CSE 557
Spring 2000
Solution for Assignment #1

Anirudh Modi

January 31, 2000

Abstract

A simple relaxation method for the solution of a 2D Laplace equation ($\nabla^2\phi = 0$) on a rectangular domain is implemented using matrix arrays in MATLAB with two different approaches. The rate of convergence of the two approaches are studied for various grid sizes. It is noted that the convergence rate is inversely proportional to the size of the grid, and that the latter approach (version 2) is twice as fast as the first approach (version 1).

1 Approach

The general solution for a Laplace equation $\nabla^2\phi(x,y) = 0$ is reasonably complex and depends on the specified boundary conditions. For simplicity, the boundary conditions in our numerical experiment are specified only on the "edges" and are set to be zero, so that the expected solution of the problem is trivial (zero everywhere), thus making the comparison easy. Two versions of the relaxation algorithm are implemented, one using the average of only the "old" values of the neighbors (version 1):

$$\text{Step 1 of 2:} \qquad \phi'_{i,j} = 0.25(\phi_{i-1,j} + \phi_{i+1,j} + \phi_{i,j-1} + \phi_{i,j+1}) \qquad \forall \quad i,j$$
$$\text{Step 2 of 2:} \qquad \phi_{i,j} = \phi'_{i,j} \qquad \forall \quad i,j$$

and the other using whatever values are stored in the neighboring locations whether or not they are old or new (version 2):

$$\text{Step 1 of 1:} \qquad \phi_{i,j} = 0.25(\phi_{i-1,j} + \phi_{i+1,j} + \phi_{i,j-1} + \phi_{i,j+1}) \qquad \forall \quad i,j$$

The MATLAB program for the above approaches is attached at the end of the report.

2 Results

The problem was solved using both the approaches mentioned above. To study the effect of the size of the grid on the convergence rate, several runs were conducted with increasing grid sizes and the number of iterations required to converge to a fixed state (where the residual was two orders lower ,i.e., 100 times smaller than the initial residual) was recorded. The data for both the approaches can be seen in figure 1. A polynomial fit was done on the data, and it was observed that a quadratic curve fits the data almost perfectly. The number of iterations required for an $N \times N$ grid for *version 1* of the program can be represented as $0.890N^2 + 1.97N + 0.06$, and the same for *version 2* of the program can be represented as $0.445N^2 + 1.00N + 0.67$. A sample convergence plot ($\log_{10}||$Frobenius norm$||$ vs number of iterations) for a 50×50 grid for both the approaches can be seen in figure 2.

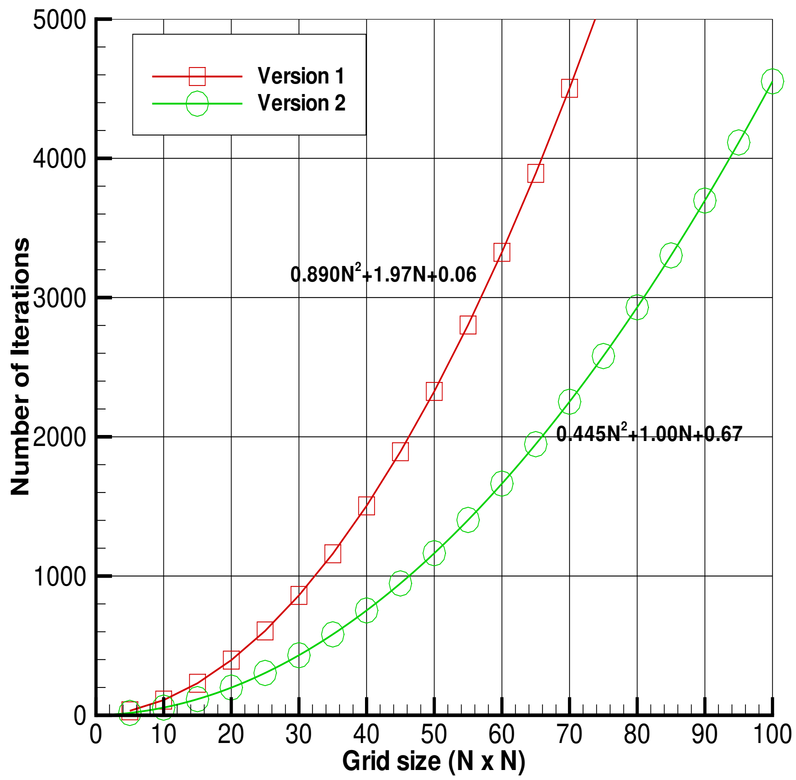

Figure 1: Effect of the grid size on the number of iterations required (for $\log_{10}||\text{Residual}|| = -2$)

