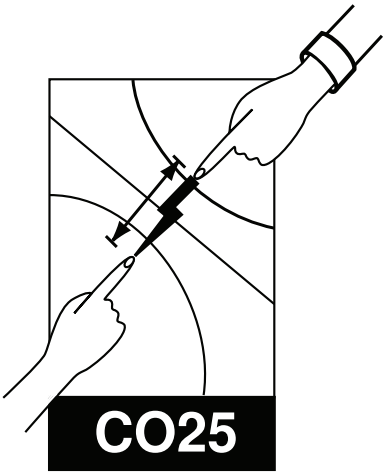
CO25: Laplace's equation



revised EG, MK January 2022

Preface

It is common practice in modern software engineering to write programs in a modular and standardised way. To help you get started, appendix A provides you with one or more specific function headers which you **MUST** use to write the functions around which your program should be built and in order to receive a satisfactory mark. Your comments in the header of each function must include:

- author and date,
- purpose (a brief description of what the function does),
- inputs and outputs,
- and, if appropriate, any constraints or limitations of use.

Do not forget that you will also need to keep good records of your progress during this practical in your logbook. If you make plots and/or write any notes electronically, ideally, you would print them and affix them into the pages of your logbook. You should have comments in function and script headers of your code, as well as comments within your code. These aspects may all be considered at marking time.

The function header template provided is based on the default language of the Lab: MATLAB. Alternatively, you may write your program in python if you wish (check for approval by Head of Lab to use any other languages). Regardless of the chosen language, the functional equivalent of the function introduced in appendix A must be included in the code.

Assessment

Please see the Part A Guide in Canvas for Computing Lab deadlines as well as availability (schedule) of Computing Lab Demonstrators who mark your work as well as offer help and advice. Before you meet with a demonstrator for marking, **you must** upload your work into Canvas (both your code and your report). Your code must be uploaded in a single file (see instructions in Canvas) and must include any functions required by the script. Your report must be written following the requirements described in AD34 — *the art of scientific report writing*[1]) and uploaded into Canvas in valid pdf format.

1 Introduction

The solution of Laplace's equation by successive over relaxation (see the references for details) is found for a square domain, and compared with the analytic solution.

2 Physical description

In this exercise, you will try to find solutions to Laplace's equation

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0 \tag{1}$$

in a rectangular domain with ψ set to some prescribed function along the boundary of the domain.

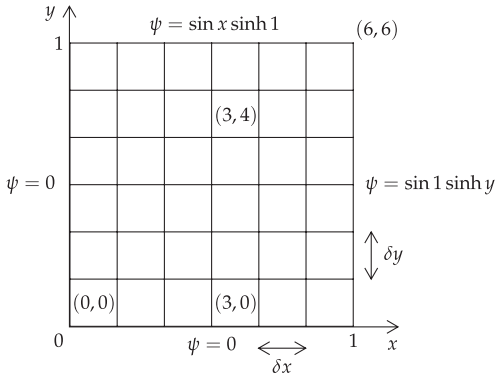
[1] `www-teaching.physics.ox.ac.uk/practical_course/Admin/AD34.pdf`

3 Numerical approach

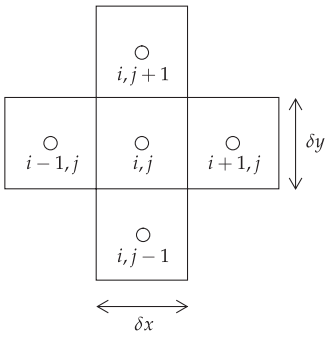
An approximation for Laplace's equation is

$$\frac{(\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j})}{(\delta x)^2} + \frac{(\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1})}{(\delta y)^2} = 0 \tag{2}$$

where the grid points are arranged as in figure 1a and δx, δy are the distances between grid points in the x and y directions.



(a) Plot of the grid layout for the case of a uniform 7×7 grid, with δx and δy the distance between neighbouring points in the x and y directions.



(b) Adjacent grid boxes in a numerical calculation.

Figure 1: Schematic of the discretisation of Laplace's equation.

