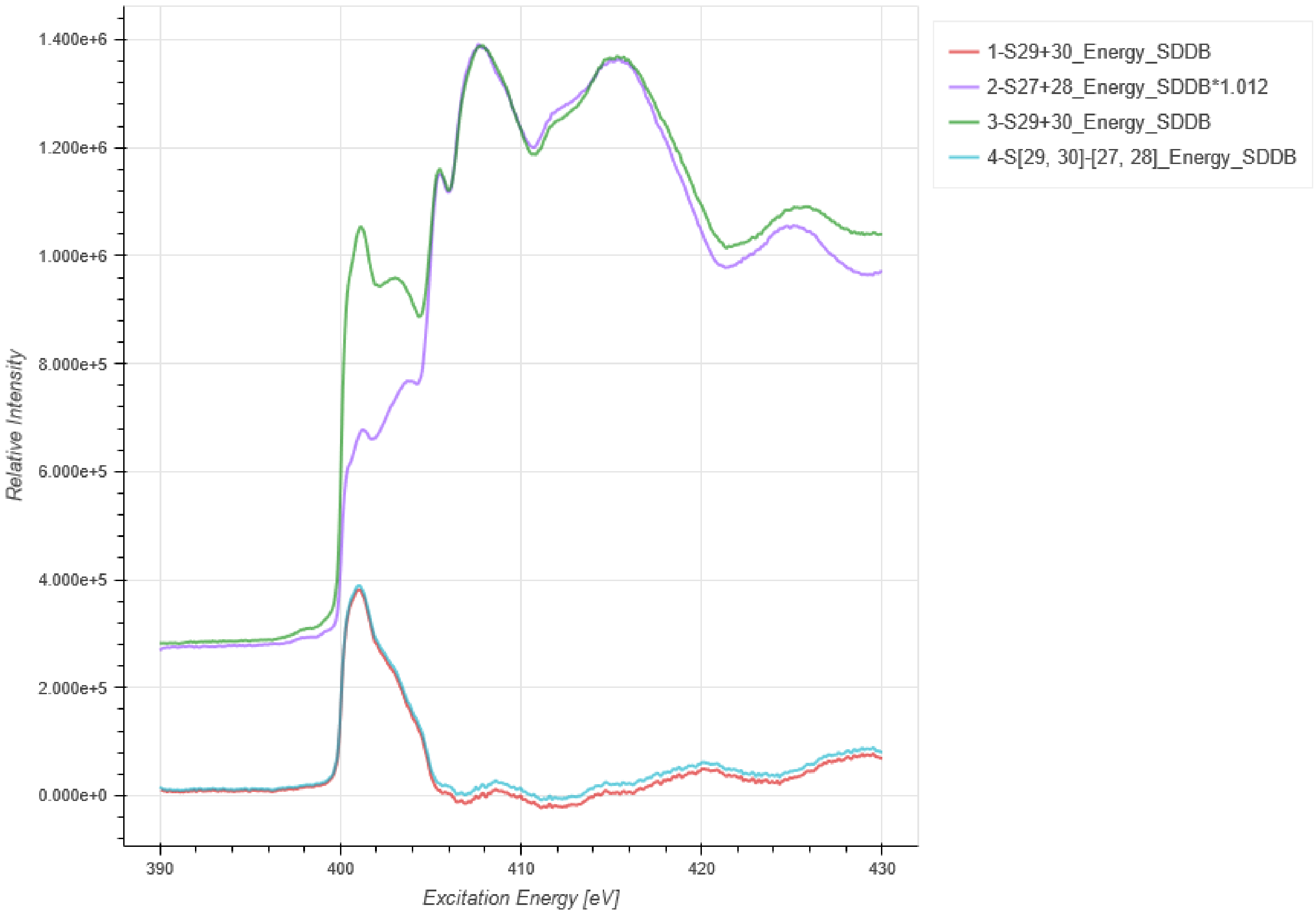
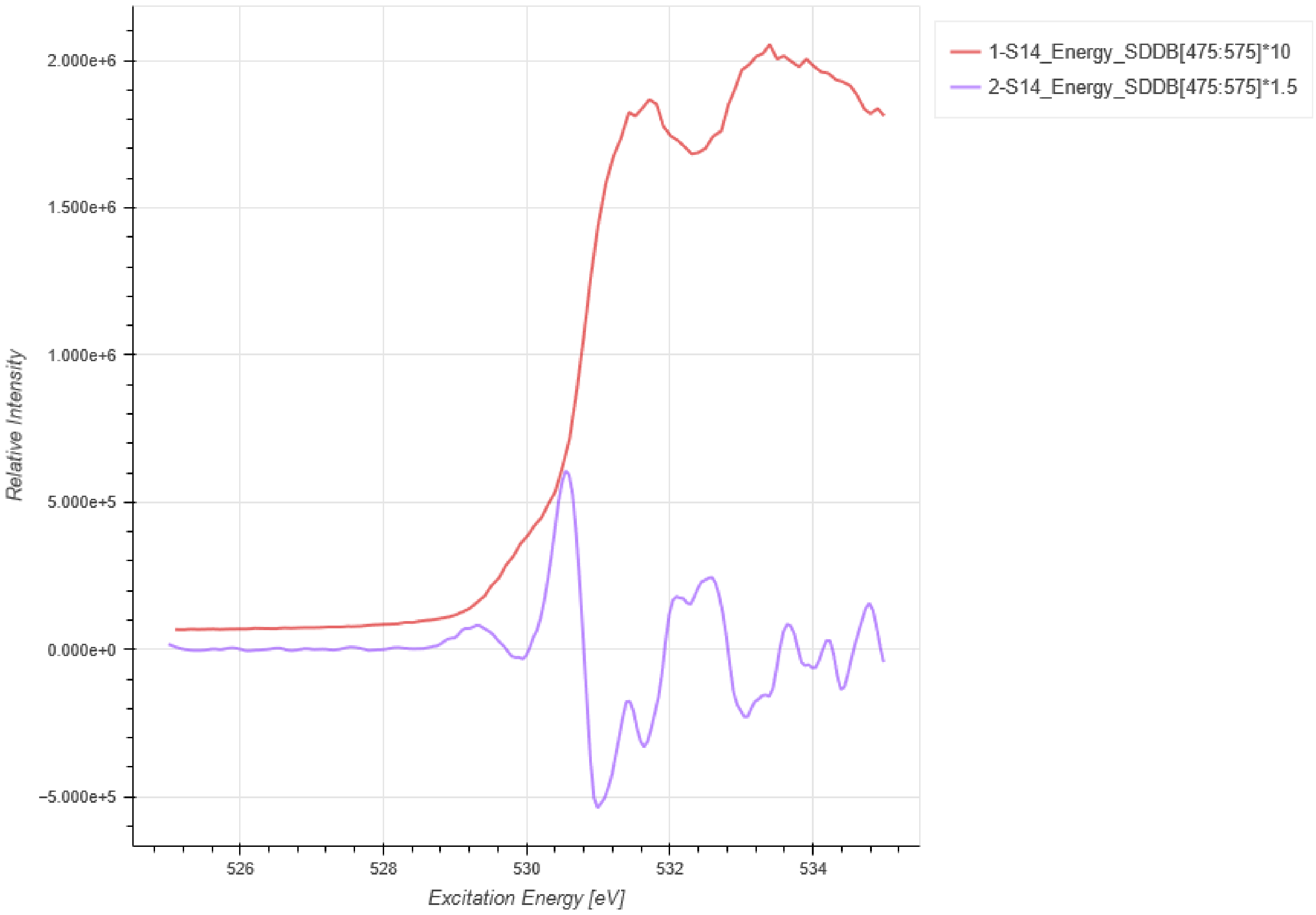
```
## PLOTING X-RAY ABSORPTION SPECTROSCOPY DATA ##
## Creates an object named XAS to load XAS DATA, EITHER TOTAL OR SUMMED OVER SPECIFIC ENERGIES
XAS = XASLoader()

## LOADING/ADDING/SUBSTRACTING 1-D/REDUCED DATA FROM A FILE ##
## Loads XES scans data from HDF5 file
XAS.load(config,'filename', 'y_stream', *args, **kwargs)
## *args = comma seperated list of scans to be plotted or added and then plotted

## Loads and sums XES scans data from HDF5 file
XAS.add(config,'filename', 'y_stream', *args, **kwargs)
## *args = comma seperated list of scans to be plotted or added and then plotted

## Loads and subtratcs XES scans data from HDF5 file
XAS.subtract(config,'filename', 'y_stream', *args, **kwargs)
## *args = s1, p1 -> The data from p1 is subtracted from s1
## *args = [s1, ..., sn], [p1, ..., pn] -> The sum of p1..pn is sub. from the sum s1...sn

## Loads and stitches non-overlapping regions
XAS.stitch(config,'filename', 'xas_y_streamstream', *args, **kwargs)
## *args = comma seperated list of scans to be stitched
## NOTE: The the scans are appended in order, overlap discared

## Loads and subtract scan from all previously loaded scans
XAS.background(config,'filename', 'y_stream', *args, **kwargs)
## *args = s1 -> The scan to be subtracted from all previous load/add/subtract actions
## *args = [s1, ..., sn] -> The sum of scans s1..sn to be subtracted from all previous load/add/subtract actions

## REQUIRED VARIABLES ##
## config = RIXS               -> RIXS Endstation
## config = RSXS               -> RSXS Endstation
## filename = hdf5 file        -> Extension .h5 not needed
## y_stream                  -> SCA detector or sum of MCA type detector
## y_stream[Start:End]       -> sums all MCA data within emission energy range
## y_stream[{S1:E1},{S2,E2}] -> ROI of image detector
## NOTE: Simple math allowed with xes_stream with contstants and variables, i.e. +, -, /, *

