1. See functions below

2.

a.

Solver	# iterations to converge	CPU time(s)	mdot_tot	phidot_tot
Jacobi	2445	283.98	0.00E+00	-5.79E-06
GS	1213	135.87	0.00E+00	-5.90E-06
GS_SOR, w=1.0	1213	143.97	0.00E+00	-5.90E-06
GS_SOR, w=1.1	990	117.54	0.00E+00	-5.87E-06
GS_SOR, w=1.2	803	95.22	0.00E+00	-5.93E-06
GS_SOR, w=1.3	645	76.45	0.00E+00	-5.92E-06
GS_SOR, w=1.4	509	60.60	0.00E+00	-5.92E-06
GS_SOR, w=1.5	390	46.29	0.00E+00	-5.97E-06
GS_SOR, w=1.6	284	33.64	0.00E+00	-6.17E-06
GS_SOR, w=1.7	188	22.30	0.00E+00	-6.13E-06
GS_SOR, w=1.8	87	10.31	0.00E+00	-4.99E-06
GS_SOR, w=1.9	137	16.24	0.00E+00	1.95E-07
GS_SOR, w=2.0	did not converge			

i. The optimum value of ω for GS_SOR is near 1.8

ii. Number of iterations required for convergence: Jacobi = 2445, GS = 1213, GS_SOR_optimal = 87

iii. Mass is completely conserved in every case. The total mDot is 0 for every case. Φ is converged within 5.79e-6 for Jacobi, within 5.90e-6 for GS, and within 4.99e-6 for GS_SOR_optimal. In this case, the optimal GS_SOR also had the minimum error in Φ.

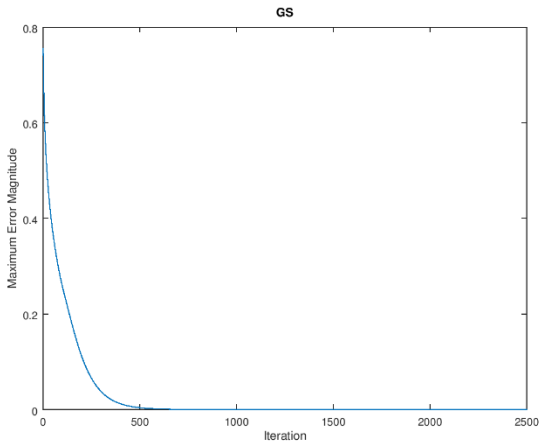
iv. The reduction in iterations from Jacobi to GS is very nearly 50%, as discussed in lecture. Although the same sweep direction was used throughout the process, no major errors appeared to accumulate during the solution. The optimal GS_SOR solver was much faster to convergence than either the Jacobi or GS cases, and was approximately 3X the number of grid points in one direction. The optimal GS_SOR solver required 3.5% the number of iterations of Jacobi and 7% the number of iterations of standard GS.

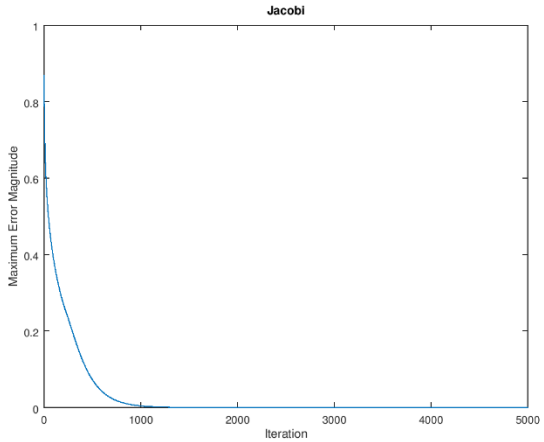
Solver	# iterations to converge	CPU time(s)	mdot_tot	phidot_tot
Jacobi	4834	542.79	0.00E+00	-5.79E-12
GS	2407	271.62	0.00E+00	-5.92E-12
GS_SOR, w=1.0	2407	293.08	0.00E+00	-5.92E-12
GS_SOR, w=1.1	1965	265.04	0.00E+00	-5.94E-12
GS_SOR, w=1.2	1596	212.10	0.00E+00	-5.97E-12
GS_SOR, w=1.3	1283	153.68	0.00E+00	-5.99E-12
GS_SOR, w=1.4	1014	121.02	0.00E+00	-5.95E-12
GS_SOR, w=1.5	778	93.00	0.00E+00	-6.12E-12
GS_SOR, w=1.6	569	67.81	0.00E+00	-6.16E-12
GS_SOR, w=1.7	378	45.79	0.00E+00	-5.96E-12
GS_SOR, w=1.8	171	20.38	0.00E+00	-4.94E-12
GS_SOR, w=1.9	275	32.80	0.00E+00	1.82E-12
GS_SOR, w=2.0	did not converge			

