

Mathematics 550 MATLAB Project 4

Unitary Diagonalization of Matrices, Schur's theorem, and Faddeev's Method for Characteristic Polynomials

Fall 2006

In this lab you will use MATLAB to study the following topics:

- Diagonalization of hermitian and normal matrices by unitary matrices
- Schur's triangularization of general matrices by unitary matrices
- Computation of eigenvalues and eigenvectors by Faddeev's Method

Tcodes For this lab you will need the Teaching Codes `polyf.m` and `syndivf.m` from a [new Teaching Codes](#) directory that I created for my section. (These should be available through a link in the preceding sentence.) Another MATLAB function, `shiftf.m` is also present, but it will not be used since it illustrates how to **completely** express a polynomial in x in terms of $x - c$ using synthetic division, but we only require the constant term of this expression. That term is obtained from a single use of `syndivf.m`. Before opening MATLAB to work on the Lab questions you should copy these codes to your directory. Instructions for other Teaching codes used in previous labs will not work for these.

Random Seed When you start your MATLAB session, initialize the random number generator by typing `rand('seed', abcd)` where `abcd` are the last four digits of your Student ID number. This will ensure that you generate your own particular random vectors and matrices.

BE SURE TO INCLUDE THIS LINE IN YOUR LAB WRITE-UP

Question 1. Unitary Diagonalization of Matrices Generate random 4×4 complex matrices A and B by

$$A = \text{rand}(4) + i*\text{rand}(4); \quad B = A + A'$$

(here $i = \sqrt{-1}$). Note that if A is a complex matrix, then the MATLAB notation A' means the **conjugate transpose** A^H . Hence B is hermitian symmetric.

(a): Use MATLAB to compute the matrix S of **normalized** eigenvectors of A and the diagonal eigenvalue matrix L of A by `[S, L] = eig(A)`. Note that this standard use of `eig` **always** normalizes eigenvectors (the columns of L) to have Euclidean (or Hermitian) length 1. Since A is a random matrix, it should have 4 distinct eigenvalues. Verify by MATLAB that $S^{-1}AS = L$. Then use MATLAB calculations to answer the following questions:

- (i) Are the columns of S mutually orthogonal?
- (ii) Is $S' * A * S = L$?
- (iii) Is $A * A' = A' * A$?

(b): Let U be the matrix of normalized eigenvectors of B and let D be the diagonal eigenvalue matrix of B . Before making any computations answer the following questions using the theory of hermitian matrices:

- (i) What can you predict about the eigenvalues of B ?
- (ii) Are the columns of U mutually orthogonal? Why?

(iii) Is $U' * B * U = D$? Why?

Now use MATLAB to calculate $[U, D] = \text{eig}(B)$ and confirm your answers to (i), (ii), and (iii).

(c): Let C be the complex matrix generated by the commands $s = \text{rand}(1)$, $t = \text{rand}(1)$, $C = s*B^2 + i*t*B^3$ (notice that each call to $\text{rand}(1)$ produces a different random number).

(i) Show by matrix algebra (without MATLAB calculation) that $C' * C = C * C'$. What does this tell you about the eigenvectors of C ?

(ii) Are the eigenvectors of B also eigenvectors of C ?

(iii) What is the relation between the eigenvalues of B and the eigenvalues of C ?

Now use MATLAB to calculate $E = U' * C * U$.

Then answer the following questions:

(iv) Why is the matrix E diagonal?

(v) What is the relation between D and E ?

Check your answer to (v) by MATLAB.

Question 2. Schur's Theorem and Householder matrices

To construct a matrix that is almost certain to have real eigenvalues, we start with a matrix with known eigenvalues and add a small quantity to each entry. If the eigenvalues of the original matrix are not too close, each eigenvalue of the modified matrix will be close to **one** of the original. This will keep them spread out and force them to be real. (For **small enough** modifications, this can be proved using Gershgorin's theorem, described in the exercises to Section 7.4) Here is one such construction. First, let

$M0 = [0, 1, 0; 0, 0, 1; 15, -23, 9]$.

This is a **companion matrix** of $x^3 - 9x^2 + 23x - 15 - (x - 1)(x - 3)(x - 5)$. Then, let

$M = M0 * M0 + \text{rand}(3, 3)$.

Since rand produces numbers between 0 and 1, the eigenvalues should be close to 1, 9 and 25.

(a) **The Schur triangularization.** Execute the command $[U, T] = \text{schur}(M)$. This claims to produce a **real orthogonal** matrix U and a **triangular** matrix T with $M = UTU^T$. **Check this:**

- (i) Insert a comment asserting that you have checked that T is triangular.
- (ii) Compute (in MATLAB) $U' * U$. Explain in a text comment why the result shows that U is orthogonal.
- (iii) Compute (in MATLAB) $M * U - T * U$. In a text comment, say why the result confirms that T is the required triangular matrix.
- (iv) In a text comment, say what this result tells you about the eigenvalues of M . Also, without further computation, say why the **first column** of U is an eigenvector of M , and what its eigenvalue is.