If the above approximation to $\nabla^2\psi = 0$ is applied at every interior point of the domain, then a system of linear equations results for $\psi_{i,j}$. This may be solved by direct methods, but a useful approach to solving large systems of equations is to use an iterative procedure. Equation 2 may be written as

$$\psi_{i,j} = \frac{(\psi_{i+1,j}+\psi_{i-1,j})/(\delta x)^2 + (\psi_{i,j+1}+\psi_{i,j-1})/(\delta y)^2}{2\left(\frac{1}{(\delta x)^2} + \frac{1}{(\delta y)^2}\right)}. \tag{3}$$

An iterative approach based on equation 3 using the Gauss-Seidel technique would work but is very slow, so a procedure known as *successive over-relaxation* is preferable, in which case we consider

$$\psi_{i,j}^{m+1} = \alpha\overline{\psi}_{i,j} + (1-\alpha)\psi_{i,j}^{m} \tag{4}$$

where $\overline{\psi}_{i,j}$ is calculated from equation 3, and m indicates iteration number.

For the case of a uniform grid ($\delta x = \delta y = \delta$), equation 4 may also be written as

$$\psi_{i,j}^{m+1} = \psi_{i,j}^{m} + \frac{\alpha R_{i,j}}{4} \tag{5}$$

where

$$R_{i,j}^{m} = \psi_{i,j+1} + \psi_{i,j-1} + \psi_{i-1,j} + \psi_{i+1,j} - 4\psi_{i,j}. \tag{6}$$

As a trial calculation, take the boundary conditions

$$\psi = 0 \quad \text{on } x = 0 \text{ and } 0 \le y \le 1 \qquad \psi = \sin x \sinh 1 \quad \text{on } y = 1 \text{ and } 0 \le x \le 1$$
$$\psi = 0 \quad \text{on } y = 0 \text{ and } 0 \le x \le 1 \qquad \psi = \sin 1 \sinh y \quad \text{on } x = 1 \text{ and } 0 \le y \le 1$$

These boundary conditions are chosen because they give for ψ a particularly simple analytic solution. The technique described above will allow you to solve not only this problem, for which a simple solution is available, but also more complicated problems arising from more complicated boundary conditions, more involved boundary shapes than the square used here, or equations more complicated than Laplace's, such as Poisson's equation (see later).

The method of solution is to sweep across the grid in a systematic manner, e.g. as shown in figure 2, using equation 5. See the comment in section 4 concerning the values of $\psi_{i,j}$ used in calculating $R^m_{i,j}$.

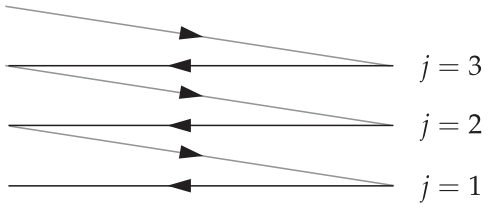


Figure 2: Sweeping across the grid systematically.

This process is iterative and m is the index of the iteration. $R_{i,j}$ is the *residual* which we would like to be zero. If it is zero for all i, j and for some m, then ψ^{m+1} will just be equal to ψ^m, and we can stop the iteration having found the solution. Of course, this will not in general happen. The object of the relaxation method is to provide a systematic way of reducing the residuals. $R_{i,j}$ will (for a convergent iteration process) get smaller for any (i, j) as the process continues and at some point you will decide to stop the iteration. This may be because some *a priori* criterion has been met or because some maximum number of iterations has been exceeded.

The parameter α is the *over-relaxation parameter* because it has a value greater than 1 (and should be less than 2 for convergence). The optimum value is given by

$$\alpha_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \left[(\cos \pi/p + \cos \pi/q)/2\right]^2}} \tag{7}$$

where p and q are the number of grid spacings in the x and y directions. In this case, since $p = q$, the formula reduces to

$$\alpha_{\text{opt}} = \frac{2}{1 + \sin \pi/p}. \tag{8}$$

4 Computation

Write a function to solve Laplace's equation using equation 5 with the template in appendix A.1. Note that in the definition of the residual you use *new* values of ψ as soon as they have been calculated. This makes programming easier. Note also that you should iterate from the boundaries where ψ is non-zero down to the boundaries on which ψ is zero. The function should also return the historical values of *three* points (one in the upper half, one in the middle, and one in the lower half of the domain) and watch how they converge. Take a 7×7 grid initially and try a value $\alpha = 1.35$ to begin with. Check that your solution is correct by comparing it with the analytic solution. When your program is working, choose values of $\alpha = 1.1, 1.25, 1.45$ and 2.1, and compare the rates of convergence. Note also whether convergence is monotonic or oscillatory.

