exercise 4

Fourier transforms and Gaussian filters

solutions due

until **December 12, 2021** at **23:59** via **ecampus**

students handing in this solution set

last name	first name	student ID	enrolled with
Bach	Franziska	123456	B-IT / RWTH Aachen
Wolfe	Frank	654321	Uni Bonn

practical advice

The problem specifications you'll find below assume that you work with python / numpy / scipy. They also assume that you have imported

```
import imageio
import numpy as np
import numpy.fft as fft
import scipy.ndimage as img
import scipy.interpolate as ipl
import matplotlib.pyplot as plt
```

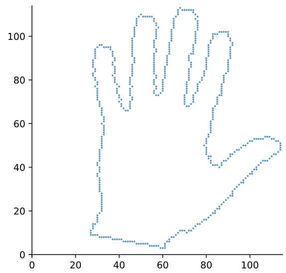
task 4.1

plotting boundaries of shape images

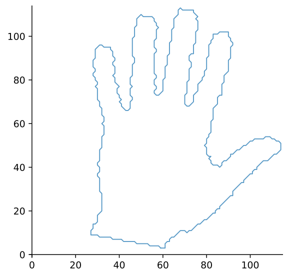
In the `Data` folder for this exercise, you will find the data file

    hand.csv

which contains a data matrix $\boldsymbol{X} \in \mathbb{R}^{2\times 537}$ whose columns $\boldsymbol{x}_k = [x_k, y_k]^\intercal$ represent 2D points. If you plot these data points using individual dots or a closed line, they will look like so



dot plot



line plot

To read these data into a *numpy* array `matX` representing matrix $\boldsymbol{X}$, you may proceed as follows

```
matX = np.loadtxt('hand.csv', delimiter=', ')
```

Now, center the data in $\boldsymbol{X}$, i.e. determine the the mean $\hat{\boldsymbol{x}}$ of the data points $\boldsymbol{x}_k$ and transform them as follows

$$\boldsymbol{x}_k \leftarrow \boldsymbol{x}_k - \hat{\boldsymbol{x}}$$

Create a dot plot and a line plot of the centered data (including coordinate axes) and enter them here

put your figure here

put your figure here

task 4.2

interpolating boundaries of shape images

In what follows, we will work with the centered data matrix $\hat{\boldsymbol{X}}$ which you computed in the previous task and we will assume that is available in a 2D *numpy* array `matXh`.

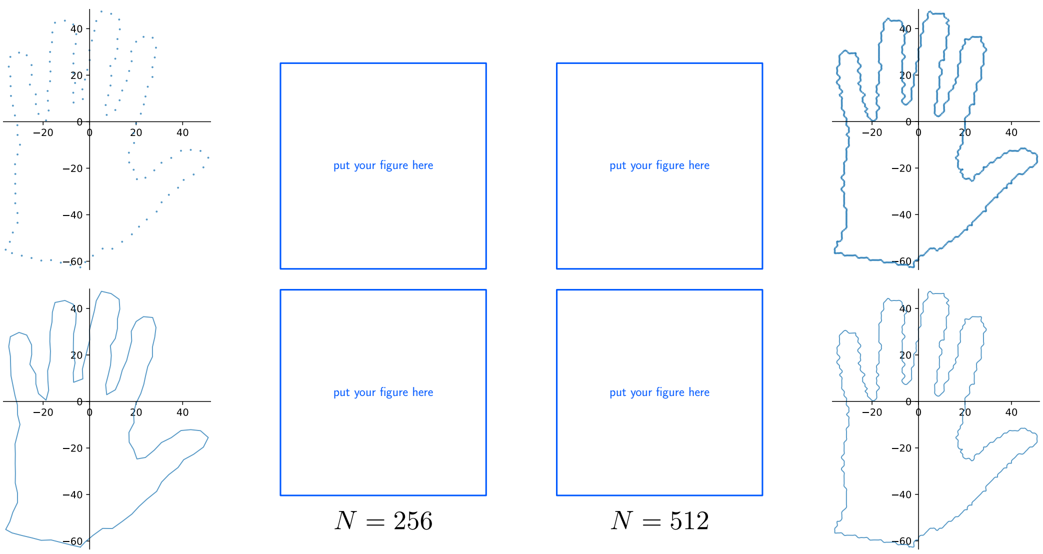
Recall that this matrix contains $n = 537$ column vectors $\boldsymbol{x}_k = [x_k, y_k]^\intercal$. Given these n data points, we will now interpolate N. Without further ado, this can be accomplished as follows

```
matXh = np.hstack((matXh, matXh[:,0].reshape(2,1)))

stps  = np.linspace(0, 2*np.pi, matXh.shape[1])
iplx  = ipl.interp1d(stps, matXh[0], kind='cubic')
iply  = ipl.interp1d(stps, matXh[1], kind='cubic')

matXi = np.vstack((iplx(np.linspace(0, 2*np.pi, N, endpoint=False)),
                   iply(np.linspace(0, 2*np.pi, N, endpoint=False))))
```

The following figure shows dot- and line plots of the results for the cases where $N = \{128, 1024\}$. Create such plots for $N = \{256, 512\}$ and enter them in the figure.



