In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal, integrate
import astropy
from astropy.convolution import Gaussian1DKernel
from astropy.modeling.models import Lorentz1D
%matplotlib inline
```

In [42]:
```
sig_lorentz = 0.6 #sigma for Fake data, has nothing to do with my r
eal sigma for Eddington Bias***
# Sample schechter function
phistar = 0.000277
mstar = 10.58
alpha = 0.213
```

In [43]:
```
gauss_astropy = Gaussian1DKernel(stddev = sig_lorentz)
```

In [44]:
```
gauss_astropy.array #gauss_astropy is a kernel not an array, to rec
over array
```

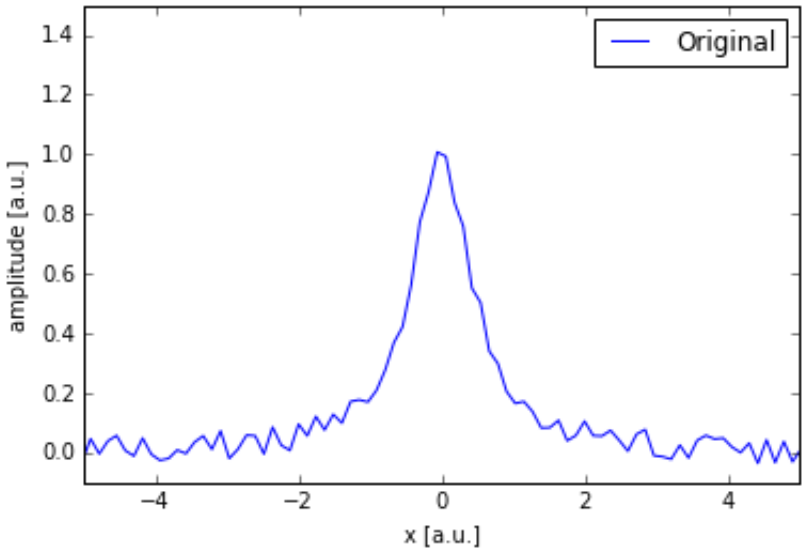
Out[44]:
```
array([ 0.00257046,  0.16579523,  0.6649038 ,  0.16579523,  0.0025
7046])
```

Fake data

Define a set of fake data to see how your gaussian function and Gaussian Kernel behave fake data taken from http://docs.astropy.org/en/v0.3/convolution/kernels.html (http://docs.astropy.org/en/v0.3/convolution/kernels.html)

In [45]:
```
lorentz = Lorentz1D(1, 0, 1)
x = np.linspace(-6, 6, 100)
data_1D = lorentz(x) + 0.1 * (np.random.rand(100) - 0.5)
```

In [46]:
```
plt.plot(x, data_1D, label='Original')
plt.xlabel('x [a.u.]')
plt.ylabel('amplitude [a.u.]')
plt.xlim(-5, 5)
plt.ylim(-0.1, 1.5)
plt.legend(prop={'size':12})
plt.show()
```



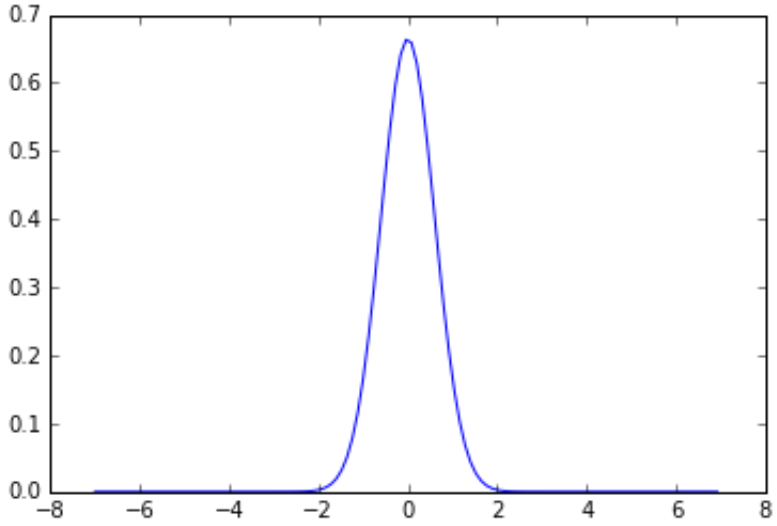