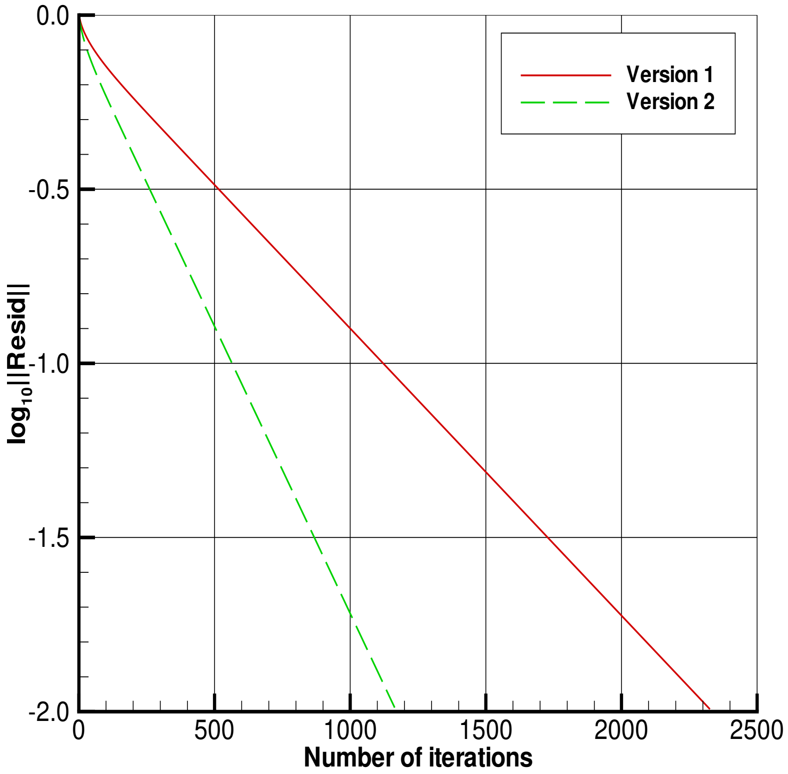

Figure 2: Convergence rate for a 50×50 grid

3 Conclusions

From the above plot (figure 1) and the polynomial fit data, it is concluded that the number of iterations required to reach the same state of convergence for a given grid size $N \times N$, is directly proportional to the size of the grid (which has N^2 nodes). For large N, it is roughly $0.890N^2$ for *version 1*, and $0.445N^2$ for *version 2*. In other words, the rate of convergence is inversely proportional to the grid size. Comparing the curves for the two different approaches, *version 1* and *version 2*, it is clear that *version 2* is faster by a factor of 2, and is hence a better relaxation algorithm for solving the above equation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                          %
%% CSE 557 - HW #1                                                          %
%% [http://www.cse.psu.edu/~plassman/cse557/assignments/hw1.html]           %
%% by Anirudh Modi (anirudh@anirudh.net) on 1/28/2000-Fri                   %
%% (modi@cse.psu.edu)                                                       %
%%                                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

num_grid = 20;  % number of different grid sizes
errLimit = 2.0; % convergence limit specified as -log10(norm)
version = 1;    % which approach to be used (1 or 2?)
plotflag = 0;   % whether the data should be plotted?

filename = sprintf('conv%d.out',version);
outf = fopen(filename,'w');

for count=1:num_grid
  k = 5*count;          % k = 5,10,....,95,100
  % print out what we are doing.
  fprintf('Version %d for HW #1 with gridsize %dx%d and log10(resid) = %g\n',...
      version,k,k,-errLimit);

  % initialize two arrays, for the relaxation iteration...
  oldM = zeros(k+2,k+2); newM = oldM;

  % x and y positions for plotting
  X = [0:k+1]./(k+1);
  Y = [0:k+1]./(k+1);

  % set the middle of oldM to ones, keeping the boundary at zero.
  oldM(2:k+1,2:k+1) = ones(k,k);
  if version == 2
        newM = oldM;
  end

  % plot the initial image
  if plotflag == 1
    graph_title = sprintf('Initial conditions (%dx%d grid)',k,k);
    figure(1); subplot(3,1,1); surf(Y,X,oldM); title(graph_title);
  end

  % do a few iterations iterations of relaxation (version 1):
  it = 0;
  errlog = 10.0;
  errlog0 = norm(oldM);
```

```
  while (-errlog < errLimit)
     it = it + 1;
     if version == 1
        for i=2:k+1
           for j=2:k+1
            newM(i,j) = 0.25*(oldM(i-1,j)+oldM(i+1,j)+oldM(i,j-1)+oldM(i,j+1));
          end
        end
        oldM(2:k+1,2:k+1) = newM(2:k+1,2:k+1);
     elseif version == 2
        for i=2:k+1
           for j=2:k+1
            newM(i,j) = 0.25*(newM(i-1,j)+newM(i+1,j)+newM(i,j-1)+newM(i,j+1));
          end
        end
     end
    % copy interior values back to oldM
    errlog = log10(norm(newM)/errlog0);
  end      % while loop for convergence ends

  fprintf('k = %d, iter = %d, errlog = %g\n',k,it,-errlog);
  fprintf(outf,'%d %d\n',k,it);

  % plot the image after "it" iterations
  if plotflag == 1
    graph_title = sprintf('After %d iterations (Version %d)',it,version);
    figure(1); subplot(3,1,2); surf(Y,X,newM); title(graph_title); pause(0);
  end

end % the loop for grid size ends
fclose(outf);
```