Finding the Roots of a Polynomial with Eigenvalues

CHAPTER 5 - PROJECT B

The purpose of this project is to show how the real roots of a polynomial can be calculated by finding the eigenvalues of a particular matrix. These eigenvalues will be found by the QR method described below.

Begin by recalling that a **polynomial of degree n** is a function of the form

$$p(t) = a_0 + a_1 t + \ldots + a_{n-1} t^{n-1} + a_n t^n$$

where a_0, a_1, … , a_{n-1}, and a_n are real numbers with $a_n \neq 0$. A **root** of a polynomial is a value of t for which $p(t) = 0$. It is often necessary (especially in calculus-based applications) to find all of the real roots of a given polynomial. In practice this can be a difficult problem even for a polynomial of low degree. For a polynomial of degree 2, every algebra student learns that the roots of $at^2 + bt + c$ can be found by the quadratic formula

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

If the polynomial is of degree 3 or 4, then there are formulas somewhat resembling the quadratic formula (but much more involved) for finding all the roots of a polynomial. However there is no general formula for finding the roots of a polynomial of degree 5 or higher using standard arithmetic properties and *n*th roots.

Example: Consider the cubic polynomial $p(t) = 6 - 5t - 2t^2 + t^3$. This polynomial is factored rather easily to find that its roots are $t = 1$, $t = -2$, and $t = 3$.

If a polynomial cannot easily be factored, numerical techniques are used to find a polynomial's roots. There are problems with this approach as well. Algorithms such as Newton's Method may not converge to a root, or may approach the root very slowly. These methods must also be applied repeatedly to find all of the roots, and usually require a cleverly chosen starting guess for the root being sought. However, there is an algorithm from linear algebra which may be used to find the real roots of a polynomial simultaneously.

Recall that the eigenvalues of an $n \times n$ matrix A are the roots of the characteristic polynomial of A, which is defined as $p(\lambda) = \det(A - \lambda I_n)$ and is a polynomial of degree n. So to know the eigenvalues of A is to know the roots of $p(\lambda)$. To find the roots of any polynomial p, then, two things are needed:

1. A way of finding a matrix A whose characteristic polynomial is $p(\lambda)$.

2. A way of finding the eigenvalues of this A which does not depend on finding the roots of $p(\lambda)$, since that is what is being attempted.

The first problem is solved by defining the **companion matrix** for a polynomial
$$p(t) = a_0 + a_1 t + \ldots + a_{n-1}t^{n-1} + a_n t^n .$$

Definition: If $p(t)$ is as given above, then the **companion matrix** for p is

$$C_p = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{pmatrix}.$$

Example (cont.): The companion matrix for the polynomial $p(t) = 6 - 5t - 2t^2 + t^3$ is
$$C_p = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & 5 & 2 \end{pmatrix}.$$

Questions:

1. Find the companion matrices for the following polynomials.

   a) $p(t) = t^2 + 2t - 4$

   b) $p(t) = t^3 - 9t^2 + 12t + 22$

   c) $p(t) = t^4 - 2t^3 - 13t^2 + 14t + 24$

   d) Find the characteristic polynomials of the matrices you just found in parts a)-c). What do you notice? (The MATLAB command **`poly(A)`** gives the coefficients for the characteristic polynomial of a matrix A.)

2. Show that the characteristic polynomial of a companion matrix for the n^{th} degree polynomial $p(t)$ is $\det(C_p - I_n) = (-1)^n p(\lambda)$ as follows.

   a) Show that if C_p is the companion matrix for a quadratic polynomial $p(t) = t^2 + a_1 t + a_0$, then $\det(C_p - I_2) = p(\lambda)$ by direct computation.

   b) Use mathematical induction to show that the result holds for $n \geq 2$. Hint: expand the necessary determinant by cofactors down the first column.

