Photon Statistics

Jayant Murthy
The Indian Institute of Astrophysics
www.iiap.res.in
jmurthy@yahoo.com

Binomial Distribution

- A binomial experiment is a statistical experiment that has the following properties:
  - The experiment consists of n repeated trials.
  - Each trial can result in just two possible outcomes. We call one of these outcomes a success and the other, a failure.
  - The probability of success, denoted by P, is the same on every trial.
  - The trials are independent; that is, the outcome on one trial does not affect the outcome on other trials.

http://stattrek.com/probability-distributions/binomial.aspx?Tutorial=AP

-

Properties

- μ is np where n is the number of trials and p is the probability.
- σ = sqrt(n * p * (1-p))

```
1. gdl
      500.000
GDL> print,0.5*1000,sqrt(0.5*0.5*1000)
      500.000      15.8114
GDL> h=fltarr(1000)
GDL> for i=0,999 do h(i) = total(randomu(seed,1000) lt 0.5) & print,mean(h),stde
v(h)
% Compiled module: MEAN.
% Compiled module: STDEV.
      499.920      15.5315
GDL> for i=0,999 do h(i) = total(randomu(seed,1000) lt 0.5) & print,mean(h),stde
v(h)
      500.377      15.4781
GDL> for i=0,999 do h(i) = total(randomu(seed,1000) lt 0.5) & print,mean(h),stde
v(h)
      500.140      15.8414
GDL> for i=0,999 do h(i) = total(randomu(seed,1000) lt 0.5) & print,mean(h),stde
v(h)
      500.217      15.5398
GDL> for i=0,999 do h(i) = total(randomu(seed,1000) lt 0.5) & print,mean(h),stde
v(h)
      500.049      15.5279
GDL> for i=0,999 do h(i) = total(randomu(seed,1000) lt 0.5) & print,mean(h),stde
v(h)
      499.790      16.0834
GDL>
```

Binomial Probability

- Probability of getting exactly k heads in n flips.
  - b(k, n, 0.5)
  - $p^k (1 - p)^{(n - k)}$ chances of getting exactly k heads
  - These can be anywhere so we have to multiply by n!/(k! * (n – k)!).

```
1. gdl
  GDL - GNU Data Language, Version 0.9.5

- For basic information type HELP,/INFO
- No startup file read (GDL_STARTUP/IDL_STARTUP env. var. not set).
- Please report bugs, feature or help requests and patches at:
  http://sourceforge.net/projects/gnudatalanguage/

GDL> .run binomial
% Compiled module: N_BANG.
% Compiled module: BINOMIAL.
GDL> .run factorial
% Compiled module: FACTORIAL.
GDL> for i=0,6 do print,binomial(i,6,0.5)
      1.00000
     0.984375
     0.890625
     0.656250
     0.343750
     0.109375
    0.0156250
GDL> print,binomial(3,6,0.5)-binomial(4,6,0.5)
     0.312500
GDL> print,factorial(6)/(factorial(3)*factorial(3))*.5^3*.5^3
      0.31250000
GDL>
```

Monte Carlo

1. vim

```
print,binomial(3, 10, 0.5) - binomial(4, 10, 0.5)
nheads = 0
for i=0, 9999 do begin
  heads = total(randomu(seed, 10) lt 0.5)
  if (heads eq 3) then nheads = nheads + 1
endfor
print,nheads/10000.
end
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"binomial_test.pro" 8L, 200C written
```

