Practical Numerical Simulation

Anthony Bourached

March 20, 2015

1 Poisson Equation

1.1 How to Run

The code is currently set to solve the problem with dirichlet boundary data. Uncomment one line (noted in the program) in order to change to Neumann boundary data. The function 'print to file' is currently set to print $\phi(1/2, 1/2)$ to screen and print the x, y and z component of each coordinate in the field. However, the latter is commented out to improve computation time if we are uninterested in plotting.

We update points by considering several different regions. The function update takes 7 parameters. The first two specify the region we're interested in ie j_{max} and j_{min}. The next four specify conditions for that region: it will skip the region between the third and fourth parameter; and same for the fifth and sixth. This can be seen from inside the function. This method may seem cumbersome to the region however the method has been chosen as it reduces computation time by having fewer for loops and if statements that would be necessary to sepcify the boundaries of A and B otherwise. I believe this is reflected in the run time of the programs.

We allow the corners to be updated twice in the update boundary function. This will make no difference to the final convergence but the extra computation time taken to update those four extra points per step is less costly and less cumbersome to the eye than adding extra conditions in the if statements for this function. This was determined by running the program in each case. The difference was found to be 0.02 seconds of run time.

1.2 The appropriate ω

We wished to determine the optimum value of ω for fastest convergence. To do this we put an extra for loop in main that looped over ω in the range from 0 to 1 in steps of 0.1. We measured the number of steps of the algorithm takes to converge to three significant figures.

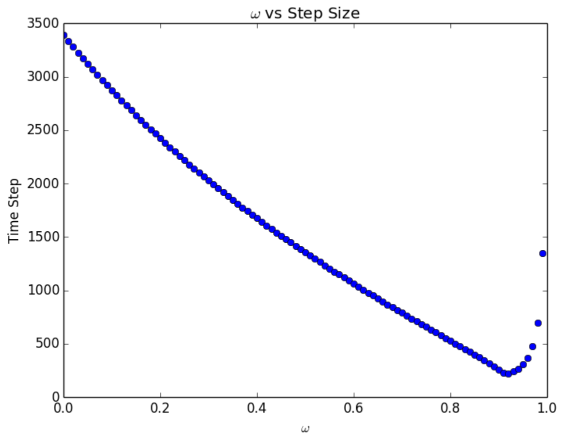

Figure 1: The effect of ω on the speed of convergence for Dirichlet Boundary conditions.

From figure 1 we can see that we clearly have fastest convergence in the range $0.85 <= \omega <= 1$. To determine the desired ω more accurately we decreased the step size with which we increase ω by a factor of 100. This can be seen in figure 2.

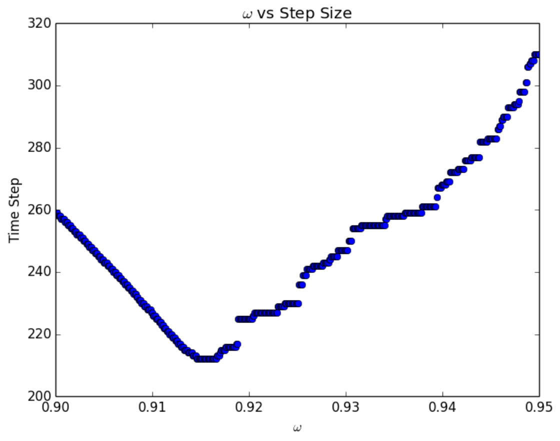

Figure 2: The effect of ω on the speed of convergence for Dirichlet Boundary conditions.

We concluded from this that our optimum choice is $\omega = 0.915$.

2 Dirichlet Boundary Conditions

$$\phi(\frac{1}{2}, \frac{1}{2}) = -0.365 \tag{1}$$

It was ensured that this value was correct by returning a flag for each given point. The flag would be 1 if the difference between the absloute value of the difference between the value at given point at iteration k+1 ($\phi^{k+1}(i,\ j)$) and k ($\phi^{k}(i,\ j)$) is less than $1 \times 10^{-3} \cdot \phi^{k+1}(i,\ j)$. Multiplying by $\phi^{k+1}(i,\ j)$ is necessary to ensure that we are of the right order of magnitude. For example, if $\phi^{k+1}(i,\ j)$ is of the order 0.0001 then to give correct to three significant figures we need $1 \times 10^{-3} \cdot \phi^{k+1}(i,\ j)$ giving order 0.0000001. In this way we were guaranteed for every point to have converged correct to at least 3 significant figures and we break the while loop using boolean logic when the following condition is met:

$$\sum_{i,\ j}^{l} \text{flag} = 0 \tag{2}$$

Where l^2 is the size of the field. A diagram of the converged field is shown in figure 3.

