ExampleMultiFrequency

January 10, 2025

This notebook shows examples of how to use AARTAstroModels with these astrophysical profiles can be used, and how the data is stored and accessed. Running this notebook takes less than ~2 minutes in a single CPU (Apple M2 Max) time!

Feel free to use this code (**with attribution to Ref.** [**1**]) for your research or to produce visualizations for your next presentation!

[1]:
```
# Importing necessary libraries
import matplotlib.pyplot as plt
import numpy as np
import subprocess

# Import custom functions and parameters
from aart_func import *
import params
from params import * # The file params.py contains all the relevant parameters␣
↪for the simulations
import fileloading
from astropy import units as u
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning:
scipy._lib.messagestream.MessageStream size changed, may indicate binary
incompatibility. Expected 56 from C header, got 64 from PyObject
```

1 Computation of the lensing bands

[2]:
```
%time !python3 lensingbands.py
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning:
scipy._lib.messagestream.MessageStream size changed, may indicate binary
incompatibility. Expected 56 from C header, got 64 from PyObject
Computing the lensing bands
Number of points in the n=0 grid  4000000
Number of points in the n=1 grid  4000000
Number of points in the n=2 grid  4000000
File  ./Results/LensingBands_a_0.9375_i_17.h5  created.
CPU times: user 144 ms, sys: 62 ms, total: 206 ms
Wall time: 28.3 s
```

1.1 Reading the sizes of the lensing bands calculation

[3]:
```
fnbands= path + "LensingBands_a_%s_i_%s.h5"%(spin_case,i_case)

print("Reading file: ",fnbands)

h5f = h5py.File(fnbands,'r')

lim0=int(h5f["lim0"][0])

h5f.close()
```

```
Reading file:  ./Results/LensingBands_a_0.9375_i_17.h5
```

2 Analytical Ray-tracing

[4]:
```
%time !python3 raytracing.py
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning:
scipy._lib.messagestream.MessageStream size changed, may indicate binary
incompatibility. Expected 56 from C header, got 64 from PyObject
Ray-tracing
Reading file:  ./Results/LensingBands_a_0.9375_i_17.h5
Analytical ray-tracing of the n=0 band points
Analytical ray-tracing of the n=1 band points
Analytical ray-tracing of the n=2 band points
File  ./Results/Rays_a_0.9375_i_17.h5  created.

A total of 12000000 photons were ray-traced
CPU times: user 183 ms, sys: 84.2 ms, total: 267 ms
Wall time: 39.7 s
```

3 Calculating the emission angle (θ_{B})

[5]:
```
%time !python3 magneticangle.py
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning:
scipy._lib.messagestream.MessageStream size changed, may indicate binary
incompatibility. Expected 56 from C header, got 64 from PyObject
Magnetic Angle
using default lband
Reading file:  ./Results/LensingBands_a_0.9375_i_17.h5
using default rtray
Reading file:  ./Results/Rays_a_0.9375_i_17.h5
File  ./Results/MagneticAngle_a_0.9375_i_17.h5  created.
CPU times: user 18.8 ms, sys: 17.8 ms, total: 36.6 ms
Wall time: 3.03 s
```

4 Setting the parameters of the astrophysical model

[6]:
```
brightparams = {
    "nu0": 90e9,  # Observation frequency
    "mass": (MMkg * u.kg).to(u.g).value,  # Black hole mass
    "scale_height": .5,  # 2 scale_height
    "theta_b": 50.0 * (np.pi / 180),  # impact parameter, if assumed fixed
    "beta": 1.0,  # legacy (not used)
    "r_ie": 10.0,  # legacy (not used)
    "rb_0": 5,  # radius at which power laws equal base values
    "n_th0": 1.9e4,  # density power law base value
    "t_e0": 7e10,  # temperature power law base value
    "b_0": 8.13, # magnetic field power law base value
    "p_dens": -0.7,  # density power law exponent
    "p_temp": -1.0,  # temperature power law exponent
    "p_mag": -1.5,  # magnetic power law exponent
    "nscale": 0.2  # Scale of inoisy if used
}

funckeys = {
                "emodelkey" : 0, # emodelkey Emission Model choice, 0 = thermal␣
 ↪ultrarelativistic, 1 = power law (1 is WIP)
        "bkey" : 2,      # Type of magnetic field profile , 0 = true function␣
 ↪from brodrick and loeb eq. 3, 1 = power law with lmfit values of 0, 2 =␣
 ↪power law from values set in brightparams
        "nnoisykey" : 0, # nnoisykey Inoisy density. 0 = no noise, 1 = noise
        "tnoisykey" : 0, # tnoisykey Inoisy temperature
        "bnoisykey" : 0, # bnoisykey Inoisy magnetic field
        "theta_bkey": 0 # Variable impact parameter, 0 for varied, 1 for fixed
}
```

