Prof. Raghuveer Parthasarathy
University of Oregon; Fall 2024

Homework 5: Additional Notes

These are all comments on Problem 3.

COORDINATES AND UNITS – BE CAREFUL!

- Your simulated image function in Homework 4 allowed the emitter *xy* position to be a user input, in "physical" units of microns, not pixels. Keep this in mind.
- When making and evaluating simulated images, you'll have to decide whether to measure positions from the edge or the center of the arrays. (In other words, is the center of a 7 px x 7 px image located at x, y indices 0, 0 or at x, y indices 3, 3?) Either is fine, but be consistent. Obviously, if you simulate center-located images thinking that xc = 0 is the center, and perfectly localize them but call this xc = 3, you'll think there's an error of 3 pixels.

RMS ERROR

The RMS error is defined as

$$\sqrt{\langle((x_c - x_0)^2 + (y_c - y_0)^2)\rangle}$$

where <> means the average over all images, x_0 and y_0 are the true *x* and *y* positions, and x_c and y_c are the *x* and *y* positions that your algorithm calculates from an image. You can report the RMS error in physical units (microns or nm) or in pixels, but be clear on what units you're using.

TESTING YOUR IMAGE SIMULATION CODE.

Problem 3 requires using the simulated image function you wrote in Homework 4. If you were unable to do this, let me know, and I'll supply you with a function or a set of simulated images.
Also, **here's a test you can run** to see if your code is (probably) correct:

Define the following function (copy the text or download it from Canvas, `checkImage.py`):

```
import numpy as np

def checkImage(simImage):
    # function to do a few numerical checks of PSFs
    # I'm basing criteria on quick examination of simulated arrays
    # Basing criteria on quick examination of simulated arrays
    # with 0.5 micron light, NA = 0.9, xc= 40 nm, yc = 0
    # Raghu
    N = simImage.shape[0] # won't check that array is square
    center_px = int((N-1)/2)
    ratio_center_right1px = simImage[center_px,center_px] / \
        simImage[center_px,center_px+1]
    print(f'Ratio #1: {ratio_center_right1px:.2f}')
    if np.abs(ratio_center_right1px-1.06)<0.18:
        print('Test 1: good')
```

```
else:
    print('Test 1: failed!')
ratio_center_left1px = simImage[center_px,center_px] / \
    simImage[center_px,center_px-1]
print(f'Ratio #2: {ratio_center_left1px:.2f}')
if np.abs(ratio_center_left1px-1.81)<0.4:
    print('Test 2: good')
else:
    print('Test 2: failed!')
ratio_center_up1px = simImage[center_px,center_px] / \
    simImage[center_px-1,center_px]
print(f'Ratio #3: {ratio_center_up1px:.2f}')
if np.abs(ratio_center_up1px-1.36)<0.26:
    print('Test 3: good')
else:
    print('Test 3: failed!')
```

Make a NxN simulated image using your Homework 4 code with

N = 7,
camera scale = 0.1 microns/px,
wavelength of light 0.5 microns
NA = 0.9
total number of photons = 2000
x-offset, 0.04 **microns** (Note that this would be 0.4 **pixels**)
y-offset, 0 **microns**
Poisson background mean value = 2

Running the above function (checkImage(im)) you should pass all three checks!

SPEED

Get your code running well for poor grid resolution and low M (number of images) before attempting your "final" results. Python's `numpy` Poisson number generation is rather slow (far slower than MATLAB's, by the way); there may be better functions out there.

SUGGESTIONS

You may find it interesting to look at zero background, large images, and very large SNR, and see what the centroid scaling graph looks like. (You needn't hand this in.)