Spectra for HD GRB simulation

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

We first 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 numpy as np
import matplotlib as mpl
```

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.

In [2]:
```
mcrat_sim5_5=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/5fps-lev5/")
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

mcrat_sim5_4=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/5fps-lev4/")
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

mcrat_sim5_3=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/5fps-lev3/")
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

mcrat_sim5_2=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/5fps-lev2/")
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

mcrat_sim5_1=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/5fps-lev1/")
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
mcrat_sim4_5=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/2.5fps-lev5/")
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

mcrat_sim4_4=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/2.5fps-lev4/")
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

mcrat_sim3_5=pm.McratSimLoad("/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/1.2
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

mcrat_sim3_3=pm.McratSimLoad("/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/1.2
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

mcrat_sim2_5=pm.McratSimLoad("/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.6
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

mcrat_sim2_2=pm.McratSimLoad("/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.6
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

mcrat_sim1_5=pm.McratSimLoad("/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.3
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

mcrat_sim1_1=pm.McratSimLoad("/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.3
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
```

We now define a blackbody function such that it outputs a blackbody spectrum to compare to our MCRaT output spectra.

In [3]:
```
def blackbody_function(energies, temp, normalization, energy_unit=unit.keV):
        """
        :param energies:
        :param temp:
        :param normalization:
        :param energy_unit:
        :return:
        """

        energies=energies*energy_unit.to(unit.erg)
        try:
                energies=energies.value
        except AttributeError:
                energies = energies

        try:
                temp=temp.value
        except AttributeError:
                temp = temp

        try:
                normalization=normalization.value
        except AttributeError:
                normalization = normalization


        model =np.empty(energies.size)
        model=(energies**3/(const.h.cgs.value*const.c.cgs.value)**2)/(np.exp(energies/
        energies = energies * unit.erg.to(energy_unit)
        model=model/np.trapz(model,x=energies)*normalization

        return model
```

In [4]:
```
data=blackbody_function(spectrum_dict5_5['energy_bin_center'],
                        1.3e9, np.trapz(spectrum_dict5_5['spectrum'],
                        x=spectrum_dict5_5['energy_bin_center']))

factor_x=(spectrum_dict5_5['energy_bin_center'][data.argmax()]/
          spectrum_dict5_5['energy_bin_center'][spectrum_dict5_5['spectrum'].argmax()
```

```
<ipython-input-3-41b60e3facbd>:28: RuntimeWarning: overflow encountered in exp
  model=(energies**3/(const.h.cgs.value*const.c.cgs.value)**2)/(np.exp(energies/(cons
t.k_B.cgs.value*temp))-1)
```

We now plot our spectra.

In [5]:
```
photon_num_min=10


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

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


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

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


levs=["Spatial Level 1","Spatial Level 2",
      "Spatial Level 3","Spatial Level 4","Spatial Level 5"]

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

mix=["Level 1, 0.3125 fps","Level 2, 0.625 fps",
     "Level 3, 1.25 fps","Level 4, 2.5 fps","Level 5, 5 fps"]

colors=['k','r','b','c','g']

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]

idx_spatial=[]

for i in spectrum_dict_spatial:
    idx_spatial.append(np.where(i['ph_num']>photon_num_min)[0])

#axarr_spex[0].set_xlabel(r'E' + ' ('+spectrum_dict_spatial[0]['energy_bin_center'].u
#                                 fontsize=14)
axarr_spex[0].set_ylabel(r'L$_E$ ('+
  spectrum_dict_spatial[0]['spectrum'][idx_spatial[0]].unit.to_string('latex_inline')
  ')', fontsize=20)

axarr_spex[0].loglog(spectrum_dict_spatial[0]['energy_bin_center'][idx_spatial[0]],
                 spectrum_dict_spatial[0]['spectrum'][idx_spatial[0]], colors[4]+".")
axarr_spex[0].errorbar(spectrum_dict_spatial[0]['energy_bin_center'][idx_spatial[0]],
                   spectrum_dict_spatial[0]['spectrum'][idx_spatial[0]],\
                   yerr=spectrum_dict_spatial[0]['spectrum_errors'][idx_spatial[0]],
                   color=colors[0], marker='o', ls='None',
                   markersize=10, label=levs[4])

