Chapter 5

Eigenvectors

We turn our attention now to a *nonlinear* problem about matrices: Finding their eigenvalues and eigenvectors. Eigenvectors $\vec{x}$ and their corresponding eigenvalues λ of a square matrix A are determined by the equation $A\vec{x} = \lambda\vec{x}$. There are many ways to see that this problem is nonlinear. For instance, there is a *product* of unknowns λ and $\vec{x}$, and to avoid the trivial solution $\vec{x} = \vec{0}$ we constrain $\|\vec{x}\| = 1$; this constraint is circular rather than linear. Thanks to this structure, our methods for finding eigenspaces will be considerably different from techniques for solving and analyzing linear systems of equations.

5.1 Motivation

Despite the arbitrary-looking form of the equation $A\vec{x} = \lambda\vec{x}$, the problem of finding eigenvectors and eigenvalues arises naturally in many circumstances. We motivate our discussion with a few examples below.

5.1.1 Statistics

Suppose we have machinery for collecting several statistical observations about a collection of items. For instance, in a medical study we may collect the age, weight, blood pressure, and heart rate of 100 patients. Then, each patient i can be represented by a point $\vec{x}_i$ in $\mathbb{R}^4$ storing these four values.

Of course, such statistics may exhibit strong correlation. For instance, patients with higher blood pressure may be likely to have higher weights or heart rates. For this reason, although we collected our data in $\mathbb{R}^4$, in reality it may—to some approximate degree—live in a lower-dimensional space better capturing the relationships between the different variables.

For now, suppose that in fact there exists a *one*-dimensional space approximating our dataset. Then, we expect all the data points to be nearly parallel to some vector $\vec{v}$, so that each can be written as $\vec{x}_i \approx c_i\vec{v}$ for different $c_i \in \mathbb{R}$. From before, we know that the best approximation of $\vec{x}_i$ parallel to $\vec{v}$ is $\text{proj}_{\vec{v}}\,\vec{x}_i$:

$$\begin{aligned}\text{proj}_{\vec{v}}\,\vec{x}_i &= \frac{\vec{x}_i \cdot \vec{v}}{\vec{v} \cdot \vec{v}}\vec{v} \text{ by definition} \\ &= (\vec{x}_i \cdot \hat{v})\hat{v} \text{ since } \vec{v} \cdot \vec{v} = \|\vec{v}\|^2\end{aligned}$$

Here, we define $\hat{v} \equiv \vec{v}/\|\vec{v}\|$. Of course, the magnitude of $\vec{v}$ does not matter for the problem at hand, so it is reasonable to search over the space of unit vectors $\hat{v}$.

Following the pattern of least squares, we have a new optimization problem:

$$\begin{aligned} &\text{minimize } \sum_i \|\vec{x}_i - \text{proj}_{\hat{v}} \vec{x}_i\|^2 \\ &\text{such that } \|\hat{v}\| = 1 \end{aligned}$$

We can simplify our optimization objective a bit:

$$\begin{aligned} \sum_i \|\vec{x}_i - \text{proj}_{\hat{v}} \vec{x}_i\|^2 &= \sum_i \|\vec{x}_i - (\vec{x}_i \cdot \hat{v})\hat{v}\|^2 \text{ by definition of projection} \\ &= \sum_i \left(\|\vec{x}_i\|^2 - (\vec{x}_i \cdot \hat{v})^2 \right) \text{ since } \|\hat{v}\| = 1 \text{ and } \|\vec{w}\|^2 = \vec{w} \cdot \vec{w} \\ &= \text{const.} - \sum_i (\vec{x}_i \cdot \hat{v})^2 \end{aligned}$$

This derivation shows that we can solve an equivalent optimization problem:

$$\begin{aligned} &\text{maximize } \|X^\top \hat{v}\|^2 \\ &\text{such that } \|\hat{v}\|^2 = 1, \end{aligned}$$

where the columns of X are the vectors $\vec{x}_i$. Notice that $\|X^\top \hat{v}\|^2 = \hat{v}^\top X X^\top \hat{v}$, so by Example 0.27 the vector $\hat{v}$ corresponds to the eigenvector of $XX^\top$ with the highest eigenvalue. The vector $\hat{v}$ is known as the first *principal component* of the dataset.

5.1.2 Differential Equations

Many physical forces can be written as functions of position. For instance, the force between two particles at positions $\vec{x}$ and $\vec{y}$ in $\mathbb{R}^3$ exerted by a spring can be written as $k(\vec{x} - \vec{y})$ by Hooke's Law; such spring forces are used to approximate forces holding cloth together in many simulation systems. Although these forces are not necessarily *linear* in position, we often approximate them in a linear fashion. In particular, in a physical system with n particles encode the positions of all the particles simultaneously in a vector $\vec{X} \in \mathbb{R}^{3n}$. Then, if we assume such an approximation we can write that the forces in the system are approximately $\vec{F} \approx A\vec{X}$ for some matrix A.

Recall Newton's second law of motion $F = ma$, or force equals mass times acceleration. In our context, we can write a diagonal *mass matrix* $M \in \mathbb{R}^{3n \times 3n}$ containing the mass of each particle in the system. Then, we know $\vec{F} = M\vec{X}''$, where prime denotes differentiation in time. Of course, $\vec{X}'' = (\vec{X}')'$, so in the end we have a *first*-order system of equations:

$$\frac{d}{dt} \begin{pmatrix} \vec{X} \\ \vec{V} \end{pmatrix} = \begin{pmatrix} 0 & I_{3n \times 3n} \\ M^{-1}A & 0 \end{pmatrix} \begin{pmatrix} \vec{X} \\ \vec{V} \end{pmatrix}$$

Here, we simultaneously compute both positions in $\vec{X} \in \mathbb{R}^{3n}$ and velocities $\vec{V} \in \mathbb{R}^{3n}$ of all n particles as functions of time.

More generally, differential equations of the form $\vec{x}' = A\vec{x}$ appear in many contexts, including simulation of cloth, springs, heat, waves, and other phenomena. Suppose we know eigenvectors

$\vec{x}_1, \ldots, \vec{x}_k$ of A, such that $A\vec{x}_i = \lambda_i \vec{x}_i$. If we write the initial condition of the differential equation in terms of the eigenvectors, as

$$\vec{x}(0) = c_1\vec{x}_1 + \cdots + c_k\vec{x}_k,$$

then the solution to the equation can be written in closed form:

$$\vec{x}(t) = c_1 e^{\lambda_1 t}\vec{x}_1 + \cdots + c_k e^{\lambda_k t}\vec{x}_k,$$

This solution is easy to check by hand. That is, if we write the initial conditions of this differential equation in terms of the eigenvectors of A, then we know its solution for all times $t \geq 0$ for free. Of course, this formula is not the end of the story for simulation: Finding the complete set of eigenvectors of A is expensive, and A may change over time.

