

**MATH 220/02 - COMPUTER ASSIGNMENT**  
**DUE DATE: April 30, 2009 (in class)**

The purpose of this project is to introduce you to MATLAB as well as to some facts regarding  
(1) inverses and accuracy in solving systems, and  
(2) eigenvalues and how to find them.

- Recall that you may work in pairs and turn in one paper per partnership. Make sure both names are clearly indicated on your report.

- To access MATLAB (and other math software) online, use your WSU Network ID to log onto :

<http://my.math.wsu.edu/>

You may use other kinds of math software available to you.

- It might be useful to consult the *MATLAB Primer* for an introduction to MATLAB :

<http://math.ucsd.edu/~driver/21d-s99/matlab-primer.html>

## 1 An ‘ill-conditioned’ matrix

We call a square matrix  $A$  *ill-conditioned* if it is invertible but can become non-invertible (*singular*) if some of its entries are changed ever so slightly. The *condition number* of  $A$  is a measure of how ill-conditioned  $A$  is and can be found using  $A$  and  $A^{-1}$ . The bigger the condition number is the more ill-conditioned  $A$  is. Well-conditioned matrices have condition numbers close to 1.

Solving linear systems whose coefficient matrices are ill-conditioned is tricky because even a small change in the data (e.g., the right-hand side vector) can lead to radically different answers. This is illustrated by the next example (see problem 41, p.133): Let

$$A = \begin{bmatrix} 4.5 & 3.1 \\ 1.6 & 1.1 \end{bmatrix}, \quad b = \begin{bmatrix} 19.249 \\ 6.843 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 19.25 \\ 6.84 \end{bmatrix}.$$

Let’s solve the systems  $Ax = b$  and  $Ax = b_1$  with MATLAB:

```
>> % Enter the matrix A at the MATLAB prompt
>> A=[4.5 3.1; 1.6 1.1]    <Enter/Run Matlab>
>> % Enter the two right-hand sides
>> b=[19.249; 6.843]     <Enter/Run Matlab>
>> b1=[19.25; 6.84]     <Enter/Run Matlab>
>> % Solve the two systems Ax=b and Ax=b1 using the inverse of A
>> x=inv(A)*b    <Enter/Run Matlab>
>> x1=inv(A)*b1  <Enter/Run Matlab>
```

**TASK 1: Print out the solutions of  $Ax = b$  and  $Ax = b_1$ . Read the NUMERICAL NOTES at the end of Section 2.3 and use MATLAB’s built-in condition number command, `cond(A)`, to find the condition number of  $A$ . Compare the two solutions and comment on the magnitude of the condition number of  $A$  and how it relates to what you have observed.**

## 2 The QR algorithm to estimate the eigenvalues of a matrix

In practice, one rarely uses the characteristic polynomial to find the eigenvalues of a matrix  $A$ . Read the NUMERICAL NOTES at the end of Section 5.2 and the paragraph before Exercise 23 of the same section. The most commonly used methods to find eigenvalues are based on the so-called *QR algorithm*, described below :

1. Factor

$$A = Q_1 R_1 ,$$

where  $Q_1$  is an orthogonal matrix (meaning  $Q_1^{-1} = Q_1^T$ ) and  $R_1$  is upper triangular with positive diagonal entries. This factorization is found using the Gram-Schmidt process (see Theorem 12 in Section 6.4).

2. Interchange the factors and form the matrix

$$A_1 := R_1 Q_1 .$$

3. Factor  $A_1$  as in step 1:

$$A_1 = Q_2 R_2 .$$

4. Form the matrix

$$A_2 := R_2 Q_2$$

and continue in this manner. It can be shown that the sequence of matrices produced

$$A, A_1, A_2, A_3, \dots$$

all have the same eigenvalues as  $A$  and that they tend to become upper triangular. **So one can estimate the eigenvalues of  $A$  by the diagonal entries of an iterate  $A_k$ .** The larger the  $k$  the better this approximation will in general be. Let's try the above method to find the eigenvalues of

$$A = \begin{bmatrix} 4 & 1 & -1 \\ 3 & 2 & -3 \\ 2 & -2 & 1 \end{bmatrix} .$$

```
>> % Enter the matrix at the MATLAB prompt
>> A=[4 1 -1; 3 2 -3; 2 -2 1] <Enter/Run Matlab>
>> % Find the QR factorization of A using the built-in function qr( )
>> [Q1,R1]=qr(A) <Enter/Run Matlab>
>> % Interchange factors
>> A1=R1*Q1 <Enter/Run Matlab>
>> % Re-factor and interchange again
>> [Q2,R2]=qr(A1) <Enter/Run Matlab>
>> A2=R2*Q2 <Enter/Run Matlab>
>> % Continue in this manner until you produce several iterates
>> % Now use the MATLAB built-in function eig( ) to see how well you did
>> eig(A) <Enter/Run Matlab>
```

**Hint.** You can produce iterates quickly by doing the following after you have entered  $A$  :

```
>> A1=A <Enter/Run Matlab>
```

and overwrite  $A1$  with the current iterate by repeating the command

```
>> [Q1,R1]=qr(A1); A1=R1*Q1 <Enter/Run Matlab>
```

**TASK 2:** Print out the 5th, 10th and 21st iterates of the QR algorithm and compare their diagonal entries to the response of the `eig(A)` command. Report your findings in a table and comment on them.