Fitting HG1G2 using astropy

September 9, 2022

1 Fitting HG1G2 photometric system(s) with astropy

[3]:
```
import numpy as np
import scipy.optimize as sco
from matplotlib import pyplot as plt
%matplotlib inline
from astropy.table import Table
from astropy import units
from astropy.modeling import models, fitting
import sbpy.photometry as phot
```

Themis data as example.

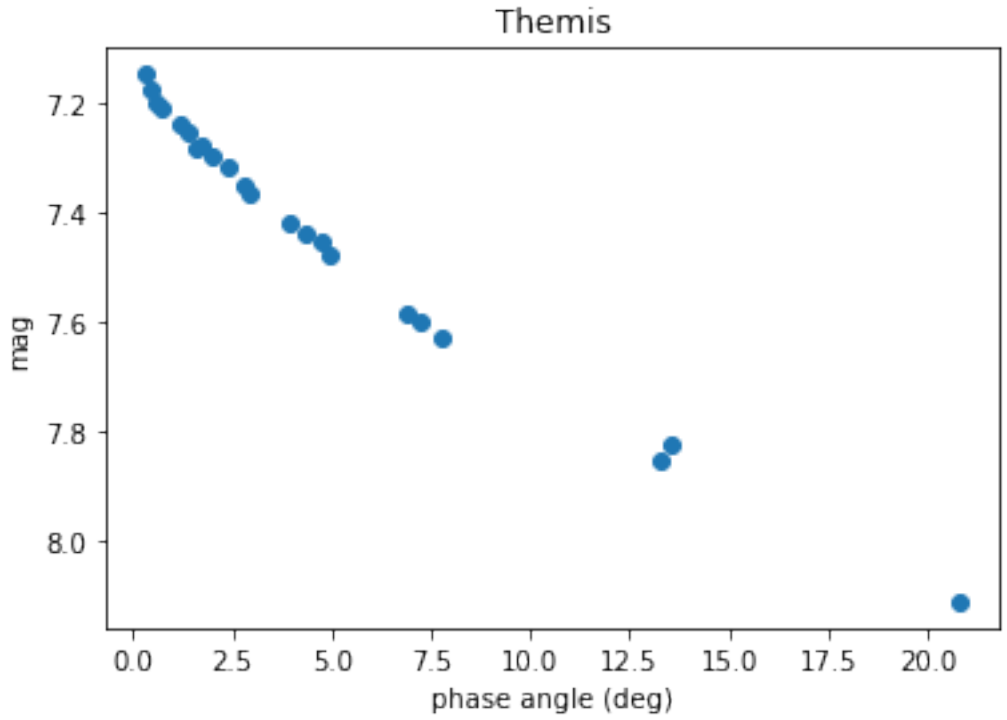
[104]:
```
t1 = [0.34, 0.44, 0.57, 0.74, 1.17, 1.39, 1.57, 1.71, 1.98,
      2.4, 2.8, 2.94, 3.95, 4.36, 4.75, 4.95, 6.89, 7.26,
      7.79, 13.28, 13.55, 20.79]
t2 = [7.146, 7.172, 7.197, 7.207, 7.239, 7.25, 7.279, 7.278,
      7.295, 7.317, 7.351, 7.366, 7.417, 7.439, 7.452, 7.478,
      7.586, 7.6, 7.626, 7.853, 7.825, 8.11]
themis = Table([t1,t2], names=('phase','V'))
themis['phase'].unit = 'deg'
themis['V'].unit = 'mag'
themis
```

[104]:
```
<Table length=22>
 phase     V
  deg     mag
float64 float64
------- -------
   0.34   7.146
   0.44   7.172
   0.57   7.197
   0.74   7.207
   1.17   7.239
   1.39    7.25
   1.57   7.279
   1.71   7.278
   1.98   7.295
```

```
  ...     ...
 3.95   7.417
 4.36   7.439
 4.75   7.452
 4.95   7.478
 6.89   7.586
 7.26     7.6
 7.79   7.626
13.28   7.853
13.55   7.825
20.79    8.11
```

How it looks.

