
  - ‘fvec’ : the function evaluated at the output
  - ‘num cyc’:
  - ‘nfev’:
  - ‘lamMax’:
  - ‘psing’:

GSASIImath.**Modulation** (*waveTypes, H, FSSdata, XSSdata, USSdata, Mast*)

H: array nRefBlk x ops X hkl FSSdata: array 2 x atoms x waves (sin,cos terms) XSSdata: array 2x3 x atoms X waves (sin,cos terms) USSdata: array 2x6 x atoms X waves (sin,cos terms) Mast: array orthogonalization matrix for Uij

GSASIImath.**ModulationDerv** (*waveTypes, H, FSSdata, XSSdata, USSdata, Mast*)

H: array ops X hkl FSSdata: array 2 x atoms x waves (sin,cos terms) XSSdata: array 2x3 x atoms X waves (sin,cos terms) USSdata: array 2x6 x atoms X waves (sin,cos terms) Mast: array orthogonalization matrix for Uij

GSASIImath.**OmitMap** (*data, reflDict, pgbars=None*)

default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**PeaksEquiv** (*data, Ind*)

Find the equivalent map peaks for those selected. Works on the contents of data['Map Peaks'].

**Parameters**

- **data** – the phase data structure
- **Ind** (*list*) – list of selected peak indices

**Returns** augmented list of peaks including those related by symmetry to the ones in Ind

GSASIImath.**PeaksUnique** (*data, Ind*)

Finds the symmetry unique set of peaks from those selected. Works on the contents of data['Map Peaks'].

**Parameters**

- **data** – the phase data structure
- **Ind** (*list*) – list of selected peak indices

**Returns** the list of symmetry unique peaks from among those given in Ind

GSASIImath.**Q2AV** (*Q*)

convert quaternion to angle (radians 0-2pi) & normalized vector  $q=r+ai+bj+ck$

GSASIImath.**Q2AVdeg** (*Q*)

convert quaternion to angle (degrees 0-360) & normalized vector  $q=r+ai+bj+ck$

GSASIImath.**Q2Mat** (*Q*)

make rotation matrix from quaternion  $q=r+ai+bj+ck$

GSASIImath.**RotateRBXYZ** (*Bmat, Cart, oriQ*)

rotate & transform cartesian coordinates to crystallographic ones no translation applied. To be used for numerical derivatives

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**SSChargeFlip** (*data, reflDict, pgbars*)

default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**SearchMap** (*generalData, drawingData, Neg=False*)

Does a search of a density map for peaks meeting the criterion of peak height is greater than mapData['cutOff']/100 of mapData['rhoMax'] where mapData is data['General']['mapData']; the map is also in mapData.

**Parameters**

- **generalData** – the phase data structure; includes the map
- **drawingData** – the drawing data structure
- **Neg** – if True then search for negative peaks (i.e. H-atoms & neutron data)

**Returns**

(peaks,mags,dzeros) where

- **peaks** : ndarray x,y,z positions of the peaks found in the map
- **mags** : ndarray the magnitudes of the peaks
- **dzeros** : ndarray the distance of the peaks from the unit cell origin
- **dcent** : ndarray the distance of the peaks from the unit cell center

GSASIImath.**SetMolCent** (*model, RBData*)

default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**TLS2Uij** (*xyz, g, Amat, rbObj*)

default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**UpdateMCSAxyz** (*Bmat, MCSA*)

default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**UpdateRBUIJ** (*Bmat, Cart, RBObj*)

default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**UpdateRBXYZ** (*Bmat, RBObj, RBData, RBType*)

default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**ValEsd** (*value, esd=0, nTZ=False*)

Format a floating point number with a given level of precision or with in crystallographic format with a “esd”, as value(esd). If esd is negative the number is formatted with the level of significant figures appropriate if abs(esd) were the esd, but the esd is not included. if the esd is zero, approximately 6 significant figures are printed. nTZ=True causes “extra” zeros to be removed after the decimal place. for example:

- “1.235(3)” for value=1.2346 & esd=0.003

- “1.235(3)e4” for value=12346. & esd=30
- “1.235(3)e6” for value=0.12346e7 & esd=3000
- “1.235” for value=1.2346 & esd=-0.003
- “1.240” for value=1.2395 & esd=-0.003
- “1.24” for value=1.2395 & esd=-0.003 with nTZ=True
- “1.23460” for value=1.2346 & esd=0.0

**Parameters**

- **value** (*float*) – number to be formatted
- **esd** (*float*) – uncertainty or if esd < 0, specifies level of precision to be shown e.g. esd=-0.01 gives 2 places beyond decimal
- **nTZ** (*bool*) – True to remove trailing zeros (default is False)

**Returns** value(esd) or value as a string

GSASIImath.**Vol2Den** (*Elements, volume*)

converts volume to density

**Parameters**

- **Elements** (*dict*) – elements in molecular formula; each must contain Num: number of atoms in formula Mass: at. wt.
- **volume** (*float*) – molecular volume in A<sup>3</sup>

**Returns** float density: material density in gm/cm<sup>3</sup>

GSASIImath.**XScattDen** (*Elements, vol, wave=0.0*)

Estimate X-ray scattering density from molecular formula & volume; ignores valence, but includes anomalous effects

**Parameters**

- **Elements** (*dict*) – elements in molecular formula; each element must contain Num: number of atoms in formula Z: atomic number
- **vol** (*float*) – molecular volume in A<sup>3</sup>
- **wave** (*float*) – optional wavelength in A

**Returns** float rho: scattering density in 10<sup>10</sup>cm<sup>-2</sup>; if wave > 0 the includes f' contribution

**Returns** float mu: if wave>0 absorption coeff in cm<sup>-1</sup> ; otherwise 0

GSASIImath.**adjHKLmax** (*SGData, Hmax*)

default doc string

**Parameters** *name* (*type*) – description

**Returns** type name: description

GSASIImath.**anneal** (*func, x0, args=(), schedule='fast', full\_output=0, T0=None, Tf=1e-12, max\_eval=None, maxaccept=None, maxiter=400, boltzmann=1.0, learn\_rate=0.5, feqs=1e-06, quench=1.0, m=1.0, n=1.0, lower=-100, upper=100, dwell=50, slope=0.9, ranStart=False, ranRange=0.1, autoRan=False, dlq=None*)

Minimize a function using simulated annealing.

Schedule is a schedule class implementing the annealing schedule. Available ones are ‘fast’, ‘cauchy’, ‘boltzmann’

**Parameters**

- **func** (*callable*) –  $f(x, *args)$  Function to be optimized.
- **x0** (*ndarray*) – Initial guess.
- **args** (*tuple*) – Extra parameters to *func*.
- **schedule** (*base\_schedule*) – Annealing schedule to use (a class).
- **full\_output** (*bool*) – Whether to return optional outputs.
- **T0** (*float*) – Initial Temperature (estimated as 1.2 times the largest cost-function deviation over random points in the range).
- **Tf** (*float*) – Final goal temperature.
- **maxeval** (*int*) – Maximum function evaluations.
- **maxaccept** (*int*) – Maximum changes to accept.
- **maxiter** (*int*) – Maximum cooling iterations.
- **learn\_rate** (*float*) – Scale constant for adjusting guesses.
- **boltzmann** (*float*) – Boltzmann constant in acceptance test (increase for less stringent test at each temperature).
- **feps** (*float*) – Stopping relative error tolerance for the function value in last four coolings.
- **quench,m,n** (*float*) – Parameters to alter fast\_sa schedule.
- **lower,upper** (*float/ndarray*) – Lower and upper bounds on  $x$ .
- **dwell** (*int*) – The number of times to search the space at each temperature.
- **slope** (*float*) – Parameter for log schedule
- **ranStart=False** (*bool*) – True for set 10% of ranges about  $x$

**Returns**

(*xmin*, *Jmin*, *T*, *feval*, *iters*, *accept*, *retval*) where

- *xmin* (*ndarray*): Point giving smallest value found.
- *Jmin* (*float*): Minimum value of function found.
- *T* (*float*): Final temperature.
- *feval* (*int*): Number of function evaluations.
- *iters* (*int*): Number of cooling iterations.
- *accept* (*int*): Number of tests accepted.
- *retval* (*int*): Flag indicating stopping condition:
  - 0: Points no longer changing
  - 1: Cooled to final temperature
  - 2: Maximum function evaluations
  - 3: Maximum cooling iterations reached
  - 4: Maximum accepted query locations reached
  - 5: Final point not the minimum amongst encountered points

*Notes:* Simulated annealing is a random algorithm which uses no derivative information from the function being optimized. In practice it has been more useful in discrete optimization than continuous optimization, as there are usually better algorithms for continuous optimization problems.

Some experimentation by trying the difference temperature schedules and altering their parameters is likely required to obtain good performance.

The randomness in the algorithm comes from random sampling in numpy. To obtain the same results you can call `numpy.random.seed` with the same seed immediately before calling `scipy.optimize.anneal`.

We give a brief description of how the three temperature schedules generate new points and vary their temperature. Temperatures are only updated with iterations in the outer loop. The inner loop is over `xrange(dwell)`, and new points are generated for every iteration in the inner loop. (Though whether the proposed new points are accepted is probabilistic.)

For readability, let `d` denote the dimension of the inputs to `func`. Also, let `x_old` denote the previous state, and `k` denote the iteration number of the outer loop. All other variables not defined below are input variables to `scipy.optimize.anneal` itself.

In the ‘fast’ schedule the updates are

```
u ~ Uniform(0, 1, size=d)
y = sgn(u - 0.5) * T * ((1 + 1/T)**abs(2u-1) - 1.0)
xc = y * (upper - lower)
x_new = x_old + xc

c = n * exp(-n * quench)
T_new = T0 * exp(-c * k**quench)
```

In the ‘cauchy’ schedule the updates are

```
u ~ Uniform(-pi/2, pi/2, size=d)
xc = learn_rate * T * tan(u)
x_new = x_old + xc

T_new = T0 / (1+k)
```

In the ‘boltzmann’ schedule the updates are

```
std = minimum( sqrt(T) * ones(d), (upper-lower) / (3*learn_rate) )
y ~ Normal(0, std, size=d)
x_new = x_old + learn_rate * y

T_new = T0 / log(1+k)
```

`GSASIImath.calcRamaEnergy(phi, psi, Coeff=[ ])`

Computes pseudo potential energy from a pair of torsion angles and a numerical description of the potential energy surface. Used to create penalty function in LS refinement:  $Eval(\phi, \psi) = C[0] * exp(-V/1000)$

where  $V = -C[3] * (\phi - C[1])^2 - C[4] * (\psi - C[2])^2 - 2 * (\phi - C[1]) * (\psi - C[2])$

#### Parameters

- **phi** (*float*) – first torsion angle ( $\phi$ )
- **psi** (*float*) – second torsion angle ( $\psi$ )
- **Coeff** (*list*) – pseudo potential coefficients

**Returns** list (sum, Eval): pseudo-potential difference from minimum & value; sum is used for penalty function.

GSASIImath.**calcTorsionEnergy** (*TOR*, *Coeff*=[ ])  
 default doc string

**Parameters** *name* (*type*) – description

**Returns** type name: description

GSASIImath.**findOffset** (*SGData*, *A*, *Fhkl*)  
 default doc string

**Parameters** *name* (*type*) – description

**Returns** type name: description

GSASIImath.**findSSOffset** (*SGData*, *SSGData*, *A*, *Fhklm*)  
 default doc string

**Parameters** *name* (*type*) – description

**Returns** type name: description

GSASIImath.**getAngSig** (*VA*, *VB*, *Amat*, *SGData*, *covData*={})  
 default doc string

**Parameters** *name* (*type*) – description

**Returns** type name: description

GSASIImath.**getAtomXYZ** (*atoms*, *cx*)  
 default doc string

**Parameters** *name* (*type*) – description

**Returns** type name: description

GSASIImath.**getCWgam** (*ins*, *pos*)  
 get CW peak profile gamma

**Parameters**

- **ins** (*dict*) – instrument parameters with at least ‘X’ & ‘Y’ as values only
- **pos** (*float*) – 2-theta of peak

**Returns** float getCWgam: peak gamma

GSASIImath.**getCWgamDeriv** (*pos*)  
 get derivatives of CW peak profile gamma wrt X & Y

**Parameters** *pos* (*float*) – 2-theta of peak

**Returns** list getCWgamDeriv: d(gam)/dX & d(gam)/dY

GSASIImath.**getCWsig** (*ins*, *pos*)  
 get CW peak profile sigma

**Parameters**

- **ins** (*dict*) – instrument parameters with at least ‘U’, ‘V’, & ‘W’ as values only
- **pos** (*float*) – 2-theta of peak

**Returns** float getCWsig: peak sigma

GSASIImath.**getCWsigDeriv** (*pos*)  
 get derivatives of CW peak profile sigma wrt U, V, & W

**Parameters** *pos* (*float*) – 2-theta of peak



**Returns** list getCWsigDeriv:  $d(\text{sig})/dU$ ,  $d(\text{sig})/dV$  &  $d(\text{sig})/dW$

GSASIImath.**getDensity** (*generalData*)

calculate crystal structure density

**Parameters** **generalData** (*dict*) – The General dictionary in Phase

**Returns** float density: crystal density in  $\text{gm}/\text{cm}^3$

GSASIImath.**getDistDerv** (*Oxyz, Txyz, Amat, Tunit, Top, SGData*)

default doc string

**Parameters** **name** (*type*) – description

**Returns** type name: description

GSASIImath.**getMass** (*generalData*)

Computes mass of unit cell contents

**Parameters** **generalData** (*dict*) – The General dictionary in Phase

**Returns** float mass: Crystal unit cell mass in AMU.

GSASIImath.**getRamaDeriv** (*XYZ, Amat, Coeff*)

Computes numerical derivatives of torsion angle pair pseudo potential with respect of crystallographic atom coordinates of the 5 atom sequence

**Parameters**

- **XYZ** (*narray*) – crystallographic coordinates of 5 atoms
- **Amat** (*narray*) – crystal to cartesian transformation matrix
- **Coeff** (*list*) – pseudo potential coefficients

**Returns** list (deriv) derivatives of pseudopotential with respect to 5 atom crystallographic xyz coordinates.

GSASIImath.**getRestAngle** (*XYZ, Amat*)

default doc string

**Parameters** **name** (*type*) – description

**Returns** type name: description

GSASIImath.**getRestChiral** (*XYZ, Amat*)

default doc string

**Parameters** **name** (*type*) – description

**Returns** type name: description

GSASIImath.**getRestDeriv** (*Func, XYZ, Amat, ops, SGData*)

default doc string

**Parameters** **name** (*type*) – description

**Returns** type name: description

GSASIImath.**getRestDist** (*XYZ, Amat*)

default doc string

**Parameters** **name** (*type*) – description

**Returns** type name: description

GSASIImath.**getRestPlane** (*XYZ, Amat*)

default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**getRestPolefig** (*ODFln, SamSym, Grid*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**getRestPolefigDerv** (*HKL, Grid, SHCoeff*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**getRestRama** (*XYZ, Amat*)  
Computes a pair of torsion angles in a 5 atom string

**Parameters**

- **XYZ** (*narray*) – crystallographic coordinates of 5 atoms
- **Amat** (*narray*) – crystal to cartesian transformation matrix

**Returns** list (phi,psi) two torsion angles in degrees

GSASIImath.**getRestTorsion** (*XYZ, Amat*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**getRho** (*xyz, mapData*)  
get scattering density at a point by 8-point interpolation param xyz: coordinate to be probed param: mapData:  
dict of map data

**Returns** density at xyz

GSASIImath.**getSyXYZ** (*XYZ, ops, SGData*)  
default doc string

**Parameters** *name (type)* – description

**Returns** type name: description

GSASIImath.**getTOFalpha** (*ins, dsp*)  
get TOF peak profile alpha

**Parameters**

- **ins** (*dict*) – instrument parameters with at least ‘alpha’ as values only
- **dsp** (*float*) – d-spacing of peak

**Returns** float getTOFalpha: peak alpha

GSASIImath.**getTOFalphaDeriv** (*dsp*)  
get derivatives of TOF peak profile beta wrt alpha

**Parameters** **dsp** (*float*) – d-spacing of peak

**Returns** float getTOFalphaDeriv: d(alp)/d(alpha)

GSASIImath.**getTOFbeta** (*ins, dsp*)  
 get TOF peak profile beta

**Parameters**

- **ins** (*dict*) – instrument parameters with at least ‘beat-0’, ‘beta-1’ & ‘beta-q’ as values only
- **dsp** (*float*) – d-spacing of peak

**Returns** float getTOFbeta: peak beat

GSASIImath.**getTOFbetaDeriv** (*dsp*)  
 get derivatives of TOF peak profile beta wrt beta-0, beta-1, & beat-q

**Parameters** **dsp** (*float*) – d-spacing of peak

**Returns** list getTOFbetaDeriv: d(beta)/d(beat-0), d(beta)/d(beta-1) & d(beta)/d(beta-q)

GSASIImath.**getTOFgamma** (*ins, dsp*)  
 get TOF peak profile gamma

**Parameters**

- **ins** (*dict*) – instrument parameters with at least ‘X’ & ‘Y’ as values only
- **dsp** (*float*) – d-spacing of peak

**Returns** float getTOFgamma: peak gamma

GSASIImath.**getTOFgammaDeriv** (*dsp*)  
 get derivatives of TOF peak profile gamma wrt X & Y

**Parameters** **dsp** (*float*) – d-spacing of peak

**Returns** list getTOFgammaDeriv: d(gam)/dX & d(gam)/dY

GSASIImath.**getTOFsig** (*ins, dsp*)  
 get TOF peak profile sigma

**Parameters**

- **ins** (*dict*) – instrument parameters with at least ‘sig-0’, ‘sig-1’ & ‘sig-q’ as values only
- **dsp** (*float*) – d-spacing of peak

**Returns** float getTOFsig: peak sigma

GSASIImath.**getTOFsigDeriv** (*dsp*)  
 get derivatives of TOF peak profile gamma wrt sig-0, sig-1, & sig-q

**Parameters** **dsp** (*float*) – d-spacing of peak

**Returns** list getTOFsigDeriv: d(sig0)/d(sig-0), d(sig)/d(sig-1) & d(sig)/d(sig-q)

GSASIImath.**getTorsionDeriv** (*XYZ, Amat, Coeff*)  
 default doc string

**Parameters** **name** (*type*) – description

**Returns** type name: description

GSASIImath.**getVCov** (*varyNames, varyList, covMatrix*)  
 obtain variance-covariance terms for a set of variables. NB: the varyList and covMatrix were saved by the last least squares refinement so they must match.