2. gdl

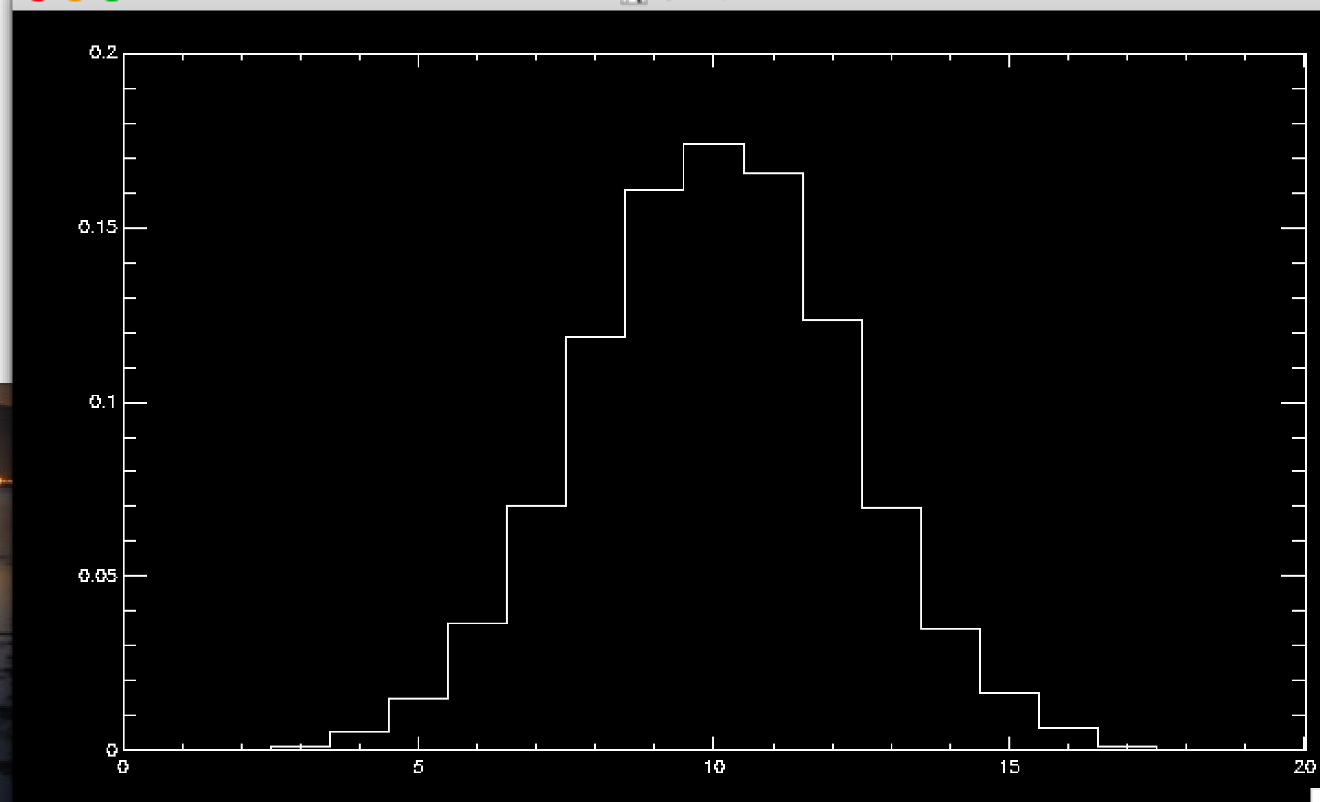
```
% Compiled module: $MAIN$.
% Compiled module: BINOMIAL.
      0.117188
      0.113300
GDL> .run binomial_test.pro
% Compiled module: $MAIN$.
      0.117188
      0.122700
GDL> .run binomial_test.pro
% Compiled module: $MAIN$.
      0.117188
      0.121200
GDL> .run binomial_test.pro
% Compiled module: $MAIN$.
      0.117188
      0.115200
GDL> .run binomial_test.pro
% Compiled module: $MAIN$.
      0.117188
      0.117300
GDL> .run binomial_test.pro
% Compiled module: $MAIN$.
      0.117188
      0.114500
GDL>
```

More Binomial

1. vim

```
nheads = fltarr(20)
for i=0, 9999 do begin
  heads = total(randomu(seed, 20) lt 0.5)
  nheads(heads) = nheads(heads) + 1
endfor
plot,nheads/10000.,psym=10
end
~
"binomial_test2.pro" 7L, 159C written
```

