Exercise sessions 2–3: Discrete Fourier transform. NMR spectroscopy.

Exercise 1 [10 pts]

Files: `test.m`

The Fourier transform relates real and reciprocal space representations of a function. While the conventional Fourier transform operates with continuous functions $f(x)$, the discrete Fourier transform (DFT) operates with discretized data $f_n, n \in \mathbf{N}$.

1. Write a Matlab function which implements the discrete Fourier transform (DFT) as explained in the lecture. Please, folow the conventions on naming and the types of input/output data.
2. Verify your DFT implementation by transforming simple model functions and comparing your result with MatLab's function `fft`. See supplied script `test.m` for examples. Make sure that your implementation handles correctly different possible input arrays.

Listing 1: 'mydft.m'

```
function result=mydft(input_array)

% mydft computes the discrete Fourier transform.
%
% Arguments:
%
%     input_array (1D array complex): data to transform;
%
% Returns: 1D array complex, transformed data of the same shape as
% an input array.

% Following is a temp;ate. This should be replaced with your own code.
result = fft(input_array);

end
```

Submit: file `mydft.m` with your implementation of DFT.

Exercise 2 [20 pts]

File: `FID.dat`

In a ^{1}H-NMR spectroscopy experiment, the transitions between the quantum states of nuclear spins of protons in applied magnetic field are probed to determine the structure of molecules and solids. Specifically, nuclear spins are excited by means of a radio frequency pulse and the signal due to their relaxation is recorded. This signal is called the free induction decay (FID). The Fourier transform of the FID results to the NMR spectrum, that is the intensity of the signal as a function of the chemical shift. Nuclear spins in different chemical environments would correspond to different peak positions. The area of the peaks in the spectrum is proportional to the number of protons of the corresponding type.

1. The columns in the data file `FID.dat` contain the real and imaginary part of the free induction decay of a molecule with formula $C_2H_6O_2$. Import the file in a two-column matrix and merge the real and imaginary parts into a single complex vector.
   Hint: Use `importdata` and `complex`.

2. The time interval between two consecutive data points is $\Delta t = 83.2\ \mu$s. Associate time to each data point, setting the initial time to zero. That is, for the n-th point, $t_n = n\Delta t$. Plot the real and imaginary parts of the free induction decay as a function of time.

3. Calculate the discrete frequencies f_n which span the interval $[-f_c,\ f_c]$ with $f_c = 1/2\Delta t$ being the Nyquist frequency.

4. Using the previously implemented function `mydft.m` perform the discrete Fourier transform of the complex FID. Plot (*i*) the real part, (*ii*) the imaginary part and (*iii*) the power spectrum as a function of the chemical shift δ_n (in ppm), defined as:

$$\delta_n = \frac{f_n - f_0}{\nu_0} \times 10^6 \tag{1}$$

   with $f_0 = 1067.93$ Hz being the internal reference and $\nu_0 = 800.224$ MHz being the operating frequency of NMR spectrometer. Compare the result of the Fourier transform obtained using your `mydft.m` function with the `fft` function of MatLab.

5. How many peaks do you observe in the spectrum? How many types of protons with distinct chemical environment does the investigated molecule have?

6. What is the ratio between the number of protons of different types? Note, the integral of the peak in the power spectrum scales as square of the number of protons.

7. Would you be able to suggest the structural formula of the molecule using available data? Note, C, O and H atoms are able to form covalent bonds with 4, 2 and 1 neighbors, respectively.

Submit: your report as a PDF file that includes

- [5 pts] plots of the real and imaginary parts of the FID as a function of time;
- [10 pts] plots of the real part, imaginary part and power spectrum as a function of the chemical shift;
- [5 pts] correct explanation of the number of peaks, the ratio of the peaks that correspond to different proton types and the structural formula of the molecule.

Exercise sessions 4–5: Fast Fourier transform. Image filtering.

Exercise 1: Radix-2 FFT algorithm [10 pts]

Files: `test.m`

The radix-2 recursive algorithm performs the discrete Fourier transform efficiently, hence belonging to the family of Fast Fourier transform (FFT) algorithms. The price paid for computational efficiency is a more sophisticated implementation involving recursion and workarounds for odd-sized arrays. The results of a discrete Fourier transform (DFT) and an FFT are **exactly** the same.