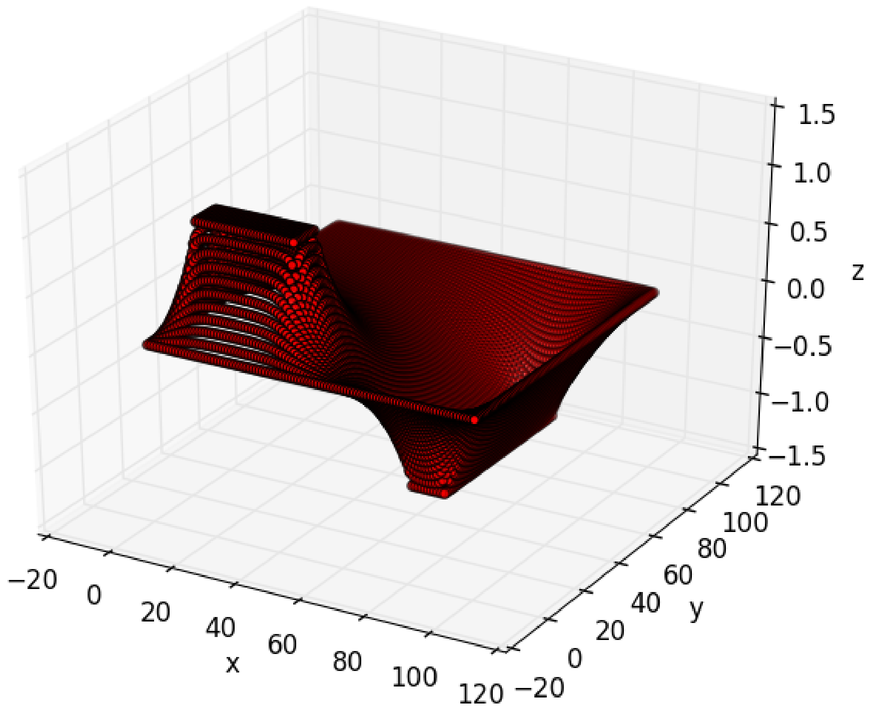

Figure 3: A plot of the converged field for Dirichlet Boundary Conditions.

3 Neumann Boundary Conditions

We determined convergence for this part in the same way as for Dirichlet conditions. We found:

$$\phi(\frac{1}{2}, \frac{1}{2}) = -0.399 \tag{3}$$

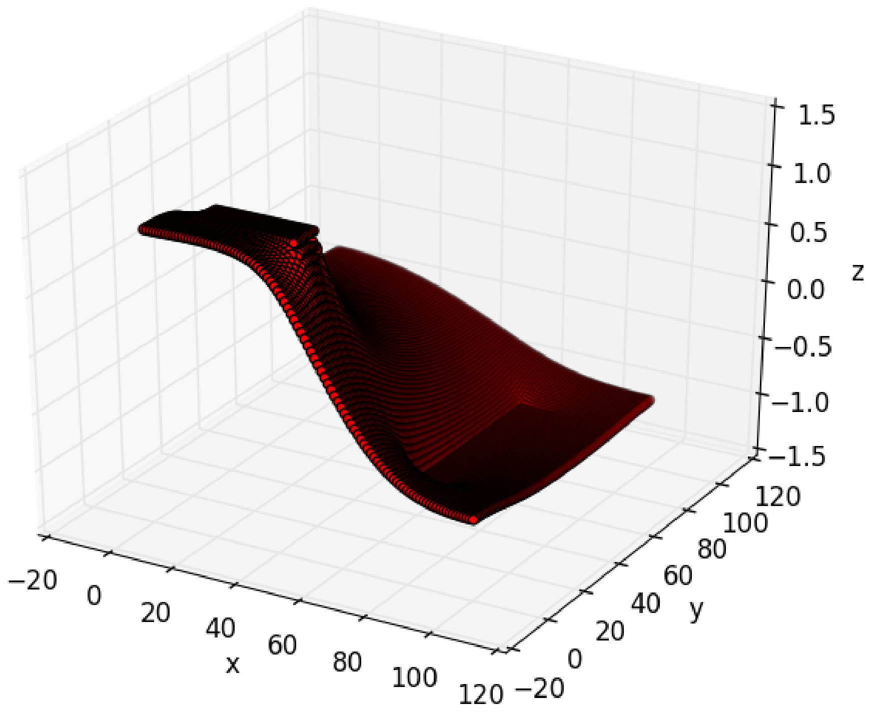

Figure 4: A plot of the converged field for Neumann Boundary Conditions.