5.2 Spectral Embedding

Suppose we have a collection of n items in a dataset and a measure $w_{ij} \geq 0$ of how similar each pair of elements i and j are; we will assume $w_{ij} = w_{ji}$. For instance, maybe we are given a collection of photographs and use w_{ij} to compare the similarity of their color distributions. We might wish to sort the photographs based on their similarity to simplify viewing and exploring the collection.

One model for ordering the collection might be to assign a number x_i for each item i, asking that similar objects are assigned similar numbers. We can measure how well an assignment groups similar objects by using the energy

$$E(\vec{x}) = \sum_{ij} w_{ij}(x_i - x_j)^2.$$

That is, $E(\vec{x})$ asks that items i and j with high similarity scores w_{ij} get mapped to nearby values.

Of course, minimizing $E(\vec{x})$ with no constraints gives an obvious minimum: $x_i = \text{const.}$ for all i. Adding a constraint $\|\vec{x}\| = 1$ does *not* remove this constant solution! In particular, taking $x_i = 1/\sqrt{n}$ for all i gives $\|\vec{x}\| = 1$ and $E(\vec{x}) = 0$ in an uninteresting way. Thus, we must remove this case as well:

$$\begin{aligned} &\text{minimize } E(\vec{x}) \\ &\text{such that } \|\vec{x}\|^2 = 1 \\ &\qquad \vec{1} \cdot \vec{x} = 0 \end{aligned}$$

Notice that our second constraint asks that the sum of $\vec{x}$ is zero.

Once again we can simplify the energy:

$$\begin{aligned} E(\vec{x}) &= \sum_{ij} w_{ij}(x_i - x_j)^2 \\ &= \sum_{ij} w_{ij}(x_i^2 - 2x_i x_j + x_j^2) \\ &= \sum_i a_i x_i^2 - 2\sum_{ij} w_{ij} x_i x_j + \sum_j b_j x_j^2 \\ &\qquad \text{for } a_i \equiv \sum_j w_{ij} \text{ and } b_j \equiv \sum_i w_{ij} \\ &= \vec{x}^\top (A - 2W + B)\vec{x} \text{ where } \text{diag}(A) = \vec{a} \text{ and } \text{diag}(B) = \vec{b} \\ &= \vec{x}^\top (2A - 2W)\vec{x} \text{ by symmetry of } W \end{aligned}$$

It is easy to check that $\vec{1}$ is an eigenvector of $2A - 2W$ with eigenvalue 0. More interestingly, the eigenvector corresponding to the *second*-smallest eigenvalue corresponds to the solution of our minimization objective above! (TODO: Add KKT proof from lecture)

5.3 Properties of Eigenvectors

We have established a variety of applications in need of eigenspace computation. Before we can explore algorithms for this purpose, however, we will more closely examine the structure of the eigenvalue problem.

We can begin with a few definitions that likely are evident at this point:

Definition 5.1 (Eigenvalue and eigenvector)**.** *An* eigenvector $\vec{x} \neq \vec{0}$ *of a matrix* $A \in \mathbb{R}^{n \times n}$ *is any vector satisfying* $A\vec{x} = \lambda\vec{x}$ *for some* $\lambda \in \mathbb{R}$*; the corresponding* λ *is known as an* eigenvalue*.* Complex *eigenvalues and eigenvectors satisfy the same relationships with* $\lambda \in \mathbb{C}$ *and* $\vec{x} \in \mathbb{C}^n$*.*

Definition 5.2 (Spectrum and spectral radius)**.** *The* spectrum *of* A *is the set of eigenvalues of* A*. The* spectral radius $\rho(A)$ *is the eigenvalue* λ *maximizing* $|\lambda|$*.*

The scale of an eigenvector is not important. In particular, scaling an eigenvector $\vec{x}$ by c yields $A(c\vec{x}) = cA\vec{x} = c\lambda\vec{x} = \lambda(c\vec{x})$, so $c\vec{x}$ is an eigenvector with the same eigenvalue. We often restrict our search by adding a constraint $\|\vec{x}\| = 1$. Even this constraint does not completely relieve ambiguity, since now $\pm\vec{x}$ are both eigenvectors with the same eigenvalue.

The algebraic properties of eigenvectors and eigenvalues easily could fill a book. We will limit our discussion to a few important theorems that affect the design of numerical algorithms; we will follow development of CITE AXLER. First, we should check that every matrix has at least one eigenvector so that our search is not in vain. Our usual strategy is to notice that if λ is an eigenvalue such that $A\vec{x} = \lambda\vec{x}$, then $(A - \lambda I_{n\times n})\vec{x} = \vec{0}$; thus, λ is an eigenvalue exactly when the matrix $A - \lambda I_{n\times n}$ is not full-rank.

Lemma 5.1 (CITE Theorem 2.1)**.** *Every matrix* $A \in \mathbb{R}^{n\times n}$ *has at least one (complex) eigenvector.*

Proof. Take any vector $\vec{x} \in \mathbb{R}^n \backslash \{\vec{0}\}$. The set $\{\vec{x}, A\vec{x}, A^2\vec{x}, \cdots, A^n\vec{x}\}$ must be linearly dependent because it contains $n+1$ vectors in n dimensions. So, there exist constants $c_0, \ldots, c_n \in \mathbb{R}$ with $c_n \neq 0$ such that

$$\vec{0} = c_0\vec{x} + c_1A\vec{x} + \cdots + c_nA^n\vec{x}.$$

We can write down a polynomial

$$f(z) = c_0 + c_1z + \cdots + c_nz^n.$$

By the Fundamental Theorem of Algebra, there exist n roots $z_i \in \mathbb{C}$ such that

$$f(z) = c_n(z - z_1)(z - z_2)\cdots(z - z_n).$$

Then, we have:

$$\begin{aligned}\vec{0} &= c_0\vec{x} + c_1A\vec{x} + \cdots + c_nA^n\vec{x} \\ &= (c_0I_{n\times n} + c_1A + \cdots + c_nA^n)\vec{x} \\ &= c_n(A - z_1I_{n\times n})\cdots(A - z_nI_{n\times n})\vec{x} \text{ by our factorization}\end{aligned}$$

