7 Mathematical Methods

7.6 Insulation (10 units)

There are no prerequisites for this project.

1 Introduction

When sheets of plastic and of other insulating materials are used in the construction of building walls and ceilings, a balance is sought between the need to minimise the loss of heat and the need to include (non-insulating) holes to allow gases, especially water vapour, to pass. In this project we will model a two dimensional analogue of this problem. A perfectly insulating sheet containing equally spaced holes is placed between two constant temperature surfaces and the resulting steady-state temperature distribution is computed. The steady-state temperature distribution is computed by solving Laplace's equation using a so-called relaxation method.

We start from the non-steady-state (Poisson's) equation appropriate to heat conduction,

$$\kappa \nabla^2 T = \rho s \frac{\partial T}{\partial t} \ , \tag{1}$$

where T is temperature, t is time, κ is the thermal conductivity of the medium, ρ is its density and s is its specific heat. We then impose the steady-state condition $\partial T/\partial t = 0$, to obtain

$$\nabla^2 T = 0 \ , \tag{2}$$

that is, Laplace's equation.

2 Empty unit square

In this section we will consider a unit square or "box" $0 \leqslant x \leqslant 1$ and $0 \leqslant y \leqslant 1$ in which the left side $x = 0$ is held at constant temperature 0, the right side $x = 1$ at constant temperature 1, and in which the top and bottom boundaries $y = 0$, 1 are perfect insulators, so that there is no temperature flux across the top and bottom. Thus the boundary conditions are

$$\begin{aligned} T(0, y) &= 0 \\ T(1, y) &= 1 \\ \frac{\partial T}{\partial y}(x, 0) = \frac{\partial T}{\partial y}(x, 1) &= 0 \end{aligned}$$

The analytic solution to this problem is easy to find; this therefore allows us to check the accuracy of our numerical method and confirm that it is working correctly.

Consider the following $N_x \times N_y$ discretisation of the unit square:

$$\begin{aligned} x_i &= i\Delta x, \qquad i = 0, 1, \ldots, N_x, \qquad \Delta x = \frac{1}{N_x} \ , \\ y_j &= j\Delta y, \qquad j = 0, 1, \ldots, N_y, \qquad \Delta y = \frac{1}{N_y} \ . \end{aligned}$$

Let us adopt the notation $T_{i,j} = T(x_i, y_j)$ for the numerical solution to (2). We approximate $\partial^2 T/\partial x^2$ and $\partial^2 T/\partial y^2$ to second order at a general internal (i.e., away from the boundary) point (x_i, y_j) with

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} \ ,$$
$$\frac{\partial^2 T}{\partial y^2} = \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} \ .$$

By choosing the special case $\Delta x = \Delta y = \Delta$, which implies $N_x = N_y$, Laplace's equation can therefore be written in discretised form to second order as

$$0 = -4T_{i,j} + T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} \ . \tag{3}$$

In this project (3) is solved using a relaxation method. So-called multigrid methods are faster, but their extra complications are not necessary here (see Press et al., 1992 for details). The relaxation method iteration is started by choosing *any* initial values for the $T_{i,j}$; initial values that are close to the solution to (3) will converge quicker than initial values that are very different from the solution. Each subsequent iteration step consists of computing new values for $T_{i,j}$ successively for each internal point using the relaxation algorithm:

$$T_{i,j}^{\text{new}} = (1-\sigma)T_{i,j}^{\text{old}} + \frac{\sigma}{4}\left(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}\right) \ , \tag{4}$$

where σ is a pre-determined constant (see below).

New values for $T_{i,j}$ on the boundary are then computed by applying the appropriate boundary conditions. Applying the first two boundary conditions is simply a matter of not changing the boundary points $T_{0,j}$ and $T_{N_x,j}$. The third boundary condition can be obtained by using a central difference approximation to $\partial T/\partial y$ at a general boundary point (x_i, y_j):

$$\frac{\partial T}{\partial y} = \frac{T_{i,j+1} - T_{i,j-1}}{2\Delta} = 0 \ . \tag{5}$$

This approximation is used with (4) to compute a new value for $T_{i,j}$ on the boundary in terms of points that are either on the boundary or internal. For example the bottom $j = 0$ boundary points are computed using

$$T_{i,0}^{\text{new}} = (1-\sigma)T_{i,0}^{\text{old}} + \frac{\sigma}{4}\left(T_{i+1,0} + T_{i-1,0} + 2T_{i,1}\right) \ , \tag{6}$$

and an analogous formula is used for the top $j = N_y$ boundary points.