GDL 0



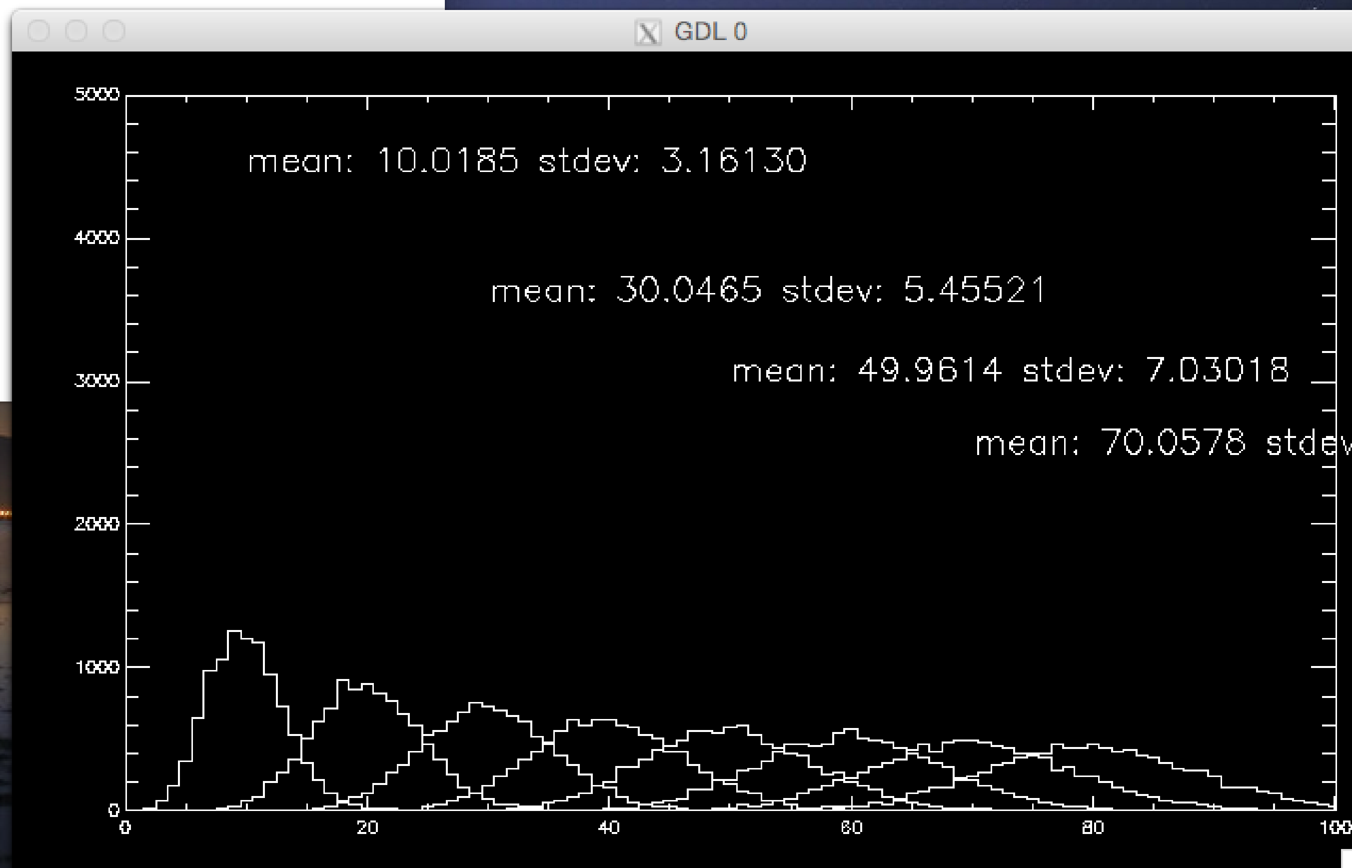
Poisson Distribution

- Similar to binomial distribution.
  - No concept of total number of flips.
- Binomial symmetrical around mean.
- Poissonian bounded by zero but extends upto infinity.
  - Finite probability of getting 1456 counts.
- $f(k, \lambda) = \lambda^k e^{-\lambda}/k!$
- $\mu = \lambda$; $\sigma^2 = \lambda$.
- For large λ, normal distribution is good approximation.

Poisson Distributions

1. vim

```
plot,[0,100],[0,5000],/nodata
for i=10,80,10 do begin
  p = randomu(seed, 10000,poisson = i)
  h=histogram(p, min=0, bin=1, max= 100)
  print,i,mean(p),stdev(p)
  oplot,h,psym=10
  if (((i + 10) mod 20) eq 0) then $
  xyouts,mean(p),max(h)+3400-i*20,$
      strcompress("mean: " + string(mean(p)) + " stdev: " + string(stdev(p))),$
      charsize=2
endfor
end
~
~
~
~
~
~
~
~
~
~
~
~
"poisson_distr.pro" 12L, 360C written
```