Thus, at least one $A - z_iI_{n\times n}$ has a null space, showing that there exists $\vec{v}$ with $A\vec{v} = z_i\vec{v}$, as needed. □

There is one additional fact worth checking to motivate our discussion of eigenvector computation:

Lemma 5.2 (CITE Proposition 2.2)**.** *Eigenvectors corresponding to different eigenvalues must be linearly independent.*

Proof. Suppose this is not the case. Then there exist eigenvectors $\vec{x}_1, \cdots, \vec{x}_k$ with distinct eigenvalues $\lambda_1, \cdots, \lambda_k$ that are linearly dependent. This implies that there are coefficients $c_1, \ldots, c_k$ not all zero with $\vec{0} = c_1\vec{x}_1 + \cdots + c_k\vec{x}_k$. If we premultiply by the matrix $(A - \lambda_2 I_{n\times n}) \cdots (A - \lambda_k I_{n\times n})$, we find:

$$\begin{aligned}\vec{0} &= (A - \lambda_2 I_{n\times n}) \cdots (A - \lambda_k I_{n\times n})(c_1\vec{x}_1 + \cdots + c_k\vec{x}_k) \\ &= c_1(\lambda_1 - \lambda_2) \cdots (\lambda_1 - \lambda_k)\vec{x}_1 \text{ since } A\vec{x}_i = \lambda_i\vec{x}_i\end{aligned}$$

Since all the λ_i's are distinct, this shows $c_1 = 0$. A similar proof shows that the rest of the c_i's have to be zero, contradicting linear dependence. □

This lemma shows that an $n \times n$ matrix can have at most n distinct eigenvalues, since a set of n eigenvalues yields n linearly independent vectors. The maximum number of linearly independent eigenvectors corresponding to a single eigenvalue λ is known as the *geometric multiplicity* of λ.

It is not true, however, that a matrix has to have *exactly* n linearly independent eigenvectors. This is the case for many matrices, which we will call *nondefective*:

Definition 5.3 (Nondefective)**.** *A matrix $A \in \mathbb{R}^{n\times n}$ is* nondefective *or* diagonalizable *if its eigenvectors span $\mathbb{R}^n$.*

We call such a matrix diagonalizable for the following reason: If a matrix is diagonalizable, then it has n eigenvectors $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^n$ with corresponding (possibly non-unique) eigenvalues $\lambda_1, \ldots, \lambda_n$. Take the columns of X to be the vectors $\vec{x}_i$, and define D to be the diagonal matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$ along the diagonal. Then, by definition of eigenvalues we have $AX = XD$; this is simply a "stacked" version of $A\vec{x}_i = \lambda_i\vec{x}_i$. In other words,

$$D = X^{-1}AX,$$

meaning A is diagonalized by a *similarity transformation* $A \mapsto X^{-1}AX$:

Definition 5.4 (Similar matrices)**.** *Two matrices A and B are* similar *if there exists T with $B = T^{-1}AT$.*

Similar matrices have the same eigenvalues, since if $B\vec{x} = \lambda x$, then $T^{-1}AT\vec{x} = \lambda\vec{x}$. Equivalently, $A(T\vec{x}) = \lambda(T\vec{x})$, showing $T\vec{x}$ is an eigenvector with eigenvalue λ.

5.3.1 Symmetric and Positive Definite Matrices

Unsurprisingly given our special consideration of normal matrices $A^\top A$, symmetric and/or positive definite matrices enjoy special eigenvector structure. If we can verify either of these properties, specialized algorithms can be used to extract their eigenvectors more quickly.

First, we can prove a property of symmetric matrices that obliterates the need for complex arithmetic. We begin by making a generalization of symmetric matrices to matrices in $\mathbb{C}^{n\times n}$:

Definition 5.5 (Complex conjugate)**.** *The* complex conjugate *of a number $z \equiv a + bi \in \mathbb{C}$ is $\bar{z} \equiv a - bi$.*

Definition 5.6 (Conjugate transpose)**.** *The* conjugate transpose *of* $A \in \mathbb{C}^{m\times n}$ *is* $A^H \equiv \bar{A}^\top$.

Definition 5.7 (Hermitian matrix)**.** *A matrix* $A \in \mathbb{C}^{n\times n}$ *is* Hermitian *if* $A = A^H$.

Notice that a symmetric matrix $A \in \mathbb{R}^{n\times n}$ is automatically Hermitian because it has no complex part. With this slight generalization in place, we can prove a symmetry property for eigenvalues. Our proof will make use of the dot product of vectors in $\mathbb{C}^n$, given by

$$\langle \vec{x}, \vec{y} \rangle = \sum_i x_i \bar{y}_i,$$

where $\vec{x}, \vec{y} \in \mathbb{C}^n$. Notice that once again this definition coincides with $\vec{x} \cdot \vec{y}$ when $\vec{x}, \vec{y} \in \mathbb{R}^n$. For the most part, properties of this inner product coincide with those of the dot product on $\mathbb{R}^n$, a notable exception being that $\langle \vec{v}, \vec{w} \rangle = \overline{\langle \vec{w}, \vec{v} \rangle}$.

Lemma 5.3. *All eigenvalues of Hermitian matrices are real.*

Proof. Suppose $A \in \mathbb{C}^{n\times n}$ is Hermitian with $A\vec{x} = \lambda\vec{x}$. By scaling we can assume $\|\vec{x}\|^2 = \langle \vec{x}, \vec{x} \rangle = 1$. Then, we have:

$$\begin{aligned}
\lambda &= \lambda \langle \vec{x}, \vec{x} \rangle \text{ since } \vec{x} \text{ has norm 1} \\
&= \langle \lambda\vec{x}, \vec{x} \rangle \text{ by linearity of } \langle \cdot, \cdot \rangle \\
&= \langle A\vec{x}, \vec{x} \rangle \text{ since } A\vec{x} = \lambda\vec{x} \\
&= (A\vec{x})^\top \vec{x} \text{ by definition of } \langle \cdot, \cdot \rangle \\
&= \vec{x}^\top \overline{(\bar{A}^\top \vec{x})} \text{ by expanding the product and using } \overline{ab} = \bar{a}\bar{b} \\
&= \langle \vec{x}, A^H \vec{x} \rangle \text{ by definition of } A^H \text{ and } \langle \cdot, \cdot \rangle \\
&= \langle \vec{x}, A\vec{x} \rangle \text{ since } A = A^H \\
&= \bar{\lambda} \langle \vec{x}, \vec{x} \rangle \text{ since } A\vec{x} = \lambda\vec{x} \\
&= \bar{\lambda} \text{ since } \vec{x} \text{ has norm 1}
\end{aligned}$$

