Nick Yallop – ny16281

Exercise 2:

**Partial Differential Equations**

Report:

*Problem 1*

*Finite Difference Method*

Taking the Taylor expansion of a function V(x) around a chosen point xi and revaluating this expansion at V(xi+h)=V(xi+1) and V(xi-h)=V(xi-1) allows one to add them to isolate the d2V(x)/dx2 (and higher, even order) and V(xi) terms and rearrange them (and truncate at d3V(x)/dx3 using the (δx)4≈0 assumption) to provide a quantised estimate to the continuous function d2V(x)/dx2 . Linearly combining the second differentials of V for all axis (x,y,z,…) provides the ‘standard finite difference method’ for the Laplacian operator. Combining that with Poisson’s Equation; one can model the distribution of laplacian of the potential in the EM field around a charge density distribution, and with boundary conditions one can establish a complete V distribution.

To initially simplify the problem one can take = 0 everywhere and model Laplace’s equation given a set of boundary conditions. i.e. there is one and only one distribution of V that satisfies Poisson’s (or in this case Laplace’s) equation everywhere and agrees with a complete set of boundary conditions (V(x,y) for all (x,y) on the boundary).

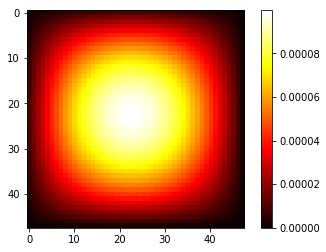
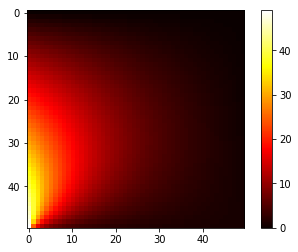
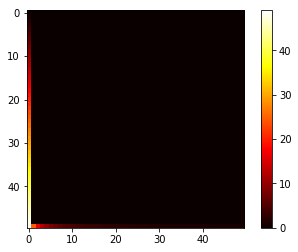
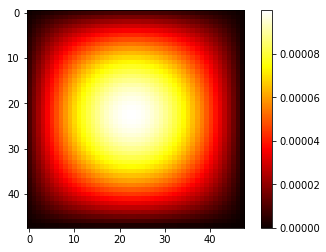
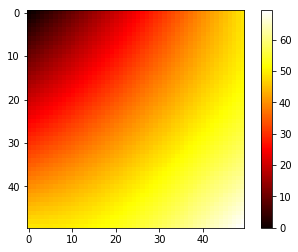
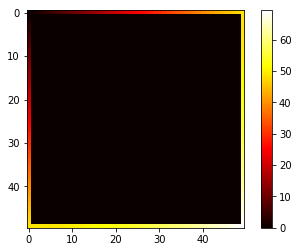
For coding, the sample space (taken to be a square for convenience) was broken into squares of equal area and the derived 2D finite difference equation;

was used. Our desired solution has everywhere so the above equation simplifies to:

i.e. the average of adjacent nodes. Thus, by fixing the boundary nodes and repeatedly replacing the current set of nodes by the set of nodes defined by the equation above, one tends towards a full solution to the Laplace equation. The chosen termination criteria was that the quantised Laplacian was less then 0.001 on all nodes.

A simple test was V(boundary) = 0 and 1 everywhere else. The completed code provided a matrix of zeros – the known solution.

Below are a set of initial conditions, followed by their final solution, followed by their error matrices. Each matrix is 50x50 (48x48 for errors on the Laplacian), and the finite difference method was used to calculate the Laplacian (and thus error) on each node as not only does (δx)4≈0, but is as close to constant as possible, so each dnV/d**e**­nare as close to 0 as possible for all n>3 and **e**ϵ{x,y}. The boundary conditions were continuous functions evaluated at quantised boundary locations to avoid non-physical discontinuities. The chosen spacing for this sampling was δx = 0.1.



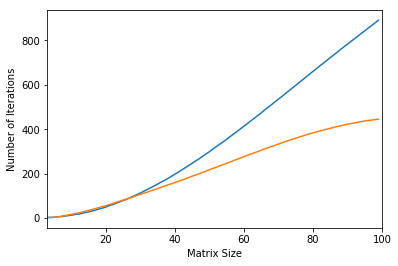
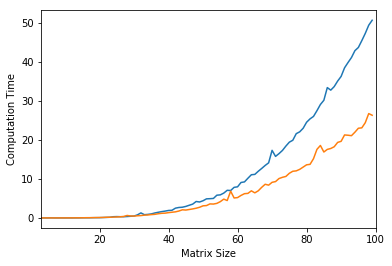
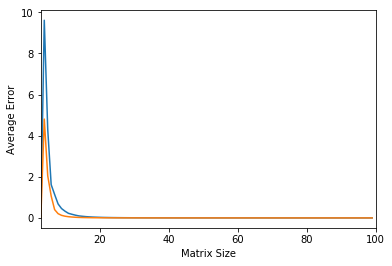
The first set represent the boundary equation [ ] and took 2564 iterations, while the second set represents the boundary equation [ ] and took 2773 iterations. The above data shows that the finite difference method is a sufficient method for problems governed by Laplace’s equation where complete, explicit boundary conditions are available. It also shows how the error (even if the chosen method for evaluating it is itself quantised/contains errors - the error is only dependant and adjacent and local potential values, so they don’t compound) on the Laplacian of each node increases exponentially with distance from the boundary. These solutions are further proof of the accuracy of the method – these at least look like the expected distributions.

