Exercise 1: Matrices and Linear Algebra

Drew Silcock
Physics Department, University of Bristol
(Dated: March 8, 2014)

I. RELAXATION METHODS

A common method of solving partial differential equations, both linear and nonlinear, is by iterative relaxation methods. Here this means taking advantage of the finite difference discretisation of the differential equation. The essential method has six steps:

Step 1: Define a regular spatial grid covering the region of interest including nodes at the boundaries.

Step 2: Set the boundary conditions of the problem by fixing the boundary nodes to the boundary values.

Step 3: Guess an initial state for the interior of the grid.

Step 4: Calculate the finite difference equations.

Step 5: Choose a convergence criterion.

Step 6: Iterate the equations at each boundary node until the convergence condition is satisfied.

As applied to the Poisson equation

$$\nabla^2\varphi = -\rho, \tag{1}$$

this means that, given φ is a smooth function $\varphi : \mathbb{R} \to \mathbb{R}$ and thus has Taylor expansion about point x_i of

$$\varphi(x) = \varphi(x_i) + (x - x_i)d_x\varphi(x_i) + \frac{(x - x_i)^2}{2}d_x^2\varphi(x_i) + \mathcal{O}((x - x_i)^3),$$

the second derivative can be approximated by

$$d_x^2\varphi(x) = \frac{\varphi(x - h) + \varphi(x + h) - 2\varphi(x)}{h^2} + \mathcal{O}(h^2) \tag{2}$$

where h is the grid spacing.

Equation 2 gives the finite difference for the second differential.

Generalising this to two dimensions and applying to Equation 1 gives the following iterative formula at each node:

$$\varphi(x_i, y_i) = \frac{1}{4}\Big[\varphi(x_{i-1}, y_j) + \varphi(x_{i+1}, y_j) + \varphi(x_i, y_{i-1}) + \varphi(x_i, y_{i+1}) + \rho(x_i, y_j)h^2\Big] \tag{3}$$

where x_i, y_j are points on the grid. If $\rho = 0$, i.e. there are no sources and Poisson's equation becomes Laplace's equation, then this iteration simply becomes the average value of the nearest neighbour points on the grid.

There are two methods of solving this equation.

A. Jacobi Method

There are at any time two distinct grids of nodes. When the finite difference equation is used to iterate a node, producing the new value, this value is stored in the new grid so that all iterations are acted upon by the old grid and produce values which go into the new grid. This means that all iterations are based entirely on the values from the original grid.

B. Gauss-Seidel Method

Another possible iteration method is known as the Gauss-Seidel method. In this method, the value produced by iterating at a node using the same finite difference equation is stored in the original grid. This means that all iterations are based partly on values from the original grid and partly on new values obtained from already completed iterations. Thus the Gauss-Seidel would be expected to converge to the final solution faster than the Jacobi method, as is shown in Section II.

II. SOLVING LAPLACE'S EQUATION IN TWO DIMENSIONS

Here Laplace's equation in two dimensions,

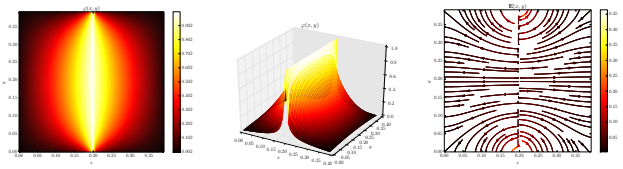
$$\nabla^2\varphi = 0, \tag{4}$$

is solved using the finite difference equation given in Equation 3 with $\rho = 0$. The chosen convergence condition is that the maximum change in any node is less than an absolute error tolerance, called ϵ. By default ϵ is set to 10^{-5}, but choosing a different scale for the charge density requires altering this to be appropriate, e.g. if the charges involved are $\mathcal{O}(10^{-6})$, the absolute error tolerance should be $\sim 10^{-11}$.All non-boundary nodes were set to random guesses between 0 and 1. The number of iterations was capped at `max_it`= 10000.

I might chang this.

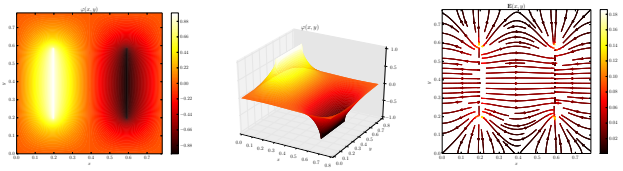
A. Basic Verification

The program has in-built boundary conditions for a parallel plate capacitor, a plane equipotential, a single point charge-like equipotential in the middle of the grid, a net with constant potential around the edges of the grid and a cross equipotential. The resulting solutions are plotted in Figures **??**.