[7]:
```
cmd= fileloading.createIntensityArgs(brightparams,funckeys=funckeys) # create␣
 ↪the neccesary command line argument
```

[8]:
```
%time subprocess.run([cmd], shell=True)
```

```
using default rtray
Reading file:  ./Results/Rays_a_0.9375_i_17.h5
using default magAng
Intensity
using default lband
Reading file:  ./Results/LensingBands_a_0.9375_i_17.h5
Reading file:  ./Results/Rays_a_0.9375_i_17.h5
File  ./Results/Intensity_a_0.9375_i_17_nu_9.00000e+10_mass_1.29248e+43_scaleh_0
.5_thetab_0.873_beta_1.00_rie_10.0_rb_5.0_nth0_1.9e+04_te0_7.0e+10_b0_8.130e+00_
pdens_-0.7_ptemp_-1.0_pmag_-1.5_nscale_0.2_emkey_0_bkey_2_nkey_0_tnkey_0_bnkey_0
_magkey_0.h5  created.
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning:
scipy._lib.messagestream.MessageStream size changed, may indicate binary
incompatibility. Expected 56 from C header, got 64 from PyObject

CPU times: user 2.43 ms, sys: 12 ms, total: 14.4 ms
Wall time: 5.42 s
```

[8]:
```
CompletedProcess(args=['python3 radialintensity.py --nu 90000000000.0 --mass
1.2924827500377644e+43 --scaleh 0.5 --thetab 0.8726646259971648 --beta 1.0 --rie
10.0 --rb0 5 --nth0 19000.0 --te0 70000000000.0 --b0 8.13 --pdens -0.7 --ptemp
-1.0 --pmag -1.5 --nscale 0.2 --emodelkey 0 --bkey 2 --nnoisykey 0 --tnoisykey 0
--bnoisykey 0 --thetabkey 0 --lband 0 --rtray 0 --magang 0'], returncode=0)
```

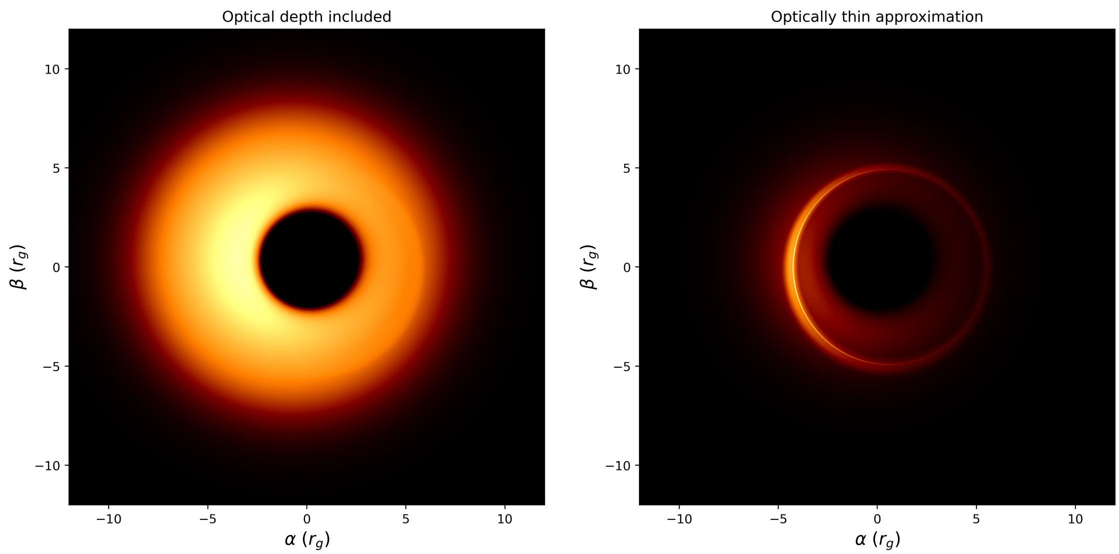
5 Load the images

[9]:
```python
fnrays = fileloading.intensityNameNoUnits(brightparams,funckeys) # find created␣
 ↪file na,e

print("Reading file: ",fnrays)

h5f = h5py.File(fnrays,'r')

# Optically thin assumption
I0=h5f['bghts0'][:] # This implies I0 is 1 pass
I1=h5f['bghts1'][:]
I2=h5f['bghts2'][:]
totalThinImage = I0 + I1 + I2

# Optical depth included RTE solution
Absorbtion_Image =h5f['bghts_full_absorbtion'][:]
h5f.close()
```

```
Reading file:  ./Results/Intensity_a_0.9375_i_17_nu_9.00000e+10_mass_1.29248e+43
_scaleh_0.5_thetab_0.873_beta_1.00_rie_10.0_rb_5.0_nth0_1.9e+04_te0_7.0e+10_b0_8
.130e+00_pdens_-0.7_ptemp_-1.0_pmag_-1.5_nscale_0.2_emkey_0_bkey_2_nkey_0_tnkey_
0_bnkey_0_magkey_0.h5
```

