Observable properties

We use this notebook to look at different observable properties of the HD GRB simulation. We calculate the deviation from the highest resolution case for each available resolution.

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
from matplotlib import rc
import matplotlib as mpl
import numpy as np
```

Out[1]:
```
'\nPlotting spectra and peak energies vs refinement level for final frame in spherical outflow simulation\n'
```

We load MCRaT output files and create our mock observations and EM property dictionaries.

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)


mcrat_sim5_4=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/5fps-lev4/")
mcrat_sim5_4.load_frame(2638, read_stokes=False)


mcrat_sim5_3=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/5fps-lev3/")
mcrat_sim5_3.load_frame(2638, read_stokes=False)


mcrat_sim5_2=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/5fps-lev2/")
mcrat_sim5_2.load_frame(2638, read_stokes=False)


mcrat_sim5_1=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/5fps-lev1/")
mcrat_sim5_1.load_frame(2638, read_stokes=False)

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)


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)


mcrat_sim4_3=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/2.5fps-lev3/")
mcrat_sim4_3.load_frame(1319, read_stokes=False)


mcrat_sim4_2=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/2.5fps-lev2/")
mcrat_sim4_2.load_frame(1319, read_stokes=False)


mcrat_sim4_1=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/2.5fps-lev1/")
mcrat_sim4_1.load_frame(1319, read_stokes=False)



mcrat_sim3_5=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/1.25fps-lev5/")
mcrat_sim3_5.load_frame(659, read_stokes=False)


mcrat_sim3_4=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/1.25fps-lev4/")
mcrat_sim3_4.load_frame(659, read_stokes=False)


mcrat_sim3_3=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/1.25fps-lev3/")
mcrat_sim3_3.load_frame(659, read_stokes=False)


mcrat_sim3_2=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/1.25fps-lev2/")
mcrat_sim3_2.load_frame(659, read_stokes=False)


mcrat_sim3_1=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/1.25fps-lev1/")
mcrat_sim3_1.load_frame(659, read_stokes=False)


mcrat_sim2_5=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.625fps-lev5/")
mcrat_sim2_5.load_frame(329, read_stokes=False)


mcrat_sim2_4=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.625fps-lev4/")
mcrat_sim2_4.load_frame(329, read_stokes=False)


mcrat_sim2_3=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.625fps-lev3/")
mcrat_sim2_3.load_frame(329, read_stokes=False)


mcrat_sim2_2=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.625fps-lev2/")
mcrat_sim2_2.load_frame(329, read_stokes=False)


mcrat_sim2_1=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.625fps-lev1/")
mcrat_sim2_1.load_frame(329, read_stokes=False)


mcrat_sim1_5=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.3125fps-lev5/")
mcrat_sim1_5.load_frame(164, read_stokes=False)


mcrat_sim1_4=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.3125fps-lev4/")
mcrat_sim1_4.load_frame(164, read_stokes=False)


mcrat_sim1_3=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.3125fps-lev3/")
mcrat_sim1_3.load_frame(164, read_stokes=False)

mcrat_sim1_2=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.3125fps-lev2/")
mcrat_sim1_2.load_frame(164, read_stokes=False)


mcrat_sim1_1=pm.McratSimLoad(
    "/MCRaT-resolution/CHOMBO/science/100-procs-per-angle/0.3125fps-lev1/")
mcrat_sim1_1.load_frame(164, read_stokes=False)
```

