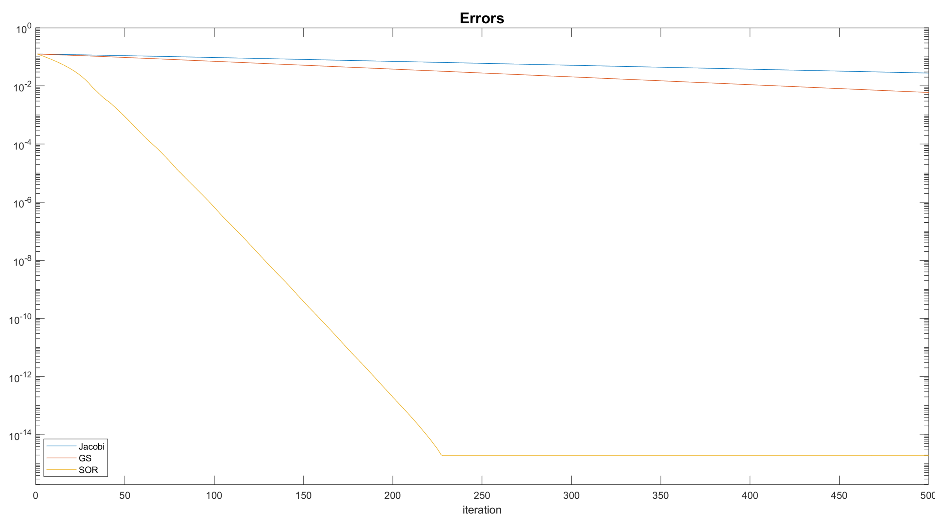
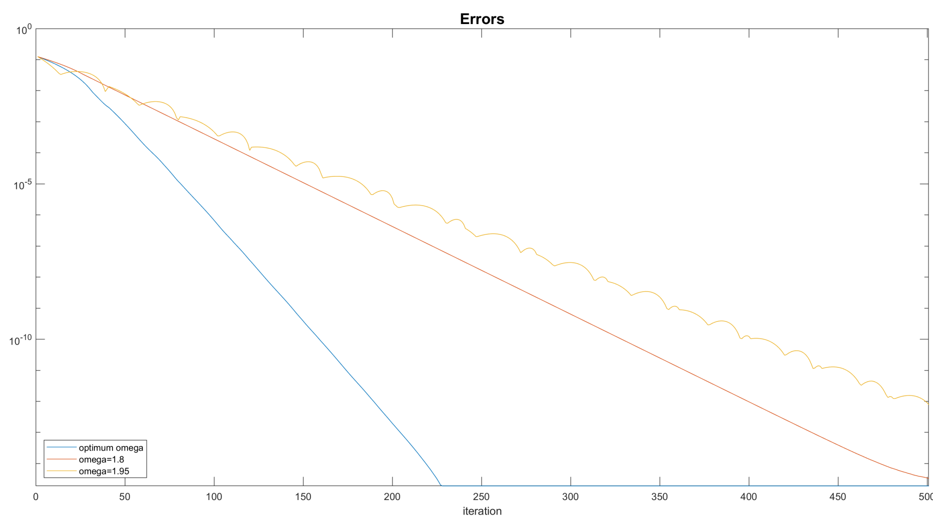
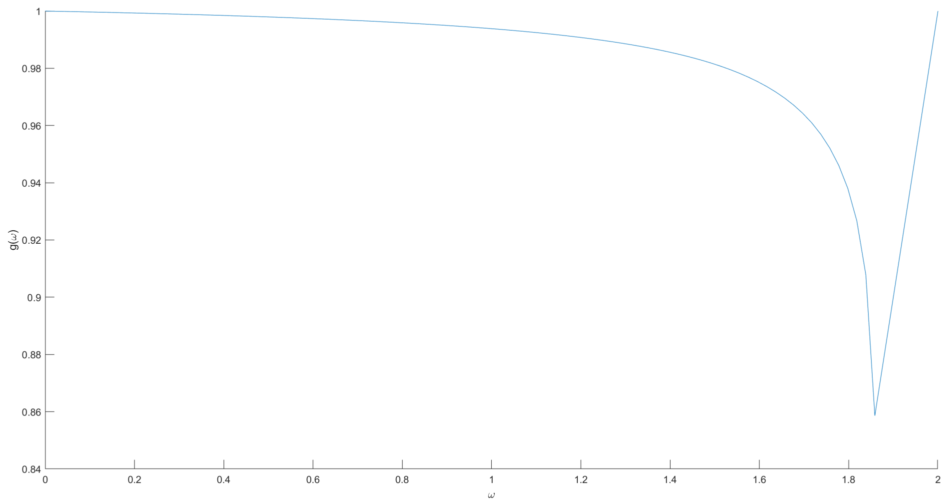
4.1 a. Run this program for each method and produce a plot similar to Figure 4.1.
b. The convergence behavior of SOR is very sensitive to the choice of ω. Try changing from the optimal ω to $\omega = 1.8$ or 1.95.
c. Let $g(\omega) = \rho(G(\omega))$ be the spectral radius of the iteration matrix G for a given value of ω. Write a program to produce a plot of $g(\omega)$ for $0 \le \omega \le 2$.
d. Try this computationally and observe that it does not work well. Explain what is wrong with this and derive the correct expression (4.24).

