Andrew Garwood
Written HW 4

Problem 1

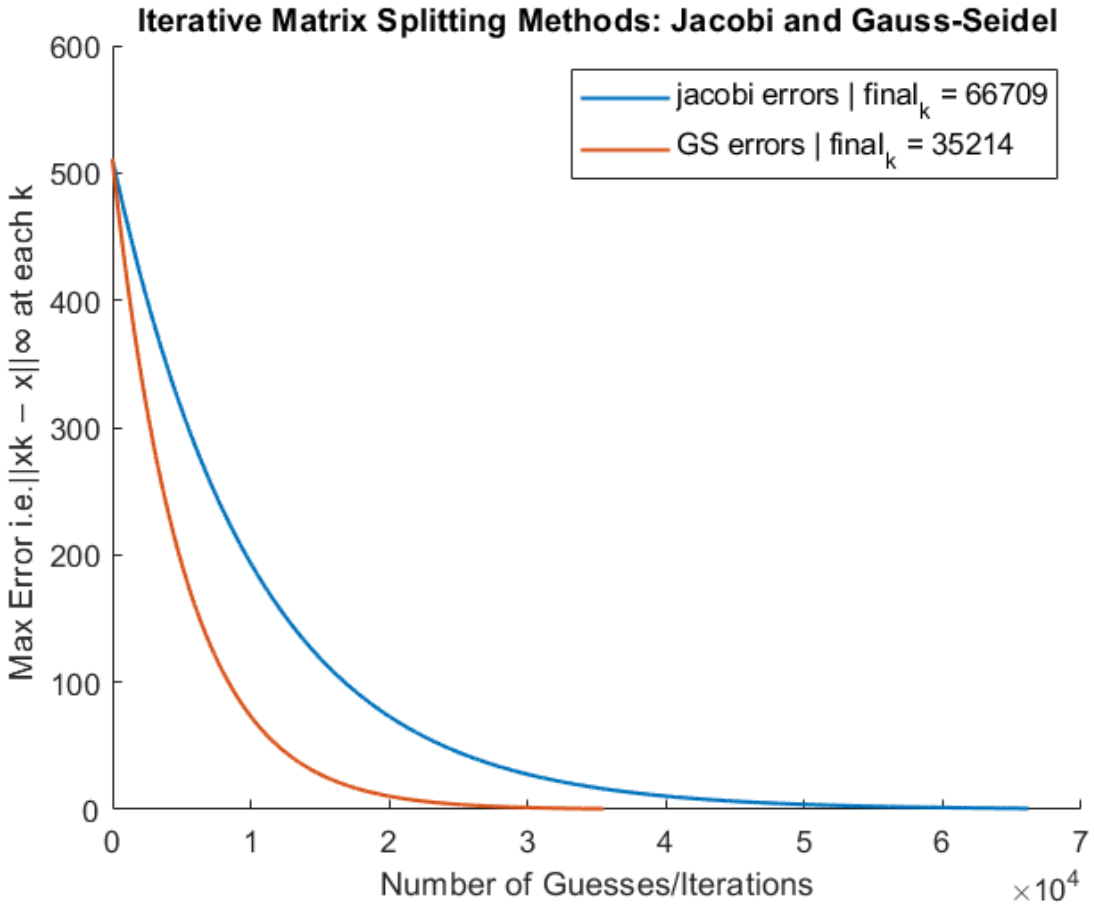
a)
```
% find ||Ax - b||
max_error = max(abs(A * x - b)); % = 6.7540e-13
% considering that the max error is extremely small, I believe it is
% reasonable to state that this error is zero
```

b, c, d)
```
% (b) Solve the system of equations Ax = b using the Jacobi method with a
% tolerance of 10−4 and an initial guess of all ones. At each guess xk,
% find the error between the guess and the true solution.
% In other words, find ||xk − x||∞ at each k.


% Solve for phi using Jacobi method.
% Jacobi Partition
P = diag(diag(A));
T = A - P;

% Get M Matrix
M = -P\T;

lambda = eig(M);
max_lambda  = max(abs(lambda))


% construct initial guess
phi0 = zeros(20, 1);
phi0(:, 1) = 1;
% max_lambda is less than 1, so no infinite while loop

% designate template for previous guess and new guess
new_phi = zeros(20, 1);
% Set previous guess to phi0
prev_phi = phi0;

tolerance = 1e-4; % initialize tolerance level
error = tolerance + 1; % ensure loop runs at least once
k = 1; % indexing variable
```