Thus $\lambda = \bar{\lambda}$, which can happen only if $\lambda \in \mathbb{R}$, as needed. □

Symmetric and Hermitian matrices also enjoy a special orthogonality property for their eigenvectors:

Lemma 5.4. *Eigenvectors corresponding to distinct eigenvalues of Hermitian matrices must be orthogonal.*

Proof. Suppose $A \in \mathbb{C}^{n\times n}$ is Hermitian, and suppose $\lambda \neq \mu$ with $A\vec{x} = \lambda\vec{x}$ and $A\vec{y} = \mu\vec{y}$. By the previous lemma we know $\lambda, \mu \in \mathbb{R}$. Then, $\langle A\vec{x}, \vec{y} \rangle = \lambda \langle \vec{x}, \vec{y} \rangle$. But since A is Hermitian we can also write $\langle A\vec{x}, \vec{y} \rangle = \langle \vec{x}, A^H\vec{y} \rangle = \langle \vec{x}, A\vec{y} \rangle = \mu \langle \vec{x}, \vec{y} \rangle$. Thus, $\lambda \langle \vec{x}, \vec{y} \rangle = \mu \langle \vec{x}, \vec{y} \rangle$. Since $\lambda \neq \mu$, we must have $\langle \vec{x}, \vec{y} \rangle = 0$. □

Finally, we can state without proof a crowning result of linear algebra, the Spectral Theorem. This theorem states that no symmetric or Hermitian matrix can be defective, meaning that an $n \times n$ matrix satisfying this property has exactly n orthogonal eigenvectors.

Theorem 5.1 (Spectral Theorem)**.** *Suppose* $A \in \mathbb{C}^{n\times n}$ *is Hermitian (if* $A \in \mathbb{R}^{n\times n}$*, suppose it is symmetric). Then, A has exactly n orthonormal eigenvectors* $\vec{x}_1, \cdots, \vec{x}_n$ *with (possibly repeated) eigenvalues* $\lambda_1, \ldots, \lambda_n$ *. In other words, there exists an orthonormal matrix X of eigenvectors and diagonal matrix D of eigenvalues such that* $D = X^\top AX$.

This theorem implies that any vector $\vec{y} \in \mathbb{R}^n$ can be decomposed into a linear combination of the eigenvectors of a Hermitian matrix A. Many calculations are easier in this basis, as shown below:

Example 5.1 (Computation using eigenvectors)**.** *Take $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^n$ to be the unit-length eigenvectors of symmetric matrix $A \in \mathbb{R}^{n \times n}$. Suppose we wish to solve $A\vec{y} = \vec{b}$. We can write*

$$\vec{b} = c_1\vec{x}_1 + \cdots + c_n\vec{x}_n,$$

where $c_i = \vec{b} \cdot \vec{x}_i$ by orthonormality. It is easy to guess the following solution:

$$\vec{y} = \frac{c_1}{\lambda_1}\vec{x}_1 + \cdots + \frac{c_n}{\lambda_n}\vec{x}_n.$$

In particular, we find:

$$\begin{aligned} A\vec{y} &= A\left(\frac{c_1}{\lambda_1}\vec{x}_1 + \cdots + \frac{c_n}{\lambda_n}\vec{x}_n\right) \\ &= \frac{c_1}{\lambda_1}A\vec{x}_1 + \cdots + \frac{c_n}{\lambda_n}A\vec{x}_n \\ &= c_1\vec{x}_1 + \cdots + c_n\vec{x}_n \\ &= \vec{b}, \text{ as desired.} \end{aligned}$$

The calculation above is both a positive and negative result. It shows that given the eigenvectors of symmetric A, operations like inversion are straightforward. On the flip side, this means that finding the full set of eigenvectors of a symmetric matrix A is "at least" as difficult as solving $A\vec{x} = \vec{b}$.

Returning from our foray into the complex numbers, we return to real numbers to prove one final useful if straightforward fact about positive definite matrices:

Lemma 5.5. *All eigenvalues of positive definite matrices are nonnegative.*

Proof. Take $A \in \mathbb{R}^{n \times n}$ positive definite, and suppose $A\vec{x} = \lambda\vec{x}$ with $\|\vec{x}\| = 1$. By positive definiteness, we know $\vec{x}^\top A\vec{x} \geq 0$. But, $\vec{x}^\top A\vec{x} = \vec{x}^\top(\lambda\vec{x}) = \lambda\|\vec{x}\|^2 = \lambda$, as needed. □

5.3.2 Specialized Properties[1]

Characteristic Polynomial

Recall that the determinant of a matrix $\det A$ satisfies the relationship that $\det A \neq 0$ if and only if A is invertible. Thus, one way to find eigenvalues of a matrix is to find roots of the *characteristic polynomial*

$$p_A(\lambda) = \det(A - \lambda I_{n \times n}).$$

We will not define determinants in our discussion here, but simplifying p_A reveals that it is an n-th degree polynomial in λ. This provides an alternative reason why there are at most n distinct eigenvalues, since there are at most n roots of this function.

From this construction, we can define the *algebraic multiplicity* of an eigenvalue as its multiplicity as a root of p_A. It is easy to see that the algebraic multiplicity is at least as large as the geometric

[1] This section can be skipped if readers lack sufficient background but is included for completeness.

multiplicity. If the algebraic multiplicity is 1, the root is called *simple*, because it corresponds to a single eigenvector that is linearly dependent with any others. Eigenvalues for which the algebraic and geometric multiplicities are not equal are called *defective*.

In numerical analysis we avoid discussing the determinant of a matrix. While it is a convenient theoretical construction, its practical use is limited. Determinants are difficult to compute. In fact, eigenvalue algorithms do not attempt to find roots of p_A since doing so would require evaluation of a determinant. Furthermore, the determinant $\det A$ has *nothing* to do with the conditioning of A, so near-zero determinant of $\det(A - \lambda I_{n\times n})$ might not show that λ is nearly an eigenvalue of A.

Jordan Normal Form

We can only diagonalize a matrix when it has a full eigenspace. All matrices, however, are similar to a matrix in Jordan normal form, which has the following form:

- Nonzero values are on the diagonal entries a_{ii} and on the "superdiagonal" $a_{i(i+1)}$.
- Diagonal values are eigenvalues repeated as many times as their multiplicity; the matrix is block diagonal about these clusters.
- Off-diagonal values are 1 or 0.

