ARPES Data k-Warping Python Script - Documentation

Rebecca Nicholls, University of Warwick

August 25, 2017

Before using the code ensure you have the following Python modules: numpy, SciPy, h5py, nexusformat, PyQt5. The program is used by running the script NeXus_warping.py. The files included are:

- NeXus_warping_GUI.py
  - This file creates a graphical user interface in the form of a Widget using the QtPy5 module. Variables needed in the main kWarping script are input via the main window. A second dialog box allows the input of paths to the data inside NeXus files if the file's program type is not supported by the program. A third dialog box can be used to display the NeXus tree to view the paths to the datacubes.
- NeXus_warping.py
  - This file contains the main script for warping the NeXus files. It will take in variables from the GUI, creating a new 3D warped datacube of size [kx pixels, ky pixels, E pixels], where kx and ky pixels are user defined. The output is a new NeXus file containing the warped datacube and axes scales (k_x, k_y, E_{kin}, E_{bin}).
- i05-1-3475.nxs
  - Test file which runs with the Program Version 'GDA 8.52.0'
  - Contains file paths
    - '/entry1/instrument/analyser/data'
    - '/entry1/instrument/analyser/energies'
    - '/entry1/instrument/analyser/angles'
    - '/entry1/instrument/anapolar/anapolar'
- i05-1-5370.nxs
  - Test file which runs with the Program Version 'GDA 9.1.0'. High energy graphene data.
  - Contains file paths
    - '/entry1/instrument/analyser/data'
    - '/entry1/instrument/analyser/energies'
    - '/entry1/instrument/analyser/angles'
    - '/entry1/instrument/analyser/analyser_polar_angle'
- i05-1-5470.nxs
  - Second test file which runs with the Program Version 'GDA 9.1.0'.
  - Contains file paths
    - '/entry1/instrument/analyser/data'
    - '/entry1/instrument/analyser/energies'
    - '/entry1/instrument/analyser/angles'
    - '/entry1/instrument/analyser/analyser_polar_angle'

1 NeXus_warping_GUI.py

Upon running the script, the function demoQButton is called, initialising the main window. The main window is defined by 'class First(QWidget)' and will prompt the user to input variables.