Thus there is a way to create the matrix A whose characteristic polynomial is $p(t)$. A method for finding the eigenvalues of A which does not use the characteristic polynomial is also needed. One method which accomplishes this is called the **QR method** because it is based on the QR factorization of A. Recall that if A is an $m\times n$ matrix with linearly independent columns, then A can be factored as $A = QR$, where Q is an $m\times n$ matrix with orthonormal columns and R is an

$n \times n$ invertible upper triangular matrix with positive entries on its main diagonal. Another exercise set which accompanies the text studies this method in depth. To introduce the algorithm, some properties that underlie this QR method for finding eigenvalues are first established.

3. Suppose A is a $n \times n$ matrix. Let $A = Q_0 R_0$ be a QR factorization of A, and create $A_1 = R_0 Q_0$. Let $A_1 = Q_1 R_1$ be a QR factorization of A_1 and create $A_2 = R_1 Q_1$.

    a) Show that $A = Q_0 A_1 Q_0^T$ (This is Exercise 31, Section 5.2.)

    b) Show that $A = (Q_0 Q_1) A_2 (Q_0 Q_1)^T$.

    c) Show that $Q_0 Q_1$ is an orthogonal matrix. (This is Exercise 37, Section 6.2.)

    d) Show that A, A_1, and A_2 all have the same eigenvalues.

The QR method for finding the eigenvalues of an $n \times n$ matrix A extends this process to create a sequence of matrices with the same eigenvalues.

The QR Method:

Step 1: Let $A = Q_0 R_0$ be a QR factorization of A; create $A_1 = R_0 Q_0$.

Step 2: Let $A_1 = Q_1 R_1$ be a QR factorization of A_1; create $A_2 = R_1 Q_1$.

Step 3: Continue this process. Once A_m has been created, let $A_m = Q_m R_m$ be a QR factorization of A_m and create $A_{m+1} = R_m Q_m$.

Step 4: Stop the process when the entries below the main diagonal of A_m are sufficiently small, or stop if it appears that convergence will not happen.

Steps 1-3 can be done using MATLAB. For example, the following commands correspond to steps 1-3 for a matrix A:

`[Q R]=qr(A)`	(Finds the QR factorization for A)
`A1=R*Q`	(Creates A_1)
`[Q R]=qr(A1)`	(Finds the QR factorization for A1)
`A2=R*Q`	(Creates A_2)

In fact, after the matrix is entered, you can type the following line in MATLAB, and then use the arrow up key to repeat the command:

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

Now repeat the command by scrolling up once.

Example (cont.): Let A be the companion matrix for the polynomial in the example; that is,

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & 5 & 2 \end{pmatrix}.$$

The QR factorization of this matrix is $A = Q_0 R_0 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{pmatrix}\begin{pmatrix} 6 & 5 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, so

$$A_1 = R_0 Q_0 = \begin{pmatrix} 2 & 6 & -5 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{pmatrix}.$$

This operation is performed again, getting

$$A_2 = R_1 Q_1 = \begin{pmatrix} -2.2361 & -5.3666 & 4.47214 \\ 0 & -2.6833 & 2.2361 \\ 0 & 0 & -1 \end{pmatrix}\begin{pmatrix} -0.8944 & -0.4472 & 0 \\ 0 & 0 & -1 \\ 0.4472 & -0.8944 & 0 \end{pmatrix}$$
$$= \begin{pmatrix} 4 & -3 & 5.3666 \\ 1 & -2 & 2.6833 \\ -0.4472 & 0.8944 & 0 \end{pmatrix}$$

This matrix is still far from upper triangular, so the process is continued. Eventually, it is found (after about 21 iterations) that

```
A =

   2.9997   -5.0292   -5.0847
  -0.0003   -1.9997   -1.3624
   0.0000    0.0000    1.0000
```

so the matrix is converging to an upper triangular matrix, and its diagonal elements are converging to the roots of *p(t)*:, $t = 3$, $t = -2$, and $t = 1$.

Question:

4. Approximate the roots of the polynomials in Question 1 by applying the QR Method to their companion matrices. Iterate until the entries below the main diagonal are all below 0.1.