[105]:
```
g,p1=plt.subplots(1,1)
p1.plot(themis['phase'],themis['V'],'o')
p1.set_title('Themis')
p1.set_xlabel('phase angle (deg)')
p1.set_ylabel('mag')
p1.invert_yaxis()
```



1.1 Non-constrained fitting, HG1G2

Define the chosen photometric function as custom model in astropy, let's use HG1G2.

[106]:
```
# Define model, assuming angles are in degrees
@models.custom_model
def hg1g2(x, H=0, G1=0.15, G2=0.6):
    return phot.HG1G2.evaluate(x*np.pi/180,H,G1,G2)
```

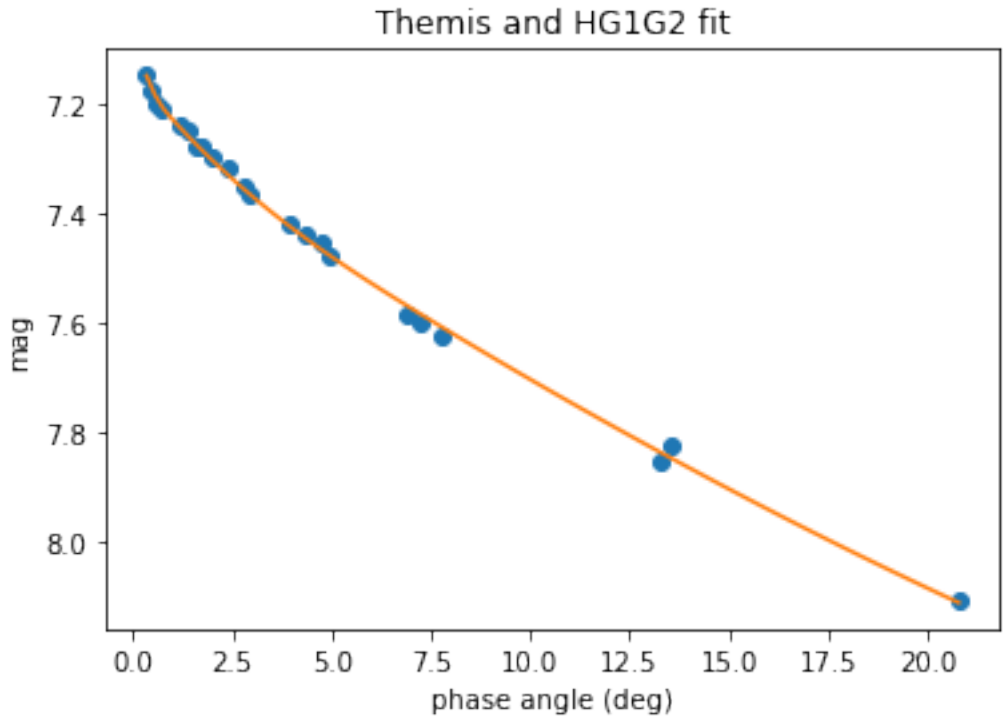
Fit without constraints, here LevMarLSQFitter seems to work well.

[107]:
```
model_init = hg1g2()
fit_fun = fitting.LevMarLSQFitter()
model = fit_fun(model_init, np.array(themis['phase']), np.array(themis['V']))
model
```

[107]:
```
<hg1g2(H=7.08834125, G1=0.62229788, G2=0.14491355)>
```

Show result.

[108]:
```
# Plot result
fx=np.linspace(themis['phase'][0],themis['phase'][-1],100)
fy=model(fx)
g,p1=plt.subplots(1,1)
p1.plot(themis['phase'],themis['V'],'o')
p1.plot(fx,fy)
p1.set_title('Themis and HG1G2 fit')
p1.set_xlabel('phase angle (deg)')
p1.set_ylabel('mag')
p1.invert_yaxis()
```



