Physik auf dem Computer **SS 2014**

Worksheet 5: Fourier Transforms

May 9, 2014

General Remarks

- The deadline for the worksheets is **Wednesday, 14 May 2014, 10:00** for the tutorial on Friday and **Friday, 16 May 2014, 10:00** for the tutorials on Tuesday and Wednesday.
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to
  - Johannes (`zeman@icp.uni-stuttgart.de`; Tuesday, 9:45–11:15)
  - Tobias (`richter@icp.uni-stuttgart.de`; Wednesday, 15:45–17:15)
  - Shervin (`shervin@icp.uni-stuttgart.de`; Friday, 11:30–13:00)

Throughout this worksheet, we will use the periodic Runge function $f(x) = \frac{1}{1+\cos^2(x)}$ on the domain $[0, 2\pi]$ and the Lennard-Jones-Function $g(x) = x^{-12} - x^{-6}$ on the domain $[1, 5]$. The Python program `ws5.py` and the IPython Notebook `ws5.ipynb` defines corresponding Python functions.

Task 5.1: Fourier Transform in NumPy (3 points)

- **5.1.1** (2 points) Write a Python program that does the following for the functions $f(x)$ and $g(x)$:
  1. It creates a data series of 2048 equidistant points on the respective domains and their function values.
  2. It uses the Python function `numpy.fft.fft()`, to compute the Fourier coefficients of the data series.
  3. It sets the Fourier coefficients of $|k| > k_{\max}$ where $k_{\max} \in \{5, 10, 50\}$ to 0, *i.e.* it truncates the Fourier series at $k_{\max}$.
  4. It back transforms the truncated Fourier series to real space with help of the Python function `numpy.fft.ifft()`. The output vector is the *reconstructed function*.
  5. It creates a plot of the function and the different reconstructed functions.

  ### Hints

  - Have a look at how the Fourier coefficients are stored to understand what coefficients you have to zero out. Read the docs of `numy.fft`.
  - Note that in principle, it is not necessary to store the Fourier coefficients that are 0. Insofar, truncating the Fourier series can be seen as a very effective data compression: instead of storing 2048 function values, it is enough to store just 50 Fourier coefficients to be able to reconstruct the data series to a high degree of accuracy.
- **5.1.2** (1 point) Why does the reconstruction of the function $g(x)$ show artefacts at the right boundary even when 50 Fourier coefficients are used, while this is not the case for function $f(x)$ even when only 10 Fourier coefficients are used?

Task 5.2: Discrete Fourier Transform (3 points)

Implement a Python function `dft_forw()` that computes the discrete Fourier transform of a data series and a Python function `dft_back()` for the back transformation. The results should be identical to the results of `numpy.fft.fft` and `numpy.fft.ifft`. Recreate the plots from the previous task using these functions.

Hints

- In the Python functions, you can use the data type `numpy.complex`. The complex number $x = 1 + 2i$ can be written as `x=1.0+2.0j` in Python. All mathematical functions (in particular `numpy.exp()`) can also use such complex numbers.
- `ws5.py` and `ws5.ipynb` define a function `verify_fourier(forward, back)`, where `forward` and `back` are Python functions that do a Fourier forward respectvely back transform of the input array. `verify_fourier` checks whether the Fourier transforms work correctly by transforming and backtransforming a random data series and gives some hints what is wrong if it doesn't. This function should be useful while developing the transform.

Task 5.3: Fast Fourier Transform (2 points)

Implement the Python functions `fft_forw()` and `fft_back()`, that do the same as `dft_forw()` and `dft_back()` but use the Fast Fourier transform algorithm. Recreate the plots from the previous tasks using these functions.

Task 5.4: Timing the Fourier Transforms (2 points)

- **5.4.1** (1 point) Using `timeit.timeit`, measure the run time of the Python DFT functions from task 5.2, the FFT functions from task 5.3 and the NumPy FFT functions for $N \in \{4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$ via the Python function `timeit.timeit`. Create a loglog plot of the run times versus N.

  Hint Do not forget to use the argument `number=1` to `timeit.timeit`, otherwise it will run the functions 1000000 times.

- **5.4.2** (1 point) What do you learn from this plot?