axarr_spex[0].loglog(spectrum_dict_spatial[1]['energy_bin_center'][idx_spatial[1]],
                 spectrum_dict_spatial[1]['spectrum'][idx_spatial[1]], colors[3]+".")
axarr_spex[0].errorbar(spectrum_dict_spatial[1]['energy_bin_center'][idx_spatial[1]],
                   spectrum_dict_spatial[1]['spectrum'][idx_spatial[1]],\
                   yerr=spectrum_dict_spatial[1]['spectrum_errors'][idx_spatial[1]],
                   color=colors[1], marker='o', ls='None',
                   markersize=10, label=levs[3])

axarr_spex[0].loglog(spectrum_dict_spatial[2]['energy_bin_center'][idx_spatial[2]],
                 spectrum_dict_spatial[2]['spectrum'][idx_spatial[2]], colors[2]+".")
axarr_spex[0].errorbar(spectrum_dict_spatial[2]['energy_bin_center'][idx_spatial[2]],
                   spectrum_dict_spatial[2]['spectrum'][idx_spatial[2]],\
                   yerr=spectrum_dict_spatial[2]['spectrum_errors'][idx_spatial[2]],
                   color=colors[2], marker='o', ls='None',
                   markersize=10, label=levs[2])


axarr_spex[0].loglog(spectrum_dict_spatial[3]['energy_bin_center'][idx_spatial[3]],
                 spectrum_dict_spatial[3]['spectrum'][idx_spatial[3]], colors[1]+".")
axarr_spex[0].errorbar(spectrum_dict_spatial[3]['energy_bin_center'][idx_spatial[3]],
                   spectrum_dict_spatial[3]['spectrum'][idx_spatial[3]],\
                   yerr=spectrum_dict_spatial[3]['spectrum_errors'][idx_spatial[3]],
                   color=colors[3], marker='o', ls='None',
                   markersize=10, label=levs[1])

axarr_spex[0].loglog(spectrum_dict_spatial[4]['energy_bin_center'][idx_spatial[4]],
                 spectrum_dict_spatial[4]['spectrum'][idx_spatial[4]], colors[0]+".")
axarr_spex[0].errorbar(spectrum_dict_spatial[4]['energy_bin_center'][idx_spatial[4]],
                   spectrum_dict_spatial[4]['spectrum'][idx_spatial[4]],\
                   yerr=spectrum_dict_spatial[4]['spectrum_errors'][idx_spatial[4]],
                   color=colors[4], marker='o', ls='None',
                   markersize=10, label=levs[0])

axarr_spex[0].plot(spectrum_dict5_5['energy_bin_center']*factor_x, data*
  spectrum_dict5_5['spectrum'].max()/data.max(), 'purple',linewidth = 3,zorder=10)

axarr_spex[0].set_xlim(5e-2,2e5)

axarr_spex[0].set_ylim(1e43,1e50)

axarr_spex[0].legend(loc = 'lower center')

idx_temporal=[]

for i in spectrum_dict_temporal:
    idx_temporal.append(np.where(i['ph_num']>photon_num_min)[0])

#axarr_spex[1].set_xlabel(r'E' + ' ('+spectrum_dict_temporal[0]['energy_bin_center'].
#                                 fontsize=14)
axarr_spex[1].set_ylabel(r'L$_E$ ('+
  spectrum_dict_temporal[0]['spectrum'][idx_temporal[0]].unit.to_string('latex_inline
  ')', fontsize=20)