In [3]:
```
observation_th1_5_5=pm.MockObservation(1, 4, 1e14, 5, mcratsimload_obj=mcrat_sim5_5)
observation_th1_5_5.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_5_4=pm.MockObservation(1, 4, 1e14, 5, mcratsimload_obj=mcrat_sim5_4)
observation_th1_5_4.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_5_3=pm.MockObservation(1, 4, 1e14, 5, mcratsimload_obj=mcrat_sim5_3)
observation_th1_5_3.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_5_2=pm.MockObservation(1, 4, 1e14, 5, mcratsimload_obj=mcrat_sim5_2)
observation_th1_5_2.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_5_1=pm.MockObservation(1, 4, 1e14, 5, mcratsimload_obj=mcrat_sim5_1)
observation_th1_5_1.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_4_5=pm.MockObservation(1, 4, 1e14, 2.5, mcratsimload_obj=mcrat_sim4_5)
observation_th1_4_5.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_4_4=pm.MockObservation(1, 4, 1e14, 2.5, mcratsimload_obj=mcrat_sim4_4)
observation_th1_4_4.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_4_3=pm.MockObservation(1, 4, 1e14, 2.5, mcratsimload_obj=mcrat_sim4_3)
observation_th1_4_3.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_4_2=pm.MockObservation(1, 4, 1e14, 2.5, mcratsimload_obj=mcrat_sim4_2)
observation_th1_4_2.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_4_1=pm.MockObservation(1, 4, 1e14, 2.5, mcratsimload_obj=mcrat_sim4_1)
observation_th1_4_1.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_3_5=pm.MockObservation(1, 4, 1e14, 1.25, mcratsimload_obj=mcrat_sim3_
observation_th1_3_5.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_3_4=pm.MockObservation(1, 4, 1e14, 1.25, mcratsimload_obj=mcrat_sim3_
observation_th1_3_4.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_3_3=pm.MockObservation(1, 4, 1e14, 1.25, mcratsimload_obj=mcrat_sim3_
observation_th1_3_3.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_3_2=pm.MockObservation(1, 4, 1e14, 1.25, mcratsimload_obj=mcrat_sim3_
observation_th1_3_2.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_3_1=pm.MockObservation(1, 4, 1e14, 1.25, mcratsimload_obj=mcrat_sim3_
observation_th1_3_1.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_2_5=pm.MockObservation(1, 4, 1e14, 0.625, mcratsimload_obj=mcrat_sim2_
observation_th1_2_5.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_2_4=pm.MockObservation(1, 4, 1e14, 0.625, mcratsimload_obj=mcrat_sim2_
observation_th1_2_4.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_2_3=pm.MockObservation(1, 4, 1e14, 0.625, mcratsimload_obj=mcrat_sim2_
observation_th1_2_3.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_2_2=pm.MockObservation(1, 4, 1e14, 0.625, mcratsimload_obj=mcrat_sim2_
observation_th1_2_2.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_2_1=pm.MockObservation(1, 4, 1e14, 0.625, mcratsimload_obj=mcrat_sim2_
observation_th1_2_1.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_1_5=pm.MockObservation(1, 4, 1e14, 0.3125, mcratsimload_obj=mcrat_sim1
observation_th1_1_5.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_1_4=pm.MockObservation(1, 4, 1e14, 0.3125, mcratsimload_obj=mcrat_sim1
observation_th1_1_4.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_1_3=pm.MockObservation(1, 4, 1e14, 0.3125, mcratsimload_obj=mcrat_sim1
observation_th1_1_3.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_1_2=pm.MockObservation(1, 4, 1e14, 0.3125, mcratsimload_obj=mcrat_sim1
observation_th1_1_2.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(


observation_th1_1_1=pm.MockObservation(1, 4, 1e14, 0.3125, mcratsimload_obj=mcrat_sim1
observation_th1_1_1.set_spectral_fit_parameters(spectral_fit_energy_range=[0.01, 4000(
```

