Worksheet 7: Signal Processing and Data Analysis

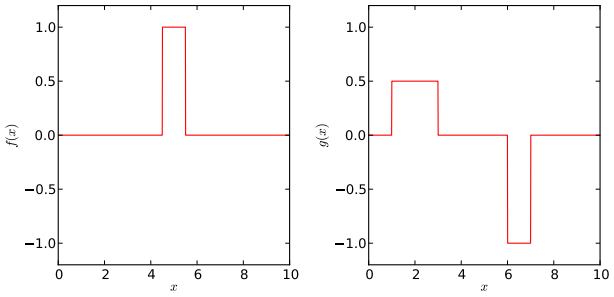
June 1, 2016

General Remarks

- The deadline for handing in the worksheets is **Tuesday, June 7th, 2016, 10:00**.
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to `mkuron@icp.uni-stuttgart.de`.
- Please try to only hand in a single file that contains your program code for all tasks. If you are asked to answer questions, you should do so in a comment in your code file or a text block in your IPython notebook. If you need to include an equation or graph, you can do that in your IPython notebook, or you may hand in a separate PDF document with all your answers, graphs and equations.

the data needed in this worksheet can be found in the file `ws7.pkl.gz` on the home page. The notebook `ws7.ipynb` and the Python script `ws7.py` show how to load the data from this file using the Python module `pickle`.

Task 7.1: Convolution and Cross-correlation (2 points)



- **7.1.1** (1 point) Sketch (*e.g.* via a drawing program) the convolution $f \star g$ of the two functions in the above plot.
- **7.1.2** (1 point) Sketch the cross-correlation $C(\mathrm{f}, g)$ of the two functions in the above plot.

Hint Really do these sketches manually and do not directly employ the Python functions. That way, you will much better understand what cross-correlation and convolution does!

Task 7.2: Radar Signals (4 points)

Radar or sonar are employed to measure the distance of different objects to the radar station. To do that, a well-defined signal is sent that is reflected by the objects. From the transit time of the signal, the distance can be determined.

Unfortunately, the reflected signal at the receiver is very noisy and signals of other origin are superimposed. Therefore, the cross-correlation of the original and the reflected signal is computed to determine the transit times.

The data file contains (artificially generated) data of a radar signal. The example programs contain the code to read the data from the file. Afterwards, `ts` contains the time axis of the signal, `signal1` and `signal2` are the sent signals. `ref1` and `ref2` contain the noisy reflected signal of both signals.

Copy the Python function `kreuzkorrelation()` from the lecture's script (chapter 4.3) that computes the cross-correlation of two data series.

- **7.2.1** (1 point) Plot both sent and reflected signals against the time. Can you guess how man radar echos are contained in the reflected signal? What transit times do these echoes have?
- **7.2.2** (1 point) Use the function to cross-correlate each signal with itself (*i.e.* to compute the autocorrelation of the signal). Plot the result.

  Answer the following questions: What is the purpose of the plots in the context of radar signals? What is the advantage of the second signal?

  #### Hints

  - To better see the autocorrelation, use the function `numpy.roll()` to shift the result by half the length of the result.
  - The second signal is called a *chirp*.
- **7.2.3** (2 points) Use the function to compute and plot the cross-correlation of the signal and their respective radar echos.

  Answer the following questions: How many radar echos are visible in the signal? What transit times can you read off from the cross-correlation plots?

Task 7.3: Data Analysis of a Simulation (4 points)

The file `simulation.npy` contains data of a Molecular Dynamics simulation of a charged colloidal particle (Charge +300, Radius 50 nm) in a solution of monovalent and multivalent ions (Charges −1, +1 and −3)[1].

The sample programs show how to load the data from the file. Here, `ts` contains the time axis and `n1`, `n2` and `n3` are the number of ions of the different types close to the surface of the colloid. What one wants to know are the mean values of these observables in thermodynamic *equilibrium*. Note that the simulation is not started in thermodynamic equilibrium – initially, the different ions were randomly distributed around the colloidal particle. Therefore, the simulation first has to *equilibrate*, *i.e.* it has to run for a while until equilibrium is reached. In the observable, this manifests itself in a clear trend until it only fluctuates around the equlibrium mean values.

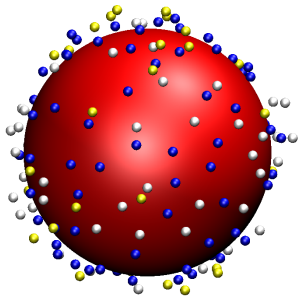


Figure 1: Charged colloid and salt ions at the surface

Therefore it is necessary to first determine at what time the simulation is equilibrated, *i.e.* when it does *not* show a trend anymore (the *equilibration time*). The mean value of the observable should only be computed with the observable values *after* the equilibration.

Unfortunately, there is no formal criterium to determine when equilibrium is reached, instead it has to be determined visually. The observable fluctuates strongly, but the mean value should be determined more accurately than these fluctuations, therefore a simple plot of the observable does not suffice. Instead, one should *smooth* the time series on different time scales to determine equilibration time.

To smooth the time series, one can convolute the data with a Gaussian of a given standard deviation σ. The standard deviation of the Gaussian plays the role of the *time scale*.

- **7.3.1** (1 point) Implement a Python function that convolves two arbitrary time series with help of NumPy's FFT functions.

  Hint To verify the results of your function, you can compare them to the function from SciPy `scipy.signal.fftconvolve()`.

- **7.3.2** (2 points) Use the function to smooth the time series `n1`, `n2` and `n3` on the time scales $100, 1000, 10000$ and create plots of the raw and smoothed time series. Be aware that the time scale of the simulation and the index of the data series are not the same!

- **7.3.3** (1 point) Estimate the equilibration time of the simulation. Extend the Python program to compute the equilibrium mean values of the three observables, *i.e.* the mean values of the data series *after* the equilibration time.

References

[1] O. Lenz and C. Holm. Simulation of charge reversal in salty environments: Giant overcharging? *Eur. Phys. J. E*, 26:191–195, 2008.