Further iterations are carried out until the $T_{i,j}$ have converged at which time they are a solution to (3). The special case $\sigma = 1$, obtained by rearrangement of (3), is called the Jacobi relaxation method. However, in practice, convergence can be considerably faster if an appropriate value $\sigma > 1$ is chosen (called over-relaxation). For all the cases studied in this project, $\sigma \approx 1.9$ is a good first choice.

> **Programming Task:** Write a program to solve (2) for the unit square $0 \leqslant x \leqslant 1$ and $0 \leqslant y \leqslant 1$ with the given boundary conditions. You should use MATLAB's 64-bit (8-byte) double-precision floating-point values or the equivalent in other programming languages. Your program should plot contours of T and should include some mechanism for deciding whether the T array has converged. You should choose $N_x = N_y$ that is large enough for the contours of T to be well resolved but small enough that the T array converges in a reasonable time.

Note that equation (4) does not specify whether some of the values of T on the right-hand side are the "old" or the "new" values. Either choice will work, but it is most efficient to "sweep" through the matrix updating each value of $T_{i,j}$ immediately as you go along; so in fact, some of the T-values on the right-hand side will be "old" (because they haven't yet been updated) and some will be "new" (because the sweeping process has already updated them). This improvement to the Jacobi method, which surprisingly increases its rate of convergence, is called the Gauss–Seidel method.

Question 1 Describe how you decided whether the T array had converged. Experiment with values of σ and comment on the effect of changing σ on the number of iterations required for convergence. Include in your write-up a contour plot of the steady-state temperature distribution of $T(x, y)$.

3 Unit square and vertical insulating wall with holes

In this section we place a vertical insulating "wall" across the unit square of section §2 with holes of size ϵ spaced a distance δ apart. Figure 1 shows a schematic example for the case $\epsilon = 2\Delta$, $\delta = 5\Delta$. The top of the wall is shown expanded to show more clearly new types of boundary gridpoint labelled A–H. New boundary conditions (similar to (6)) will be needed for these types of gridpoint and for I and J at the bottom of the wall. At corners of the wall (e.g., gridpoint F) you should assume that the corner is smoothed off (rounded) so that the normal is diagonal to the grid there.

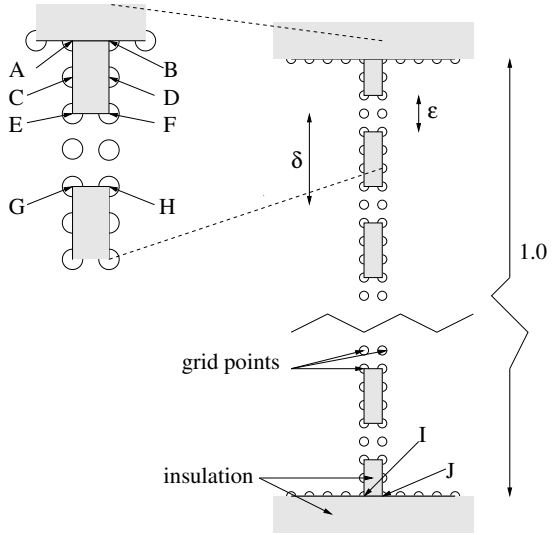

Figure 1: Insulating wall with holes

Programming Task: Modify the program of section §2 to include a vertical insulating wall at the middle of the box (or as close as possible) with holes of size ϵ, spaced δ apart. Your program should arrange the holes and the insulator pieces as symmetrically as possible about the centre of the box so that there is a length of insulator at the top and bottom of the wall as in Fig. 1. Explain *briefly* how your program assigns the locations of the insulating sections and how it deals with special cases. Note that it is not always possible to obtain perfect symmetry.

Question 2 Write down and explain the formulae you used to compute $T_{i,j}^{\text{new}}$ at the new types of boundary grid point A–J. Include in your write-up a contour plot of the steady-state temperature distribution of $T(x, y)$ for the case $\epsilon = 4\Delta$, $\delta = 8\Delta$, $N_x = N_y = 129$. Why is 129 a better choice than 128? What effect does putting holes in the wall have on the temperature distribution in the unit square? Examine what happens to temperature along horizontal cross-sections through the temperature distribution by plotting $T(x, y_0)$ against x for a few values of y_0. Choose values of y_0 that are near the centre of the box and make sure you include cases that go through the centre of a hole and through the the centre of an insulating section.