In [4]:
```
unfitted_spectrum_dict_th1_5_5=observation_th1_5_5.spectrum(
  observation_th1_5_5.detected_photons.detection_time.min()-1,
  observation_th1_5_5.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_5_4=observation_th1_5_4.spectrum(
  observation_th1_5_4.detected_photons.detection_time.min()-1,
  observation_th1_5_4.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_5_3=observation_th1_5_3.spectrum(
  observation_th1_5_3.detected_photons.detection_time.min()-1,
  observation_th1_5_3.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_5_2=observation_th1_5_2.spectrum(
  observation_th1_5_2.detected_photons.detection_time.min()-1,
  observation_th1_5_2.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_5_1=observation_th1_5_1.spectrum(
  observation_th1_5_1.detected_photons.detection_time.min()-1,
  observation_th1_5_1.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_4_5=observation_th1_4_5.spectrum(
  observation_th1_4_5.detected_photons.detection_time.min()-1,
  observation_th1_4_5.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_4_4=observation_th1_4_4.spectrum(
  observation_th1_4_4.detected_photons.detection_time.min()-1,
  observation_th1_4_4.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_4_3=observation_th1_4_3.spectrum(
  observation_th1_4_3.detected_photons.detection_time.min()-1,
  observation_th1_4_3.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_4_2=observation_th1_4_2.spectrum(
  observation_th1_4_2.detected_photons.detection_time.min()-1,
  observation_th1_4_2.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_4_1=observation_th1_4_1.spectrum(
  observation_th1_4_1.detected_photons.detection_time.min()-1,
  observation_th1_4_1.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_3_5=observation_th1_3_5.spectrum(
  observation_th1_3_5.detected_photons.detection_time.min()-1,
  observation_th1_3_5.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_3_4=observation_th1_3_4.spectrum(
  observation_th1_3_4.detected_photons.detection_time.min()-1,
  observation_th1_3_4.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_3_3=observation_th1_3_3.spectrum(
  observation_th1_3_3.detected_photons.detection_time.min()-1,
  observation_th1_3_3.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_3_2=observation_th1_3_2.spectrum(
  observation_th1_3_2.detected_photons.detection_time.min()-1,
  observation_th1_3_2.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_3_1=observation_th1_3_1.spectrum(
  observation_th1_3_1.detected_photons.detection_time.min()-1,
  observation_th1_3_1.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_2_5=observation_th1_2_5.spectrum(
  observation_th1_2_5.detected_photons.detection_time.min()-1,
  observation_th1_2_5.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_2_4=observation_th1_2_4.spectrum(
  observation_th1_2_4.detected_photons.detection_time.min()-1,
  observation_th1_2_4.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_2_3=observation_th1_2_3.spectrum(
  observation_th1_2_3.detected_photons.detection_time.min()-1,
  observation_th1_2_3.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_2_2=observation_th1_2_2.spectrum(
  observation_th1_2_2.detected_photons.detection_time.min()-1,
  observation_th1_2_2.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_2_1=observation_th1_2_1.spectrum(
  observation_th1_2_1.detected_photons.detection_time.min()-1,
  observation_th1_2_1.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_1_5=observation_th1_1_5.spectrum(
  observation_th1_1_5.detected_photons.detection_time.min()-1,
  observation_th1_1_5.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_1_4=observation_th1_1_4.spectrum(
  observation_th1_1_4.detected_photons.detection_time.min()-1,
  observation_th1_1_4.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_1_3=observation_th1_1_3.spectrum(
  observation_th1_1_3.detected_photons.detection_time.min()-1,
  observation_th1_1_3.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_1_2=observation_th1_1_2.spectrum(
  observation_th1_1_2.detected_photons.detection_time.min()-1,
  observation_th1_1_2.detected_photons.detection_time.max()+1)

unfitted_spectrum_dict_th1_1_1=observation_th1_1_1.spectrum(
  observation_th1_1_1.detected_photons.detection_time.min()-1,
  observation_th1_1_1.detected_photons.detection_time.max()+1)
```

