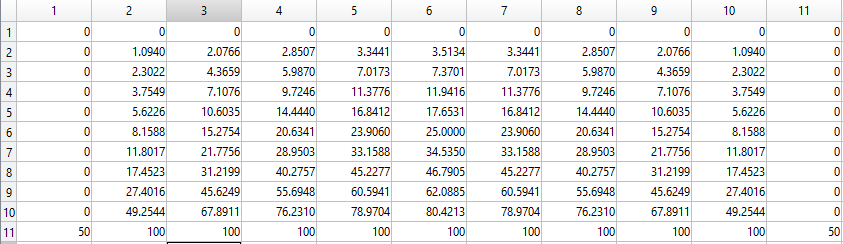
**QUESTION A**

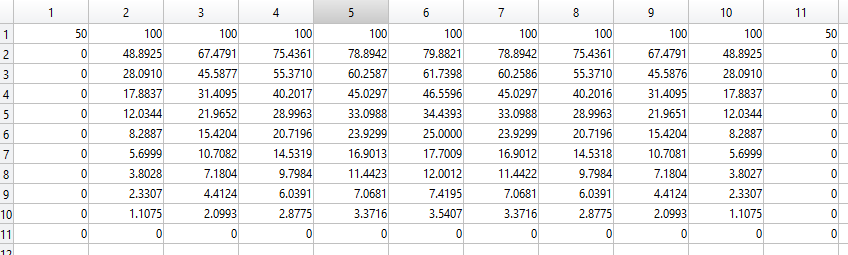
The analytical solution to the problem is implemented in ***laplacianAnalytic.m.*** The exact solution obtained is shown in the figure below:

****

**QUESTION B**

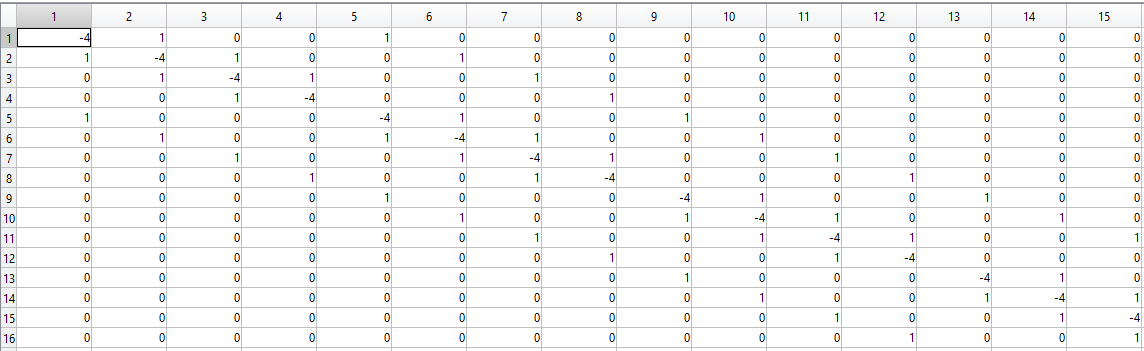
(i)

The code was written into a function named ***solveSOR*** and tested in file ***laplacianNumeric.m***. The resulting matrix is shown in the figure below:



(ii)

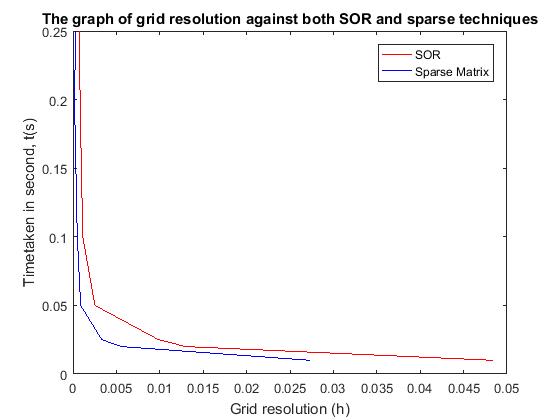
A banded sparse matrix generator function was written and named ***computerSparse*** which generates sparse matrix for any arbitrary rectangular 2D problem. The function was tested in ***laplacianSparse.m***. The resulting banded sparse matrix of h=0.2 grid resolution is shown in the figure below:



Computational Efficiency of Sparse Matrix Technique and SOR

The computational efficiency of both techniques are tested in ***sparseVsSOR.m.*** Below is the tabularized time taken in seconds for both sparse matrix and SOR techniques for different grid resolutions or discretization levels:

|  |  |  |  |
| --- | --- | --- | --- |
| h (resolution) | SOR time (s) | (A\b) WITHOUT sparse matrix generation time (s) | (A\b) WITH sparse matrix generation time (s) |
| 0.010 | 0.048300 | 0.027240 | 4.184827 |
| 0.020 | 0.012971 | 0.005580 | 0.264360 |
| 0.025 | 0.009860 | 0.003316 | 0.105036 |
| 0.050 | 0.002526 | 0.000880 | 0.007524 |
| 0.100 | 0.001127 | 0.000493 | 0.001754 |
| 0.200 | 0.000810 | 0.000171 | 0.001134 |
| 0.250 | 0.000781 | 0.000310 | 0.009230 |



**OBSERVATIONS:**

1. It is observed from both the table and the graph above that, sparse matrix approach to the solution tends to be faster when compare to successive over relaxation method, however it can also be noted from the above table that, if the time taken to generate the matrix is considered in the whole process of sparse matrix technique, then it can be conclude that SOR is better than the sparse matrix technique.
2. As contained in table containing SOR solution above and exact/analytic solution contained in table in QUESTION A above, choosing column 2 and value next to the 100V potential, we have 49.2544 (exact value) and 48.8925 (SOR) solution for h = 0.1. Calculating the absolute error for this value:

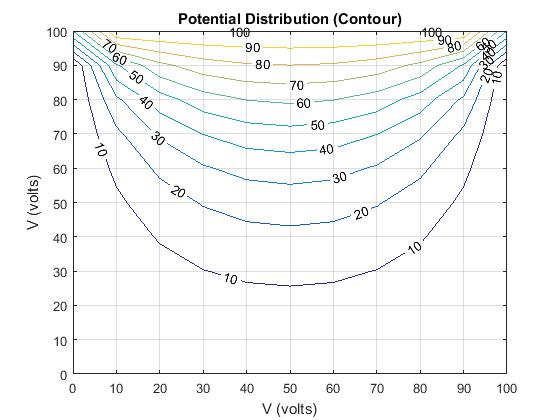
(49.254384342897060- 48.892536888651650) x 100 = 0.7401% error

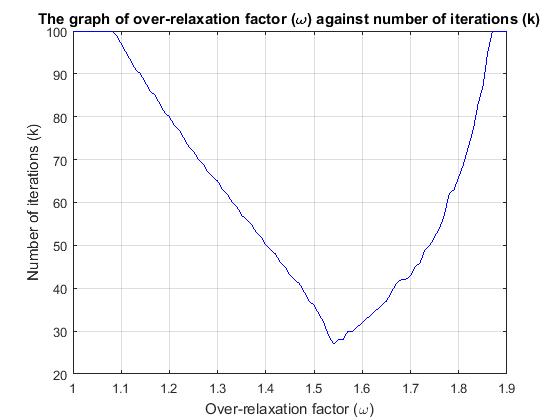
48.892536888651650

The above error shown in the calculation above accounts for the discretization error in the numerical solution.

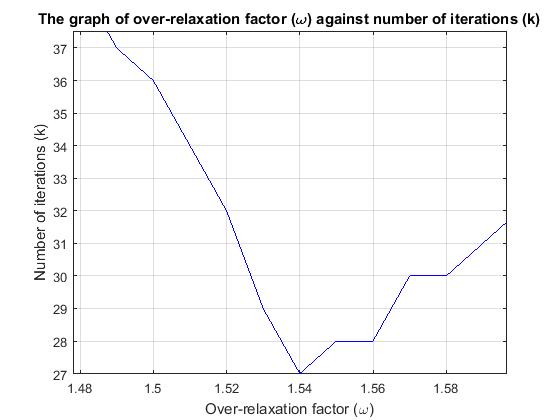
**QUESTION C**

The solution to this question was tested in the file named ***omegaVsNumberOfIterations.m*.** The result is presented in the figures below:





(Normal Graph)

  
 (Enlarged Figure)

**OBSERVATIONS:**

1. The optimum over relaxation value is 1.54 and it is having minimum number of iterations – 27 iterations, as show in the figure above (Enlarged figure).
2. In graph labeled (Normal Figure) depicting the number of iterations against over relation constant it can be observed that after it converged at optimum value 1.54 it then diverged again.
3. The flux calculation was done in the file ***laplacianNumeric.m*** as the program used to determine optimum value of omega above contain loops and ended at the last diverging value of over relaxation constant, however, flux calculation in both files tends to or approximately equal to zero.

(9.2054x10-4). This result shows that, in steady state the total flux across the boundary is zero.

