

# Chapter 7

## 1. Introduction

In this chapter we'll explore eigenvectors and eigenvalues from geometric perspectives, learn how to use MATLAB to algebraically identify them, and ultimately see how these notions are famously applied to the diagonalization of matrices – a result itself that has a myriad of applications. Throughout, we'll look at examples from the field of dynamical systems to provide the motivation for desiring a diagonal form of a matrix. We'll work with the m-file `eigshow` and the following MATLAB commands: `eig`.

## 2. The Defining Equation

The innocent looking equation  $A\vec{x} = \lambda\vec{x}$  defines eigenvectors and associated eigenvalues whenever there are nonzero solutions ( $\vec{x} \neq \vec{0}$ ) to this equation (recall that  $\lambda$  can be zero). Geometrically this equation says that if you find a vector  $\vec{x}$  such that  $\vec{x}$  and  $A\vec{x}$  are parallel, then  $\vec{x}$  is an eigenvector of  $A$ .

By calling the m-file `eigshow` from the Command Window, you'll launch a program that displays a unit vector,  $\vec{x}$ , and its image,  $A\vec{x}$ , under multiplication by the matrix  $A$ . The matrix  $A$  is shown in the pull-down menu at the top of the new window. Clicking on the Figure window and holding the mouse button down will allow you to drag  $\vec{x}$  around the unit circle. Wherever you can discover  $\vec{x}$  parallel to its image (the vectors will turn red), you will have found an eigenvector. Go ahead and experiment with other matrices from the pull-down menu.

It's also possible to load a 2x2 matrix into `eigshow`. For example,

```
>> A = [1 2; 3 4];  
>> eigshow(A)
```

This new matrix  $A$  should appear in the pull-down menu now.

**2a.** 7.1 #12 Use `eigshow` to estimate the eigenvectors

**2b.** Load the matrix from the coyote and roadrunner example (in Section 7.1 of the text) into `eigshow` and compare the eigenvectors you find to the vectors mentioned in Cases 1 and 2 found on pages 293-294. Notice the eigenvectors in MATLAB will be normalized.

**2c.** Repeat the work in Case 3 on page 294 with  $\vec{x}_0 = \begin{bmatrix} 500 \\ 1000 \end{bmatrix}$  (What does this vector mean for this application?) Show how to use MATLAB to determine the coordinates  $c_1$  and  $c_2$ . Give formulas for  $c(t)$  and  $r(t)$ .

**2d.** 7.1 #50

### 3. Algebraically Determining Eigenvalues for a Matrix

Trying to solve the defining equation,  $A\vec{x} = \lambda\vec{x}$ , for non-zero vectors first leads us to discovering the eigenvalues of the matrix  $A$ . Begin by putting all your  $\vec{x}$ 's on one side...

$A\vec{x} - \lambda\vec{x} = \vec{0} \Leftrightarrow (A - \lambda I)\vec{x} = \vec{0}$ . We know this equation has non-zero solutions<sup>1</sup> iff  $\det(A - \lambda I) = 0$ .

Notes: We call  $\det(A - \lambda I)$  the *characteristic polynomial*, and  $\det(A - \lambda I) = 0$  the *characteristic equation*. You may also see the characteristic polynomial written  $\det(\lambda I - A)$ . Some people prefer this last form as it guarantees a positive leading coefficient, but the same people pay a higher price later on when computing eigenvectors.

Observation: The roots of the characteristic equation are the eigenvalues of  $A$ .

(Good time to pause – this is a big deal.)

Certainly, one can use MATLAB's `solve` command to get the roots of  $\det(A - \lambda I)$ . For example, try entering the following in MATLAB...

```
>> syms lambda;  
>> A = [.86 .08; -.12 1.14];  
>> solve(det(A - lambda*eye(2)))
```

**MATLAB Note:** `solve(polynomial)` will provide the roots of the polynomial, that is, `solve(polynomial)` solves the equation  $\text{polynomial} = 0$ . Although this isn't too cumbersome, MATLAB has an alternate method for getting eigenvalues of a matrix: the `eig` command.

---

<sup>1</sup> Remember, this now means eigenvectors.