In [5]:
```
fitted_spectrum_dict_th1_5_5=observation_th1_5_5.spectrum(
  observation_th1_5_5.detected_photons.detection_time.min()-1,
  observation_th1_5_5.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_5_4=observation_th1_5_4.spectrum(
  observation_th1_5_4.detected_photons.detection_time.min()-1,
  observation_th1_5_4.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_5_3=observation_th1_5_3.spectrum(
  observation_th1_5_3.detected_photons.detection_time.min()-1,
  observation_th1_5_3.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_5_2=observation_th1_5_2.spectrum(
  observation_th1_5_2.detected_photons.detection_time.min()-1,
  observation_th1_5_2.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_5_1=observation_th1_5_1.spectrum(
  observation_th1_5_1.detected_photons.detection_time.min()-1,
  observation_th1_5_1.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_4_5=observation_th1_4_5.spectrum(
  observation_th1_4_5.detected_photons.detection_time.min()-1,
  observation_th1_4_5.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_4_4=observation_th1_4_4.spectrum(
  observation_th1_4_4.detected_photons.detection_time.min()-1,
  observation_th1_4_4.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_4_3=observation_th1_4_3.spectrum(
  observation_th1_4_3.detected_photons.detection_time.min()-1,
  observation_th1_4_3.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_4_2=observation_th1_4_2.spectrum(
  observation_th1_4_2.detected_photons.detection_time.min()-1,
  observation_th1_4_2.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_4_1=observation_th1_4_1.spectrum(
  observation_th1_4_1.detected_photons.detection_time.min()-1,
  observation_th1_4_1.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_3_5=observation_th1_3_5.spectrum(
  observation_th1_3_5.detected_photons.detection_time.min()-1,
  observation_th1_3_5.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_3_4=observation_th1_3_4.spectrum(
  observation_th1_3_4.detected_photons.detection_time.min()-1,
  observation_th1_3_4.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_3_3=observation_th1_3_3.spectrum(
  observation_th1_3_3.detected_photons.detection_time.min()-1,
  observation_th1_3_3.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_3_2=observation_th1_3_2.spectrum(
  observation_th1_3_2.detected_photons.detection_time.min()-1,
  observation_th1_3_2.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_3_1=observation_th1_3_1.spectrum(
  observation_th1_3_1.detected_photons.detection_time.min()-1,
  observation_th1_3_1.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_2_5=observation_th1_2_5.spectrum(
  observation_th1_2_5.detected_photons.detection_time.min()-1,
  observation_th1_2_5.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_2_4=observation_th1_2_4.spectrum(
  observation_th1_2_4.detected_photons.detection_time.min()-1,
  observation_th1_2_4.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_2_3=observation_th1_2_3.spectrum(
  observation_th1_2_3.detected_photons.detection_time.min()-1,
  observation_th1_2_3.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_2_2=observation_th1_2_2.spectrum(
  observation_th1_2_2.detected_photons.detection_time.min()-1,
  observation_th1_2_2.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_2_1=observation_th1_2_1.spectrum(
  observation_th1_2_1.detected_photons.detection_time.min()-1,
  observation_th1_2_1.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_1_5=observation_th1_1_5.spectrum(
  observation_th1_1_5.detected_photons.detection_time.min()-1,
  observation_th1_1_5.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_1_4=observation_th1_1_4.spectrum(
  observation_th1_1_4.detected_photons.detection_time.min()-1,
  observation_th1_1_4.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_1_3=observation_th1_1_3.spectrum(
  observation_th1_1_3.detected_photons.detection_time.min()-1,
  observation_th1_1_3.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_1_2=observation_th1_1_2.spectrum(
  observation_th1_1_2.detected_photons.detection_time.min()-1,
  observation_th1_1_2.detected_photons.detection_time.max()+1,
  spectrum_unit=unit.count/unit.s/unit.keV,
  fit_spectrum=True,sample_num=1e4)

fitted_spectrum_dict_th1_1_1=observation_th1_1_1.spectrum(
  observation_th1_1_1.detected_photons.detection_time.min()-1,
  observation_th1_1_1.detected_photons.detection_time.max()+1,
  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/mclib.py:41: RuntimeWarning: invalid value encountered in double_scalars
  model[kk]=((alpha-beta)*break_energy)**(alpha-beta)*energies[kk]**(beta)*np.exp(beta-alpha)
```