The total heat flux across the boundary $x = 1$ is a gauge of the quality of the insulating layer. Given that the heat flux at any point (x, y) is given by $-\kappa \nabla T(x, y)$, where κ is the thermal conductivity from equation (1), define a suitable measure Q of the insulator's quality. How would Q differ for a good insulator versus a good conductor? Comment on the insulating properties of the wall with $\epsilon = 4\Delta$, $\delta = 8\Delta$.

Question 3 Investigate what happens to the insulator quality Q when you vary ϵ and δ (but keep $N_x = N_y$ constant) according to

$$\epsilon = k\delta^{\alpha} \ , \tag{7}$$

where k and α are suitable real constants. In our discrete model, since ϵ and δ must be multiples of Δ, you would need to choose the nearest integer multiple of Δ for ϵ for a given δ or vice-versa. Include in your write-up a few plots that illustrate what happens to Q as you decrease δ: choose relationships that show interesting behaviour. Comment on your plots and on the physical significance of (7).

4 Infinite length vertical insulating wall with holes

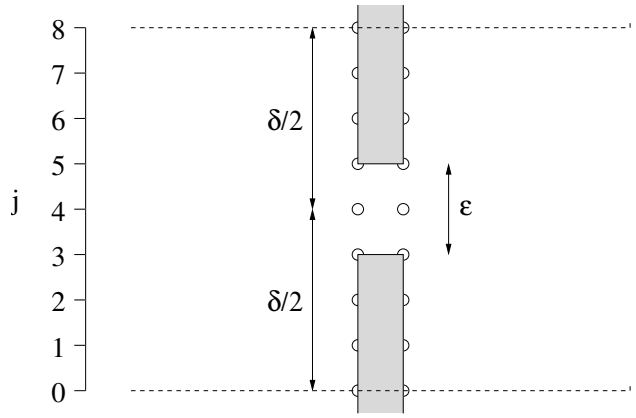

Figure 2: Insulating wall with holes, periodic boundary

In this section we simulate an infinitely long vertical insulating wall with holes by taking a single, finite section of the wall and imposing a periodic boundary condition at the top and bottom of the region. A periodic boundary condition means $T(x, y_{\text{bot}}) = T(x, y_{\text{top}})$. With this boundary condition only one central hole is needed. Figure 2 shows a schematic example for the case $\epsilon = 2\Delta$, $\delta = 8\Delta$ so that $T_{i,8} = T_{i,0}$. Thus for the case shown in Fig. 2, the $j = 0$

(boundary) points are computed using (4) but with the periodic boundary taken into account by replacing j with j modulo 8, thus

$$T_{i,0}^{\text{new}} = (1-\sigma)T_{i,0}^{\text{old}} + \frac{\sigma}{4}\left(T_{i+1,0} + T_{i-1,0} + T_{i,1} + T_{i,7}\right) \ . \tag{8}$$

Similarly, the $j = 7$ points are computed using

$$T_{i,7}^{\text{new}} = (1-\sigma)T_{i,7}^{\text{old}} + \frac{\sigma}{4}\left(T_{i+1,7} + T_{i-1,7} + T_{i,0} + T_{i,6}\right) \ . \tag{9}$$

Programming Task: Write a program to implement a relaxation method algorithm for the situation illustrated in Fig. 2 but for general N_x, δ and ϵ.

Question 4 Investigate what happens to Q when you vary ϵ and δ (but keep N_x constant) in this new model. Include in your write-up a couple of illustrative plots. Is there any need to change your definition of the quality Q? How does this periodic boundary condition model relate to the model in Question 3?

Question 5 Adapt the infinite length vertical insulating wall with holes program for the case $\delta = 8\epsilon$ to investigate what happens to the steady-state temperature distribution near the hole(s) as the wall thicknes is varied. Try values of wall thickness in the range $\frac{\epsilon}{4}$ to 4ϵ. To do this you will need to use as large a number of gridpoints as is consistent with a not unreasonable run time of your program.

Note that you may find it helpful to start with the smallest wall thickness, save the gridpoint temperature distribution, use it to construct a first approxination for the next wall thickness and repeat the process.

Question 6 The original task was to investigate, for insulating sheets, the balance between the need to minimise the loss of heat and the need to include (non-insulating) holes to allow gases, especially water vapour, to pass. In the light of what you have learned from this model, what advice would you give a manufacturer of insulating sheets? What are the limitations of the model and what steps could be taken towards greater realism?

Reference

Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., 2002: *Numerical recipes in C: The art of Scientific computing*, Cambridge University Press.