**Parameters**

- **varyNames** (*list*) – variable names to find v-cov matrix for

- **varyList** (*list*) – full list of all variables in v-cov matrix
- **covMatrix** (*npararray*) – full variance-covariance matrix from the last least squares refinement

**Returns** npararray vcov: variance-covariance matrix for the variables given in varyNames

GSASIImath.**getWave** (*Parms*)

returns wavelength from Instrument parameters dictionary

**Parameters** **Parms** (*dict*) – Instrument parameters; must contain: Lam: single wavelength or Lam1: Ka1 radiation wavelength

**Returns** float wave: wavelength

GSASIImath.**invQ** (*Q*)

get inverse of quaternion  $q=r+ai+bj+ck$ ;  $q^* = r-ai-bj-ck$

GSASIImath.**makeQuat** (*A, B, C*)

Make quaternion from rotation of A vector to B vector about C axis

**Parameters** **A,B,C** (*np.array*) – Cartesian 3-vectors

**Returns** quaternion & rotation angle in radians  $q=r+ai+bj+ck$

GSASIImath.**mcsaSearch** (*data, RBdata, reflType, reflData, covData, pgbars*)

default doc string

**Parameters** **name** (*type*) – description

**Returns** type name: description

GSASIImath.**normQ** (*QA*)

get length of quaternion & normalize it  $q=r+ai+bj+ck$

GSASIImath.**printRho** (*SGLaue, rho, rhoMax*)

default doc string

**Parameters** **name** (*type*) – description

**Returns** type name: description

GSASIImath.**prodQQ** (*QA, QB*)

Grassman quaternion product QA,QB quaternions;  $q=r+ai+bj+ck$

GSASIImath.**prodQVQ** (*Q, V*)

compute the quaternion vector rotation  $qvq^{-1} = v'$   $q=r+ai+bj+ck$

GSASIImath.**randomAVdeg** (*r0, r1, r2, r3*)

create random angle (deg),vector from 4 random number in range (-1,1)

GSASIImath.**randomQ** (*r0, r1, r2, r3*)

create random quaternion from 4 random numbers in range (-1,1)

GSASIImath.**setPeakparms** (*Parms, Parms2, pos, mag, ifQ=False, useFit=False*)

set starting peak parameters for single peak fits from plot selection or auto selection

**Parameters**

- **Parms** (*dict*) – instrument parameters dictionary
- **Parms2** (*dict*) – table lookup for TOF profile coefficients
- **pos** (*float*) – peak position in 2-theta, TOF or Q (ifQ=True)
- **mag** (*float*) – peak top magnitude from pick

- **ifQ** (*bool*) – True if pos in Q
- **useFit** (*bool*) – True if use fitted CW Parms values (not defaults)

**Returns** list XY: peak list entry: for CW: [pos,0,mag,1,sig,0,gam,0] for TOF: [pos,0,mag,1,alp,0,bet,0,sig,0,gam,0] NB: mag refinement set by default, all others off

GSASIImath.**sortArray** (*data, pos, reverse=False*)  
 data is a list of items sort by pos in list; reverse if True

GSASIImath.**wavekE** (*wavekE*)  
 Convert wavelength to energy & vise versa  
 :param float waveKe:wavelength in A or energy in kE  
 :returns float waveKe:the other one



## GSASIIINDEX: CELL INDEXING MODULE

Cell indexing program: variation on that of A. Coehlo includes cell refinement from peak positions (not zero as yet)

This needs a bit of refactoring to remove the little bit of GUI code referencing wx

GSASIIindex.**A2values** (*ibrav, A*)  
needs a doc string

GSASIIindex.**DoIndexPeaks** (*peaks, controls, bravais, ifX20=True*)  
needs a doc string

GSASIIindex.**FitHKL** (*ibrav, peaks, A, Pwr*)  
needs a doc string

GSASIIindex.**FitHKLt** (*difC, ibrav, peaks, A, Z, Zref*)  
needs a doc string

GSASIIindex.**FitHKLz** (*wave, ibrav, peaks, A, Z, Zref*)  
needs a doc string

GSASIIindex.**FitHKLzSS** (*wave, ibrav, peaks, A, V, Vref, Z, Zref*)  
needs a doc string

GSASIIindex.**IndexPeaks** (*peaks, HKL*)  
needs a doc string

GSASIIindex.**IndexSSPeaks** (*peaks, HKL*)  
needs a doc string

GSASIIindex.**TestData** ()  
needs a doc string

GSASIIindex.**Values2A** (*ibrav, values*)  
needs a doc string

GSASIIindex.**calc\_M20** (*peaks, HKL, ifX20=True*)  
needs a doc string

GSASIIindex.**calc\_M20SS** (*peaks, HKL*)  
needs a doc string

GSASIIindex.**findBestCell** (*dlg, ncMax, A, Ntries, ibrav, peaks, V1, ifX20=True*)  
needs a doc string

GSASIIindex.**getDmax** (*peaks*)  
needs a doc string

GSASIIindex.**getDmin** (*peaks*)  
needs a doc string

GSASIIindex.**halfCell** (*ibrav, A, peaks*)  
needs a doc string

GSASIIindex.**monoCellReduce** (*ibrav, A*)  
needs a doc string

GSASIIindex.**oddPeak** (*indx, peaks*)  
needs a doc string

GSASIIindex.**ran2axis** (*k, N*)  
needs a doc string

GSASIIindex.**ranAbyR** (*Bravais, A, k, N, ranFunc*)  
needs a doc string

GSASIIindex.**ranAbyV** (*Bravais, dmin, dmax, V*)  
needs a doc string

GSASIIindex.**ranaxis** (*dmin, dmax*)  
needs a doc string

GSASIIindex.**rancell** (*Bravais, dmin, dmax*)  
needs a doc string

GSASIIindex.**refinePeaks** (*peaks, ibrav, A, ifX20=True*)  
needs a doc string

GSASIIindex.**refinePeaksT** (*peaks, difC, ibrav, A, Zero, ZeroRef*)  
needs a doc string

GSASIIindex.**refinePeaksZ** (*peaks, wave, ibrav, A, Zero, ZeroRef*)  
needs a doc string

GSASIIindex.**refinePeaksZSS** (*peaks, wave, Inst, SGData, SSGData, maxH, ibrav, A, vec, vecRef, Zero, ZeroRef*)  
needs a doc string

GSASIIindex.**rotOrthoA** (*A*)  
needs a doc string

GSASIIindex.**scaleAbyV** (*A, V*)  
needs a doc string

GSASIIindex.**sortM20** (*cells*)  
needs a doc string

GSASIIindex.**swapMonoA** (*A*)  
needs a doc string



## GSASII PLOT: PLOTTING ROUTINES

**class** GSASIIplot . **G2Plot3D** (*parent, id=-1, dpi=None, \*\*kwargs*)  
Creates a 3D Matplotlib plot in the GSAS-II graphics window

**class** GSASIIplot . **G2PlotMpl** (*parent, id=-1, dpi=None, \*\*kwargs*)  
Creates a Matplotlib 2-D plot in the GSAS-II graphics window

**class** GSASIIplot . **G2PlotNoteBook** (*parent, id=-1*)  
create a tabbed window for GSAS-II graphics

**Delete** (*name*)  
delete a tabbed page

**OnNotebookKey** (*event*)  
Called when a keystroke event gets picked up by the notebook window rather the child. This is not expected, but somehow it does sometimes on the Mac and perhaps Linux.

Assume that the page associated with the currently displayed tab has a child, .canvas; give that child the focus and pass it the event.

**OnPageChanged** (*event*)  
respond to someone pressing a tab on the plot window

**Rename** (*oldName, newName*)  
rename a tab

**add3D** (*name=''*)  
Add a tabbed page with a 3D plot

**addMpl** (*name=''*)  
Add a tabbed page with a matplotlib plot

**addOgl** (*name=''*)  
Add a tabbed page with an OpenGL plot

**clear** ()  
clear all pages from plot window

**class** GSASIIplot . **G2PlotOgl** (*parent, id=-1, dpi=None, \*\*kwargs*)  
Creates an OpenGL plot in the GSAS-II graphics window

**class** GSASIIplot . **GSASIItoolbar** (*plotCanvas*)  
Override the matplotlib toolbar so we can add more icons

**OnArrow** (*event*)  
reposition limits to scan or zoom by button press

**OnHelp** (*event*)  
Respond to press of help button on plot toolbar

**OnKey** (*event*)

Provide user with list of keystrokes defined for plot as well as an alternate way to access the same functionality

GSASIIplot.**OnStartMask** (*G2frame*)

Initiate the start of a Frame or Polygon map

**Parameters**

- **G2frame** (*wx.Frame*) – The main GSAS-II tree “window”
- **eventkey** (*str*) – a single letter (‘f’ or ‘p’) that determines what type of mask is created.

GSASIIplot.**OnStartNewDzero** (*G2frame*)

Initiate the start of adding a new d-zero to a strain data set

**Parameters**

- **G2frame** (*wx.Frame*) – The main GSAS-II tree “window”
- **eventkey** (*str*) – a single letter (‘a’) that triggers the addition of a d-zero.

GSASIIplot.**Plot3DSngl** (*G2frame, newPlot=False, Data=None, hklRef=None, Title=False*)

3D Structure factor plotting package - displays reflections as rings proportional to F, F\*\*2, etc. as requested as 3D array

GSASIIplot.**PlotCalib** (*G2frame, Inst, XY, Sigs, newPlot=False*)

plot of CW or TOF peak calibration

GSASIIplot.**PlotCovariance** (*G2frame, Data*)

needs a doc string

GSASIIplot.**PlotDeltSig** (*G2frame, kind*)

needs a doc string

GSASIIplot.**PlotExposedImage** (*G2frame, newPlot=False, event=None*)

General access module for 2D image plotting

GSASIIplot.**PlotISFG** (*G2frame, newPlot=False, type=''*)

Plotting package for PDF analysis; displays I(q), S(q), F(q) and G(r) as single or multiple plots with waterfall and contour plots as options

GSASIIplot.**PlotImage** (*G2frame, newPlot=False, event=None, newImage=True*)

Plot of 2D detector images as contoured plot. Also plot calibration ellipses, masks, etc.

GSASIIplot.**PlotIntegration** (*G2frame, newPlot=False, event=None*)

Plot of 2D image after image integration with 2-theta and azimuth as coordinates

GSASIIplot.**PlotPatterns** (*G2frame, newPlot=False, plotType='PWDR'*)

Powder pattern plotting package - displays single or multiple powder patterns as intensity vs 2-theta, q or TOF. Can display multiple patterns as “waterfall plots” or contour plots. Log I plotting available.

GSASIIplot.**PlotPeakWidths** (*G2frame*)

Plotting of instrument broadening terms as function of 2-theta. Seen when “Instrument Parameters” chosen from powder pattern data tree

GSASIIplot.**PlotPowderLines** (*G2frame*)

plotting of powder lines (i.e. no powder pattern) as sticks

GSASIIplot.**PlotRama** (*G2frame, phaseName, Rama, RamaName, Names=[], PhiPsi=[], Coeff=[]*)

needs a doc string

GSASIIplot.**PlotRigidBody** (*G2frame, rbType, AtInfo, rbData, defaults*)

RB plotting package. Can show rigid body structures as balls & sticks

GSASIIplot.**PlotSelectedSequence** (*G2frame, ColumnList, TableGet, SelectX, fitnum=None, fitvals=None*)

Plot a result from a sequential refinement

**Parameters**

- **G2frame** (*wx.Frame*) – The main GSAS-II tree “window”
- **ColumnList** (*list*) – list of int values corresponding to columns selected as y values
- **TableGet** (*function*) – a function that takes a column number as argument and returns the column label, the values and there ESDs (or None)
- **SelectX** (*function*) – a function that returns a selected column number (or None) as the X-axis selection

GSASIIplot.**PlotSizeStrainPO** (*G2frame, data, hist='', Start=False*)

Plot 3D mustrain/size/preferred orientation figure. In this instance data is for a phase

GSASIIplot.**PlotSngl** (*G2frame, newPlot=False, Data=None, hklRef=None, Title=''*)

Structure factor plotting package - displays zone of reflections as rings proportional to F, F\*\*2, etc. as requested

GSASIIplot.**PlotStrain** (*G2frame, data, newPlot=False*)

plot of strain data, used for diagnostic purposes

GSASIIplot.**PlotStructure** (*G2frame, data, firstCall=False*)

Crystal structure plotting package. Can show structures as balls, sticks, lines, thermal motion ellipsoids and polyhedra

GSASIIplot.**PlotTRImage** (*G2frame, tax, tay, taz, newPlot=False*)

a test plot routine - not normally used

GSASIIplot.**PlotTexture** (*G2frame, data, Start=False*)

Pole figure, inverse pole figure plotting. dict generalData contains all phase info needed which is in data

GSASIIplot.**PlotTorsion** (*G2frame, phaseName, Torsion, TorName, Names=[ ], Angles=[ ], Coeff=[ ]*)

needs a doc string

GSASIIplot.**PlotXY** (*G2frame, XY, XY2=None, labelX=None, labelY=None, newPlot=False, Title=''*)

simple plot of xy data, used for diagnostic purposes



## GSASII POWDER CALCULATION MODULE

GSASIIpwd.**Absorb** (*Geometry, MuR, Tth, Phi=0, Psi=0*)

Calculate sample absorption :param str Geometry: one of 'Cylinder','Bragg-Brentano','Tilting Flat Plate in transmission','Fixed flat plate' :param float MuR: absorption coeff \* sample thickness/2 or radius :param Tth: 2-theta scattering angle - can be numpy array :param float Phi: flat plate tilt angle - future :param float Psi: flat plate tilt axis - future

GSASIIpwd.**AbsorbDerv** (*Geometry, MuR, Tth, Phi=0, Psi=0*)

needs a doc string

GSASIIpwd.**CalcPDF** (*data, inst, xydata*)

needs a doc string

GSASIIpwd.**Dict2Values** (*parmdict, varylist*)

Use before call to leastsq to setup list of values for the parameters in parmdict, as selected by key in varylist

GSASIIpwd.**DoPeakFit** (*FitPgm, Peaks, Background, Limits, Inst, Inst2, data, prevVaryList=[ ], oneCycle=False, controls=None, dlg=None*)

Called to perform a peak fit, refining the selected items in the peak table as well as selected items in the background.

### Parameters

- **FitPgm** (*str*) – type of fit to perform. At present “LSQ” is the only option that works
- **Peaks** (*list*) – a list of peaks. Each peak entry is a list with 8 values: four values followed by a refine flag where the values are: position, intensity, sigma (Gaussian width) and gamma (Lorentzian width). From the Histogram/”Peak List” tree entry, dict item “peaks”
- **Background** (*list*) – describes the background. List with two items. Item 0 specifies a background model and coefficients. Item 1 is a dict. From the Histogram/Background tree entry.
- **Limits** (*list*) – min and max x-value to use
- **Inst** (*dict*) – Instrument parameters
- **Inst2** (*dict*) – more Instrument parameters
- **data** (*numpy.array*) – a 5xn array. data[0] is the x-values, data[1] is the y-values, data[2] are weight values, data[3], [4] and [5] are calc, background and difference intensities, respectively.
- **prevVaryList** (*list*) – Used in sequential refinements to override the variable list. Defaults as an empty list.
- **oneCycle** (*bool*) – True if only one cycle of fitting should be performed
- **controls** (*dict*) – a dict specifying two values, Ftol = controls['min dM/M'] and derivType = controls['deriv type']. If None default values are used.

- **dlg** (*wx.Dialog*) – A dialog box that is updated with progress from the fit. Defaults to None, which means no updates are done.

GSASIIpwd.**GetAsfMean** (*ElList, Sthl2*)

Calculate various scattering factor terms for PDF calcs

**Parameters**

- **ElList** (*dict*) – element dictionary contains scattering factor coefficients, etc.
- **Sthl2** (*np.array*) – numpy array of sin theta/lambda squared values

**Returns** mean(f<sup>2</sup>), mean(f)<sup>2</sup>, mean(compton)

GSASIIpwd.**GetNumDensity** (*ElList, Vol*)

needs a doc string

GSASIIpwd.**LorchWeight** (*Q*)

needs a doc string

GSASIIpwd.**Oblique** (*ObCoeff, Tth*)

currently assumes detector is normal to beam

GSASIIpwd.**Polarization** (*Pola, Tth, Azm=0.0*)

Calculate angle dependent x-ray polarization correction (not scaled correctly!)

**Parameters**

- **Pola** – polarization coefficient e.g 1.0 fully polarized, 0.5 unpolarized
- **Azm** – azimuthal angle e.g. 0.0 in plane of polarization
- **Tth** – 2-theta scattering angle - can be numpy array which (if either) of these is “right”?

**Returns** (pola, dpdPola) \* pola = ((1-Pola)\*npcosd(Azm)\*\*2+Pola\*nsind(Azm)\*\*2)\*npcosd(Tth)\*\*2+(1-Pola)\*nsind(Azm)\*\*2+Pola\*npcosd(Azm)\*\*2 \* dpdPola: derivative needed for least squares

GSASIIpwd.**Ruland** (*RulCoff, wave, Q, Compton*)

needs a doc string

GSASIIpwd.**SetBackgroundParms** (*Background*)

needs a doc string

GSASIIpwd.**SurfaceRough** (*SRA, SRB, Tth*)

Suortti (J. Appl. Cryst, 5,325-331, 1972) surface roughness correction :param float SRA: Suortti surface roughness parameter :param float SRB: Suortti surface roughness parameter :param float Tth: 2-theta(deg) - can be numpy array

GSASIIpwd.**SurfaceRoughDerv** (*SRA, SRB, Tth*)

Suortti surface roughness correction derivatives :param float SRA: Suortti surface roughness parameter (dimensionless) :param float SRB: Suortti surface roughness parameter (dimensionless) :param float Tth: 2-theta(deg) - can be numpy array :return list: [dydSRA,dydSRB] derivatives to be used for intensity derivative

GSASIIpwd.**TestData** ()

needs a doc string

GSASIIpwd.**Transmission** (*Geometry, Abs, Diam*)

Calculate sample transmission

**Parameters**

- **Geometry** (*str*) – one of ‘Cylinder’,‘Bragg-Brentano’,‘Tilting flat plate in transmission’,‘Fixed flat plate’

- **Abs** (*float*) – absorption coeff in cm-1
- **Diam** (*float*) – sample thickness/diameter in mm

GSASIIpwd.**Values2Dict** (*parmdict, varylist, values*)

Use after call to `leastsq` to update the parameter dictionary with values corresponding to keys in `varylist`

GSASIIpwd.**calcIncident** (*Iparm, xdata*)

needs a doc string

**class** GSASIIpwd.**cauchy\_gen** (*momtype=1, a=None, b=None, xtol=1e-14, badvalue=None, name=None, longname=None, shapes=None, extradoc=None, seed=None*)

needs a doc string

GSASIIpwd.**ellipseSize** (*H, Sij, GB*)

needs a doc string

GSASIIpwd.**ellipseSizeDerv** (*H, Sij, GB*)

needs a doc string

GSASIIpwd.**factorize** (*num*)

Provide prime number factors for integer `num` :returns: dictionary of prime factors (keys) & power for each (data)

**class** GSASIIpwd.**fcjde\_gen** (*momtype=1, a=None, b=None, xtol=1e-14, badvalue=None, name=None, longname=None, shapes=None, extradoc=None, seed=None*)

Finger-Cox-Jephcoat D(2phi,2th) function for S/L = H/L Ref: J. Appl. Cryst. (1994) 27, 892-900.

#### Parameters

- **x** – array -1 to 1
- **t** – 2-theta position of peak
- **s** – sum(S/L,H/L); S: sample height, H: detector opening, L: sample to detector opening distance
- **dx** – 2-theta step size in deg

#### Returns

for `fcj.pdf`

- $T = x * dx + t$
- $s = S/L + H/L$
- if  $x < 0$ :

$$fcj.pdf = [1/\sqrt{(\cos(T)**2/\cos(t)**2)-1} - 1/s] / |\cos(T)|$$

- if  $x \geq 0$ : `fcj.pdf = 0`

GSASIIpwd.**getBackground** (*px, parmDict, bakType, dataType, xdata*)

needs a doc string

GSASIIpwd.**getBackgroundDerv** (*hfx, parmDict, bakType, dataType, xdata*)

needs a doc string

GSASIIpwd.**getEpsVoigt** (*pos, alp, bet, sig, gam, xdata*)

needs a doc string

GSASIIpwd.**getFCJVoigt** (*pos, intens, sig, gam, shl, xdata*)

needs a doc string

GSASIIpwd.**getFCJVoigt3** (*pos, sig, gam, shl, xdata*)  
needs a doc string

GSASIIpwd.**getFWHM** (*pos, Inst*)  
 $1.17741 * \pi / 180 = \text{sqrt}(8 \ln 2) / (2 \pi / 180)$

GSASIIpwd.**getHKLMpeak** (*dmin, Inst, SGData, SSGData, Vec, maxH, A*)  
needs a doc string

GSASIIpwd.**getHKLpeak** (*dmin, SGData, A, Inst=None*)  
needs a doc string

GSASIIpwd.**getPeakProfile** (*dataType, parmDict, xdata, varyList, bakType*)  
needs a doc string

GSASIIpwd.**getPeakProfileDerv** (*dataType, parmDict, xdata, varyList, bakType*)  
needs a doc string

GSASIIpwd.**getPsVoigt** (*pos, sig, gam, xdata*)  
needs a doc string

GSASIIpwd.**getWidthsCW** (*pos, sig, gam, shl*)  
Compute the peak widths used for computing the range of a peak for constant wavelength data. On low-angle side, 10 FWHM are used, on high-angle side 15 are used (for peaks above 90 deg, these are reversed.)