Below are graphs for the computation time, iteration number and average error for the renormalized boundary distributions;

Blue = [ ]

Orange = [ ]

in the range 2< dim <101. The renormalisation of the boundary conditions shouldn’t have any effect on the computation time or relative distribution but was included to *ensure* results were comparable. The chosen spacing for this sampling was δx = 0.1.



The number of iterations required increases exponentially up until about 30 where, for the orange curve, the centre of the matrix is uniform so there are already sufficient iterations to solve the centre values, while for the blue curve the final distribution doesn’t have any large (semi-)constant areas and the whole distribution has a considerably non-uniform gradient so it tends towards linear for a time, followed by similar flattening behaviour towards the end of the graph (thus the linear area is likely just these two factors equating). This flattening stage is more postponed the less uniform the final distribution is. A charge distribution will also have a large, similar effect.

Even though the iteration number flattens out for the orange curve, both computation time curves show exponential behaviour over the whole range. This is due to the larger matrices requiring summation over more nodes, even though the computation of any given node may be quicker - .

The chosen average error was the mean and was calculated by summing over all terms in the error matrix and divided by (dim-2)2. From dim≈10 onwards the mean error remained below 0.001, thanks to the termination criteria. Before then the matrix is too small to achieve the criterion *and* the boundary conditions so the error (mean Laplacian) is abnormally high.

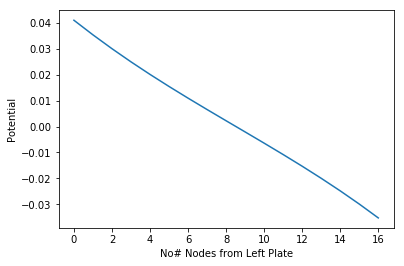
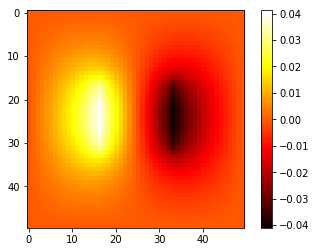
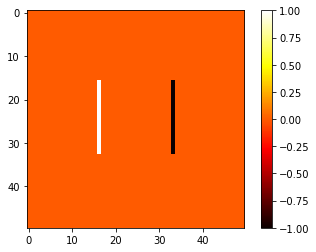
*Physical Problem*

For the parallel plate problem the same finite difference equation, but with ρ(x,y)≠0 was used. Here dnV/d**e**­n is not near zero so the Laplacian becomes less accurate in judging the error at over a node as it assumes the same truncation as the original method, so it will itself have a computational error. However, everywhere, so this was then used as a measure of the error (though the dnV/d**e**­n≠0 problem persists).

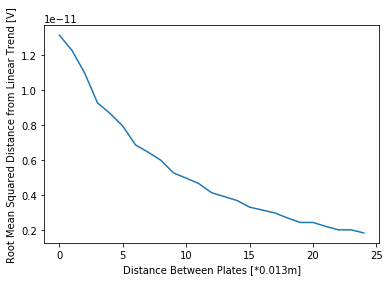
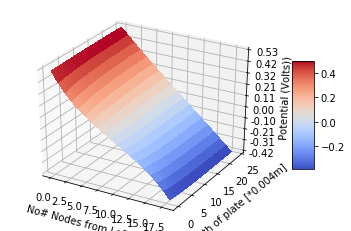
Each plate was modelled as a single line of charge density with charge density 1/(length of plate) and the equation;

was used to iterate each node (derived from the Taylor expansion of ). The 1D restriction for the plates are met in the limit that their length tends to infinity, which is in turn met when their areas tends to infinity (this is also required for the symmetry of the plate along its length, allowing one to model it as a 2D shape on these matrices, which becomes our 1D line). All of these factors mean even with a perfect distribution solution, V(between plates) still would not display perfect linear behaviour.

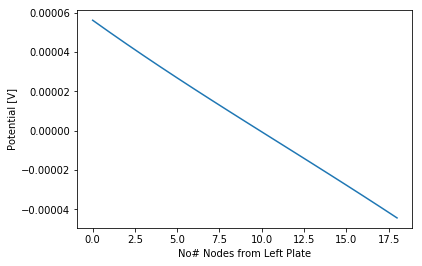
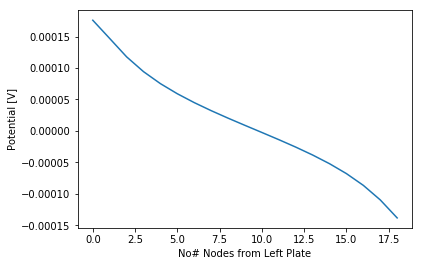
An example charge density, followed by its potential distribution, followed by the potential between the centres of the plates can be seen below (dim = 50).



