QR Factorization: Givens Rotations

We will transform A into the product QR by applying on the left a sequence of orthogonal matrices called Givens matrices that transform A into an upper triangular matrix R. Each Givens matrix product zeros out a matrix element in the creation of R.

1 Givens Rotations

Here is the idea:

Apply $n-1$ Givens matrices, $J_{i1}(2 \leq i \leq n)$ on the left of A so that $a_{21}, a_{31}, \ldots, a_{n1}$ are zeroed out, and

$$(J_{n1}...J_{31}J_{21})A \rightarrow \begin{bmatrix} X & X & \ldots & X \\ 0 & X & \ldots & X \\ 0 & X & \ldots & X \\ \vdots & \vdots & \vdots & \vdots \\ 0 & X & \ldots & X \end{bmatrix}$$

Now use $n-2$ Givens matrices to zero out the elements at indices (3,2), (4,2),..., (n,2), and we have

$$(J_{n2}...J_{42}J_{32})(J_{n1}...J_{31}J_{21})A \rightarrow \begin{bmatrix} X & X & \ldots & X \\ 0 & X & \ldots & X \\ 0 & 0 & \ldots & X \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & X \end{bmatrix}$$

Let J_i be the product of Givens matrices acting on column i. Continue this process until we have

$$(J_{n-1}J_{n-2}...J_2J_1)A \rightarrow \begin{bmatrix} X & X & \ldots & X & X \\ 0 & X & \ldots & X & X \\ 0 & 0 & \ldots & X & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & X \end{bmatrix} = R$$

and

$$A = (J_{n-1}J_{n-2}...J_2J_1)^T R = QR$$

Q is a product of orthogonal matrices, and so it is orthogonal.

A Givens matrix is the matrix of the form

$$J(i,j,c,s) = \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & c & s & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -s & c & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \dots & 0 & \dots & 1 \end{bmatrix}$$

The value c is on the diagonal at indices (i,i) and $(j,j), i < j$. The value $-s$ is at index (j,i) and s is at index (i,j). The remaining diagonal entries are 1, and all off-diagonal elements at indices other than (j,i) and (i,j) are zero.

We want a Givens matrix to be orthogonal ($J(i,j,c,s)^T J(i,j,c,s) = I$), and clearly the columns in the definition are orthogonal, however, each column must have unit length. This requires that $c^2 + s^2 = 1$.

Example 1 Let $J(1,3,c,s) = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix}$. Then

$$J(1,3,c,s)^T J(1,3,c,s) = \begin{bmatrix} c & 0 & -s \\ 0 & 1 & 0 \\ s & 0 & c \end{bmatrix} \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix} = \begin{bmatrix} c^2+s^2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & c^2+s^2 \end{bmatrix}$$

So, if $c^2 + s^2 = 1, J(1,3,c,s)$ is orthogonal.

In the $(n \times n)$ case,

$$\begin{bmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & c & -s & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & s & c & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \dots & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & c & s & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -s & c & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \dots & 0 & \dots & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & c^2+s^2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & c^2+s^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \dots & 0 & \dots & 1 \end{bmatrix}$$

Require that $c^2+s^2=1$ and recall the identity $\sin^2\theta+\cos^2\theta=1$. For instance, the matrix

$$J(4,6,c,s) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\left(\frac{\pi}{6}\right) & 0 & \sin\left(\frac{\pi}{6}\right) \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\sin(\frac{\pi}{6}) & 0 & \cos(\frac{\pi}{6}) \end{bmatrix}$$

where $i=4, j=6, c=\cos(\frac{\pi}{6}), s=\sin(\frac{\pi}{6})$ is orthogonal since $\sin^2\theta+\cos^2\theta=1$. In general, we can choose any angle θ, let $c=\cos(\theta), s=\sin(\theta)$ and always obtain a Givens matrix. When we make such a choice, we will use the notation $J(i,j,\theta)$. Such a matrix is actually a rotation matrix that rotates a pair of coordinate axes through an angle θ in the (i,j) plane, so it is also known as a Givens rotation. For example, in $\mathbb{R}^2$ the Givens matrix is $J(1,2,\theta)=\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$; it rotates a vector clockwise through the angle θ.

1.1 Zeroing a particular entry in a vector

The Givens QR decomposition algorithm relies on being able to place zeros at specified locations in a matrix. Let $\mathbf{x}=\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$, and form the product $J(i,j,c,s)\mathbf{x}$:

$$J(i,j,c,s)\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & c & s & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -s & c & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \dots & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_j \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ cx_i+sx_j \\ \vdots \\ -sx_i+cx_j \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}$$

Note that the product changes only the components i and j of $\mathbf{x}$.