In [6]:
```
unfitted_spectrum_dict_th1_arr = [[0 for x in range(5)] for y in range(5)]

unfitted_spectrum_dict_th1_arr[4][4] = unfitted_spectrum_dict_th1_5_5

unfitted_spectrum_dict_th1_arr[4][3] = unfitted_spectrum_dict_th1_5_4

unfitted_spectrum_dict_th1_arr[4][2] = unfitted_spectrum_dict_th1_5_3

unfitted_spectrum_dict_th1_arr[4][1] = unfitted_spectrum_dict_th1_5_2

unfitted_spectrum_dict_th1_arr[4][0] = unfitted_spectrum_dict_th1_5_1

unfitted_spectrum_dict_th1_arr[3][4] = unfitted_spectrum_dict_th1_4_5

unfitted_spectrum_dict_th1_arr[3][3] = unfitted_spectrum_dict_th1_4_4

unfitted_spectrum_dict_th1_arr[3][2] = unfitted_spectrum_dict_th1_4_3

unfitted_spectrum_dict_th1_arr[3][1] = unfitted_spectrum_dict_th1_4_2

unfitted_spectrum_dict_th1_arr[3][0] = unfitted_spectrum_dict_th1_4_1

unfitted_spectrum_dict_th1_arr[2][4] = unfitted_spectrum_dict_th1_3_5

unfitted_spectrum_dict_th1_arr[2][3] = unfitted_spectrum_dict_th1_3_4

unfitted_spectrum_dict_th1_arr[2][2] = unfitted_spectrum_dict_th1_3_3

unfitted_spectrum_dict_th1_arr[2][1] = unfitted_spectrum_dict_th1_3_2

unfitted_spectrum_dict_th1_arr[2][0] = unfitted_spectrum_dict_th1_3_1

unfitted_spectrum_dict_th1_arr[1][4] = unfitted_spectrum_dict_th1_2_5

unfitted_spectrum_dict_th1_arr[1][3] = unfitted_spectrum_dict_th1_2_4

unfitted_spectrum_dict_th1_arr[1][2] = unfitted_spectrum_dict_th1_2_3

unfitted_spectrum_dict_th1_arr[1][1] = unfitted_spectrum_dict_th1_2_2

unfitted_spectrum_dict_th1_arr[1][0] = unfitted_spectrum_dict_th1_2_1

unfitted_spectrum_dict_th1_arr[0][4] = unfitted_spectrum_dict_th1_1_5

unfitted_spectrum_dict_th1_arr[0][3] = unfitted_spectrum_dict_th1_1_4

unfitted_spectrum_dict_th1_arr[0][2] = unfitted_spectrum_dict_th1_1_3

unfitted_spectrum_dict_th1_arr[0][1] = unfitted_spectrum_dict_th1_1_2

unfitted_spectrum_dict_th1_arr[0][0] = unfitted_spectrum_dict_th1_1_1
```

In [7]:
```
fitted_spectrum_dict_th1_arr = [[0 for x in range(5)] for y in range(5)]

fitted_spectrum_dict_th1_arr[4][4] = fitted_spectrum_dict_th1_5_5

fitted_spectrum_dict_th1_arr[4][3] = fitted_spectrum_dict_th1_5_4

fitted_spectrum_dict_th1_arr[4][2] = fitted_spectrum_dict_th1_5_3

fitted_spectrum_dict_th1_arr[4][1] = fitted_spectrum_dict_th1_5_2

fitted_spectrum_dict_th1_arr[4][0] = fitted_spectrum_dict_th1_5_1

fitted_spectrum_dict_th1_arr[3][4] = fitted_spectrum_dict_th1_4_5

fitted_spectrum_dict_th1_arr[3][3] = fitted_spectrum_dict_th1_4_4

fitted_spectrum_dict_th1_arr[3][2] = fitted_spectrum_dict_th1_4_3

fitted_spectrum_dict_th1_arr[3][1] = fitted_spectrum_dict_th1_4_2

fitted_spectrum_dict_th1_arr[3][0] = fitted_spectrum_dict_th1_4_1

fitted_spectrum_dict_th1_arr[2][4] = fitted_spectrum_dict_th1_3_5

fitted_spectrum_dict_th1_arr[2][3] = fitted_spectrum_dict_th1_3_4

fitted_spectrum_dict_th1_arr[2][2] = fitted_spectrum_dict_th1_3_3

fitted_spectrum_dict_th1_arr[2][1] = fitted_spectrum_dict_th1_3_2

fitted_spectrum_dict_th1_arr[2][0] = fitted_spectrum_dict_th1_3_1

fitted_spectrum_dict_th1_arr[1][4] = fitted_spectrum_dict_th1_2_5

fitted_spectrum_dict_th1_arr[1][3] = fitted_spectrum_dict_th1_2_4

fitted_spectrum_dict_th1_arr[1][2] = fitted_spectrum_dict_th1_2_3

fitted_spectrum_dict_th1_arr[1][1] = fitted_spectrum_dict_th1_2_2

fitted_spectrum_dict_th1_arr[1][0] = fitted_spectrum_dict_th1_2_1

fitted_spectrum_dict_th1_arr[0][4] = fitted_spectrum_dict_th1_1_5

fitted_spectrum_dict_th1_arr[0][3] = fitted_spectrum_dict_th1_1_4

fitted_spectrum_dict_th1_arr[0][2] = fitted_spectrum_dict_th1_1_3

fitted_spectrum_dict_th1_arr[0][1] = fitted_spectrum_dict_th1_1_2

fitted_spectrum_dict_th1_arr[0][0] = fitted_spectrum_dict_th1_1_1
```