[10]:
```python
fig, (ax0,ax1) = plt.subplots(1,2,figsize=[15,7],dpi=400)

im0 = ax0.imshow(Absorbtion_Image,␣
 ↪origin="lower",cmap="afmhot",extent=[-lim0,lim0,-lim0,lim0],vmax=np.
 ↪nanmax(Absorbtion_Image)*1.2)

ax0.set_xlim(-12,12) # units of M
ax0.set_ylim(-12,12)



ax0.set_xlabel(r"$\alpha$"+" "+r"($r_g$)",fontsize=14)
```

```
ax0.set_ylabel(r"$\beta$"+" "+r"($r_g$)",fontsize=14)
ax0.set_title("Optical depth included")

im1 = ax1.imshow(totalThinImage,␣
 ↪origin="lower",cmap="afmhot",extent=[-lim0,lim0,-lim0,lim0],vmax=np.
 ↪nanmax(totalThinImage)*1.2)

ax1.set_xlim(-12,12) # units of M
ax1.set_ylim(-12,12)


ax1.set_xlabel(r"$\alpha$"+" "+r"($r_g$)",fontsize=14)
ax1.set_ylabel(r"$\beta$"+" "+r"($r_g$)",fontsize=14)
ax1.set_title("Optically thin approximation")

plt.savefig('TimeAveraged.png',dpi=400,bbox_inches='tight')
```



6 Using inoisy to add time-variability

[11]:
```
funckeys = {
                "emodelkey" : 0, # emodelkey Emission Model choice, 0 = thermal␣
 ↪ultrarelativistic, 1 = power law (1 is WIP)
        "bkey" : 2,       # Type of magnetic field profile , 0 = true function␣
 ↪from brodrick and loeb eq. 3, 1 = power law with lmfit values of 0, 2 =␣
 ↪power law from values set in brightparams
        "nnoisykey" : 1, # nnoisykey Inoisy density. 0 = no noise, 1 = noise
        "tnoisykey" : 1, # tnoisykey Inoisy temperature. 0 = no noise, 1 = noise
```

```
        "bnoisykey" : 1, # bnoisykey Inoisy magnetic field. 0 = no noise, 1 =␣
 ↪noise
        "theta_bkey": 0 # Variable impact parameter, 0 for varied, 1 for fixed
}

cmd= fileloading.createIntensityArgs(brightparams,funckeys=funckeys) # create␣
 ↪the neccesary command line argument
```

[12]:
```
%time subprocess.run([cmd], shell=True)
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning:
scipy._lib.messagestream.MessageStream size changed, may indicate binary
incompatibility. Expected 56 from C header, got 64 from PyObject

using default rtray
Reading file:  ./Results/Rays_a_0.9375_i_17.h5
using default magAng
Intensity
using default lband
Reading file:  ./Results/LensingBands_a_0.9375_i_17.h5
Reading file:  ./Results/Rays_a_0.9375_i_17.h5
File  ./Results/Intensity_a_0.9375_i_17_nu_9.00000e+10_mass_1.29248e+43_scaleh_0
.5_thetab_0.873_beta_1.00_rie_10.0_rb_5.0_nth0_1.9e+04_te0_7.0e+10_b0_8.130e+00_
pdens_-0.7_ptemp_-1.0_pmag_-1.5_nscale_0.2_emkey_0_bkey_2_nkey_1_tnkey_1_bnkey_1
_magkey_0.h5  created.
CPU times: user 2.22 ms, sys: 20.3 ms, total: 22.5 ms
Wall time: 5.7 s
```

[12]:
```
CompletedProcess(args=['python3 radialintensity.py --nu 90000000000.0 --mass
1.2924827500377644e+43 --scaleh 0.5 --thetab 0.8726646259971648 --beta 1.0 --rie
10.0 --rb0 5 --nth0 19000.0 --te0 70000000000.0 --b0 8.13 --pdens -0.7 --ptemp
-1.0 --pmag -1.5 --nscale 0.2 --emodelkey 0 --bkey 2 --nnoisykey 1 --tnoisykey 1
--bnoisykey 1 --thetabkey 0 --lband 0 --rtray 0 --magang 0'], returncode=0)
```