Set a maximum iteration number of 30, but also put in a test to stop the program if convergence is reached sooner. Print your solution and plot it as a contour diagram. You will need to write an additional script to plot the graphs and perform the analysis.

▶ You may opt at any point to discuss results or plots with a demonstrator before proceeding.

One day of extra practical credit can be earned by examining both of the following applications of your Laplace Equation solver.

Optional
1 day credit

5 Solve Poisson's equation

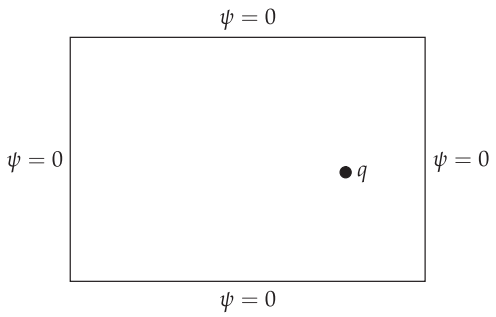


Figure 3: Schematic of a line charge at a point within an earthed container.

An example of an inhomogeneous elliptic partial differential equation is Poisson's equation:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = f(x, y) \tag{9}$$

The finite difference form of this equation is

$$\psi_{i,j} = \frac{(\psi_{i+1,j}+\psi_{i-1,j})/(\delta x)^2 + (\psi_{i,j+1}+\psi_{i,j-1})/(\delta y)^2}{2\left(\frac{1}{(\delta x)^2} + \frac{1}{(\delta y)^2}\right)} - \frac{f_{i,j}}{2\left(\frac{1}{(\delta x)^2} + \frac{1}{(\delta y)^2}\right)} \tag{10}$$

The corresponding solution is equation 5, derived from equation 4, with $\psi_{i,j}$ calculated from equation 10. You should refine your mesh to 25 points (and readjust appropriately). Complete the Poisson's equation solver function given in appendix A.2.

Poisson's equation arises in many branches of physics, but the boundary conditions are frequently different. You might be familiar with the case of line charges in an earthed container. Simulate a line charge by taking $f = q$ at the point (x_0, y_0), but zero at other points as shown in figure 3. Make different choices for the point (x_0, y_0).

Now consider a dipole, i.e. a line charge at (x_0, y_0) and a line charge of opposite sign at (x_1, y_1). Vary the distance between the line charges. You may consider other distributions of charge or other applications of Poisson's equation.

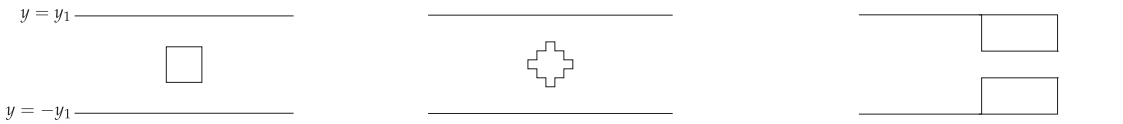
6 Perfect irrotational fluid motion

For irrotational incompressible flow of an ideal fluid (no viscosity) in two dimensions (independent of z), the stream function ψ satisfies Laplace's equation. The (u, v) components of the fluid velocity are related to the stream function by

$$u = \frac{\partial \psi}{\partial y} \qquad v = -\frac{\partial \psi}{\partial x} \tag{11}$$

For plane uniform flow between two parallel plates $\psi = U_0 y$. Change your boundary condition to $\psi = U_0 y_1$ on the upper plate and $\psi = -U_0 y_1$ on the lower plate, where the gap between the planes is $2y_1$. Check your solution.

Now introduce the rectangular obstacle shown in figure 4a and solve Laplace's equation for the case of $\psi = 0$ on this obstacle.



(a) Rectangular obstacle. (b) Circular obstacle. (c) Reduced width of outflow.