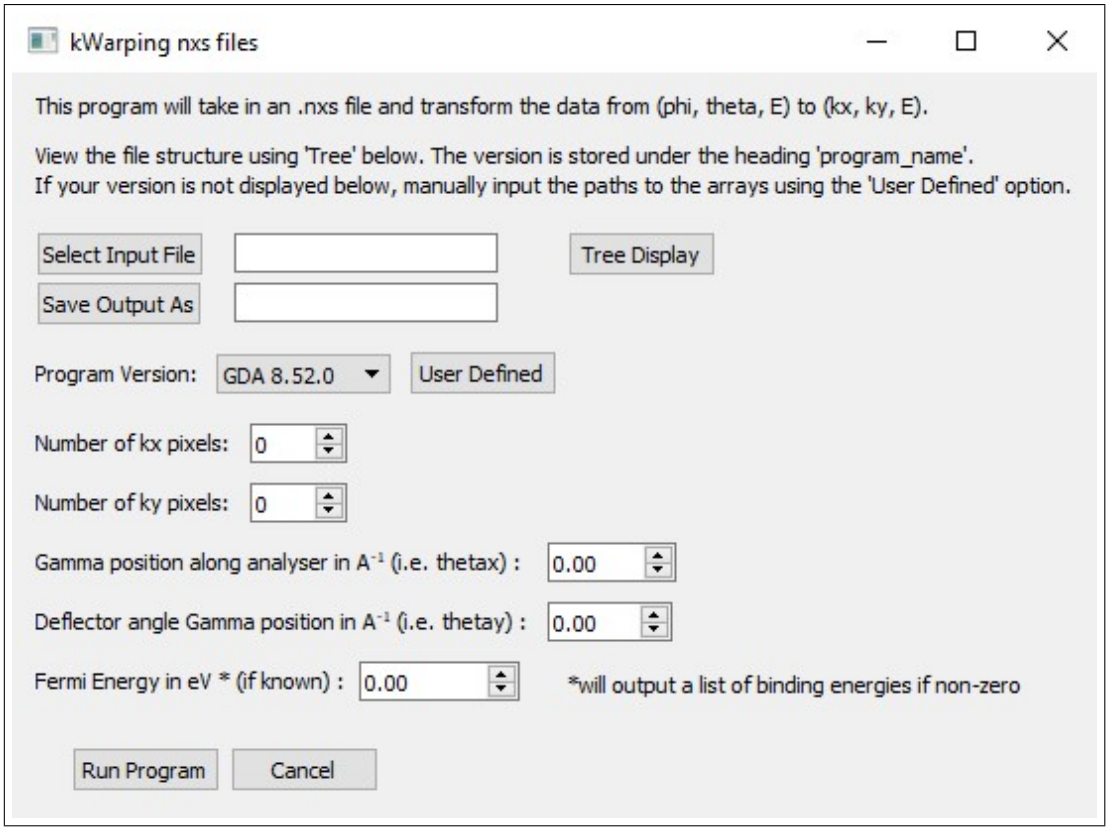


Figure 1: Main window widget of the k-Warping GUI. User inputs are required.

Select Input File : Here the user should select a NeXus file with a valid file structure (see example files). The NeXus file must contain a 3D array of intensity data with axes (ϕ,θ,E_{kin}), a 1D array containing all of the kinetic energies, a 1D array of angles (θ) and a 1D array of the anapolar angles (ϕ).

Save Output As : Here the user should define a file name and destination for the warped NeXus file. Ideally save the file as 'filename.nxs'. If the file is saved only as 'filename', the code should add the file type suffix and it should still run in DAWN. If no file destination is selected, the output will be saved in the same directory as the .py file, with the name 'inputFile_warped.nxs'. Incremental numbering is in place if the same file is warped multiple times, saving to same destination.

Program Version : This will vary by file, and this drop down box (QComboBox) should be adjusted as new program versions are implemented. This will require a minor change in both NeXus_warping_GUI.py and NeXus_warping.py. If the program version is unknown, use the Tree Display tool. If the program version does not match any in the list, insert the paths by hand using the 'User Defined' tool (see example files above for the path style).

Tree Display : If the user does not know the Program Version of their file, it can be displayed here. If there is an attribute in the NeXus file stored under 'program_name', the GUI will display it in a separate box. If not, the box will be left empty and paths will need to be defined using the 'User Defined' option.

User Defined : If the user has an unusual file which does not match any of the given program ver-

sions, they will need to input the paths to the data by hand. The Tree Display tool aids this.

Number of kx pixels : Defines the size of the warped datacube. Should be less than ky pixels.

Number of ky pixels : Defines the size of the warped datacube. Should be greater than kx pixels.

Gamma offsets : These can be left as zero if they are unknown. If required, these will adjust the ranges on the 1D array inputs for θ and ϕ.

Fermi Energy : Can also be left empty. Inputting E_F will give a list of binding energies alongside a list of kinetic energies, with $E_{bin} = E_{kin} - E_F$. Photon energy is neglected here.

Run Program : Passes all of the input variables through to NeXus_warping.py and runs the warping operation. Once clicked, the main window will code and the script will continue to run until the output file has been saved. Stopping the code must be done directly from the console. A printout of 'number of slices warped/ total number of slices' is displayed, so the user knows where in the datacube the code has reached.

Cancel Terminates the code.

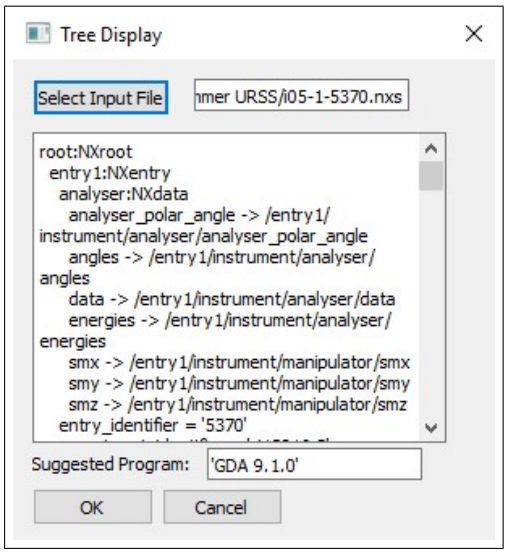


Figure 2: Dialog window from the k-Warping GUI displaying the NeXus tree of the input file i05-1-5370.nxs. The attribute 'program_name' exists, hence a suggested version is displayed at the bottom of the window in a QLineEdit box.

