Suppose \( A \in \mathbb{R}^{n \times n} \) \( A = A^t \)

Simple power iteration:

\[
\begin{align*}
\mathbf{b} & = \text{random vector} \\
\mathbf{w}_0 & = \frac{\mathbf{b}}{\| \mathbf{b} \|} \\
\text{FOR } \ k = 1, 2, 3, \ldots \\
\mathbf{w}_k & = \mathbf{A} \mathbf{w}_{k-1} \\
\mathbf{b} & = \mathbf{A} \mathbf{w}_{k-1} \\
\mathbf{w}_k & = \frac{1}{\| \mathbf{b} \|} \mathbf{b} \\
\end{align*}
\]

END

Then \( \mathbf{w}_k \mathbf{A} \mathbf{w}_k \rightarrow \lambda_1 \) at speed \( O \left( \frac{1}{\kappa_1^2} \right) \)

The Rayleigh quotient \( \mathbf{w}_k^T \mathbf{A} \mathbf{w}_k \rightarrow \lambda_1 \) at speed \( O \left( \frac{1}{\kappa_1^2} \right) \)

Algorithm 27.3. Rayleigh Quotient Iteration

\[
\begin{align*}
\mathbf{v}^{(0)} & = \text{some vector with } \| \mathbf{v}^{(0)} \| = 1 \\
\lambda^{(0)} & = (\mathbf{v}^{(0)})^T \mathbf{A} \mathbf{v}^{(0)} = \text{corresponding Rayleigh quotient} \\
\text{for } k = 1, 2, \ldots \\
\text{Solve } (\mathbf{A} - \lambda^{(k-1)} I) \mathbf{w} = \mathbf{v}^{(k-1)} \text{ for } \mathbf{w} \\
\mathbf{v}^{(k)} & = \mathbf{w} / \| \mathbf{w} \| \\
\lambda^{(k)} & = (\mathbf{v}^{(k)})^T \mathbf{A} \mathbf{v}^{(k)} \\
\end{align*}
\]

Convergence is extremely fast

Tridiagonalize \( A \) first to accelerate iteration

\( \Rightarrow O(m) \) cost per step
**Block Power Iteration**

Suppose: \( A \in \mathbb{R}^{m \times m} \) \( A = A^t \)

\[ B_0 = \text{random matrix of size } m \times b. \]

For \( t = 1, 2, 3, \ldots \)

\[ B_t = A B_{t-1} \]

END

\([Q, \omega] = \text{gr}(B_t)\]

Then \( \text{col}(Q) \) approximates \( \text{span} \{ y_j \}_{j=1}^b \)

Eigs of \( Q^t A Q \) approximate \( \lambda_1, \lambda_2, \ldots, \lambda_b \).

Numerically unstable!

Stabilized version:

\[ B_0 = \text{random matrix of size } m \times b \]

\([Q_0, \omega] = \text{gr}(B_0, 0)\]

For \( t = 1, 2, 3, \ldots \)

\[ B = A Q_{t-1} \]

\([Q_t, \omega] = \text{gr}(B, 0)\]

END.

If exact arithmetic is used, then

\[ \text{span}(Q_t) = \text{sp} \text{col}(Q_t) = \text{col}(Q) = \text{col}(A^t B_0). \]
ALGORITHMS FOR FINDING ALL EVALS

Suppose $A \in \mathbb{C}^{m \times m}$ and we seek all evals/evcs of $A$.

General template:

Step 1. Find Hessenberg form $A = QH(Q^*)^T$ with $Q$ unitary & $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} & & & \\ h_{21} & h_{22} & h_{23} & & & \\ 0 & h_{32} & h_{33} & & & \\ 0 & 0 & h_{43} & & & \\ & & & \ddots & & \\ & & & & & \end{bmatrix}$

Can be done using Householder reflectors, deterministic, very stable, $O(m^3)$ cost.

Step 2. Use iterative method to drive $H$ to upper triangular form, so that $H = \tilde{Q}T\tilde{Q}^*$ is the Schur factorization.

The result is the Schur fact. $A = (\tilde{Q}\tilde{Q}^*)T(\tilde{Q}\tilde{Q}^*)^*$

Special case: If $A = A^*$, then $H$ is tridiagonal, and $T$ is diagonal. Fewer flops, faster convergence.
Iterative Methods for Diagonalizing a Matrix

Suppose \( A \in \mathbb{R}^{m \times m} \) & \( A = A^k \) for simplicity.

Basic QR iteration:

\[
A_0 = A \\
\text{For } t = 1, 2, 3, \ldots \\
[Q_k, R_k] = qr(A_{t-1}) \\
A_t = R_k Q_k
\]

End

Observe: \( A_1 = R_1 Q_1 = Q_1^* A_0 Q_1 \) so \( A_1 \) & \( A_0 \) are similar.

\( A_0 = Q_1 R_1 \Rightarrow R_1 = Q_1^* A_0 \)

Convergence can be greatly accelerated using "shifts".

---

**Algorithm 28.2. "Practical" QR Algorithm**

\[
(Q^{(0)})^T A^{(0)} Q^{(0)} = A
\]

for \( k = 1, 2, \ldots \)

Pick a shift \( \mu^{(k)} \)

\( Q^{(k)} R^{(k)} = A^{(k-1)} - \mu^{(k)} I \)

\( A^{(k)} = R^{(k)} Q^{(k)} + \mu^{(k)} I \)

QR factorization of \( A^{(k-1)} - \mu^{(k)} I \)

Recombine factors in reverse order

If any off-diagonal element \( a_{j,j+1}^{(k)} \) is sufficiently close to zero,

set \( A_{j,j+1} = A_{j+1,j} = 0 \) to obtain

\[
\begin{bmatrix}
A_1 & 0 \\
0 & A_2
\end{bmatrix} = A^{(k)}
\]

and now apply the QR algorithm to \( A_1 \) and \( A_2 \).

---

Combination of block power iteration and shifted inverse iteration.