1.2 Non-constrained fitting, HG12

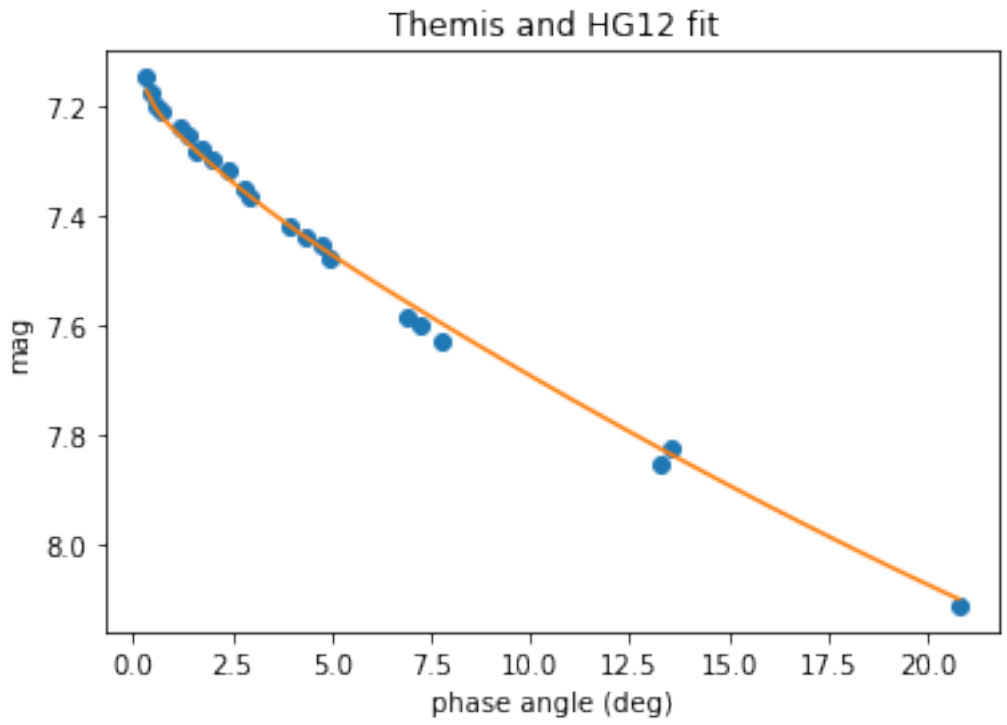
[109]:
```
# Define model, assuming angles are in degrees
@models.custom_model
def hg12(x, H=0, G12=0.15):
    return phot.HG12.evaluate(x*np.pi/180,H,G12)
```

[110]:
```
model_init = hg12()
fit_fun = fitting.LevMarLSQFitter()
model = fit_fun(model_init, np.array(themis['phase']), np.array(themis['V']))
model
```

[110]:
```
<hg12(H=7.12038847, G12=0.67615769)>
```

Show result.

[111]:
```
# Plot result
fx=np.linspace(themis['phase'][0],themis['phase'][-1],100)
fy=model(fx)
g,p1=plt.subplots(1,1)
p1.plot(themis['phase'],themis['V'],'o')
p1.plot(fx,fy)
p1.set_title('Themis and HG12 fit')
p1.set_xlabel('phase angle (deg)')
p1.set_ylabel('mag')
p1.invert_yaxis()
```



1.3 Non-constrained fitting, HG12*

[112]:
```
# Define model, assuming angles are in degrees
@models.custom_model
def hg12star(x, H=0, G12=0.5):
    return phot.HG12_Pen16.evaluate(x*np.pi/180,H,G12)
```

