Numerical Analysis 1 – Class 6

Friday, February 17th, 2023

Subjects covered

- QR decomposition for square or rectangular matrices.
- Power iteration for finding dominant eigenvalue & corresponding eigenvector.
- Inverse iteration and Rayleigh iteration for finding other eigenvalue/vector pairs.
- Simultaneous iteration for finding all eigenvalues/vectors of a matrix.
- QR algorithm for finding multiple eigenvalues.
- Principal Component Analysis and relationship to SVD.

Readings

- "Gram-Schmidt orthogonalization and the QR decomposition", S. Brorson. (Available on Canvas).
- “Iterative Methods for Computing Eigenvalues and Eigenvectors”, Maysum Panju. (Linked on Canvas.)
- “Understanding the QR Algorithm”, D. S. Watkins. (Available on Canvas).
- “A Tutorial on Principal Component Analysis”, J. Shlens. (Linked on Canvas.)

Problems

Most of the following problems require you to write a program. For each program you write, please make sure you also write a test which validates your program. Please use Canvas to upload your submissions under the “Assignments” link for this problem set.

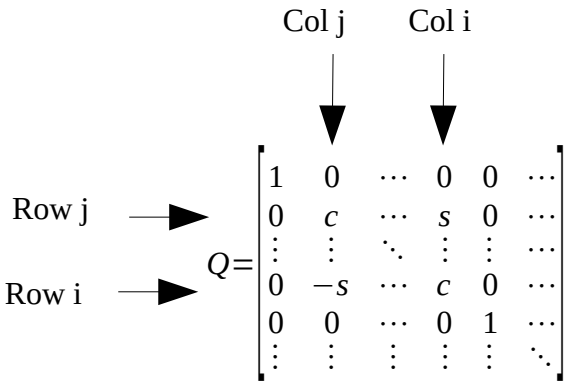
Problem 1 – Jacobi rotations

In this and the next problem you will derive a method to find the eigenvalues of a symmetric matrix A by iteratively applying similarity transformations which diagonalize A. A similarity transformation is an operation of form

$$Q^T A Q \rightarrow B$$

Here we will use a specially constructed orthogonal matrix Q which zeros out two off-diagonal elements of A. This operation is called a “Jacobi rotation”. In this problem you will write a program which returns Q.

The goal of this problem is: given a symmetric matrix A and a [row, column] index pair $[i, j]$, find a matrix of form



such that

$$Q^T A Q = B$$

and B has zeros at the positions $[i, j]$ and $[j, i]$. For example,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & -s & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & s & 0 & c & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{12} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{13} & a_{23} & a_{33} & a_{34} & a_{35} \\ a_{14} & a_{24} & a_{34} & a_{44} & a_{45} \\ a_{15} & a_{25} & a_{35} & a_{45} & a_{55} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & s & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -s & 0 & c & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & 0 & b_{25} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} \\ b_{41} & 0 & b_{43} & b_{44} & b_{45} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} \end{bmatrix}$$

The quantities c and s are the factors to be determined; we assume they obey $c^2 + s^2 = 1$.

Please do the following:

- Verify that Q is an orthogonal matrix given the assumption $c^2 + s^2 = 1$. This is a pencil-and-paper exercise. Hint: Show that you can create a block-diagonal matrix with the rotation part (the s and c elements) isolated into the upper left corner using a similarity transform involving permutation matrices. Then consider the properties of this block-diagonal matrix.
- Derive expressions for c, s by considering their desired action on a 2x2 matrix:

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} b_{11} & 0 \\ 0 & b_{22} \end{bmatrix}$$

  You will need to solve a quadratic equation to find expressions for c and s. This becomes important in the next step.
- Now that you know how to get c and s, write a program which takes as input a matrix A and an off-diagonal position $[i, j]$, and returns the Jacobi rotation matrix Q_{ij} . The desired Q_{ij} will zero out the elements of A at positions $[i, j]$ and $[j, i]$ via the similarity transform $Q_{ij}^T A Q_{ij}$. Note: When computing your values for c, s, make sure you take the solution of the quadratic equation with smaller magnitude. This is important for the Jacobi method

(problem 2).

- Your test program should generate random matrices and show that any desired off-diagonal element pair may be zeroed out using your matrix Q_{ij} .

Problem 2 – Jacobi method for finding eigenvalues

In the last problem you wrote a program which implemented the Jacobi rotation to zero out off-diagonal elements of an input matrix *A*. Since a Jacobi rotation $Q^T A Q$ is a similarity transform, and the matrix *Q* is orthogonal, the eigenvalues of the input matrix *A* are preserved. Therefore, you can use Jacobi rotations to diagonalize a matrix and thereby find its eigenvalues. This algorithm is called the "Jacobi method".

The algorithm looks like this:

1. Take as input a symmetric matrix A .
2. Find the largest magnitude (i.e. largest $|a_{ij}|$) off-diagonal element in A . If the element is small enough (i.e. less than some tolerance), the matrix is "sufficiently diagonal", and you can return the diagonal elements of A as the desired eigenvalues.
3. Otherwise, create the Jacobi matrix Q_{ij} required to zero out the elements a_{ij} and a_{ji} .
4. Apply the similarity transform to zero out the elements: $A_{n+1}=Q_{ij}^T A_n Q_{ij}$
5. Go back to step 2 and iterate again.

Please write a program to implement the Jacobi method. Feel free to test your implementation against Matlab's eig() function.

Problem 3 – PCA on a fun dataset

In this problem, you will perform PCA on a dataset. I have hidden an object into a larger dataset. The object is saved as a point cloud. Your goal is to write a program implementing PCA to examine the dataset, find its dimensionality, find the principal axes of the data, then do a projection to find the hidden object.

The dataset to process is called "Datafile.csv", available on Canvas. You may assume the data is organized in rows (as in class). That is, each row is a measurement series, with different samples along the columns. Note that I have added a small amount of noise to all data to simulate the effect of measurement error. Therefore, once you discover the figure, it may have a little bit of noise on it.

You don't need a test function for this problem. Rather, hand in a plot showing the object you found. Also, please answer the following questions:

- How many dimensions of the data hold useful information, and how many are noise?
- What object did I hide in the data?