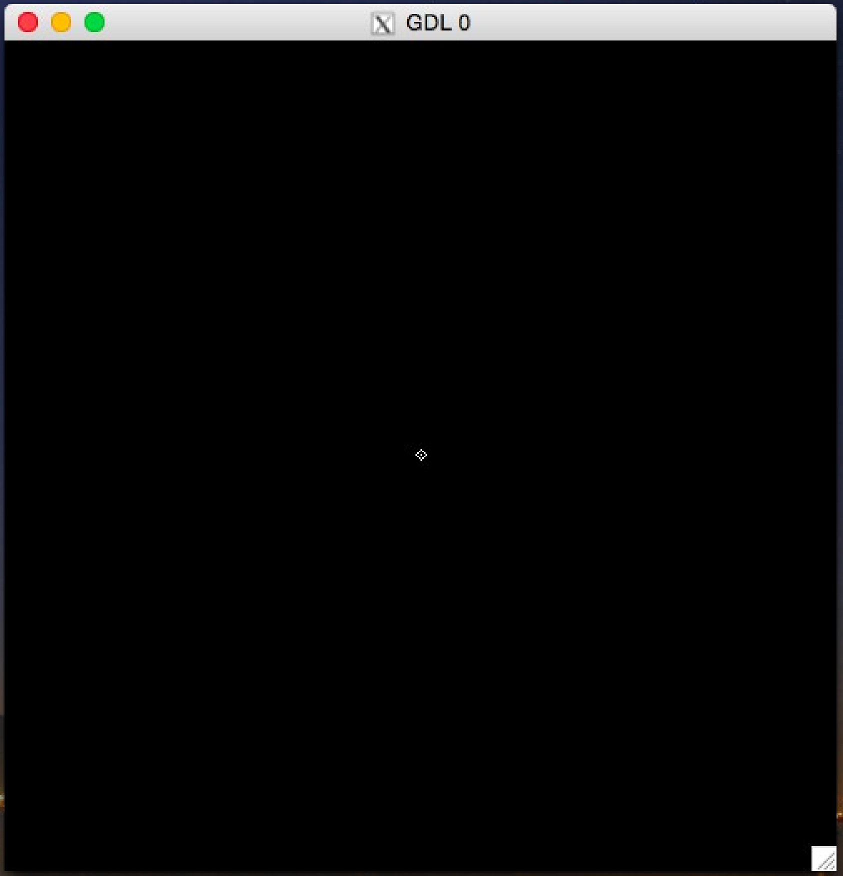
IDL Implementation

- rand = randomu(seed, 1000, poisson = mean)
- noise = poidev(mean, seed = seed)

Real Data

2. vim

```
mn = .1
p = randomu(seed, 1000, poisson = mn)
im = fltarr(500,500)
im(250, 250) = total(p)
window,xs=500,ys=500
plot,/nodata,[0,499],[0,499],xstyle=5,ystyle=5,$
        pos=[0,0,1,1],xrange=[0,499],yrange=[0,499]
tv,bytscl(im,0,10)
plots,[250,250],psym=4
end
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

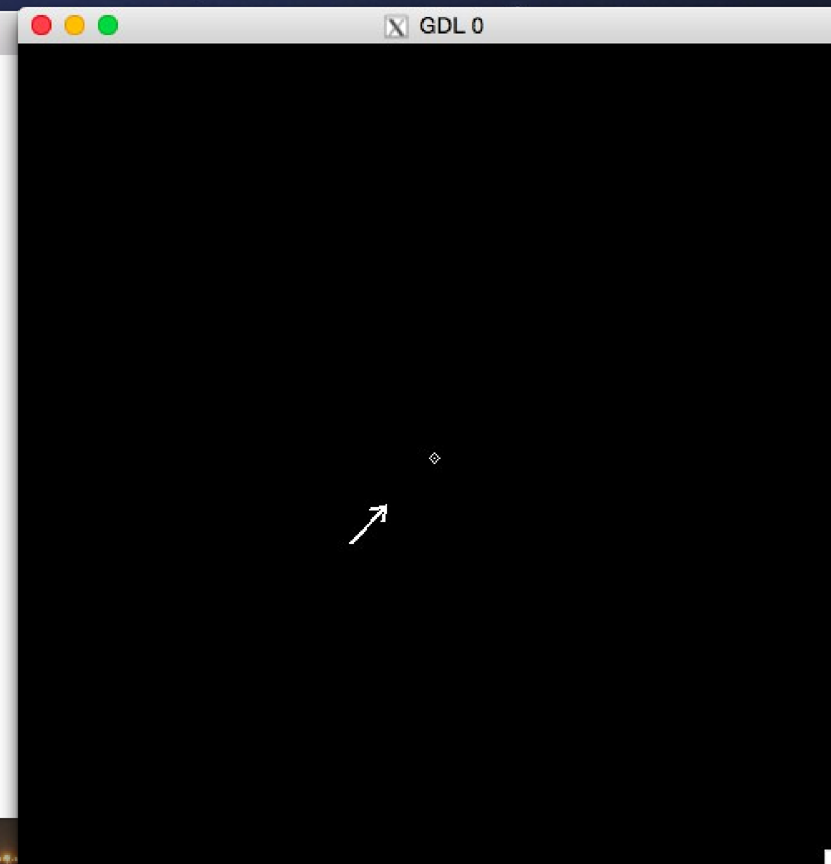


Can't tell the sky without arrows

2. vim