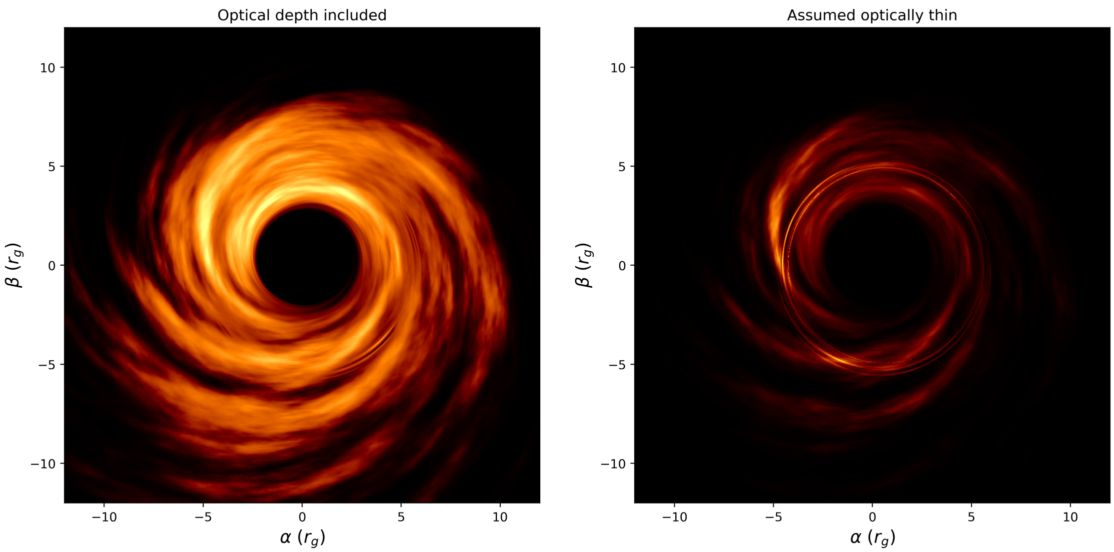
[13]:
```
fnrays = fileloading.intensityNameNoUnits(brightparams,funckeys) # find created␣
 ↪file na,e

print("Reading file: ",fnrays)

h5f = h5py.File(fnrays,'r')

# Optically thin assumption
I0=h5f['bghts0'][:] # This implies I0 is 1 pass
I1=h5f['bghts1'][:]
I2=h5f['bghts2'][:]
totalThinImage = I0 + I1 + I2

# Optical depth included RTE solution
```

```
Absorbtion_Image =h5f['bghts_full_absorbtion'][:]
h5f.close()
```

```
Reading file:  ./Results/Intensity_a_0.9375_i_17_nu_9.00000e+10_mass_1.29248e+43
_scaleh_0.5_thetab_0.873_beta_1.00_rie_10.0_rb_5.0_nth0_1.9e+04_te0_7.0e+10_b0_8
.130e+00_pdens_-0.7_ptemp_-1.0_pmag_-1.5_nscale_0.2_emkey_0_bkey_2_nkey_1_tnkey_
1_bnkey_1_magkey_0.h5
```

[14]:

```
fig, (ax0,ax1) = plt.subplots(1,2,figsize=[15,7],dpi=400)

im0 = ax0.imshow(Absorbtion_Image,␣
 ↪origin="lower",cmap="afmhot",extent=[-lim0,lim0,-lim0,lim0],vmax=np.
 ↪nanmax(Absorbtion_Image)*1.2)

ax0.set_xlim(-12,12) # units of M
ax0.set_ylim(-12,12)


ax0.set_xlabel(r"$\alpha$"+" "+r"($r_g$)",fontsize=14)
ax0.set_ylabel(r"$\beta$"+" "+r"($r_g$)",fontsize=14)
ax0.set_title("Optical depth included")

im1 = ax1.imshow(totalThinImage,␣
 ↪origin="lower",cmap="afmhot",extent=[-lim0,lim0,-lim0,lim0],vmax=np.
 ↪nanmax(totalThinImage)*1.2)

ax1.set_xlim(-12,12) # units of M
ax1.set_ylim(-12,12)


ax1.set_xlabel(r"$\alpha$"+" "+r"($r_g$)",fontsize=14)
ax1.set_ylabel(r"$\beta$"+" "+r"($r_g$)",fontsize=14)
ax1.set_title("Assumed optically thin")

plt.savefig('Snapshots.png',dpi=400,bbox_inches='tight')
```


Optical depth included
Assumed optically thin
10
5
0
−5
−10
β (r_g)
−10
−5
0
5
10
α (r_g)

6.1 License