Resolution Levels / AMR differences

We use this notebook to generate plots for the analysis of HD properties as the AMR changes.

We first import necessary libraries

In [1]:

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import matplotlib as mpl
import astropy.constants as const
```

We load in the HD properties at different AMR levels given by MCRaT, both at the beginning and end of the simulation.

In [2]:

```python
plt.rcParams.update(plt.rcParamsDefault)

res_lev=[1,2,3,4,5]
r=[[8.907950e+09,8.847328e+09],[4.446371e+09,4.431216e+09],[2.225083e+09,2.213717e+09
   [1.112067e+09,1.107331e+09],[5.561521e+08,5.535472e+08]]
theta=[[9.817500e-03,9.817500e-03],[4.908750e-03,4.908750e-03],[2.454375e-03,2.454375e
   [1.227187e-03,1.227187e-03],[6.135937e-04,6.135937e-04]]
temp=[[3.605174e+06,1.463763e+06],[1.412884e+06,1.169092e+06],[1.628582e+06,9.976432e
      [1.358494e+06,6.729956e+05],[1.434641e+06,6.291365e+05]]
gamma=[[9.039360e+00,6.174512e+00],[7.841741e+00,6.449077e+00],[8.484358e+00,6.258624e
   [8.219275e+00,6.344830e+00],[8.404782e+00,6.295365e+00]]
density=[[1.256534e-08,4.343332e-09],[8.260724e-09,4.465632e-09],[9.366805e-09,3.1299
   [1.047633e-08,3.193103e-09],[1.081867e-08,2.735922e-09]]

dr=[]
dtheta=[]
dtemp=[]
dgamma=[]
ddens=[]

for i in range(len(r)):
    dr.append((r[i][0]+r[i][1])/2)

for i in range(len(theta)):
    dtheta.append((theta[i][0]+theta[i][1])/2)

for i in range(len(temp)):
    dtemp.append((temp[i][0]+temp[i][1])/2)

for i in range(len(gamma)):
    dgamma.append((gamma[i][0]+gamma[i][1])/2)

for i in range(len(density)):
    ddens.append((density[i][0]+density[i][1])/2)

data=[dr,dtheta,dtemp,dgamma,ddens]
names=['radius','theta','temperature','gamma','density']
```

In [3]:

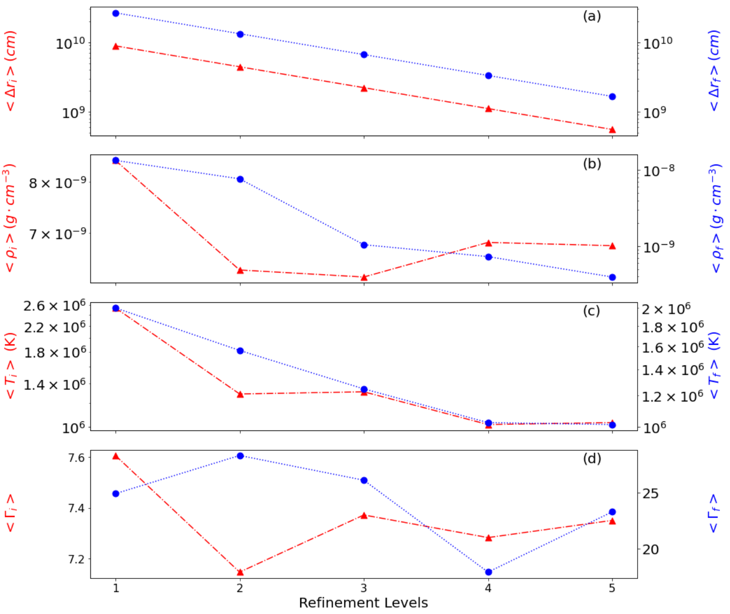
```python
rf=[[2.667530e+10,2.644858e+10],[1.333196e+10,1.322993e+10],[6.673096e+09,6.624855e+0
    [3.334768e+09,3.317025e+09],[1.666940e+09,1.658247e+09]]
thetaf=[[2.454375e-03,2.454375e-03],[1.227187e-03,1.227187e-03],[6.135937e-04,6.13593
        [3.067969e-04,3.067969e-04],[1.533984e-04,1.533984e-04]]
tempf=[[3.458137e+06,5.449138e+05],[2.685422e+06,4.372125e+05],[2.067201e+06,4.268187
       [1.579140e+06,4.693752e+05],[1.559288e+06,4.657468e+05]]
gammaf=[[4.348969e+01,6.328779e+00],[4.804116e+01,8.599458e+00],[4.615522e+01,6.06604
        [2.693778e+01,8.861176e+00],[4.225968e+01,4.307514e+00]]
densityf=[[2.686014e-08,5.022645e-13],[1.536223e-08,7.687351e-13],[2.088323e-09,1.980
       [1.455175e-09,6.371366e-13],[7.844489e-10,2.953128e-13]]