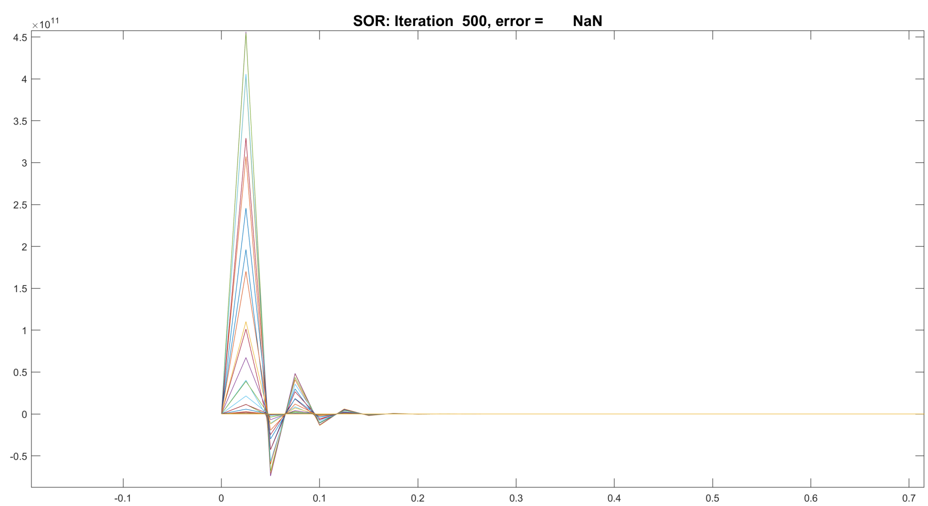
Solution.







We see by the figure below that this method clearly does not work well. Given that the true solution was supposed to be a second order polynomial, this naïve SOR method does not hold.



Now, starting from equations (2.22), we have

$$u_i^{GS} = \frac{1}{2}(u_{i-1}^{[k+1]} + u_{i+1}^{[k]} - h^2 f_i)$$

$$u_i^{[k+1]} = u_i^{[k]} + \omega(u_i^{GS} - u_i^{[k]})$$

Hence,

$$u_i^{[k+1]} = u_i^{[k]} + \omega(\frac{1}{2}((u_{i-1}^{[k+1]} + u_{i+1}^{[k]} - h^2 f_i) - u_i^{[k]})) =$$

$$u_i^{[k]} - \omega u_i^{[k]} + \frac{\omega}{2}(u_{i-1}^{[k+1]} + u_{i+1}^{[k]} - h^2 f_i) = (1-\omega)u_i^{[k]} + \frac{\omega}{2}(u_{i-1}^{[k+1]} + u_{i+1}^{[k]} - h^2 f_i)$$

So. we have that

$$u_i^{[k+1]} = (1-\omega)Du^{[k]} + \frac{\omega}{2}(Lu^{[k+1]} + Uu^{[k]} - h^2 f_i)$$

$$\omega\frac{1}{\omega}((1-\omega)D + \frac{\omega}{2}U)u^{[k]} + \frac{\omega}{2}Lu^{[k+1]} - \frac{h^2\omega}{2}f_i$$

Therefore, we see that

$$M = \frac{1}{\omega}(D - \omega L) \text{ and } N = \frac{1}{\omega}((1-\omega)D + \omega U)$$

4.2 a. We can also define a backwards Gauss-Seidel method by setting

$$u_i^{[k+1]} = \frac{1}{2}(u_{i-1}^{[k]} + u_{i+1}^{[k+1]} - h^2 f_i), \text{for } i = m, m-1, m-2, ..., 1$$

Show that this is a matrix splitting method of the type described in Section 4.2 with $M = D - U$ and $N = L$.
b. Implement this method in *iter_bvp_Asplit.m* and observe that it converges at the same rate as forward Gauss-Siedel for this problem.