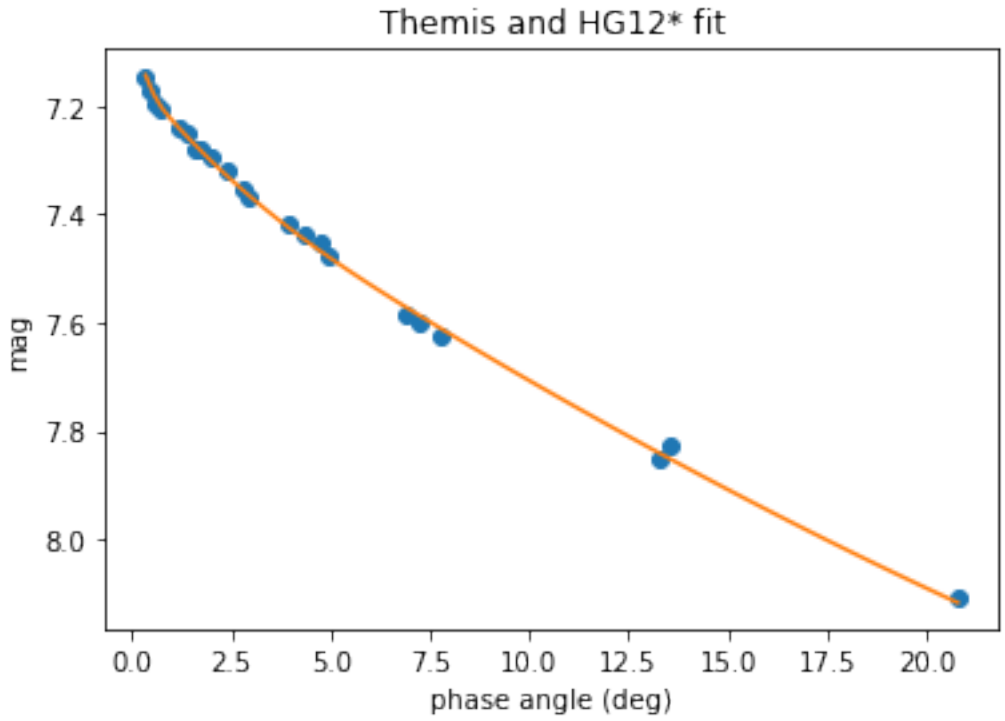
[113]:
```
model_init = hg12star()
fit_fun = fitting.LevMarLSQFitter()
model = fit_fun(model_init, np.array(themis['phase']), np.array(themis['V']))
model
```

[113]:
```
<hg12star(H=7.0824331, G12=0.73008948)>
```

Show result.

[114]:
```
# Plot result
fx=np.linspace(themis['phase'][0],themis['phase'][-1],100)
fy=model(fx)
g,p1=plt.subplots(1,1)
p1.plot(themis['phase'],themis['V'],'o')
p1.plot(fx,fy)
p1.set_title('Themis and HG12* fit')
p1.set_xlabel('phase angle (deg)')
```

```
p1.set_ylabel('mag')
p1.invert_yaxis()
```



1.4 Constrained fitting, HG1G2

Define the chosen photometric function as custom model in astropy, let's use HG1G2.

[188]:
```
# Define model, assuming angles are in degrees
@models.custom_model
def hg1g2(x, H=0, G1=0.15, G2=0.6):
    return phot.HG1G2.evaluate(x*np.pi/180,H,G1,G2)
```

Fit with constraints by setting inequalities 0<=G1,G2,1-G1-G2<=1 to parameters and using SLSQ-PLSQFitter.

Unfortunately, the use of 'ineqcons' is so poorly documented in astropy that I cannot understand in which format they should be given. Therefore, the example below does not work.

[190]:
```
model_init = hg1g2(ineqcons={'fun': lambda x: x[1]},
                             {'fun': lambda x: x[2]},
                             {'fun': lambda x: 1-x[1]-x[2]}))
fit_fun = fitting.SLSQPLSQFitter()
# Command below will create error(s)
#model = fit_fun(model_init, np.array(themis['phase']), np.array(themis['V']))
#model
```

Show result.

[ ]:
```
# Plot result
fx=np.linspace(themis['phase'][0],themis['phase'][-1],100)
fy=model(fx)
g,p1=plt.subplots(1,1)
p1.plot(themis['phase'],themis['V'],'o')
p1.plot(fx,fy)
p1.set_title('Themis and HG1G2 fit')
p1.set_xlabel('phase angle (deg)')
p1.set_ylabel('mag')
p1.invert_yaxis()
```

1.5 Constrained fitting, HG12

[183]:
```
# Define model, assuming angles are in degrees
@models.custom_model
def hg12(x, H=0, G12=0.15):
    return phot.HG12.evaluate(x*np.pi/180,H,G12)
```

Constraints as min/max for G12-parameter. To my understanding, the bounds (-0.0818919,0.909714) are valid for G12 in Muinonen et al. HG12-system.