drf=[]
dthetaf=[]
dtempf=[]
dgammaf=[]
ddensf=[]

for i in range(len(rf)):
    drf.append((rf[i][0]+rf[i][1])/2)

for i in range(len(thetaf)):
    dthetaf.append((thetaf[i][0]+thetaf[i][1])/2)

for i in range(len(tempf)):
    dtempf.append((tempf[i][0]+tempf[i][1])/2)

for i in range(len(gammaf)):
    dgammaf.append((gammaf[i][0]+gammaf[i][1])/2)

for i in range(len(densityf)):
    ddensf.append((densityf[i][0]+densityf[i][1])/2)

dataf=[drf,dthetaf,dtempf,dgammaf,ddensf]
names=['radius','theta','temperature','gamma','density']
```

We now plot initial and final HD properties.

In [4]:

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


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


fig = plt.figure()
fig.set_figwidth(14)
fig.set_figheight(15)
gs = fig.add_gridspec(4, hspace=0.15)
axs = gs.subplots(sharex=True)
#fig.suptitle('HD properties at Different Refinement Levels',y=0.92)
axs[0].scatter(res_lev,data[0],color = 'r', marker = '^', s = 80)
#axs[0].scatter(res_lev,dataf[0],color = 'b')
axs[0].set_ylabel('$<\Delta r_{i}> (cm)$', fontsize=20, color = 'r')
axs[0].set_yscale('log')
axs[0].plot(res_lev,data[0], color = 'r', linestyle = '-.')
#axs[0].plot(res_lev,dataf[0], linestyle='dotted', color = 'b')
#axs[0].tick_params(axis ='y', labelcolor = 'r')

plt.tick_params(axis='both', which='major', labelsize=16)
#for tick in axs[0].yaxis.get_major_ticks():
#                tick.label.set_fontsize(16)

axs[1].scatter(res_lev,data[4],color = 'r', marker = '^', s = 80)
#axs[1].scatter(res_lev,dataf[4],color = 'b')
axs[1].plot(res_lev,data[4], linestyle='-.',color = 'r')
#axs[1].plot(res_lev,dataf[4], linestyle='dotted',color = 'b')
axs[1].set_ylabel(r'$<\rho_{i} >(g\cdot cm^{-3})$', fontsize = 20, color = 'r')
#axs[1].tick_params(axis ='y', labelcolor = 'r')
axs[1].set_yscale('log')

#for tick in axs[1].yaxis.get_major_ticks():
#                tick.label.set_fontsize(16)


axs[2].scatter(res_lev,data[2],color = 'r', marker = '^', s = 80)
axs[2].plot(res_lev,data[2], linestyle='-.',color = 'r')
#axs[2].scatter(res_lev,dataf[2],color = 'b')
#axs[2].plot(res_lev,dataf[2], linestyle='dotted',color = 'b')

axs[2].set_yscale('log')
#axs[2].yaxis.set_ticks(logax)
#axs[2].yaxis.set_major_formatter(formatter)
axs[2].set_ylabel('$<T_{i}>$ (K)', fontsize = 20, color = 'r')
#axs[2].tick_params(axis ='y', labelcolor = 'r')



#for tick in axs[2].yaxis.get_major_ticks():
#                tick.label.set_fontsize(16)

axs[3].scatter(res_lev,data[3],color = 'r', marker = '^', s = 80)
#axs[3].scatter(res_lev,dataf[3],color = 'b')
axs[3].set_ylabel('$< \Gamma_{i} >$', fontsize = 20, color = 'r')
axs[3].plot(res_lev,data[3], linestyle='-.',color = 'r')
#axs[3].plot(res_lev,dataf[3], linestyle='dotted',color = 'b')