axarr_spex[1].loglog(spectrum_dict_temporal[0]['energy_bin_center'][idx_temporal[0]],
                 spectrum_dict_temporal[0]['spectrum'][idx_temporal[0]], colors[4]+"
axarr_spex[1].errorbar(spectrum_dict_temporal[0]['energy_bin_center'][idx_temporal[0]
                   spectrum_dict_temporal[0]['spectrum'][idx_temporal[0]],\
                   yerr=spectrum_dict_temporal[0]['spectrum_errors'][idx_temporal[0]
                   color=colors[0], marker='o', ls='None',
                   markersize=10, label=fps[4])

axarr_spex[1].loglog(spectrum_dict_temporal[1]['energy_bin_center'][idx_temporal[1]],
                 spectrum_dict_temporal[1]['spectrum'][idx_temporal[1]], colors[3]+"
axarr_spex[1].errorbar(spectrum_dict_temporal[1]['energy_bin_center'][idx_temporal[1]
                   spectrum_dict_temporal[1]['spectrum'][idx_temporal[1]],\
                   yerr=spectrum_dict_temporal[1]['spectrum_errors'][idx_temporal[1]
                   color=colors[1], marker='o', ls='None',
                   markersize=10, label=fps[3])

axarr_spex[1].loglog(spectrum_dict_temporal[2]['energy_bin_center'][idx_temporal[2]],
                 spectrum_dict_temporal[2]['spectrum'][idx_temporal[2]], colors[2]+"
axarr_spex[1].errorbar(spectrum_dict_temporal[2]['energy_bin_center'][idx_temporal[2]
                   spectrum_dict_temporal[2]['spectrum'][idx_temporal[2]],\
                   yerr=spectrum_dict_temporal[2]['spectrum_errors'][idx_temporal[2]
                   color=colors[2], marker='o', ls='None',
                   markersize=10, label=fps[2])


axarr_spex[1].loglog(spectrum_dict_temporal[3]['energy_bin_center'][idx_temporal[3]],
                 spectrum_dict_temporal[3]['spectrum'][idx_temporal[3]], colors[1]+"
axarr_spex[1].errorbar(spectrum_dict_temporal[3]['energy_bin_center'][idx_temporal[3]
                   spectrum_dict_temporal[3]['spectrum'][idx_temporal[3]],\
                   yerr=spectrum_dict_temporal[3]['spectrum_errors'][idx_temporal[3]
                   color=colors[3], marker='o', ls='None',
                   markersize=10, label=fps[1])

axarr_spex[1].loglog(spectrum_dict_temporal[4]['energy_bin_center'][idx_temporal[4]],
                 spectrum_dict_temporal[4]['spectrum'][idx_temporal[4]], colors[0]+"
axarr_spex[1].errorbar(spectrum_dict_temporal[4]['energy_bin_center'][idx_temporal[4]
                   spectrum_dict_temporal[4]['spectrum'][idx_temporal[4]],\
                   yerr=spectrum_dict_temporal[4]['spectrum_errors'][idx_temporal[4]
                   color=colors[4], marker='o', ls='None',
                   markersize=10, label=fps[0])

axarr_spex[1].plot(spectrum_dict5_5['energy_bin_center']*factor_x,
  data*spectrum_dict5_5['spectrum'].max()/data.max(), 'purple',linewidth = 3,zorder=1

#axarr_spex[0].set_xlim(10**-1,3e4)

axarr_spex[1].set_ylim(1e43,1e50)

axarr_spex[1].legend()

idx_mixed=[]

for i in spectrum_dict_mixed:
    idx_mixed.append(np.where(i['ph_num']>photon_num_min)[0])

axarr_spex[2].set_xlabel(r'E' + ' ('+
  spectrum_dict_mixed[0]['energy_bin_center'].unit.to_string('latex_inline')+
    ')', fontsize=20)

axarr_spex[2].set_ylabel(r'L$_E$ ('+
  spectrum_dict_mixed[0]['spectrum'][idx_mixed[0]].unit.to_string('latex_inline')+
  ')', fontsize=20)

axarr_spex[2].loglog(spectrum_dict_mixed[0]['energy_bin_center'][idx_mixed[0]],
                 spectrum_dict_mixed[0]['spectrum'][idx_mixed[0]], colors[4]+".")

