Exercises Week #1: Frequency domain processing

Solutions should be submitted via moodle before April 20th, 2021 8.00 a.m. (CET)

Exercise 1 – Convolution in 1D and 2D (Difficulty level: easy/medium)

Let's study the convolution of the following 1D and 2D arrays:

```
# 1D arrays
f = np.array([0, 1, 2, 3, 2, 1, 0])
g = np.array([1, 2, 3])
h = np.array([1, -2, 1])

# 2D arrays
F = np.array([[0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 1, 0, 0, 0],
              [0, 0, 1, 2, 1, 0, 0],
              [0, 1, 2, 2, 2, 1, 0],
              [0, 0, 1, 2, 1, 0, 0],
              [0, 0, 0, 1, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0]])
G = np.array([[1, 2, 3],
              [2, 3, 0],
              [3, 0, 0]])
H = np.array([[0, 1, 0],
              [1, -4, 1],
              [0, 1, 0]])
```

Task 1A. (Difficulty level: easy)
First, compute the following convolutions manually:

$$f \otimes g, \quad g \otimes f, \quad f \otimes h, \quad f \otimes f, \quad F \otimes G, \quad F \otimes H\,.$$

The results do not need to be submitted.

Task 1B. (Difficulty level: easy) Check your manually computed results by computing the convolution with NumPy's function `np.convolve` (for 1D arrays) and with SciPy's function `scipy.signal.convolve2d` (for 2D arrays).

Hint: The module `scipy.signal` is already imported at the beginning of the template file and abbreviated `sig`:

```
import scipy.signal as sig
help(sig.convolve2d)
```

Task 1C. (Difficulty level: medium) Compute $f \otimes g$ by applying the convolution theorem:

- Pad the filter `g` with zeros so as to match the size of the signal `f`.
- Move the center of the padded filter to the start of the array by using `ifftshift`.
- Transform `f` and `g` into the frequency domain.
- Multiply the (transformed) arrays element-wise.
- Transform the result back into the spatial domain.
- Compare the result with $f \otimes g$ obtained in 1B.

Task 1D. (Difficulty level: medium) Likewise compute $F \otimes G$ by applying the convolution theorem. Follow the procedure outlined in 1C.
Hint: `ifftshift` also works for multidimensional arrays, use `np.fft.fft2` and `np.fft.ifft2` to map 2D arrays into the frequency domain and back to the spatial domain.

Exercise 2 – Filtering a one-dimensional noisy signal (Difficulty level: medium)

