FEEG6002 Advance Computational Method Coursework 2015/16: PDE methods.

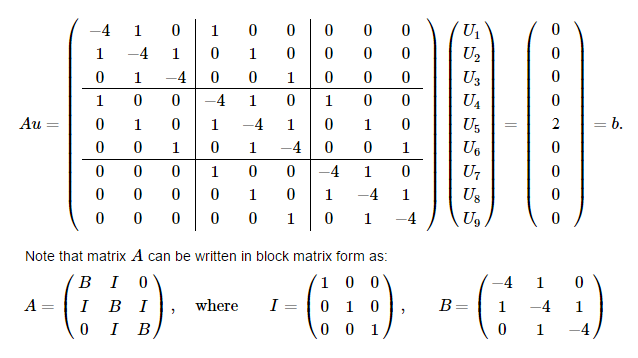
Name : Alan Tan Kay Meng

Student ID : 25816322

Question 5 : Give a short write-up for the codes you have written in questions 1-4.

Q1:

In Question 1 consist of 3 main function which is get\_A, get\_rho, and laplace2d. Function get\_A and get\_rho returns matrix A and matrix b. Boundary condition is given 2 for ρ(0.5,0.5) and zeros everywhere else for the PDE ∇2*u* ρ,which is the b vector created by function get\_rho. The figure below extracted from FEEG6002 lecture notes numerical method 05 shows the arrangement of the A matrix, b matrix and the equation to solve PDE (1).



The origin of the B matrix is from the second derivative of the first central difference

Where the mesh separation is h. Thus, if N = number of mesh grid, h = 1 / (N-1). This is why in the function laplace2d for question 1, A = get\_A(n) \* (1/(h\*\*2)).

The code in the next line U = sp.linalg.solve(A, b) is where the build in solver solve for A.U = b to solve for vector U. After solving for the U, dot product of Matrix A and the solved vector U is computed to check if the answer of PDE (1) at ρ(0.5,0.5) is equal to 2 in the code line 77, CheckU = np.dot(A,U).

The function laplace2d is coded to print the value of CheckU at midpoint of vector in code line 90.

“ Q1: Value of the dot product A.u1 is 2.000 at (0.5,0.5). “

Q2:

Question 2 code consist of 2 functions, iterate() and gauss\_seidel(). In the iterate function, A matrix is extracted from get\_A() and is solved with boundary condition written within the code. This means that the function iterate() is made specifically to solve PDE (1). By using Gauss-Seidel method with successive over relaxation given by the below equation,

where is the relaxation parameter. The matrix is solved row by row by giving the iterate function an initial guess for x of length N2. For example, the first row in PDE(1) , . By using the above equation, .

After the matrix is solved row by row, the process is repeated until the solution of x converged. This is done in the function gauss\_seidel which copy the previous x vector, compute a new x vector and compare the difference between the two vectors. If the difference of x\_old and x\_new is less than the tolerance, which in this case 1.0E-09 the x vector is said to be converged. Number of iteration required to converge the vector is recorded to be 50 for N = 9.

Default value for the relaxation parameter ω is set to be equal to 1. In the code line 218 is where the function calculating the optimal value of ω using the below equation when number of iteration, niter is equal to parameter k = 10 set in the function.

2 difference of x vector dx1 and dx2 is extracted at 10th and 11th iteration for the gauss\_seidel function to calculate the optimal value for ω. Thus the above equation can be rewritten for our case in the following form.

After 50 iteration, the converged x value substituted into A.X to validate value of ρ(0.5,0.5).

“Q2: Value of the dot product A.x2 is 2.000 at (0.5,0.5). “

“Q2: niter = 50, optimal omega = 1.53462 “

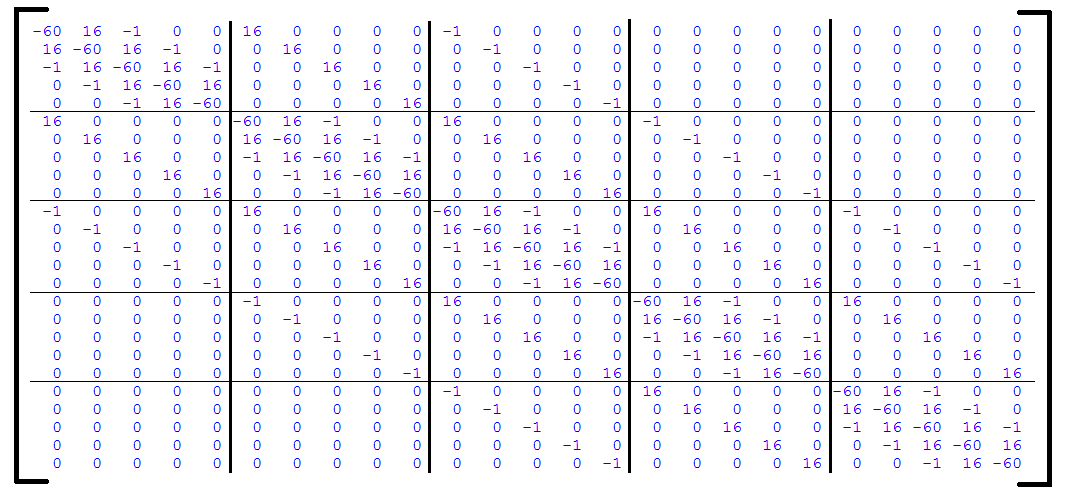
Q3:

The given stencil yields the following equation,

where and are the mesh size in the x and y directions respectively. Assuming a uniform grid

We can discretize the continuous Laplace equation in the bounded into a discrete eq:

The A matrix generated using the same method used in Q1 is shown below with N = 5 grid.