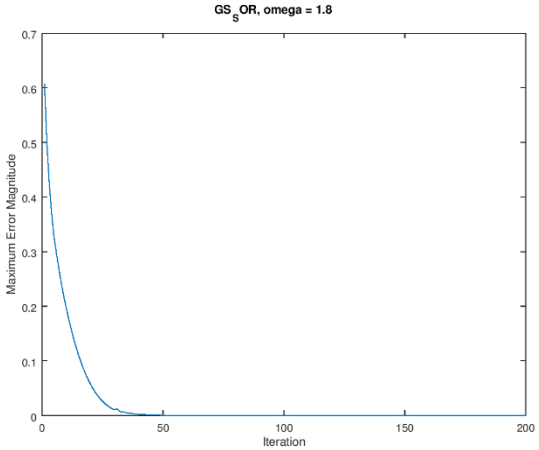
b.

i. Number of iterations required for convergence: Jacobi = 4834, GS = 2407, GS_SOR_optimal = 1.8
ii. CPU time required for convergence: Jacobi = 543 s, GS = 272 s, GS_SOR_optimal = 20.38 s
iii. Plots of the maximum iteration error magnitude (errmax) as a function of iteration number for Jacobi, GS and optimal GS_SOR are shown below. All curves have very similar shapes, but the convergence rate of the GS is twice as fast as the Jacobi, and the optimal GS_SOR is much faster than either Jacobi or GS (note the differing x-axis scales).
iv. Plots of the maximum residual magnitude (resmax) as a function of iteration number for Jacobi, GS and optimal GS_SOR are shown below. Interestingly, the rate of convergence for residual error magnitude appears to be much faster than for error magnitude. Additionally, the optimal GS_SOR appears to have some slight oscillitory behavior.