GSASIIpwd.**getWidthsTOF** (*pos, alp, bet, sig, gam*)  
needs a doc string

GSASIIpwd.**getdEpsVoigt** (*pos, alp, bet, sig, gam, xdata*)  
needs a doc string

GSASIIpwd.**getdFCJVoigt3** (*pos, sig, gam, shl, xdata*)  
needs a doc string

GSASIIpwd.**getdPsVoigt** (*pos, sig, gam, xdata*)  
needs a doc string

GSASIIpwd.**getgamFW** (*g, s*)  
needs a doc string

GSASIIpwd.**makeFFTsizeList** (*nmin=1, nmax=1023, thresh=15*)  
Provide list of optimal data sizes for FFT calculations

**Parameters**

- **nmin** (*int*) – minimum data size  $\geq 1$
- **nmax** (*int*) – maximum data size  $> \text{nmin}$
- **thresh** (*int*) – maximum prime factor allowed

**Returns** list of data sizes where the maximum prime factor is  $< \text{thresh}$

**class** GSASIIpwd.**norm\_gen** (*momtype=1, a=None, b=None, xtol=1e-14, badvalue=None, name=None, longname=None, shapes=None, extradoc=None, seed=None*)  
needs a doc string



## GSAS-II SMALL ANGLE SCATTERING SUBMODULES

### 13.1 GSASII small angle calculation module

GSASIIIsasd.**CylinderARFF** (*Q, R, args*)

Compute form factor for cylinders - can use numpy arrays param float Q: Q value array (A-1) param float R: cylinder radius (A) param array args: [float AR]: cylinder aspect ratio = L/D = L/2R returns float: form factor

GSASIIIsasd.**CylinderARVol** (*R, args*)

Compute cylinder volume for radius & aspect ratio = L/D - numpy array friendly param float: R radius (A) param array args: [float AR]: =L/D=L/2R aspect ratio returns float:volume

GSASIIIsasd.**CylinderDFF** (*Q, L, args*)

Compute form factor for cylinders - can use numpy arrays param float Q: Q value array (A-1) param float L: cylinder half length (A) param array args: [float R]: cylinder radius (A) returns float: form factor

GSASIIIsasd.**CylinderDVol** (*L, args*)

Compute cylinder volume for length & diameter - numpy array friendly param float: L half length (A) param array args: [float D]: diameter (A) returns float:volume (A<sup>3</sup>)

GSASIIIsasd.**CylinderFF** (*Q, R, args*)

Compute form factor for cylinders - can use numpy arrays param float Q: Q value array (A-1) param float R: cylinder radius (A) param array args: [float L]: cylinder length (A) returns float: form factor

GSASIIIsasd.**CylinderVol** (*R, args*)

Compute cylinder volume for radius & length - numpy array friendly param float R: diameter (A) param array args: [float L]: length (A) returns float:volume (A<sup>3</sup>)

GSASIIIsasd.**DiluteSF** (*Q, args=[ ]*)

Default: no structure factor correction for dilute system

GSASIIIsasd.**G\_matrix** (*q, r, contrast, FFfxn, Volfxn, args=()*)

Calculates the response matrix  $G(Q, r)$

#### Parameters

- **q** (*float*) –  $Q$
- **r** (*float*) –  $r$
- **contrast** (*float*) –  $|\Delta\rho|^2$ , the scattering contrast
- **FFfxn** (*function*) – form factor function FF( $q, r, args$ )
- **Volfxn** (*function*) – volume function Vol( $r, args$ )

**Returns float**  $G(Q, r)$

GSASIIIsasd.**GaussCume** (*x, pos, args*)

Standard Normal cumulative distribution - numpy friendly on x axis param float x: independent axis (can be numpy array) param float pos: location of distribution param float scale: width of distribution (sigma) param float shape: not used returns float: Normal cumulative distribution

GSASIIIsasd.**GaussDist** (*x, pos, args*)

Standard Normal distribution - numpy friendly on x axis param float x: independent axis (can be numpy array) param float pos: location of distribution param float scale: width of distribution (sigma) param float shape: not used returns float: Normal distribution

GSASIIIsasd.**HardSpheresSF** (*Q, args*)

Computes structure factor for not dilute monodisperse hard spheres Refs.: PERCUS, YEVICK PHYS. REV. 110 1 (1958), THIELE J. CHEM PHYS. 39 474 (1968), WERTHEIM PHYS. REV. LETT. 47 1462 (1981)

param float Q: Q value array (A-1) param array args: [float R, float VolFrac]: interparticle distance & volume fraction returns numpy array S(Q)

GSASIIIsasd.**IPG** (*datum, sigma, G, Bins, Dbins, IterMax, Qvec=[ ], approach=0.8, Power=-1, report=False*)

An implementation of the Interior-Point Gradient method of Michael Merritt & Yin Zhang, Technical Report TR04-08, Dept. of Comp. and Appl. Math., Rice Univ., Houston, Texas 77005, U.S.A. found on the web at <http://www.caam.rice.edu/caam/trs/2004/TR04-08.pdf> Problem addressed: Total Non-Negative Least Squares (TNNLS) :param float datum[:]: :param float sigma[:]: :param float[][] G: transformation matrix :param int IterMax: :param float Qvec: data positions for Power = 0-4 :param float approach: 0.8 default fitting parameter :param int Power: 0-4 for Q^Power weighting, -1 to use input sigma

GSASIIIsasd.**InterPrecipitatesSF** (*Q, args*)

Computes structure factor for precipitates in a matrix Refs.: E-Wen Huang, Peter K. Liaw, Lionel Porcar, Yun Liu, Yee-Lang Liu, Ji-Jung Kai, and Wei-Ren Chen, APPLIED PHYSICS LETTERS 93, 161904 (2008) R. Giordano, A. Grasso, and J. Teixeira, Phys. Rev. A 43, 6894 1991 param float Q: Q value array (A-1) param array args: [float R, float VolFr]: "radius" & volume fraction returns numpy array S(Q)

GSASIIIsasd.**LSWCume** (*x, pos, args=[ ]*)

Lifshitz-Slyozov-Wagner Ostwald ripening cumulative distribution - numpy friendly on x axis param float x: independent axis (can be numpy array) param float pos: location of distribution param float scale: not used param float shape: not used returns float: LSW cumulative distribution

GSASIIIsasd.**LSWDist** (*x, pos, args=[ ]*)

Lifshitz-Slyozov-Wagner Ostwald ripening distribution - numpy friendly on x axis ref: param float x: independent axis (can be numpy array) param float pos: location of distribution param float scale: not used param float shape: not used returns float: LSW distribution

GSASIIIsasd.**LogNormalCume** (*x, pos, args*)

Standard LogNormal cumulative distribution - numpy friendly on x axis ref: <http://www.itl.nist.gov/div898/handbook/index.htm> 1.3.6.6.9 param float x: independent axis (can be numpy array) param float pos: location of distribution param float scale: width of distribution (sigma) param float shape: shape parameter returns float: LogNormal cumulative distribution

GSASIIIsasd.**LogNormalDist** (*x, pos, args*)

Standard LogNormal distribution - numpy friendly on x axis ref: <http://www.itl.nist.gov/div898/handbook/index.htm> 1.3.6.6.9 param float x: independent axis (can be numpy array) param float pos: location of distribution param float scale: width of distribution (m) param float shape: shape - (sigma of log(LogNormal)) returns float: LogNormal distribution

**exception** GSASIIIsasd.**MaxEntException**

Any exception from this module

GSASIIIsasd.**MaxEnt\_SB** (*datum, sigma, G, base, IterMax, image\_to\_data=None, data\_to\_image=None, report=False*)

do the complete Maximum Entropy algorithm of Skilling and Bryan

**Parameters**

- **datum[]** (*float*) –
- **sigma[]** (*float*) –
- **G** (*float[][]*) – transformation matrix
- **base[]** (*float*) –
- **IterMax** (*int*) –
- **image\_to\_data** (*obj*) – opus function (defaults to opus)
- **data\_to\_image** (*obj*) – tropus function (defaults to tropus)

**Returns** `float[] f(r)dr`

GSASIIIsasd. **SchulzZimmCume** (*x, pos, args*)

Schulz-Zimm cumulative distribution - numpy friendly on x axis param float x: independent axis (can be numpy array) param float pos: location of distribution param float scale: width of distribution (sigma) param float shape: not used returns float: Normal distribution

GSASIIIsasd. **SchulzZimmDist** (*x, pos, args*)

Schulz-Zimm macromolecule distribution - numpy friendly on x axis ref: <http://goldbook.iupac.org/S05502.html> param float x: independent axis (can be numpy array) param float pos: location of distribution param float scale: width of distribution (sigma) param float shape: not used returns float: Schulz-Zimm distribution

GSASIIIsasd. **SphereFF** (*Q, R, args=()*)

Compute hard sphere form factor - can use numpy arrays param float Q: Q value array (usually in A-1) param float R: sphere radius (Usually in A - must match Q-1 units) param array args: ignored returns float: form factors as array as needed

GSASIIIsasd. **SphereVol** (*R, args=()*)

Compute volume of sphere - numpy array friendly param float R: sphere radius param array args: ignored returns float: volume

GSASIIIsasd. **SpheroidFF** (*Q, R, args*)

Compute form factor of cylindrically symmetric ellipsoid (spheroid) - can use numpy arrays for R & AR; will return corresponding numpy array param float Q : Q value array (usually in A-1) param float R: radius along 2 axes of spheroid param array args: [float AR]: aspect ratio so 3rd axis = R\*AR returns float: form factors as array as needed

GSASIIIsasd. **SpheroidVol** (*R, args*)

Compute volume of cylindrically symmetric ellipsoid (spheroid) - numpy array friendly param float R: radius along 2 axes of spheroid param array args: [float AR]: aspect ratio so radius of 3rd axis = R\*AR returns float: volume

GSASIIIsasd. **SquareWellSF** (*Q, args*)

Computes structure factor for not dilute monodisperse hard sphere with a square well potential interaction. Refs.: SHARMA,SHARMA, PHYSICA 89A,(1977),213-

**Parameters**

- **Q** (*float*) – Q value array (A-1)
- **args** (*array*) – [float R, float VolFrac, float depth, float width]: interparticle distance, volume fraction (<0.08), well depth ( $e/kT < 1.5kT$ ), well width

**Returns** numpy array S(Q) well depth > 0 attractive & values outside above limits nonphysical cf. Monte Carlo simulations

GSASIIIsasd.**StickyHardSpheresSF** (*Q, args*)

Computes structure factor for not dilute monodisperse hard spheres Refs.: PERCUS, YEVICK PHYS. REV. 110 1 (1958), THIELE J. CHEM PHYS. 39 474 (1968), WERTHEIM PHYS. REV. LETT. 47 1462 (1981)

param float Q: Q value array (A-1) param array args: [float R, float VolFrac]: sphere radius & volume fraction returns numpy array S(Q)

GSASIIIsasd.**UniDiskFF** (*Q, R, args*)

Compute form factor for unified disk - can use numpy arrays param float Q: Q value array (A-1) param float R: cylinder radius (A) param array args: [float T]: disk thickness (A) returns float: form factor

GSASIIIsasd.**UniDiskVol** (*R, args*)

Compute disk volume for radius & thickness - numpy array friendly param float R: diameter (A) param array args: [float T]: thickness returns float: volume (A<sup>3</sup>)

GSASIIIsasd.**UniRodARFF** (*Q, R, args*)

Compute form factor for unified rod of fixed aspect ratio - can use numpy arrays param float Q: Q value array (A-1) param float R: cylinder radius (A) param array args: [float AR]: cylinder aspect ratio = L/D = L/2R returns float: form factor

GSASIIIsasd.**UniRodARVol** (*R, args*)

Compute rod volume for radius & aspect ratio - numpy array friendly param float R: diameter (A) param array args: [float AR]: =L/D=L/2R aspect ratio returns float: volume (A<sup>3</sup>)

GSASIIIsasd.**UniRodFF** (*Q, R, args*)

Compute form factor for unified rod - can use numpy arrays param float Q: Q value array (A-1) param float R: cylinder radius (A) param array args: [float R]: cylinder radius (A) returns float: form factor

GSASIIIsasd.**UniRodVol** (*R, args*)

Compute cylinder volume for radius & length - numpy array friendly param float R: diameter (A) param array args: [float L]: length (A) returns float: volume (A<sup>3</sup>)

GSASIIIsasd.**UniSphereFF** (*Q, R, args=0*)

Compute form factor for unified sphere - can use numpy arrays param float Q: Q value array (A-1) param float R: cylinder radius (A) param array args: ignored returns float: form factor

GSASIIIsasd.**UniSphereVol** (*R, args=()*)

Compute volume of sphere - numpy array friendly param float R: sphere radius param array args: ignored returns float: volume

GSASIIIsasd.**UniTubeFF** (*Q, R, args*)

Compute form factor for unified tube - can use numpy arrays assumes that core of tube is same as the matrix/solvent so contrast is from tube wall vs matrix param float Q: Q value array (A-1) param float R: cylinder radius (A) param array args: [float L,T]: tube length & wall thickness(A) returns float: form factor

GSASIIIsasd.**UniTubeVol** (*R, args*)

Compute tube volume for radius, length & wall thickness - numpy array friendly param float R: diameter (A) param array args: [float L,T]: tube length & wall thickness(A) returns float: volume (A<sup>3</sup>) of tube wall

GSASIIIsasd.**print\_arr** (*text, a*)

print the contents of an array to the console

GSASIIIsasd.**print\_vec** (*text, a*)

print the contents of a vector to the console

## 13.2 Substances: Define Materials

Defines materials commonly found in small angle & reflectometry experiments. GSASII substances as a dictionary "Substances.Substances" with these materials.

Each entry in “Substances” consists of:

```
'key': {'Elements': {element: {'Num': number in formula}, ...}, 'Density': value, 'Volume':, value}
```

Density & Volume are optional, if one missing it is calculated from the other; if both are missing then Volume is estimated from composition & assuming  $10\text{\AA}^3$  for each atom, Density is calculated from that Volume. See examples below for what is needed.



## *GSAS-II SCRIPTS*

### **14.1 *testDeriv: Check derivative computation***

Use this to check derivatives used in structure least squares refinement against numerical values computed in this script.

To use set `DEBUG=True` in `GSASIIstrMain.py` (line 22, as of version 1110); run the least squares - one cycle is sufficient. Do the “Save Results”; this will write the file `testDeriv.dat` in the local directory.

Then run this program to see plots of derivatives for all parameters refined in the last least squares. Shown will be numerical derivatives generated over all observations (including penalty terms) and the corresponding analytical ones produced in the least squares. They should match.

```
testDeriv.main()  
    Starts main application to compute and plot derivatives
```

### **14.2 *GSASIItestplot: Plotting for testDeriv***

Plotting module used for script `testDeriv`.

```
class GSASIItestplot.Plot (parent, id=-1, dpi=None, **kwargs)  
    Creates a plotting window  
class GSASIItestplot.PlotNotebook (id=-1)  
    creates a Wx application and a plotting notebook
```

### **14.3 *scanCCD: reduce data from scanning CCD***

Quickly prototyped routine for reduction of data from detector described in B.H. Toby, T.J. Madden, M.R. Suchomel, J.D. Baldwin, and R.B. Von Dreele, “A Scanning CCD Detector for Powder Diffraction Measurements”. *Journal of Applied Crystallography*. 46(4): p. 1058-63 (2013).

```
scanCCD.main()  
    starts main application to merge data from scanning CCD
```

### **14.4 *makeMacApp: Create Mac Applet***

This script creates an AppleScript app to launch GSAS-II. The app is created in the directory where the GSAS-II script is located. A softlink to Python is created, but is named GSAS-II, so that GSAS-II shows up as the name of the

app rather than Python in the menu bar, etc. Note that this requires finding an app version of Python (expected name `.../Resources/Python.app/Contents/MacOS/Python` in directory tree of the calling python interpreter).

Run this script with one optional argument, the path to the `GSASII.py`. The script path may be specified relative to the current path or given an absolute path, but will be accessed via an absolute path. If no arguments are supplied, the `GSASII.py` script is assumed to be in the same directory as this file.

```
makeMacApp.AppleScript = ‘(* GSAS-II AppleScript by B. Toby (brian.toby@anl.gov)\n It can launch GSAS-II by double  
    Contains an AppleScript to start GSAS-II launching python and the GSAS-II python script
```

```
makeMacApp.RunPython (image, cmd)  
    Run a command in a python image
```

## 14.5 *unit\_tests: Self-test Module*

A script that can be run to test a series of self-tests in GSAS-II. At present, only modules `GSASIIspc` and `GSASIIlattice` have self-tests.

```
unit_tests.test_GSASIIlattice()
```

Test registered self-tests in `GSASIIlattice`. Takes no input and returns nothing. Throws an Exception if a test fails.

```
unit_tests.test_GSASIIspc()
```

Test registered self-tests in `GSASIIspc`. Takes no input and returns nothing. Throws an Exception if a test fails.



## GSAS-II EXPORT MODULES

Exports are implemented by deriving a class from `GSASIIIO.ExportBaseClass`. Initialization of `self.exporttype` determines the type of export that will be performed ('project', 'phase', 'single', 'powder', 'image', 'map' or (someday) 'pdf') and of `self.multiple` determines if only a single phase, data set, etc. can be exported at a time (when `False`) or more than one can be selected.

Powder export routines may optionally define a `Writer()` method that accepts the histogram tree name as well as a file name to be written. This allows `ExportPowder()` to use the exporter independent of the GUI.

### 15.1 Module `G2export_examples`: Examples

Code to demonstrate how GSAS-II data export routines are created. The classes defined here, `ExportPhaseText`, `ExportSingleText`, `ExportPowderReflText`, and `ExportPowderText` each demonstrate a different type of export. Also see `G2export_map.ExportMapASCII` for an example of a map export.

**class** `G2export_examples.ExportPhaseText` (`G2frame`)

Used to create a text file for a phase

**Parameters** `G2frame` (`wx.Frame`) – reference to main GSAS-II frame

**Exporter** (`event=None`)

Export a phase as a text file

**class** `G2export_examples.ExportPowderReflText` (`G2frame`)

Used to create a text file of reflections from a powder data set

**Parameters** `G2frame` (`wx.Frame`) – reference to main GSAS-II frame

**Exporter** (`event=None`)

Export a set of powder reflections as a text file

**class** `G2export_examples.ExportPowderText` (`G2frame`)

Used to create a text file for a powder data set

**Parameters** `G2frame` (`wx.Frame`) – reference to main GSAS-II frame

**Exporter** (`event=None`)

Export a set of powder data as a text file

**class** `G2export_examples.ExportSingleText` (`G2frame`)

Used to create a text file with single crystal reflection data

**Parameters** `G2frame` (`wx.Frame`) – reference to main GSAS-II frame

**Exporter** (`event=None`)

Export a set of single crystal data as a text file

## 15.2 Module `G2export_csv`: Spreadsheet export

Code to create .csv (comma-separated variable) files for GSAS-II data export to a spreadsheet program, etc.

**class** `G2export_csv.ExportMultiPowderCSV` (*G2frame*)

Used to create a csv file for a stack of powder data sets suitable for display purposes only; no y-calc or weights are exported only x & y-obs :param wx.Frame G2frame: reference to main GSAS-II frame

**Exporter** (*event=None*)

Export a set of powder data as a csv file

**class** `G2export_csv.ExportPhaseCSV` (*G2frame*)

Used to create a csv file for a phase

**Parameters** **G2frame** (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export a phase as a csv file

**class** `G2export_csv.ExportPowderCSV` (*G2frame*)

Used to create a csv file for a powder data set

**Parameters** **G2frame** (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export a set of powder data as a csv file

**class** `G2export_csv.ExportPowderRef1CSV` (*G2frame*)

Used to create a csv file of reflections from a powder data set

**Parameters** **G2frame** (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export a set of powder reflections as a csv file

**class** `G2export_csv.ExportSingleCSV` (*G2frame*)

Used to create a csv file with single crystal reflection data

**Parameters** **G2frame** (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export a set of single crystal data as a csv file

**class** `G2export_csv.ExportStrainCSV` (*G2frame*)

Used to create a csv file with single crystal reflection data

**Parameters** **G2frame** (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export a set of single crystal data as a csv file

`G2export_csv.WriteList` (*obj, headerItems*)

Write a CSV header

**Parameters**

- **obj** (*object*) – Exporter object
- **headerItems** (*list*) – items to write as a header

## 15.3 Module *G2export\_PDB: Macromolecular export*

Code to export a phase into the venerated/obsolete (pick one) ASCII PDB format. Also defines exporter `ExportPhaseCartXYZ` which writes atom positions in orthogonal coordinates for a phase.

**class** `G2export_PDB.ExportPhaseCartXYZ` (*G2frame*)

Used to create a Cartesian XYZ file for a phase

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export as a XYZ file

**class** `G2export_PDB.ExportPhasePDB` (*G2frame*)

Used to create a PDB file for a phase

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export as a PDB file

## 15.4 Module *G2export\_image: 2D Image data export*

Demonstrates how an image is retrieved and written. Uses a SciPy routine to write a PNG format file.