(b) **A Householder matrix.** Write the appropriate MATLAB statement to define $u1$ as the first column of U . Then, let $v1 = u1 - [1; 0; 0]$ and $H1 = \text{eye}(3) - 2 * v1 * v1' / (v1' * v1)$.

- (i) Insert a comment indicating why $u1$ is **known** to have length 1.
- (ii) Check (in MATLAB) that the square of $H1$ is the identity, and check (visually) that $H1$ is symmetric. Add a comment indicating why this shows that $H1$ is orthogonal.
- (iii) This construction is supposed to give an orthogonal matrix whose first column is the same as the first column of U . Check this (visually), compute $U1 = H1 * U$ (it really should be $H1' * U$, but $H1$ is symmetric, so a simpler statement can be used), and add a comment indicating why the contents of the first row and first column of the answer agree with the theoretically expected results.
- (iv) Why is $U1$ expected to be orthogonal? Is this confirmed by a MATLAB computation of $U1 * U1'$? (Answer both questions in a text comment.)

- (v) Compute (in MATLAB) $H1 * M * H1$ (it really should be $H1' * M * H1$, but $H1$ is symmetric, so a simpler statement can be used). What property does the **second** column of $U1$ have for this matrix? Use MATLAB to check your answer.

Question 3. The Faddeev-Frame computation of characteristic polynomials

Familiarity with [Supplement 7](#) is assumed. Although the strength of that method is its ability to be used for hand computation with **exact** data, the implementation as a MATLAB function is an aid to understanding the algorithm. Although MATLAB was found to be fast enough with a 20 by 20 matrix, the result was useless because computations were limited to machine floating point accuracy. We construct a special 10 by 10 matrix of rank 2 whose characteristic polynomial is x^8 times a quadratic. This will allow the eigenvalues to be found by the quadratic formula.

Use the function `rmat` constructed in a previous lab to build a matrix B by the following commands:

```
A1=rmat(10,1); A2=rmat(1,10); B=A1*A2; B(1,1)=-1;
```

Initially, B is rank 1, but changing a single entry destroys this property. However, all columns except the first are still multiples of $A1$, so the result has rank 2.

(a) Finding the characteristic polynomial Use the function `polyf` that you copied at the start of this project to compute the characteristic polynomial of B by writing `[v,C3]=polyf(B);`. Be sure to include the semicolon at the end of the command, since $C3$ is very large. After this, the coefficients of the characteristic polynomial of B are stored in v . As in the supplement, the characteristic polynomial is found in the form $x^{10} - v_1x^9 - v_2x^8 - \dots$, where the remaining coefficients are expected to be zero. (Here, the indices are written as subscript in mathematical notation.)

Because of the way the coefficients are written, the nonzero eigenvalues are

$$\frac{v_1 \pm \sqrt{v_1^2 + 4v_2}}{2}.$$

The **layers** of the array $C3$ (identified by fixing the third index) are the matrix coefficients of $\text{adj}(xI - B)$. This allows them to be expressed in terms of subdeterminants of B , and the special form of the matrix means that these matrices will soon be zero. The numbers in this calculation (up to this point) remain small, so you can use `rank(C3(:, :, k))` for $k = 1, 2, 3, \dots$ to generate minimal output while finding when one gets a zero matrix.

- (i) Write the characteristic polynomial from the value of v .
- (ii) Select one of the nonzero eigenvalues and let r be its value computed from the v_i by the quadratic formula.
- (iii) What are the ranks of the first few layers of $C3$? When the rank becomes zero, the matrix B_k (as described in the supplement) is the zero matrix, and all later matrices in the computation will be zero, so you can stop when you find a matrix of rank zero. The columns of the previous matrix span the **nullspace** of B , which is the eigenspace for the eigenvalue zero. (If it is 8, there will be a basis of eigenvectors (i.e., the matrix can be diagonalized).

(b) Finding other eigenvectors Now, take the value of r computed from the quadratic formula and use the `syndivf` function to evaluate $\text{adj}(xI - B)$ at $x = r$. Giving the command

```
[Q3,V] = syndivf(C3,c);
```

will produce the desired matrix as V (you may produce the matrix under another name if you prefer) while recording the quotient as a three dimensional array $Q3$. This quotient will not be used here.

In the usual case where r is a simple eigenvalue, the matrix V has **every** column as an eigenvector for r . Because of rounding, you may not appear to satisfy all definitions of eigenvectors, but **ratios** of corresponding entries in the last layer of array and the matrix found by multiplying this layer by B should all be seen to be r .

Conclude your work by giving an eigenvector for your choice of r . Include some check that your answer is correct (to machine accuracy).

End of Lab 4