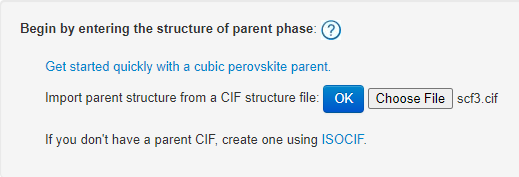
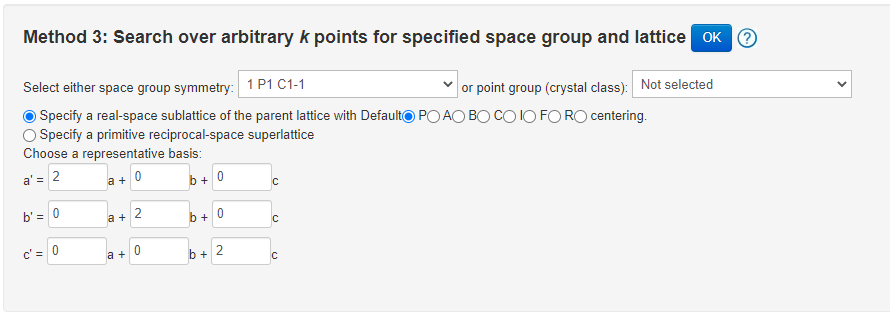
**SAPA Tutorial**

**Generating ISODISTORT CIF**

* Choose input parent structure to be used
* Obtain cif of said parent structure

There are two ways to obtain the ISODISTORT cif:

**Method 1 – Online Method**

* Navigate browser to <https://stokes.byu.edu/iso/isodistort.php>
* Upload cif file here
* Scroll down to method 3
* Select space group symmetry P1
* ****Input supercell transformation
* Click OK
* Click OK again on the new window
* Choose “CIF file” option and click OK

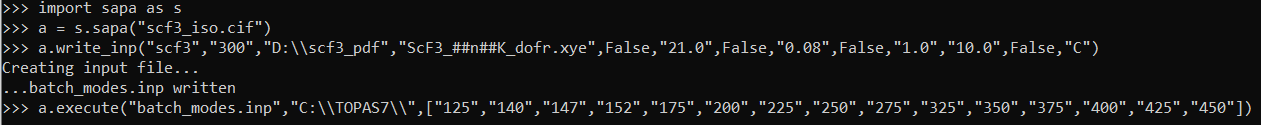
**Method 2 – Python Method**

* Install selenium and webdriver-manager via pip; e.g. open a command prompt with admin privileges and enter “pip install selenium” and “pip install webdriver-manager”
* NB: this process requires google chrome to be installed
* Open a python session
* Import ciftoiso (use from ciftoiso import \*)
* Execute cif\_to\_iso function, with cif file name and supercell basis as arguments
  + Supercell basis is inputted as a list [a1,a2,a3,b1,b2,b3,c1,c2,c3] e.g. for a 2x2x2 supercell, input [2,0,0,0,2,0,0,0,2] – NB – this is the default setting
  + Some errors are expected in the output, but most can be ignored. If there is no error saying something like “failed to find subgroup\_cif.txt” then it has executed correctly