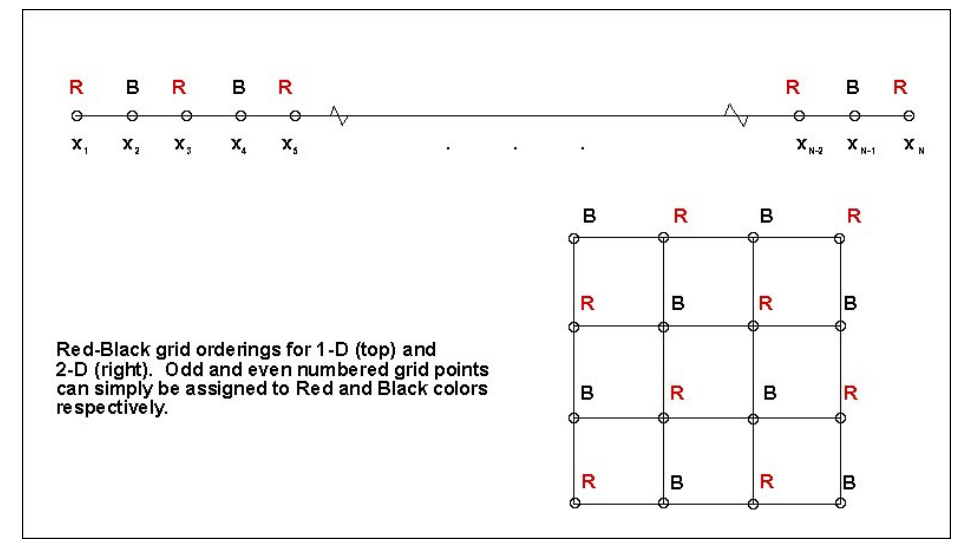


In the code line 298 is where the PDE is solved with the laplace2dq3() function which is similar to the solver used in Q1. Dot product of the A matrix and solved U3 is computed and result is printed to the display.

“Q3: Value of the dot product A.u3 is 2.000 at (0.5,0.5). ”

Q4:

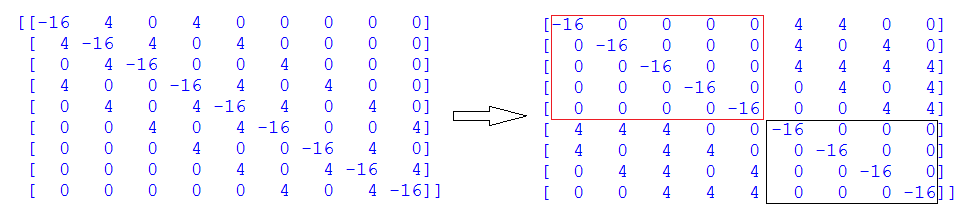
A red-black matrix is often useful when trying to compute an inherently sequential problem in parallel. For example, by using a red black coloring scheme for grid points, the Gauss-Seidel method can be vectorized to compute the solution on all Red grid points simultaneously, followed by all Black grid points. To create the red-black matrix, one should use a corresponding red-black grid which chooses alternating grid points in such a manner that odd grid points are colored red and even numbered grid points are colored black or vice-versa. The graph below shows a red-black ordering for 1D and 2D grids, this idea can also be extended similarly to 3D.



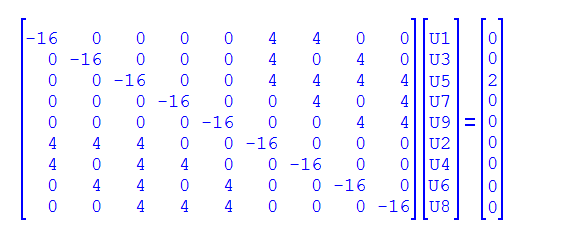
A red-black matrix should take the block form

Where D1 and D2 are strictly diagonal matrices and C is diagonally sparse. This would in general correspond to the solution of the linear system.

The matrix A\*(1/h2) from Q2 is rearranged into a red-black matrix using the function redblackA(N) and is shown in the figure below. The boundary condition generating function get\_rho(N) is rearranged with function redblackb(N) and queue to be solved using Guass-seidel method.



The following PDE is solved using Gauss-seidel method.



The solved vector x is reverted back to normal sequence with the function redblackb\_rev(x). Similar procedure to validate the value at (0.5,0.5) using dot product of Matrix A and vector x, ( A.X4 ) after reverting vector x back to the original arrangement.

“Q4: Value of the dot product A.x4 is 2.000 at (0.5,0.5). “

“Q4: niter = 53, optimal omega = 1.46287 “

Solved U vector for all question attempted using grid N=3:

Q1: [-0.03125 -0.0625 -0.03125 -0.0625 -0.1875 -0.0625 -0.03125 -0.0625 -0.03125]

Q2: [-0.03125 -0.0625 -0.03125 -0.0625 -0.1875 -0.0625 -0.03125 -0.0625 -0.03125]

Q3: [-0.03012001 -0.05835752 -0.03012001 -0.05835752 -0.16224802 -0.05835752 -0.03012001 -0.05835752 -0.03012001]

Q4: [-0.03125 -0.0625 -0.03125 -0.0625 -0.1875 -0.0625 -0.03125 -0.0625 -0.03125]