Peak energies for spherical outflow

We use this notebook to generate plots for spherical outflow analysis. This notebook looks at peak energies at different resolutions.

First we import necessary libraries

In [1]:

```
%matplotlib notebook
import processmcrat as pm
import astropy.units as unit
from astropy import constants as const
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
```

We lead the MCRaT output files, and set our mock observations to be $\theta_{\mathrm{obs}}=1^{\circ}$, $\Delta\theta=4^{\circ}$, $r_{\mathrm{obs}}=10^{14}$ cm and framerate = 5 fps. The spectral fit energy range is $10^{-2}-4\times10^{4}$ keV. We set the spectrum dictionaries to be fitted spectra.

In [2]:

```
mcrat_sim5_5=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/spatial-res-levs/fi
mcrat_sim5_5.load_frame(2638, read_stokes=False)
observation5_5=pm.MockObservation(1, 4, 1e14, 5, mcratsimload_obj=mcrat_sim5_5)
observation5_5.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict5_5=observation5_5.spectrum(observation5_5.detected_photons.detection_ti
                                          observation5_5.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim5_4=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/spatial-res-levs/fi
mcrat_sim5_4.load_frame(2638, read_stokes=False)
observation5_4=pm.MockObservation(1, 4, 1e14, 5, mcratsimload_obj=mcrat_sim5_4)
observation5_4.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict5_4=observation5_4.spectrum(observation5_4.detected_photons.detection_ti
                                          observation5_4.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim5_3=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/spatial-res-levs/fi
mcrat_sim5_3.load_frame(2638, read_stokes=False)
observation5_3=pm.MockObservation(1, 4, 1e14, 5, mcratsimload_obj=mcrat_sim5_3)
observation5_3.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict5_3=observation5_3.spectrum(observation5_3.detected_photons.detection_ti
                                          observation5_3.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim5_2=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/spatial-res-levs/fi
mcrat_sim5_2.load_frame(2638, read_stokes=False)
observation5_2=pm.MockObservation(1, 4, 1e14, 5, mcratsimload_obj=mcrat_sim5_2)
observation5_2.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict5_2=observation5_2.spectrum(observation5_2.detected_photons.detection_ti
                                          observation5_2.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim5_1=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/spatial-res-levs/fi
mcrat_sim5_1.load_frame(2638, read_stokes=False)
observation5_1=pm.MockObservation(1, 4, 1e14, 5, mcratsimload_obj=mcrat_sim5_1)
observation5_1.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict5_1=observation5_1.spectrum(observation5_1.detected_photons.detection_ti
                                          observation5_1.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim4_5=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/temporal-res-levs/fi
mcrat_sim4_5.load_frame(1319, read_stokes=False)
observation4_5=pm.MockObservation(1, 4, 1e14, 2.5, mcratsimload_obj=mcrat_sim4_5)
observation4_5.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict4_5=observation4_5.spectrum(observation4_5.detected_photons.detection_ti
                                          observation4_5.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim4_4=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/mixed-res-levs/final
mcrat_sim4_4.load_frame(1319, read_stokes=False)
observation4_4=pm.MockObservation(1, 4, 1e14, 2.5, mcratsimload_obj=mcrat_sim4_4)
observation4_4.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict4_4=observation4_4.spectrum(observation4_4.detected_photons.detection_ti
                                          observation4_4.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim3_5=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/temporal-res-levs/fi
mcrat_sim3_5.load_frame(659, read_stokes=False)
observation3_5=pm.MockObservation(1, 4, 1e14, 1.25, mcratsimload_obj=mcrat_sim3_5)
observation3_5.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict3_5=observation3_5.spectrum(observation3_5.detected_photons.detection_ti
                                          observation3_5.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim3_3=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/mixed-res-levs/final
mcrat_sim3_3.load_frame(659, read_stokes=False)
observation3_3=pm.MockObservation(1, 4, 1e14, 1.25, mcratsimload_obj=mcrat_sim3_3)
observation3_3.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict3_3=observation3_3.spectrum(observation3_3.detected_photons.detection_ti
                                          observation3_3.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim2_5=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/temporal-res-levs/fi
mcrat_sim2_5.load_frame(329, read_stokes=False)
observation2_5=pm.MockObservation(1, 4, 1e14, 0.625, mcratsimload_obj=mcrat_sim2_5)
observation2_5.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict2_5=observation2_5.spectrum(observation2_5.detected_photons.detection_ti
                                          observation2_5.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim2_2=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/mixed-res-levs/final
mcrat_sim2_2.load_frame(329, read_stokes=False)
observation2_2=pm.MockObservation(1, 4, 1e14, 0.625, mcratsimload_obj=mcrat_sim2_2)
observation2_2.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict2_2=observation2_2.spectrum(observation2_2.detected_photons.detection_ti
                                          observation2_2.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim1_5=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/temporal-res-levs/fi
mcrat_sim1_5.load_frame(164, read_stokes=False)
observation1_5=pm.MockObservation(1, 4, 1e14, 0.3125, mcratsimload_obj=mcrat_sim1_5)
observation1_5.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict1_5=observation1_5.spectrum(observation1_5.detected_photons.detection_ti
                                          observation1_5.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)

mcrat_sim1_1=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/spherical-outflow/3000-8000-photons/mixed-res-levs/final
mcrat_sim1_1.load_frame(164, read_stokes=False)
observation1_1=pm.MockObservation(1, 4, 1e14, 0.3125, mcratsimload_obj=mcrat_sim1_1)
observation1_1.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 40000.0])
spectrum_dict1_1=observation1_1.spectrum(observation1_1.detected_photons.detection_ti
                                          observation1_1.detected_photons.detection_time.ma
                                          spectrum_unit=unit.count/unit.s/unit.keV,
                                          fit_spectrum=True,sample_num=1e4)
```

```
/Users/josearita-escalante/opt/anaconda3/lib/python3.8/site-packages/processmcrat/mcli
b.py:41: RuntimeWarning: invalid value encountered in double_scalars
  model[kk]=((alpha-beta)*break_energy)**(alpha-beta)*energies[kk]**(beta)*np.exp(beta
-alpha)
/Users/josearita-escalante/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/mi
npack.py:828: OptimizeWarning: Covariance of the parameters could not be estimated
  warnings.warn('Covariance of the parameters could not be estimated',
```