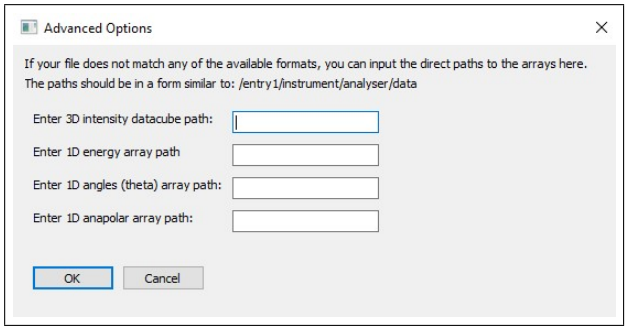
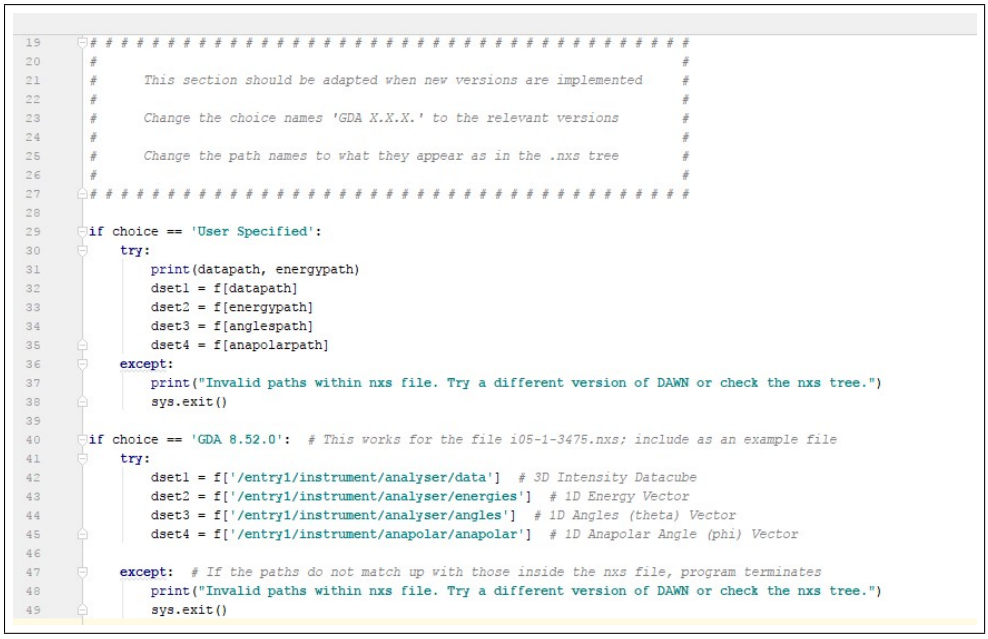


Figure 3: Dialog window from the k-Warping GUI for the input of User Defined paths. Each individual path to the arrays for intensity data, energies, theta and phi must be entered into the QDialog boxes.

2 NeXus_warping.py

This file does the actual warping of the data. The variables from the GUI are passed through to here. The code must be updated when new versions are implemented, see Fig.2. The paths to the datasets contained within the NeXus file must be specified so that the H5PY module can access them. The associated part of the GUI is given in Fig. 2. A new option in the QComboBox (the dropdown list) must be created using the .addItem command.