Example 2

$$J(2,4,c,s)\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & 0 & s \\ 0 & 0 & 1 & 0 \\ 0 & -s & 0 & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ cx_2 + sx_4 \\ x_3 \\ -sx_2 + cx_4 \end{bmatrix}$$

Assume we have a vector $\mathbf{x} \in \mathbb{R}^n$ and want to zero out $x_j, j > 1$ as illustrated below

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{bmatrix}; \quad J(i,j,c,s)\mathbf{x} = \begin{bmatrix} x_1 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

We must determine the values of i, j, c, s to use and then form the Givens rotation by creating the identity matrix, inserting c in locations (i,i) and (j,j), $-s$ in locaation (j,i) and s into position (i,j). Since multiplication by a Givens matrix only affects components i and j of the vector, we can reduce finding c and s to a two-dimensional problem.

Create a Givens matrix $\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$ that rotates vector $\begin{bmatrix} x \\ y \end{bmatrix}$ in $\mathbb{R}^2$ onto the x-axis. In this way we zero the y-coordinate. We have

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \\ -\frac{y}{\sqrt{x^2+y^2}} & \frac{x}{\sqrt{x^2+y^2}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2} \\ 0 \end{bmatrix}$$

The vector $\begin{bmatrix} x \\ y \end{bmatrix}$ after rotation will have an x-coordinate of $\sqrt{x^2+y^2}$ and y-coordinate of 0.

Summary: To zero out entry j of vector $\mathbf{x}$, choose index $i < j$ and compute

$$c = \frac{x_i}{\sqrt{x_i^2 + y_i^2}}, \quad s = \frac{y_i}{\sqrt{x_i^2 + y_i^2}}.$$

Let $J(i,j,c,s) = I$, followed by

$$J(i,i,c,s) = J(j,j,c,s) = c$$

$$J(j,i,c,s) = -s$$

$$J(i,j,c,s) = s$$

and compute $J(i,j,c,s)\mathbf{x}$.

Example 3

(a) Zero out component 2 of the (3×1) vector $\mathbf{x} = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix}$.

Choose $i = 1, j = 2$ and compute $c = \frac{1}{\sqrt{5}}, s = \frac{2}{\sqrt{5}}$. The Givens rotation matrix is

$$J(1,2,c,s) = \begin{bmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 \\ -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now form the product $J(1,2,c,s)\mathbf{x}$ to obtain

$$\begin{bmatrix} -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 \\ -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} \sqrt{5} \\ 0 \\ 3 \end{bmatrix}$$

(b) Zero out component 3 of the (4×1) vector $\mathbf{x} = \begin{bmatrix} 3 & -7 & -1 & 5 \end{bmatrix}^T$. Choose $i = 1, j = 3$, and then $c = \frac{3}{\sqrt{10}}, s = -\frac{1}{\sqrt{10}}$ and form

$$J(1,3,c,s) = \begin{bmatrix} \frac{3}{\sqrt{10}} & 0 & -\frac{1}{\sqrt{10}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{10}} & 0 & \frac{3}{\sqrt{10}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ -7 \\ -1 \\ 5 \end{bmatrix} = \begin{bmatrix} \sqrt{10} \\ -7 \\ 0 \\ 5 \end{bmatrix}$$

It is necessary to choose $i < j$, and any index $i < j$ can be used. Choose $i = 2, j = 3$, form $J(2,3,c,s)$, and verify that $J(2,3,c,s)\mathbf{x}$ zeros out x_3.

2 Creating a sequence of zeros in a vector using Givens rotation

Given a vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$, we can use a product of Givens matrices to zero out the elements of $\mathbf{x}$ below entry x_i. To zero out x_{i+1}, compute $J(i, i+1, c_{i+1}, s_{i+1})\mathbf{x} = \overline{\mathbf{x}_{i+1}}$. To zero out x_{i+2}, compute $J(i, i+2, c_{i+2}, s_{i+2})\overline{\mathbf{x_{i+1}}} = \overline{\mathbf{x}_{i+2}}$ and continue the process until computing $J(i, n, c_n, s_n)\overline{\mathbf{x_{n-1}}} = \overline{\mathbf{x}_n}$. In summary, the product

$$J(i,n,c_n,s_n) \dots J(i,i+2,c_{i+2},s_{i+2})J(i,i+1,c_{i+1},s_{i+1})\mathbf{x}$$

transforms $\mathbf{x}$ into a vector of the form $\begin{bmatrix} x_1 & x_2 & \dots & x_{i-1} & * & 0 & \dots & 0 \end{bmatrix}^T$.