1. Write a function that implements the radix-2 variant of the FFT algorithm as explained in the lecture. Consider only the simplest case of $N = 2^r$ sampling points, where r is a natural number.

2. Verify your FFT implementation by transforming simple model functions and comparing your result with MatLab's function `fft`. Take a look into the supplied test script `test.m` for some examples. Make sure that your implementation handles correctly every possible input array. Check whether your implementation of the FFT is indeed faster than a simpler straightforward implementation of DFT.

Listing 1: 'myfft.m'

```
function result=myfft(input_array)

% myfft computes discrete Fourier transform with a radix-2.
%
% Arguments:
%
%     input_array (1D array complex, size 2^N, N>0): data to
%     transform;
%
% Returns: 1D array complex, transformed data of the same shape as
% an input array.

% Following is a stub. This should be replaced with your own code.
result = fft(input_array);

end
```

Submit: file `myfft.m` with your implementation of the radix-2 FFT algorithm.

Exercise 2: Fourier ptychography [20 pts]

Files: `image_intensity.png`, `image_phase.png`, 25 files named `ptychography_i_j.png` and `cutoff.png` in archive `images.zip`

The amount of information extracted by a microscope in an experiment is intrinsically limited, and is subject to the competition between resolution and field of view of the image. Fourier ptychography is a method devised to address this issue using properties of the Fourier Transform (FT).

A typical microscope works by illuminating a thin sample with light (a plane wave), and passing the scattered light through two lenses. The resolution of the lens is limited by its angle of acceptance, and is proportional to $\frac{\lambda}{n \sin\theta}$ where λ is the wavelength of light, n the refractive index of the lens and θ the

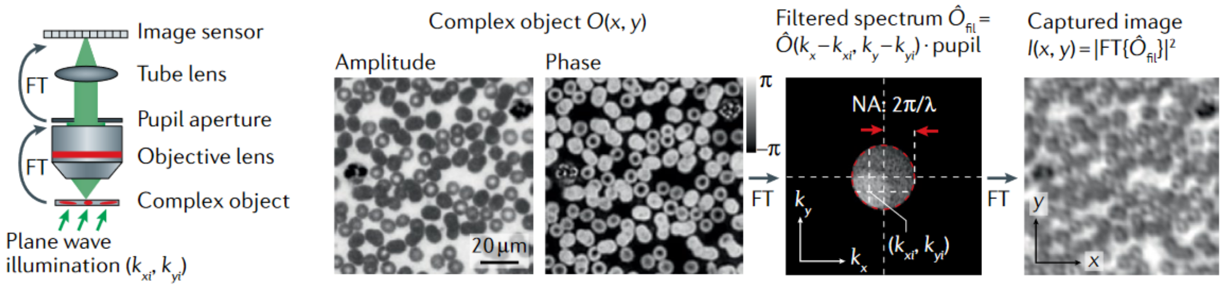

Image sensor
Tube lens
Pupil aperture
Objective lens
Complex object
FT
FT
Plane wave illumination (k_{xi}, k_{yi})
Complex object $O(x, y)$
Amplitude
Phase
20 µm
π
−π
FT
Filtered spectrum $\hat{O}_{fil} = \hat{O}(k_x - k_{xi}, k_y - k_{yi}) \cdot$ pupil
NA · 2π/λ
k_y
k_x
(k_{xi}, k_{yi})
FT
Captured image $I(x, y) = |FT\{\hat{O}_{fil}\}|^2$
y
x

Figure 1: An illustration of the microscope image acquisition process used in Fourier ptychography. Sample is illuminated at an angle, with light of wavevector (k_{xi}, k_{yi}, k_z). This results in the shift of the Fourier transform by vector $(-k_{xi}, -k_{yi})$. Images reproduced from G. Zheng *et al.*, Nat. Rev. Phys. 3, 207 (2021). ©Springer Nature Limited 2021.