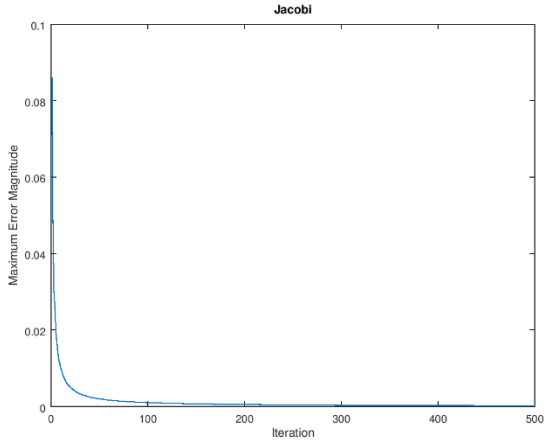
Error magnitude plots

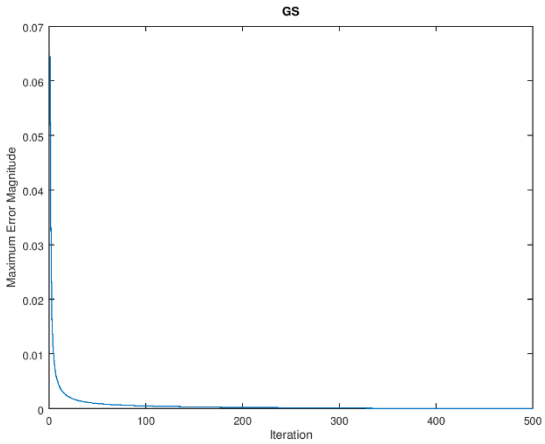


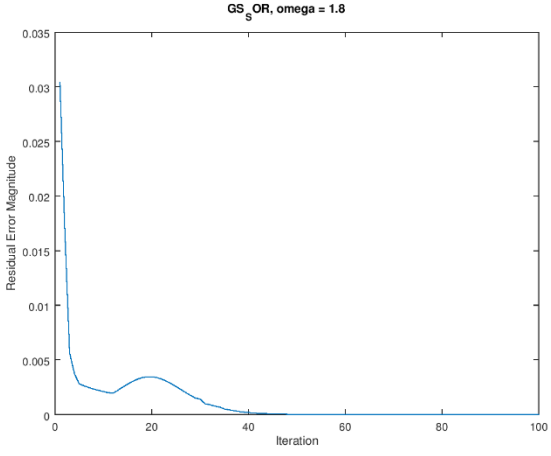




Below are the residual magnitude plots (ignore the y-axis label):







