Humboldt-Universität zu Berlin
Institut für Physik
AG Theoretische Optik & Photonik
https://top.physik.hu-berlin.de/research

Computational Photonics/ Computerorientierte Photonik (SoSe 2019)

Exercise 1 (released on Apr 11, 2019) *Due on Apr 18, 2019 at 8 am!*

Lecturer: Prof. Dr. Kurt Busch
Tutor: Dr. Dan-Nha Huynh

1 Introduction to Matlab

a) Visualization (1D) Use the `plot` function to plot a graph of a Gaussian bell curve $f(x) = \exp(-x^2)$. Since Matlab is most efficient when working with matrices and vectors, it is best practice to store all points x at which f shall be evaluated in one vector x_i and use Matlab's feature of evaluating a function for all elements of a vector at once, resulting in another vector with all function values $f_i = f(x_i)$.

b) Visualization (2D) Experiment with the visualization of the two-dimensional Gaussian $g(x, y) = \exp(-(x^2 + y^2))$. Useful functions can be `meshgrid`, `mesh`, `surface`, `pcolor` and `contour`.

c) Visualization of Vector Fields In electrodynamics, we often deal with vector fields. Use the function `quiver` to visualize the vector fields $\mathbf{E}_1(x, y) = (-y, x)$ and $\mathbf{E}_2(x, y) = -(y, x)$.

d) Numerical Integration Implement a function in Matlab that computes the function $h(x) = \frac{\exp(1-\frac{1}{x})}{x}$. The input of your function shall be a vector x_i representing the space points on the abscissa and the function should return the vector h_i with the corresponding function values.

The function `integral` numerically integrates functions. Use `integral` to integrate $h(x)$ over the interval (0,1). In GNU Octave, you may use the function `quadl`.
Think about the integration limits. Which problem could exist and how could you resolve it?

2 Fourier Transformation

From the lecture, we recall that the numerical Fourier transform is calculated according to

$$F_n = \sum_{k=0}^{N-1} \exp(-i2\pi nk/N) f_k, \tag{1}$$

where f_k denotes the sampling points of the function $f(t)$ and F_n represents the corresponding Fourier coefficients.

a) Discrete Fourier Transformation Implement a function that calculates the Fourier coefficients F_n of the function $f(t)$ by regarding equation (**??**) as a matrix–vector product. Set up an appropriate $(N \times N)$-matix W_{nk} and calculate the Fourier coefficients by multiplying it with a vector whose components are the discrete sampling values $f_k = f(t_k)$.

b) Fast Fourier Transformation In the FFT algorithm, the sum in the numerical Fourier transformation is split into two parts, one with the even-numbered and the other with the odd-numbered coefficients. These two new sums can be rewritten as the discrete Fourier transform of the even/odd discrete sampling points f_{2j} and f_{2j+1}. Thus, the number of sampling points has to be an integer power of 2 ($N = 2^l$). A comparatively easy way to implement the FFT is via a recursive function, i.e. a function that repeatedly calls itself. In particular, it is not necessary to explicitly sort the input data into bit-reversed order. At each recursion step, the length of the input vector for subsequent function calls is halved, until the trivial case of a Fourier transform of length 1 is reached.

Implement the Fourier transformation by a recursive algorithm. (Hint: It may be helpful to explicitly write out the Fourier transform for F_k and $F_{k+\frac{N}{2}}$ in terms of the even and odd subtransforms as well as the so-called twiddle factor $w = \mathrm{e}^{-\frac{2\pi i}{N}}$).

Compare the computation times of both implementations for various numbers N of discretization points and try to find the respective scaling laws. (Hints: useful functions in Matlab are `tic` and `toc`; doubly logarithmic plots help to recognize the exponent of an algebraic function).

c) Comparison with Matlab FFT Matlab provides its own `fft` routine. Use this function to calculate the Fourier transform and compare the results and the computation time with those of your own program.

d) Aliasing If the Fourier transform of a function is not bandwidth limited, you have to use a method called oversampling in order to numerically calculate the Fourier transform of this function. This means that you calculate more Fourier coefficients than actually required. Thereby, the spurious information from spectral components outside the transformed window is distributed over a large number of coefficients. Calculate the discrete Fourier transform of a Gaussian pulse $f(t) = \exp(-t^2)$ and compare it with the analytically calculated Fourier coefficients. What happens if you choose the sampling period Δt too large?

(In order to compare the results, you have to use $\tilde{f}^{\mathrm{exact}}(\nu_n) \approx N\Delta t \exp(in\pi)\, \tilde{f}_n^{\mathrm{fft}}$. Why?)

3 Transformation of Maxwell's equations to dimensionless units

When simulating Maxwell's equations, we do not want to deal with the natural constants (in our case μ_0 and ϵ_0). Define new field- and time-variables to transform both Maxwell's curl equations

$$\frac{\partial \mathbf{H}(\mathbf{r},t)}{\partial t} = -\frac{1}{\mu_0 \mu(\mathbf{r})} \nabla \times \mathbf{E}(\mathbf{r},t), \qquad \frac{\partial \mathbf{E}(\mathbf{r},t)}{\partial t} = \frac{1}{\epsilon_0 \epsilon(\mathbf{r})} \nabla \times \mathbf{H}(\mathbf{r},t)$$

in dimensionless units, i.e., eliminate ϵ_0 and μ_0 by a variable transformation.

— Discussion in the exercise on Apr 18, 2019 —

Problems marked with * are voluntary. All others are **due** on **Apr 18, 2019** at **8am** via e-mail to dhuynh@physik.hu-berlin.de or has to be brought to the exercise class on a flash drive. Analytical solutions are accepted handwritten or in the .pdf format, numerical solution in a digital version.