**class** `G2export_image.ExportImagePNG` (*G2frame*)

Used to create a PNG file for a GSAS-II image

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export an image

## 15.5 Module *G2export\_map: Map export*

Code to write Fourier/Charge-Flip atomic density maps out in formats that can be read by external programs. At present a GSAS format that is supported by FOX and DrawXTL (`ExportMapASCII`) and the CCP4 format that is used by COOT (`ExportMapCCP4`) are implemented.

**class** `G2export_map.ExportMapASCII` (*G2frame*)

Used to create a text file for a phase

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export a map as a text file

**class** `G2export_map.ExportMapCCP4` (*G2frame*)

Used to create a text file for a phase

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export a map as a text file

## 15.6 Module `G2export_shelx`: Examples

Code to export coordinates in the SHELX .ins format (as best as I can makes sense of it).

**class** `G2export_shelx.ExportPhaseShelx` (*G2frame*)

Used to create a SHELX .ins file for a phase

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export as a SHELX .ins file

## 15.7 Module `G2export_CIF`: CIF Exports

This implements a complex exporter `ExportCIF` that can implement an entire project in a complete CIF intended for submission as a publication. In addition, there are two subclasses of `ExportCIF`: `ExportPhaseCIF` and `ExportDataCIF` that export a single phase or data set. Note that `self.mode` determines what is written:

- `self.mode="simple"` creates a simple CIF with only coordinates or data, while
- `self.mode="full"` creates a complete CIF of project.

`G2export_CIF.CIF2dict` (*cf*)

copy the contents of a CIF out from a `PyCifRW` block object into a dict

**Returns** `cifblk`, `loopstructure` where `cifblk` is a dict with CIF items and `loopstructure` is a list of lists that defines which items are in which loops.

**class** `G2export_CIF.CIFdefHelp` (*parent, msg, helpwin, helptxt*)

Create a help button that displays help information on the current data item

**Parameters**

- **parent** – the panel which will be the parent of the button
- **msg** (*str*) – the help text to be displayed
- **helpwin** (*wx.Dialog*) – Frame for CIF editing dialog
- **helptxt** (*wx.TextCtrl*) – `TextCtrl` where help text is placed

**class** `G2export_CIF.CIFtemplateSelect` (*frame, panel, tmlate, G2dict, repaint, title, defaultname=''*)

Create a set of buttons to show, select and edit a CIF template

**Parameters**

- **frame** – `wx.Frame` object of parent
- **panel** – `wx.Panel` object where widgets should be placed
- **tmlate** (*str*) – one of 'publ', 'phase', or 'instrument' to determine the type of template
- **G2dict** (*dict*) – GSAS-II dict where CIF should be placed. The key "CIF\_template" will be used to store either a list or a string. If a list, it will contain a dict and a list defining loops. If an str, it will contain a file name.
- **repaint** (*function*) – reference to a routine to be called to repaint the frame after a change has been made
- **title** (*str*) – A line of text to show at the top of the window
- **defaultname** (*str*) – specifies the default file name to be used for saving the CIF.

**class** G2export\_CIF.**EditCIFpanel** (*parent, cifblk, loopstructure, cifdic={}, OKbuttons=[]*, *\*\*kw*)  
 Creates a scrolled panel for editing CIF template items

**Parameters**

- **parent** (*wx.Frame*) – parent frame where panel will be placed
- **cifblk** – dict or PyCifRW block containing values for each CIF item
- **loopstructure** (*list*) – a list of lists containing the contents of each loop, as an example:

```
[ ["_a", "_b"], ["_c"], ["_d_1", "_d_2", "_d_3"]]
```

this describes a CIF with this type of structure:

```
loop_ _a _b <a1> <b1> <a2> ...
loop_ _c <c1> <c2>...
loop _d_1 _d_2 _d_3 ...
```

Note that the values for each looped CIF item, such as `_a`, are contained in a list, for example as `cifblk["_a"]`

- **cifdic** (*dict*) – optional CIF dictionary definitions
- **OKbuttons** (*list*) – A list of `wx.Button` objects that should be disabled when information in the CIF is invalid
- **(other)** – optional keyword parameters for `wx.ScrolledPanel`

**CIFEntryWidget** (*dct, item, dataname*)

Create an entry widget for a CIF item. Use a validated entry for num values where int is required when limits are integers and floats otherwise. At present this does not allow entry of the special CIF values of `..` and `??` for numerical values and highlights them as invalid. Use a selection widget when there are specific enumerated values for a string.

**ControlOKButton** (*setvalue*)

Enable or Disable the OK button(s) for the dialog. Note that this is passed into the `ValidatedTxtCtrl` for use by validators.

**Parameters** *setvalue* (*bool*) – if True, all entries in the dialog are checked for validity. The first invalid control triggers disabling of buttons. If False then the OK button(s) are disabled with no checking of the invalid flag for each control.

**DoLayout** ()

Update the Layout and scroll bars for the Panel. Clears `self.LayoutCalled` so that next change to panel can request a new update

**OnAddRow** (*event*)

add a row to a loop

**OnLayoutNeeded** (*event*)

Called when an update of the panel layout is needed. Calls `self.DoLayout` after the current operations are complete using `CallAfter`. This is called only once, according to flag `self.LayoutCalled`, which is cleared in `self.DoLayout`.

**class** G2export\_CIF.**EditCIFtemplate** (*parent, cifblk, loopstructure, defaultname*)

Create a dialog for editing a CIF template. The edited information is placed in `cifblk`. If the CIF is saved as a file, the name of that file is saved as `self.newfile`.

**Parameters**

- **parent** (*wx.Frame*) – parent frame or None

- **cifblk** – dict or PyCifRW block containing values for each CIF item
- **loopstructure** (*list*) – a list of lists containing the contents of each loop, as an example:

```
[ ["_a", "_b"], ["_c"], ["_d_1", "_d_2", "_d_3"]]
```

this describes a CIF with this type of structure:

```
loop_ _a _b <a1> <b1> <a2> ...
loop_ _c <c1> <c2>...
loop _d_1 _d_2 _d_3 ...
```

Note that the values for each looped CIF item, such as `_a`, are contained in a list, for example as `cifblk["_a"]`

- **defaultname** (*str*) – specifies the default file name to be used for saving the CIF.

**Post ()**

Display the dialog

**Returns** True unless Cancel has been pressed.

**class** `G2export_CIF.ExportCIF (G2frame)`

Used to create a CIF of an entire project

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export a CIF. Export can be full or simple (as set by `self.mode`). “simple” skips data, distances & angles, etc. and can only include a single phase while “full” is intended for for publication submission.

**class** `G2export_CIF.ExportDataCIF (G2frame)`

Used to create a simple CIF containing diffraction data only. Uses exact same code as `ExportCIF` except that `self.mode` is set to “simple” and `self.currentExportType` is set to “single” or “powder” in `self.InitExport`. Shows up in menus as Data-only CIF.

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**class** `G2export_CIF.ExportPhaseCIF (G2frame)`

Used to create a simple CIF of at most one phase. Uses exact same code as `ExportCIF` except that `self.mode` is set to “simple” in `self.InitExport`. Shows up in menu as Quick CIF.

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

`G2export_CIF.LoadCIFdic ()`

Create a composite core+powder CIF lookup dict containing information about all items in the CIF dictionaries, loading pickled files if possible. The routine looks for files named `cif_core.cpickle` and `cif_pd.cpickle` in every directory in the path and if they are not found, files `cif_core.dic` and/or `cif_pd.dic` are read.

**Returns** the dict with the definitions

`G2export_CIF.PickleCIFdict (fil)`

Loads a CIF dictionary, cherry picks out the items needed by local code and sticks them into a python dict and writes that dict out as a cPickle file for later reuse. If the write fails a warning message is printed, but no exception occurs.

**Parameters** `fil` (*str*) – file name of CIF dictionary, will usually end in `.dic`

**Returns** the dict with the definitions

`G2export_CIF.dict2CIF (dblk, loopstructure, blockname='Template')`

Create a PyCifRW CIF object containing a single CIF block object from a dict and loop structure list.

**Parameters**

- **dblk** – a dict containing values for each CIF item
- **loopstructure** (*list*) – a list of lists containing the contents of each loop, as an example:

```
[ ["_a", "_b"], ["_c"], ["_d_1", "_d_2", "_d_3"]]
```

this describes a CIF with this type of structure:

```
loop_ _a _b <a1> <b1> <a2> ...
loop_ _c <c1> <c2>...
loop _d_1 _d_2 _d_3 ...
```

Note that the values for each looped CIF item, such as `_a`, are contained in a list, for example as `cifblk["_a"]`

- **blockname** (*str*) – an optional name for the CIF block. Defaults to ‘Template’

**Returns** the newly created PyCifRW CIF object

## 15.8 Module `G2export_pwdr`: Export powder input files

Creates files used by GSAS (FXYE) & TOPAS (XYE) as input

**class** `G2export_pwdr.ExportPowderFXYE` (*G2frame*)

Used to create a FXYE file for a powder data set

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export one or more sets of powder data as FXYE file(s)

**WriteInstFile** (*hist, Inst*)

Write an instrument parameter file

**Writer** (*TreeName, filename=None, prmname=''*)

Write a single PWDR entry to a FXYE file

**class** `G2export_pwdr.ExportPowderXYE` (*G2frame*)

Used to create a Topas XYE file for a powder data set

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export one or more sets of powder data as XYE file(s)

## 15.9 Module `G2export_FIT2D`: Fit2D “Chi” export

Code to create .chi (Fit2D like) files for GSAS-II powder data export

**class** `G2export_FIT2D.ExportPowderCSV` (*G2frame*)

Used to create a csv file for a powder data set

**Parameters** `G2frame` (*wx.Frame*) – reference to main GSAS-II frame

**Exporter** (*event=None*)

Export a set of powder data as a Fit2D .chi file





## GSAS-II IMPORT MODULES

Imports are implemented by deriving a class from `GSASIIIO.ImportPhase`, `GSASIIIO.ImportStructFactor` or `GSASIIIO.ImportPowderData` (which are in turn derived from `GSASIIIO.ImportBaseclass`) to implement import of a phase, a single crystal or a powder dataset, respectively. Module file names (*G2phase\_*, *G2pwd\_* and *G2sfact\_*, etc.) are used to determine which menu an import routine should be placed into. (N.B. this was an unnecessary choice; this could be done from the class used.)

This list may not include all currently defined formats, since modules may be loaded from anywhere in the path.

### 16.1 Writing an Import Routine

When writing a import routine, one should create a new class derived from `GSASIIIO.ImportPhase`, `GSASIIIO.ImportStructFactor` or `GSASIIIO.ImportPowderData`. As described below, all these classes will implement an `__init__()` and a `Reader()` method, and most will supply a `ContentsValidator()` method, too. See the `ImportPhase`, `ImportStructFactor` or `ImportPowderData` class documentation for details on what values each type of `Reader()` should set.

#### 16.1.1 `__init__()`

The class should supply a `__init__` method which calls the parent `__init__` method and specifies the following parameters:

- *extensionlist*: a list of extensions that may be used for this type of file.
- *strictExtension*: Should be True if only files with extensions in *extensionlist* are allows; False if all file types should be offered in the file browser. Also if False, the import class will be used on all files when “guess from format” is tried, though readers with matching extensions will be tried first.
- *formatName*: a string to be used in the menu. Should be short.
- *longFormatName*: a longer string to be used to describe the format in help.

#### 16.1.2 `Reader()`

The class must supply a `Reader` method that actually performs the reading. All readers must have at a minimum these arguments:

```
def Reader(self, filename, filepointer, ParentFrame, **unused):
```

where the arguments have the following uses:

- *filename*: a string with the name of the file being read

- *filepointer*: a file object (created by `open()`) that accesses the file and is points to the beginning of the file when `Reader` is called.
- *ParentFrame*: a reference to the main GSAS-II (tree) windows, for the unusual `Reader` routines that will create GUI windows to ask questions.

In addition, the following keyword parameters are defined that `Reader` routines may optionally use:

- *buffer*: a dict that can be used to retain information between repeated calls of the routine
- *blocknum*: counts the number of times that a reader is called
- *usedRandList*: a list of previously used random Id values that can be checked to determine that a value is unique.

As an example, the *buffer* dict is used for CIF reading to hold the parsed CIF file so that it can be reused without having to reread the file from scratch.

### Reader return values

The `Reader` routine should return the value of `True` if the file has been read successfully. Optionally, use *self.warnings* to indicate any problems.

If the file cannot be read, the `Reader` routine should return `False` or raise an `GSASIIIO.ImportBaseclass.ImportException` exception. (Why either? Sometimes an exception is the easiest way to bail out of a called routine.) Place text in *self.errors* and/or use:

```
ImportException('Error message')
```

to give the user information on what went wrong during the reading.

### self.warnings

Use *self.warnings* to indicate any information that should be displayed to the user if the file is read successfully, but perhaps not completely or additional settings will need to be made.

### self.errors

Use *self.errors* to give the user information on where and why a read error occurs in the file. Note that text supplied with the `raise` statement will be appended to *self.errors*.

### self.repeat

Set *self.repeat* to `True` (the default is `False`) if a `Reader` should be called again to read a second block from a file. Most commonly (only?) used for reading multiple powder histograms from a single file. Variable *self.repeatcount* is used to keep track of the block numbers.

### support routines

Note that the base class (`GSASIIIO.ImportBaseclass`) supplies two routines, `BlockSelector()` and `MultipleBlockSelector()` that are useful for selecting amongst one or more datasets (and perhaps phases) for `Reader()` routines that may encounter more than one set of information in a file. Likewise, when an operation will take some time to complete, use `ShowBusy()` and `DoneBusy()` to show the user that something is happening.

### 16.1.3 ContentsValidator()

Defining a `ContentsValidator` method is optional, but is usually a good idea, particularly if the file extension is not a reliable identifier for the file type. The intent of this routine is to take a superficial look at the file to see if it has the expected characteristics of the expected file type. For example, are there numbers in the expected places?

This routine is passed a single argument:

- *filepointer*: a file object (created by `open()`) that accesses the file and is points to the beginning of the file when `ContentsValidator` is called.

Note that `GSASIIIO.ImportBaseclass.CIFValidator()` is a `ContentsValidator` for validating CIF files.

### 16.1.4 ReInitialize()

Import classes are substantiated only once and are used as needed. This means that if something needs to be initialized before the `Reader()` will be called to read a new file, it must be coded. The `ReInitialize()` method is provided for this and it is always called before the `ContentsValidator` method is called. Use care to call the parent class `ReInitialize()` method, if this is overridden.

#### ContentsValidator return values

The `ContentsValidator` routine should return the value of `True` if the file appears to match the type expected for the class.

If the file cannot be read by this class, the routine should return `False`. Preferably one will also place text in *self.errors* to give the user information on what went wrong during the reading.

## 16.2 Phase Import Routines

Phase import routines are classes derived from `GSASIIIO.ImportPhase`. They must be found in files named *G2phase\*.py* that are in the Python path and the class must override the `__init__` method and add a `Reader` method. The distributed routines are:

### 16.2.1 Module G2phase: PDB, .EXP & JANA m40,m50

A set of short routines to read in phases using routines that were previously implemented in GSAS-II: PDB, GSAS .EXP and JANA m40-m50 file formats

```
class G2phase.EXP_ReaderClass
    Routine to import Phase information from GSAS .EXP files

    ContentsValidator (filepointer)
        Look for a VERSION tag in 1st line

    ReadEXPPhase (G2frame, filepointer)
        Read a phase from a GSAS .EXP file.

    Reader (filename, filepointer, ParentFrame=None, **unused)
        Read a phase from a GSAS .EXP file using ReadEXPPhase ()

class G2phase.JANA_ReaderClass
    Routine to import Phase information from a JANA2006 file
```

**ContentsValidator** (*filepointer*)

Taking a stab at validating a .m50 file (look for cell & at least one atom)

**ReadJANAPhase** (*filename, parent=None*)

Read a phase from a JANA2006 m50 & m40 files.

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)

Read a m50 file using `ReadJANAPhase()`

**class** `G2phase.PDB_ReaderClass`

Routine to import Phase information from a PDB file

**ContentsValidator** (*filepointer*)

Taking a stab at validating a PDB file (look for cell & at least one atom)

**ReadPDBPhase** (*filename, parent=None*)

Read a phase from a PDB file.

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)

Read a PDF file using `ReadPDBPhase()`

### 16.2.2 Module `G2phase_GPX`: Import phase from GSAS-II project

Copies a phase from another GSAS-II project file into the current project.

**class** `G2phase_GPX.PhaseReaderClass`

Opens a .GPX file and pulls out a selected phase

**ContentsValidator** (*filepointer*)

Test if the 1st section can be read as a cPickle block, if not it can't be .GPX!

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)

Read a phase from a .GPX file. Does not (yet?) support selecting and reading more than one phase at a time.

### 16.2.3 Module `G2phase_CIF`: Coordinates from CIF

Parses a CIF using PyCifRW from James Hester and pulls out the structural information.

If a CIF generated by ISODISTORT is encountered, extra information is added to the phase entry and constraints are generated.

**class** `G2phase_CIF.CIFPhaseReader`

Implements a phase importer from a possibly multi-block CIF file

**ISODISTORT\_proc** (*blk, atombllist, ranIdlookup*)

Process ISODISTORT items to create constraints etc.

## 16.3 Powder Data Import Routines

Powder data import routines are classes derived from `GSASIIIO.ImportPowderData`. They must be found in files named `G2pwd*.py` that are in the Python path and the class must override the `__init__` method and add a `Reader` method.

The distributed routines are:

### 16.3.1 Module *G2pwd\_GPX: GSAS-II projects*

Routine to import powder data from GSAS-II .gpx files

**class** `G2pwd_GPX.GSAS2_ReaderClass`

Routines to import powder data from a GSAS-II file This should work to pull data out from a out of date .GPX file as long as the details of the histogram data itself don't change

**ContentsValidator** (*filepointer*)

Test if the 1st section can be read as a cPickle block, if not it can't be .GPX!

**Reader** (*filename, filepointer, ParentFrame=None, \*\*kwarg*)

Read a dataset from a .GPX file. If multiple datasets are requested, use self.repeat and buffer caching.

### 16.3.2 Module *G2pwd\_fxyc: GSAS data files*

Routine to read in powder data in a variety of formats that are defined for GSAS.

**class** `G2pwd_fxyc.GSAS_ReaderClass`

Routines to import powder data from a GSAS files

**ContentsValidator** (*filepointer*)

Validate by checking to see if the file has BANK lines

**Reader** (*filename, filepointer, ParentFrame=None, \*\*kwarg*)

Read a GSAS (old formats) file of type FXY, FXYE, ESD or STD types. If multiple datasets are requested, use self.repeat and buffer caching.

`G2pwd_fxyc.sfloat` (*S*)

convert a string to a float, treating an all-blank string as zero

`G2pwd_fxyc.sint` (*S*)

convert a string to an integer, treating an all-blank string as zero

### 16.3.3 Module *G2pwd\_xyc: Topas .xyc data*

Routine to read in powder data from a Topas-compatible .xyc file

**class** `G2pwd_xyc.xyc_ReaderClass`

Routines to import powder data from a .xyc file

**ContentsValidator** (*filepointer*)

Look through the file for expected types of lines in a valid Topas file

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)

Read a Topas file

### 16.3.4 Module *G2pwd\_CIF: CIF powder data*

Routine to read in powder data from a CIF.

**class** `G2pwd_CIF.CIFpwdReader`

Routines to import powder data from a CIF file

**ContentsValidator** (*filepointer*)

Use standard CIF validator

**Reader** (*filename, filepointer, ParentFrame=None, \*\*kwarg*)

Read powder data from a CIF. If multiple datasets are requested, use self.repeat and buffer caching.

## 16.4 Single Crystal Data Import Routines

Single crystal data import routines are classes derived from `GSASIIIO.ImportStructFactor`. They must be found in files named `G2sfact*.py` that are in the Python path and the class must override the `__init__` method and add a `Reader` method. The distributed routines are:

### 16.4.1 Module `G2sfact`: simple HKL import

Read structure factors from a simple hkl file. Two routines are provided to read from files containing F or F<sup>2</sup> values.