Figure 4:

Next try to simulate a circular object (really a cylinder), as shown in figure 4b. The analytic solution for this case is

$$\psi = U\left(r - \frac{a^2}{r}\right)\sin\theta \tag{12}$$

where (r, θ) are plane polar coordinates and a is the radius of the cylinder. Compare your solution with the analytic solution. You should refine your mesh to 25 points (and readjust appropriately).

Now take the object out but reduce the width of the outflow region, as in figure 4c. The value of the stream function on the boundaries does not change when the outflow narrows.

7 Final considerations

Discuss some of the following in your write-up:

The mathematical origin of the expressions for the second derivatives in approximating the Laplace/Poisson equation and the associated errors. Could alternative expressions be used for the derivatives?

The origin of equation 7, the more general issue of numerical stability when solving PDEs, and alternative methods for solving PDEs beyond Laplace's equation (e.g. the diffusion equation). You may wish to consult our local short option on numerical methods[7].

8 Preparing for assessment

Part A Computing Practicals are an exercise in computer programming as well as in scientific report writing. Your report must follow the guidance in AD34 — *the art of scientific report writing*[a]. To write your report, you have a few options; Microsoft Word or LaTeX typesetting are most common. Whatever your choice, the system you use must be able to produce text-readable PDF format (not PDF created from a scanned image on a printer).

When you have finished writing your code (and have tested it completely) and your report is complete, you must upload your work (both the code and the report) electronically to Canvas (you will find the links in the Part A Practicals Computing section). Be sure to complete these uploads in advance of meeting with a demonstrator for marking. Deadlines and Computing Demonstrator schedules are also found in Canvas.

At marking time, you should have at hand your computer with your code, a copy of your report and your logbook (where you wrote extra notes during your program development). Be prepared to describe your code and demonstrate its execution.

[a] www-teaching.physics.ox.ac.uk/practical_course/Admin/AD34.pdf

A Functions to be implemented

A.1 Laplace equation

```
function [psi, hist_values] = solve_laplace(init_psi, alpha, N_iter)
% Author: ??? , Date: ??/??/????
% This function solves the Laplace's equation using the over-relaxation method
% Input:
% * init_psi: 2D matrix containing the initial \psi, including boundaries.
% * alpha: the coefficient of over-relaxation.
% * N_iter: maximum number of iterations performed.
%
% Output:
% * psi: 2D matrix of the value of \psi after (up to) N_iter iterations.
% * hist_values: (N_iter x 3) matrix that contains historical values of 3 points during
%       the iteration (1 in the upper half, 1 in the middle, and 1 in the lower half).
%
% Constraints:
% * The boundaries of \psi are kept constant during the iterations.
%
% Example use:
% >> init_psi = zeros(7,7);
% >> % example of boundary conditions: all ones
% >> init_psi(1,:) = 1;
% >> init_psi(end,:) = 1;
% >> init_psi(:,1) = 1;
% >> init_psi(:,end) = 1;
% >> [psi, hist_vals] = solve_laplace(init_psi, 1.1, 30);
end
```

A.2 Poisson equation

```
function [psi] = solve_poisson(init_psi, fixed_psi, source)
% Author: ??? , Date: ??/??/????
% Solve Poisson's equation, i.e. del^2 \psi = source
% Input:
% * init_psi: 2D matrix containing the initial \psi values.
% * fixed_psi: 2D matrix that flags which elements in  \psi are constants
%              (i.e. 1 if it is kept constant, 0 otherwise).
% * source: 2D matrix indicating the source term of Poisson's equation.
%
% Output:
% * psi: 2D matrix of \psi after solving Poisson's equation
%
% Constraints:
% * The value of \alpha and number of iterations must be automatically adjusted inside
%       the function.
% * init_psi, fixed_psi, and source must have the same size, it is the caller's
%       responsibility to make sure they are the same size, not the function's.
%
% Example use:
% >> init_psi = zeros(9,9);
% >> source = zeros(9,9);
% >> source(5,5) = 10;
% >> % fix the boundary values
% >> fixed_psi = ones(9,9);
% >> fixed_psi(2:end-1,2:end-1) = 0; % interior not fixed
% >> psi = solve_poisson(init_psi, fixed_psi, source);
end
```

Bibliography

[1] R. W. Hornbeck, *Numerical Methods*, Prentice Hall, 1982.

