Assignment 6 Solutions

Aaron Cahn
University of Wisconsin-Madison
cahn@cs.wisc.edu

May 8, 2015

1 Solutions

1.1 Question 1

Listing 1: Matlab Commands

```
% Qustion 1

% generate the A matrix with intervals of eigenvalues
A=generate_interval_matrix(100,[-3 4 1],1);
b=rand(100,1);

 fig=figure;
 for n=1:10-1
     % run my_gmres to obtain Q and S
     [Q,S]=my_gmres(A,b,n);

     % compare spectrums
     specA=eig(A);
     specS=eig(S);

     % plot the eigenvalues of A and Sn
     subplot(3,3,n)
     plot(real(specA),zeros(size(specA,1)),'bo', ...
          real(specS), zeros(size(specS,1)),'rx')
     title(sprintf('n=%d',n))
 end
```

The code above was run on various incarnations of the matrix A. For each expirement the matrix A is 100x100 in dimension. Each expirement varied the number of intervals the eigenvalues of A were centered around and how large this interval was. The number of intervals ranged from 1 to 5.

The conclusion of the expirements was very clear. The speed of convergence of $\sigma(S_n)$ towards $\sigma(A)$ was proportional to the number of intervals the eigenvalues were centered around. For example, consider the plots below. This expirement was run with three intervals centered at -3, 4, and 1. From Figure 1 we see that after three iterations the $\sigma(S_n)$ is starts to align well with the $\sigma(A)$ (*i.e.,* the three eigenvalues of S_n align with the three intervals). This is true in general. Specifically the convergence happens at a rate of $\frac{1}{|I|}$ where $|I|$ is the number of intervals the eigenvalues of A are centered around.

Listing 2: Matlab Commands

```
% runs the gmres algorithm for n iterations
function [Q,S,xtil] = my_gmres(A,b,n)
    % track Q, S, and X's
    Q = []; S = []; X={};

    % pre-process steps
    b1 = norm(b);
    Q(:,1) = b/norm(b);

    % run n iterations of gmres
    for i=1:n
        v=A*Q(:,i);
        S(:,i)=Q'*v;
        v=v-Q(:,1:i)*S(:,i);
        Q(:,i+1)=v/norm(v);
        S(i+1,i)=norm(v);
        b1=[b1;0];
        X{i}=Q(:,1:i)*(S\b1);
    end
    % return Sn and final x
    xtil = X{n};
    S=S(1:end-1,:);
end
```

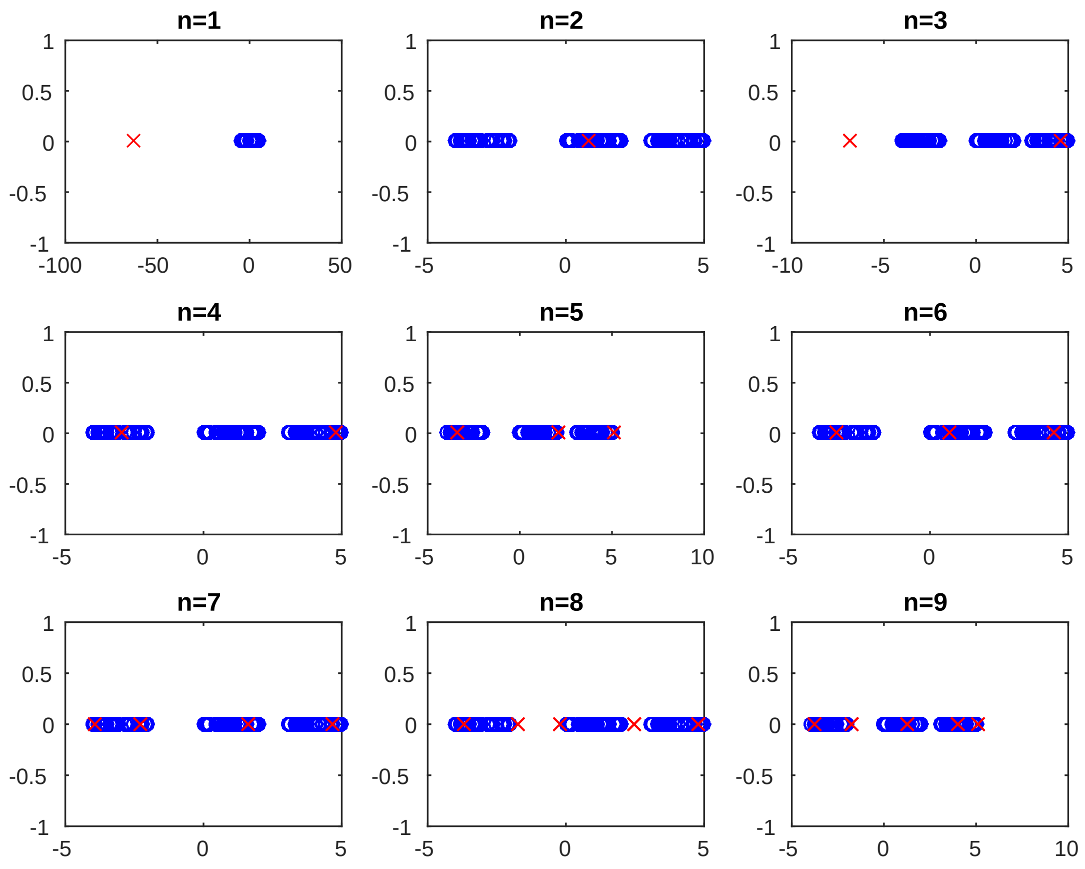

Figure 1: Plots of $\sigma(S_n)$ (red) and $\sigma(A)$ (blue). Plots correspond to stopping GMRES after n iterations.