Thus, the shape looks something like the following

$$\begin{pmatrix} \lambda_1 & 1 & & & & & \\ & \lambda_1 & 1 & & & & \\ & & \lambda_1 & & & & \\ & & & \lambda_2 & 1 & & \\ & & & & \lambda_2 & & \\ & & & & & \lambda_3 & \\ & & & & & & \ddots \end{pmatrix}$$

Jordan normal form is attractive theoretically because it always exists, but the $1/0$ structure is discrete and unstable under numerical perturbation.

5.4 Computing Eigenvalues

The computation and estimation of the eigenvalues of a matrix is a well-studied problem with many potential solutions. Each solution is tuned for a different situation, and achieving maximum conditioning or speed requires experimentation with several techniques. Here, we cover a few of the most popular and straightforward solutions to the eigenvalue problem frequently encountered in practice.

5.4.1 Power Iteration

For now, suppose that $A \in \mathbb{R}^{n\times n}$ is symmetric. Then, by the spectral theorem we can write eigenvectors $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^n$; we sort them such that their corresponding eigenvalues satisfy $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.

Suppose we take an arbitrary vector $\vec{v}$. Since the eigenvectors of A span $\mathbb{R}^n$, we can write:

$$\vec{v} = c_1\vec{x}_1 + \cdots + c_n\vec{x}_n.$$

Then,

$$\begin{aligned}
A\vec{v} &= c_1A\vec{x}_1 + \cdots + c_nA\vec{x}_n \\
&= c_1\lambda_1\vec{x}_1 + \cdots + c_n\lambda_n\vec{x}_n \text{ since } A\vec{x}_i = \lambda_i\vec{x}_i \\
&= \lambda_1\left(c_1\vec{x}_1 + \frac{\lambda_2}{\lambda_1}c_2\vec{x}_2 + \cdots + \frac{\lambda_n}{\lambda_1}c_n\vec{x}_n\right) \\
A^2\vec{v} &= \lambda_1^2\left(c_1\vec{x}_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^2 c_2\vec{x}_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^2 c_n\vec{x}_n\right) \\
&\vdots \\
A^k\vec{v} &= \lambda_1^k\left(c_1\vec{x}_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k c_2\vec{x}_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^k c_n\vec{x}_n\right)
\end{aligned}$$

Notice that as $k \to \infty$, the ratio $(\lambda_i/\lambda_1)^k \to 0$ unless $\lambda_i = \lambda_1$, since λ_1 has the largest magnitude of any eigenvalue by definition. Thus, if $\vec{x}$ is the projection of $\vec{v}$ onto the space of eigenvectors with eigenvalues λ_1, then as $k \to \infty$ the following approximation holds more and more exactly:

$$A^k\vec{v} \approx \lambda_1^k\vec{x}.$$

This observation leads to an exceedingly simple algorithm for computing an eigenvector $\vec{x}$ of A corresponding to the largest eigenvalue λ_1:

1. Take $\vec{v}_1 \in \mathbb{R}^n$ to be an arbitrary nonzero vector.
2. Iterate until convergence for increasing k:

$$\vec{v}_k = A\vec{v}_{k-1}$$

This algorithm, known as *power iteration*, will produce vectors $\vec{v}_k$ more and more parallel to the desired $\vec{x}_1$. It is guaranteed to converge, even when A is asymmetric, although the proof of this fact is more involved than the derivation above. The one time that this technique may fail is if we accidentally choose $\vec{v}_1$ such that $c_1 = 0$, but the odds of this occurring are slim to none.

Of course , if $|\lambda_1| > 1$, then $\|\vec{v}_k\| \to \infty$ as $k \to \infty$, an undesirable property for floating point arithmetic. Recall that we only care about the *direction* of the eigenvector rather than its magnitude, so scaling has no effect on the quality of our solution. Thus, to avoid this divergence situation we can simply normalize at each step, producing the *normalized power iteration* algorithm:

1. Take $\vec{v}_1 \in \mathbb{R}^n$ to be an arbitrary nonzero vector.
2. Iterate until convergence for increasing k:

$$\begin{aligned}
\vec{w}_k &= A\vec{v}_{k-1} \\
\vec{v}_k &= \frac{\vec{w}_k}{\|\vec{w}_k\|}
\end{aligned}$$

Notice that we did not decorate the norm $\|\cdot\|$ with a particular subscript. Mathematically, *any* norm will suffice for preventing the divergence issue, since we have shown that all norms on $\mathbb{R}^n$ are equivalent. In practice, we often use the *infinity* norm $\|\cdot\|_\infty$; in this case it is easy to check that $\|\vec{w}_k\| \to |\lambda_1|$.

5.4.2 Inverse Iteration

We now have a strategy for finding the *largest*-magnitude eigenvalue λ_1. Suppose A is invertible, so that we can evaluate $\vec{y} = A^{-1}\vec{v}$ by solving $A\vec{y} = \vec{v}$ using techniques covered in previous chapters.

If $A\vec{x} = \lambda\vec{x}$, then $\vec{x} = \lambda A^{-1}\vec{x}$, or equivalently

$$A^{-1}\vec{x} = \frac{1}{\lambda}\vec{x}.$$

Thus, we have shown that $1/\lambda$ is an eigenvalue of A^{-1} with eigenvector $\vec{x}$. Notice that if $|a| \geq |b|$ then $|b|^{-1} \geq |a|^{-1}$ for any $a, b \in \mathbb{R}$, so the smallest-magnitude eigenvalue of A is the *largest*-magnitude eigenvector of A^{-1}. This observation yields a strategy for finding λ_n rather than λ_1 called *inverse power iteration*:

1. Take $\vec{v}_1 \in \mathbb{R}^n$ to be an arbitrary nonzero vector.
2. Iterate until convergence for increasing k:
   (a) Solve for $\vec{w}_k$: $A\vec{w}_k = \vec{v}_{k-1}$
   (b) Normalize: $\vec{v}_k = \frac{\vec{w}_k}{\|\vec{w}_k\|}$

We repeatedly are solving systems of equations using the same matrix A, which is a perfect application of factorization techniques from previous chapters. For instance, if we write $A = LU$, then we could formulate an equivalent but considerably more efficient version of inverse power iteration:

1. Factor $A = LU$
2. Take $\vec{v}_1 \in \mathbb{R}^n$ to be an arbitrary nonzero vector.
3. Iterate until convergence for increasing k:
   (a) Solve for $\vec{y}_k$ by forward substitution: $L\vec{y}_k = \vec{v}_{k-1}$
   (b) Solve for $\vec{w}_k$ by back substitution: $U\vec{w}_k = \vec{y}_k$
   (c) Normalize: $\vec{v}_k = \frac{\vec{w}_k}{\|\vec{w}_k\|}$

5.4.3 Shifting

Suppose λ_2 is the eigenvalue with second-largest magnitude of A. Given our original derivation of power iteration, it is easy to see that power iteration converges fastest when $|\lambda_2/\lambda_1|$ is small, since in this case the power $(\lambda_2/\lambda_1)^k$ decays quickly. Contrastingly, if this ratio is nearly 1 it may take many iterations of power iteration before a single eigenvector is isolated.