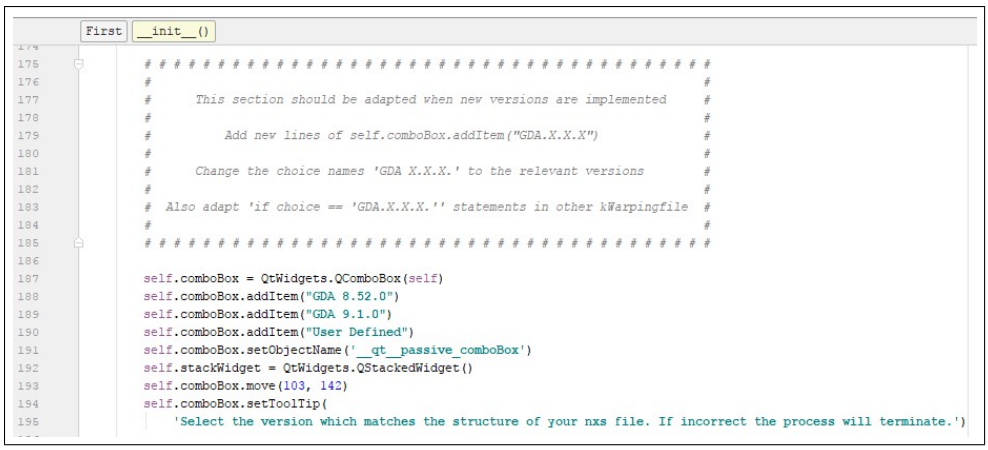
```
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
#                                                                             #
#      This section should be adapted when new versions are implemented       #
#                                                                             #
#      Change the choice names 'GDA X.X.X.' to the relevant versions          #
#                                                                             #
#      Change the path names to what they appear as in the .nxs tree          #
#                                                                             #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

if choice == 'User Specified':
    try:
        print(datapath, energypath)
        dset1 = f[datapath]
        dset2 = f[energypath]
        dset3 = f[anglespath]
        dset4 = f[anapolarpath]
    except:
        print("Invalid paths within nxs file. Try a different version of DAWN or check the nxs tree.")
        sys.exit()

if choice == 'GDA 8.52.0':  # This works for the file i05-1-3475.nxs; include as an example file
    try:
        dset1 = f['/entry1/instrument/analyser/data']  # 3D Intensity Datacube
        dset2 = f['/entry1/instrument/analyser/energies']  # 1D Energy Vector
        dset3 = f['/entry1/instrument/analyser/angles']  # 1D Angles (theta) Vector
        dset4 = f['/entry1/instrument/anapolar/anapolar']  # 1D Anapolar Angle (phi) Vector

    except:  # If the paths do not match up with those inside the nxs file, program terminates
        print("Invalid paths within nxs file. Try a different version of DAWN or check the nxs tree.")
        sys.exit()
```

Figure 4: Update this section of the code when a new version is implemented.



```
First  __init__()

        # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
        #                                                                       #
        #      This section should be adapted when new versions are implemented #
        #                                                                       #
        #          Add new lines of self.comboBox.addItem("GDA.X.X.X")          #
        #                                                                       #
        #      Change the choice names 'GDA X.X.X.' to the relevant versions    #
        #                                                                       #
        #  Also adapt 'if choice == 'GDA.X.X.X.'' statements in other kWarpingfile #
        #                                                                       #
        # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

        self.comboBox = QtWidgets.QComboBox(self)
        self.comboBox.addItem("GDA 8.52.0")
        self.comboBox.addItem("GDA 9.1.0")
        self.comboBox.addItem("User Defined")
        self.comboBox.setObjectName('__qt__passive_comboBox')
        self.stackWidget = QtWidgets.QStackedWidget()
        self.comboBox.move(103, 142)
        self.comboBox.setToolTip(
            'Select the version which matches the structure of your nxs file. If incorrect the process will terminate.')
```

Figure 5: Update this section of the GUI when a new version is implemented.

3 Possible Errors

1. **'Invalid paths within nxs file. Try a different version of DAWN or check the nxs tree.'**
   The most common error will be that the input NeXus file and the Program Version selected are incompatible. If this is the case, the code will terminate. Try checking the version using the Tree Display dialog box. If the 'Suggested Program' box remains empty, or outputs an option not in the drop down list you may have an unusual file. In this case, use the 'User Defined' dialog box to input the paths to the data.

2. **You can't find the output file**.
   If you forgot to define a save path for your output file ('Save Output As'), the output file will have saved to the same directory as the Python file.

3. **'Invalid file. Program terminated.'**
   You have probably input a file that can't be opened by the h5py module, i.e. a non-NeXus file. Ensure your file is a NeXus file (.nxs) with a 3D intensity datacube of the form $[\phi, \theta, E_{kin}]$.