`G2sfact.ColumnValidator` (*parent, filepointer, nCol=5*)

Validate a file to check that it contains columns of numbers

**class** `G2sfact.HKLF_ReaderClass`

Routines to import F, sig(F) reflections from a HKLF file

**ContentsValidator** (*filepointer*)

Make sure file contains the expected columns on numbers

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)

Read the file

**class** `G2sfact.ISIS_SXD_INT_ReaderClass`

Routines to import neutron TOF F\*\*2, sig(F\*\*2) reflections from a ISIS int file

**ContentsValidator** (*filepointer*)

Make sure file contains the expected columns on numbers & count number of data blocks - "Banks"

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)

Read the file

**class** `G2sfact.M90_ReaderClass`

Routines to import F\*\*2, sig(F\*\*2) reflections from a JANA M90 file

**ContentsValidator** (*filepointer*)

Discover how many columns are in the m90 file - could be 9-12 depending on satellites

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)

Read the file

**class** `G2sfact.NT_HKLF2_ReaderClass`

Routines to import neutron TOF F\*\*2, sig(F\*\*2) reflections from a HKLF file

**ContentsValidator** (*filepointer*)

Make sure file contains the expected columns on numbers & count number of data blocks - "Banks"

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)

Read the file

**class** `G2sfact.NT_JANA2K_ReaderClass`

Routines to import neutron TOF F\*\*2, sig(F\*\*2) reflections from a JANA2000 file

**ContentsValidator** (*filepointer*)

Make sure file contains the expected columns on numbers & count number of data blocks - "Banks"

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)  
Read the file

**class** `G2sfact.SHELX4_ReaderClass`  
Routines to import F\*\*2, sig(F\*\*2) reflections from a Shelx HKLF 4 file

**ContentsValidator** (*filepointer*)  
Make sure file contains the expected columns on numbers

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)  
Read the file

**class** `G2sfact.SHELX5_ReaderClass`  
Routines to import F\*\*2, sig(F\*\*2) twin/incommensurate reflections from a fixed format SHELX HKLF5 file

**ContentsValidator** (*filepointer*)  
Discover how many columns before F^2 are in the SHELX HKL5 file - could be 3-6 depending on satellites

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)  
Read the file

## 16.4.2 Module `G2sfact_CIF: CIF import`

Read structure factors from a CIF reflection table.

**class** `G2sfact_CIF.CIFhklReader`  
Routines to import Phase information from a CIF file

**ContentsValidator** (*filepointer*)  
Use standard CIF validator

**Reader** (*filename, filepointer, ParentFrame=None, \*\*kwarg*)  
Read single crystal data from a CIF. If multiple datasets are requested, use self.repeat and buffer caching.

## 16.5 Small Angle Scattering Data Import Routines

Small angle scattering data import routines are classes derived from `GSASIIIO.ImportSmallAngle`. They must be found in files named `G2sad*.py` that are in the Python path and the class must override the `__init__` method and add a `Reader` method. The distributed routines are:

### 16.5.1 Module `G2sad_xye: read small angle data`

Routines to read in small angle data from an .xye type file, with two-theta or Q steps.

**class** `G2sad_xye.txt_CWNeutronReaderClass`  
Routines to import neutron CW q SAXD data from a .nsad or .ndat file

**ContentsValidator** (*filepointer*)  
Look through the file for expected types of lines in a valid q-step file

**class** `G2sad_xye.txt_XRayReaderClass`  
Routines to import X-ray q SAXD data from a .xsad or .xdat file

**ContentsValidator** (*filepointer*)  
Look through the file for expected types of lines in a valid q-step file

**class** `G2sad_xye.txt_nmCWNeutronReaderClass`  
Routines to import neutron CW q in nm-1 SAXD data from a .nsad or .ndat file

**ContentsValidator** (*filepointer*)

Look through the file for expected types of lines in a valid q-step file

**class** `G2sad_1ye.txt_nmXRayReaderClass`

Routines to import X-ray q SAXD data from a .xsad or .xdat file, q in nm-1

**ContentsValidator** (*filepointer*)

Look through the file for expected types of lines in a valid q-step file

## 16.6 Image Import Routines

Image import routines are classes derived from , `GSASIIIO.ImportImage`. They must be found in files named `G2img*.py` that are in the Python path and the class must override the `__init__` method and add a `Reader` method. The distributed routines are:

### 16.6.1 Module `G2img_ADSC`: *.img* image file

Routine to read an ADSC *.img* file

### 16.6.2 Module `G2img_EDF`: *.edf* image file

Routine to read a Read European detector data edf file

### 16.6.3 Module `G2img_G2`: *Python pickled image*

Routine to read an image that has been pickled in Python. Images in this format are created by the “Sum image data” command.

### 16.6.4 Module `G2img_GE`: *summed GE image file*

Routine to read a summed GE image from APS Sector 1

### 16.6.5 Module `G2img_MAR`: *MAR image files*

Routine to read several MAR formats, *.mar3450*, *.mar2300*, *.mar2560*

### 16.6.6 Module `G2img_Rigaku`: *.stl* image file

Routine to read a Rigaku R-Axis IV image file

### 16.6.7 Module `G2img_TIF`: *.tif* image file

Routine to read an image in Tagged-image file (TIF) format as well as a variety of slightly incorrect pseudo-TIF formats used at instruments around the world.

**class** `G2img_1TIF.TIF_ReaderClass`

Routine to read an image in Tagged-image file (TIF) format as well as a variety of slightly incorrect pseudo-TIF formats



**ContentsValidator** (*filepointer*)

Does the header match the required TIF header?

**Reader** (*filename, filepointer, ParentFrame=None, \*\*unused*)

Read the TIF file using Bob's routine

### 16.6.8 *Module G2img\_png: png image file*

Routine to read an image in .png (Portable Network Graphics) format. For now, the only known use of this is with converted CheMin tif files so default parameters are that machine.



## ***REQUIRED PACKAGES***

Note that GSAS-II requires the Python extension packages

- wxPython (<http://wxpython.org/docs/api/>),
- NumPy (<http://docs.scipy.org/doc/numpy/reference/>),
- SciPy (<http://docs.scipy.org/doc/scipy/reference/>),
- matplotlib (<http://matplotlib.org/contents.html>)
- PIL or Pillow (<https://pillow.readthedocs.org>) and
- PyOpenGL (<http://pyopengl.sourceforge.net/documentation>)

These packages are not distributed as part of the Python standard library and must be obtained separately (or in a bundled Python package such as the Enthought Python Distribution/Canopy or Continuum.io's anaconda). The PyOpenGL package will be installed into Python by GSAS-II if not found, so it does not need to be included in the Python bundle, but the setuptools package (<https://pythonhosted.org/setuptools/>) is needed by GSAS-II to install PyOpenGL.



**a**

atmdata, 49

**c**

config\_example, 31

**d**

defaultIparms, 50

**e**

ElementTable, 48

**f**

FormFactors, 48

**g**

G2export\_CIF, 134

G2export\_csv, 131

G2export\_examples, 131

G2export\_FIT2D, 137

G2export\_image, 133

G2export\_map, 133

G2export\_PDB, 132

G2export\_pwdr, 137

G2export\_shelx, 133

G2img\_1TIF, 146

G2img\_ADSC, 146

G2img\_CheMin, 147

G2img\_EDF, 146

G2img\_GE, 146

G2img\_MAR, 146

G2img\_Rigaku, 146

G2img\_SumG2, 146

G2phase, 141

G2phase\_CIF, 142

G2phase\_GPX, 142

G2pwd\_CIF, 143

G2pwd\_fxyc, 143

G2pwd\_GPX, 142

G2pwd\_xyc, 143

G2sad\_xyc, 145

G2sfact, 144

G2sfact\_CIF, 145

gltext, 46

GSASII, 1

GSASIIconstrGUI, 76

GSASIIctrls, 51

GSASIIdata, 27

GSASIIddataGUI, 75

GSASIIElem, 32

GSASIIElemGUI, 75

GSASIIexprGUI, 78

GSASIIgrid, 64

GSASIIimage, 94

GSASIIimgGUI, 76

GSASIIindex, 111

GSASIIIO, 66

GSASIIlattice, 34

GSASIIlog, 30

GSASIImapvars, 88

GSASIImath, 96

GSASIIobj, 6

GSASIIpath, 27

GSASIIphsGUI, 75

GSASIIplot, 114

GSASIIpwd, 117

GSASIIpwdGUI, 76

GSASIIpy3, 73

GSASIIrestrGUI, 78

GSASIIasad, 123

GSASIIspc, 41

GSASIIstrIO, 85

GSASIIstrMain, 81

GSASIIstrMath, 82

GSASIItestplot, 129

**i**

ImageCalibrants, 49

**m**

makeMacApp, 129

**r**

ReadMarCCDFrame, 73

## S

[scanCCD](#), 129

[Substances](#), 126

## t

[testDeriv](#), 129

## U

[unit\\_tests](#), 130

## A

A2cell() (in module GSASIIlattice), 35  
 A2Gmat() (in module GSASIIlattice), 35  
 A2invcell() (in module GSASIIlattice), 35  
 A2values() (in module GSASIIindex), 113  
 Absorb() (in module GSASIIpwd), 119  
 AbsorbDerv() (in module GSASIIpwd), 119  
 add3D() (GSASIIplot.G2PlotNoteBook method), 115  
 AddHatomDialog (class in GSASIIgrid), 64  
 AddHelp (class in GSASIIctrls), 51  
 addMpl() (GSASIIplot.G2PlotNoteBook method), 115  
 addOgl() (GSASIIplot.G2PlotNoteBook method), 115  
 adjHKLmax() (in module GSASIImath), 103  
 AllOps() (in module GSASIIspc), 41  
 anneal() (in module GSASIImath), 103  
 AppleScript (in module makeMacApp), 130  
 ApplyEdit() (GSASIIctrls.GridFractionEditor method), 56  
 ApplyModulation() (in module GSASIImath), 97  
 ApplyRBModelDervs() (in module GSASIIstrMath), 82  
 ApplyRBModels() (in module GSASIIstrMath), 82  
 ApplyStringOps() (in module GSASIIspc), 42  
 ApplyXYZshifts() (in module GSASIIstrMath), 82  
 ASCIIValidator (class in GSASIIctrls), 51  
 askSaveDirectory() (GSASIIIO.ExportBaseclass method), 68  
 askSaveFile() (GSASIIIO.ExportBaseclass method), 68  
 assignVars (GSASIIobj.ExpressionObj attribute), 21  
 atmdata (module), 49  
 AtomIdLookup (in module GSASIIobj), 19  
 AtomRanIdLookup (in module GSASIIobj), 19  
 Atoms record description, 12  
 AtomTLS2UIJ() (in module GSASIImath), 97  
 AtomUij2TLS() (in module GSASIImath), 97  
 AV2Q() (in module GSASIImath), 97  
 AVdeg2Q() (in module GSASIImath), 97

## B

Banks (GSASIIIO.ImportStructFactor attribute), 72  
 BessIn() (in module GSASIImath), 97  
 BessJn() (in module GSASIImath), 97  
 BestPlane() (in module GSASIIstrMain), 81

bind() (gltext.TextElement method), 47  
 Bind() (GSASII.GSASII method), 1  
 Bind() (GSASIIctrls.G2TreeCtrl method), 56  
 Bind() (GSASIIctrls.GSNoteBook method), 56  
 Bind() (GSASIIgrid.DataFrame method), 64  
 BlockSelector() (GSASIIIO.ImportBaseclass method), 70  
 ButtonBindingLookup (in module GSASIIlog), 30  
 ButtonLogEntry (class in GSASIIlog), 30

## C

calc\_M20() (in module GSASIIindex), 113  
 calc\_M20SS() (in module GSASIIindex), 113  
 calc\_rDsq() (in module GSASIIlattice), 39  
 calc\_rDsq2() (in module GSASIIlattice), 39  
 calc\_rDsqSS() (in module GSASIIlattice), 39  
 calc\_rDsqT() (in module GSASIIlattice), 39  
 calc\_rDsqTSS() (in module GSASIIlattice), 39  
 calc\_rDsqZ() (in module GSASIIlattice), 39  
 calc\_rDsqZSS() (in module GSASIIlattice), 39  
 calc\_rV() (in module GSASIIlattice), 39  
 calc\_rVsq() (in module GSASIIlattice), 39  
 calc\_V() (in module GSASIIlattice), 39  
 calcFij() (in module GSASIIimage), 96  
 calcIncident() (in module GSASIIpwd), 121  
 CalcPDF() (in module GSASIIpwd), 119  
 calcRamaEnergy() (in module GSASIImath), 105  
 calcTorsionEnergy() (in module GSASIImath), 105  
 CallScrolledMultiEditor() (in module GSASIIctrls), 51  
 cauchy\_gen (class in GSASIIpwd), 121  
 cell2A() (in module GSASIIlattice), 39  
 cell2AB() (in module GSASIIlattice), 39  
 cell2Gmat() (in module GSASIIlattice), 39  
 cell2GS() (in module GSASIIlattice), 39  
 CellAbsorption() (in module GSASIIlattice), 35  
 CellBlock() (in module GSASIIlattice), 35  
 cellFill() (in module GSASIIstrIO), 88  
 cellVary() (in module GSASIIstrIO), 88  
 CentCheck() (in module GSASIIlattice), 35  
 centered (gltext.Text attribute), 46  
 centered (gltext.TextElement attribute), 47  
 ChargeFlip() (in module GSASIImath), 97  
 CheckConstraints() (in module GSASIImapvars), 91

- CheckElement() (in module GSASIIElem), 32  
 checkEllipse() (in module GSASIIImage), 96  
 CheckImageFile() (in module GSASIIIO), 66  
 CheckInput() (GSASIIctrls.NumberValidator method), 58  
 CheckNotebook() (GSASII.GSASII method), 1  
 CheckVars() (GSASIIexprGUI.ExpressionDialog method), 79  
 CheckVars() (GSASIIobj.ExpressionObj method), 20  
 CIF2dict() (in module G2export\_CIF), 134  
 CIFdefHelp (class in G2export\_CIF), 134  
 CIFEntryWidget() (G2export\_CIF.EditCIFpanel method), 135  
 CIFhklReader (class in G2sfact\_CIF), 145  
 CIFPhaseReader (class in G2phase\_CIF), 142  
 CIFpwdReader (class in G2pwd\_CIF), 143  
 CIFtemplateSelect (class in G2export\_CIF), 134  
 CIFValidator() (GSASIIIO.ImportBaseclass method), 70  
 CleanupMasks() (in module GSASIIimgGUI), 76  
 clear() (GSASIIplot.G2PlotNoteBook method), 115  
 Clone() (GSASIIctrls.ASCIIValidator method), 51  
 Clone() (GSASIIctrls.NumberValidator method), 58  
 CloseFile() (GSASIIIO.ExportBaseclass method), 67  
 ColumnValidator() (in module G2sfact), 144  
 combinations() (in module GSASIIlattice), 39  
 compiledExpr (GSASIIobj.ExpressionCalcObj attribute), 19  
 CompileVarDesc() (in module GSASIIobj), 19  
 ComptonFac() (in module GSASIIElem), 33  
 ComputeDepESD() (in module GSASIImapvars), 91  
 config\_example (module), 31  
 Constraint definition object description, 7  
 ConstraintDialog (class in GSASIIconstrGUI), 76  
 Constraints object description, 7  
 ContentsValidator() (G2img\_1TIF.TIF\_ReaderClass method), 146  
 ContentsValidator() (G2phase.EXP\_ReaderClass method), 141  
 ContentsValidator() (G2phase.JANA\_ReaderClass method), 141  
 ContentsValidator() (G2phase.PDB\_ReaderClass method), 142  
 ContentsValidator() (G2phase\_GPX.PhaseReaderClass method), 142  
 ContentsValidator() (G2pwd\_CIF.CIFpwdReader method), 143  
 ContentsValidator() (G2pwd\_fxye.GSAS\_ReaderClass method), 143  
 ContentsValidator() (G2pwd\_GPX.GSAS2\_ReaderClass method), 143  
 ContentsValidator() (G2pwd\_xye.xye\_ReaderClass method), 143  
 ContentsValidator() (G2sad\_xye.txt\_CWNeutronReaderClass method), 145  
 ContentsValidator() (G2sad\_xye.txt\_nmCWNeutronReaderClass method), 145  
 ContentsValidator() (G2sad\_xye.txt\_nmXRayReaderClass method), 146  
 ContentsValidator() (G2sad\_xye.txt\_XRayReaderClass method), 145  
 ContentsValidator() (G2sfact.HKLF\_ReaderClass method), 144  
 ContentsValidator() (G2sfact.ISIS\_SXD\_INT\_ReaderClass method), 144  
 ContentsValidator() (G2sfact.M90\_ReaderClass method), 144  
 ContentsValidator() (G2sfact.NT\_HKLF2\_ReaderClass method), 144  
 ContentsValidator() (G2sfact.NT\_JANA2K\_ReaderClass method), 144  
 ContentsValidator() (G2sfact.SHELX4\_ReaderClass method), 145  
 ContentsValidator() (G2sfact.SHELX5\_ReaderClass method), 145  
 ContentsValidator() (G2sfact\_CIF.CIFhklReader method), 145  
 ContentsValidator() (GSASIIIO.ImportBaseclass method), 70  
 ControlOKButton() (G2export\_CIF.EditCIFpanel method), 135  
 ControlOKButton() (GSASIIctrls.ScrolledMultiEditor method), 61  
 ConvertRelativeHistNum() (GSASIIctrls.G2TreeCtrl method), 56  
 ConvertRelativePhaseNum() (GSASIIctrls.G2TreeCtrl method), 56  
 CopyPlotCtrls() (in module GSASIIpwdGUI), 77  
 CopySelectedHistItems() (in module GSASIIpwdGUI), 77  
 CosAngle() (in module GSASIIlattice), 35  
 CosSinAngle() (in module GSASIIlattice), 36  
 Covariance description, 8  
 createTexture() (gltext.TextElement method), 47  
 criticalEllipse() (in module GSASIIlattice), 39  
 CrsAng() (in module GSASIIlattice), 36  
 CylinderARFF() (in module GSASIIIsasd), 123  
 CylinderARVol() (in module GSASIIIsasd), 123  
 CylinderDFF() (in module GSASIIIsasd), 123  
 CylinderDVOL() (in module GSASIIIsasd), 123  
 CylinderFF() (in module GSASIIIsasd), 123  
 CylinderVol() (in module GSASIIIsasd), 123
- ## D
- Data object descriptions  
   Atoms record, 12  
   Constraint Definition, 7  
   Constraints, 7  
   Covariance, 8



