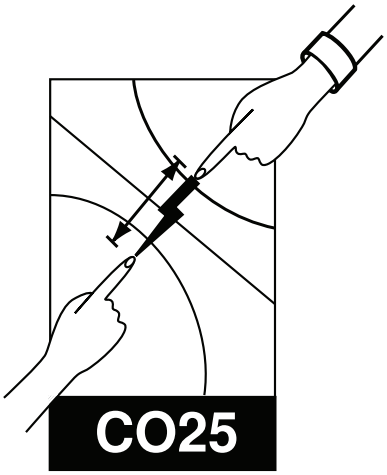
Laplace's equation



revised EG, MK January 2018

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, you should 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 template provided is for the default language of the Lab: MATLAB. If you have checked with a demonstrator about using another language and they have authorised you to do so, you must use the functional equivalent of the appendix A function headers in that language.

Assessment

You must upload your work electronically (both your code and your report (AD34 — *the art of scientific report writing*) via WebLearn as described in the Part A Handbook **in the week BEFORE** you come in to meet with a demonstrator for marking. This will be strictly enforced in the final weeks (after week 6), while in earlier weeks, demonstrators may use their discretion (i.e. allow marking for electronic submissions in the same week). Demonstrators are available for Part A Computing marking (and advice!) during Hilary Term weeks 1–8 on Mondays and Tuesdays from 10:00–13:00 and 14:00–17:00.

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

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

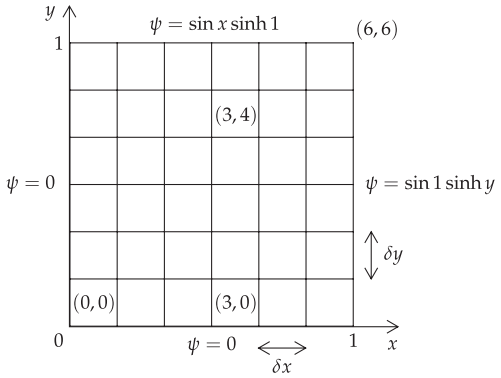
in a rectangular domain with ψ set to some prescribed function along the boundary of the domain.

3 Numerical approach

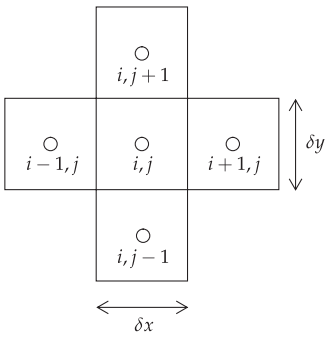
An approximation for Laplace's equation is

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

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 $R^m_{i,j}$.

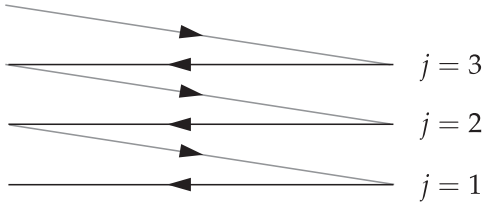


Figure 2: Sweeping across the grid systematically.

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

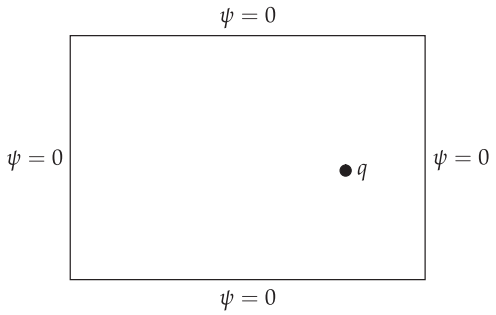


Figure 3: Schematic of a line charge at a point within an earthed container.

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.

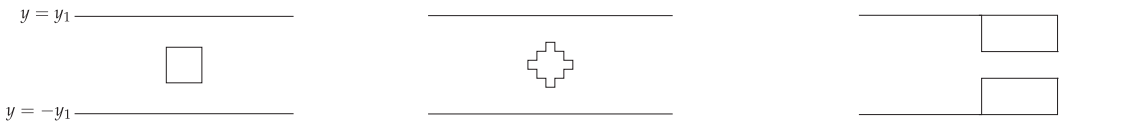
6 Perfect irrotational fluid motion

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 be written in the style described in AD34 — *the art of scientific report writing*. To write your report, you have many 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 assembling 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 WebLearn. The upload must be **in the week before** you meet with a demonstrator for marking. The upload procedure is described in the Part A Handbook.

Demonstrators are available for Part A Computing marking (and advice!) during Hilary Term weeks 1–8 on Mondays and Tuesdays from 10:00–13:00 and 14:00–17:00.

In the week after you upload your work to WebLearn, come into the Computing Lab to meet with a demonstrator for marking. **At marking time, be prepared** with a printed copy of your report and your logbook (where you wrote extra notes during your program development), and be prepared for the demonstrator to download your code from WebLearn for you to describe it and demonstrate its execution.

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
```

References

[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[1].

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

