# Computational Physics II — Project 5

Jan O. Haerter

April 7, 2022

**Hand-in date: 6 a.m., Tuesday, Apr 26, 2022**

In total, **100 pts** can be achieved by solving the problems below (see annotation).

---

## 1 Discrete Fourier Transform

Many implementations of the Fast Fourier Transform (FFT) exist and the pedagogical value of re-implementing one ourselves is limited. The purpose of this exercise is rather, to gain some experience with the Discrete Fourier Transform (DFT) and its applications.

1. Use the sound recorder of your laptop to record a short (maybe five seconds) wav file, say a single guitar string or singing a single note Use pythons wavefile module to read in the wav file (5 pts).

   ```
   from scipy.io import wavfile
   samplerate, dataA = wavfile.read('filename.wav')
   ```

2. In two separate panels plot the entire timeseries as well as a short section (say, .1 sec). Describe the functional form of the short section: is there a dominant frequency? (10 pts)

3. implement a discrete Fourier transform *hint*: for some speedup, store all $W_N^k$ for $k \in \{0, \ldots, N-1\}$ in an $N$-element array and re-use these values. (5 pts)

4. import python's implementation of the Fast Fourier Transform

   ```
   import scipy.fft as fft
   ```

   and obtain the power spectrum $S(\omega) = \tilde{f}^*(\omega)\tilde{f}(\omega)$ for a section of your signal, plot it as a function of $\omega$ and discuss the findings. Does the spectrum match your intuition about the note you have recorded? Does your DFT give the same spectrum as the built-in FFT (apart from round-off errors)? On a linear vertical scale, is there a clear maximum frequency, i.e. a spike (apart from low-amplitude noise)? Note this frequency, $\omega_{max}$, for later use. (15 pts)

5. use a timer to time both your version of the DFT and the python FFT

   ```
   import time
   time.perf_counter()
   ```

For sample sizes of $N = 10^a$ with $a \in \{1, 5\}$ perform both DFT and FFT and determine the scaling of computing time vs. $N$. (10 pts; *Hint*: Consider plotting on a log-log scale to extract the power-law exponent for DFT.)

6. FFT the entire timeseries. Now, FFT back to the time domain and verify that the timeseries is recovered. Remove a fraction $f_a = 1 - (1/2)^a$, $a \in \{1, 10\}$ of your frequency data by setting the fraction $f_a$ of lowest-amplitude Fourier modes to zero (as measured by the power spectral density $S(\omega)$). From the remaining, non-zero amplitudes again back-transform to the time domain and plot the short .1 sec section of the resulting timeseries. How does this section look, qualitatively, as you remove more and more data (higher compression)? (15 pts) Using

```
wavefile.write('filename_out.wav', samplerate, timeseries)
```

write a wave file for each $a$ (Note that the timeseries might have to be normalized/rescaled to avoid distortions). Listen to these files using your laptop and document your impressions: Which acoustic changes do you find with larger $a$? How is this impression reflected in the plot of the timeseries? (10 pts)

7. Use only each 10th or 100th sample in your original timeseries to perform the FFT. From $\tilde{f}(\omega)$, obtained using each of these two smaller sample sizes, try to recover $f(t)$ by back-transforming. Plot the .1 sec section of your timeseries to compare the data for the different cases. Discuss your findings along the Shannon-Nyquist theorem. You may find the frequency $\omega_{max}$, obtained higher up, useful. (15 pts)

# 2 Green's function for ODEs

Consider the forced harmonic oscillator

$$\frac{d^2y}{dt^2} + \beta\frac{dy}{dt} + \omega^2 y = f(t)\,, \tag{1}$$

where

$$\frac{d^2G(t;\tau)}{dt^2} + \beta\frac{dG(t;\tau)}{dt} + \omega^2 G(t;\tau) = \delta(t-\tau) \tag{2}$$

defines the Green's function $G(t;\tau)$. Obtain the Green's function $\tilde{G}(\omega;\tau)$ by Fourier transform. (15 pts)

**General remarks for all Projects.** You will have to (i) analyze the problem, (ii) select an algorithm (if not specified), (iii) write a Python program, (iv) run the program, (v) visualize the data numerical data, and (vi) extract an answer to the physics question from the data. Which checks did you perform to validate the code? State the results you got for these tests. For each project you will submit a short report describing the physics problem, your way of attacking it, and the results you obtained. Provide the documented Python code in such a form that we can run the code. A Jupyter Notebook including the code and report is fine but not necessary.