Matlab Project 3: Methods for Eigenvalues

Due: Friday 10 April (5:00pm EDT)

Goal: Explore some important methods used to compute eigenvalues and eigenvectors

Introduction

In this class we have learned the standard method for computing eigenvalues (and eigenvectors) of an $n \times n$ matrix A "by hand":

- Find the characteristic polynomial $p(\lambda) = \det(A - \lambda I)$.
- Find the roots of $p(\lambda) = 0$: these are the eigenvalues $\lambda_1, \ldots, \lambda_n$.
- For each eigenvalue λ_i, solve $(A - \lambda_i I)\mathbf{v} = \mathbf{0}$ for corresponding eigenvector(s) $\mathbf{v}$.

This method is useful for matrices which are small enough (e.g., 2×2 or 3×3) or have enough structure (the nonzero entries are relatively few and have a simple pattern) to make the calculation practical. However, in other cases this approach just isn't practical, for the following reasons:

- Finding the characteristic polynomial (usually by cofactor expansion) requires far too much work (the operation count is proportional to n factorial, which is huge).
- Finding the roots of $p(\lambda) = 0$ is difficult: there are no formulas for roots of polynomials of degree $n > 4$, and the roots can be very sensitive to roundoff errors in numerical calculations.

So how are eigenvalues computed in practice? This project will explore two important numerical methods: the QR method and the power method. Work through the steps below and write all answers on the answer sheet as requested. *For this project you may work with one other student if you wish.*

Note: For this project you will construct and submit a *script* (i.e., an `M-file`) to do all calcuations. Instructions for creating, editing, and using such a script were given in the handout for Project #2: Ill-Conditioned Systems. More information is available in the MATLAB Help system: look under MATLAB—Programming Scripts and Functions—Scripts—Create Scripts. You can edit this script as desired as you work through this project, but when you're done, ensure that your script does all calculations and produces all output needed for this project—and nothing else. The script which you submit must run without modification to produce all output needed for this project.

Important: *Name your script using your Clarkson email name.*[1] For example, if your name was Bernie Sanders you would submit a file named `sandersb.m`. If you're working with a partner, submit only one script file (choose either name).

[1] This is needed so I don't receive fifty files named `project2.m`

Step 1: The MATLAB command `eig`

The standard MATLAB command for computing eigenvalues and eigenvectors is `eig`. Specifically, if A is a square matrix then

`[V,D] = eig(A)`

returns a matrix V with eigenvectors as its columns and a diagonal matrix D with the corresponding eigenvalues down the main diagonal.[2] These matrices are defined so that $AV = VD$; if the matrix A is diagonalizable (i.e., has n linearly independent eigenvectors) then V is invertible, and thus plays the role of the matrix P in the diagonalization $A = PDP^{-1}$ of section 5.3. Additional information for the `eig` command is available in the MATLAB help browser or by typing `help eig`.

To see how to use the `eig` command, first find the eigenvalues and eigenvectors of the matrix

$$A = \begin{bmatrix} 6 & -2 & 0 \\ -2 & 9 & 0 \\ 1 & 4 & 3 \end{bmatrix} \tag{1}$$

by hand. Then check these results using `eig`, and *write the eigenvalues and eigenvectors computed by `eig` on the answer sheet* (report all entries to four decimal places, which is the standard format produced by MATLAB). These should match what you found by hand: the eigenvalues should be the same, and the eigenvectors should be multiples of those you found by hand.[3]

Once you're sure you know how to use `eig`, use it to compute the eigenvalues and eigenvectors of the 7×7 matrix

$$B = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_6 \end{bmatrix} \tag{2}$$

where for the seven numbers a_0, a_1, $\dots a_6$ use *the seven digits of your student number (don't forget the minus signs)*. For example, if your student number were 0654321 you would use $a_0 = 0$, $a_1 = 6$, $a_2 = 5$, $a_3 = 4$, $a_4 = 3$, $a_5 = 2$, and $a_6 = 1$. If you work with a partner, just use the student number of either of you. *Write the eigenvalues of B as computed by `eig` on the answer sheet* (report all entries to four decimal places). *Also write the student number you used for your matrix B.*

Using cofactor expansion it's easy to show that the characteristic polynomial of this matrix is

$$p(\lambda) = \det(B - \lambda I) = (-1)^7(a_0 + a_1\lambda + a_2\lambda^2 + \cdots + a_6\lambda^6 + \lambda^7). \tag{3}$$

The matrix B is called the *companion matrix* of the polynomial p (see Supplementary Exercises 19–23 on page 329). The connection between (2) and (3) is much more than a mere curiosity. Finding roots of polynomials is hard (if you don't believe that, try to find the roots of the polynomial p for your student number). How is it done in practice? *By finding the eigenvalues of the associated companion matrix.* This is what the MATLAB command `roots` does—see its documentation page in the MATLAB Help Browser. Thus, instead of finding eigenvalues of matrices by finding roots of polynomials (as we do by hand), MATLAB finds roots of polynomials by finding eigenvalues of matrices(!).

[2] If you don't need the eigenvectors, you can just use `D = eig(A)` to return a *vector* D of the eigenvalues.

[3] The command `eig` scales the eigenvectors to have norm (length) 1; if in MATLAB you normalize each eigenvector `v` you find by hand as `v/norm(v)` you should end up with the same vector (except possibly for the sign).