$N = 128$ $N = 256$ $N = 512$ $N = 1024$

task 4.3

Fourier transforms of periodic complex valued functions

Note that we can alternatively think of the 2D data points $\boldsymbol{x}_k = [x_k, y_k]^\intercal$ from the previous tasks in terms of complex numbers

$$z_k = x_k + i\, y_k$$

Also note that we can therefore think of our use of *ipl.interp1d* in the previous task as having created a continuous function $f : [0, 2\pi) \to \mathbb{C}$ which we then sampled at points $j \cdot \frac{2\pi}{N}$ where $j = 0, \ldots, N-1$ to obtain a vector or 1D array

$$\boldsymbol{z} = \big[z_0, z_1, \ldots z_{N-1}\big]$$

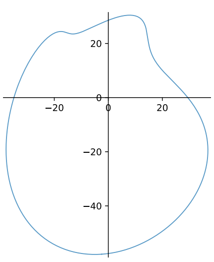
where $z_j = f\big(j \cdot \frac{2\pi}{N}\big)$. Finally, note that we may think of array $\boldsymbol{z}$ as containing the values of one period of the discrete periodic function $f[j] = f\big(j \cdot \frac{2\pi}{N}\big)$ where $j = J \mod N$ and $J \in \mathbb{Z}$.

To compute array $\boldsymbol{z}$ in practice, i.e. not just conceptually, you may reuse your results from the previous task. Letting $N = 256$, recompute array `arrXi` and then execute

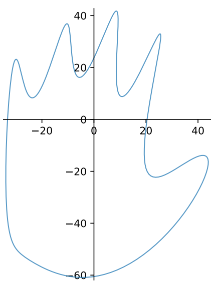
```
arrZ = matXi[0] + 1j * matXi[1]
```

Now compute the Fourier transform of array $\boldsymbol{z}$ and low pass filter it with cut-off frequencies $\omega_l \in \{2, 4\}$. (Remember our discussion of *fft.fftfreq* in lecture 10 and of *fft.fftshift* and *fft.ifftshift* in lecture 11.) Compute the inverse Fourier transform of the filtered signal and create line plots of your results and enter them below.

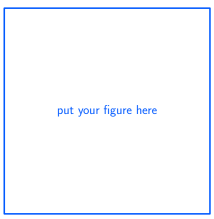
To see how these should look like, here are results for $\omega_l \in \{0.5, 1.0\}$.



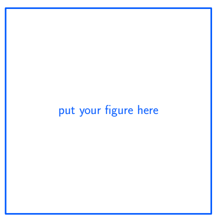
$\omega_l = 0.5$



$\omega_l = 1.0$



$\omega_l = 2.0$



$\omega_l = 4.0$

task 4.4

Gaussian filtering in practice

In the `Data` folder for this exercise, you will find the intensity image

`cat.png`

Read it into a *numpy* array `arrF`.

In the lecture we discussed Gaussian filtering in the space domain and saw that we can think of the process as moving a corresponding filter mask across the image to be filtered. Back then, the question arose how to handle pixels close to the boundary of the image and your instructor said not to obsess about this issue . . .

Indeed, Gaussian filtering is such a common task that *scipy* has us covered. Its *`ndimage`* module contains the function *`gaussian_filter`* which you may use like this

```
sigm = 7.
arrH = img.gaussian_filter(arrF, sigma=sigm, mode='constant')
```

Note, however, that `constant` is not the only possible value for parameter `mode`. This parameter tell the function how to handle the "boundary pixel problem" and if you read the *scipy* manual, you will find that it can also be set to `reflect`, `mirror`, `nearest`, ...

Consider $\sigma \in \{1, 3, 9\}$ and the `mode` parameters just mentioned to compute filtered images `arrH`, write these images as PNG files, and enter them in the figure on the next page.

put your figure here	put your figure here	put your figure here	put your figure here
$\sigma = 1$, `const`	$\sigma = 1$, `reflect`	$\sigma = 1$, `mirror`	$\sigma = 1$, `nearest`
put your figure here	put your figure here	put your figure here	put your figure here
$\sigma = 3$, `const`	$\sigma = 3$, `reflect`	$\sigma = 3$, `mirror`	$\sigma = 3$, `nearest`
put your figure here	put your figure here	put your figure here	put your figure here
$\sigma = 9$, `const`	$\sigma = 9$, `reflect`	$\sigma = 9$, `mirror`	$\sigma = 9$, `nearest`

What do you observe? Does the manner in which images boundaries are handled make a huge difference in practice?

enter your discussion here . . .