```
mn = .1
p = randomu(seed, 1000, poisson = mn)
im = fltarr(500,500)
im(250, 250) = total(p)
window,xs=500,ys=500
plot,/nodata,[0,499],[0,499],xstyle=5,ystyle=5,$
        pos=[0,0,1,1],xrange=[0,499],yrange=[0,499]
tv,bytscl(im,0,10)
plots,[250,250],psym=4
one_arrow,200,200,45,""
end
```

"single_star1.pro" 11L, 276C written



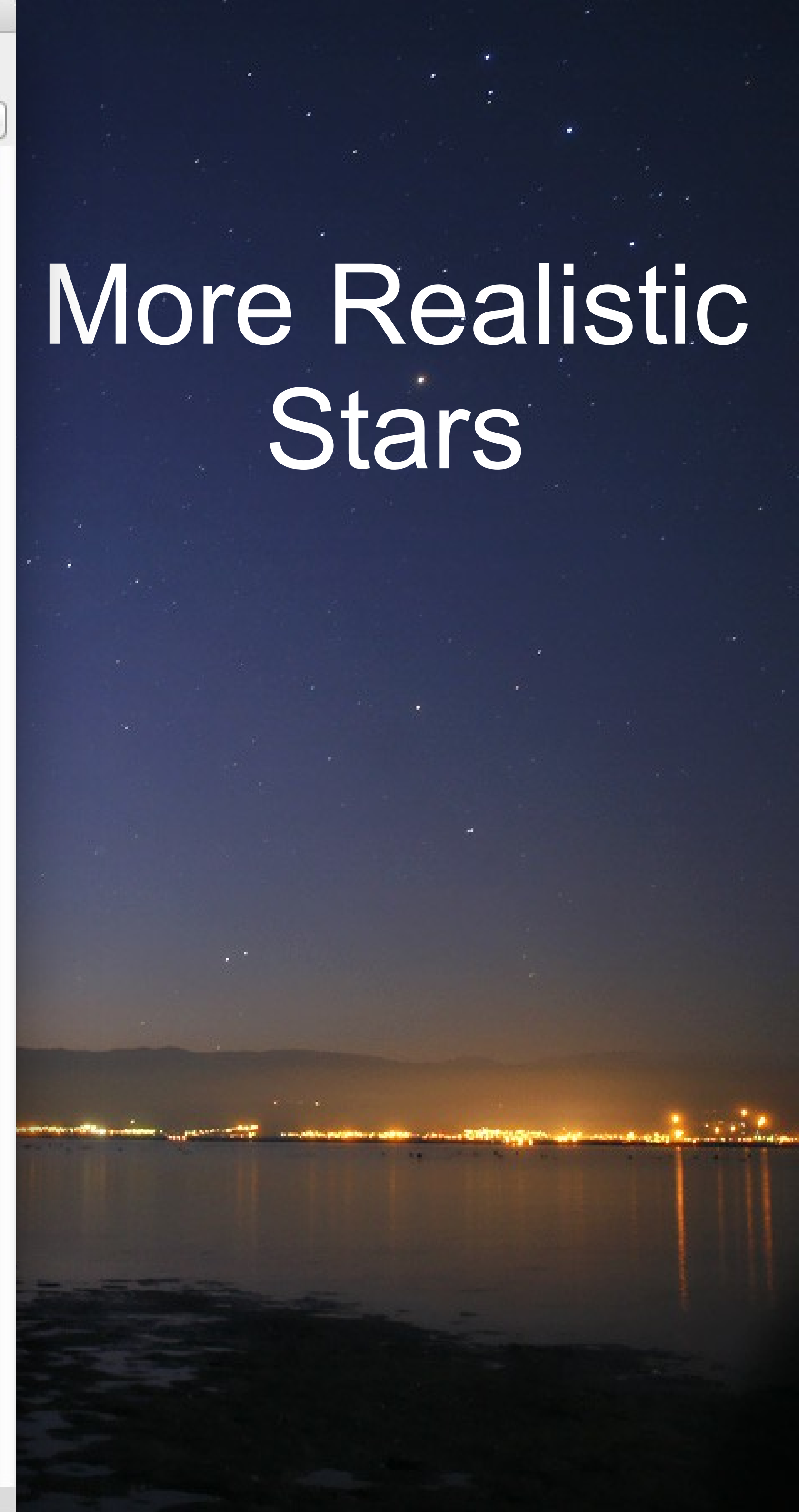
More Realistic Stars

single_star1_with_psf.pro (~/user/education/course/data_analysis/programs/)

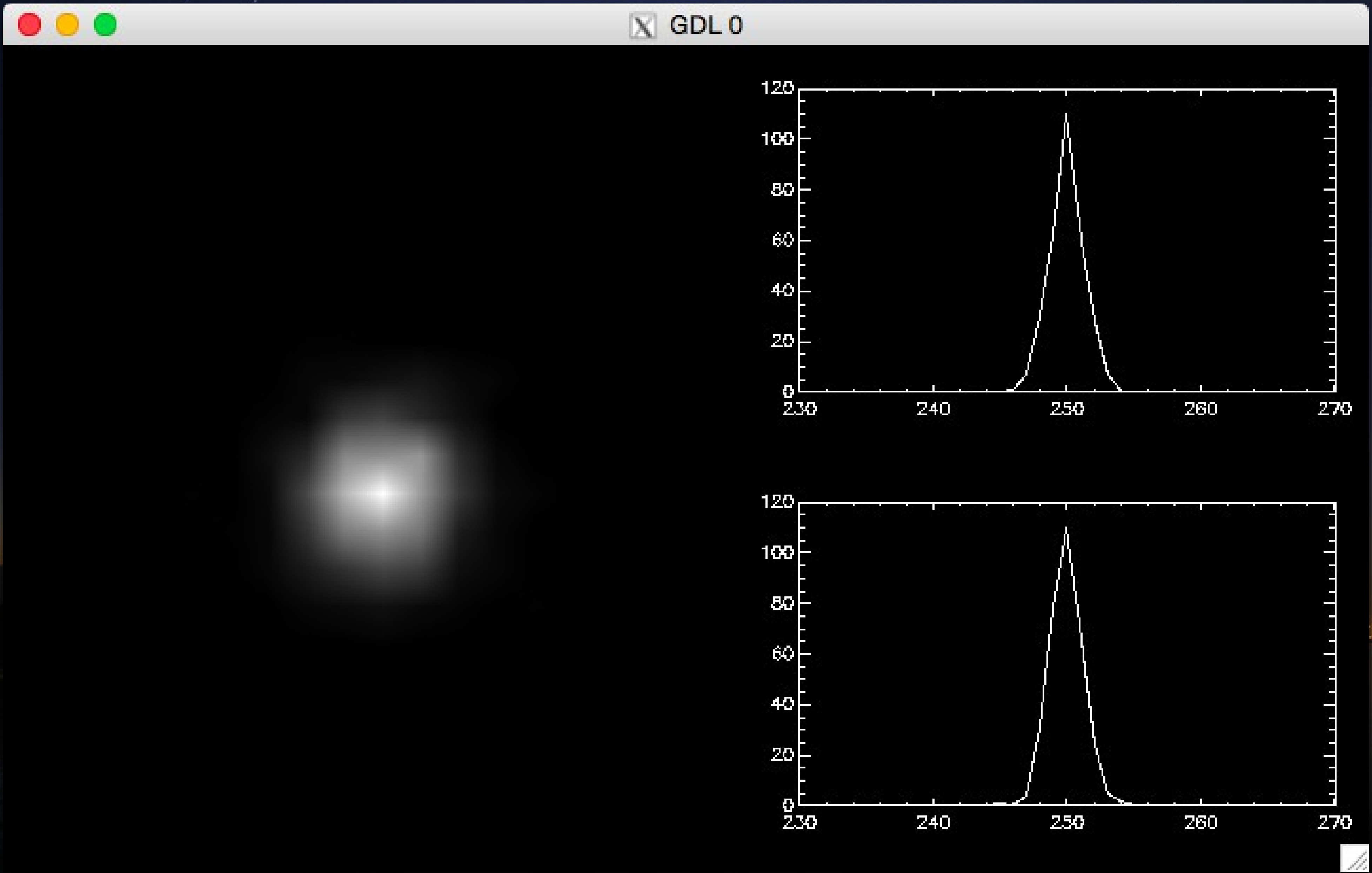
```
fwhm = 3  ;pixels
star_flux = 1 ;Counts per second
integration_time = 1000 ;seconds
im = fltarr(500,500) ;My data array
star_cent = [250, 250]

