Erin Obermayer
Computational Math - HW1

Problem 1:

1. The graph does not converge. It oscillates in an odd behavior approximately between numbers slightly greater than 0 and -1.
2. Plot (see code attached).
3. After checking the eigenvalues of A using the eig MATLAB function, I see that all of the eigenvalues are imaginary. I would guess that having imaginary eigenvalues causes the oscillatory behavior of the algorithm. The Power Iteration Algorithm probably doesn't work with imaginary numbers.

Problem 2:

By Hand: (on paper)

HW 1

1) $A = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 2 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \end{bmatrix}$ $A^T = \begin{bmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 \\ -2 & 0 & 0 & 0 \end{bmatrix}$

$$A^TA = \begin{bmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 \\ -2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & -2 \\ 2 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0.4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

$$(A^TA)^T = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0.4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} = A^TA$$

So, A^TA is symmetric. ✓

2) Let B be a $n \times n$ (square) matrix.

Consider $(B^TB)^T = B^T(B^T)^T$
$= B^TB.$

By defn, B^TB is symmetric.

Consider $(BB^T)^T = (B^T)^T B^T$
$= BB^T.$

By defn, BB^T is symmetric. $\square$

→ In general, for any square matrix B, it is NOT true that $B^TB = BB^T$. Counterexample:

Let $B = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$. Then $B^T = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$

$B^TB = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix}$ and

$BB^T = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 2 \\ 2 & 1 \end{bmatrix}$.

Thus, $B^TB \neq BB^T$.

3. I think the spectrum of BB^T and B^TB is the same. I am guessing that the symmetry of these matrices has something to do with this.

Using MATLAB:

1. (See code attached).
2. What I Observe
    a. Both converge to 4. This differs from Problem 1 because it actually converges to a real number rather than oscillating.
    b. This may suggest that although these two matrix productions use a matrix with imaginary eigenvalues (A), taking the products creates matrices with real eigenvalues. Also, since both converge to 4, that means both have a dominant eigenvalue of 4. This shows that they share at least one eigenvalue in common.

By Hand: (on paper)

1. <u>Thm</u>: The spectrum of AA^T and A^TA are the same.

<u>Proof</u>: Let A be a square matrix.

2. Assume λ is an eigenvalue of A^TA with corresponding eigenvector x, so:

$$(A^TA)x = \lambda x$$

3. We can left multiply by A on both sides:

$$A(A^TA)x = A\lambda x$$

Since λ is a scalar, we can pull it to the left on the RHS:

$$A(A^TA)x = \lambda Ax$$

4. Now, we can group terms:

$$AA^T(Ax) = \lambda(Ax)$$

We know Ax is a vector, call it v:

$$AA^Tv = \lambda v.$$

This shows that λ is an eigenvalue of AA^T with corresponding eigenvector v. Thus, the eigenvalues of A^TA and AA^T are equal. $\square$

→ The eigenvectors of A^TA and AA^T are <u>NOT</u> equal. From our proof, we see that for A^TA, λ has corresponding eigenvector x, while for AA^T, λ has corresponding eigenvector v. We can't assume that x and v are equal.

Problem 3:

The eigenvalues of C are 3,1, and 2 (given by eig MATLAB function).

Problem 3:

1.

μ	converges to / behavior
-5	1
-1	1
0	1
1.25	1
1.49	1 (w/ some converging oscillations)
1.5	oscillating btwn 0.95 and 1.05
1.51	2 (w/ converging oscillations)
2.49	2
2.5	oscillating btwn 1.9 and 2.2
2.51	3 (w/ increasing converging oscillations)
5	3

2. $\mu = 1.5$ and $\mu = 2.5$ don't seem to converge. These values are exactly between the actual eigenvalues of C. I think having μ exactly between 2 eigenvalues confuses the algorithm, so it doesn't know which λ to choose. Also, I noticed these values are equal to 2 of the diagonal entries in C. That means the subtraction $C - \mu I$ will produce some 0 values, which could create an issue.

3. In using this method to find all eigenvalues of a matrix, you could face difficulty in choosing μ values that work & don't break the code. This would be hard if you don't know the eigenvalues beforehand, but that is the point of the algorithm, so it does not seem like a practical method.