Example 4 Let $\mathbf{x} = \begin{bmatrix} 5 \\ -1 \\ 3 \end{bmatrix}$ and zero out the second and third components of $\mathbf{x}$ using Givens rotations.

$$\overline{\mathbf{x}_2} = J(1,2,c_2,s_2)\mathbf{x} = \begin{bmatrix} \frac{5}{\sqrt{26}} & -\frac{1}{\sqrt{26}} & 0 \\ \frac{1}{\sqrt{26}} & \frac{5}{\sqrt{26}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ -1 \\ 3 \end{bmatrix} = \begin{bmatrix} \sqrt{26} \\ 0 \\ 3 \end{bmatrix}$$

$$\overline{\mathbf{x}_3} = J(1,3,c_3,s_3)\overline{\mathbf{x}_2} = \begin{bmatrix} \frac{\sqrt{26}}{\sqrt{35}} & 0 & \frac{3}{\sqrt{35}} \\ 0 & 1 & 0 \\ -\frac{3}{\sqrt{35}} & 0 & \frac{\sqrt{26}}{\sqrt{35}} \end{bmatrix} \begin{bmatrix} \sqrt{26} \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} \sqrt{35} \\ 0 \\ 3 \end{bmatrix}$$

3 Product of a Givens matrix with a general matrix

If A is an $(m \times n)$ matrix, a huge advantage when dealing with Givens matrices is the fact that you can compute a product $J(i,j,c,s)A$ without ever constructing $J(i,j,c,s)$.

Example 5: If $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$, then

$$J(1,3,c,s)A = \begin{bmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} = \begin{bmatrix} ca_{11} + sa_{31} & ca_{12} + sa_{32} & ca_{13} + sa_{33} \\ a_{21} & a_{22} & a_{23} \\ ca_{31} - sa_{11} & ca_{32} - sa_{12} & ca_{33} - sa_{13} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

The product only affects rows 1 and 3.

In the general case where $J(i,j,c,s)$ is $(m \times m)$ and A is $(m \times n)$ the product $J(i,j,c,s)A$ only affects rows i and j. Rows i and j , $i < j$ have the following form:

$$\text{Row i:} \quad ca_{i1} + sa_{j1}, ca_{i2} + sa_{j2}, \ldots, ca_{in} + sa_{jn}$$

$$\text{Row j:} \quad ca_{j1} - sa_{i1}, ca_{j2} - sa_{i2}, \ldots, ca_{jn} - sa_{in}$$

This makes the product $J(i,j,c,s)A$ very efficient to compute just changing rows i and j.

4 Zeroing-out column entries in a matrix using Givens rotations

We know how to zero out all the entries of a vector $\mathbf{x}$ below an entry x_i, and we know how to form the product of a Givens matrix with another matrix. Now we will learn how to zero out all the entries in column i below a matrix entry $A(i,i)$. This is what we must do to create an upper triangular matrix.

First, consider the problem of computing the product $J(i,j,c,s)A, j > i$, so that the result will have a zero at index (j,i) and only rows i and j will change. Think of column i, as a vector and find c and s using a_{ii} and a_{ji} so that the product will zero out a_{ji}. Form a Givens matrix with c at indices (i,i) and (j,j), $-s$ at index (j,i) and s at index (i,j) and compute $J(i,j,c,s)A$ as it discussed above. Entries in both rows i and j will change. This makes no difference since we are only interested in zeroing out a_{ji}.

To zero out every element in column i below a_{ii}, compute the sequence

$$\overline{A_{i+1}} = J(i, i{+}1, c_{i+1}, s_{i+1})A, \quad \overline{A_{i+2}} = J(i, i{+}2, c_{i+2}, s_{i+2})\overline{A_{i+1}}, \ldots, \overline{A_m} = J(i, m, c_m, d_m)\overline{A_{m-1}}$$

Example 5 Let $A = \begin{bmatrix} 1 & 3 & -6 & -1 \\ 4 & 8 & 7 & 3 \\ 2 & 3 & 4 & 5 \\ -9 & 6 & 3 & 2 \end{bmatrix}$.