My convolve: Now I define the gaussian I was using for convolution The original example comes from http://stackoverflow.com/questions/24148902/python-convolution-with-a-gaussian (http://stackoverflow.com/questions/24148902/python-convolution-with-a-gaussian) But I am using a normalized version instead

In [47]:
```
gx = np.arange(-7,7, (max(x)-min(x))/len(x))
```

In [48]:
```
gauss_eq1 = (1/(sig_lorentz*np.sqrt(2*np.pi)))*(np.exp(-(gx**2)/(2*
sig_lorentz**2)))
```

In [49]:
```
plt.plot(gx, gauss_eq1)
```

Out[49]:
```
[<matplotlib.lines.Line2D at 0x114db8ad0>]
```



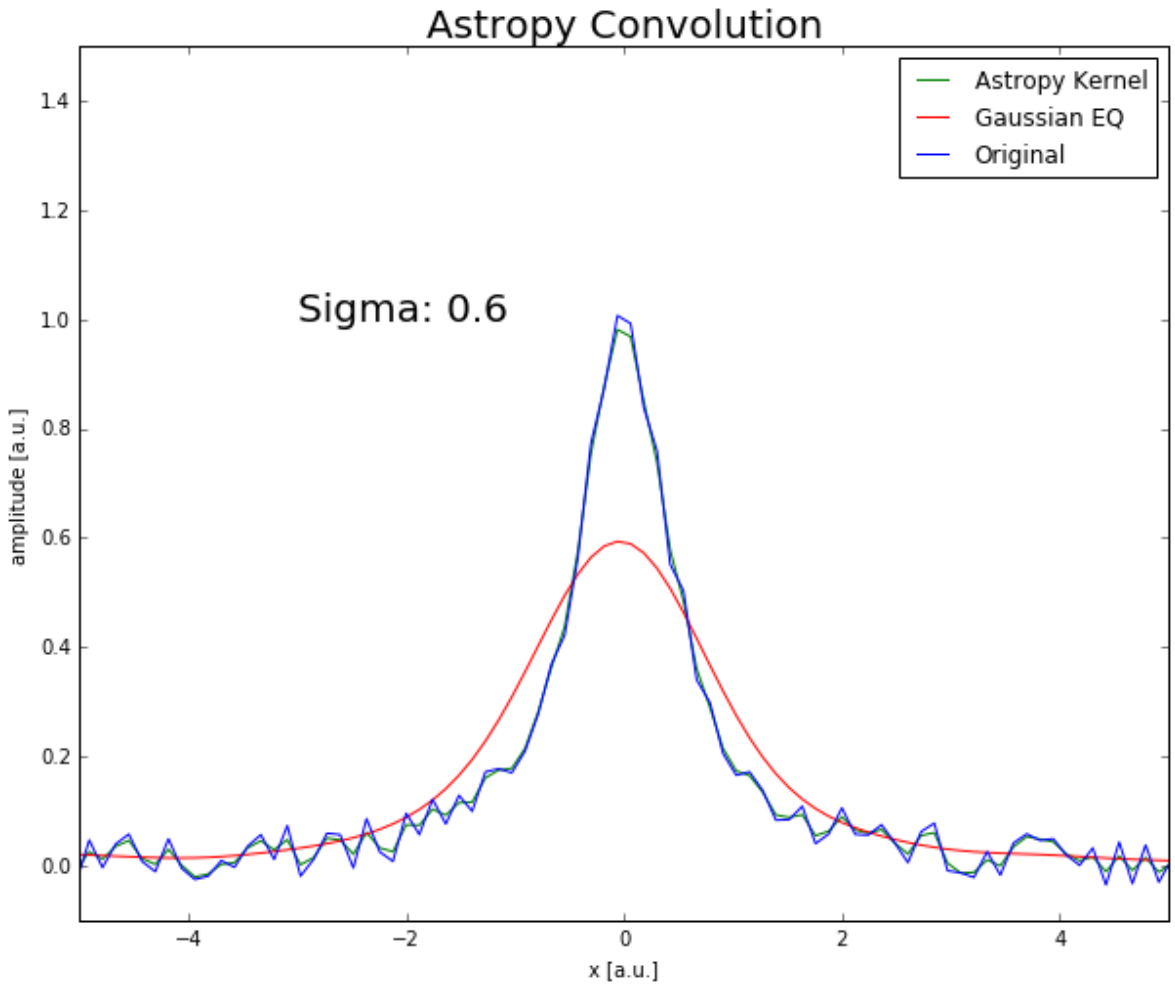
Now I can use both the Gaussian1D Kernel and my defined gauss_eq1 to convolve the fake Lorentz data There is always a curious error: ValueError: Convolution kernel must have odd dimensions And I must change the range of the gaussian to compensate. Another interesting feature is that even though the Gaussian is already normalized I have to use normalize_kernel to get the right results

In [50]:
```
lorentza = astropy.convolution.convolve(data_1D, gauss_astropy, bou
ndary='extend', normalize_kernel=True)
```

In [51]:
```
lorentzeq = astropy.convolution.convolve(data_1D, gauss_eq1, bounda
ry='extend', normalize_kernel=True)
```

In [52]:
```
plt.figure(figsize=(10,8), facecolor='white', edgecolor='w')
plt.xlabel('x [a.u.]')