maximum angle of acceptance. Mathematically, the process can be modeled as follows: let the studied object be described by a complex 2D function $O(x, y)$. For a lens of infinite resolution, the Fourier Transform $\hat{O}(k_x, k_y)$ of the image would be formed in the back focal plane. However, for a real lens the reciprocal space image is rather $\hat{O}(k_x, k_y) \cdot a(k_x, k_y)$, where $a(k_x, k_y)$ is an amplitude transfer function of finite radius. This function can be effectively approximated as a low-pass filter of radius $r_{cut} = \frac{2\pi n \sin\theta}{\lambda}$, a function value 1 below r_{cut}, 0 otherwise. Upon passing through the second lens, an image $FT^{-1}(\hat{O}(k_x, k_y) \cdot a(k_x, k_y))$ is formed on the detector. The detector then records the intensity of light, which is proportional to $|FT^{-1}(\hat{O}(k_x, k_y) \cdot a(k_x, k_y))|^2$. As can be seen, resolution of the image is reduced by the low-pass filter, and all phase information is lost.

In Fourier ptychography, instead of taking one image under perpendicular illumination $\propto e^{ik_{zi}z}$, several images are taken under different angles of illumination, *i.e.* using tilted plane wave $\propto e^{ik_{zi}z}e^{ik_{yi}y}e^{ik_{xi}x}$. Tilting the angle of illumination adds additional phase to every point of our complex object, resulting in $O(x, y)e^{ik_{yi}y}e^{ik_{xi}x}$. Using properties of the Fourier Transform, it can be shown that the FT of this object is $\hat{O}(k_x - k_{xi}, k_y - k_{yi})$, and the recorded intensity is proportional to $|FT^{-1}(\hat{O}(k_x - k_{xi}, k_y - k_{yi}) \cdot a(k_x, k_y))|^2$. Hence, the cutoff circle is effectively shifted by k_{xi}, k_{yi} (recall the shift theorem!) and an image formed involving higher frequencies was recorded. If one records images under several angles, it is possible to reconstruct the Fourier Transform of the original object to a higher cutoff radius, hence enhancing the resulting resolution.

The reconstruction is performed using the following algorithm:

1. Start with a complex guess function $I(x, y)$, which can be as trivial as a constant with zero phase, and calculate its Fourier transform $\hat{I}(k_x, k_y)$
2. Obtain a new object $\hat{I_{cut}}(k_x, k_y)$ by cutting out a circle C of radius r_{cut} around the centre k_{xj}, k_{yj} in Fourier space, *i.e.* set all values of $\hat{I}(k_x, k_y)$ to 0 outside the circle.
3. Perform inverse FT of $\hat{I_{cut}}(k_x, k_y)$, resulting in complex function $I_{cut}(x, y)$
4. Retain the phase of $I_{cut}(x, y)$, but replace the magnitude with the experimentally recorded one, *i.e.* from the image formed by illumination with tilted light $\propto e^{ik_{yj}y}e^{ik_{xj}x}$
5. Perform Fourier Transform of the resulting object, and use it to replace the values of $\hat{I}(k_x, k_y)$ inside circle C.
6. Repeat steps 2–5 for every experimental image (*i.e.* for each circle in Fourier space).
7. Repeat steps 2–6 until convergence (in the exercise we will not test for convergence, but rather repeat these steps a certain number of times).

The goal of this exercise is to demonstrate resolution enhancement and phase retrieval by means of Fourier ptychography. The archive on Moodle contains 27 image files. `image_intensity.png` and `image_phase.png`