We now sort them on the type of resolution that is changed.

In [3]:

```
spectrum_dict_spatial=[spectrum_dict5_5, spectrum_dict5_4,
                       spectrum_dict5_3, spectrum_dict5_2,
                       spectrum_dict5_1]

spectrum_dict_temporal=[spectrum_dict5_5, spectrum_dict4_5,
                        spectrum_dict3_5, spectrum_dict2_5,
                        spectrum_dict1_5]

spectrum_dict_mixed=[spectrum_dict5_5, spectrum_dict4_4,
                     spectrum_dict3_3, spectrum_dict2_2,
                     spectrum_dict1_1]
```

We now extract peak energies and and their respective errors.

In [4]:

```
e_pk_spatial=[]
e_pk_err_spatial=[]

e_pk_temporal=[]
e_pk_err_temporal=[]

e_pk_mixed=[]
e_pk_err_mixed=[]


for i in range(5):
    peak_e, peak_e_err = pm.calc_epk_error(spectrum_dict_spatial[i]['fit']['alpha'],
                                           spectrum_dict_spatial[i]['fit']['break_energy
                                           alpha_error=spectrum_dict_spatial[i]['fit_erro
                                           break_energy_error=
                                              spectrum_dict_spatial[i]['fit_errors']['bre
    e_pk_spatial.append(peak_e)
    e_pk_err_spatial.append(peak_e_err)


for i in range(5):
    peak_e, peak_e_err = pm.calc_epk_error(spectrum_dict_temporal[i]['fit']['alpha'],
                                           spectrum_dict_temporal[i]['fit']['break_energy
                                           alpha_error=spectrum_dict_temporal[i]['fit_er
                                           break_energy_error=
                                              spectrum_dict_temporal[i]['fit_errors']['b
    e_pk_temporal.append(peak_e)
    e_pk_err_temporal.append(peak_e_err)


for i in range(5):
    peak_e, peak_e_err = pm.calc_epk_error(spectrum_dict_mixed[i]['fit']['alpha'],
                                           spectrum_dict_mixed[i]['fit']['break_energy'],
                                           alpha_error=spectrum_dict_mixed[i]['fit_errors
                                           break_energy_error=
                                              spectrum_dict_mixed[i]['fit_errors']['break
    e_pk_mixed.append(peak_e)
    e_pk_err_mixed.append(peak_e_err)
```