1.2 Question 2

Listing 3: Matlab Commands

```
% Part (ii)
n: 5 error: 0.104672
n: 10 error: 0.000154
n: 20 error: 0.000000
n: 90 error: 0.000000
```

Listing 4: Matlab Commands

```
% Question 2

% Generate A from generate_interval_matrix
m = 100; n = [4 5 10]; e = 1;
A = generate_interval_matrix(m,n,e);

% Generate a random b
b = rand(100,1);

% Find the actual x (partt(i))
x = A\b

% Use function from Q1 (part(ii))
n = [5 10 20 90];
errs = []; Q = {}; S = {};
for i=1:4
    % gather data
    [Q{i},S{i},xtil]=my_gmres(A,b,n(i));
    errs = [errs norm(A*xtil-b)];
    disp(sprintf('n: %d error: %f',n(i),norm(A*xtil-b)));
end
plot(n,errs,'b--o')
```

As with the expirement from Question 1 the number of intervals was kept constant at 3 and the interval ranges were varied from 0.5 to 2. The results obtained reflect the sentiments expressed above. The convergence (relative error decreased) of $\sigma(S_n)$ to $\sigma(A)$ occurred at a rate of $\frac{1}{|I|}$. Therefore after ten iterations the error was extremely small (0.000154) and became almost negligible after that. After ninety iterations the $\sigma(S_n)$ was essentially the same as the $\sigma(A)$ and after one hundred there was complete convergence. However, it makes little sense to iterate this long as the cost trade off of using an indirect method is lost (*i.e.,* the complexity is $O(m^3)$. Figure 2 demonstrates the reduction in error at the different values of n.

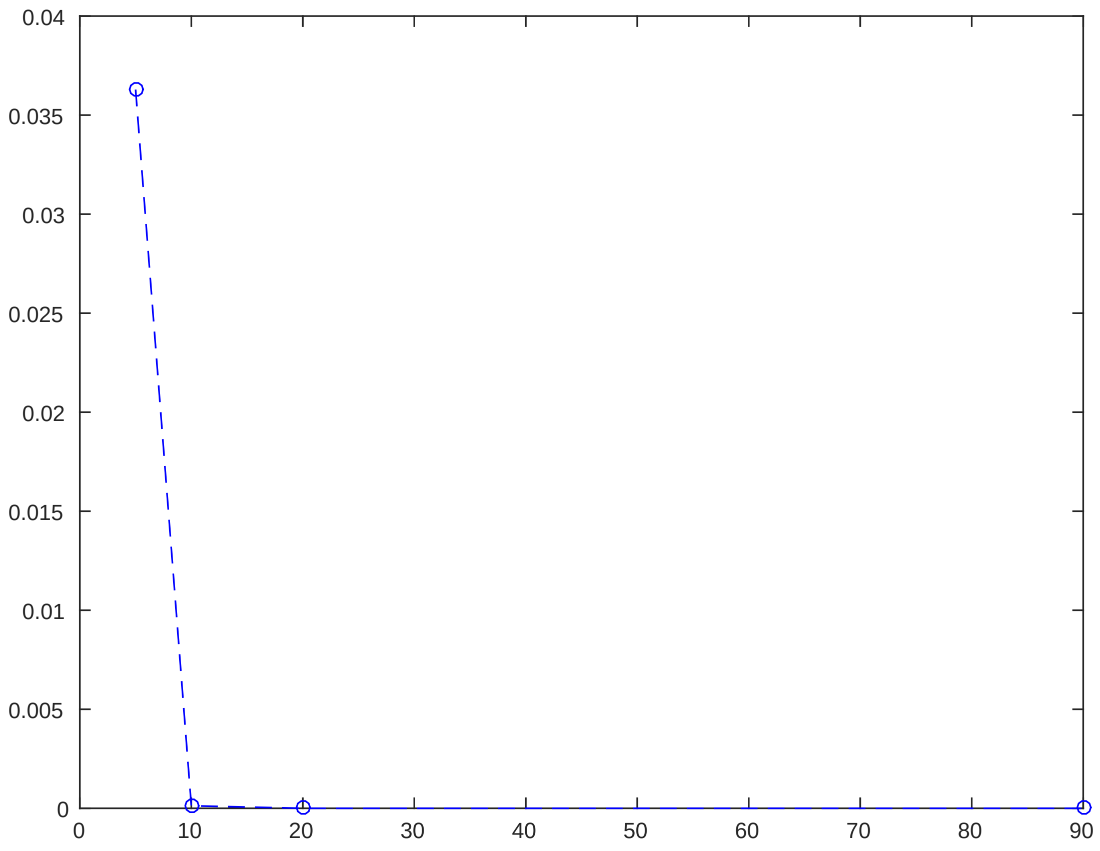

Figure 2: Plot showing the relative error $||A\tilde{x} - b||_2$ for different values of n.