Zero out all entries in column 1 below a_{11}. Compute $c = \frac{1}{\sqrt{1^2+4^2}} \approx 0.2425$, $s = \frac{4}{\sqrt{1^2+4^2}} \approx 0.9701$ and form $J(1, 2, c_2, s_2) = \begin{bmatrix} 0.2425 & 0.9701 & 0 & 0 \\ -0.9701 & 0.2425 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

$$\overline{A_2} = J(1, 2, c_2, s_2)A = \begin{bmatrix} 4.1231 & 8.4887 & 5.3358 & 2.6679 \\ 0 & -0.9701 & 7.5186 & 1.6977 \\ 2 & 3 & 4 & 5 \\ -9 & 6 & 3 & 2 \end{bmatrix}$$

Form $J(1, 3, c_3, s_3) = \begin{bmatrix} 0.8997 & 0 & 0.4364 & 0 \\ 0 & 1 & 0 & 0 \\ -0.4364 & 0 & 0.8997 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and compute

$$\overline{A_3} = J(1, 3, c_3, s_3)\overline{A_2} = \begin{bmatrix} 4.5826 & 8.9469 & 6.5465 & 4.5826 \\ 0 & -0.9701 & 7.5186 & 1.6977 \\ 0 & -1.0056 & 1.2702 & 3.3343 \\ -9 & 6 & 3 & 2 \end{bmatrix}$$

Form $J(1, 4, c_4, s_4) = \begin{bmatrix} -0.4537 & 0 & 0 & 0.8911 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.8911 & 0 & 0 & -0.4537 \end{bmatrix}$ and compute

$$\overline{A_4} = J(1, 4, c_4, s_4)\overline{A_3} = \begin{bmatrix} -10.0995 & 1.2872 & -0.2970 & -0.2970 \\ 0 & -0.9701 & 7.5186 & 1.6977 \\ 0 & -1.0056 & 1.2702 & 3.3343 \\ 0 & -10.6954 & -7.1951 & -4.9912 \end{bmatrix}$$

Form

$$P = J(1, 4, c_4, s_4)J(1, 3, c_3, s_3)J(1, 2, c_2, s_2) = \begin{bmatrix} -0.0990 & -0.3961 & -0.1980 & 0.8911 \\ -0.9701 & 0.2425 & 0 & 0 \\ -0.1059 & -0.4234 & 0.8997 & 0 \\ -0.1945 & -0.7778 & -0.3889 & -0.4537 \end{bmatrix},$$

and $PA = \overline{A_i}$. The matrix P is the product of orthogonal matrices, and so P is orthogonal.

Note that we can continue with $\overline{A_4}$ and use Givens rotations to zero out the elements in column 2 below $\overline{A_4}(2, 2)$. By zeroing out the element at index (4,3), we will have an upper triangular matrix.

It is critical that c and s be computed as accurately as possible when implicitly multiplying by $J(i, j, c, s)$.

5 The Givens algorithm for the QR decomposition

The computation of c and s can have problems with overflow or underflow. The following algorithm provides improvement in overall accuracy by employing the normalization procedure. The algorithm takes care of the case where $s_j = 0$ by assigning $c = 1$ and $s = 0$ so that the Givens rotation is the identity matrix. The signs of c and s may be different from those obtained from $c = \frac{x_i}{\sqrt{x_i^2+x_j^2}}, s = \frac{x_j}{\sqrt{x_i^2+x_j^2}}$, but that does not change the rotation's effect. All the necessary tools are now in place to construct Q and R using Givens rotations.

An $(m \times n)$ matrix $A = [a_{ij}]$ is upper triangular if $a_{ij} = 0$ for $i > j$. Another way of putting it is that all entries below a_{ii} are 0.

For example, the matrices $A = \begin{bmatrix} 1 & -1 & 7 & 12 & 1 & 3 \\ 0 & 2 & 8 & 2 & 4 & 1 \\ 0 & 0 & 3 & -9 & 10 & 6 \\ 0 & 0 & 0 & 4 & 2 & 1 \\ 0 & 0 & 0 & 0 & 5 & 18 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 2 & 6 & -1 \\ 0 & 4 & -2 & 8 \\ 0 & 0 & -1 & -9 \\ 0 & 0 & 0 & -4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ are upper triangular.

The decomposition algorithm uses Givens rotations to zero out elements below the diagonal element until arriving at an upper triangular matrix. The number of steps we should execute depends on the dimension of A. If A is an $(n \times n)$ matrix , we need to execute the process $k = n - 1$ times.