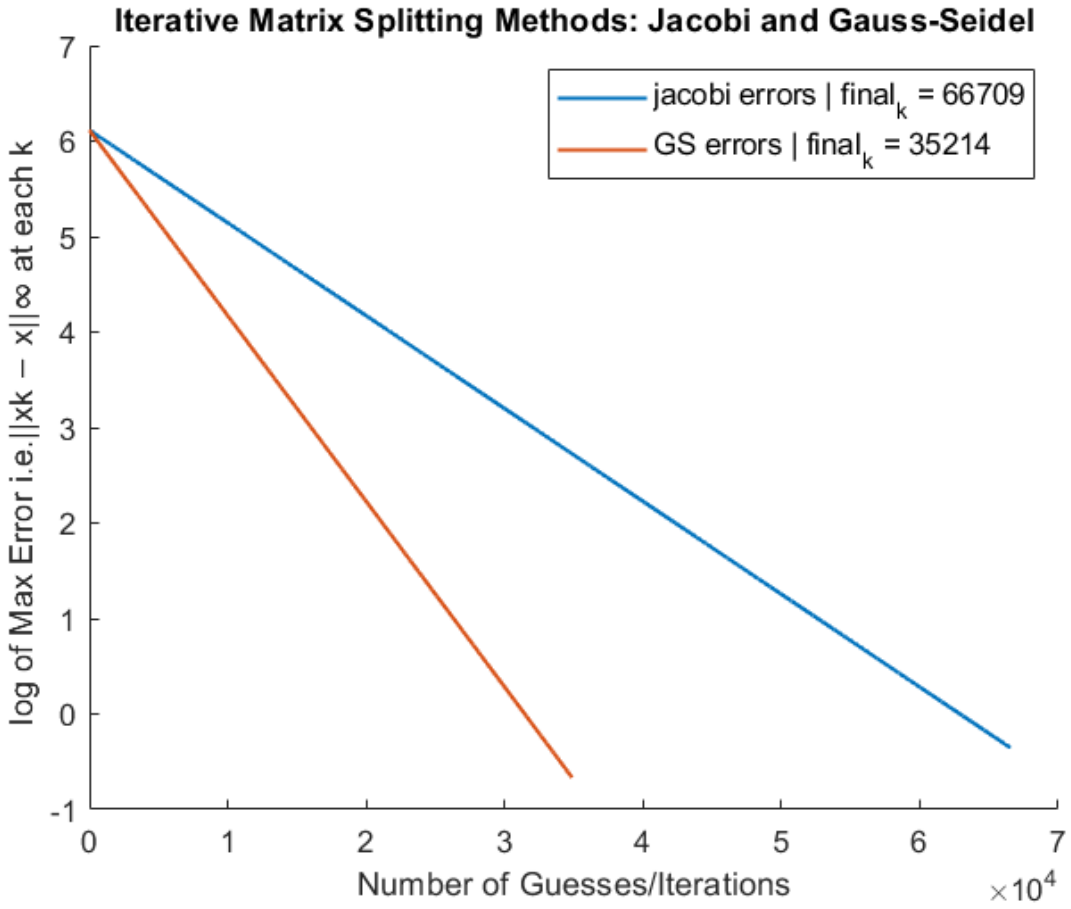
```
% At each guess Xk find error between guess and solution
true_solution = x;

% arbitrarily set size of guess error vector
jacobi_errors = zeros(1, 101);

% add first guess error measurement to vector
jacobi_errors(k) = max(abs(phi0 - true_solution));


if max_lambda < 1
   while error >= tolerance
      new_phi = P\(-T * prev_phi + b);
      error = max(abs(new_phi - prev_phi));

      k = k + 1; % increment/set current guess number

      jacobi_errors(k) = max(abs(new_phi - true_solution));


      prev_phi = new_phi; % set previous guess to new one to progress!!
   end
end

% (c) Repeat part (b) using the Gauss-Seidel method.

% Solve for phi using GS method.
% GS Partition
P = tril(A);
T = A - P;

% Get M Matrix
M = -P\T;

lambda = eig(M);
max_lambda  = max(abs(lambda));

% construct initial guess
phi0 = zeros(20, 1);
phi0(:, 1) = 1;
% max_lambda is less than 1, so no infinite while loop

% designate template for previous guess and new guess
```

```
new_phi = zeros(20, 1);
% Set previous guess to phi0
prev_phi = phi0;

tolerance = 1e-4; % initialize tolerance level
error = tolerance + 1; % ensure loop runs at least once
k = 1; % indexing variable

% At each guess Xk find error between guess and solution
true_solution = x;

% arbitrarily set size of guess error vector
GS_errors = zeros(1, 101);

% add first guess error measurement to vector
GS_errors(k) = max(abs(phi0 - true_solution));


if max_lambda < 1
   while error >= tolerance
      new_phi = P\(-T * prev_phi + b);
      error = max(abs(new_phi - prev_phi));

      k = k + 1; % increment/set current guess number

      GS_errors(k) = max(abs(new_phi - true_solution));


      prev_phi = new_phi; % set previous guess to new one to progress!!
   end
end

% Plot that ish
% x_jacobi = 0:length(jacobi_errors) - 1;
% x_GS = 0:length(GS_errors) - 1;
% figure()
% hold on
% semilogy(x_jacobi, log(jacobi_errors), 'DisplayName', 'jacobi errors | final_k = 66709',
'LineWidth', 1.25)
% semilogy(x_GS, log(GS_errors), 'DisplayName', 'GS errors | final_k = 35214', 'LineWidth',
1.25)
%
% hold off
% xlabel('Number of Guesses/Iterations')
```

```
% ylabel('log of Max Error i.e.||xk − x||∞ at each k ')
% title('Iterative Matrix Splitting Methods: Jacobi and Gauss-Seidel')
% % legend('FontSize', 10, 'Position', [0.5 0.5 0.1 0.2])
% legend('FontSize', 10)
```