[2] W. H. Press et al. *Numerical Recipes: the Art of Scientific Computing*, Cambridge University Press (2007).

[3] P. K. Kundu, *Fluid Mechanics*, Academic Press 1990.

[4] D. E. Potter, *Computational Physics*, Wiley, 1973.

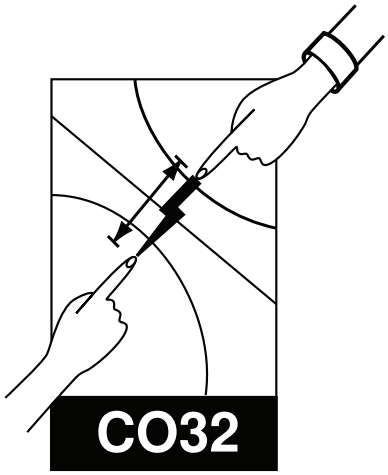
[5] R. S. Varga, *Matrix Iterative Analysis*, 2nd edition, Springer, 2000.

[6] J. Todd (editor), *Survey of Numerical Analysis*, McGraw Hill, 1962.

[7] A. O'Hare, *Numerical Methods for Physicists*, Oxford Physics, 2005[2].

[2] http://www-teaching.physics.ox.ac.uk/computing/NumericalMethods/nummethods.html

CO32: Fourier optics



revised EG, MK, NC, CH January 2022

Preface

It is common practice in modern software engineering to write programs in a modular and standardised way. To help you get started, appendix A provides you with one or more specific function headers which you **MUST** use to write the functions around which your program should be built and in order to receive a satisfactory mark. Your comments in the header of each function must include:

- author and date,
- purpose (a brief description of what the function does),
- inputs and outputs,
- and, if appropriate, any constraints or limitations of use.

Do not forget that you will also need to keep good records of your progress during this practical in your logbook. If you make plots and/or write any notes electronically, ideally, you would print them and affix them into the pages of your logbook. You should have comments in function and script headers of your code, as well as comments within your code. These aspects may all be considered at marking time.

The function header template provided is based on the default language of the Lab: MATLAB. Alternatively, you may write your program in python if you wish (check for approval by Head of Lab to use any other languages). Regardless of the chosen language, the functional equivalent of the function introduced in appendix A must be included in the code.

Assessment

Please see the Part A Guide in Canvas for Computing Lab deadlines as well as availability (schedule) of Computing Lab Demonstrators who mark your work as well as offer help and advice. Before you meet with a demonstrator for marking, **you must** upload your work into Canvas (both your code and your report). Your code must be uploaded in a single file (see instructions in Canvas) and must include any functions required by the script. Your report must be written following the requirements described in AD34 — *the art of scientific report writing*[1]) and uploaded into Canvas in valid pdf format.

1 Introduction

Real light sources are, in practice, almost never monochromatic. They are in general composed of multiple frequencies superimposed on each other. These frequencies can be separated into a Fourier Series, which splits a periodic function into a large number of sine and cosine waves with certain amplitudes and frequencies. A typical source will have primary frequencies with additional harmonics (integral multiples) of these frequencies.

As the period of the function gets larger, the separation between frequencies gets smaller, and eventually a new continuous function can be introduced to express the frequency spectrum. This is known as the *Fourier Transform* and is used to describe the diffraction patterns in optics.

In this practical you will be investigating the *Discrete Fourier Transform*, a form of the Fourier Transform that is particularly suited to computational physics, and using it to investigate various diffraction patterns [1].

[1] `www-teaching.physics.ox.ac.uk/practical_course/Admin/AD34.pdf`

2 The discrete Fourier transform

The Fourier transform of a continuous function, $f(x)$, is written as

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x u} dx \tag{1}$$

and its inverse as

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{-2\pi i x u} du \tag{2}$$

The (1D) Fourier transform can be approximated as a sum over discrete values

$$F(u) = \sum_{x=-N}^{N-1} \left(f(x) e^{-\frac{\pi i x u}{N}} \right). \tag{3}$$

This sum is a form of what is known as the Discrete Fourier Transform (DFT). If we write the function $f(x)$ in complex form $f(x) = \mathcal{R}f(x) + i\mathcal{I}f(x)$, we can write an expression for the complex transform (remembering that $e^{i\phi} = cos\phi + isin\phi$).