If the eigenvalues of A are $\lambda_1, \ldots, \lambda_n$, then it is easy to see that the eigenvalues of $A - \sigma I_{n\times n}$ are $\lambda_1 - \sigma, \ldots, \lambda_n - \sigma$. Then, one strategy for making power iteration converge quickly is to choose σ such that:

$$\left|\frac{\lambda_2 - \sigma}{\lambda_1 - \sigma}\right| < \left|\frac{\lambda_2}{\lambda_1}\right|.$$

Of course, guessing such a σ can be an art, since the eigenvalues of A obviously are not known initially. Similarly, if we think that σ is near an eigenvalue of A, then $A - \sigma I_{n\times n}$ has an eigenvalue close to 0 that we can reveal by inverse iteration.

One strategy that makes use of this observation is known as *Rayleigh quotient iteration*. If we have a fixed guess at an eigenvector $\vec{x}$ of A, then by NUMBER the least-squares approximation of the corresponding eigenvalue σ is given by

$$\sigma \approx \frac{\vec{x}^\top A\vec{x}}{\|\vec{x}\|_2^2}.$$

This fraction is known as a Rayleigh quotient. Thus, we can attempt to increase convergence by iterating as follows:

1. Take $\vec{v}_1 \in \mathbb{R}^n$ to be an arbitrary nonzero vector or initial guess of an eigenvector.
2. Iterate until convergence for increasing k:
   (a) Write the current estimate of the eigenvalue
   $$\sigma_k = \frac{\vec{v}_{k-1}^\top A\vec{v}_{k-1}}{\|\vec{v}_{k-1}\|_2^2}$$
   (b) Solve for $\vec{w}_k$: $(A - \sigma_k I_{n\times n})\vec{w}_k = \vec{v}_{k-1}$
   (c) Normalize: $\vec{v}_k = \frac{\vec{w}_k}{\|\vec{w}_k\|}$

This strategy converges much faster given a good starting guess, but the matrix $A - \sigma_k I_{n\times n}$ is different each iteration and cannot be prefactored using LU or most other strategies. Thus, fewer iterations are necessary but each iteration takes more time!

5.4.4 Finding Multiple Eigenvalues

So far, we have described techniques for finding a single eigenvalue/eigenvector pair: power iteration to find the largest eigenvalue, inverse iteration to find the smallest, and shifting to target values in between. Of course, for many applications a single eigenvalue will not suffice. Thankfully we can extend our strategies to handle this case as well.

Deflation

Recall our power iteration strategy: Choose an arbitrary $\vec{v}_1$, and iteratively multiply it by A until only the largest eigenvalue λ_1 survives. Take $\vec{x}_1$ to be the corresponding eigenvector.

We were quick to dismiss an unlikely failure mode of this algorithm, however, when $\vec{v}_1 \cdot \vec{x}_1 = 0$. In this case, no matter how many times you premultiply by A you will never recover a vector

parallel to $\vec{x}_1$, since you cannot amplify a zero component. The probability of choosing such a $\vec{v}_1$ is exactly zero, so in all but the most pernicious of cases power iteration remains safe.

We can turn this drawback on its head to formulate a strategy for finding more than one eigenvalue when A is symmetric. Suppose we find $\vec{x}_1$ and λ_1 via power iteration as before. Now, we restart power iteration, but before beginning project $\vec{x}_1$ out of $\vec{v}_1$. Then, since the eigenvectors of A are orthogonal, power iteration will recover the *second*-largest eigenvalue!

Due to numerical issues, it may be the case that applying A to a vector introduces a small component parallel to $\vec{x}_1$. In practice we can avoid this effect by projecting in each iteration. In the end, this strategy yields the following algorithm for computing the eigenvalues in order of descending magnitude:

- For each desired eigenvalue $\ell = 1, 2, \ldots$
  1. Take $\vec{v}_1 \in \mathbb{R}^n$ to be an arbitrary nonzero vector.
  2. Iterate until convergence for increasing k:
     (a) Project out the eigenvectors we already have computed:
     $$\vec{u}_k = \vec{v}_{k-1} - \text{proj}_{\text{span}\{\vec{x}_1,\ldots,\vec{x}_{\ell-1}\}} \vec{v}_{k-1}$$
     (b) Multiply $A\vec{u}_k = \vec{w}_k$
     (c) Normalize: $\vec{v}_k = \frac{\vec{w}_k}{\|\vec{w}_k\|}$
  3. Add the result of iteration to the set of $\vec{x}_i$'s

The inner loop is equivalent to power iteration on the matrix AP, where P projects out $\vec{x}_1, \ldots, \vec{x}_{\ell-1}$. It is easy to see that AP has the same eigenvectors as A; its eigenvalues are $\lambda_\ell, \ldots, \lambda_n$ with the remaining eigenvalues taken to zero.

More generally, the strategy of *deflation* involves modifying the matrix A so that power iteration reveals an eigenvector you have not yet computed. For instance, AP is a modification of A so that the large eigenvalues we already have computed are zeroed out.

Our projection strategy fails if A is asymmetric, since in that case its eigenvectors may not be orthogonal. Other less obvious deflation strategies can work in this case. For instance, suppose $A\vec{x}_1 = \lambda_1\vec{x}_1$ with $\|\vec{x}_1\| = 1$. Take H to be the Householder matrix such that $H\vec{x}_1 = \vec{e}_1$, the first standard basis vector. Similarity transforms once again do not affect the set of eigenvectors, so we might try conjugating by H. Consider what happens when we multiply $HAH^\top$ by $\vec{e}_1$:

$$\begin{aligned} HAH^\top\vec{e}_1 &= HAH\vec{e}_1 \text{ since } H \text{ is symmetric} \\ &= HA\vec{x}_1 \text{ since } H\vec{x}_1 = \vec{e}_1 \text{ and } H^2 = I_{n\times n} \\ &= \lambda_1 H\vec{x}_1 \text{ since } A\vec{x}_1 = \lambda_1\vec{x}_1 \qquad\qquad = \lambda_1\vec{e}_1 \text{ by definition of } H \end{aligned}$$

Thus, the first column of $HAH^\top$ is $\lambda_1\vec{e}_1$, showing that $HAH^\top$ has the following structure (CITE HEATH):

$$HAH^\top = \begin{pmatrix} \lambda_1 & \vec{b}^\top \\ \vec{0} & B \end{pmatrix}.$$

The matrix $B \in \mathbb{R}^{(n-1)\times(n-1)}$ has eigenvalues $\lambda_2, \ldots, \lambda_n$. Thus, another strategy for deflation is to construct smaller and smaller B matrices with each eigenvalue computed using power iteration.