In [5]:

```
characters=[' ','k','e','V']
for i in range(5):
    for j in characters:

        e_pk_spatial[i]=str(e_pk_spatial[i]).replace(j,'')
        e_pk_err_spatial[i]=str(e_pk_err_spatial[i]).replace(j,'')

        e_pk_temporal[i]=str(e_pk_temporal[i]).replace(j,'')
        e_pk_err_temporal[i]=str(e_pk_err_temporal[i]).replace(j,'')

        e_pk_mixed[i]=str(e_pk_mixed[i]).replace(j,'')
        e_pk_err_mixed[i]=str(e_pk_err_mixed[i]).replace(j,'')

    e_pk_spatial[i]=float(e_pk_spatial[i])
    e_pk_err_spatial[i]=float(e_pk_err_spatial[i])

    e_pk_temporal[i]=float(e_pk_temporal[i])
    e_pk_err_temporal[i]=float(e_pk_err_temporal[i])

    e_pk_mixed[i]=float(e_pk_mixed[i])
    e_pk_err_mixed[i]=float(e_pk_err_mixed[i])

epk_spatial=[]
epk_err_spatial=[]

epk_temporal=[]
epk_err_temporal=[]

epk_mixed=[]
epk_err_mixed=[]

for i in range(1,6):
    epk_spatial.append(e_pk_spatial[-i])
    epk_err_spatial.append(e_pk_err_spatial[-i])

    epk_temporal.append(e_pk_temporal[-i])
    epk_err_temporal.append(e_pk_err_temporal[-i])

    epk_mixed.append(e_pk_mixed[-i])
    epk_err_mixed.append(e_pk_err_mixed[-i])

fps=['0.3125','0.625','1.25','2.5','5']

levs=[1,2,3,4,5]

mix = ['(1,0.3125)','(2,0.625)','(3,1.25)','(4,2.5)','(5,5)']
```

We now plot these quantities for each type of resolution.

In [6]:

```
plt.rcParams.update({'font.size': 20})

label_size = 20
mpl.rcParams['ytick.labelsize'] = label_size


f, axarr = plt.subplots(3, sharex=False, sharey=False)
axarr_spex = axarr

formatter = mpl.ticker.ScalarFormatter(useMathText=True)
formatter.set_scientific(True)
formatter.set_powerlimits((0, 1))

axarr_spex[0].yaxis.set_major_formatter(formatter)

axarr_spex[1].yaxis.set_major_formatter(formatter)

axarr_spex[2].yaxis.set_major_formatter(formatter)

f.set_figwidth(12)
f.set_figheight(15)

axarr_spex[0].scatter(levs,epk_spatial)

axarr_spex[0].errorbar(levs,epk_spatial, epk_err_spatial,
  fmt='none', barsabove=True, ecolor='blue')
axarr_spex[0].set_xticks(range(1,6))

axarr_spex[0].set_ylabel(r'E' + ' ('+
  spectrum_dict_spatial[0]['energy_bin_center'].unit.to_string('latex_inline')+')')

axarr_spex[0].set_xlabel("Spatial Refinement Levels")

axarr_spex[1].scatter(fps,epk_temporal)

axarr_spex[1].errorbar(fps,epk_temporal, epk_err_temporal,
  fmt='none', barsabove=True, ecolor='blue')

axarr_spex[1].set_ylabel(r'E' + ' ('+
  spectrum_dict_temporal[0]['energy_bin_center'].unit.to_string('latex_inline')+')')

axarr_spex[1].set_xlabel("Temporal Refinement Levels (fps)")

axarr_spex[2].scatter(mix,epk_mixed)

axarr_spex[2].errorbar(mix,epk_mixed, epk_err_mixed,
  fmt='none', barsabove=True, ecolor='blue')

axarr_spex[2].set_ylabel(r'E' + ' ('+
  spectrum_dict_mixed[0]['energy_bin_center'].unit.to_string('latex_inline')+')')

axarr_spex[2].set_xlabel("Mixed Refinement Levels (Spatial, Temporal (fps))")

#plt.yscale('log')
axarr_spex[0].annotate('(a)',xy=(0.9, 0.9), xycoords="axes fraction")

axarr_spex[1].annotate('(b)',xy=(0.9, 0.9), xycoords="axes fraction")

axarr_spex[2].annotate('(c)',xy=(0.9, 0.9), xycoords="axes fraction")

plt.tight_layout()
plt.savefig('e_pk_spherical.pdf',dpi=600, bbox_inches='tight')
plt.show()
```