plt.ylabel('amplitude [a.u.]')
plt.xlim(-5, 5)
plt.ylim(-0.1, 1.5)
plt.title('Astropy Convolution', fontsize=20)
plt.plot(x, lorentza, color='green', label='Astropy Kernel')
plt.plot(x, lorentzeq, color='red', label='Gaussian EQ')
plt.plot(x, data_1D, label='Original', color='blue')
plt.legend(prop={'size':12})
plt.text(-3,1, 'Sigma: '+str(sig_lorentz), fontsize=20)
```

Out[52]:
```
<matplotlib.text.Text at 0x11504f4d0>
```



What does this look in for a Schechter Function?

In [19]:
```
mass = np.arange(9,12,0.1)
sig = 0.12 #sigma for Schechter
sch = (phistar * np.log(10.) * 10**((mass-mstar)*(alpha+1)) * np.ex
p(-(10**((mass-mstar)))))
```

In [20]:
```
gauss_asc = Gaussian1DKernel(stddev = sig)
gx_sc = np.arange(-18,18, (max(mass)-min(mass))/len(mass))
```

In [21]:
```
gauss_eqsc = (1/(sig*np.sqrt(2*np.pi)))*(np.exp(-(gx_sc**2)/(2*sig*
*2)))
```

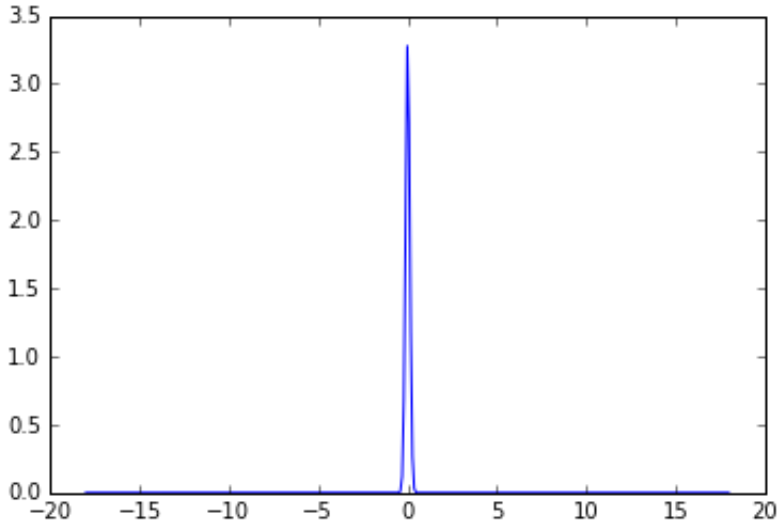
In [22]:
```
from scipy import integrate
```

In [23]:
```
integrate.trapz(gauss_eqsc, gx_sc)
```

Out[23]:
```
1.0000000000000331
```

In [24]:
```
plt.plot(gx_sc, gauss_eqsc)
```

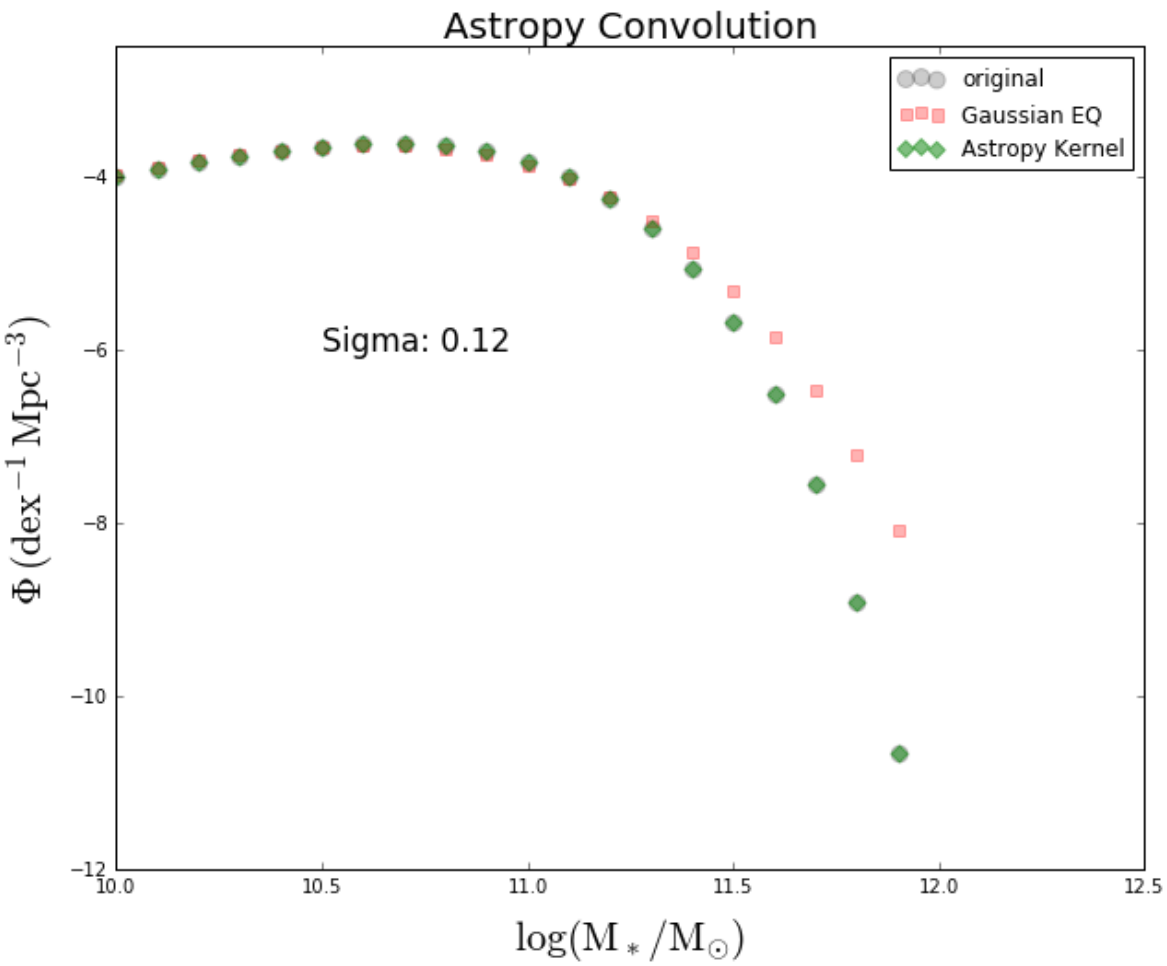