We define the way in which we calculate ζ values.

In [8]:
```
photon_num_min=10

def luminosity(spectrum_dict):
    lum = np.trapz(spectrum_dict['spectrum'][np.where(spectrum_dict['ph_num']>photon_n
    spectrum_dict['energy_bin_center'][np.where(spectrum_dict['ph_num']>photon_num_min
    return lum

def zeta_lum(spectrum_dict, spectrum_dict_high_res):
    lum = luminosity(spectrum_dict)
    lum_high_res = luminosity(spectrum_dict_high_res)
    zeta = (lum - lum_high_res) / lum_high_res
    return zeta

def zeta_e_pk(spectrum_dict, spectrum_dict_high_res):
    peak_e, peak_e_err = pm.calc_epk_error(spectrum_dict['fit']['alpha'],
                                           spectrum_dict['fit']['break_energy'],\
                                           alpha_error=spectrum_dict['fit_errors']['alpha
                                           break_energy_error=
                                              spectrum_dict['fit_errors']['break_energy_
    peak_e_high_res, peak_e_err_high_res = pm.calc_epk_error(spectrum_dict_high_res['
                                           spectrum_dict_high_res['fit']['break_energy'],
                                           alpha_error=spectrum_dict_high_res['fit_errors
                                           break_energy_error=
                                              spectrum_dict_high_res['fit_errors']['break
    zeta = (peak_e - peak_e_high_res) / peak_e_high_res
    return zeta

def zeta_beta(spectrum_dict, spectrum_dict_high_res):
    beta = spectrum_dict['fit']['beta']
    beta_high_res = spectrum_dict_high_res['fit']['beta']
    zeta = (beta - beta_high_res) / beta_high_res
    return zeta

def zeta_alpha(spectrum_dict, spectrum_dict_high_res):
    alpha = spectrum_dict['fit']['alpha']
    alpha_high_res = spectrum_dict_high_res['fit']['alpha']
    zeta = (alpha - alpha_high_res) / alpha_high_res
    return zeta
```

We populate matrices with ζ values

In [9]:
```
z_lum_th1_matrix = [[0 for x in range(5)] for y in range(5)]

for i in range(5):
    for j in range(5):
        z_lum_th1_matrix[i][j]=zeta_lum(unfitted_spectrum_dict_th1_arr[i][j],
                                        unfitted_spectrum_dict_th1_arr[4][4])
```

In [10]:
```
z_e_pk_th1_matrix = [[0 for x in range(5)] for y in range(5)]

for i in range(5):
    for j in range(5):
        z_e_pk_th1_matrix[i][j] = zeta_e_pk(fitted_spectrum_dict_th1_arr[i][j],
                                            fitted_spectrum_dict_th1_arr[4][4])
```