Defined functions:

```
%%% begin jacobi %%%

function jacobi

  % globals needed
  global ap ano aso aea awe q phidir n m epsit resmax errmax nitmax xc yc
  global phiold phinew iterstore

  iterstore = 0;  % storage variable for iteration count
  phiold = phidir;  % initialize array for "old" phi values
  phinew = phidir;  % initialize array for "new" phi values

  resTemp = zeros(m, n);  % initialize local array to store residuals

  % set internal nodes to be zero
  phiold(2:end-1, 2:end-1) = 0;
  phinew(2:end-1, 2:end-1) = 0;

  nit = 0;
  while max(max(abs(phidir - phinew))) > epsit*max(max(phinew));

    nit = nit + 1;

    for j = 2:m+1
      for i = 2:n+1
        phinew(i, j) = (q(i,j) - ano(i,j)*phiold(i,j+1) ...
                        - aso(i,j)*phiold(i,j-1) ...
                        - aea(i,j)*phiold(i+1,j) ...
                        - awe(i,j)*phiold(i-1,j))/ap(i,j);
      end
    end

    errmax(nit) = max(max(abs(phidir - phinew)));

    % Periodically show results
    if mod(nit, 50) == 0
      fprintf('Jacobi iteration %.0f, errmax %.4e\n', nit, errmax(nit));
    end

    for j = 2:m+1
      for i = 2:n+1
        resTemp(i,j) = q(i,j) - ano(i,j)*phinew(i,j+1) ...
                        - aso(i,j)*phinew(i,j-1) ...
                        - aea(i,j)*phinew(i+1,j) ...
                        - awe(i,j)*phinew(i-1,j) ...
                        - ap(i,j)*phinew(i,j);
      end
    end

    resmax(nit) = max(max(resTemp));
```

```
    phiold = phinew;
    phi = phinew;

    iterstore = nit;  % save number of iterations in global var
    % Break out of the while loop if maximum number of iterations is reached
    if nit == nitmax;
      break
    end

  end  % end while loop

  if nit == nitmax;
    fprintf('Jacobi solution did not converge in %4.0f iterations.\n',
nitmax)
  else
    fprintf('Jacobi solution converged in %4.0f iterations.\n', nit)
  end

end

%%% end jacobi %%%



%%% begin gs

function gs

  % globals needed
  global ap ano aso aea awe q phidir n m epsit resmax errmax nitmax xc yc
  global phinew iterstore

  iterstore = 0;  % storage variable for iteration count
  phinew = phidir;  % initialize array for "new" phi values

  resTemp = zeros(m, n);  % initialize local array to store residuals

  % set internal nodes to be zero
  phinew(2:end-1, 2:end-1) = 0;

  nit = 0;
  while max(max(abs(phidir - phinew))) > epsit*max(max(phinew));

    nit = nit + 1;

    for j = 2:m+1
      for i = 2:n+1
        phinew(i, j) = (q(i,j) - ano(i,j)*phinew(i,j+1) ...
                        - aso(i,j)*phinew(i,j-1) ...
                        - aea(i,j)*phinew(i+1,j) ...
                        - awe(i,j)*phinew(i-1,j))/ap(i,j);
      end
```

```
    end

    errmax(nit) = max(max(abs(phidir - phinew)));

    % Periodically show results
    if mod(nit, 50) == 0
      fprintf('GS iteration %.0f, errmax %.4e\n', nit, errmax(nit));
    end

    for j = 2:m+1
      for i = 2:n+1
        resTemp(i,j) = q(i,j) - ano(i,j)*phinew(i,j+1) ...
                        - aso(i,j)*phinew(i,j-1) ...
                        - aea(i,j)*phinew(i+1,j) ...
                        - awe(i,j)*phinew(i-1,j) ...
                        - ap(i,j)*phinew(i,j);
      end
    end

    resmax(nit) = max(max(resTemp));

    phiold = phinew;
    phi = phinew;

    iterstore = nit;  % save number of iterations in global var
    % Break out of the while loop if maximum number of iterations is reached
    if nit == nitmax;
      break
    end

  end  % end while loop

  if nit == nitmax;
    fprintf('GS solution did not converge in %4.0f iterations.\n', nitmax)
  else
    fprintf('GS solution converged in %4.0f iterations.\n', nit)
  end

end

%%% end of gs



%%% begin gs_sor

function gs_sor

  % globals needed
  global ap ano aso aea awe q phidir n m epsit resmax errmax nitmax xc yc
  global phinew omega phiold iterstore

  iterstore = 0;  % storage variable for iteration count
  phinew = phidir;  % initialize array for "new" phi values
```

```
  phiold = phidir;  % initialize array for "old" phi values

  resTemp = zeros(m, n);  % initialize local array to store residuals

  % set internal nodes to be zero
  phiold(2:end-1, 2:end-1) = 0;
  phinew(2:end-1, 2:end-1) = 0;

  nit = 0;
  while max(max(abs(phidir - phinew))) > epsit*max(max(phinew));

    nit = nit + 1;

    for j = 2:m+1
      for i = 2:n+1
        phinew(i, j) = omega*(q(i,j) - ano(i,j)*phiold(i,j+1) ...
                        - aso(i,j)*phinew(i,j-1) ...
                        - aea(i,j)*phiold(i+1,j) ...
                        - awe(i,j)*phinew(i-1,j))/ap(i,j) ...
                        + (1-omega)*phiold(i,j);
      end
    end

    errmax(nit) = max(max(abs(phidir - phinew)));  % compute errmax

    % Periodically show results
    if mod(nit, 50) == 0
      fprintf('GS-SOR [%1.1f] iteration %.0f, errmax %.4e\n', omega, nit, ...
              errmax(nit));
    end

    for j = 2:m+1
      for i = 2:n+1
        resTemp(i,j) = q(i,j) - ano(i,j)*phinew(i,j+1) ...
                        - aso(i,j)*phinew(i,j-1) ...
                        - aea(i,j)*phinew(i+1,j) ...
                        - awe(i,j)*phinew(i-1,j) ...
                        - ap(i,j)*phinew(i,j);
      end
    end

    resmax(nit) = max(max(resTemp));

    phiold = phinew;
    phi = phinew;


    iterstore = nit;  % save number of iterations in global var
    % Break out of the while loop if maximum number of iterations is reached
    if nit == nitmax;
      break
    end

  end  % end while loop
```

```
  if nit == nitmax;
    fprintf('GS-SOR [w=%1.1f] did not converge in %4.0f iterations.\n', ...
            omega, nitmax)
  else
    fprintf('GS-SOR [w=%1.1f] converged in %4.0f iterations.\n', ...
            omega, nit)
  end

end

%%% end of gs_sor
```