**3a.** With the matrix  $A$  above, enter `eig(A)` in MATLAB. Try also `rats(eig(A))`. Check out and compare the way MATLAB stores the answer for `solve()`, `eig()`, and `rats(eig())`. **MATLAB Warning:** If the eigenvalues are not rational numbers, MATLAB will vomit “approximating fractions.” For example, `rats(sqrt(2))` yields 1393/985 (this agrees with  $\sqrt{2}$  out to 6 decimal places). You can see the differences for `ans` in the Workspace window.

**3b.** 7.2 #27 (May skip sketching the trajectories.)

#### 4. Algebraically Finding the Eigenvectors of a Matrix

Once you have the eigenvalues for a matrix, you can go back to  $(A - \lambda I)\vec{x} = \vec{0}$ , plug in the actual eigenvalues for  $\lambda$  and solve each of the different homogeneous systems for the different  $\lambda$ s.

For example, using the eigenvalue  $\lambda = 1.1$  from the coyote and roadrunner example, we

have  $(A - \lambda I)\vec{x} = \left( \begin{bmatrix} .86 & .08 \\ -.12 & 1.14 \end{bmatrix} - \begin{bmatrix} 1.1 & 0 \\ 0 & 1.1 \end{bmatrix} \right) \vec{x} = \begin{bmatrix} -.24 & .08 \\ -.12 & .04 \end{bmatrix} \vec{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ . This system has

solutions  $t \begin{bmatrix} \frac{1}{3} \\ 1 \end{bmatrix}$ . Any vector in this solution space serves as an eigenvector for  $\lambda = 1.1$ .

**4a.** Use the scheme just shown to identify an eigenvector for  $\lambda = .9$  from the coyote and roadrunner example.

Now, you can go around talking about the  $\ker(A - 1.1I)$ , but you’ll sound really sophisticated calling this kernel the *eigenspace associated with  $\lambda = 1.1$* . The dimension of an eigenspace associated with  $\lambda$ , called the geometric multiplicity of  $\lambda$ , is simply the dimension of the associated kernel.

The ultimate eigen-word: If you can find  $n$  linearly independent eigenvectors for an  $n \times n$  matrix  $A$ , then we say that there exists an *eigenbasis* for  $A$ .

MATLAB provides a way to get eigenvectors and eigenvalues in one command. By typing `[P, D] = eig(A)` in the command window, MATLAB will produce a matrix  $P$  whose columns are of eigenvectors of  $A$  that correspond column-wise to eigenvalues displayed in the diagonal matrix  $D$ .

Let's revisit Example 5 on page 321 to see what numbers MATLAB would produce for this problem.

```
>> A = [0 .95 .6; .8 0 0; 0 .5 0];  
>> [P,D] = eig(A)
```

P =

```
    0.7454    -0.4992    0.2981  
    0.5963     0.6656   -0.5963  
    0.2981   -0.5547    0.7454
```

D =

```
    1.0000         0         0  
         0   -0.6000         0  
         0         0   -0.4000
```

The columns of P don't look like the vectors used in the eigenbasis on page 322, but they are equivalent. MATLAB *produces eigenvectors of unit length*. However, let's proceed

with what we have and express the initial state vector  $\vec{x}_0 = \begin{bmatrix} 750 \\ 200 \\ 200 \end{bmatrix}$  as a linear combination

of our three eigenvectors:

```
>> b = [750; 200; 200];  
>> x = inv(P)*b;  
>> c1 = x(1);  
>> c2 = x(2);  
>> c3 = x(3);  
>> p1 = P(:,1);  
>> p2 = P(:,2);  
>> p3 = P(:,3);  
>> % Check that x0 = c1*p1 + c2*p2 + c3*p3
```

$$\begin{aligned}\bar{x}(t) &= A^t \bar{x}_0 = A^t \left( 670.8204 \begin{bmatrix} 0.7454 \\ 0.5963 \\ 0.2981 \end{bmatrix} - 901.3878 \begin{bmatrix} -0.4992 \\ 0.6656 \\ -0.5547 \end{bmatrix} - 670.8204 \begin{bmatrix} 0.2981 \\ -0.5963 \\ 0.7454 \end{bmatrix} \right) \\ &= 670.8204 A^t \begin{bmatrix} 0.7454 \\ 0.5963 \\ 0.2981 \end{bmatrix} - 901.3878 A^t \begin{bmatrix} -0.4992 \\ 0.6656 \\ -0.5547 \end{bmatrix} - 670.8204 A^t \begin{bmatrix} 0.2981 \\ -0.5963 \\ 0.7454 \end{bmatrix}\end{aligned}$$

