CS2810 Mathematics of Data Models	Gold
Spring 2019	February 20, 2019

Homework 05

Due: February 17, 9PM
Point total: 60

Instructions:

- Submit your PDF and/or .py file to Blackboard by the due date and time. Please do not zip your files together, as this interferes with Blackboard's preview functionality. Always show all your work, and for full credit, you must use the method that the problem instructs you to use (unless none is mentioned). Handwritten or typeset solutions are both acceptable, but unreadable submissions will be penalized. You may discuss problems with other students, but you may not write up solutions together, copy solutions from a common whiteboard, or otherwise share your written work or code. Do not use code or language that is copied from the Internet or other students; attribute the ideas *and* rephrase in your own words.

Problem 1 (15 points)

Given the linear recurrence $T(N) = 2T(N-1) + 3T(N-2)$, convert this recurrence to matrix form, find the eigenvalues of the matrix, and determine the asymptotic growth rate of the recurrence.

Solution: The matrix is $\begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix}$, the eigenvalues are 3 and -1, and the asymptotic growth rate is therefore $\Theta(3^N)$.

Problem 2 (15 points)

Find the diagonalization $A = PDP^{-1}$ of this matrix, then use the diagonalization to find its tenth power. $A = \begin{bmatrix} -1 & 1.5 \\ -4 & 4 \end{bmatrix}$

Solution: Solving for eigenvalues gives $\lambda = 2, 1$, and solving for eigenvectors gives $\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \end{bmatrix}$. So $P = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$, and solving for P^{-1} gives $\begin{bmatrix} -2 & 1.5 \\ 1 & -0.5 \end{bmatrix}$. So the diagonalization is $\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -2 & 3/2 \\ 1 & -1/2 \end{bmatrix}$.

The tenth power is then $PD^{10}P^{-1} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1024 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -2 & 3/2 \\ 1 & -1/2 \end{bmatrix} = \begin{bmatrix} -2045 & 1534.5 \\ -4092 & 3070 \end{bmatrix}$

Problem 3 (10 points)

For each matrix, find its condition number using both the ℓ_1 and ℓ_∞ norms. Then state whether it is the more well-conditioned or the more ill-conditioned of the two.

i. $\begin{bmatrix} 1 & 2 \\ 1 & 2.1 \end{bmatrix}$

Solution: The inverse is $\begin{bmatrix} 21 & -20 \\ -10 & 10 \end{bmatrix}$ The ℓ_1 norm condition number is $4.1 * 31 = 127.1$. The ℓ_∞ norm is $3.1 * 41 = 127.1$ as well. This is the more ill-conditioned of the two matrices.

ii. $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

Solution: The inverse is $\begin{bmatrix} -2 & 1 \\ 3/2 & -1/2 \end{bmatrix}$.

The ℓ_1 norm is $6 * 3 = 18$. The $\ell_i nfty$ norm is $7 * 2 = 14$. This is the more well-conditioned of the two matrices.

Problem 4 (10 points)

Show that if two matrices are similar to the same diagonal matrix, they are also similar to each other. (You may wish to use the fact that $(AB)^{-1} = B^{-1}A^{-1}$.)

Solution: If A and B are both similar to the same diagonal matrix D, then $A = PDP^{-1}$ and $B = QDQ^{-1}$ for some invertible matrices P and Q. Now, we can multiply both sides on the right by P and Q respectively to get $AP = PD$ and $BQ = QD$. Multiply on the left by P^{-1} and Q^{-1} to get $P^{-1}AP = D = Q^{-1}BQ$. Using $P^{-1}AP = Q^{-1}BQ$ we can multiply on the left by Q to get $QP^{-1}AP = BQ$; then multiply on the right by Q^{-1} to get $QP^{-1}APQ^{-1} = B$. We claim that $PQ^{-1} = (QP^{-1})^{-1}$ since $(QP^{-1})^{-1} = (P^{-1})^{-1}Q^{-1} = PQ^{-1}$ by the identity provided. Thus if $R = QP^{-1}$, then $RAR^{-1} = B$, and by the definition of similarity, A and B are similar.

Problem 5 (10 points)

Condition numbers might not be so bad if you have a fair amount of precision in your floating-point numbers to start. Here, you'll see how condition number interacts with Python's 64-bit floating point numbers.

Define a matrix $A = \begin{bmatrix} 1 & 1 + 1 \times 10^{-8} \\ 1 & 1 - 1 \times 10^{-8} \end{bmatrix}$

Using `numpy.linalg.cond()`, find the condition number of this matrix. Then multiply by $x = \begin{bmatrix} 100 \\ 200 \end{bmatrix}$

to get a vector $\vec{v}$, and finally, solve for $\vec{x_0}$ in $Ax_0 = \vec{v}$ by applying the inverse of A (`numpy.linalg.inv()`). How different is x_0 from x? If 64-bit floating point numbers generally have about 16 digits of precision, do these results support the rule that you lose about k digits of precision, if the condition number is 10^k? (Report the condition number, x_0, and the answers to the questions.)

Solution: The condition number is 200000000.70735085, according to my Python install. x_0 gets solved as `array([100., 199.99999809])`. That's a difference of about 0.000002, which means we only have about 8 digits of precision. This means we've lost 8, which isn't exactly the same as the log base 10 of the condition number, but close.