In [11]:
```
z_alpha_th1_matrix = [[0 for x in range(5)] for y in range(5)]

for i in range(5):
    for j in range(5):
        z_alpha_th1_matrix[i][j]=zeta_alpha(fitted_spectrum_dict_th1_arr[i][j],
                                            fitted_spectrum_dict_th1_arr[4][4])
```

In [12]:
```
z_beta_th1_matrix = [[0 for x in range(5)] for y in range(5)]

for i in range(5):
    for j in range(5):
        z_beta_th1_matrix[i][j]=zeta_beta(fitted_spectrum_dict_th1_arr[i][j],
                                          fitted_spectrum_dict_th1_arr[4][4])
```

We create a matrix with the absolute value of all ζ values

In [13]:
```
res_levs = ['1','2','3','4','5']

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

In [14]:
```
z_lum_th1_abs_matrix = [[0 for x in range(5)] for y in range(5)]

for i in range(5):
    for j in range(5):
        z_lum_th1_abs_matrix[i][j] = np.abs(z_lum_th1_matrix[i][j])
```

In [15]:
```
z_e_pk_th1_abs_matrix = [[0 for x in range(5)] for y in range(5)]

for i in range(5):
    for j in range(5):
        z_e_pk_th1_abs_matrix[i][j] = np.abs(z_e_pk_th1_matrix[i][j])
```

In [16]:
```
z_alpha_th1_abs_matrix = [[0 for x in range(5)] for y in range(5)]

for i in range(5):
    for j in range(5):
        z_alpha_th1_abs_matrix[i][j] = np.abs(z_alpha_th1_matrix[i][j])
```

In [17]:
```
z_beta_th1_abs_matrix = [[0 for x in range(5)] for y in range(5)]

for i in range(5):
    for j in range(5):
        z_beta_th1_abs_matrix[i][j] = np.abs(z_beta_th1_matrix[i][j])
```

We now use all these matrices to plot the $\zeta_{\rm Prop}$ that we are interested in.

In [18]:
```
x_list = ['1','2','3','4','5']

x_ticks = [0,1,2,3,4]

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

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

mpl.rcParams.update(mpl.rcParamsDefault)

font_size = 12 # Adjust as appropriate.


fig, ax = plt.subplots(2,2)

#fig.set_figwidth(12)
#fig.set_figheight(12)

ax[0][0].imshow(z_lum_th1_abs_matrix)
ax[0][1].imshow(z_e_pk_th1_abs_matrix)
ax[1][0].imshow(z_alpha_th1_abs_matrix)
ax[1][1].imshow(z_beta_th1_abs_matrix)

mapp00 = ax[0][0].imshow(z_lum_th1_abs_matrix, cmap = 'Blues')
mapp01 = ax[0][1].imshow(z_e_pk_th1_abs_matrix, cmap = 'Blues')
mapp10 = ax[1][0].imshow(z_alpha_th1_abs_matrix, cmap = 'Blues')
mapp11 = ax[1][1].imshow(z_beta_th1_abs_matrix, cmap = 'Blues')

ax[0][0].tick_params(top=True, labeltop=True, bottom=False, labelbottom=False)
ax[1][0].tick_params(top=True, labeltop=True, bottom=False, labelbottom=False)
ax[0][1].tick_params(top=True, labeltop=True, bottom=False, labelbottom=False)
ax[1][1].tick_params(top=True, labeltop=True, bottom=False, labelbottom=False)

ax[0][0].set_xticks(ticks = x_ticks, labels = x_list, fontsize = font_size)
ax[0][0].set_yticks(ticks = x_ticks, labels = y_list, fontsize = font_size)
ax[1][0].set_xticks(ticks = x_ticks, labels = x_list, fontsize = font_size)
ax[1][0].set_yticks(ticks = x_ticks, labels = y_list, fontsize = font_size)
ax[0][1].set_xticks(ticks = x_ticks, labels = x_list, fontsize = font_size)
ax[0][1].set_yticks(ticks = x_ticks, labels = y_list, fontsize = font_size)
ax[1][1].set_xticks(ticks = x_ticks, labels = x_list, fontsize = font_size)
ax[1][1].set_yticks(ticks = x_ticks, labels = y_list, fontsize = font_size)