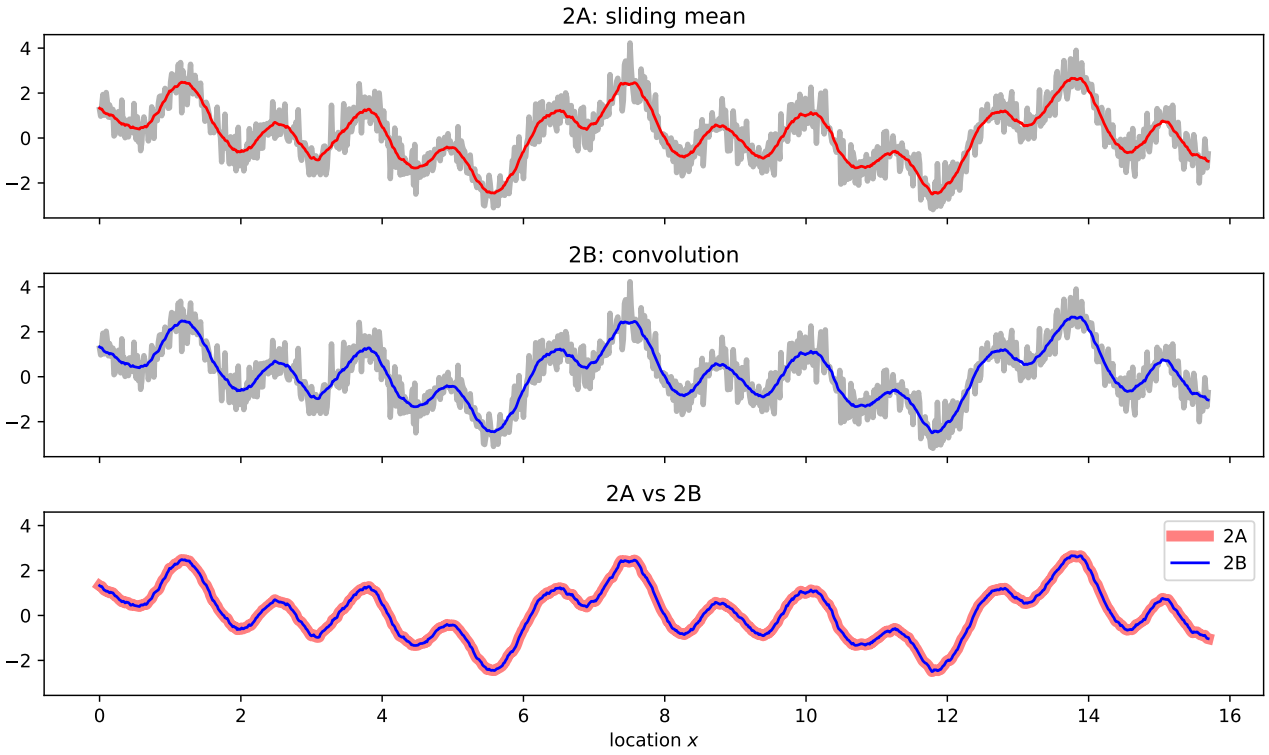
To eliminate noise and small variations between consecutive data points, we can use *smoothing techniques*. Sliding window methods process the data in small batches by sliding a window (of a reasonable size) along the data. Statistical operations such as *mean* are then applied to the data that lie inside the window. To generate a smoothed version of the data, the actual values are replaced by the computed values.

Task 2A. (Difficulty level: medium) In the initial part of the template, a noisy signal `y` at locations `x` is generated. Write a program that computes a moving average of `y`. The average value obtained for one data point should be the mean of the previous 10 data points, the data point itself and the following 10 data points. At the start and end of the signal, some of the preceding/subsequent data points will be missing such that the number of points contributing to the average decreases. The smoothed signal should have the same number of data points as the original signal (see top panel in figure below).

Task 2B. (Difficulty level: medium) Compute the moving average of `y` by using the fact that a sliding average can be expressed as the convolution of the signal with a rectangular pulse. Make

sure that the convolution does not shift the signal.

Task 2C. (Difficulty level: easy) Plot your results obtained in 2A and 2B. Plot the original signal `y` in transparent black and the smoothed signals in red (2A) and blue (2B) as in figure below. Compare both version of computing the average (see bottom panel in figure).



Exercise 3 – Filtering an image (Difficulty level: medium)

The algorithms used in Exercise 2 can also be applied to images. Load the image `'noisycells.tif'` by using the command `plt.imread` and convert it to a `double` array (see figure below).

Task 3A. (Difficulty level: medium) Filter the image by averaging all pixels within a patch of size 5×5 (2D version of Task 2A with kernel size 5 instead of 21). The expected result is shown below.

Task 3B. (Difficulty level: medium) Filter the image by using the convolution theorem (2D version of Task 2B). The expected result is shown below and should match the result from 3A.

Task 3C. (Difficulty level: medium) Generate a binary image by thresholding the image obtained in 3A (or 3B) at a minimum intensity of 50, i.e. all pixels with intensity greater than 50 are set to `True`, pixels with intensity equal to or smaller than 50 are set to `False`. The expected result is shown below.

Task 3D. (Difficulty level: medium) Apply filter `H` from Exercise 1 to the binary image generated in 3C. Again binarize the resulting image by setting all pixels with zero intensity to `True` and all other pixels to `False`. The expected result is shown below.

Task 3E. (Difficulty level: easy) Create a figure similar to: