St. Francis Xavier University
Department of Computer Science

CSCI 554: Matrix Computation
Lecture 9: Eigenvalues and Eigenvectors II
Winter 2022

1 Another Special Matrix

In one of our earliest lectures, we studied two special forms of matrices: triangular matrices and positive definite matrices. More recently, we also introduced the notion of similar matrices. In our last lecture, we observed that it's very easy to find the eigenvalues of a given matrix if that matrix is triangular, and so with our knowledge of similar matrices, it would be beneficial to us if we had a method of reducing an arbitrary matrix to a triangular matrix by way of a similarity transformation.

Unfortunately, in the general case, there is no method that always reduces an arbitrary matrix to one in triangular form in a finite number of steps; this is a consequence of Abel's theorem, which established that there exists no general formula to compute the roots of a polynomial equation of degree $n \geq 5$. Fortunately for us, however, we can develop a way of transforming a matrix to one that is "almost" in triangular form, and we can then apply an iterative method to such a matrix to find its eigenvalues. This approximately-triangular form of the matrix is known as its *Hessenberg form*, named for the German mathematician Karl Hessenberg, who first introduced the notion.

1.1 Hessenberg Matrices

We say that an $n \times n$ matrix A is an *upper Hessenberg matrix* if $a_{ij} = 0$ whenever $i > (j + 1)$. Note that this condition is just slightly looser than the triangular matrix condition that $a_{ij} = 0$ for all $i > j$. Thus, an upper Hessenberg matrix has the general form

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1(n-2)} & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & a_{23} & & a_{2(n-2)} & a_{2(n-1)} & a_{2n} \\ 0 & a_{32} & a_{33} & & a_{3(n-2)} & a_{3(n-1)} & a_{3n} \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & 0 & & a_{(n-1)(n-2)} & a_{(n-1)(n-1)} & a_{(n-1)n} \\ 0 & 0 & 0 & \cdots & 0 & a_{n(n-1)} & a_{nn} \end{bmatrix},$$

where the subdiagonal immediately below the main diagonal is nonzero, and all entries below that subdiagonal are zero.

In the following sections, we will investigate an algorithm whose performance can be improved if we're given as input an upper Hessenberg matrix. The natural question, then, is how can we transform an arbitrary matrix to an equivalent upper Hessenberg matrix?

As it turns out, the Hessenberg transformation method is quite similar to our method for computing a QR decomposition using reflection matrices. Suppose we're given a general $n \times n$ matrix A. We begin by partitioning A into four quadrants:

$$A = \begin{bmatrix} a_{11} & c^\intercal \\ b & \widehat{A} \end{bmatrix}$$

Here, a_{11} is a single value, b and c are vectors each with $(n - 1)$ components, and $\widehat{A}$ is the bottom-right $(n - 1) \times (n - 1)$ submatrix of A.

Now, take $\widehat{Q}_1$ to be a reflection matrix such that $\widehat{Q}_1 b = \begin{bmatrix}-\tau_1 & 0 & \dots & 0\end{bmatrix}^\intercal$, where $|\tau_1| = \|b\|_2$. Furthermore, let

$$Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & \widehat{Q}_1 \end{bmatrix}.$$

With this, we compute

$$A_{1/2} = Q_1 A = \left[\begin{array}{c|c} a_{11} & c^\intercal \\ \hline -\tau_1 & \\ 0 & \\ \vdots & \widehat{Q}_1 \widehat{A} \\ 0 & \end{array}\right],$$

which has the correct amount of zero entries in the first column. (Recall that an upper Hessenberg matrix has zero entries at all entries a_{ij} when $i > (j+1)$.) Essentially, what we've done in this step is the first step of a QR decomposition, except we left two nonzero entries in the first column rather than one.

We will then take this matrix and apply a similarity transformation. Let $A_1 = Q_1 A Q_1^{-1}$. Since $Q_1^{-1} = Q_1$, we have that

$$A_1 = A_{1/2} Q_1 = \left[\begin{array}{c|ccc} a_{11} & * & \cdots & * \\ \hline -\tau_1 & & & \\ 0 & & & \\ \vdots & & \widehat{Q}_1 \widehat{A} \widehat{Q}_1 & \\ 0 & & & \end{array}\right] = \left[\begin{array}{c|ccc} a_{11} & * & \cdots & * \\ \hline -\tau_1 & & & \\ 0 & & & \\ \vdots & & \widehat{A}_1 & \\ 0 & & & \end{array}\right],$$

where $*$ denotes some nonzero entries.

We now zero out entries in the second column of A_1 by taking a reflection matrix $\widehat{Q}_2$ in much the same way as before. Then, let

$$Q_2 = \left[\begin{array}{cc|ccc} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & 0 \\ \hline 0 & 0 & & & \\ \vdots & & & \widehat{Q}_2 & \\ 0 & 0 & & & \end{array}\right].$$

With this, we compute

$$A_{3/2} = Q_2 A_1 = \left[\begin{array}{cc|ccc} a_{11} & * & * & \cdots & * \\ -\tau_1 & * & * & \cdots & * \\ \hline 0 & -\tau_2 & & & \\ \vdots & \vdots & & \widehat{Q}_2 \widehat{A}_1 & \\ 0 & 0 & & & \end{array}\right],$$

and we then apply another similarity transformation to get

$$A_2 = A_{3/2} Q_2 = \left[\begin{array}{cc|ccc} a_{11} & * & * & \cdots & * \\ -\tau_1 & * & * & \cdots & * \\ \hline 0 & -\tau_2 & & & \\ \vdots & \vdots & & \widehat{Q}_2 \widehat{A}_1 \widehat{Q}_2 & \\ 0 & 0 & & & \end{array}\right] = \left[\begin{array}{cc|ccc} a_{11} & * & * & \cdots & * \\ -\tau_1 & * & * & \cdots & * \\ \hline 0 & -\tau_2 & & & \\ \vdots & \vdots & & \widehat{A}_2 & \\ 0 & 0 & & & \end{array}\right].$$

After repeating this procedure for a total of $(n-2)$ iterations, we obtain a matrix B in upper Hessenberg form that is similar to A; specifically, $B = Q^* A Q$, where $Q = Q_1 Q_2 \cdots Q_{n-2}$ and $Q^* = Q_{n-2} Q_{n-1} \cdots Q_1$, Q is orthogonal, and Q^* denotes the *conjugate transpose* of Q (i.e., the transpose of a complex-valued matrix).

Translating this procedure into pseudocode, we end up with an algorithm that greatly resembles our QR decomposition procedure. The major difference between our QR decomposition procedure and our present method is that we must multiply by reflector matrices on both the left and the right.

Algorithm 1: Upper Hessenberg matrix transformation

1: $B \leftarrow A$
2: **for** $1 \leq k \leq (n-2)$ **do**
3: $\quad \beta \leftarrow \max\{|a_{ik}| \mid (k+1) \leq i \leq n\}$
4: $\quad \gamma_k \leftarrow 0$
5: $\quad$ **if** $\beta \neq 0$ **then**
6: $\quad\quad A_{(k+1..n),k} \leftarrow \beta^{-1} A_{(k+1..n),k}$ ▷ "Part 1": lines 6–14
7: $\quad\quad \tau_k \leftarrow \sqrt{A_{k+1,k}^2 + \cdots + A_{n,k}^2}$
8: $\quad\quad$ **if** $A_{k+1,k} < 0$ **then**
9: $\quad\quad\quad \tau_k \leftarrow -\tau_k$
10: $\quad\quad \eta \leftarrow A_{k+1,k} + \tau_k$
11: $\quad\quad A_{k+1,k} \leftarrow 1$
12: $\quad\quad A_{(k+2..n),k} \leftarrow A_{(k+2..n),k} \,/\, \eta$
13: $\quad\quad \gamma_k \leftarrow \eta \,/\, \tau_k$
14: $\quad\quad \tau_k \leftarrow \tau_k \beta$
15: $\quad\quad B^{\intercal}_{(k+1..n),1} \leftarrow A^{\intercal}_{(k+1..n),k} \, A_{(k+1..n),(k+1..n)}$ ▷ "Part 2": lines 15–17
16: $\quad\quad B^{\intercal}_{(k+1..n),1} \leftarrow -\gamma_k B^{\intercal}_{(k+1..n),1}$
17: $\quad\quad A_{(k+1..n),(k+1..n)} \leftarrow A_{(k+1..n),(k+1..n)} + A_{(k+1..n),k} B^{\intercal}_{(k+1..n),1}$
18: $\quad\quad B_{(1..n),1} \leftarrow A_{(1..n),(k+1..n)} A_{(k+1..n),k}$ ▷ "Part 3": lines 18–20
19: $\quad\quad B_{(1..n),1} \leftarrow -\gamma_k B_{(1..n),1}$
20: $\quad\quad A_{(1..n),(k+1..n)} \leftarrow A_{(1..n),(k+1..n)} + B_{(1..n),1} A^{\intercal}_{(k+1..n),k}$
21: $\quad\quad A_{k+1,k} \leftarrow -\tau_k$
22: $\tau_{n-1} \leftarrow -A_{n,n-1}$
23: **return** B ▷ $B = Q^{\intercal} A Q$

Although this algorithm looks quite complicated, we can see more clearly what it's doing by breaking it down into three major "parts": lines 6–14 are used to construct the reflector matrix $\widehat{Q}_k$, lines 15–17 perform multiplication on the left by $\widehat{Q}_k$, and lines 18–20 perform multiplication on the right by $\widehat{Q}_k$.

Counting the number of flops used by this algorithm, we get that the first part (constructing the reflector matrix) takes about $O(n^2)$ flops, the second part (multiplication on the left) takes approximately $\frac{4}{3}n^3$ flops, and the third part (multiplication on the right) takes approximately $2n^3$ flops. Thus, this algorithm altogether uses on the order of $\frac{10}{3}n^3$ flops.

Of course, it is possible to improve on the performance of this algorithm by employing, for example, block matrix computations. In addition, if we know in advance that the input matrix A is *Hermitian*—essentially, the complex version of a symmetric matrix—then the number of flops used in the transformation process can be reduced to $\frac{4}{3}n^3$.

2 The QR Algorithm

We now have enough background knowledge to finally introduce our method of finding the eigenvalues of an arbitrary matrix A. Interestingly enough, this method is essentially an iterated application of an earlier concept we learned in a past lecture: the QR decomposition of a matrix! Since we cannot directly compute the roots of a polynomial of degree 5 or greater by Abel's theorem, we must resort to using such iterative methods to find the eigenvalues of matrices of dimension 5×5 or greater.

Inconveniently, the name of the method we are about to study is the *QR algorithm*. Note that, even though the QR algorithm makes use of QR decompositions, the two methods are in no way related beyond that; the QR algorithm is strictly devoted to computing eigenvalues, while the QR decomposition is the decomposition of one matrix into two special matrices. In a sense, the QR decomposition is more "primitive" than the QR algorithm.

So, how can we use QR decompositions to find eigenvalues? Suppose we have a matrix $A \in \mathbb{C}^{n \times n}$, and assume that A is invertible. At a high level, the QR algorithm proceeds as follows:

1. Let $A_0 = A$.
2. On the ith iteration:
   (a) Compute the QR decomposition $A_{i-1} = Q_i R_i$.
   (b) Reverse this matrix product to obtain $R_i Q_i = A_i$.
3. Repeat until the matrix converges to an upper triangular form, and obtain the eigenvalues of the matrix from the diagonal entries.

That's all we need to do! This process works because, for each matrix A_i, we have that $A_i = Q_i^* A_{i-1} Q_i$, which means that A_i is similar to A_{i-1}. Since we know that similar matrices share the same eigenvalues, the matrix we obtain through repeated iterations of the QR algorithm must have the same eigenvalues as the original matrix A. Moreover, since we know that the eigenvalues of an upper triangular matrix appear along its diagonal, converging to a matrix of this form allows us to read off the eigenvalues directly.

The pseudocode for the QR algorithm is a straightforward translation of our high-level description:

Algorithm 2: QR algorithm

```
A_0 ← A
for 1 ≤ i ≤ n do                          ▷ n is the desired number of iterations
    Q_i, R_i ← QR decomposition of A_{i-1}
    A_i ← R_i Q_i
return A_n
```

Note that our assumption that A is invertible simply ensures that the decomposition $A_{i-1} = Q_i R_i$ can be represented uniquely, where the diagonal entries are all positive. However, we need not necessarily maintain this property, so we can drop this condition in the general case.

Example 1. Consider the matrix

$$A = \begin{bmatrix} 8 & 2 \\ 2 & 5 \end{bmatrix}.$$

We can see that the eigenvalues of this matrix are $\lambda_1 = 9$ and $\lambda_2 = 4$. However, if we didn't know this, we could employ the QR algorithm to find these eigenvalues.

Let $A_0 = A$, and perform a QR decomposition to get $A_0 = Q_1 R_1$, where

$$Q_1 = \begin{bmatrix} -0.97 & -0.24 \\ -0.24 & 0.97 \end{bmatrix} \text{ and } R_1 = \begin{bmatrix} -8.24 & -3.15 \\ 0 & 4.36 \end{bmatrix}.$$

Now, reverse these factors to compute

$$A_1 = R_1 Q_1 = \begin{bmatrix} 8.76 & -1.05 \\ -1.05 & 4.23 \end{bmatrix}.$$

Already, we can see that the values along the diagonal of this matrix have gotten closer to the true eigenvalues.

Taking repeated QR decompositions of our A_i matrices, we get the following:

$$A_2 = R_2Q_2 = \begin{bmatrix} 8.951 & 0.489 \\ 0.489 & 4.048 \end{bmatrix},$$
$$A_3 = R_3Q_3 = \begin{bmatrix} 8.9904 & -0.2191 \\ -0.2191 & 4.0096 \end{bmatrix},$$
$$A_4 = R_4Q_4 = \begin{bmatrix} 8.99809 & 0.09750 \\ 0.09750 & 4.00190 \end{bmatrix}, \text{ and}$$
$$A_5 = R_5Q_5 = \begin{bmatrix} 8.999624 & -0.043351 \\ -0.043351 & 4.000376 \end{bmatrix}.$$

With enough iterations, the entries along the diagonal of our A_i matrix converge to the eigenvalues of A, while the off-diagonal entries converge to zero.

How well does our QR algorithm perform? As is evident, the bottleneck comes from the sheer number of QR decompositions we must perform as intermediate steps. Each of these QR decompositions uses $\frac{4}{3}n^3$ flops, and the resultant matrix multiplication to obtain the matrix A_i also takes $O(n^3)$ flops. In addition to this, the convergence rate is rather slow, meaning we must repeat the QR decomposition and matrix multiplication steps potentially many times before we settle on an answer. While it is possible to accelerate the convergence rate, we will place our focus here on the QR decomposition bottleneck to reduce the cost of this step specifically.

2.1 Combining QR and Hessenberg

Earlier in this lecture, we saw a method for converting an arbitrary $n \times n$ matrix into a matrix in upper Hessenberg form. As it turns out, having a matrix in such a form can greatly improve the time it takes to compute a QR decomposition.

One very nice property of upper Hessenberg matrices that makes this form amenable to use in QR decompositions is the fact that the form is preserved after applying the decomposition method.

Theorem 2. *Let A_{i-1} be a matrix in upper Hessenberg form, and suppose that A_i is obtained from A_{i-1} following a QR decomposition. Then A_i is also in upper Hessenberg form.*

Proof. Observe that we can rearrange the equation $A_{i-1} = Q_iR_i$ to be of the form $Q_i = A_{i-1}R_i^{-1}$. We know from linear algebra that if the inverse of an upper triangular matrix exists, then the inverse is itself upper triangular. Thus, R_i^{-1} is an upper triangular matrix. One can also show that the product of an upper Hessenberg matrix with an upper triangular matrix is, again, an upper Hessenberg matrix. Thus, Q_i is an upper Hessenberg matrix, and as a result, $A_i = R_iQ_i$ must also be an upper Hessenberg matrix. □

Now, suppose we have an upper Hessenberg matrix A, and we wish to apply the QR decomposition to A. We can transform A to a matrix in upper triangular form by applying $(n-1)$ rotation matrices to reduce the $(n-1)$ subdiagonal entries to zero. We do this iteratively: first, we find a rotation matrix Q_1 where the product Q_1^*A has a zero entry at index $(2,1)$; in this way, we are only altering entries in the first and second rows of A. Next, we find a rotation matrix Q_2 where the product $Q_2^*Q_1^*A$ has a zero entry at index $(3,2)$; this ensures that only the second and third rows of Q_1^*A are changed. Since the only entries in the intersection of these rows with the first column are zeroes, those zeroes are preserved, and specifically the zero at index $(2,1)$ is preserved.

We continue this process for all rows and columns until we arrive at an upper triangular matrix

$$R = Q_{n-1}^* \cdots Q_2^*Q_1^*A.$$

We then take $Q = Q_1Q_2 \cdots Q_{n-1}$, which gives us $R = Q^*A$ or, equivalently, $A = QR$.

To complete one step of the QR algorithm, we take these matrices Q and R and use them to compute $A_1 = RQ$. As a consequence of how we computed the matrix Q, we can obtain A_1 simply by multiplying R

on the right by each of the rotation matrices Q_1 through Q_{n-1}. Applying Q_1 recombines the first and second columns of the matrix, and the only zero in these columns that will be modified is the one located at index $(2, 1)$. Likewise, applying Q_2 modifies the zero at index $(3, 2)$, and so on. Thus, the process of computing A_1 creates nonzero entries along the subdiagonal, and so A_1 is an upper Hessenberg matrix.

Most importantly, the construction technique we just outlined gives us a noticeable improvement on the time required to compute a single QR decomposition.

Theorem 3. *Computing the QR decomposition of an upper Hessenberg matrix requires no more than $O(n^2)$ flops.*

Proof. Following our construction, we see that we can obtain the matrix A_i from A_{i-1} by multiplying A_{i-1} on the left by $(n-1)$ rotation matrices and multiplying R_i on the right by $(n-1)$ rotation matrices. The cost of each multiplication by a single rotation matrix is $O(n)$ flops, and so the total number of flops used is $O(n^2)$. □

Putting everything together, we see that reducing our arbitrary input matrix to a matrix in upper Hessenberg form requires $O(n^3)$ flops, but we need only perform this step once at the beginning of our computation. For the remainder, we can iteratively apply our QR decomposition method to this upper Hessenberg matrix at an improved cost of $O(n^2)$ flops per iteration.

3 The Singular Value Decomposition (Redux)

The singular values of a matrix A are closely related to A's eigenvalues and eigenvectors. In a previous lecture, we introduced the statement of the singular value decomposition theorem and investigated some applications of the decomposition, but we didn't prove the theorem itself or give a method of computing the decomposition. In this section, we finally return to these matters, now that we have a sufficient knowledge of eigenvalues and eigenvectors.

For our coming discussion, we will need to know how to describe two properties of a matrix $A \in \mathbb{R}^{n \times m}$.

- The *column space* of A, sometimes also called the *range* and denoted $C(A)$, is the set of all linear combinations of column vectors of A; that is, the span of the columns of A. In other terms,

$$C(A) = \{Ax \mid x \in \mathbb{R}^m\}.$$

- The *null space* of A, denoted nullity(A), is the set of all m-dimensional column vectors x such that $Ax = 0$. In other terms,

$$\text{nullity}(A) = \{x \in \mathbb{R}^m \mid Ax = 0\}.$$

The dimension of the column space is called the *rank* of A, denoted rank(A). From this, we obtain an important result known as the *rank-nullity theorem*: $\text{rank}(A) + \text{nullity}(A) = m$.

As we discuss the singular value decomposition again, we will find ourselves often referring to two particular matrix products: $AA^\intercal \in \mathbb{R}^{n \times n}$ and $A^\intercal A \in \mathbb{R}^{m \times m}$. Thus, before we continue, we'll briefly review a few properties of these matrix products.

Proposition 4. *Let $A \in \mathbb{R}^{n \times m}$ be a matrix. Then each of the following properties hold:*

1. $\text{rank}(AA^\intercal) = \text{rank}(A^\intercal A) = \text{rank}(A) = \text{rank}(A^\intercal)$.
2. $\text{nullity}(A^\intercal A) = \text{nullity}(A)$.
3. $\text{nullity}(AA^\intercal) = \text{nullity}(A^\intercal)$.

Furthermore, we can establish some properties relating to the eigenvalues and eigenvectors of the matrix A and its transpose.

Proposition 5. *Let $A \in \mathbb{R}^{n \times m}$ be a matrix. Then:*

1. *$AA^\intercal$ and $A^\intercal A$ have the same (potentially repeated) nonzero eigenvalues;*
2. *if v is an eigenvector of $A^\intercal A$ associated with a nonzero eigenvalue λ, then Av is an eigenvector of $AA^\intercal$ also associated with λ; and*
3. *if v_1 and v_2 are orthogonal eigenvectors of $A^\intercal A$, then Av_1 and Av_2 are also orthogonal eigenvectors of $AA^\intercal$.*

Finally, since we know that both $AA^\intercal$ and $A^\intercal A$ are symmetric, we can make use of the following result.

Proposition 6. *Let $B \in \mathbb{R}^{n \times n}$ be a symmetric matrix having eigenvectors v_i and v_j associated with the eigenvalues λ_i and λ_j, respectively, where $\lambda_i \neq \lambda_j$. Then v_i and v_j are orthogonal.*

Having stated each of these results, we will put them to use in our proof of the SVD theorem that we introduced in an earlier lecture.

Theorem 7 (Geometric SVD theorem)**.** *Let $A \in \mathbb{R}^{n \times m}$ be a nonzero real-valued matrix where* $\operatorname{rank}(A) = r$*. Then $\mathbb{R}^m$ has an orthonormal basis $\{v_1, \ldots, v_m\}$, $\mathbb{R}^n$ has an orthonormal basis $\{u_1, \ldots, u_n\}$, and there exist values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ such that*

$$Av_i = \begin{cases} \sigma_i u_i & \text{for } 1 \leq i \leq r; \\ 0 & \text{for } (r+1) \leq i \leq m, \end{cases} \quad \text{and } A^\intercal u_i = \begin{cases} \sigma_i v_i & \text{for } 1 \leq i \leq r; \\ 0 & \text{for } (r+1) \leq i \leq n. \end{cases}$$

Each of v_1 through v_m are eigenvectors of $A^\intercal A$, each of u_1 through u_n are eigenvectors of $AA^\intercal$, and each of σ_1^2 through σ_r^2 are nonzero eigenvalues of both $AA^\intercal$ and $A^\intercal A$.

Proof. Let $\{v_1, \ldots, v_m\}$ be an orthonormal basis of $\mathbb{R}^m$ consisting of eigenvectors of $A^\intercal A$, and let λ_1 through λ_m be the associated eigenvalues. Assume that the eigenvectors are ordered so that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$. Since $A^\intercal A$ is a positive semidefinite matrix, we know that all of these eigenvalues are nonnegative real numbers, and so we can find such an ordering.

Since $r = \operatorname{rank}(A^\intercal A)$, we must have that $\lambda_r > 0$ while $\lambda_{r+1} = \cdots = \lambda_m = 0$. For each $1 \leq i \leq r$, take

$$\sigma_i = \|Av_i\|_2 \quad \text{and} \quad u_i = \frac{1}{\sigma_i} Av_i.$$

From this, we can see that $Av_i = \sigma_i u_i$ and $\|u_i\|_2 = 1$ for all $1 \leq i \leq r$. As a consequence of the third property of Proposition 5, each of the vectors u_1 through u_r are orthogonal and, hence, orthonormal.

Now, observe that $\sigma_i^2 = \|Av_i\|_2^2 = \langle Av_i, Av_i \rangle = \langle A^\intercal Av_i, v_i \rangle = \langle \lambda_i v_i, v_i \rangle = \lambda_i$; that is, the squares of each value σ_i are the eigenvalues λ_i. From this observation, we get that $A^\intercal u_i = (1/\sigma_i) A^\intercal Av_i = (\lambda_i / \sigma_i) v_i = \sigma_i v_i$.

Lastly, we specify the vectors u_{r+1} through u_n, where $r < n$. Again, by the third property of Proposition 5, the vectors $\{u_1, \ldots, u_r\}$ are eigenvectors of $AA^\intercal$ associated with some nonzero eigenvalues. Since $AA^\intercal \in \mathbb{R}^{n \times n}$ and $\operatorname{rank}(AA^\intercal) = r$, we know by the rank-nullity theorem that $\operatorname{nullity}(AA^\intercal)$ has dimension $(n - r)$. Since each of u_{r+1} through u_n are associated with a zero eigenvalue, each of these vectors must be orthogonal to the vectors u_1 through u_r by Proposition 6. Therefore, $\{u_1, \ldots, u_n\}$ forms an orthonormal basis of $\mathbb{R}^n$ consisting of eigenvectors of $AA^\intercal$, and since $\operatorname{nullity}(AA^\intercal) = \operatorname{nullity}(A^\intercal)$ from Proposition 4, we get that $A^\intercal u_i = 0$ for all $(r+1) \leq i \leq n$. □

Immediately from the proof of the SVD theorem, we know that

- each of the singular values σ_i of A are uniquely determined, as they correspond to the positive square roots of the nonzero eigenvalues of $A^\intercal A$;
- each of the vectors $\{u_1, \ldots, u_n\}$ are the left singular vectors of A; and
- each of the vectors $\{v_1, \ldots, v_m\}$ are the right singular vectors of A.

From this, we can begin to formulate a method to compute the singular value decomposition of a matrix A. All we need to do is compute the eigenvalues and eigenvectors of both $AA^\intercal$ and $A^\intercal A$! Following this method, we obtain the singular values from the eigenvalues themselves, the left singular vectors from the eigenvectors of $AA^\intercal$, and the right singular vectors from the eigenvectors of $A^\intercal A$.

Example 8. Consider the matrix

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 0 & 2 \end{bmatrix}.$$

We will compute the singular value decomposition of this matrix.

First, to obtain the eigenvalues of A, we can choose to either work with $A^\intercal A$ (which is a 3×3 matrix) or with $AA^\intercal$ (which is a 2×2 matrix). Let's work with the latter since it's smaller. Then we have

$$AA^\intercal = \begin{bmatrix} 5 & 2 \\ 2 & 8 \end{bmatrix}.$$

The characteristic polynomial of this matrix is $\lambda^2 - 13\lambda + 36 = (\lambda - 9)(\lambda - 4)$, so the eigenvalues of $AA^\intercal$ are $\lambda_1 = 9$ and $\lambda_2 = 4$. From this, we find that the singular values of A are $\sigma_1 = \sqrt{\lambda_1} = 3$ and $\sigma_2 = \sqrt{\lambda_2} = 2$.

Next, we get the left singular vectors from $AA^\intercal$. Solving the equation $(\lambda_1 I - AA^\intercal)u = 0$, we find that any multiple of the vector $\begin{bmatrix} 1 & 2 \end{bmatrix}^\intercal$ is an eigenvector of $AA^\intercal$ associated with the eigenvalue λ_1. Likewise, solving the equation $(\lambda_2 I - AA^\intercal)u = 0$, any multiple of the vector $\begin{bmatrix} 2 & -1 \end{bmatrix}^\intercal$ is an eigenvector of $AA^\intercal$ associated with λ_2. Since we want u_1 and u_2 to have a unit Euclidean norm, we will take

$$u_1 = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{and} \quad u_2 = \frac{1}{\sqrt{5}} \begin{bmatrix} 2 \\ -1 \end{bmatrix}.$$

Finally, we get the right singular vectors from $A^\intercal A$. Although we could compute this directly from the eigenvectors of $A^\intercal A$, as we did with the left singular vectors, we can alternatively use the formula $v_i = \sigma_i^{-1} A^\intercal u_i$ for $i \in \{1, 2\}$. Following this approach produces

$$v_1 = \frac{1}{3\sqrt{5}} \begin{bmatrix} 5 \\ 2 \\ 4 \end{bmatrix} \quad \text{and} \quad v_2 = \frac{1}{\sqrt{5}} \begin{bmatrix} 0 \\ 2 \\ -1 \end{bmatrix}.$$

Now, we must find the remaining right singular vector v_3. We know that $Av_3 = 0$, so solving the linear equation $Av = 0$ for v and normalizing, we get

$$v_3 = \frac{1}{3} \begin{bmatrix} -2 \\ 1 \\ 2 \end{bmatrix}.$$

Having computed all of these values, we can construct the singular value decomposition $A = U\Sigma V^\intercal$ where $U \in \mathbb{R}^{2\times 2}$, $\Sigma \in \mathbb{R}^{2\times 3}$, and $V \in \mathbb{R}^{3\times 3}$. Doing so gives us

$$U = \begin{bmatrix} u_1 & u_2 \end{bmatrix} = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix},$$

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}, \text{ and}$$

$$V = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \frac{1}{3\sqrt{5}} \begin{bmatrix} 5 & 0 & -2\sqrt{5} \\ 2 & 6 & \sqrt{5} \\ 4 & -3 & 2\sqrt{5} \end{bmatrix}.$$