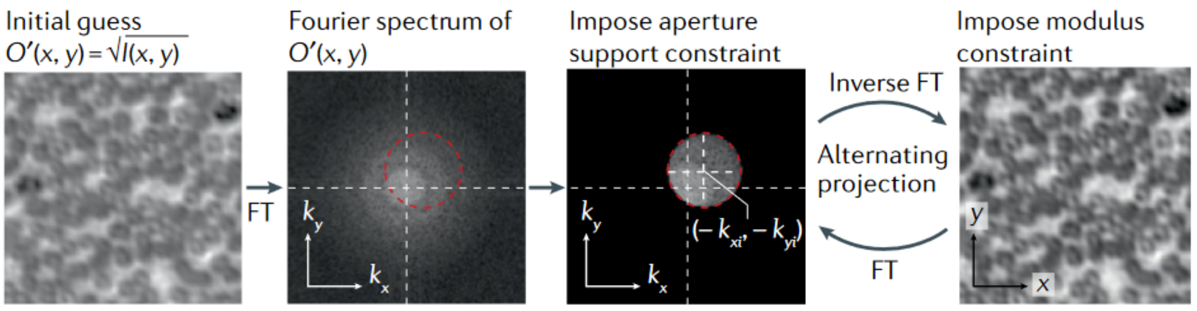

Figure 2: An illustration of the phase retrieval algorithm. Initial guess is taken and circular filter centered at $(-k_{xi}, -k_{yi})$ is applied to its Fourier transform. Inverse FT is performed, phase of the guess is retained, but the magnitude is replaced with experimentally recorded one. Fourier transform of the result is performed and the values of the Fourier transform within the circular filter are replaced (alternating projection). Images reproduced from G. Zheng *et al.*, Nat. Rev. Phys. 3, 207 (2021). ©Springer Nature Limited 2021.

are the original intensity and phase, while `ptychography_i_j.png`, with $i, j = -2, -1, 0, 1, 2$, are the simulated “microscope” images. The simulation took the Fourier transform of the original complex object, applied circular filters in Fourier space, took inverse Fourier transform and recorded the resulting intensity. The circular filter had a radius of 100 pixels, and its centre displaced by [i*50, j*50] relative to the centre of the image.

1. Display `image_intensity.png` and `image_phase.png` with Matlab. To do that:
   - load the image as a red-green-blue (RGB) three-dimensional array with `imread`,
   - convert it to the two-dimensional a grayscale array using `rgb2gray` and `double`,
   - plot the image with `imshow`.
2. Perform the Fourier transform of any `ptychography_i_j.png`, $i \neq 0$ and $j \neq 0$, using the `fft2` and `fftshift` functions of Matlab. Visualize the result using `imshow`. Explain why the result observed in Fourier space is not contained within the original circular low-pass filter.

   Hint: To see clearly the Fourier transformed image, you will need to tune the contrast. To do so you can either apply a linear and/or logarithmic transformation to the pixel intensities.
3. Next few points will walk you through the implementation of the algorithm outlined above. Take an array of ones with the size of the original image. This will be our initial guess for the complex object we are trying to recreate (do not worry about the initial guess being real). Perform the Fourier transform of the guess, and apply a cutoff filter with **radius of 100**, centred at [255.5,255.5], to the transformed image. Display the result (see `cutoff.png` and the listing below).

```
N = 525; %replace with the actual size

cutoff = zeros(N);

for i=1:N
for j=1:N
if (i-(N-1)/2)^2 + (j-(N-1)/2)^2 < 100^2
cutoff(i,j)=1;
else
cutoff(i,j)=0;
end
end
end
```

4. This particular cutout corresponds to the `ptychography_0_0.png` image. Take the inverse Fourier transform of the result from point 3, keep the phase of the object, but replace the magnitude by the square root of the intensities from `ptychography_0_0.png`. Perform the Fourier transform of the result and use it to replace the values within the cutout from point 3. Display the result.

5. Repeat points 3 and 4 for every `ptychography_i_j.png`, without displaying the results. To shift the cutoff filter, you can use the `circshift` function of Matlab. Combine the 25 cutouts to form a better guess Fourier transform of the original complex object. In the overlapping regions, take values of any of the overlapping circles. Perform inverse Fourier transform and display the intensity and phase of the guess.

6. Repeat this protocol 20 times. Display the final intensity and phase. Comment in relation to the provided images `image_intensity.png` and `image_phase.png`, as well as `ptychography_0_0.png` recorded with perpendicular incident light.

   Note: for proper visual comparison of the intensities, the plots of the original and final guess should have the same low-high range. This can be achieved by explicitly specifying the range:
   `implot(G,[min(O,[],'all'),max(O,[],'all')])`, where `G` and `O` are the image matrices of the guess and the original, respectively.