- Drawing atoms record, 12
  - Phase, 8
  - Powder Data, 13
  - Powder Reflections, 15
  - Rigid Body Data, 10
  - Single crystal data, 15
  - Single Crystal Reflections, 16
  - Space Group Data, 11
  - Superspace Group Data, 11
  - DataFrame (class in GSASIIgrid), 64
  - debug (in module config\_example), 32
  - DefaultControls (in module GSASIIobj), 19
  - defaultParms (module), 50
  - defSize (GSASIIexprGUI.ExpressionDialog attribute), 79
  - Delete() (GSASIIplot.G2PlotNoteBook method), 115
  - DeleteElement (class in GSASIIElemGUI), 76
  - deleteTexture() (gltext.TextElement method), 48
  - Den2Vol() (in module GSASIImath), 98
  - dependentVar (GSASIIexprGUI.ExpressionDialog attribute), 79
  - depVarDict (GSASIIexprGUI.ExpressionDialog attribute), 79
  - dervHKLf() (in module GSASIIstrMath), 84
  - dervRefine() (in module GSASIIstrMath), 84
  - dict2CIF() (in module G2export\_CIF), 136
  - Dict2Deriv() (in module GSASIImapvars), 91
  - Dict2Map() (in module GSASIImapvars), 92
  - Dict2Values() (in module GSASIIpwd), 119
  - Dict2Values() (in module GSASIIstrMath), 82
  - dictLogged (class in GSASIIlog), 31
  - DiluteSF() (in module GSASIIasad), 123
  - DisAglDialog (class in GSASIIgrid), 65
  - DisAglTor() (in module GSASIIstrMain), 81
  - DoIndexPeaks() (in module GSASIIindex), 113
  - DoLayout() (G2export\_CIF.EditCIFpanel method), 135
  - DoNothing() (in module GSASIIpath), 27
  - DoPeakFit() (in module GSASIIpwd), 119
  - downdate (class in GSASIIctrls), 64
  - Draw() (GSASIIgrid.AddHatomDialog method), 64
  - Draw() (GSASIIgrid.DisAglDialog method), 65
  - draw\_text() (gltext.Text method), 46
  - draw\_text() (gltext.TextElement method), 48
  - Drawing atoms record description, 12
  - Dsp2pos() (in module GSASIIlattice), 36
  - dumpTree() (GSASIIIO.ExportBaseclass method), 68
- ## E
- EdgeFinder() (in module GSASIIimage), 95
  - EditCIFpanel (class in G2export\_CIF), 134
  - EditCIFtemplate (class in G2export\_CIF), 135
  - EditExpression() (GSASIIobj.ExpressionObj method), 20
  - El2EstVol() (in module GSASIImath), 98
  - El2Mass() (in module GSASIImath), 98
  - ElButton() (GSASIIElemGUI.DeleteElement method), 76
  - ElButton() (GSASIIElemGUI.PickElement method), 76
  - ElementTable (module), 48
  - ElemPosition() (in module GSASIIspc), 42
  - ellipseSize() (in module GSASIIpwd), 121
  - ellipseSizeDerv() (in module GSASIIpwd), 121
  - Enable\_logging (in module config\_example), 32
  - EnableSeqRefineMenu() (GSASII.GSASII method), 1
  - EnumSelector (class in GSASIIctrls), 52
  - eObj (GSASIIobj.ExpressionCalcObj attribute), 20
  - ErrorDialog() (GSASII.GSASII method), 1
  - errRefine() (in module GSASIIstrMath), 85
  - EvalExpression() (GSASIIobj.ExpressionCalcObj method), 19
  - exceptHook() (in module GSASIIpath), 28
  - ExitMain() (GSASII.GSASII method), 1
  - EXP\_ReaderClass (class in G2phase), 141
  - ExportBaseclass (class in GSASIIIO), 66
  - ExportCIF (class in G2export\_CIF), 136
  - ExportDataCIF (class in G2export\_CIF), 136
  - Exporter() (G2export\_CIF.ExportCIF method), 136
  - Exporter() (G2export\_csv.ExportMultiPowderCSV method), 132
  - Exporter() (G2export\_csv.ExportPhaseCSV method), 132
  - Exporter() (G2export\_csv.ExportPowderCSV method), 132
  - Exporter() (G2export\_csv.ExportPowderReflCSV method), 132
  - Exporter() (G2export\_csv.ExportSingleCSV method), 132
  - Exporter() (G2export\_csv.ExportStrainCSV method), 132
  - Exporter() (G2export\_examples.ExportPhaseText method), 131
  - Exporter() (G2export\_examples.ExportPowderReflText method), 131
  - Exporter() (G2export\_examples.ExportPowderText method), 131
  - Exporter() (G2export\_examples.ExportSingleText method), 131
  - Exporter() (G2export\_FIT2D.ExportPowderCSV method), 137
  - Exporter() (G2export\_image.ExportImagePNG method), 133
  - Exporter() (G2export\_map.ExportMapASCII method), 133
  - Exporter() (G2export\_map.ExportMapCCP4 method), 133
  - Exporter() (G2export\_PDB.ExportPhaseCartXYZ method), 133
  - Exporter() (G2export\_PDB.ExportPhasePDB method), 133

- Exporter() (G2export\_pwdr.ExportPowderFXYE method), 137  
 Exporter() (G2export\_pwdr.ExportPowderXYE method), 137  
 Exporter() (G2export\_shelx.ExportPhaseShelx method), 134  
 ExportImagePNG (class in G2export\_image), 133  
 ExportMapASCII (class in G2export\_map), 133  
 ExportMapCCP4 (class in G2export\_map), 133  
 ExportMultiPowderCSV (class in G2export\_csv), 132  
 ExportPhaseCartXYZ (class in G2export\_PDB), 133  
 ExportPhaseCIF (class in G2export\_CIF), 136  
 ExportPhaseCSV (class in G2export\_csv), 132  
 ExportPhasePDB (class in G2export\_PDB), 133  
 ExportPhaseShelx (class in G2export\_shelx), 134  
 ExportPhaseText (class in G2export\_examples), 131  
 ExportPowder() (in module GSASIIIO), 68  
 ExportPowderCSV (class in G2export\_csv), 132  
 ExportPowderCSV (class in G2export\_FIT2D), 137  
 ExportPowderFXYE (class in G2export\_pwdr), 137  
 ExportPowderList() (in module GSASIIIO), 69  
 ExportPowderRefCSV (class in G2export\_csv), 132  
 ExportPowderRefText (class in G2export\_examples), 131  
 ExportPowderText (class in G2export\_examples), 131  
 ExportPowderXYE (class in G2export\_pwdr), 137  
 ExportSelect() (GSASIIIO.ExportBaseclass method), 67  
 ExportSingleCSV (class in G2export\_csv), 132  
 ExportSingleText (class in G2export\_examples), 131  
 ExportStrainCSV (class in G2export\_csv), 132  
 expr (GSASIIexprGUI.ExpressionDialog attribute), 80  
 exprDict (GSASIIobj.ExpressionCalcObj attribute), 20  
 expression (GSASIIobj.ExpressionObj attribute), 21  
 ExpressionCalcObj (class in GSASIIobj), 19  
 ExpressionDialog (class in GSASIIexprGUI), 78  
 ExpressionObj (class in GSASIIobj), 20  
 exprVarLst (GSASIIexprGUI.ExpressionDialog attribute), 80  
 ExtensionValidator() (GSASIIIO.ImportBaseclass method), 70  
 ExtractFileFromZip() (in module GSASIIIO), 69
- ## F
- factorize() (in module GSASIIpwd), 121  
 fcjde\_gen (class in GSASIIpwd), 121  
 FileDlgFixExt() (in module GSASIIIO), 69  
 Fill2ThetaAzimuthMap() (in module GSASIIimage), 95  
 FillAtomLookUp() (in module GSASIImath), 98  
 fillgmat() (in module GSASIIlattice), 39  
 FillMainMenu() (GSASII.GSASII method), 1  
 Filter() (GSASIIctrls.G2MultiChoiceDialog method), 55  
 FindAtomIndexByIDs() (in module GSASIImath), 98  
 findBestCell() (in module GSASIIindex), 113  
 FindBondsDraw() (in module GSASIIphsGUI), 75  
 findOffset() (in module GSASIImath), 106  
 findSSOffset() (in module GSASIImath), 106  
 FitDetector() (in module GSASIIimage), 95  
 FitHKL() (in module GSASIIindex), 113  
 FitHKLTL() (in module GSASIIindex), 113  
 FitHKLZ() (in module GSASIIindex), 113  
 FitHKLZSS() (in module GSASIIindex), 113  
 FitStrain() (in module GSASIIimage), 95  
 FitStrSta() (in module GSASIIimage), 95  
 FixValence() (in module GSASIIElem), 33  
 Flnh() (in module GSASIIlattice), 36  
 fmtVarDescr() (in module GSASIIobj), 24  
 font (gltext.Text attribute), 47  
 font (gltext.TextElement attribute), 48  
 font\_size (gltext.Text attribute), 47  
 foreground (gltext.Text attribute), 47  
 foreground (gltext.TextElement attribute), 48  
 FormatPadValue() (in module GSASIIpy3), 74  
 FormatSigFigs() (in module GSASIIpy3), 74  
 FormatValue() (in module GSASIIpy3), 74  
 FormFactors (module), 48  
 FormulaEval() (in module GSASIIpy3), 74  
 Fourier4DMap() (in module GSASIImath), 98  
 FourierMap() (in module GSASIImath), 98  
 FPcalc() (in module GSASIIElem), 33  
 freeVars (GSASIIobj.ExpressionObj attribute), 21  
 fxnpgdict (GSASIIobj.ExpressionCalcObj attribute), 20
- ## G
- G2CheckBox (class in GSASIIctrls), 52  
 G2ChoiceButton (class in GSASIIctrls), 52  
 G2ColumnIDDialo (class in GSASIIctrls), 53  
 G2export\_CIF (module), 134  
 G2export\_csv (module), 131  
 G2export\_examples (module), 131  
 G2export\_FIT2D (module), 137  
 G2export\_image (module), 133  
 G2export\_map (module), 133  
 G2export\_PDB (module), 132  
 G2export\_pwdr (module), 137  
 G2export\_shelx (module), 133  
 G2HistoDataDialog (class in GSASIIctrls), 53  
 G2HtmlWindow (class in GSASIIctrls), 54  
 G2img\_1TIF (module), 146  
 G2img\_ADSC (module), 146  
 G2img\_CheMin (module), 147  
 G2img\_EDF (module), 146  
 G2img\_GE (module), 146  
 G2img\_MAR (module), 146  
 G2img\_Rigaku (module), 146  
 G2img\_SumG2 (module), 146  
 G2LoggedButton (class in GSASIIctrls), 54  
 G2logList (in module GSASIIlog), 30  
 G2MessageBox() (in module GSASIIctrls), 54

- G2MultiChoiceDialog (class in GSASIIctrls), 54
- G2phase (module), 141
- G2phase\_CIF (module), 142
- G2phase\_GPX (module), 142
- G2Plot3D (class in GSASIIplot), 115
- G2PlotMpl (class in GSASIIplot), 115
- G2PlotNoteBook (class in GSASIIplot), 115
- G2PlotOgl (class in GSASIIplot), 115
- G2pwd\_CIF (module), 143
- G2pwd\_fxyc (module), 143
- G2pwd\_GPX (module), 142
- G2pwd\_xyc (module), 143
- G2sad\_xyc (module), 145
- G2sfact (module), 144
- G2sfact\_CIF (module), 145
- G2SingleChoiceDialog (class in GSASIIctrls), 55
- G2TreeCtrl (class in GSASIIctrls), 55
- G2VarObj (class in GSASIIobj), 21
- G\_matrix() (in module GSASIIIsasd), 123
- GaussCume() (in module GSASIIIsasd), 123
- GaussDist() (in module GSASIIIsasd), 124
- GenAtom() (in module GSASIIspc), 42
- GenerateConstraints() (in module GSASIImapvars), 92
- GenHBravais() (in module GSASIIlattice), 36
- GenHKLf() (in module GSASIIspc), 42
- GenHLaue() (in module GSASIIlattice), 36
- GenPfHKLs() (in module GSASIIlattice), 36
- GenSHCoeff() (in module GSASIIlattice), 37
- GenSSHLaue() (in module GSASIIlattice), 37
- GenWildCard() (in module GSASIIobj), 22
- GetAbsorb() (in module GSASIIstrMath), 82
- GetAbsorbDerv() (in module GSASIIstrMath), 82
- GetAllPhaseData() (in module GSASIIstrIO), 85
- GetAngleSig() (in module GSASIImath), 98
- getAngSig() (in module GSASIImath), 106
- GetAsfMean() (in module GSASIIpwd), 120
- GetAtomCoordsById() (in module GSASIImath), 99
- GetAtomFXU() (in module GSASIIstrMath), 82
- GetAtomInfo() (in module GSASIIElem), 33
- GetAtomItemsById() (in module GSASIImath), 99
- GetAtoms() (GSASIIIO.ExportBaseclass method), 67
- GetAtomsById() (in module GSASIImath), 99
- GetAtomSSFXU() (in module GSASIIstrMath), 83
- getAtomXYZ() (in module GSASIImath), 106
- GetAzm() (in module GSASIIimage), 95
- getBackground() (in module GSASIIpwd), 121
- getBackgroundDerv() (in module GSASIIpwd), 121
- getBackupName() (in module GSASIIstrIO), 88
- GetBLtable() (in module GSASIIElem), 33
- getBLvalues() (in module GSASIIElem), 34
- GetBraviasNum() (in module GSASIIlattice), 37
- GetCell() (GSASIIIO.ExportBaseclass method), 67
- getCellEsd() (in module GSASIIstrIO), 88
- GetConfigValue() (in module GSASIIpath), 27
- GetConstraints() (in module GSASIIstrIO), 85
- GetControls() (in module GSASIIstrIO), 85
- GetCSuinel() (in module GSASIIspc), 42
- GetCSxinel() (in module GSASIIspc), 42
- getCWgam() (in module GSASIImath), 106
- getCWgamDerv() (in module GSASIImath), 106
- getCWsig() (in module GSASIImath), 106
- getCWsigDerv() (in module GSASIImath), 106
- GetData() (GSASIIctrls.G2HistoDataDialog method), 54
- GetData() (GSASIIgrid.AddHatomDialog method), 64
- GetData() (GSASIIgrid.DisAglDialog method), 65
- GetDATSig() (in module GSASIImath), 99
- getDensity() (in module GSASIImath), 107
- GetDependentVars() (in module GSASIImapvars), 92
- getdEpsVoigt() (in module GSASIIpwd), 122
- GetDepVar() (GSASIIexprGUI.ExpressionDialog method), 79
- GetDepVar() (GSASIIobj.ExpressionObj method), 20
- getDescr() (in module GSASIIobj), 24
- GetDetectorXY() (in module GSASIIimage), 95
- GetDetXYfromThAzm() (in module GSASIIimage), 95
- getdFCJVoigt3() (in module GSASIIpwd), 122
- getDistDerv() (in module GSASIImath), 107
- GetDistSig() (in module GSASIImath), 99
- getDmax() (in module GSASIIindex), 113
- getDmin() (in module GSASIIindex), 113
- getdPsVoigt() (in module GSASIIpwd), 122
- GetDsp() (in module GSASIIimage), 95
- GetEdfData() (in module GSASIIIO), 69
- GetEllipse() (in module GSASIIimage), 95
- GetEllipse2() (in module GSASIIimage), 95
- getEpsVoigt() (in module GSASIIpwd), 121
- getFCJVoigt() (in module GSASIIpwd), 121
- getFCJVoigt3() (in module GSASIIpwd), 121
- GetFFC5() (in module GSASIIElem), 33
- GetFFtable() (in module GSASIIElem), 33
- getFFvalues() (in module GSASIIElem), 34
- GetFileList() (GSASII.GSASII method), 1
- GetFobsSq() (in module GSASIIstrMath), 83
- GetFormFactorCoeff() (in module GSASIIElem), 33
- GetFprime() (in module GSASIIstrIO), 86
- getFWHM() (in module GSASIIpwd), 122
- GetG2Image() (in module GSASIIIO), 69
- getgamFW() (in module GSASIIpwd), 122
- GetGESumData() (in module GSASIIIO), 69
- GetHistogramData() (in module GSASIIstrIO), 86
- GetHistogramNames() (GSASII.GSASII method), 1
- GetHistogramNames() (in module GSASIIstrIO), 86
- GetHistogramPhaseData() (in module GSASIIstrIO), 86
- GetHistograms() (in module GSASIIstrIO), 86
- GetHistsLikeSelected() (in module GSASIIpwdGUI), 77
- GetHKLfdatafromTree() (GSASII.GSASII method), 1
- getHKLmax() (in module GSASIIlattice), 40
- getHKLMpeak() (in module GSASIIpwd), 122

- getHKLpeak() (in module GSASIIpwd), 122  
 GetHStrainShift() (in module GSASIIstrMath), 83  
 GetHStrainShiftDerv() (in module GSASIIstrMath), 83  
 GetImageData() (in module GSASIIIO), 69  
 GetImgData() (in module GSASIIIO), 69  
 GetImportFile() (GSASII.GSASII method), 1  
 GetIndependentVars() (GSASIIobj.ExpressionObj method), 20  
 GetIndependentVars() (in module GSASIImapvars), 92  
 GetIntensityCorr() (in module GSASIIstrMath), 83  
 GetIntensityDerv() (in module GSASIIstrMath), 83  
 GetItemOrder() (in module GSASIIctrls), 56  
 GetKcl() (in module GSASIIlattice), 37  
 GetKclKsl() (in module GSASIIlattice), 37  
 GetKNsym() (in module GSASIIspc), 43  
 GetKsl() (in module GSASIIlattice), 37  
 GetMagFormFacCoeff() (in module GSASIIElem), 33  
 GetMAR345Data() (in module GSASIIIO), 70  
 getMass() (in module GSASIImath), 107  
 GetNewCellParms() (in module GSASIIstrMath), 83  
 GetNumDensity() (in module GSASIIpwd), 120  
 GetNXUPQsym() (in module GSASIIspc), 43  
 GetOprPtrName() (in module GSASIIspc), 43  
 GetPatternTreeDataNames() (in module GSASIIgrid), 65  
 GetPatternTreeItemId() (in module GSASIIgrid), 65  
 GetPawleyConstr() (in module GSASIIstrIO), 86  
 getPeakPos() (in module GSASIIlattice), 40  
 getPeakProfile() (in module GSASIIpwd), 122  
 getPeakProfileDerv() (in module GSASIIpwd), 122  
 GetPhaseData() (GSASII.GSASII method), 2  
 GetPhaseData() (in module GSASIIstrIO), 86  
 GetPhaseInfofromTree() (GSASII.GSASII method), 2  
 GetPhaseNames() (GSASII.GSASII method), 2  
 GetPhaseNames() (in module GSASIIstrIO), 86  
 GetPNGData() (in module GSASIIIO), 70  
 GetPowderIparm() (GSASII.GSASII method), 2  
 GetPowderPeaks() (in module GSASIIIO), 70  
 getPowderProfile() (in module GSASIIstrMath), 85  
 getPowderProfileDerv() (in module GSASIIstrMath), 85  
 GetPrefOri() (in module GSASIIstrMath), 83  
 GetPrefOriDerv() (in module GSASIIstrMath), 83  
 getPsVoigt() (in module GSASIIpwd), 122  
 GetPWDRdatafromTree() (GSASII.GSASII method), 2  
 GetPwdrExt() (in module GSASIIstrMath), 83  
 GetPwdrExtDerv() (in module GSASIIstrMath), 83  
 getRamaDerv() (in module GSASIImath), 107  
 GetReflPos() (in module GSASIIstrMath), 83  
 GetReflPosDerv() (in module GSASIIstrMath), 83  
 GetRelativeHistNum() (GSASIIctrls.G2TreeCtrl method), 56  
 GetRelativePhaseNum() (GSASIIctrls.G2TreeCtrl method), 56  
 getRestAngle() (in module GSASIImath), 107  
 getRestChiral() (in module GSASIImath), 107  
 getRestDeriv() (in module GSASIImath), 107  
 getRestDist() (in module GSASIImath), 107  
 getRestPlane() (in module GSASIImath), 107  
 getRestPolefig() (in module GSASIImath), 108  
 getRestPolefigDerv() (in module GSASIImath), 108  
 GetRestrains() (in module GSASIIstrIO), 86  
 getRestRama() (in module GSASIImath), 108  
 getRestTorsion() (in module GSASIImath), 108  
 getRho() (in module GSASIImath), 108  
 GetRigaku() (in module GSASIIIO), 70  
 GetRigidBodies() (in module GSASIIstrIO), 86  
 GetRigidBodyModels() (in module GSASIIstrIO), 87  
 GetSampleSigGam() (in module GSASIIstrMath), 83  
 GetSampleSigGamDerv() (in module GSASIIstrMath), 83  
 GetSelection() (GSASIIctrls.G2ColumnIDDIALOG method), 53  
 GetSelection() (GSASIIctrls.G2SingleChoiceDialog method), 55  
 GetSelections() (GSASIIctrls.G2MultiChoiceDialog method), 55  
 GetSHCoeff() (in module GSASIImath), 99  
 getSyXYZ() (in module GSASIImath), 108  
 getTextElement() (gltext.Text method), 47  
 getTexture() (gltext.Text method), 47  
 getTexture\_size() (gltext.Text method), 47  
 GetTifData() (in module GSASIIIO), 70  
 getTOFalpha() (in module GSASIImath), 108  
 getTOFalphaDerv() (in module GSASIImath), 108  
 getTOFbeta() (in module GSASIImath), 108  
 getTOFbetaDerv() (in module GSASIImath), 109  
 getTOFgamma() (in module GSASIImath), 109  
 getTOFgammaDerv() (in module GSASIImath), 109  
 getTOFsig() (in module GSASIImath), 109  
 getTOFsigDerv() (in module GSASIImath), 109  
 getTorsionDerv() (in module GSASIImath), 109  
 GetTorsionSig() (in module GSASIImath), 99  
 GetTth() (in module GSASIIimage), 95  
 GetTthAzm() (in module GSASIIimage), 95  
 GetTthAzmDsp() (in module GSASIIimage), 95  
 GetTthAzmG() (in module GSASIIimage), 95  
 GetUsedHistogramsAndPhases() (in module GSASIIstrIO), 87  
 GetUsedHistogramsAndPhasesfromTree() (GSASII.GSASII method), 2  
 GetValue() (GSASIIctrls.SingleStringDialog method), 62  
 GetValues() (GSASIIctrls.MultiStringDialog method), 57  
 getVarDescr() (in module GSASIIobj), 24  
 GetVarDocs() (GSASIIctrls.SelectConfigSetting method), 61  
 GetVaried() (GSASIIobj.ExpressionObj method), 20  
 GetVariedVarVal() (GSASIIobj.ExpressionObj method), 20  
 getVCov() (in module GSASIImath), 109