But each “ $A^t$  times eigenvector” can be replaced with “ $\lambda^t$  times eigenvector”, so

$$\bar{x}(t) = A^t \bar{x}_0 = 670.8204(1)^t \begin{bmatrix} 0.7454 \\ 0.5963 \\ 0.2981 \end{bmatrix} - 901.3878(-0.6)^t \begin{bmatrix} -0.4992 \\ 0.6656 \\ -0.5547 \end{bmatrix} - 670.8204(-0.4)^t \begin{bmatrix} 0.2981 \\ -0.5963 \\ 0.7454 \end{bmatrix}$$

This yields

$$\begin{aligned}j(t) &= 670.8204(0.7454) - 901.3878(-0.6)^t(-0.4992) - 670.8204(-0.4)^t(0.2981) \\ m(t) &= 670.8204(0.5963) - 901.3878(-0.6)^t(0.6656) - 670.8204(-0.4)^t(-0.5963) \\ a(t) &= 670.8204(0.2981) - 901.3878(-0.6)^t(-0.5547) - 670.8204(-0.4)^t(0.7454)\end{aligned}$$

And in the long run,  $j \rightarrow 500$ ,  $m \rightarrow 400$ , and  $a \rightarrow 200$ .

Admittedly the numbers here aren't nearly as nice as those in the text, but in this example we didn't invest any time in trying to find integer representations for the eigenvectors.

**4b.** 7.3 #41

## 5. Diagonalization

In Example 5 on page 321 it first appears that we need an expression for  $A^t$  (MATLAB can't give you a general expression for this matrix!). For this example however, a little cleverness was used to get around this problem – the initial state vector was decomposed into a linear combination of eigenvectors, then the eigen-relationships allowed us to write

$$\bar{x}(t) = A^t \bar{x}_0 = A^t (c_1 \bar{v}_1 + c_2 \bar{v}_2 + c_3 \bar{v}_3) = c_1 A^t \bar{v}_1 + c_2 A^t \bar{v}_2 + c_3 A^t \bar{v}_3 = c_1 \lambda_1^t \bar{v}_1 + c_2 \lambda_2^t \bar{v}_2 + c_3 \lambda_3^t \bar{v}_3$$

In the absence of an initial state vector (or regardless of the initial state vector), can you still analyze the behavior of a dynamical system? In other words, can an expression for  $A^t$  be discovered?

To begin answering this question, we turn your attention to diagonal matrices. Diagonal matrices have a few intriguing traits: Powers of a diagonal matrix are a snap to compute and diagonal matrices commute with each other, but not with general matrices.

**5a.** Try computing  $\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}^3 = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} =$

$$\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} d & 0 & 0 \\ 0 & e & 0 \\ 0 & 0 & f \end{bmatrix} = \begin{bmatrix} d & 0 & 0 \\ 0 & e & 0 \\ 0 & 0 & f \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} =$$

$$\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} =$$

**5b.** Suppose an  $n \times n$  matrix  $A$  has an eigenbasis  $(\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n)$  with  $A\vec{v}_i = \lambda_i \vec{v}_i$  and consider

$$A[\vec{v}_1 \mid \vec{v}_2 \mid \dots \mid \vec{v}_n] = [A\vec{v}_1 \mid A\vec{v}_2 \mid \dots \mid A\vec{v}_n] = [\lambda_1 \vec{v}_1 \mid \lambda_2 \vec{v}_2 \mid \dots \mid \lambda_n \vec{v}_n]$$

Based on your observations in problem 5a, try factoring the last matrix on the right into two square matrices. One of your matrices will be  $S$ , call the other  $D$ .

**5c.** Continuing from problem 5b, the matrix  $S = [\vec{v}_1 \mid \vec{v}_2 \mid \dots \mid \vec{v}_n]$  is invertible, why? Use the equation  $AS = (\text{your factorization})$  to figure  $S^{-1}AS$ .