plt.xlabel('Refinement Levels', fontsize = 20)
#axs[3].tick_params(axis ='y', labelcolor = 'r')

axs[0].annotate('(a)',xy=(0.9, 0.9), xycoords="axes fraction", fontsize = 20)

axs[1].annotate('(b)',xy=(0.9, 0.9), xycoords="axes fraction", fontsize = 20)

axs[2].annotate('(c)',xy=(0.9, 0.9), xycoords="axes fraction", fontsize = 20)

axs[3].annotate('(d)',xy=(0.9, 0.9), xycoords="axes fraction", fontsize = 20)

#axs[3].set_xticklabels(res_lev, fontsize=16)

for tick in axs[3].yaxis.get_major_ticks():
                tick.label.set_fontsize(16)

plt.xticks(range(1,6))

axs0 = axs[0].twinx()
axs0.scatter(res_lev,dataf[0],color = 'b', s = 80)
axs0.set_ylabel('$<\Delta r_{f}> (cm)$', fontsize=20, color = 'b')
axs0.set_yscale('log')
#axs[0].plot(res_lev,data[0], linestyle='dotted', color = 'k')
axs0.plot(res_lev,dataf[0], linestyle='dotted', color = 'b')
#axs0.tick_params(axis ='y', labelcolor = 'blue')

axs[0].get_shared_y_axes().join(axs[0], axs0)


axs1 = axs[1].twinx()
axs1.scatter(res_lev,dataf[4],color = 'b', s = 80)
#axs[1].plot(res_lev,data[4], linestyle='dotted',color = 'k')
axs1.plot(res_lev,dataf[4], linestyle='dotted',color = 'b')
axs1.set_ylabel(r'$<\rho_{f} >(g\cdot cm^{-3})$', fontsize = 20, color = 'b')
axs1.set_yscale('log')
#axs1.tick_params(axis ='y', labelcolor = 'blue')

axs2 = axs[2].twinx()
```

```
axs2.scatter(res_lev,dataf[2],color = 'b', s = 80)
axs2.plot(res_lev,dataf[2], linestyle='dotted',color = 'b')
#axs[2].yaxis.set_major_formatter(formatter)
#axs2.tick_params(axis ='y', labelcolor = 'blue')
axs2.set_yscale('log')
axs2.set_ylabel('$<T_{f}>$ (K)', fontsize = 20, color = 'b')


axs3 = axs[3].twinx()
axs3.scatter(res_lev,dataf[3],color = 'b', s = 80)
axs3.set_ylabel('$< \Gamma_{f} >$', fontsize = 20, color = 'b')
#axs[3].plot(res_lev,data[3], linestyle='dotted',color = 'k')
axs3.plot(res_lev,dataf[3], linestyle='dotted',color = 'b')
#axs3.tick_params(axis ='y', labelcolor = 'blue')
#plt.xlabel('Refinement Levels', fontsize = 20)

#plt.rc('font', size=15)
for i in range(len(axs[2].yaxis.get_minor_ticks())):
    if i%2==1:
        axs[2].yaxis.get_minor_ticks()[i].label1.set_visible(False)


# Hide x labels and tick labels for all but bottom plot.
for ax in axs:
    ax.label_outer()

fig.align_ylabels()

#plt.savefig('hdprops.pdf',dpi=600,bbox_inches='tight')

plt.show()
```



We now take a look at the size of cells compared to the angle opening.

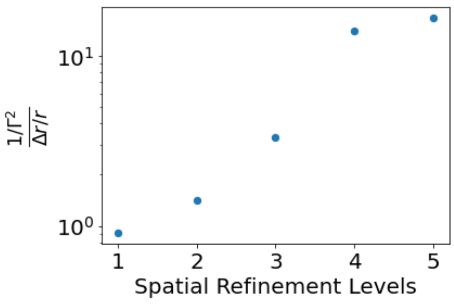
In [5]:

```
r0 = 1.5e13

theta_1_over_gamma2 = []

theta_r_over_r0 = []

for i in range(len(dataf)):
    theta_1_over_gamma2.append((1/dataf[3][i])**2)
    theta_r_over_r0.append(dataf[0][i]/r0)

theta_ratio = []

for i in range(5):
    theta_ratio.append((theta_1_over_gamma2[i]/theta_r_over_r0[i]))
```

We now plot this quantity.

