Assignment 5 Solutions

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

April 15, 2015

1 Solutions

1.1 Question 1

1.1.1 Part A

Method 1: LU factor $A - sI$ for $s = 0$ and $s = 1$ and inspecting the number of negative values on the diagonal of U.

Case 1: $s = 0$,

$$A_0 = \begin{pmatrix} -1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix} \tag{1}$$

$$L_0 = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, U_0 = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 1.5 \end{bmatrix} \tag{2}$$

$$L_1 = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0.5 & 1 \end{bmatrix}, U_1 = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 1.5 \end{bmatrix} \tag{3}$$

Therefore, when we inpsect the diagonal of U_1 we see that there is one negative value. This means that there is one eigenvalue in A that is less than zero.

Case 2: $s = 1$,

$$A_1 = \begin{pmatrix} -2 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \tag{4}$$

$$L_0 = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, U_0 = \begin{bmatrix} -2 & 1 & 0 \\ 0 & 0.5 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{5}$$

$$L_1 = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}, U_1 = \begin{bmatrix} -2 & 1 & 0 \\ 0 & 0.5 & 1 \\ 0 & 0 & -1 \end{bmatrix} \tag{6}$$

Therefore, when we inpsect the diagonal of U_1 we see that there are two negative values. This means that there are two eigenvalues in A that are less than one. Utilizing the knowledge from both cases we have found that there is one eigenvalue in the range $[0, 1]$.

Method 2: Compute the number of sign changes in the sequence of main principal minors of $A - sI$ for $s = 0$ and $s = 1$.

Listing 1: Matlab Commands

```
function [d] = mpm(A)
    n = size(A);
    d = []; d(1) = 1;
    d(2) = A(1,1);
    for i=2:n
        d(i+1)=A(i,i)*d(i)-A(i,i-1)^2*d(i-1);
    end
end
```

Listing 2: Matlab Commands

```
% case 1: A-0*I
mpm(A)
ans =
    [1]    [-1]    [-2]    [-3]

% case 2: A-1*I
mpm(A-eye(3))
ans =
    [1]    [-2]    [-1]    [1]
```

We see that in case 1 there is one sign change in the sequence of main principal minors and that in case 2 there are two sign changes in the sequence of main principal minors. Therefore, as with method 1 we have shown there is a single eigenvalue in the range $[0, 1]$.

1.1.2 Part B

The best estimate count of arthmetic operations for the two methods was 116 flops for method 1 and 36 flops for method 2. Therefore, it would seem that method 2 is more efficient for tri-diagonal matrices and it makes intuitive sense that this gap will become larger as the matrices grow in size (*i.e.,* method 2 will look better and better relative to method 1).

1.1.3 Part C

The code below generates 100 symmetric tridiagonal matrices of size 50x50 and runs both Method 1 and Method 2 from part A.

Listing 3: Matlab Commands

```
lu_times = []; mpm_times = []; m = 50;
for i=1:100
    % create the tridiagonal matrix
    B=rand(m); A=B*B';
    A=diag(diag(A,-1),-1)+diag(diag(A))+diag(diag(A,1),1);

    % Method 1: LU factorization
    tic;
    % case 1: A-0*I
    [L,U] = lu_sym(A);
    pevals1 = length(find(diag(U)>0));
    % case 2: A-1*I
    [L,U] = lu_sym(A-eye(m));
    pevals2 = length(find(diag(U)>0));
    % eigen values in the range
    pevals = pevals1-pevals2;
    lu_times(i) = toc;

    % Method 2: MPM sequences
    tic;
    % case 1: A-0*I
    [d] = mpm(A);
    % count the sign changes
    pevals1 = sign_changes(d);
    % case 2: A-1*I
    [d] = mpm(A-eye(m));
    % count the sign changes
    pevals2 = sign_changes(d);
    % eigen values in the range
    pevals = pevals2-pevals1;
    mpm_times(i) = toc;

end

avg_lu=sum(lu_times)/m
avg_mpm=sum(mpm_times)/m
```

The results of the code snippet above are presented below. It is clear as stated in part B that Method 2 dominates Method 1 in terms of running speed. This contrast grows as the size of the symmetric tridiagonal matrix grows.

Listing 4: Matlab Commands

```
q1_partC

avg_lu =

    0.1649


avg_mpm =

   7.5590e-04
```

1.2 Question 2

1.2.1 Part A

Since the Gerschgorin's disks are pairwise disjoint we know that each disk must contain one eigenvalue. Therefore the cardinality of $\sigma(A) = n$. This means that the algebraic multiplicity m_a of each eigenvalue is $m_a = 1$. We know that the geometric multiplicity m_g follows the rule $m_g \leq m_a$ for all eigenvalues of A. Therefore, since $m_a = 1$ we can say that $m_g = 1$.

1.2.2 Part B

Leon is correct because as stated above $m_g = m_a$ for eigenvalues of A.

1.2.3 Part C

Nina is correct because there are n eigenvalues; one per disk. If there was a complex eigenvalue it's complex conjugate would be an eigenvalue as well. This would require two eigenvalues to be loacted in a single disk. However this would contradict what we have shown in (a). Therefore all eigenvalues of A are real.

1.2.4 Part D

Elvis is not correct. The problem is that there can be two distinct eigenvalues in the real that have the same magnitude. For example, suppose a matrix B as eigenvalues $\lambda_1 = 6, \lambda_2 = -6$. It is clear, $|\lambda_1| = |\lambda_2|$. In this case the power method will not converge because there is no dominant eigenvalue.

1.3 Question 3

Listing 5: Matlab Commands

```
m = length(A);
[P,D] = eig(A);
D= diag(D);

% Step 1: Power Method to find the first two dominant eigenvalues

% most dominant eigenvalue
[eval1,evec1,err1] = power_method(A,rand(m,1),10^-2);
fprintf('1st dom from pm: %f\ntrue 1st dom: %f\n\n',eval1,D(1));
% second most dominant eigenvalue (deflate eval1 from A)
e = [1 zeros(1,m-1)]'; w = e-evec1; w = w/norm(w);
H = eye(m)-2*w*w';
A1 = H*A*H; tilA = A1(2:end,2:end);
[eval2,evec2,err2] = power_method(tilA,rand(m-1,1),10^-2);
fprintf('2nd dom from pm: %f\ntrue 2nd dom: %f\n\n',eval2,D(2));

% Step 2: Inverse Power Method to find the first two least dominant evals

% least dominant eigenvalue
[eval1,evec1,err1] = inv_power_method(A,rand(m,1),10^-2);
fprintf('1st least dom from pm: %f\ntrue 1st least dom: %f\n\n',eval1,D(m))
% second least dominant eigenvalue (deflate eval1 from A)
e = [1 zeros(1,m-1)]'; w = e-evec1; w = w/norm(w);
H = eye(m)-2*w*w';
A1 = H*A*H; tilA = A1(2:end,2:end);
[eval2,evec2,err2] = inv_power_method(tilA,rand(m-1,1),10^-2);
fprintf('2nd least dom from pm: %f\ntrue 2nd least dom: %f\n\n',eval2,D(m-
```

Listing 6: Matlab Commands

```
q3
1st dom from pm: 6.112115
true 1st dom: 6.116471

2nd dom from pm: -1.288738
true 2nd dom: -1.286597

1st least dom from pm: -0.115228
true 1st least dom: -0.115301
```

```
2nd least dom from pm: -0.342898
true 2nd least dom: -0.344068

diary off
```

1.4 Question 4

1.4.1 Part A

1.4.2 Part B