**QUESTION D**

The solution code to this problem is contained in ***cgVsSOR.m.***

|  |  |  |  |
| --- | --- | --- | --- |
| h (resolution) | SOR time (s) | (CG) WITHOUT sparse matrix generation time (s) | (CG) WITH sparse matrix generation time (s) |
| 0.010 | 0.052283 | 0.010634 | 4.608548 |
| 0.020 | 0.016308 | 0.004182 | 0.279933 |
| 0.025 | 0.008184 | 0.003220 | 0.125786 |
| 0.050 | 0.002619 | 0.002329 | 0.009401 |
| 0.100 | 0.000962 | 0.001525 | 0.002925 |
| 0.200 | 0.000841 | 0.001206 | 0.001735 |
| 0.250 | 0.000360 | 0.001071 | 0.001635 |

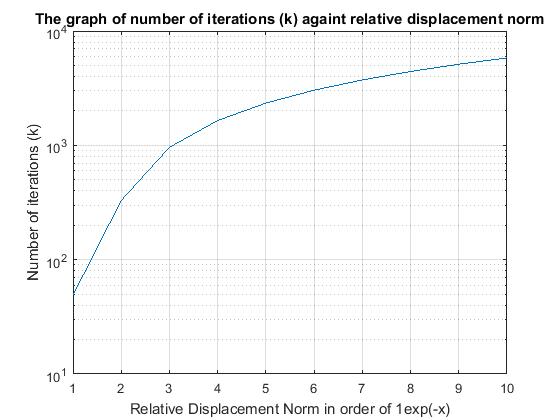
**OBSERVATIONS:**

From the table above, it can be observed and concluded that:

1. Conjugate gradient method is faster than successive over relaxation technique method when we have large number of grid (the largest in this case is 9801x9801 matrix for h=0.01). However, SOR tends to be preferable and faster for smaller number of grids (part colored red as shown in the table above).
2. Furthermore, it can also be noted that SOR is faster than CG when time to generate the sparse matrix is included.

**QUESTION E**

Solution to this question is located in ***numberOfIterVsDispNorm.m***. The resulting plot is shown in the figure below:

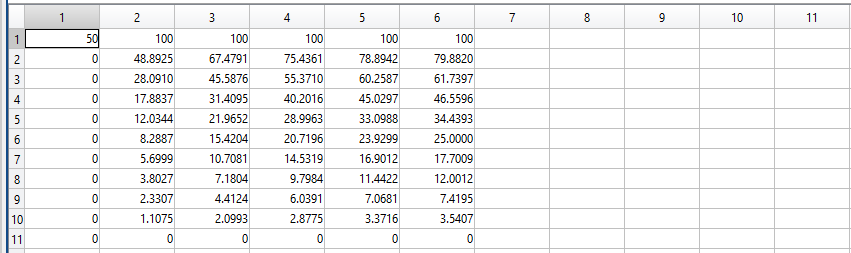


**OBSERVATIONS:**

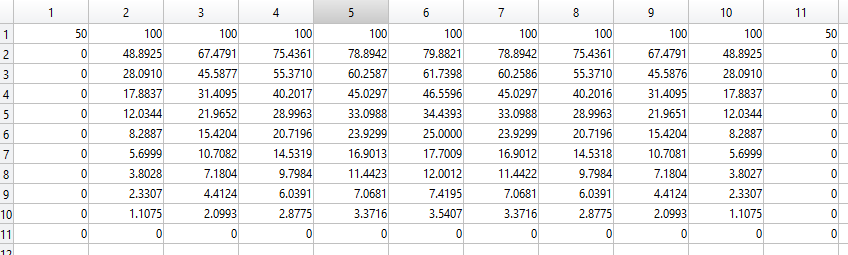
1. As shown in the figure above, it can be observed that the number of iterations increases as the relative displacement norm goes to zero. In fact, the code was set to 10000 maximum number of iterations to be able to observe this.
2. When relative displacement norm, ɛ, is set to 1e-16, it is observed that number of iteration increased to a very high value, 10780 iterations.

**QUESTION F**

The symmetric solution of SOR technique which is the solution to this question is implemented in ***laplacianSymmetry.m***. The solution is shown below:



Checking it against the full solution contained in QUESTION B above, the figure is repeated below for sake of comparison:



**OBSERVATIONS:**

1. From the two tables above, it can be concluded that the resulting matrix is the symmetric half of the full solution presented in Question B (also shown above). As it represents the symmetric half of the full solution from x = 0.1 to x = 0.5.
2. However, the resulting symmetric half obtained in the solution to this question is not in itself symmetric. But it is positive definite because it has non-negative value.
3. The boundary condition used for the symmetric solution is Neumann boundary condition given by dɸ/dn = 0
4. The finite difference of this problem at the boundary is given by:

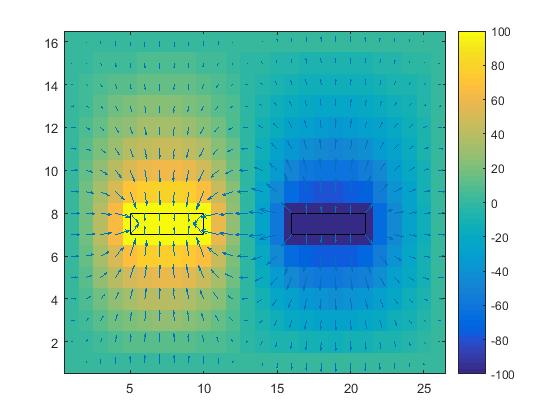
R = (ɸ(i-1,j) + ɸ (i+1,j) + 2ɸ (i,j-1))/4;

ɸ(i, j) = ɸ(i,j) + ω\*(R - ɸ(i,j));

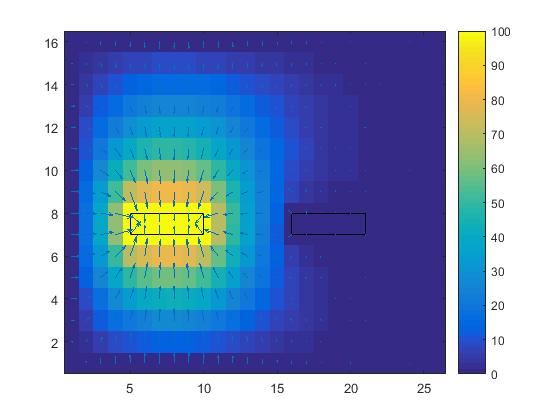
**QUESTION G**

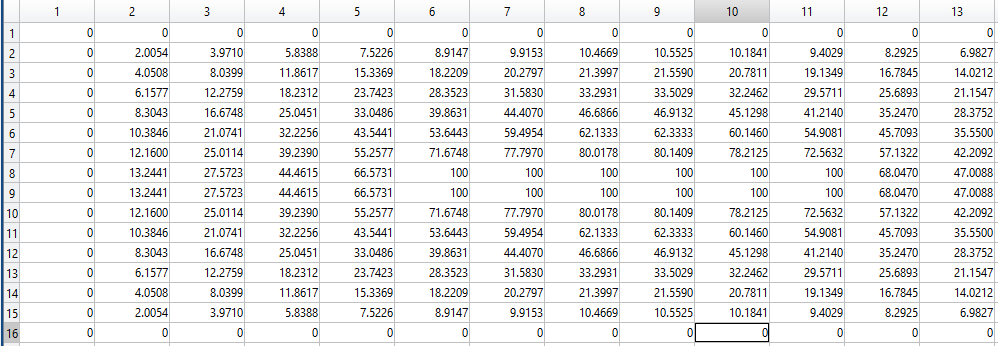
Capacitance matrix for the solution is implemented for this question in ***capacitanceMatrix.m***. The following figures were obtained for different value of potentials to the conductors.

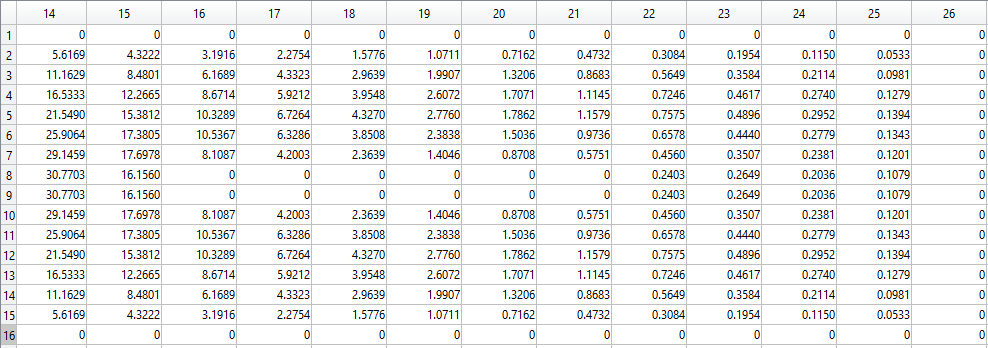
When conductor 1 is set to 100V and conductor 2 is set to -100, figure below shows the flux movement between the two conductors:



When conductor 1 is set to 100V and conductor 2 is grounded, the figure below was obtained:







**OBSERVATIONS:**

1. As shown in the figures above, the resulting matrix is symmetry on the y plane.
2. The obtained capacitance matrix is:

C = 8.80346015938367e-13 -6.00380433265055e-12

-6.00380433265055e-12 8.80346015938367e-13