$$\mathcal{R}F(u) = \sum_{x=-N}^{N-1} \left[\mathcal{R}f(x) \cos\left(-\frac{\pi x u}{N}\right) - \mathcal{I}f(x) \sin\left(-\frac{\pi x u}{N}\right) \right] \tag{4}$$

$$\mathcal{I}F(u) = \sum_{x=-N}^{N-1} \left[\mathcal{R}f(x) \sin\left(-\frac{\pi x u}{N}\right) + \mathcal{I}f(x) \cos\left(-\frac{\pi x u}{N}\right) \right] \tag{5}$$

3 Calculations

3.1 The DFT

Write the Fourier transformation function for a one-dimensional signal based on the template given in appendix A.1 and write a script that calls the function to tackle both problem A and problem B.

For these problems, set N to 250. Note, however, that the array you will need to store will have 500 elements. Since the index of a MATLAB array always starts from one, you will have to transform your x (or u) variable such that the value of a function at $x = -250$ will be stored in the first element of the array.

Problem A

Create a single slit centred on the origin (the centre of your array) with height 1 and width 10 (i.e. it is made up of $10 + 1$ elements of your array). Set the array for the imaginary parts to be zero and the array containing the real parts to be 1 for the 11 elements defining the slit, and zero otherwise.

Calculate the DFT of this single slit function and plot the both the real component of the amplitude and the total amplitude of the transform. Don't forget to rearrange the array(s) containing the transform.

Move the slit so that it is centred at -10, with a height of 1. Plot the real and total amplitude of the transform and compare it to the single slit centred on the origin. Discuss the differences in your write-up.

For the single slit centred on the origin, double the width of the slit while keeping the height at 1. Plot the amplitude of the transform and comment on the effect.

Finally, halve the height of the last slit.

Problem B

Create 2 slits (a double slit), one centred at -15 and the other at $+15$ each of height 1 and width 20 (i.e. each made up of $20+1$ elements of your array). Set the array for the imaginary parts to be zero and the array containing the real parts to be 1 for the 42 elements defining the two slits, and zero otherwise. Plot the amplitude of this transform. Remember to rearrange the array(s) containing your transforms.

Now move the slits to be at ± 25. Compare the amplitude of the transform to the one centred at ± 15.

Double the width of the previous slits and plot the amplitude of the transform.

Halve the height of the slits and plot the amplitude of the transform.

▶ Compare and explain the differences between the transforms of the 4 double slits you have plotted.

3.2 Convolution

The convolution of two functions $f(x)$ and $g(x)$ is given by the following integral

$$h(X) = f(x) * g(x) = \int_{-\infty}^{\infty} f(x)g(X-x)\,\mathrm{d}x \tag{6}$$

If we denote the Fourier transforms of $f(x)$, $g(x)$ and $h(x)$ by $F(u)$, $G(u)$, and $H(u)$, respectively, and if $h(x) = f(x) * g(x)$, the convolution theorem states that

$$\begin{aligned} \mathrm{FT}\{f * g\} &= \mathrm{FT}\{f\} \times \mathrm{FT}\{g\} &(7)\\ H(u) &= F(u) \times G(u) &(8) \end{aligned}$$

i.e. the convolution of two functions in real space is equivalent to the product of their Fourier transforms in frequency space.

Write a convolution function using the template given in appendix A.2.

Convolve a single slit (centred on the origin, width 20 and height 1) with itself.

Calculate the transform of the single slit and the convolution. Square the Fourier transform and compare it to the transform of the convolution to verify the convolution theorem.

▶ You may opt at any point to discuss results or plots with a demonstrator before proceeding.

Optional
1 day credit

4 The two-dimensional DFT

In two dimensions the discrete fourier transform can be calculated using a $2N \times 2M$ grid:

$$F(u,v) = \frac{1}{4NM} \sum_{x=-N}^{N-1} \sum_{y=-M}^{M-1} f(x,y) e^{-\pi i(xu/N + yv/M)}. \tag{9}$$

Write the Fourier transformation function for a two-dimensional signal based on the template given in appendix A.3 and write a script that calls the function to tackle any *two* of the following problems.

