Physik auf dem Computer **SS 2016**

Worksheet 6: Discrete Fourier Transforms

May 25th, 2016

General Remarks

- The deadline for handing in the worksheets is **Tuesday, May 31th, 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.

Task 6.1: 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 task 5.2.1 on the previous worksheet using these functions to show that you get sensible results.

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.
- `ws6.py` and `ws6.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 6.2: Fast Fourier Transform (5 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 task using these functions.

Task 6.3: Timing the Fourier Transforms (2 points)

- **6.3.1** (1 point) Using `timeit.timeit`, measure the run time of the forward and back Python DFT functions from task 6.1, the forward and back FFT functions from task 6.2 and the forward and back NumPy FFT functions for $N \in \{4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$ via the Python function `timeit.timeit`. As input data for the FFT/DFT, you may use random numbers or the values of $g(x)$ or $h(x)$ from the previous worksheet. Create a double-logarithmic 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.

- **6.3.2** (1 point) What do you learn from this plot?