1.3 Question 3

Listing 5: Matlab Commands

```
% Generate B
B=rand(30,10);
D=rand(10,10);
 for k=100:100:999
       % Generate C
       C=D;
       C(1:2,:)=C(1:2,:)*k;
       % Generate A
       A=B*C;
       [U,S,V] = svd(A);
end
```

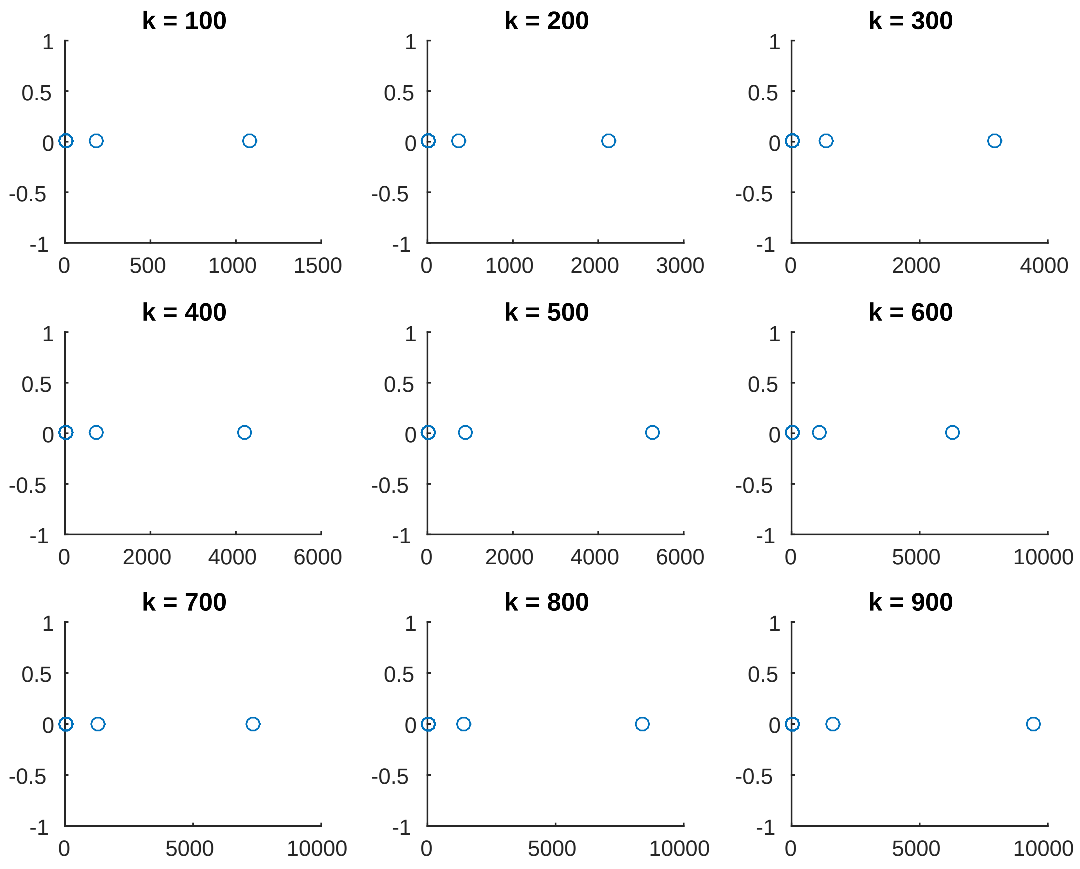

Figure 3: Plots of singular values of A at various k. Note, the first two singular values are much larger than the rest ($k * 10$ and k times larger respectively).

The experiment setup was similar to the two expirements above, except the value of k was the variable in this case. Figure 3 shows one of these expirements. It is clear from Figure 3 that the first two singular values of A are much larger than the other singular values of A. Specifically, the first two singular values are roughly $k * 10$ times and k times larger than the other singular values. This makes intuitive sense as A is built out of an underlying matrix where the first two rows are k times larger than the other rows. Therefore, the first two singular values and left singular vectors of A capture the structure of A quite well.

Listing 6: Matlab Commands

```
function [A] = generate_interval_matrix(m,n,e)
    I = [];
    for i=1:size(n,2)
        % generate random values in the interval
        I = [I (((n(i)+e)-(n(i)-e)).* ...
            rand(round(m/size(n,2)),1)+(n(i)-e))'];
    end
    % m/n has remainder (add those values)
    if size(I,2) < m
        I = [I (((n(i)+e)-(n(i)-e)).* ...
            rand(m-size(I,2),1)+(n(i)-e))'];
    end
    % generate A with eigenvalues from I
    D = diag(sort(I,'descend'));
    P = rand(m);
    A = P*D*inv(P);
end
```