getVersion() (GSASIIctrls.downdate method), 64  
 GetVersionNumber() (in module GSASIIpath), 27  
 getWave() (in module GSASIImath), 110  
 getWidthsCW() (in module GSASIIpwd), 122  
 getWidthsTOF() (in module GSASIIpwd), 122  
 GetXsectionCoeff() (in module GSASIIElem), 34  
 GetXYZDist() (in module GSASIImath), 100  
 Glnh() (in module GSASIIlattice), 37  
 gltext (module), 46  
 Gmat2A() (in module GSASIIlattice), 37  
 Gmat2AB() (in module GSASIIlattice), 37  
 Gmat2cell() (in module GSASIIlattice), 37  
 GPXBackup() (in module GSASIIstrIO), 85  
 GramSchmidtOrtho() (in module GSASIImapvars), 93  
 GridFractionEditor (class in GSASIIctrls), 56  
 GroupConstraints() (in module GSASIImapvars), 93  
 GSAS2\_ReaderClass (class in G2pwd\_GPX), 143  
 GSAS\_ReaderClass (class in G2pwd\_fxyc), 143  
 GSASII (class in GSASII), 1  
 GSASII (module), 1  
 GSASII.CopyDialog (class in GSASII), 1  
 GSASII.SumDialog (class in GSASII), 6  
 GSASIIconstrGUI (module), 76  
 GSASIIctrls (module), 51  
 GSASIIdata (module), 27  
 GSASIIddataGUI (module), 75  
 GSASIIElem (module), 32  
 GSASIIElemGUI (module), 75  
 GSASIIexprGUI (module), 78  
 GSASIIgrid (module), 64  
 GSASIIimage (module), 94  
 GSASIIimgGUI (module), 76  
 GSASIIindex (module), 111  
 GSASIIIO (module), 66  
 GSASIIlattice (module), 34  
 GSASIIlog (module), 30  
 GSASIImain (class in GSASII), 6  
 GSASIImapvars (module), 88  
 GSASIImath (module), 96  
 GSASIIobj (module), 6  
 GSASIIpath (module), 27  
 GSASIIphsGUI (module), 75  
 GSASIIplot (module), 114  
 GSASIIpwd (module), 117  
 GSASIIpwdGUI (module), 76  
 GSASIIpy3 (module), 73  
 GSASIIrestrGUI (module), 78  
 GSASIIsasd (module), 123  
 GSASIIspc (module), 41  
 GSASIIstrIO (module), 85  
 GSASIIstrMain (module), 81  
 GSASIIstrMath (module), 82  
 GSASIItestplot (module), 129  
 GSASIItoolbar (class in GSASIIplot), 115

GSGrid (class in GSASIIctrls), 56  
 GSNoteBook (class in GSASIIctrls), 56

## H

halfCell() (in module GSASIIindex), 113  
 HardSpheresSF() (in module GSASIIsasd), 124  
 Help\_mode (in module config\_example), 32  
 HelpButton (class in GSASIIctrls), 57  
 HessianLSQ() (in module GSASIImath), 100  
 HessRefine() (in module GSASIIstrMath), 83  
 HistIdLookup (in module GSASIIobj), 22  
 HistRanIdLookup (in module GSASIIobj), 22  
 HKLF\_ReaderClass (class in G2sfact), 144  
 HorizontalLine() (in module GSASIIctrls), 57  
 HowDidIgetHere() (in module GSASIIgrid), 65  
 HStrainNames() (in module GSASIIspc), 43  
 Hx2Rh() (in module GSASIIlattice), 38

## I

image: Image data object description, 16  
 image: Image object descriptions, 16  
 ImageCalibrants (module), 49  
 ImageCalibrate() (in module GSASIIimage), 96  
 ImageCompress() (in module GSASIIimage), 96  
 ImageIntegrate() (in module GSASIIimage), 96  
 ImageLocalMax() (in module GSASIIimage), 96  
 ImageRecalibrate() (in module GSASIIimage), 96  
 Import\_directory (in module config\_example), 32  
 ImportBaseclass (class in GSASIIIO), 70  
 ImportBaseclass.ImportException, 70  
 ImportImage (class in GSASIIIO), 71  
 ImportPhase (class in GSASIIIO), 71  
 ImportPowderData (class in GSASIIIO), 71  
 ImportSmallAngleData (class in GSASIIIO), 71  
 ImportStructFactor (class in GSASIIIO), 71  
 IndexAllIds() (in module GSASIIobj), 22  
 IndexParmDict() (in module GSASIIexprGUI), 80  
 IndexPeakListSave() (in module GSASIIIO), 72  
 IndexPeaks() (in module GSASIIindex), 113  
 IndexSSPeaks() (in module GSASIIindex), 113  
 InitExport() (GSASIIIO.ExportBaseclass method), 68  
 InitParameters() (GSASIIIO.ImportImage method), 71  
 InitParameters() (GSASIIIO.ImportStructFactor method), 72  
 InitVars() (in module GSASIImapvars), 93  
 InstallGridToolTip() (GSASIIctrls.GSGrid method), 56  
 InterPrecipitateSF() (in module GSASIIsasd), 124  
 invcell2Gmat() (in module GSASIIlattice), 40  
 InvokeDebugOpts() (in module GSASIIpath), 27  
 InvokeMenuCommand() (in module GSASIIlog), 30  
 invpolcal() (in module GSASIIlattice), 40  
 invQ() (in module GSASIImath), 110  
 IPG() (in module GSASIIsasd), 124  
 IPyBreak() (in module GSASIIpath), 27

IPyBreak\_base() (in module GSASIIpath), 27  
 isBound() (gltext.TextElement method), 48  
 IsHistogramInAnyPhase() (in module GSASIIpwdGUI), 77  
 ISIS\_SXD\_INT\_ReaderClass (class in G2sfact), 144  
 ISODISTORT\_proc() (G2phase\_CIF.CIFPhaseReader method), 142  
 ItemSelector() (in module GSASIIctrls), 57

## J

JANA\_ReaderClass (class in G2phase), 141

## L

lastError (GSASIIobj.ExpressionObj attribute), 21  
 Latt2text() (in module GSASIIspc), 43  
 lblLookup (GSASIIobj.ExpressionCalcObj attribute), 20  
 listLogged (class in GSASIIlog), 31  
 LoadCIFdic() (in module G2export\_CIF), 136  
 LoadConfigFile() (in module GSASIIpath), 28  
 LoadDefaultExpressions() (in module GSASIIexprGUI), 80  
 LoadExercise() (GSASIIctrls.OpenTutorial method), 59  
 LoadExpression() (GSASIIobj.ExpressionObj method), 20  
 LoadImage() (in module GSASIIIO), 72  
 loadParmDict() (GSASIIIO.ExportBaseclass method), 68  
 loadTree() (GSASIIIO.ExportBaseclass method), 68  
 LoadTutorial() (GSASIIctrls.OpenTutorial method), 59  
 LogEntry (class in GSASIIlog), 30  
 logging\_debug (in module config\_example), 32  
 LogInfo (in module GSASIIlog), 30  
 LogNormalCume() (in module GSASIIIsasd), 124  
 LogNormalDist() (in module GSASIIIsasd), 124  
 LogOff() (in module GSASIIlog), 30  
 LogOn() (in module GSASIIlog), 30  
 LogVarChange() (in module GSASIIlog), 30  
 LookupAtomId() (in module GSASIIobj), 23  
 LookupAtomLabel() (in module GSASIIobj), 23  
 LookupHistId() (in module GSASIIobj), 23  
 LookupHistName() (in module GSASIIobj), 23  
 LookupPhaseId() (in module GSASIIobj), 23  
 LookupPhaseName() (in module GSASIIobj), 23  
 LookupWildCard() (in module GSASIIobj), 23  
 LorchWeight() (in module GSASIIpwd), 120  
 LSWCume() (in module GSASIIIsasd), 124  
 LSWDist() (in module GSASIIIsasd), 124

## M

M90\_ReaderClass (class in G2sfact), 144  
 main() (in module GSASII), 6  
 main() (in module GSASIIstrMain), 82  
 main() (in module scanCCD), 129  
 main() (in module testDeriv), 129

Make2ThetaAzimuthMap() (in module GSASIIimage), 96  
 MakeButtonLog() (in module GSASIIlog), 30  
 makeFFTsizeList() (in module GSASIIpwd), 122  
 MakeLSParmDict() (GSASII.GSASII method), 3  
 makeMacApp (module), 129  
 makeMat() (in module GSASIIimage), 96  
 MakePWDRfilename() (GSASIIIO.ExportBaseclass method), 68  
 makeQuat() (in module GSASIImath), 110  
 makeRing() (in module GSASIIimage), 96  
 MakeTabLog() (in module GSASIIlog), 30  
 MakeTreeLog() (in module GSASIIlog), 30  
 MakeUniqueLabel() (in module GSASIIobj), 23  
 Map2Dict() (in module GSASIImapvars), 93  
 marFrame (class in ReadMarCCDFrame), 73  
 MaxEnt\_SB() (in module GSASIIIsasd), 124  
 MaxEntException, 124  
 MaxIndex() (in module GSASIIlattice), 38  
 mcsaSearch() (in module GSASIImath), 110  
 MenuBinding() (GSASII.GSASII method), 3  
 MenuBindingLookup (in module GSASIIlog), 30  
 MenuLogEntry (class in GSASIIlog), 30  
 Modulation() (in module GSASIImath), 100  
 ModulationDerv() (in module GSASIImath), 101  
 monoCellReduce() (in module GSASIIindex), 114  
 MovePatternTreeToGrid() (in module GSASIIgrid), 65  
 MoveToUnitCell() (in module GSASIIspc), 43  
 MT2text() (in module GSASIIspc), 43  
 Muiso2Shkl() (in module GSASIIspc), 43  
 MultiIntegerDialog (class in GSASIIconstrGUI), 76  
 MultipleBlockSelector() (GSASIIIO.ImportBaseclass method), 70  
 MultipleChoicesDialog (class in GSASIIIO), 72  
 MultipleChoicesDialog() (GSASIIIO.ImportBaseclass method), 70  
 MultiStringDialog (class in GSASIIctrls), 57  
 MustrainCoeff() (in module GSASIIspc), 43  
 MustrainNames() (in module GSASIIspc), 43  
 MyHelp (class in GSASIIctrls), 57  
 MyHtmlPanel (class in GSASIIctrls), 58

## N

norm\_gen (class in GSASIIpwd), 122  
 normQ() (in module GSASIImath), 110  
 NT\_HKLF2\_ReaderClass (class in G2sfact), 144  
 NT\_JANA2K\_ReaderClass (class in G2sfact), 144  
 NumberValidator (class in GSASIIctrls), 58

## O

Oblique() (in module GSASIIpwd), 120  
 oddPeak() (in module GSASIIindex), 114  
 OdfChk() (in module GSASIIlattice), 38  
 OmitMap() (in module GSASIImath), 101

- OnAddPhase() (GSASII.GSASII method), 3  
 OnAddRow() (G2export\_CIF.EditCIFpanel method), 135  
 OnApplyChanges() (GSASIIctrls.SelectConfigSetting method), 61  
 OnArrow() (GSASIIplot.GSASIItoolbar method), 115  
 OnBoolSelect() (GSASIIctrls.SelectConfigSetting method), 61  
 OnChange() (GSASIIctrls.SelectConfigSetting method), 61  
 OnChar() (GSASIIctrls.ASCIIValidator method), 51  
 onChar() (GSASIIctrls.G2MultiChoiceDialog method), 55  
 OnChar() (GSASIIctrls.NumberValidator method), 58  
 OnChar() (GSASIIexprGUI.ExpressionDialog method), 79  
 OnCheck() (GSASIIctrls.G2MultiChoiceDialog method), 55  
 OnCheckUpdates() (GSASIIctrls.MyHelp method), 58  
 OnChoice() (GSASIIctrls.OrderBox method), 59  
 OnChoice() (GSASIIexprGUI.ExpressionDialog method), 79  
 OnDataDelete() (GSASII.GSASII method), 3  
 OnDeletePhase() (GSASII.GSASII method), 3  
 OnDepChoice() (GSASIIexprGUI.ExpressionDialog method), 79  
 OnDummyPowder() (GSASII.GSASII method), 3  
 OnFileClose() (GSASII.GSASII method), 3  
 OnFileExit() (GSASII.GSASII method), 3  
 OnFileOpen() (GSASII.GSASII method), 3  
 OnFileSave() (GSASII.GSASII method), 3  
 OnFileSaveas() (GSASII.GSASII method), 3  
 OnHelp() (GSASIIplot.GSASIItoolbar method), 115  
 OnHelpAbout() (GSASIIctrls.MyHelp method), 58  
 OnHelpById() (GSASIIctrls.AddHelp method), 51  
 OnHelpById() (GSASIIctrls.MyHelp method), 58  
 OnImageRead() (GSASII.GSASII method), 3  
 OnImageSum() (GSASII.GSASII method), 3  
 OnImportGeneric() (GSASII.GSASII method), 3  
 OnImportImage() (GSASII.GSASII method), 4  
 OnImportPhase() (GSASII.GSASII method), 4  
 OnImportPowder() (GSASII.GSASII method), 4  
 OnImportSfact() (GSASII.GSASII method), 4  
 OnImportSmallAngle() (GSASII.GSASII method), 4  
 OnInit() (GSASII.GSASIImain method), 6  
 OnKey() (GSASIIplot.GSASIItoolbar method), 115  
 OnKeyDown() (GSASIIctrls.ValidatedTxtCtrl method), 64  
 OnLayoutNeeded() (G2export\_CIF.EditCIFpanel method), 135  
 OnMacroRecordStatus() (GSASII.GSASII method), 4  
 OnMakePDFs() (GSASII.GSASII method), 4  
 OnModeSelect() (GSASIIctrls.OpenTutorial method), 59  
 OnNotebookKey() (GSASIIplot.G2PlotNoteBook method), 115  
 OnOk() (GSASIIgrid.AddHatomDialog method), 64  
 OnOk() (GSASIIgrid.DisAgIDialog method), 65  
 OnPageChanged() (GSASIIplot.G2PlotNoteBook method), 115  
 OnPatternTreeItemActivated() (GSASII.GSASII method), 4  
 OnPatternTreeItemCollapsed() (GSASII.GSASII method), 5  
 OnPatternTreeItemDelete() (GSASII.GSASII method), 5  
 OnPatternTreeItemExpanded() (GSASII.GSASII method), 5  
 OnPatternTreeKeyDown() (GSASII.GSASII method), 5  
 OnPatternTreeSelChanged() (GSASII.GSASII method), 5  
 OnPreferences() (GSASII.GSASII method), 5  
 onPress() (GSASIIctrls.G2LoggedButton method), 54  
 OnPwdrSum() (GSASII.GSASII method), 5  
 OnReadPowderPeaks() (GSASII.GSASII method), 5  
 OnRefine() (GSASII.GSASII method), 5  
 OnRenameData() (GSASII.GSASII method), 5  
 OnReplayPress() (in module GSASIIlog), 31  
 OnReset() (GSASIIgrid.DisAgIDialog method), 65  
 OnSave() (GSASIIctrls.SelectConfigSetting method), 61  
 onSelDir() (GSASIIctrls.SelectConfigSetting method), 61  
 OnSelection() (GSASIIctrls.SelectConfigSetting method), 61  
 onSelectionChanged() (GSASIIctrls.G2TreeCtrl method), 56  
 OnSelectVersion() (GSASIIctrls.MyHelp method), 58  
 OnSeqRefine() (GSASII.GSASII method), 5  
 OnSize() (GSASII.GSASII method), 5  
 OnStartMask() (in module GSASIIplot), 116  
 OnStartNewDzero() (in module GSASIIplot), 116  
 OnTutorialSelected() (GSASIIctrls.OpenTutorial method), 59  
 OnValidate() (GSASIIexprGUI.ExpressionDialog method), 79  
 OpenFile() (GSASIIIO.ExportBaseclass method), 68  
 OpenPowderInstprm() (GSASII.GSASII method), 5  
 OpenTutorial (class in GSASIIctrls), 59  
 Opposite() (in module GSASIIspc), 43  
 OrderBox (class in GSASIIctrls), 59  
 owner\_cnt (gltex.TextElement attribute), 48
- ## P
- Parameter dictionary, 18  
 Parameters (GSASIIIO.ImportStructFactor attribute), 72  
 parmDict (GSASIIexprGUI.ExpressionDialog attribute), 80  
 ParseExpression() (GSASIIobj.ExpressionObj method), 21  
 PDB\_ReaderClass (class in G2phase), 142  
 pdbBreak() (in module GSASIIpath), 28  
 PDFSave() (in module GSASIIIO), 72

PeakListSave() (in module GSASIIIO), 72  
 PeaksEquiv() (in module GSASIImath), 101  
 PeaksUnique() (in module GSASIImath), 101  
 penaltyDeriv() (in module GSASIIstrMath), 85  
 penaltyFxn() (in module GSASIIstrMath), 85  
 peneCorr() (in module GSASIIimage), 96  
 permutations() (in module GSASIIlattice), 40  
 Phase object description, 8  
 PhaseIdLookup (in module GSASIIobj), 24  
 PhaseRanIdLookup (in module GSASIIobj), 24  
 PhaseReaderClass (class in G2phase\_GPX), 142  
 PhaseSelector() (GSASIIIO.ImportPhase method), 71  
 PickElement (class in GSASIIElemGUI), 76  
 PickElements (class in GSASIIElemGUI), 76  
 PickleCIFdict() (in module G2export\_CIF), 136  
 PickTwoDialog (class in GSASIIctrls), 59  
 Plot (class in GSASIItestplot), 129  
 Plot3DSngl() (in module GSASIIplot), 116  
 PlotCalib() (in module GSASIIplot), 116  
 PlotCovariance() (in module GSASIIplot), 116  
 PlotDeltSig() (in module GSASIIplot), 116  
 PlotExposedImage() (in module GSASIIplot), 116  
 PlotImage() (in module GSASIIplot), 116  
 PlotIntegration() (in module GSASIIplot), 116  
 PlotISFG() (in module GSASIIplot), 116  
 PlotNotebook (class in GSASIItestplot), 129  
 PlotPatterns() (in module GSASIIplot), 116  
 PlotPeakWidths() (in module GSASIIplot), 116  
 PlotPowderLines() (in module GSASIIplot), 116  
 PlotRama() (in module GSASIIplot), 116  
 PlotRigidBody() (in module GSASIIplot), 116  
 PlotSelectedSequence() (in module GSASIIplot), 116  
 PlotSizeStrainPO() (in module GSASIIplot), 117  
 PlotSngl() (in module GSASIIplot), 117  
 PlotStrain() (in module GSASIIplot), 117  
 PlotStructure() (in module GSASIIplot), 117  
 PlotTexture() (in module GSASIIplot), 117  
 PlotTorsion() (in module GSASIIplot), 117  
 PlotTRImage() (in module GSASIIplot), 117  
 PlotXY() (in module GSASIIplot), 117  
 pointInPolygon() (in module GSASIIimage), 96  
 Polarization() (in module GSASIIpwd), 120  
 polfcal() (in module GSASIIlattice), 40  
 Pos2dsp() (in module GSASIIlattice), 38  
 Post() (G2export\_CIF.EditCIFtemplate method), 136  
 PostfillDataMenu() (GSASIIgrid.DataFrame method), 64  
 Powder data object description, 13  
 Powder reflection object description, 15  
 PrefillDataMenu() (GSASIIgrid.DataFrame method), 65  
 PreviewFile() (GSASII.GSASII method), 5  
 print\_arr() (in module GSASIIIsasd), 126  
 print\_vec() (in module GSASIIIsasd), 126  
 PrintDistAngle() (in module GSASIIstrMain), 81  
 PrintIndependentVars() (in module GSASIImapvars), 93

