For a symmetric matrix $A$, an eigenvector $\mathbf{v}$ is a vector that satisfies

$$A \mathbf{v} = \lambda \mathbf{v}$$

the corresponding eigenvalue $\lambda$.

For an $N \times N$ matrix there are $N$ eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots \mathbf{v}_N$ with corresponding eigenvalues $\lambda_1, \lambda_2, \lambda_3, \ldots \lambda_N$.

The eigenvectors are orthogonal:

$$\mathbf{v}_i \cdot \mathbf{v}_j = 0 \quad \text{if} \quad i \neq j$$

and they are normalized

$$\mathbf{v}_i \cdot \mathbf{v}_i = 1 \quad \text{(this is just convention)}$$

If we consider the matrix $V$, where each column is an eigenvector $\mathbf{v}_i$, we can write all $N$ equations

$$A \mathbf{v}_i = \lambda_i \mathbf{v}_i$$

into a single equation

$$A \mathbf{V} = \mathbf{V} \mathbf{D}$$
where $D$ is a diagonal matrix with the eigenvalues $\lambda_i$.

Because the eigenvectors are orthogonal, the matrix $V$ is **orthogonal**

$$V^T V = V V^T = I$$

**QR ALGORITHM** is a popular technique to diagonalize real symmetric and Hermitian matrices.

The algorithm uses the **QR decomposition** of the matrix. We will detail about the decomposition next. For now we just need to know that this is a way to break the matrix $A$ into the product

$$Q \cdot R$$

where $Q$ is an **orthogonal** matrix and $R$ is an **upper-triangular** matrix.
Let us start by writing

\[ A = Q_1 R_1 \]

\[ \Rightarrow \text{ multiply by } Q_1^T \]

\[
Q_1^T A = Q_1^T Q_1 R_1 = R_1
\]

\[ \Rightarrow \text{ Define a new matrix } \]

\[ A_1 = R_1 Q_1 \]

from above \[ R_1 = Q_1^T A \]

\[ A_1 = Q_1^T A Q_1 \]

\[ \Rightarrow \text{ Repeat the process } \]

\[ A_2 = Q_2 R_2 \]

\[ Q_2^T A_2 = R_2 \]

\[ \text{ new matrix } \]

\[ A_2 = R_2 Q_2 \]

\[ A_2 = Q_2^T A_2 Q_2 \]

so

\[ A_2 = Q_2^T Q_1^T A Q_1 Q_2 \]
Repeating the process many times

\[ A_1 = Q_1^T A D_1 \]
\[ A_2 = Q_2^T Q_1^T A D_1 D_2 \]
\[ A_3 = Q_3^T Q_2^T Q_1^T A D_1 D_2 D_3 \]
\[ \vdots \]
\[ A_k = (Q_k^T \ldots Q_1^T) A (Q_1 \ldots Q_k) \]

It can be proven that if we continue this process long enough, the matrix \( A_k \) will eventually become diagonal. The off-diagonal elements become smaller and smaller the more iterations we do.

In practice, \( A_k \) is approximately a diagonal matrix \( D \)

Let us define

\[ V = Q_1, Q_2, Q_3, \ldots, Q_k \]

From above

\[ V^T A V = D \]
Multiplying by \( V \)

\[
AV = VD
\]

which is exactly the original equation we had to solve.

Therefore

1. The diagonal elements of
   
   
   \[
   A_k = \begin{pmatrix} a_{k1} & \cdots & a_{k1} \\ \vdots & \ddots & \vdots \\ a_{kk} & \cdots & a_{kk} \end{pmatrix}
   \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_k \end{pmatrix}
   \]

   are the eigenvalues

2. Each column of

   \[
   V = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_k \end{pmatrix}
   \]

   is an eigenvector

**RECIPE**

1. Crack an \( N \times N \) identity matrix \( V \).

   Choose the target accuracy \( \epsilon \) for the off-diagonal elements of the eigenvalue matrix.
2) Calculate the QR decomposition
   \( A = QR \) (see below how to do it)

3) Update \( A \) to the new value
   \[ A = RQ \]

4) Multiply \( V \) on the right by \( Q \)
   \[ V = VQ \]

5) Check the off-diagonal elements of the new \( A \). If they are less than \( \varepsilon \), we are done. Otherwise go back to step 2.

**QR decomposition**

Let us think of \( A \) as a set of \( N \) column vectors \( a_0, a_1, \ldots, a_{N-1} \)

\[ \text{using Python numbering} \]
Let us define two sets of vectors:

\[ m_0, m_1, \ldots, m_{n-1} \quad \text{and} \quad g_0, g_1, \ldots, g_{n-1} \]

\[
\begin{align*}
\mathbf{m}_0 &= \mathbf{a}_0 \\
\mathbf{m}_1 &= \mathbf{a}_1 - (\mathbf{g}_0 \cdot \mathbf{a}_1) \mathbf{g}_0 \\
\mathbf{m}_2 &= \mathbf{a}_2 - (\mathbf{g}_0 \cdot \mathbf{a}_2) \mathbf{g}_0 - (\mathbf{g}_1 \cdot \mathbf{a}_2) \mathbf{g}_1 \\
&\vdots
\end{align*}
\]

General formulas:

\[
\mathbf{m}_i = \mathbf{a}_i - \sum_{j=0}^{i-1} (\mathbf{q}_j \cdot \mathbf{a}_i) \mathbf{q}_j
\]

\[
\mathbf{q}_i = \frac{\mathbf{m}_i}{|\mathbf{m}_i|}
\]

It can be shown that the vectors \( q_i \) are \underline{ORTHONORMAL}:

\[
\sum_{i=0}^{n-1} q_i q_j = \begin{cases} 
1 & \text{if } i = j \\
0 & \text{if } i \neq j
\end{cases}
\]
\[ q_i \cdot q_j = \begin{cases} 
1 & \text{if } i \neq j \\
0 & \text{if } i = j 
\end{cases} \]

\[ \Rightarrow \text{Rearranging the definitions} \]

\[
\begin{align*}
    a_0 &= 1 \omega_0 \, q_0 \\
    a_1 &= \omega_1 \, q_1 + (q_0 \cdot a_1) \, q_0 \\
    a_2 &= \omega_2 \, q_2 + (q_0 \cdot a_2) q_0 + (q_1, a_2) q_1 \\
    &\ldots
\end{align*}
\]

\[ \Rightarrow \text{This can be written in a matrix form} \]

\[ A = \begin{pmatrix} 
1 & 1 & 1 \\
a_0 & a_1 & a_2 \\
1 & 1 & 1
\end{pmatrix} = \begin{pmatrix} 
q_0 & q_1 & q_2 \\
1 & 1 & 1
\end{pmatrix} \begin{pmatrix} 
1 & 0 & 0 \\
q_0, a_1 & q_0, a_2 \\
0 & 1 & 0 \\
0 & 0 & 1 \omega_1
\end{pmatrix} \]

\[ Q \text{ orthogonal matrix} \]
\[ R \text{ upper triangular matrix} \]

This is the Q R decomposition.

(1) Starting with a certain matrix \( A \), we use the \( q \)'s and \( g \)'s.
about to write \( A_{\text{old}} = QR \)

2. Update \( A_{\text{new}} = RQ \)

i) If the process is repeated, use \( A_{\text{new}} \) above and its corresponding new \( u \)'s and \( g \)'s to get new \( Q \) and \( R \) and restart the steps 1 and 2