Solution.
Let $M = D - U$ and $N = L$. Then, plugging these into

$$Mu_i^{[k+1]} = Nu_i^{[k]} + f$$

yields

$$(D - U)u_i^{[k+1]} = Lu_i^{[k]} + f_i$$

Hence,

$$Du_i^{[k+1]} - Uu_i^{[k+1]} = Lu_i^{[k]} + f_i$$

So,

$$Du_i^{[k+1]} = Uu_i^{[k+1]} + Lu_i^{[k]} + f_i$$

and therefore

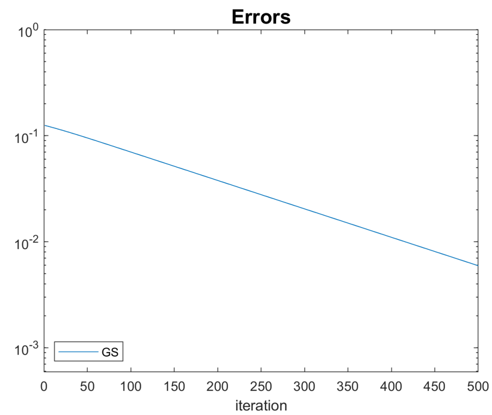
$$\frac{-2}{h^2} Iu_i^{[k+1]} = \frac{-1}{h^2}(u_{i+1}^{[k+1]} + u_{i-1}^{[k]}) + f_i$$

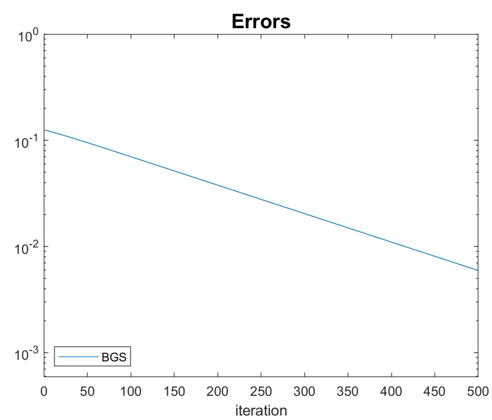
Thus,

$$u_i^{[k+1]} = \frac{1}{2}(u_{i+1}^{[k+1]} + u_{i-1}^{[k]} - h^2 f_i)$$

which is the type of splitting method desired.

It is obviuos by the figures below that the the rate of convergence for the forward and the backward Gauss-Siedel methods is the same.


Errors
10^0
10^-1
10^-2
10^-3
GS
0 50 100 150 200 250 300 350 400 450 500
iteration