PrintRestrains() (in module GSASIIstrIO), 87  
 printRho() (in module GSASIImath), 110  
 ProcessConstraints() (in module GSASIIstrIO), 87  
 prodQQ() (in module GSASIImath), 110  
 prodQVQ() (in module GSASIImath), 110  
 ProjFileOpen() (in module GSASIIIO), 72  
 ProjFileSave() (in module GSASIIIO), 72  
 PutG2Image() (in module GSASIIIO), 72

## Q

Q2AV() (in module GSASIImath), 101  
 Q2AVdeg() (in module GSASIImath), 101  
 Q2Mat() (in module GSASIImath), 101

## R

ran2axis() (in module GSASIIindex), 114  
 ranAbyR() (in module GSASIIindex), 114  
 ranAbyV() (in module GSASIIindex), 114  
 ranaxis() (in module GSASIIindex), 114  
 rancell() (in module GSASIIindex), 114  
 randomAVdeg() (in module GSASIImath), 110  
 randomQ() (in module GSASIImath), 110  
 ReadCheckConstraints() (in module GSASIIstrIO), 87  
 ReadCIF() (in module GSASIIIO), 72  
 Reader() (G2img\_1TIF.TIF\_ReaderClass method), 147  
 Reader() (G2phase.EXP\_ReaderClass method), 141  
 Reader() (G2phase.JANA\_ReaderClass method), 142  
 Reader() (G2phase.PDB\_ReaderClass method), 142  
 Reader() (G2phase\_GPX.PhaseReaderClass method), 142  
 Reader() (G2pwd\_CIF.CIFpwdReader method), 143  
 Reader() (G2pwd\_fxye.GSAS\_ReaderClass method), 143  
 Reader() (G2pwd\_GPX.GSAS2\_ReaderClass method), 143  
 Reader() (G2pwd\_xye.xye\_ReaderClass method), 143  
 Reader() (G2sfact.HKLF\_ReaderClass method), 144  
 Reader() (G2sfact.ISIS\_SXD\_INT\_ReaderClass method), 144  
 Reader() (G2sfact.M90\_ReaderClass method), 144  
 Reader() (G2sfact.NT\_HKLF2\_ReaderClass method), 144  
 Reader() (G2sfact.NT\_JANA2K\_ReaderClass method), 144  
 Reader() (G2sfact.SHELX4\_ReaderClass method), 145  
 Reader() (G2sfact.SHELX5\_ReaderClass method), 145  
 Reader() (G2sfact\_CIF.CIFhkIReader method), 145  
 ReadEXPPPhase() (G2phase.EXP\_ReaderClass method), 141  
 ReadImageData() (in module GSASIIIO), 73  
 ReadJANAPhase() (G2phase.JANA\_ReaderClass method), 142  
 ReadLoadImage() (in module GSASIIIO), 73  
 ReadMarCCDFrame (module), 73



- ReadPDBPhase() (G2phase.PDB\_ReaderClass method), 142
- ReadPowderInstprm() (GSASII.GSASII method), 5
- ReadPowderIparm() (GSASII.GSASII method), 5
- Refine() (in module GSASIIstrMain), 81
- RefineCore() (in module GSASIIstrMain), 81
- refinePeaks() (in module GSASIIindex), 114
- refinePeaksT() (in module GSASIIindex), 114
- refinePeaksZ() (in module GSASIIindex), 114
- refinePeaksZSS() (in module GSASIIindex), 114
- ReInitialize() (GSASIIIO.ImportBaseclass method), 71
- ReInitialize() (GSASIIIO.ImportImage method), 71
- ReInitialize() (GSASIIIO.ImportPowderData method), 71
- ReInitialize() (GSASIIIO.ImportSmallAngleData method), 71
- ReInitialize() (GSASIIIO.ImportStructFactor method), 72
- release() (gltext.TextElement method), 48
- Rename() (GSASIIplot.G2PlotNoteBook method), 115
- Repaint() (GSASIIexprGUI.ExpressionDialog method), 79
- Repaint() (GSASIIlog.TabLogEntry method), 31
- Repaint() (GSASIIlog.TreeLogEntry method), 31
- Replay() (GSASIIlog.MenuLogEntry method), 30
- Replay() (GSASIIlog.TabLogEntry method), 31
- Replay() (GSASIIlog.TreeLogEntry method), 31
- Replay() (GSASIIlog.VarLogEntry method), 31
- ReplayLog() (in module GSASIIlog), 31
- RestartTimer() (GSASIIexprGUI.ExpressionDialog method), 79
- RetDistAngle() (in module GSASIIstrMain), 81
- reVarDesc (in module GSASIIobj), 24
- Rh2Hx() (in module GSASIIlattice), 38
- Rigid Body Data description, 10
- RotateRBXYZ() (in module GSASIImath), 101
- rotdMat() (in module GSASIIlattice), 40
- rotdMat4() (in module GSASIIlattice), 40
- rotOrthoA() (in module GSASIIindex), 114
- Ruland() (in module GSASIIpwd), 120
- RunPython() (in module makeMacApp), 130
- ## S
- SamAng() (in module GSASIIlattice), 38
- SaveIntegration() (in module GSASIIIO), 73
- SaveMenuCommand() (in module GSASIIlog), 31
- scaleAbyV() (in module GSASIIindex), 114
- scanCCD (module), 129
- ScatFac() (in module GSASIIElem), 34
- SCExtinction() (in module GSASIIstrMath), 83
- SchulzZimmCume() (in module GSASIIsasd), 125
- SchulzZimmDist() (in module GSASIIsasd), 125
- ScrolledMultiEditor (class in GSASIIctrls), 60
- SearchMap() (in module GSASIImath), 101
- sec2HMS() (in module GSASIIlattice), 40
- SelectConfigSetting (class in GSASIIctrls), 61
- SelectDownloadLoc() (GSASIIctrls.OpenTutorial method), 59
- SelectEdit1Var() (in module GSASIIctrls), 61
- SelectG2var() (GSASIIexprGUI.ExpressionDialog method), 79
- selections() (in module GSASIIlattice), 40
- selftestlist (in module GSASIIlattice), 40
- selftestlist (in module GSASIIspc), 46
- SeqRefine() (in module GSASIIstrMain), 82
- SetBackgroundParms() (in module GSASIIpwd), 120
- setCentered() (gltext.Text method), 47
- SetConfigValue() (in module GSASIIpath), 28
- SetCopyNames() (in module GSASIIpwdGUI), 77
- SetDataMenuBar() (in module GSASIIgrid), 65
- SetDefaultSample() (in module GSASIIpwdGUI), 77
- SetDefaultSASDModel() (in module GSASIIpwdGUI), 77
- SetDefaultSubstances() (in module GSASIIpwdGUI), 77
- SetDepVar() (GSASIIobj.ExpressionObj method), 21
- setEvalResult() (GSASIIexprGUI.ExpressionDialog method), 80
- setFont() (gltext.Text method), 47
- setFont\_size() (gltext.Text method), 47
- setForeground() (gltext.Text method), 47
- SetHistogramData() (in module GSASIIstrIO), 87
- SetHistogramPhaseData() (in module GSASIIstrIO), 87
- SetMolCent() (in module GSASIImath), 102
- SetNewPhase() (in module GSASIIIO), 73
- setPeakparms() (in module GSASIImath), 110
- SetPhaseData() (in module GSASIIstrIO), 87
- SetRange() (GSASIIctrls.G2MultiChoiceDialog method), 55
- SetRigidBodyModels() (in module GSASIIstrIO), 87
- SetSelections() (GSASIIctrls.G2MultiChoiceDialog method), 55
- SetSeqResult() (in module GSASIIstrIO), 87
- setText() (gltext.Text method), 47
- SetupCalc() (GSASIIobj.ExpressionCalcObj method), 19
- SetupSampleLabels() (in module GSASIIpwdGUI), 77
- SetUsedHistogramsAndPhases() (in module GSASIIstrIO), 87
- SetVersionNumber() (in module GSASIIpath), 28
- sfloat() (in module G2pwd\_fxyc), 143
- sfloat() (in module GSASIIIO), 73
- SGErrors() (in module GSASIIspc), 43
- SGMessageBox (class in GSASIIgrid), 65
- SGpolar() (in module GSASIIspc), 44
- SGPrint() (in module GSASIIspc), 43
- SGProd() (in module GSASIIspc), 43
- SGPtGroup() (in module GSASIIspc), 44
- SHELX4\_ReaderClass (class in G2sfact), 145
- SHELX5\_ReaderClass (class in G2sfact), 145

- ShortHistNames (in module GSASIIobj), 24
  - ShortPhaseNames (in module GSASIIobj), 24
  - Show() (GSASIIctrls.MultiStringDialog method), 57
  - Show() (GSASIIctrls.SingleStringDialog method), 62
  - Show() (GSASIIexprGUI.ExpressionDialog method), 79
  - Show() (GSASIIgrid.SGMessageBox method), 65
  - ShowBanner() (in module GSASIIstrIO), 88
  - ShowControls() (in module GSASIIstrIO), 88
  - showError() (GSASIIexprGUI.ExpressionDialog method), 80
  - ShowHelp() (in module GSASIIctrls), 62
  - ShowLogStatus() (in module GSASIIIlog), 31
  - ShowLSParms (class in GSASIIgrid), 65
  - ShowLSParms() (GSASII.GSASII method), 6
  - ShowStringValidity() (GSASIIctrls.ValidatedTxtCtrl method), 64
  - ShowTutorialPath() (GSASIIctrls.OpenTutorial method), 59
  - ShowValidity() (GSASIIctrls.NumberValidator method), 59
  - ShowWebPage() (in module GSASIIctrls), 62
  - SHPOcal() (in module GSASIIstrMath), 83
  - SHPOcalDerv() (in module GSASIIstrMath), 84
  - SHTXcal() (in module GSASIIstrMath), 84
  - SHTXcalDerv() (in module GSASIIstrMath), 84
  - Single Crystal data object description, 15
  - Single Crystal reflection object description, 16
  - SingleFloatDialog (class in GSASIIctrls), 62
  - SingleStringDialog (class in GSASIIctrls), 62
  - sint() (in module G2pwd\_fxye), 143
  - sint() (in module GSASIIIO), 73
  - sortArray() (in module GSASIImath), 111
  - sortHKLd() (in module GSASIIlattice), 40
  - sortM20() (in module GSASIIindex), 114
  - Space Group Data description, 11
  - SpaceGroup() (in module GSASIIspc), 45
  - SpcGroup() (in module GSASIIspc), 45
  - spglist (in module GSASIIspc), 46
  - SphereFF() (in module GSASIIIsasd), 125
  - SphereVol() (in module GSASIIIsasd), 125
  - SpheroidFF() (in module GSASIIIsasd), 125
  - SpheroidVol() (in module GSASIIIsasd), 125
  - splitSSym() (in module GSASIIspc), 46
  - SquareWellSF() (in module GSASIIIsasd), 125
  - SSChargeFlip() (in module GSASIImath), 101
  - ssdict (in module GSASIIspc), 46
  - SSGModCheck() (in module GSASIIspc), 44
  - SSGPrint() (in module GSASIIspc), 44
  - SSLatt2text() (in module GSASIIspc), 44
  - SSMT2text() (in module GSASIIspc), 44
  - SSpaceGroup() (in module GSASIIspc), 44
  - SSpcGroup() (in module GSASIIspc), 44
  - SStructureFactor2() (in module GSASIIstrMath), 84
  - SStructureFactorDerv() (in module GSASIIstrMath), 84
  - StandardizeSpcName() (in module GSASIIspc), 45
  - Starting\_directory (in module config\_example), 32
  - StartProject() (GSASII.GSASII method), 6
  - StickyHardSpheresSF() (in module GSASIIIsasd), 125
  - StoreEquivalence() (in module GSASIImapvars), 93
  - StringOpsProd() (in module GSASIIspc), 45
  - StripIndents() (in module GSASIIctrls), 62
  - StructureFactor2() (in module GSASIIstrMath), 84
  - StructureFactorDerv() (in module GSASIIstrMath), 84
  - Substances (module), 126
  - Superspace Group Data description, 11
  - SurfaceRough() (in module GSASIIpwd), 120
  - SurfaceRoughDerv() (in module GSASIIpwd), 120
  - svnFindLocalChanges() (in module GSASIIpath), 28
  - svnGetLog() (in module GSASIIpath), 28
  - svnGetRev() (in module GSASIIpath), 28
  - svnInstallDir() (in module GSASIIpath), 29
  - svnSwitchDir() (in module GSASIIpath), 29
  - svnUpdateDir() (in module GSASIIpath), 29
  - svnUpdateProcess() (in module GSASIIpath), 29
  - svnUpgrade() (in module GSASIIpath), 29
  - svnVersion() (in module GSASIIpath), 29
  - svnVersionNumber() (in module GSASIIpath), 29
  - SwapIndx() (in module GSASIIlattice), 38
  - swapMonoA() (in module GSASIIindex), 114
  - SymOpDialog (class in GSASIIgrid), 65
  - SytSym() (in module GSASIIspc), 45
- ## T
- Table (class in GSASIIctrls), 62
  - TabLogEntry (class in GSASIIIlog), 31
  - test0() (in module GSASIIspc), 46
  - test1() (in module GSASIIlattice), 41
  - test1() (in module GSASIIspc), 46
  - test2() (in module GSASIIlattice), 41
  - test2() (in module GSASIIspc), 46
  - test3() (in module GSASIIlattice), 41
  - test3() (in module GSASIIspc), 46
  - test4() (in module GSASIIlattice), 41
  - test5() (in module GSASIIlattice), 41
  - test6() (in module GSASIIlattice), 41
  - test7() (in module GSASIIlattice), 41
  - test8() (in module GSASIIlattice), 41
  - test9() (in module GSASIIlattice), 41
  - test\_GSASIIlattice() (in module unit\_tests), 130
  - test\_GSASIIspc() (in module unit\_tests), 130
  - TestData() (in module GSASIIindex), 113
  - TestData() (in module GSASIIpwd), 120
  - testDeriv (module), 129
  - TestValid() (GSASIIctrls.ASCIIValidator method), 51
  - TestValid() (GSASIIctrls.NumberValidator method), 59
  - Text (class in gltext), 46
  - text (gltext.Text attribute), 47
  - text (gltext.TextElement attribute), 48

- text\_element (gltext.Text attribute), 47  
 TextElement (class in gltext), 47  
 texture (gltext.Text attribute), 47  
 texture (gltext.TextElement attribute), 48  
 texture\_size (gltext.Text attribute), 47  
 texture\_size (gltext.TextElement attribute), 48  
 textureIndex() (in module GSASIIlattice), 41  
 TIF\_ReaderClass (class in G2img\_1TIF), 146  
 TLS2Uij() (in module GSASIImath), 102  
 TOF2dsp() (in module GSASIIlattice), 38  
 TransferFromWindow() (GSASIIctrls.ASCIIValidator method), 51  
 TransferFromWindow() (GSASIIctrls.NumberValidator method), 59  
 TransferToWindow() (GSASIIctrls.ASCIIValidator method), 51  
 TransferToWindow() (GSASIIctrls.NumberValidator method), 59  
 Transmission() (in module GSASIIpwd), 120  
 Transpose (in module config\_example), 32  
 TreeLogEntry (class in GSASIIlog), 31  
 trim() (in module GSASIIIO), 73  
 Tutorial\_location (in module config\_example), 32  
 txt\_CWNeutronReaderClass (class in G2sad\_xye), 145  
 txt\_nmCWNeutronReaderClass (class in G2sad\_xye), 145  
 txt\_nmXRayReaderClass (class in G2sad\_xye), 146  
 txt\_XRayReaderClass (class in G2sad\_xye), 145
- ## U
- U6toUij() (in module GSASIIlattice), 38  
 Uij2betaij() (in module GSASIIlattice), 38  
 Uij2Ueqv() (in module GSASIIlattice), 38  
 UijtoU6() (in module GSASIIlattice), 38  
 UniDiskFF() (in module GSASIIIsasd), 126  
 UniDiskVol() (in module GSASIIIsasd), 126  
 uniqueCombinations() (in module GSASIIlattice), 41  
 UniRodARFF() (in module GSASIIIsasd), 126  
 UniRodARVol() (in module GSASIIIsasd), 126  
 UniRodFF() (in module GSASIIIsasd), 126  
 UniRodVol() (in module GSASIIIsasd), 126  
 UniSphereFF() (in module GSASIIIsasd), 126  
 UniSphereVol() (in module GSASIIIsasd), 126  
 unit\_tests (module), 130  
 UniTubeFF() (in module GSASIIIsasd), 126  
 UniTubeVol() (in module GSASIIIsasd), 126  
 UpdateBackground() (in module GSASIIpwdGUI), 77  
 UpdateConstraints() (in module GSASIIconstrGUI), 76  
 UpdateControls() (in module GSASIIgrid), 65  
 UpdateDDData() (in module GSASIIddataGUI), 75  
 UpdateDict() (GSASIIobj.ExpressionCalcObj method), 19  
 UpdateImageControls() (in module GSASIIimgGUI), 76  
 UpdateIndexPeaksGrid() (in module GSASIIpwdGUI), 77  
 UpdateInstrumentGrid() (in module GSASIIpwdGUI), 77  
 UpdateLimitsGrid() (in module GSASIIpwdGUI), 77  
 UpdateMasks() (in module GSASIIimgGUI), 76  
 UpdateMCSAxyz() (in module GSASIImath), 102  
 UpdateModelsGrid() (in module GSASIIpwdGUI), 77  
 UpdateNotebook() (in module GSASIIgrid), 66  
 UpdateParameters() (GSASIIIO.ImportStructFactor method), 72  
 UpdatePDFGrid() (in module GSASIIpwdGUI), 77  
 UpdatePeakGrid() (in module GSASIIpwdGUI), 77  
 UpdatePhaseData() (in module GSASIIphsGUI), 75  
 UpdatePWHKPlot() (in module GSASIIgrid), 66  
 UpdateRBUIJ() (in module GSASIImath), 102  
 UpdateRBXYZ() (in module GSASIImath), 102  
 UpdateReflectionGrid() (in module GSASIIpwdGUI), 77  
 UpdateRestrains() (in module GSASIIrestrGUI), 78  
 UpdateRigidBodies() (in module GSASIIconstrGUI), 76  
 UpdateSampleGrid() (in module GSASIIpwdGUI), 77  
 UpdateSeqResults() (in module GSASIIgrid), 66  
 UpdateStressStrain() (in module GSASIIimgGUI), 76  
 UpdateSubstanceGrid() (in module GSASIIpwdGUI), 77  
 UpdateUnitCellsGrid() (in module GSASIIpwdGUI), 78  
 UpdateVariedVars() (GSASIIobj.ExpressionObj method), 21  
 UpdateVars() (GSASIIobj.ExpressionCalcObj method), 19  
 usedVars (GSASIIexprGUI.ExpressionDialog attribute), 80
- ## V
- ValEsd() (in module GSASIImath), 102  
 ValidatedTxtCtrl (class in GSASIIctrls), 62  
 ValidateTutorialDir() (GSASIIctrls.OpenTutorial method), 59  
 Values2A() (in module GSASIIindex), 113  
 Values2Dict() (in module GSASIIpwd), 121  
 Values2Dict() (in module GSASIIstrMath), 84  
 VarDesc (in module GSASIIobj), 24  
 VarDescr() (in module GSASIIobj), 24  
 VarKeys() (in module GSASIImapvars), 94  
 VarLogEntry (class in GSASIIlog), 31  
 varLookup (GSASIIobj.ExpressionCalcObj attribute), 20  
 varName (GSASIIexprGUI.ExpressionDialog attribute), 80  
 varname() (GSASIIobj.G2VarObj method), 22  
 varRefflag (GSASIIexprGUI.ExpressionDialog attribute), 80  
 VarRemapShow() (in module GSASIImapvars), 94  
 varSelect (GSASIIexprGUI.ExpressionDialog attribute), 80  
 varValue (GSASIIexprGUI.ExpressionDialog attribute), 80

Vol2Den() (in module GSASIImath), 103

## W

wavekE() (in module GSASIImath), 111

whichsvn() (in module GSASIIpath), 29

Write() (GSASIIIO.ExportBaseclass method), 68

WriteInstFile() (G2export\_pwdr.ExportPowderFXYE method), 137

WriteList() (in module G2export\_csv), 132

Writer() (G2export\_pwdr.ExportPowderFXYE method), 137

wxInspector (in module config\_example), 32

## X

XScattDen() (in module GSASIImath), 103

xye\_ReaderClass (class in G2pwd\_xye), 143