;Set up probability array for PSF
psf_prob = fltarr(11,11) ;Make sure we include entire PSF
psf      = psf_gaussian(npixel = 11, fwhm = fwhm)
psf_prob(0) = psf(0)
for i = 1, 120 do psf_prob(i) = psf_prob(i - 1) + psf(i)
psf_prob = psf_prob/total(psf)
psf_prob[120] = 1. ;Just for safety reasons.

;Go through frame by frame
for itime = 0, integration_time - 1 do begin
print,itime,max(im)
   p = randomu(seed, poisson = star_flux)
   if (p gt 0)then begin  ;We only have to worry if there is a photon
      for ip = 1, p do begin ;Minimum of 1 photon and continue to the end
        r = randomu(seed)
        psf_pos = min(where(psf_prob ge r))
        xpos = star_cent[0] - (5 - (psf_pos mod 11))
        ypos = star_cent[1] - (5 - (psf_pos / 11))
    if ((xpos ge 0) and (xpos lt 500) and $
            (ypos ge 0) and (ypos lt 500))then $
            im[xpos, ypos] = im[xpos, ypos] + 1
      endfor
   endif
tv,bytscl(rebin(im(240:260,240:260),420,420),0,max(im))
endfor
!p.multi=[3,2,2]
plot,im(250,*),xrange=[230,270],psym=10
!p.multi=[1,2,2]
plot,im(*,250),xrange=[230,270],psym=10
end
```



Results


GDL 0
120
100
80
60
40
20
0
230
240
250
260
270
120
100
80
60
40
20
0
230
240
250
260
270

Multiple Star Simulation

multiple_star_psf.pro (~/user/education/course/data_analysis/programs/)

```
pro multiple_star_psf,star_list_file,fwhm,integration_time,bkgd = bkgd

star_cent = fltarr(3, 7)
openr,1,star_list_file
readf,1,star_cent
close,1

;Set up PSF array
npsf = fwhm*2 + 1
psf = psf_gaussian(npixel = npsf, fwhm = fwhm)
psf_prob = fltarr(npsf, npsf)
psf_prob[0] = psf[0]
for i=1,npsf^2 - 1 do psf_prob[i] = psf_prob[i-1] + psf[i]
psf_prob = psf_prob/total(psf)
psf_prob[npsf^2-1] = 1.

im = fltarr(500,500) ;My data array
im_bkgd = fltarr(500,500)+bkgd
for itime = 0, integration_time - 1 do begin
print,itime,max(im)
    for istar = 0,6 do begin
        p = randomu(seed, poisson = star_cent(2,istar))
        if (p gt 0)then begin  ;We only have to worry if there is a photon
            for ip = 1, p do begin ;Minimum of 1 photon and continue to the end
                r = randomu(seed)
                psf_pos = min(where(psf_prob ge r))
                xpos = star_cent[0, istar] - (5 - (psf_pos mod npsf))
                ypos = star_cent[1, istar] - (5 - (psf_pos / npsf))
                if ((xpos ge 0) and (xpos lt 500) and $
                    (ypos ge 0) and (ypos lt 500))then $
                    im[xpos, ypos] = im[xpos, ypos] + 1
            endfor;ip
        endif
     endfor;istar
     im_noise = poidev(im_bkgd, seed = seed)
     im = im + im_noise
tv,bytscl(im,0,max(im))
endfor
end
```