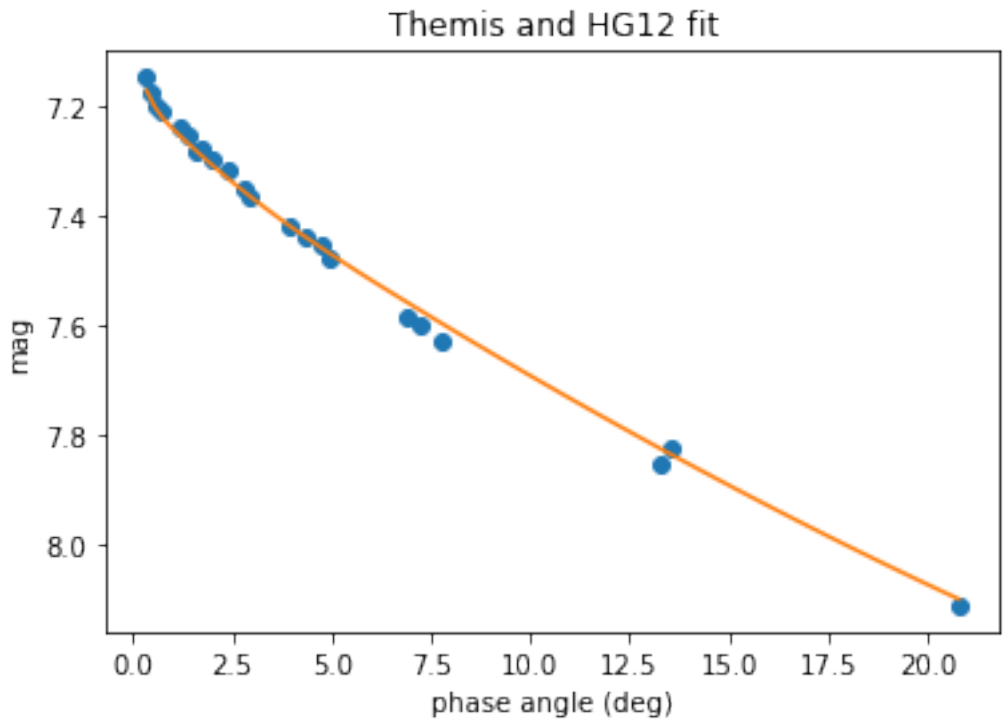
[186]:
```
model_init = hg12(bounds={'G12': (-0.0818919,0.909714)})
fit_fun = fitting.SLSQPLSQFitter()
model = fit_fun(model_init, np.array(themis['phase']), np.array(themis['V']))
model
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: 0.005658128573830853
            Iterations: 11
            Function evaluations: 37
            Gradient evaluations: 11
```

[186]:
```
<hg12(H=7.12039359, G12=0.6762039)>
```

Show result.

[187]:
```
# Plot result
fx=np.linspace(themis['phase'][0],themis['phase'][-1],100)
fy=model(fx)
g,p1=plt.subplots(1,1)
p1.plot(themis['phase'],themis['V'],'o')
p1.plot(fx,fy)
p1.set_title('Themis and HG12 fit')
p1.set_xlabel('phase angle (deg)')
p1.set_ylabel('mag')
p1.invert_yaxis()
```



1.6 Constrained fitting, HG12*

[180]:
```
# Define model, assuming angles are in degrees
@models.custom_model
def hg12star(x, H=0, G12=0.15):
    return phot.HG12_Pen16.evaluate(x*np.pi/180,H,G12)
```

Constraints as min/max for G12-parameter.

[181]:
```
model_init = hg12star(bounds={'G12': (0,1)})
fit_fun = fitting.SLSQPLSQFitter()
model = fit_fun(model_init, np.array(themis['phase']), np.array(themis['V']))
model
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: 0.0025925565041128866
            Iterations: 6
            Function evaluations: 20
            Gradient evaluations: 6
```

[181]:
```
<hg12star(H=7.08241653, G12=0.72993023)>
```

Show result.

[182]:
```
# Plot result
fx=np.linspace(themis['phase'][0],themis['phase'][-1],100)
fy=model(fx)
g,p1=plt.subplots(1,1)
p1.plot(themis['phase'],themis['V'],'o')
p1.plot(fx,fy)
p1.set_title('Themis and HG12* fit')
p1.set_xlabel('phase angle (deg)')
p1.set_ylabel('mag')
p1.invert_yaxis()
```