QR Iteration

Deflation has the drawback that we must compute each eigenvector separately, which can be slow and can accumulate error if individual eigenvalues are not accurate. Our remaining strategies attempt to find more than one eigenvector at a time.

Recall that similar matrices A and $B = T^{-1}AT$ must have the same eigenvalues. Thus, an algorithm attempting to find the eigenvalues of A can freely apply similarity transformations to A. Of course, applying T^{-1} in general may be a difficult proposition, since it effectively would require inverting T, so we seek T matrices whose inverses are easy to apply.

One of our motivators for deriving QR factorization was that the matrix Q is *orthogonal*, satisfying $Q^{-1} = Q^\top$. Thus, Q and Q^{-1} are equally straightforward to apply, making orthogonal matrices strong choices for similarity transformations.

But which orthogonal matrix Q should we choose? Ideally Q should involve the structure of A while being straightforward to compute. It is unclear how to apply simple transformations like Householder matrices strategically to reveal multiple eigenvalues,[2] but we do know how to generate one such Q simply by factoring $A = QR$. Then, we could conjugate A by Q to find:

$$Q^{-1}AQ = Q^\top AQ = Q^\top(QR)Q = (Q^\top Q)RQ = RQ$$

Amazingly, conjugating $A = QR$ by the orthogonal matrix Q is identical to writing the product RQ!

Based on this reasoning, in the 1950s, multiple groups of European mathematicians hypothesized the same elegant iterative algorithm for finding the eigenvalues of a matrix A:

1. Take $A_1 = A$.
2. For $k = 1, 2, \ldots$
   (a) Factor $A_k = Q_k R_k$.
   (b) Write $A_{k+1} = R_k Q_k$.

By our derivation above, the matrices A_k all have the same eigenvalues as A. Furthermore, suppose the A_k's converge to some A_∞. Then, we can factor $A_\infty = Q_\infty R_\infty$, and by convergence we know $A_\infty = Q_\infty R_\infty = R_\infty Q_\infty$. By NUMBER, the eigenvalues of R_∞ are simply the values along the diagonal of R_∞, and by NUMBER the product $R_\infty Q_\infty = A_\infty$ in turn must have the same eigenvalues. Finally, by construction A_∞ has the same eigenvalues as A. So, we have shown that *if* QR iteration converges, it reveals the eigenvalues of A in a straightforward way.

Of course, the derivation above assumes that there exists A_∞ with $A_k \to A_\infty$ as $k \to \infty$. In fact, QR iteration is a stable method guaranteed to converge in many important situations, and convergence can even be improved by shifting strategies. We will not derive exact conditions here but instead can provide some intuition for why this seemingly arbitrary strategy should converge. We provide some intuition below for the symmetric case $A = A^\top$, which is easier to analyze thanks to the orthogonality of eigenvectors in this case.

Suppose the columns of A are given by $\vec{a}_1, \ldots, \vec{a}_n$, and consider the matrix A^k for large k. We can write:

$$A^k = A^{k-1} \cdot A = \begin{pmatrix} | & | & & | \\ A^{k-1}\vec{a}_1 & A^{k-1}\vec{a}_2 & \cdots & A^{k-1}\vec{a}_n \\ | & | & & | \end{pmatrix}$$

[2] More advanced techniques, however, do exactly this!

By our derivation of power iteration, the first column of A^k in general is parallel to the eigenvector $\vec{x}_1$ with largest magnitude $|\lambda_1|$ since we took a vector $\vec{a}_1$ and multiplied it by A many times. Applying our intuition from deflation, suppose we project $\vec{a}_1$ out of the second column of A^k. This vector must be nearly parallel to $\vec{x}_2$, since it is the *second*-most dominant eigenvalue! Proceeding inductively, factoring $A^k = QR$ would yield a set of near-eigenvectors as the columns of Q, in order of decreasing eigenvalue magnitude, with the corresponding eigenvalues along the diagonal of R.

Of course, computing A^k for large k takes the condition number of A to the k-th power, so QR on the resulting matrix is likely to fail; this is clear to see since *all* the columns of A^k should look like $\vec{x}_1$ for large k. We can make the following observation, however:

$$\begin{aligned}
A &= Q_1 R_1 \\
A^2 &= (Q_1 R_1)(Q_1 R_1) \\
&= Q_1 (R_1 Q_1) R_1 \\
&= Q_1 Q_2 R_2 R_1 \text{ using the notation of QR iteration above, since } A_2 = R_1 Q_1 \\
&\vdots \\
A^k &= Q_1 Q_2 \cdots Q_k R_k R_{k-1} \cdots R_1
\end{aligned}$$

Grouping the Q_i variables and the R_i variables separately provides a QR factorization of A^k. Thus, we expect the columns of $Q_1 \cdots Q_k$ to converge to the eigenvectors of A.

By a similar argument, we can find

$$\begin{aligned}
A &= Q_1 R_1 \\
A_2 &= R_1 Q_1 \text{ by our construction of QR iteration} \\
&= Q_2 R_2 \text{ by definition of the factorization} \\
A_3 &= R_2 Q_2 \text{ from QR iteration} \\
&= Q_2^\top A_2 Q_2 \text{ since } A_2 = Q_2 R_2 \\
&= Q_2^\top R_1 Q_1 Q_2 \text{ since } A_2 = R_1 Q_1 \\
&= Q_2^\top Q_1^\top A Q_1 Q_2 \text{ since } A = Q_1 R_1 \\
&\vdots \\
A_{k+1} &= Q_k^\top \cdots Q_1^\top A Q_1 \cdots Q_k \text{ inductively} \\
&= (Q_1 \cdots Q_k)^\top A (Q_1 \cdots Q_k)
\end{aligned}$$

where A_k is the k-th matrix from QR iteration. Thus, A_{k+1} is simply the matrix A conjugated by the product $\bar{Q}_k \equiv Q_1 \cdots Q_k$. We argued earlier that the columns of $\bar{Q}_k$ converge to the eigenvectors of A. Thus, since conjugating by the matrix of eigenvectors yields a diagonal matrix of eigenvalues, we know $A_{k+1} = \bar{Q}_k^\top A \bar{Q}$ will have approximate eigenvalues of A along its diagonal as $k \to \infty$.