Out[24]:
```
[<matplotlib.lines.Line2D at 0x1131fcb50>]
```



In [25]:
```
sch_cona = astropy.convolution.convolve(sch, gauss_asc, boundary='e
xtend', normalize_kernel=True)
sch_coneq = astropy.convolution.convolve(sch, gauss_eqsc, boundary
='extend', normalize_kernel=True)
```

In [30]:
```
plt.figure(figsize=(10,8), facecolor='white', edgecolor='w')
plt.ylim(-12,-2.5)
plt.xlim(10, 12.5)
plt.xlabel(r'$\mathrm{log(M_*/M_{\odot})}$',fontsize=22,labelpad=1
0)
plt.ylabel(r'$\Phi\,\mathrm{(dex^{-1}\,Mpc^{-3})}$',fontsize=22)
plt.title('Astropy Convolution', fontsize=20)
plt.scatter(mass, np.log10(sch), marker='o', color='black', s = 80,
label = 'original', alpha=0.2)
plt.scatter(mass, np.log10(sch_coneq), marker='s', color='red', alp
ha=0.3, s = 40, label='Gaussian EQ')
plt.scatter(mass, np.log10(sch_cona), marker='D', color='green', al
pha=0.5, s = 40, label='Astropy Kernel')
plt.text(10.5, -6, 'Sigma: '+str(sig), fontsize = 17)
plt.legend()
```

```
Out[30]: <matplotlib.legend.Legend at 0x113eed550>
```



So for now I am using the Gaussian Equation to convolve the Schechter Function