#plt.ylabel('Temporal Levels (fps)', fontsize = 20)
#plt.xlabel('Spatial Refinement Levels', fontsize = 20)

ax[0][0].xaxis.set_label_position('top')

cb00 = plt.colorbar(mappable = mapp00, ax = ax[0][0])

cb01 = plt.colorbar(mappable = mapp01, ax = ax[0][1])

cb10 = plt.colorbar(mappable = mapp10, ax = ax[1][0])

cb11 = plt.colorbar(mappable = mapp11, ax = ax[1][1])

#plt.suptitle(r'EM properties at $\theta_{obs}=1$')

cb00.ax.tick_params(labelsize=font_size)
cb00.set_label(r'$\zeta_{L_{iso}}$', size = font_size)

cb01.ax.tick_params(labelsize=font_size)
cb01.set_label(r'$\zeta_{E_{pk}}$', size = font_size)

cb10.ax.tick_params(labelsize=font_size)
cb10.set_label(r'$\zeta_{\alpha}$', size = font_size)

cb11.ax.tick_params(labelsize=font_size)
cb11.set_label(r'$\zeta_{\beta}$', size = font_size)

ax[0][0].annotate('(a)', xy = (-2,-1), size = font_size, annotation_clip=False)
ax[0][1].annotate('(b)', xy = (-2,-1), size = font_size, annotation_clip=False)
ax[1][0].annotate('(c)', xy = (-2,-1), size = font_size, annotation_clip=False)
ax[1][1].annotate('(d)', xy = (-2,-1), size = font_size, annotation_clip=False)

for i in range(len(res_levs)):
    for j in range(len(fps)):
        if round(float(z_lum_th1_abs_matrix[j][i]),2)>40:
            text00 = ax[0][0].text(i, j, round(float(z_lum_th1_abs_matrix[j][i]),2),
                           ha="center", va="center", color="white", size=7)
        else:
            text00 = ax[0][0].text(i, j, round(float(z_lum_th1_abs_matrix[j][i]),2),
                           ha="center", va="center", color="k", size=7)
        if round(float(z_e_pk_th1_abs_matrix[j][i]),2)>0.6:
            text01 = ax[0][1].text(i, j, round(float(z_e_pk_th1_abs_matrix[j][i]),2),
                           ha="center", va="center", color="white", size=7)
        else:
            text01 = ax[0][1].text(i, j, round(float(z_e_pk_th1_abs_matrix[j][i]),2),
                           ha="center", va="center", color="k", size=7)

        if round(float(z_alpha_th1_abs_matrix[j][i]),2)>0.005:
            if round(float(z_alpha_th1_abs_matrix[j][i]),2)>0.6:
                text10 = ax[1][0].text(i, j, round(float(z_alpha_th1_abs_matrix[j][i]
                               ha="center", va="center", color="white", size=7)
            else:
                text10 = ax[1][0].text(i, j, round(float(z_alpha_th1_abs_matrix[j][i]
                               ha="center", va="center", color="k", size=7)
        else:
            text10 = ax[1][0].text(i, j, round(float(z_alpha_th1_abs_matrix[j][i]),3),
                           ha="center", va="center", color="k", size=7)


        if round(float(z_beta_th1_abs_matrix[j][i]),2)>0.4:
            text11 = ax[1][1].text(i, j, round(float(z_beta_th1_abs_matrix[j][i]),2),
                           ha="center", va="center", color="white", size=7)
        else:
            text11 = ax[1][1].text(i, j, round(float(z_beta_th1_abs_matrix[j][i]),2),
                           ha="center", va="center", color="k", size=7)


fig.suptitle('Spatial Refinement Levels', size = font_size)
fig.supylabel('Temporal Refinement Levels (fps)', size = font_size)

plt.tight_layout()
plt.savefig('zeta_th1_all_100s.pdf',dpi=600, bbox_inches='tight')
plt.show()
```