Krylov Subspace Methods

Our justification of QR iteration involved analyzing the columns of A^k as $k \to \infty$ as an extension of power iteration. More generally, for a vector $\vec{b} \in \mathbb{R}^n$, we can examine the so-called *Krylov matrix*

$$K_k = \begin{pmatrix} | & | & | & & | \\ \vec{b} & A\vec{b} & A^2\vec{b} & \cdots & A^{k-1}\vec{b} \\ | & | & | & & | \end{pmatrix}.$$

Methods analyzing K_k to find eigenvectors and eigenvalues generally are known as *Krylov subspace methods*. For instance, the *Arnoldi iteration* algorithm uses Gram-Schmidt orthogonalization to maintain an orthogonal basis $\{\vec{q}_1, \ldots, \vec{q}_k\}$ for the column space of K_k:

1. Begin by taking $\vec{q}_1$ to be an arbitrary unit-norm vector
2. For $k = 2, 3, \ldots$
   (a) Take $\vec{a}_k = A\vec{q}_{k-1}$
   (b) Project out the $\vec{q}$'s you already have computed:
   $$\vec{b}_k = \vec{a}_k - \text{proj}_{\text{span}\{\vec{q}_1,\ldots,\vec{q}_{k-1}\}}\vec{a}_k$$
   (c) Renormalize to find the next $\vec{q}_k = \vec{b}_k / \|\vec{b}_k\|$.

The matrix Q_k whose columns are the vectors found above is an orthogonal matrix with the same column space as K_k, and eigenvalue estimates can be recovered from the structure of $Q_k^\top A Q_k$. The use of Gram-Schmidt makes this technique unstable and timing gets progressively worse as k increases, however, so many extensions are needed to make it feasible. For instance, one strategy involves running some iterations of Arnoldi, using the output to generate a better guess for the initial $\vec{q}_1$, and restarting. Methods in this class are suited for problems requiring multiple eigenvectors at one of the ends of the spectrum without computing the complete set.

5.5 Sensitivity and Conditioning

As warned, we have only outlined a few eigenvalue techniques out of a rich and long-standing literature. Almost any algorithmic technique has been experimented with for finding spectra, from iterative methods to root-finding on the characteristic polynomial to methods that divide matrices into blocks for parallel processing.

Just as in linear solvers, we can evaluate the conditioning of an eigenvalue *problem* independently of the solution technique. This analysis can help understand whether a simplistic iterative scheme will be successful for finding the eigenvectors of a given matrix or if more complex methods are necessary; it is important to note that the conditioning of an eigenvalue problem is *not* the same as the condition number of the matrix for solving systems, since these are separate problems.

Suppose a matrix A has an eigenvector $\vec{x}$ with eigenvalue λ. Analyzing the conditioning of the eigenvalue problem involves analyzing the stability of $\vec{x}$ and λ to perturbations in A. To this end, we might perturb A by a small matrix δA, thus changing the set of eigenvectors. In particular, we can write eigenvectors of $A + \delta A$ as perturbations of eigenvectors of A by solving the problem

$$(A + \delta A)(\vec{x} + \delta\vec{x}) = (\lambda + \delta\lambda)(\vec{x} + \delta\vec{x}).$$

Expanding both sides yields:

$$A\vec{x} + A\delta\vec{x} + \delta A \cdot \vec{x} + \delta A \cdot \delta\vec{x} = \lambda\vec{x} + \lambda\delta\vec{x} + \delta\lambda \cdot \vec{x} + \delta\lambda \cdot \delta\vec{x}$$

Assuming δA is small, we will assume[3] that $\delta\vec{x}$ and $\delta\lambda$ also are small. Products between these variables then are negligible, yielding the following approximation:

$$A\vec{x} + A\delta\vec{x} + \delta A \cdot \vec{x} \approx \lambda\vec{x} + \lambda\delta\vec{x} + \delta\lambda \cdot \vec{x}$$

Since $A\vec{x} = \lambda\vec{x}$, we can subtract this value from both sides to find:

$$A\delta\vec{x} + \delta A \cdot \vec{x} \approx \lambda\delta\vec{x} + \delta\lambda \cdot \vec{x}$$

We now apply an analytical trick to complete our derivation. Since $A\vec{x} = \lambda\vec{x}$, we know $(A - \lambda I_{n\times n})\vec{x} = \vec{0}$, so $A - \lambda I_{n\times n}$ is not full rank. The transpose of a matrix is full-rank only if the matrix is full-rank, so we know $(A - \lambda I_{n\times n})^\top = A^\top - \lambda I_{n\times n}$ also has a null space vector $\vec{y}$. Thus $A^\top\vec{y} = \lambda\vec{y}$; we can call $\vec{y}$ the *left* eigenvector corresponding to $\vec{x}$. We can left-multiply our perturbation estimate above by $\vec{y}^\top$:

$$\vec{y}^\top(A\delta\vec{x} + \delta A \cdot \vec{x}) \approx \vec{y}^\top(\lambda\delta\vec{x} + \delta\lambda \cdot \vec{x})$$

Since $A^\top\vec{y} = \lambda\vec{y}$, we can simplify:

$$\vec{y}^\top\delta A \cdot \vec{x} \approx \delta\lambda\vec{y}^\top\vec{x}$$

Rearranging yields:

$$\delta\lambda \approx \frac{\vec{y}^\top(\delta A)\vec{x}}{\vec{y}^\top\vec{x}}$$

Assume $\|\vec{x}\| = 1$ and $\|\vec{y}\| = 1$. Then, if we take norms on both sides we find:

$$|\delta\lambda| \lessapprox \frac{\|\delta A\|_2}{|\vec{y} \cdot \vec{x}|}$$

So in general conditioning of the eigenvalue problem depends on the size of the perturbation δA– as expected–and the angle between the left and right eigenvectors $\vec{x}$ and $\vec{y}$. We can use $1/\vec{x}\cdot\vec{y}$ as an approximate condition number. Notice that $\vec{x} = \vec{y}$ when A is symmetric, yielding a condition number of 1; this reflects the fact that the eigenvectors of symmetric matrices are orthogonal and thus maximally separated.

5.6 Problems

[3] This assumption should be checked in a more rigorous treatment!