**Generating Input File**

* Open a python session in a directory containing your isodistort cif and the python file “sapa.py” and your PDF data files
* Import the sapa module ( import sapa as s)
* Make a base sapa object – this uses the isodistort cif as an argument – e.g. a = s.sapa(“scf3\_iso.cif”)
* Call the write\_inp function with the relevant arguments. The arguments (in order) are detailed below:
* Arguments for write\_inp are as follows
  + sample – an identifier for this sapa run, e.g. “scf3”
  + filenameformat – data file names should be in a format to allow for easy cycling, e.g. scf3\_125K\_dofr.xye – the portion of the file name being cycled over, e.g. the temp, should be replaced by ##n## - e.g. scf3\_##n##K\_dofr.xye
  + isneutron - = False if using Xray data, = True if using neutron data
  + qmax – qmax value used to generate PDFs (NB: if this changes between different temperatures, then see a later example on defining variables that have fixed values for each temperature)
  + dq – value of non-lorentzian component of dQ damping
  + range = [start\_x, finish\_x] e.g. to start at 1 and finish at 10 angstrom, use range = [1, 10]
  + lattice - = “C” to restrict lattice parameters to be equal, = “T” for a,b to be equal, =”O” for none to be equal
* Optional arguments:
  + cycles – sets number of refinements per irrep at each temperature, default = 300
  + path - sets location of data files, default is the current working directory
  + bin – define bin size of pdf data – default is 0.02 (recommended value in XPDF Gudrun tutorial)
  + isrebin – set to True if you want topas to rebin your data (default False)
  + lor – value for Lorentzian dQ damping (default = 0)
  + alpha – value of alpha for Q dependent bragg peak FWHM (default = 0) – setting this to non-zero values noticeably slows down refinements
  + filename – set filename, default = “batch\_modes.inp”

**Executing Input File**

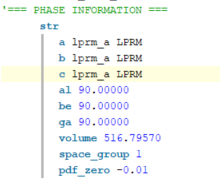
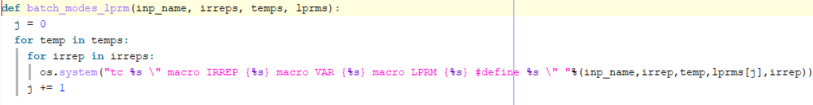
* Open a python session in a directory containing your isodistort cif, your sapa input file and the python file “sapa.py”
* Create a base sapa object as before (or continue on from the previous part)
* Run sapa using the “execute” function – e.g. sapa.execute(args). Arguments are:
  + Temps – a list of temperatures (or other variable). These should be in the same format as they appear in the filename. NB: It is recommended to use numeric values here, so they output files can be easily sorted
* Executing generates an output .txt file in the format IRREP\_sample\_temp\_out.txt for each temperature and irrep containing the values at convergence for every refined parameter for each cycle
* The execution will take some time –the script produces an estimated time remaining, based on the average time elapsed per refinement. Since different irreps can take different lengths of time, this should
* By default, the Topas output to the console is suppressed – this can be turned on by setting the optional verbose argument to True (e.g. a.execute(temps=[…],verbose=True) )
* If the command is being ran in the same session as the write\_inp command, then the script will use the filename and sample designation used there. If not, the execute command will prompt you to input a filename and sample designation.
* The execution script generates a monitoring file which contains each irrep and temperature combination and a flag designating whether topas has finished running that combination. If the script was to stop midway through (via a computer crash for example), this file can be used to restart the sapa run. The restart function takes no arguments, but will prompt for a filename and sample name if they are not already set.

**Running SAPA with fixed variables per temperature**

This example will demonstrate how to edit the input file and write an execution script to run SAPA with variables that are known for each PDF, but change between them. For example, you could have lattice parameters determined via bragg diffraction which you want to fix in SAPA, or a distortion amplitude that is “frozen in”.

**Fixed Lattice Parameters**

This example will show how to fix lattice parameters for a cubic system as a function of temperature, e.g. where you know the correct lattice parameter for each temperature from another refinement.

* Edit the .inp file and replace the numerical values for the lattice parameters with a new keyword – this will get replaced in the command line – e.g. LPRM in the below image
* Define a list of lattice parameters to cycle through
* The below python function can be then used to cycle through the edited input file – the values in the lprms list in python replace LPRM in the input file
* To add further variables, further keywords can be added to the input file. The python function then needs to be edited – for each keyword, “macro KEYWORD {%s}” needs to be added to the argument of os.system after “macro LPRM {%s}” and the remainder of the line needs to be changed to %(inp\_name, irrep, temp, lprms[j], new\_variable[j], irrep))