Step 2: The QR Method

This last discussion begs the question: how does the MATLAB command `eig` actually compute eigenvalues? It does it using the QR method. While implementing this method in practice is complicated, the basic idea is simple: Factor A into the product $A = QR$, multiply Q and R in reverse order to get a new matrix $A_1 = RQ$, and repeat over and over. Here the matrix Q satisfies $QQ^T = I$ (which makes it an *orthogonal matrix*, which we'll learn about in chapter 6) and the matrix R is upper-triangular.[4] Since Q is invertible, we have $A_1 = Q^{-1}AQ$ so A and A_1 are similar matrices and thus have the same eigenvalues (this is Exercise 23 of section 5.2, page 282). Therefore, each step in this iteration produces a new matrix having the same eigenvalues as A; under very general conditions this iteration converges to an *upper-triangular* matrix, the diagonal entries of which are the eigenvalues of A.

An easy way to run the QR method in MATLAB is to enter the matrix A and then repeat the following commands:

```
[Q,R] = qr(A); A = R*Q
```

This code writes the new version of A over the old version at each step. To do this by hand (to see how it works), type these commands on the command line and then use the up-arrow key to repeat them again and again. Once you understand what's going on, put these into a loop (using `for` and `end`) in your script (i.e., your `M-file`). Use this version of the QR method to compute the eigenvalues of the matrix A given in (1). Adjust the number of iterations (trips through the loop) until the matrix A no longer changes in the first four decimal places of each entry. Then *write the resulting (final) matrix A on the answer sheet*; this should be upper-triangular with the eigenvalues on the main diagonal (they should match the eigenvalues you found in Step 1).

[4] The details of the factorization don't concern us here (see section 6.4); we'll use the command `qr` to compute it.

Step 3: The Power Method

The QR method is the best general method for computing all eigenvalues of a matrix of moderate size (say, up to around $n = 1000$ or so), and it can compute eigenvectors as well.[5] What if you don't need all eigenvalues, and/or your matrix is much larger? For example, the Google PageRank algorithm (which determines the order in which to list results of web searches) is based on finding the largest eigenvalue and associated eigenvector of a matrix of size $n \times n$, where n is *the total number of pages linked on the Internet*. As of 2002 this number was about 2.7 billion (how much larger is it today?); using the QR algorithm for a problem of this size is out of the question.[6] For problems like this the *power method* is the way to go.

The power method is based on the idea that multiplying a vector $\mathbf{x}$ by a matrix scales the components of $\mathbf{x}$ in the directions of the eigenvectors by the corresponding eigenvalues. Do this again and again, and the component corresponding to the largest eigenvalue will begin to dominate. A basic explanation is given in section 5.8 (pages 319–324). Here, we will use a simple version to compute the eigenvalue λ_m having largest magnitude (and the corresponding eigenvector) of the matrix $C = B^T B$ [where B is the matrix given in (2)] as follows. First, use

```
C=B'*B; x = ones(7,1)
```

to define the matrix C (note the single quote ' for the transpose) and initialize $\mathbf{x}$ as a vector with each entry equal to 1. Then repeat the following until converged:

```
y = C*x; j=1; lambda_m = y(j)/x(j), x = y/norm(y)
```

As before, you can type these commands on the command line and then use the up-arrow key to repeat them again and again to see what's going on. Once you understand this calculation, put it in a loop (using `for` and `end`) in your script. Here we're estimating the eigenvalue λ_m by comparing the jth entries in $\mathbf{x}$ and $\mathbf{y}$; using $j = 1$ will work unless it divides by zero, in which case you'll need to change j to use some other entry which is not zero. The last step rescales $\mathbf{x}$ to have norm (length) 1; without this, the vector will continue to grow (or shrink) until it overflows (or underflows) what the computer can store. Adjust the number of iterations (trips through the loop) until the values of λ_m and $\mathbf{x}$ do not change in the first four decimal places from one iteration to the next. *Write these final values of λ_m and $\mathbf{x}$ on the answer sheet.*

Submit the following on Moodle:

1. The completed answer sheet (as a single PDF file). You may do this by:
   - Printing the page, writing your answers on it by hand, and scanning it (e.g., using a multifunction printer or an app such as (Camscanner), or
   - Writing your answers directly onto the PDF form (e.g., on a tablet).
2. Your completed MATLAB script (i.e., your `M-file`). This must run without modification to produce all output as required for this project. *Be sure to submit it with the proper name (see the first page), and do not convert it to any other file type. Staple this to the answer sheet.*

You may work alone or with *one* other person. If you work with someone, hand in only one copy of your work (with both names on top). *No groups bigger than two. No collaboration between groups.*

[5] Making this efficient is complicated; for details see *Matrix Computations* by Golub and Van Loan (or MA573).

[6] For some background on this problem, see "The World's Largest Matrix Computation" by Cleve Moler, *MATLAB News & Notes*, October 2002.

Matlab Project: Methods for Eigenvalues **Name:**_____

ANSWER SHEET **Name:**_____

Step 1: The MATLAB command `eig`

Eigenvalues and eigenvectors of A (report to four decimal places):

$\lambda_1 =$ $\lambda_2 =$ $\lambda_3 =$

$\mathbf{v}_1 =$ $\mathbf{v}_2 =$ $\mathbf{v}_3 =$

Eigenvalues of B (report to four decimal places):

$\lambda_1 =$ $\lambda_2 =$ $\lambda_3 =$

$\lambda_4 =$ $\lambda_5 =$ $\lambda_6 =$

$\lambda_7 =$ Student number you used for B:

Step 2: The QR Method

Final version of the matrix A (to four decimal places in each entry):

$A =$

Step 3: The Power Method

Final estimate of the eigenvalue of C having largest magnitude (to four decimal places):

$\lambda_m =$

Corresponding eigenvector (to four decimal places):

$\mathbf{x} =$

MATLAB input and output:

Staple a printed copy of your input and output (see instructions) to this answer sheet.