Exercise 3

Deadline: 08.05.2013, 16.00

Regulations:

You may hand in the exercises in groups of *up to three persons*.

Hand in a printed version of your solutions at the beginning of the exercises. Your writeup should include the syntax-highlighted code and all relevant figures.

In addition, send a zipped directory containing your source code to `thorben.kroeger@iwr.uni-heidelberg.de`. Your subject line should be `[IA13][EX%02d] your names`. Please include the `pdf` file of your writeup, too.

Please cross-reference your code files in your writeup, such that it is clear which file has to be run.

Remember to comment your code!

1 Convolution in 1-D (4pt)

You are producing the following signal of length $n = 10$

$$g(x) = [0, 0, 0, 0, 1, 1, 1, 0, 0, 0],$$

and want to send it through a signal filter of which you have measured the impulse response to the signal $p(x)$ to be

$$p(x) = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$$
$$h(x) = [0, 0, 0, 0, 1, 2, 1, 0, 0, 0]$$

- What output signal will you measure? Compute by hand.
- Write down the circulant matrix H of the filter. How can the output be obtained using H?
- Describe the action of this filter.
- In the above example, the sum of the output is not the sum of the input signal. An amplification has taken place. Explain what condition the coefficients of the filter need to fulfill to keep the sum of the input and output signals the same. (*Hint*: DC term)

2 The Discrete Fourier transform in 1-D (4pt)

Give a closed-form solution, as a function of a temporal index l, of the inverse discrete fourier transform of

$$\mathring{g}(k) = \sqrt{n} \cdot \frac{i}{2}(\delta_{+2} - \delta_{-2}) \tag{1}$$

with length n. Here, δ_d is a discrete signal in k-space which is one at $k = d$ and zero elsewhere.

3 The discrete 2-D Fourier transform (8pt)

In this exercise we will create two special images, whose convolution $a * b$ is always exactly zero.

- Argue why we can achieve $a * b = 0$ by creating two images that have no spectral overlap.

- Implement the following experiment
  - Create an odd-sized matrix, named `a`, such that the matrix entries are zero in an inner rectangle R_{low} (which represents the low frequencies), have equal amplitude and random phase outside in a frame $R_{\text{medium}} \setminus R_{\text{low}}$ around this rectangle (which represents the medium frequencies) and are zero for the high frequencies. However, the phases should obey the constraints that the inverse Fourier transform is purely real; add an appropriate assert statement to check this.
    Hint: When developing your code, it is advisable to debug using a small matrix, for example 7×7. The `set_printoptions` (see implementation hints below) helps to display such a matrix in a readable way using `print array`.
  - Create a second matrix of the same size, named `b`, whose matrix elements are different from zero only in the inner rectangle R_{low}, where they should have equal amplitudes and random phases subject to the constraint that the inverse Fourier transform is purely real. Again, add an appropriate assert statement for this condition.
  - Transform both matrices into the spatial domain (name the resulting matrices `A` and `B`) by shifting the low frequencies to the corners and applying an inverse Fourier transform
  - Plot `A` and `B` as images using a size of 51×51. To get nice looking images, use very small widths of the rectangles R_{low} and R_{medium}.
    Hint: When plotting images with `matplotlib.pyploy.imshow`, always use `interpolation='nearest'`.
  - Convolve `A` and `B` and test that the result is zero (within numerical accuracy). For the convolution you must use periodic boundary conditions (use the `convolve2d` function with the appropriate keyword argument).
- Repeat the same for an even-sized matrix and consider that some care must be taken with the first row and column in order to ensure that the processes are real in the spatial domain.

Implementation hints.
In your implementation, it should be sufficient to use the functions imported below:

```
from numpy import complex, flipud, fliplr, set_printoptions, ones, \
    zeros, all, tile
from numpy.random import random
from numpy.fft import fftshift, fft2, ifft2, ifftshift
from scipy.signal import fftconvolve, convolve2d
set_printoptions(precision=2)

import matplotlib
matplotlib.use('Qt4Agg')
from matplotlib import pyplot as plot
```

In numpy, complex matrices are created and used like this:

```
a = random((3,3)) + 1j* random((3,3))
print a.real #the real part as an array
print a.imag #the imaginary part as an array
a[0,1].real = 42 #modify the real part of pixel (0,1)
a[0,1].imag = 41 #modify the imaginary part of pixel (0,1)
```

To test a condition for all entries of a matrix, use a construction like

```
assert all( abs(array) < 1E-12 )
```

It is easiest to work with images in k-space which have $k = (0, 0)$ in the center. To compute the inverse Fourier transform, remember that we first have to move $k = (0, 0)$ into the upper left corner. To do this, use the `ifftshift` function, the inverse Fourier transform can then be applied using `ifft2`.

4 Correlation theorem (4pt)

Recall that a convolution (operator $*$) of two 1-D signals a and g (of length n) can be written as

$$\begin{aligned} h_l &= a * g \\ &= \sum_{j=0}^{n-1} a_j g_{l-j} \\ &= A \cdot g, \end{aligned}$$

where A is the circulant matrix that represents signal a.

The *cross-correlation* of the same signals is defined as

$$\begin{aligned} h_l &= a \star g \\ &= \sum_{j=0}^{n-1} a_j^* g_{l+j}. \end{aligned}$$

What is the matrix notation for $h_l = a \star g$?

The cross-correlation theorem states that

$$\mathcal{F}[a \star g] = \mathcal{F}[a]^* \odot \mathcal{F}[g],$$

where we use $\{\}^*$ to denote the complex conjugate and $\odot$ to denote element-wise multiplication.

Prove this theorem using matrix notation in a similar way to how we proved the convolution theorem in the lecture.