Problem A

Create a single source in the centre of a square (50×50) lattice. For simplicity create a cross shaped source i.e. $[25][25] = [25][26] = [25][24] = [24][25] = [26][25] = 1$. You can change the strength of the source if you like.

Plot both the real part of the source and the amplitude of the Fourier transform.

Problem B

Create 2 single (2×2 square) sources in a square (50×50) lattice. You can place the sources anywhere in the lattice you wish and set the source strength to 1.

Plot both the real part of the source and the amplitude of the Fourier transform.

Problem C

Create a single slit in a square (50×50) lattice. The slit should have a width of 3 and a length of 20 and should be centred at the centre of the lattice. Set the source strength to 1.

Plot both the real part of the source and the amplitude of the Fourier transform.

Problem D

Create a double slit in a square (50×50) lattice. Each slit should have a width of 3 and a length of 20 and should be at a distance ± 10 from the centre of the lattice. Set the source strength to 1.

Plot both the real part of the source and the amplitude of the Fourier transform.

5 Preparing for assessment

Part A Computing Practicals are an exercise in computer programming as well as in scientific report writing. Your report must follow the guidance in AD34 — *the art of scientific report writing*[a]. To write your report, you have a few options; Microsoft Word or LaTeX typesetting are most common. Whatever your choice, the system you use must be able to produce text-readable PDF format (not PDF created from a scanned image on a printer).

When you have finished writing your code (and have tested it completely) and your report is complete, you must upload your work (both the code and the report) electronically to Canvas (you will find the links in the Part A Practicals Computing section). Be sure to complete these uploads in advance of meeting with a demonstrator for marking. Deadlines and Computing Demonstrator schedules are also found in Canvas.

At marking time, you should have at hand your computer with your code, a copy of your report and your logbook (where you wrote extra notes during your program development). Be prepared to describe your code and demonstrate its execution.

[a] www-teaching.physics.ox.ac.uk/practical_course/Admin/AD34.pdf

A Functions to be implemented

A.1 Fourier transform 1D

```
function [Y] = ft1(X)
% Author: ??? , Date: ??/??/????
% Fourier transform function for a 1-dimensional signal.
% Input:
% * X: a column vector of signal with N elements (N x 1).
%
% Output:
% * Y: the Fourier transformed signal in the form of a column vector of N complex
%      elements.
%
% Constraints:
% * X is a one-dimensional column vector
% * Do not use the built-in Fourier transform functions from MATLAB
%
% Example use:
% >> t = linspace(-1, 1, 1000);
% >> x = exp(-t.*t/(2*0.2^2));
% >> y = ft1(x);
% >> plot(abs(y));
end
```

A.2 Convolution 1D

```
function [w] = convolution1(u, v)
% Author: ??? , Date: ??/??/????
% Perform convolution of two 1-dimensional signals, u and v
% Input:
% * u, v: column vectors of signal with N elements each (N x 1).
%
% Output:
% * w: the convolved signal of u and v with the same size as the input vector, N x 1
%      (the full vector would have size 2N-1 but here we only take the central part of
%      the signal, [ceil((N+1)/2):ceil((3*N-1)/2)] ).
%
% Constraints:
% * u and v are one-dimensional vectors of the same size.
% * Do not use the built-in functions from MATLAB that perform convolution.
%
% Example use:
% >> t = linspace(-1, 1, 1000);
% >> x = exp(-t.*t/(2*0.2^2));
% >> y = ft1(x);
% >> plot(abs(y));
end
```

A.3 Fourier transform 2D

```
function [Y] = ft2(X)
% Author: ??? , Date: ??/??/????
% Fourier transform function for 2-dimensional signal.
% Input:
% * X: a matrix of 2D signal with size (2M x 2N).
%
% Output:
% * Y: the Fourier transformed signal in the form of a (2M x 2N) matrix with complex
%      elements.
%
% Constraints:
% * X is a two-dimensional matrix.
% * Do not use the built-in Fourier transform functions from MATLAB.
%
% Example use:
% >> [t0, t1] = meshgrid(linspace(-1, 1, 1000));
% >> x = exp(-(t0.*t0+t1.*t1)/(2*0.2^2));
% >> y = ft2(x);
% >> imagesc(abs(y));
end
```

Bibliography

[1] E. Hecht, *Optics*, 4th edition, Addison Wesley