In [6]:

```
#plt.scatter(res_lev, theta_1_over_gamma2, label = '1/Gamma^2')
#plt.scatter(res_lev,theta_r_over_r0, label = 'delta r / r')
#formatter = mpl.ticker.ScalarFormatter(useMathText=True)
plt.scatter(res_lev,theta_ratio, s = 40)
#plt.title('theta_gamma and theta_r')
plt.xticks(range(1,6), size =20)
plt.xlabel('Spatial Refinement Levels',size = 20)
plt.ylabel(r'$\frac{1/\Gamma^2}{\Delta r /r}$', size = 25)
plt.yscale('log')
#plt.legend()
plt.tight_layout()
#plt.savefig('theta_ratio.pdf',dpi=600,bbox_inches='tight')
plt.show()
```



We analyze the lightcrossing distance for different temporal/spatial resolutions.

In [7]:

```
lightcrossing_array = [[0 for x in range(5)] for y in range(5)]

deltari_array = [[0 for x in range(5)] for y in range(5)]

deltarf_array = [[0 for x in range(5)] for y in range(5)]

lightcrossing = 3e10 / np.array([0.3125,0.625,1.25,2.5,5])

for i in range(5):
    for j in range(5):
        lightcrossing_array[i][j] = lightcrossing[i]
        deltari_array[i][j] = data[0][j]
        deltarf_array[i][j] = dataf[0][j]

light_over_radius_i = [[0 for x in range(5)] for y in range(5)]

light_over_radius_f = [[0 for x in range(5)] for y in range(5)]

for i in range(5):
    for j in range(5):
        light_over_radius_i[i][j] = lightcrossing_array[i][j]/deltari_array[i][j]
        light_over_radius_f[i][j] = lightcrossing_array[i][j]/deltarf_array[i][j]
```

We now plot this quantity.

In [8]:

```
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)

from matplotlib import rc

font_size = 12 # Adjust as appropriate.

fig, ax = plt.subplots(1,2)

fig.set_figwidth(7)
fig.set_figheight(3)

ax[0].imshow(light_over_radius_i)

mapp0 = ax[0].imshow(light_over_radius_i, cmap = 'Blues')

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

ax[0].set_xticks(ticks = x_ticks, labels = x_list, fontsize = font_size)
ax[0].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].xaxis.set_label_position('top')

cb0 = plt.colorbar(mappable = mapp0, ax = ax[0])

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

cb0.ax.tick_params(labelsize=font_size)
cb0.set_label(r'$\frac{c\Delta t}{\Delta r}$', size = font_size)


for i in range(len(x_list)):
    for j in range(len(y_list)):
```

```
        if round(float(light_over_radius_i[j][i]),2)>100:
            text = ax[0].text(i, j, round(float(light_over_radius_i[j][i]),2),
                           ha="center", va="center", color="white", size=7)
        else:
            text = ax[0].text(i, j, round(float(light_over_radius_i[j][i]),2),
                           ha="center", va="center", color="k", size=7)


ax[1].imshow(light_over_radius_f)

mapp1 = ax[1].imshow(light_over_radius_f, cmap = 'Blues')

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

ax[1].set_xticks(ticks = x_ticks, labels = x_list, fontsize = font_size)
ax[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[1].xaxis.set_label_position('top')

cb1 = plt.colorbar(mappable = mapp1, ax = ax[1])

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

cb1.ax.tick_params(labelsize=font_size)
cb1.set_label(r'$\frac{c\Delta t}{\Delta r}$', size = font_size)


for i in range(len(x_list)):
    for j in range(len(y_list)):
        if round(float(light_over_radius_f[j][i]),2)>50:
            text = ax[1].text(i, j, round(float(light_over_radius_f[j][i]),2),
                           ha="center", va="center", color="white", size=7)
        else:
            text = ax[1].text(i, j, round(float(light_over_radius_f[j][i]),2),
                           ha="center", va="center", color="k", size=7)

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

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

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