Now, if  $D$  is a diagonal matrix and  $S^{-1}AS = D$ , it doesn't take much to get to  $A = SDS^{-1}$ . (Imagine if  $D$  commuted with all compatible matrices; what would have to be true about  $A$ ?)

Note also that

$$A^3 = (SDS^{-1})^3 = (SDS^{-1})(SDS^{-1})(SDS^{-1}) = SDS^{-1}SDS^{-1}SDS^{-1} = SD(S^{-1}S)D(S^{-1}S)DS^{-1} = \underline{\hspace{2cm}}$$

Summary: If a matrix  $A$  has an eigenbasis, we can get a diagonal form associated with  $A$ , and get an expression for  $A^t$  by exploiting the simple calculation of computing powers of a diagonal matrix.

For example, revisiting Example 5 on page 321, the matrix  $A = \begin{bmatrix} 0 & .95 & .6 \\ .8 & 0 & 0 \\ 0 & .5 & 0 \end{bmatrix}$  can be

diagonalized to  $D$  by the matrix  $V$ , using MATLAB:

```
>> [P,D] = eig(A)
```

P =

```
    0.7454    -0.4992    0.2981
    0.5963     0.6656   -0.5963
    0.2981   -0.5547    0.7454
```

D =

```
    1.0000         0         0
         0   -0.6000         0
         0         0   -0.4000
```

As we expect, the main diagonal of  $D$  shows the eigenvalues corresponding, column-by-column, to the eigenvectors that make up the columns of  $V$ . And the  $t^{\text{th}}$  power of  $A$  can be expressed as  $A^t = VD^tV^{-1}$ .

Here is one way to get MATLAB to carry out this calculation.

```
>> syms t
>> Dt = [1^t 0 0; 0 (-.6)^t 0; 0 0 (-.4)^t]
```

then

```
>> V*Dt*inv(V)
```

ans =

(\* lots of numbers – try it! \*)

But you can see from the display that  $\lim_{t \rightarrow \infty} A^t = \frac{1}{224} \begin{bmatrix} 100 & 125 & 60 \\ 80 & 100 & 48 \\ 40 & 50 & 24 \end{bmatrix}$  and therefore that

$$\lim_{t \rightarrow \infty} \vec{x}(t) = \lim_{t \rightarrow \infty} \begin{bmatrix} j(t) \\ m(t) \\ a(t) \end{bmatrix} = \lim_{t \rightarrow \infty} A^t \begin{bmatrix} j_0 \\ m_0 \\ a_0 \end{bmatrix} = \frac{1}{224} \begin{bmatrix} 100 & 125 & 60 \\ 80 & 100 & 48 \\ 40 & 50 & 24 \end{bmatrix} \begin{bmatrix} j_0 \\ m_0 \\ a_0 \end{bmatrix} = \frac{1}{224} \begin{bmatrix} 100j_0 + 125m_0 + 60a_0 \\ 80j_0 + 100m_0 + 48a_0 \\ 40j_0 + 50m_0 + 24a_0 \end{bmatrix}$$

Recall that  $\begin{bmatrix} j_0 \\ m_0 \\ a_0 \end{bmatrix} = \begin{bmatrix} 750 \\ 200 \\ 200 \end{bmatrix}$  in the original example. Using that initial state vector given in

the example does in fact give  $\frac{1}{224} \begin{bmatrix} 100(750) + 125(200) + 60(200) \\ 80(750) + 100(200) + 48(200) \\ 40(750) + 50(200) + 24(200) \end{bmatrix} = \begin{bmatrix} 500 \\ 400 \\ 200 \end{bmatrix}$  as before.

But now that we have a more general expression for  $A^t$ , we can more easily analyze the behavior of the system with different initial state vectors.

**5d.** With  $A = \begin{bmatrix} 0.1 & 0.3 & 0.4 & 0.2 \\ 0 & 0.3 & 0.5 & 0.2 \\ 0.5 & 0.2 & 0.1 & 0.2 \\ 0.1 & 0.5 & 0 & 0.4 \end{bmatrix}$ , show how you can use MATLAB to get an expression for  $A^t$