
Subject: Re: Point Spread Function Simulation
Posted by [pgrigis](#) on Thu, 24 Sep 2009 15:35:57 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Sep 24, 11:23 am, Jacob Mendez <jake.men...@gmail.com> wrote:
> Hello, I'm trying to generate a simulated 2D point spread function to
> test a program that I am writing and I am wondering what might be a
> good way to do this.

It's pretty easy:

```
nx=512;dimensions of images  
ny=512  
sigma=10;width of psf
```

```
x=findgen(nx)-nx/2;x axis of image  
y=findgen(ny)-ny/2
```

```
xx=x#(x*0+1);create 2D coordinates  
yy=(y*0+1)#y
```

```
zz=exp(-0.5*(xx^2+yy^2)/sigma^2);compute PSF
```

Ciao,
Paolo

>
> My program will take 2D point spread functions and generate 1D line
> spread functions at incremental rotations of the 2D PSF.
>
> Thanks

Subject: Re: Point Spread Function Simulation
Posted by [Dan Larson](#) on Fri, 25 Sep 2009 13:48:51 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Sep 24, 11:35 am, Paolo <pgri...@gmail.com> wrote:
> On Sep 24, 11:23 am, Jacob Mendez <jake.men...@gmail.com> wrote:
>
>> Hello, I'm trying to generate a simulated 2D point spread function to
>> test a program that I am writing and I am wondering what might be a
>> good way to do this.
>
> It's pretty easy:
>
> nx=512;dimensions of images

```

> ny=512
> sigma=10;width of psf
>
> x=findgen(nx)-nx/2;x axis of image
> y=findgen(ny)-ny/2
>
> xx=x#(x*0+1);create 2D coordinates
> yy=(y*0+1)#y
>
> zz=exp(-0.5*(xx^2+yy^2)/sigma^2);compute PSF
>
> Ciao,
> Paolo
>
>
>
>
>> My program will take 2D point spread functions and generate 1D line
>> spread functions at incremental rotations of the 2D PSF.
>
>> Thanks- Hide quoted text -
>
> - Show quoted text -

```

Jacob,

It depends what you are trying to do. Are you trying to simulate a real point spread function, for example as you might find through a high numerical aperture objective? Do you want to look at low signal to noise diffraction limited spots? Or do you just need a 2D-Gaussian?

Here is some code I use to generate point spread functions to simulate data that might be collected in a fluorescence microscope. The diffraction limited spots have shot noise (Poisson distributed) and background noise (Gaussian distributed).

/Dan

```

FUNCTION psf, N, psf_width, bn_width, black_level, x_dim, y_dim, x0,
y0, Show=show

```

```

; psf generates a photon spot with background noise and poisson
noise. psf_width is the stddev of the
; gaussian spot in units of pixels. bn_width is the stddev of the
gaussian distributed background noise.
;

```

```
; Dan larson  
; 5/22/00
```

```
;*****define the index arrays*****
```

```
x_dim = long(x_dim)  
y_dim = long(y_dim)  
array=lindgen(x_dim, y_dim)  
xarr=array mod x_dim  
yarr=array/x_dim
```

```
;*****define some parameters for the point spread  
function*****
```

F=1.0/(sqrt(2.0)*psf_width) ;This factor shows up repeatedly in the error function calculation

```
a = 0.0  
b = 0.0      ; a,b,c,d are transformed integration limits for the  
error function  
c = 0.0  
d = 0.0
```

```
spot=dblarr(x_dim, y_dim)  
poisson_noise=dblarr(x_dim, y_dim)  
background_noise=bn_width*randomn(seed, x_dim, y_dim) ; gaussian  
distributed background noise with a black level  
a=F*(yarr - 0.5 - y0)  
b=F*(yarr + 0.5 - y0)  
c=F*(xarr - 0.5 - x0)  
d=F*(xarr + 0.5 - x0)  
spot =N*0.25*(errorf(a)-errorf(b))*(errorf(c)-errorf(d))  
index=where(spot GT 0, count)
```

```
;handle the case of no photons
```

```
if count gt 0 then begin
```

```
    location=array_indices(spot, index)  
    for i=0, count-1 do begin  
        poisson_noise(location[0, i], location[1, i])=randomn(seed,  
poisson=spot[location[0, i], location[1, i]])  
        ;if (photon GT 0.0) then poisson_noise(i-1, j-1)= randomn(seed,  
poisson=photon) else poisson_noise(i-1, j-1) = 0.0; poisson  
distributed shot noise
```

```
    endfor  
    endif else begin  
        poisson_noise=spot  
    endelse  
    ideal_spot= black_level+spot  
    poisson_spot= black_level+poisson_noise
```

```
background_spot=black_level+background_noise
```

```
;*****this is the important spot*****
```

```
noisy_spot= black_level+background_noise + poisson_noise
```

```
if keyword_set(show) then tvscl, noisy_spot
```

```
;print, total(spot), total(poisson_noise)
```

```
A={ideal_spot:ideal_spot, poisson_spot:poisson_spot,
```

```
noisy_spot:noisy_spot, background_spot:background_spot, $
```

```
spot_photons:total(spot), noisy_spot_photons:total(poisson_noise),
```

```
max_noisy_spot:max(noisy_spot)}
```

```
return, A
```

```
end
```