Let us consider $m > n$:

Look at a (5×3) matrix $A_1 = \begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \\ X & X & X \\ X & X & X \end{bmatrix}$. Here are the transformations that occur to A_1.

$$A_1 = \begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \\ X & X & X \\ X & X & X \end{bmatrix} \Rightarrow \begin{bmatrix} X & X & X \\ 0 & X & X \\ 0 & X & X \\ 0 & X & X \\ 0 & X & X \end{bmatrix} \Rightarrow \begin{bmatrix} X & X & X \\ 0 & X & X \\ 0 & 0 & X \\ 0 & 0 & X \\ 0 & 0 & X \end{bmatrix} \Rightarrow \begin{bmatrix} X & X & X \\ 0 & X & X \\ 0 & 0 & X \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

We executed three steps until we could not continue to zero out elements below the diagonal, so $k = 3 = n$.

Let $m < n$: Look at a (4×6) matrix

$$A_2 = \begin{bmatrix} X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \end{bmatrix} \Rightarrow \begin{bmatrix} X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & X & X & X & X \end{bmatrix} \Rightarrow \begin{bmatrix} X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & 0 & X & X & X \end{bmatrix}$$

We executed three steps until we could not continue to zero out elements below the diagonal, so $k = 3 = m - 1$.

From these examples, we see that the sequence of steps in the Givens QR decomposition algorithm is $k = min(m-1, n)$. To compute Q, start with $Q = I$. As the algorithm progresses, build Q and R.

Theorem 5.1 *If A is an $m \times n$ matrix, it can be expressed in the form $A = QR$, where Q is an $m \times n$ orthogonal matrix and R is an $m \times n$ upper triangular matrix.*

Example 6

Listing 1: QR Factorization: Givens Rotations example

```
import numpy as np

def givens_rotation(a, b):
    """Computes the cosine and sine for a Givens rotation."""
    if b == 0:
        c, s = 1, 0
    else:
        r = np.hypot(a, b)  # Computes sqrt(a^2 + b^2) in a stable way
        c = a / r
        s = -b / r
    return c, s

def qr_givens(A):
    """Computes the QR decomposition of A using Givens rotations."""
    m, n = A.shape
    Q = np.eye(m)  # Initialize as identity matrix
    R = A.copy().astype(float)

    for j in range(n):  # Iterate over columns
        for i in range(j+1, m):  # Iterate over rows below diagonal
            c, s = givens_rotation(R[j, j], R[i, j])
```

```
            # Apply rotation to R
            G = np.eye(m)
            G[[j, i], [j, i]] = c
            G[i, j], G[j, i] = s, -s

            R = G @ R  # Rotate R
            Q = Q @ G.T  # Accumulate Q

    return Q, R

# Example matrix
A = np.array([[4, 3],
              [6, 3],
              [8, 6]], dtype=float)

# Compute QR using Givens rotations
Q_givens, R_givens = qr_givens(A)

# Compute QR using NumPy's built-in function
Q_numpy, R_numpy = np.linalg.qr(A)
# Print results
print("=== Givens Rotation QR ===")
print("Q matrix:\n", np.round(Q_givens, 6))
print("R matrix:\n", np.round(R_givens, 6))

print("\n=== NumPy QR ===")
print("Q matrix:\n", np.round(Q_numpy, 6))
print("R matrix:\n", np.round(R_numpy, 6))

# Verify orthogonality
print("Givens Q^T Q:\n", np.round(Q_givens.T @ Q_givens, 6))
print("NumPy Q^T Q:\n", np.round(Q_numpy.T @ Q_numpy, 6))

# Verify QR reconstruction
print("Givens Q @ R:\n", np.round(Q_givens @ R_givens, 6))
print("NumPy Q @ R:\n", np.round(Q_numpy @ R_numpy, 6))
```

```
=== Givens Rotation QR ===
Q matrix:
 [[ 0.371391  0.249136  0.894427]
 [ 0.557086 -0.830455 -0.      ]
 [ 0.742781  0.498273 -0.447214]]
R matrix:
 [[10.77033   7.242118]
 [ 0.        1.245682]
 [-0.       -0.      ]]
```

```
=== NumPy QR ===
Q matrix:
 [[-0.371391  0.249136]
 [-0.557086 -0.830455]
 [-0.742781  0.498273]]
R matrix:
 [[-10.77033   -7.242118]
 [  0.         1.245682]]


Givens Q^T Q:
 [[ 1. -0.  0.]
 [-0.  1.  0.]
 [ 0.  0.  1.]]
NumPy Q^T Q:
 [[1. 0.]
 [0. 1.]]


Givens Q @ R:
 [[4. 3.]
 [6. 3.]
 [8. 6.]]
NumPy Q @ R:
 [[4. 3.]
 [6. 3.]
 [8. 6.]]
```