To test if the model produces the desired results the solver was used on iterations of the initial conditions between short, almost point-like plates and plates that take up 90% of the length of the matrix (for the results below the matrix was taken to be 1m x 1m at 50x50 nodes). The potential between the centres of the plates against the plate lengths can be found below, followed by the sum of d2V/dx2 between the plates against plate length.



Both of these graphics show how the potential tends towards a linear trend, which is to be expected if -ΔV = E = (V/d). Below are the first (L = 0.0267m) and last (L = 0.9m) potential distributions between the centres of the plates.



*Backward Time Implicit Heat Equation*

Using a similar Taylor expansion method as before, except around tm;

Thus, each node in the previous time step can be constructed from the nodes in the current time step. Constructing a matrix equation ( V(m) = A.V(m-1) ) to allow the whole of the previous step to be constructed from the current time step, then inverting that matrix allows the current step to be constructed from the previous step. Redefining (m-1) -> (m) allows for the next step to be constructed implicitly from the current step – thus allowing for stability under all initial conditions.

*Physical Problem*

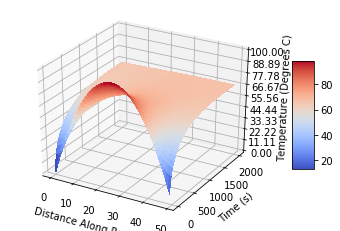
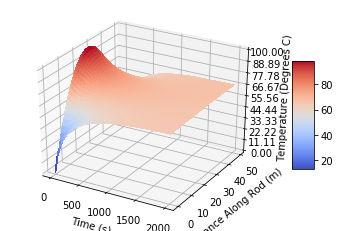
To test this method, it was simplified to 1D s.t. the equation became;

and, to account for the lack of flow from outside the boundary of the sample area, the first and last diagonal terms in the time-step matrix , A, were changed to to account for only the one adjacent node.

Then the physical constants needed to be decided on. The thermal diffusivity, α, follows the relation , where k is the thermal conductivity, ρ is the density and cp is the specific heat capacity. Using the given values

The first test was the initial condition of a uniform temperature across the rod (). As expected, the whole of each temperature vector remained at that constant temperature for indefinite time (with minute fluctuations introduced from errors).

The second test was that of an initial exponential distribution;

Again, these are the expected results and at each steps the magnitude of the temperature vector remained (nearly) constant (<0.000000001% error), thus the rod remained perfectly insulated.

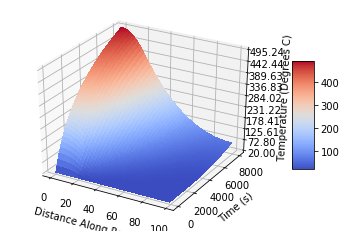
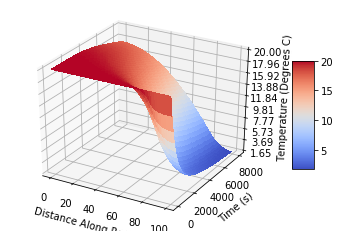
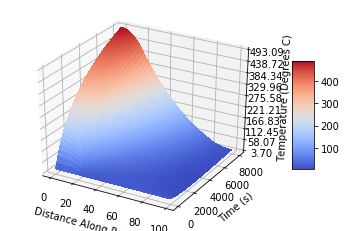
To expand the method for heat flow from the surroundings a step was added after each matrix multiplication that accounted for it. To model the heat flow into the rod over a time step some assumptions were made. The first was that the surroundings remained at constant temperature, the second was that the distribution of surroundings was: 1000oC for an iteger number of sites, 0oC ice for an iteger number of sites from the other end and perfect insulation elsewhere, and the third was that the heat flow density per time-step, Q, is proportional to the size of the time step, , a ‘thermal contact conductance, , and the difference between the site temperature and the external temperaturep [2]:

This amount was added to each site after each time-step iteration.

for air = 1.9 x 10-5 m2 s-1 [3]

for ice = 0.0012 m2 s-1 [4]

Below are the results for a 1m long rod, of 100 sites. Top left – 20cm in oven. Top right – 20cm in ice. Bottom right – 20cm in oven, 20cm in ice.

[1] - Lide, David R., ed. (2009). *CRC Handbook of Chemistry and Physics* (90th ed.). Boca Raton, Florida: CRC Press. p. 2-65. ISBN 978-1-4200-9084-0.

[2] -  Holman, J. P. (1997). Heat Transfer, 8th Edition. McGraw-Hill.

[3] - glossary.ametsoc.org/wiki/Thermal\_diffusivity

[4] - https://www.engineeringtoolbox.com/ice-thermal-properties-d\_576.html