IPBMA. Exercise 4.

Analyzing the visibility of a nodule inside a noisy image

Analyzing the visibility of a nodule within a noisy image using Python functions. These functions will be called from the main program. The functions to be built, will be called ***createQuantumImage(), insertNoduleQImage(), detectorNoiseP_1_1(), detectorNoiseN_1_1(), plotMiddleLine(), plotMiddleLineCnt(), plotCellDistribution()***, and will include the following parameters:

createQuantumImage(N0, n, pointSize): We suppose that we have a source of x-ray photons that produce N0 photons/mm2. *createQuantumImage()* is a function that transforms those N0 photons/mm^2 into a quantum image with a specific number of points. In this quantum image, each point represents a number of photons N1.

i) N0 → Number of incident photons/mm^2. It means photons from an X-ray source.
ii) n → Number of points of the quantum image
iii) pointSize → Size of "each point" in the quantum image

Output→ Numpy array (2D), whose values will be the number of photons of each point of the quantum image when we suppose homogeneous radiation.

insertNoduleQImage(img, noduleSize, noduleContrast): Insert a nodule into a homogeneous quantum image.

i) img → Image where the nodule will be inserted.
ii) noduleSize → Diameter of the nodule in mm
iii) noduleContrast → Nodule contrast in %

Output→ Numpy array (2D), whose values will be the image's pixel values where the nodule was inserted.

detectorNoiseP_1_1(qImage): Simulate a detector following a Poisson distribution.

i) qImage → Incident quantum image

Output→ Numpy array (2D), whose values will be the image's pixel values of the image detected. We suppose detected values follow a Poisson distribution. You must use the function *numpy.random.poisson()* to simulate the Poisson distribution.

detectorNoiseN_1_1(qImage): Simulate a detector following a Normal distribution.

i) qImage → Incident quantum image

Output→ Numpy array (2D), whose values will be the image's pixel values of the image detected. We suppose detected values follow a Normal distribution. You must use the function *numpy.random.randn()* to simulate the Normal distribution.

plotMiddleLine(img, N): Plot the pixel values included in a line that crosses the center of an image from left to right.

i) img → Image from which the center line will be plotted
ii) N → Average number of photons of each point of the original quantum image (before the nodule was inserted)

.

plotCellDistribution(img, numberOfBins):

i) img → Image from which the pixel value distribution will be plotted.
ii) numberOfBins → Number of bins used to group the pixel values when building the distribution.

Note.- each team of students has to bring a zip file called *lastName_Name_P4.zip*, to the following address: *pablogtahoces@gmail.com*. The subject of the e-mail, should be: IPBMA_P4. Inside the zip should be included:

- A jupyter notebook, showing how the software works (see the example).
- An html file of the notebook.
- The .py files with the Python functions that were created.
- A pdf file explaining how the *rand()* and *poisson()* functions of the subpacket numpy.random works in your code.
- All the necessary files to verify the correct operation of the application.

Deadline → October 16, 10:00.