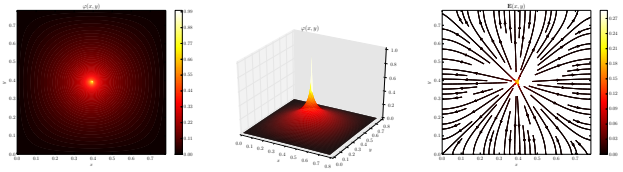
(a) Contour plot (b) Surface plot (c) Vector **E** field plot

FIG. 1: Plots of the solution to the Laplace equation with boundary conditions of an equipotential plane.



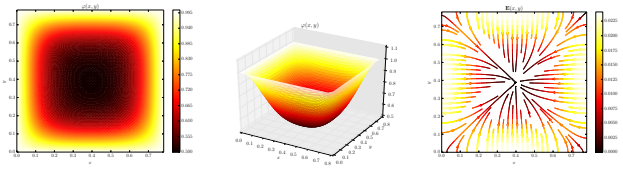
(a) Contour plot (b) Surface plot (c) Vector **E** field plot

FIG. 2: Plots of the solution to the Laplace equation for a parallel plate capacitor.



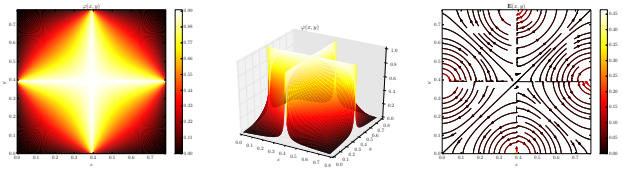
(a) Contour plot (b) Surface plot (c) Vector **E** field plot

FIG. 3: Plots of the solution to the Laplace equation for a single constant potential point.



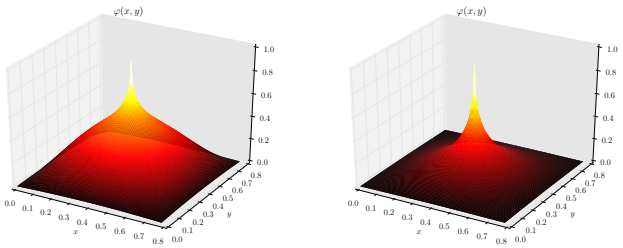
(a) Contour plot (b) Surface plot (c) Vector **E** field plot

FIG. 4: Solution to the Laplace equation with edges of the grid held at constant potential.



(a) Contour plot (b) Surface plot (c) Vector **E** field plot

FIG. 5: Solution to the Laplace equation with equipotential cross overlaid on grid.



(a) $\epsilon = 10^{-2}$ (b) $\epsilon = 10^{-6}$

FIG. 6: Comparison of the effect of altering the absolute error tolerance, ϵ, in the convergence condition on the resulting solution.

B. Convergence Condition

An absolute convergence condition is implemented to decide when the program has successfully iterated to the solution of the Laplace equation with the set boundary conditions. As compared to a relative convergence conditions, e.g. that no value changes by more than $X\%$, this has the disadvantage of depending on the boundary conditions applied to the system. So an absolute error tolerance of $\epsilon = 0.01\,\mathrm{V}$ may be appropriate if the highest boundary condition is $\mathcal{O}(100)\mathrm{V}$, but inappropriate if the potential boundary condition is $\mathcal{O}(0.1)\mathrm{V}$. This is compensated for by the ability to manually change the error tolerance with the `--error` flag. By default this value is set at 10^{-4}. When the boundary conditions are a single constant potential of $1\,\mathrm{V}$ at the centre of the grid, the effect of a relatively large tolerance is to fatten the potential field around the point in the centre, as shown in Figure 6. This is because a too high ϵ means the iterations stop before a true solution is found. As there are no charges involved, the effect of each iteration is merely to average from the surrounding nodes. Hence when the program finishes the iterations early, it means it has prematurely ceased this averaging process and thus is higher near the central peak than it should physically be.

C. Iteration Method

Both the Jacobi and Gauss-Seidel iteration methods are implemented, with the option of which to use left up to the user with the positional argument `method` determining which to use. It is noticed that the Gauss-Seidel method is significantly faster than the Jacobi, particularly when computing the solution at high grid densities.

D. Grid Density

The default parameters of the grid density are a 50 by 50 grid. Changing the density significantly affects the time taken to find a converging solution. For the

Analy
of
how
chang
ing
the
con-
ver-
gence
con-
di-
tion
chang
the
re-
sults
out
at
the
end,
mayb
with
par-
tic-
ular
refer-
ence
to

Jacobi method, the time taken to find the solution to the Laplace equation with the boundary conditions of a parallel plate capacitor...Combining high density with very low tolerance results in extremely long convergence times, as shown by .

Detailed time analysis, possibly with table.

E. Grid Edges

III. CALCULATING POTENTIAL AND ELECTRIC FIELD OF PARALLEL PLATE CAPACITOR

Detailed analysis of what happens when you've got both high grid density (100x100) and low tolerance (1e-6).

IV. SOLVING THE DIFFUSION EQUATION FOR IRON POKER