axarr_spex[2].errorbar(spectrum_dict_mixed[0]['energy_bin_center'][idx_mixed[0]],
                   spectrum_dict_mixed[0]['spectrum'][idx_mixed[0]],\
                   yerr=spectrum_dict_mixed[0]['spectrum_errors'][idx_mixed[0]],
                   color=colors[0], marker='o', ls='None',
                   markersize=10, label=mix[4])

axarr_spex[2].loglog(spectrum_dict_mixed[1]['energy_bin_center'][idx_mixed[1]],
                 spectrum_dict_mixed[1]['spectrum'][idx_mixed[1]], colors[3]+".")

axarr_spex[2].errorbar(spectrum_dict_mixed[1]['energy_bin_center'][idx_mixed[1]],
                   spectrum_dict_mixed[1]['spectrum'][idx_mixed[1]],\
                   yerr=spectrum_dict_mixed[1]['spectrum_errors'][idx_mixed[1]],
                   color=colors[1], marker='o', ls='None',
                   markersize=10, label=mix[3])

axarr_spex[2].loglog(spectrum_dict_mixed[2]['energy_bin_center'][idx_mixed[2]],
                 spectrum_dict_mixed[2]['spectrum'][idx_mixed[2]], colors[2]+".")

axarr_spex[2].errorbar(spectrum_dict_mixed[2]['energy_bin_center'][idx_mixed[2]],
                   spectrum_dict_mixed[2]['spectrum'][idx_mixed[2]],\
                   yerr=spectrum_dict_mixed[2]['spectrum_errors'][idx_mixed[2]],
                   color=colors[2], marker='o', ls='None',
                   markersize=10, label=mix[2])


axarr_spex[2].loglog(spectrum_dict_mixed[3]['energy_bin_center'][idx_mixed[3]],
                 spectrum_dict_mixed[3]['spectrum'][idx_mixed[3]], colors[1]+".")

axarr_spex[2].errorbar(spectrum_dict_mixed[3]['energy_bin_center'][idx_mixed[3]],
                   spectrum_dict_mixed[3]['spectrum'][idx_mixed[3]],\
                   yerr=spectrum_dict_mixed[3]['spectrum_errors'][idx_mixed[3]],
                   color=colors[3], marker='o', ls='None',
                   markersize=10, label=mix[1])

axarr_spex[2].loglog(spectrum_dict_mixed[4]['energy_bin_center'][idx_mixed[4]],
                 spectrum_dict_mixed[4]['spectrum'][idx_mixed[4]], colors[0]+".")

axarr_spex[2].errorbar(spectrum_dict_mixed[4]['energy_bin_center'][idx_mixed[4]],
                   spectrum_dict_mixed[4]['spectrum'][idx_mixed[4]],\
                   yerr=spectrum_dict_mixed[4]['spectrum_errors'][idx_mixed[4]],
                   color=colors[4], marker='o', ls='None',
                   markersize=10, label=mix[0])

axarr_spex[2].plot(spectrum_dict5_5['energy_bin_center']*factor_x,
  data*spectrum_dict5_5['spectrum'].max()/data.max(), 'purple',linewidth = 3,zorder=1

#axarr_spex[0].set_xlim(10**-1,3e4)

axarr_spex[2].set_ylim(1e43,1e50)

axarr_spex[2].legend(loc = 'lower center')

axarr_spex[0].annotate('(a)',xy=(0.02, 0.9), xycoords="axes fraction")

axarr_spex[1].annotate('(b)',xy=(0.02, 0.9), xycoords="axes fraction")

axarr_spex[2].annotate('(c)',xy=(0.02, 0.9), xycoords="axes fraction")

#plt.title('''spectra vs mixed refinement levels,
#          spherical outflow final frame''')
plt.tight_layout()
#plt.savefig('spectra_science_100s.pdf',dpi=600, bbox_inches = 'tight')
plt.show()
```