Errors
10^0
10^-1
10^-2
10^-3
BGS
0 50 100 150 200 250 300 350 400 450 500
iteration

```
% Solve 1d BVP obtained by discretizing u''(x) = f(x)
% with Dirichlet boundary condtions  u(0) = alpha, u(1) = beta
% Discretized using centered difference.
%
% Compare matrix splitting methods:  Jacobi, Gauss-Seidel, and SOR
%
% From  http://www.amath.washington.edu/¬rjl/fdmbook/  (2007)

%4.1.a. We iterate through each of the different techinques and
%       create plots to mirror illustration 4.1

method = 'SOR';    % 'Jacobi' or 'GS' or 'SOR'
nplot = 10;           % iterate u is plotted every nplot iterations
maxiter = 500;         % number of iterations to take

m = 39;
ax = 0;
bx = 1;
alpha = 0;
beta = 0;
f = @(x) ones(size(x));   % f(x) = 1

h = (bx-ax) / (m+1);
x = linspace(ax, bx, m+2)';

% determine exact solution to linear system by setting up
% system Au = f and solving with backslash:
e = ones(m+2,1);

A = 1/h^2 * spdiags([e -2*e e], [-1 0 1], m+2, m+2);
A(1,1:2) = [1 0];
A(m+2,m+1:m+2) = [0 1];
rhs = f(x);
rhs(1) = alpha;
rhs(m+2) = beta;
ustar = A\rhs;


% Decompose A = DA - LA - UA:
DA = diag(diag(A));    % diagonal part of A
LA = DA - tril(A);     % strictly lower triangular part of A (negated)
UA = DA - triu(A);     % strictly upper triangular part of A (negated)

% set up iteration matrix:
switch method
   case 'Jacobi'
      M = DA;
      N = LA + UA;
   case 'GS'
      M = DA - LA;
      N = UA;
% Added this case to impliment the backwards GS method
   case 'BGS'
```

```
54         M = DA - UA;
55         N = LA;
56      case 'SOR'
57         % use optimal omega for this problem:
58
59   %4.1.b. iterating the SOR method through different omega values ...
         commented
60   %        out below
61
62         omega = 2 / (1 + sin(pi*h));
63         %omega = 1.8;
64         %omega = 1.95;
65         M = 1/omega * (DA - omega*LA);
66         N = 1/omega * ((1-omega)*DA + omega*UA);
67    end
68
69
70   u = zeros(size(x));    % initial guess for iteration
71
72   figure(1)
73   clf
74   plot(x,ustar,'r')
75   hold on
76   plot(x,u)
77
78   error = nan(maxiter+1,1);
79   error(1) = max(abs(u-ustar));
80
81
82   % Iteration:
83   % ----------
84
85   %Bad iteration method for the SOR method(commented out for use of ...
         actual code)
86   %To run the bad iteration, uncomment lines 90-92 and comment out lines
87   %94-96
88
89   %for iter=1:maxiter
90   %    uGS = (DA - LA) \ (UA*u + rhs);
91   %    u = u + omega * (uGS - u);
92
93   for iter=1:maxiter
94
95      u = M \ (N*u + rhs);
96
97      error(iter+1) = max(abs(u-ustar));
98      if mod(iter,nplot)==0
99         % plot u every nplot iterations
100        plot(x,u)
101        title(sprintf('%s: Iteration %4i, error = %9.3e',...
102               method,iter,error(iter+1)),'FontSize',15)
103        drawnow
104        pause(.1)
105     end
```

```
end

% plot errors vs. iteration:
figure(2)
semilogy(error)
axis([0 maxiter min(error)/10 1])
title('Errors','FontSize',15)
xlabel('iteration')
hold on
%legend({'GS','BGS'},'Location','southwest') used as a legend for the
%comparison grpahs in the command window.

% compute spectral radius of iteration matrix G:
G = M\N;
rhoG = max(abs(eig(full(G))));
disp(sprintf('Spectral radius of G for %s is %5f',method,rhoG))
```

```
%Stephanie Klumpe
%4.1.c.
%Spectral radius iterated over omega values between 0 and 2
clear;
close all;
clc;
nplot = 10;           % iterate u is plotted every nplot iterations
maxiter = 500;         % number of iterations to take

m = 39;
ax = 0;
bx = 1;
alpha = 0;
beta = 0;
f = @(x) ones(size(x));   % f(x) = 1

h = (bx-ax) / (m+1);
x = linspace(ax, bx, m+2)';

% determine exact solution to linear system by setting up
% system Au = f and solving with backslash:
e = ones(m+2,1);

A = 1/h^2 * spdiags([e -2*e e], [-1 0 1], m+2, m+2);
A(1,1:2) = [1 0];
A(m+2,m+1:m+2) = [0 1];
rhs = f(x);
rhs(1) = alpha;
rhs(m+2) = beta;
ustar = A\rhs;


% Decompose A = DA - LA - UA:
DA = diag(diag(A));    % diagonal part of A
LA = DA - tril(A);     % strictly lower triangular part of A (negated)
UA = DA - triu(A);     % strictly upper triangular part of A (negated)
omega=linspace(1e-16,2);
for i=1:length(omega)
    M = 1/omega(i) * (DA - omega(i)*LA);
    N = 1/omega(i) * ((1-omega(i))*DA + omega(i)*UA);
    u = zeros(size(x));   % initial guess for iteration
    G = M\N;
    rhoG(i) = max(abs(eig(full(G))));
end
figure
axis([0 2 0.84 1])
xlabel('\omega')
ylabel('g(\omega)')
hold on;
plot(omega, rhoG);
hold off;
```