## **kwargs ##
## norm = True                       -> Scales the data such that its range is 0 to 1.
## twin_y = True                     -> Adds these plots to a secondary scale
## xoffset = [(S1,P1),...,(SN,PN)]   -> Adjusts x-axis scale to map SN to PN
## xcoffset = value                  -> Shifts x-axis scale by a constant value
## yoffset = [(S1,P1),...,(SN,PN)]   -> Adjusts y-axis scale to map SN to PN
## ycoffset = value                  -> Shifts y-axis scale by a constant value
## grid = [start,stop,delta]       -> Change x-axis grid to be uniform
## savgol = (wind len, poly ord, derv) -> Smooths and takes derivative
## binsize = bins                    -> Bins data, specify the number of points (extra points removed)


## SET RANGE OF Y and X VALUES ##
XAS.xlim(min, max)
XAS.ylim(min, max)
## NOTE: These ranges will be preserved in the data export

## PLOTTING SCAN DATA ##
XAS.plot(**kwargs)

## **kwargs ##
## title = 'New Title of plot' -> Replaces default title with user defined
## xlabel = 'x-axis label'     -> Replaces default x-axis label with user defined
## ylabel = 'y-axis label'     -> Replaces default y-axis label with user defined
## plot_height = value         -> The plot height in points, default is 600
## plot_width = value          -> The plot width in points, default is 900
## norm = True                 -> Normalizes all the data between 0 and 1
## waterfall = offset          -> Normalizes as above and shifts each by the offset

## EXPORTING PLOT DATA ##
XAS.export('filename', **kwargs)

## REQUIRED VARIABLES ##
## filename = filename to be used for ASCII file, do not add extension
## NOTE: Data is exported as it displayed, only options in plotting methods are ignored.

## **kwargs ##
## split_files = True -> Saves each data stream with number appended to the filename
## Compare h-BN Polarization XAS
hBN_XAS = XASLoader()
hBN_XAS.add(RIXS,'HDF5_Notebook', 'SDDB',29,30)
hBN_XAS.background(RIXS,'HDF5_Notebook', 'SDDB*1.012',27,28)
hBN_XAS.add(RIXS,'HDF5_Notebook', 'SDDB*1.012',27,28)
hBN_XAS.add(RIXS,'HDF5_Notebook', 'SDDB',29,30)
hBN_XAS.subtract(RIXS,'HDF5_Notebook', 'SDDB',[29,30],[27,28])
hBN_XAS.plot()
hBN_XAS.export('hBN_XAS', split_files = True)
```



```
## BGO O 1s XAS, both with O Ka ROI and total
BGO_XAS = XASLoader()
BGO_XAS.load(RIXS,'HDF5_Notebook', 'SDDB[475:575]*10',14)
BGO_XAS.load(RIXS,'HDF5_Notebook', 'SDDB[475:575]*1.5',14, grid_x=(525,560, 0.05),savgol = (20,5,2))
BGO_XAS.xlim(525,535)
BGO_XAS.plot()
BGO_XAS.export('BGO_XAS')
```


2.000e+6
1.500e+6
1.000e+6
5.000e+5
0.000e+0
−5.000e+5
Relative Intensity
Excitation Energy [eV]
1-S14_Energy_SDDB[475:575]*10
2-S14_Energy_SDDB[475:575]*1.5