e, f)



% (f) What do the slopes of the lines from part (e) represent? Can you relate them
% to a quantity from lecture?

% slope = (log(method_error_k1) - log(method_error_k2)) / (k1 - k2)

% Jacobi log error slope = -0.00009721799004
% Jacobi max abs lambda = 0.999902786604301
% Jacobi max abs lambda + abs(jacobi_log_error) = 1.000000004594341

% GS log error slope = -0.0001944366197
% GS max abs Lambda = 0.999805582659051
% GS max abs lambda + abs(gs_log_error) = 1.000000019278751

% I grow tired, my apologies.

Problem 2

```
%SOR method
% The coding homework asked you to use ω = 1.5 to solve the system A114φ = ρ.
% You should have found that SOR (with ω = 1.5) was slightly faster than
% GaussSeidel. In this problem, we will try to find an optimal value of ω
% and see how that impacts the speed of our splitting method.



% (a) For every value of ω between ω = 1 and ω = 1.999 in increments of 0.001,
% calculate M = −P^−1 * T using the corresponding P and T from equation (1)
% and then find the absolute value of the largest eigenvalue λ1 of this matrix M.
% Make a plot of the absolute value of λ1 versus ω. (That is, ω should be on the
% x-axis and the absolute value of the largest eigenvalue of M should be on the
% y-axis.)

% A = diagonal + upperTri + lowerTri
% Then P = (1 / omega) * D + L, T = ((omega - 1) / omega) * D + U
% omega between 1 and 2. 1 => GS

D = diag(diag(A));
L = tril(A) - D;
U = triu(A) - D;

max_lambdas = zeros(1, length(1:.001:1.999));
i = 1;
for omega = 1:.001:1.999
   P = (1 / omega) * D + L;
   T = ((omega - 1) / omega) * D + U;

   M = -P\T;
   lambda = eig(M);
   max_lambdas(i) = max(abs(lambda));
   i = i + 1;
end

x = 1:.001:1.999;
figure()
hold on
plot(x, max_lambdas, 'DisplayName', 'Max_eigenvals', 'LineWidth', 1.25)

hold off
xlabel('omega 1 to 1.999, intervals of 0.001')
```

```
ylabel('abs max(eigenvalue) of M')
title('SOR method, finding optimal omega')
legend('FontSize', 10, 'Position', [0.5 0.5 0.1 0.2])
% legend('FontSize', 10)

% c
% optimal_omega approx 1.973
% min(max_lambdas) = 0.973

% d
% SOR with optimal omega took 461 guesses/iterations

% maximum error was  8.142494574814307e-06 !!!

% This is a lot faster than the Jacobi And Gauss-Seidel Methods
% Jacobi took 10348, Gauss_seidel took 6104

% (e) In lecture, we mentioned that calculating eigenvalues was a fairly slow process.
% Do you think that the speed increase over Jacobi or Gauss-Seidel was worth
% the time spent finding an optimal ω? Would your answer change if we had
% to solve many systems of equations with the same matrix A? Explain your
% reasoning

% For a small matrix, it would be worth it, however, because the
% eig(Matrix) function is O(n^3) i do not think it would be worthwhile to
% find an optimal omega in applications with huge amounts of data. . i.e.
% large matrices

% For the same matrix A? If I am reading the question right, then you mean
% the A that we read in at the beginning of this question.  Then yes I think
% it would be worth it because A is relatively small  so we would be
% spending marginally more time to find a more precise answer; however, as
% I stated above, if A is arbitrarily large, then  I do not think it would
% be worthwhile.
```