Francis's QR $\equiv$ QR

Let A be properly upper Hessenberg.

Shifted QR	**Francis's Shifted QR**
Set $A_0 = A$ *for* $j = 0, 1, \ldots$ (i) Compute $p_j(A_j)$. (ii) Find reflectors $Q_j^{(1)}, \ldots Q_j^{(n-1)}$ such that $Q_j^{(n-1)} \cdots Q_j^{(1)} p_j(A_j) = R_j$ is upper triangular (iii) Compute $A_{j+1} := Q_j^{(n-1)} \cdots Q_j^{(1)} A_j Q_j^{(1)} \cdots Q_j^{(n-1)}$.	Set $A_0 = A$ *for* $j = 0, 1, \ldots$ (i) Compute $p_j(A_j)e_1$. (ii) Find reflector $Q_j^{(1)}$ such that $Q_j^{(1)} p_j(A_j)e_1 = \alpha_j e_1$ (iii) Compute $Q_j^{(1)} A_j Q_j^{(1)}$. (iv) Find reflectors $\hat{Q}_j^{(2)}, \ldots, \hat{Q}_j^{(p)}$ such that $A_{j+1} := \hat{Q}_j^{(p)} \cdots \hat{Q}_j^{(2)} Q_j^{(1)} A_j Q_j^{(1)} \hat{Q}_j^{(2)} \cdots \hat{Q}_j^{(p)}$ is upper Hessenberg.

Shifted QR finds $Q_j := Q_j^{(1)} \cdots Q_j^{(n-1)}$ such that $p_j(A_j) = Q_j R_j$ is a QR decomposition of $p_j(A_j)$ and sets $A_{j+1} = Q_j^* A_j Q_j$.

But Francis's Shifted QR **also** finds a QR decomposition $p_j(A_j) = \tilde{Q}_j \tilde{R}_j$ and sets $A_{j+1} = \tilde{Q}_j^* A_j \tilde{Q}_j$ where

$$\tilde{Q}_j := Q_j^{(1)} \hat{Q}_j^{(2)} \cdots \hat{Q}_j^{(p)} \text{ and } \tilde{R}_j := \alpha_j K(A_{j+1}, e_1)[K(A_j, e_1)]^{-1}.$$

Computing eigenvectors via QR algorithm

Computing eigenvectors via QR algorithm

In general, the Schur form to which the iterates A_j converge is

Computing eigenvectors via QR algorithm

In general, the Schur form to which the iterates A_j converge is

- upper triangular if either A is complex or A is real with only real eigenvalues;

Computing eigenvectors via QR algorithm

In general, the Schur form to which the iterates A_j converge is

- upper triangular if either A is complex or A is real with only real eigenvalues;
- quasi-upper triangular if A is real with complex eigenvalues;

Computing eigenvectors via QR algorithm

In general, the Schur form to which the iterates A_j converge is

- upper triangular if either A is complex or A is real with only real eigenvalues;
- quasi-upper triangular if A is real with complex eigenvalues;
- diagonal if A is normal;

Computing eigenvectors via QR algorithm

In general, the Schur form to which the iterates A_j converge is

- upper triangular if either A is complex or A is real with only real eigenvalues;
- quasi-upper triangular if A is real with complex eigenvalues;
- diagonal if A is normal;
- real diagonal if A is Hermitian or real symmetric.

Computing eigenvectors via QR algorithm

In general, the Schur form to which the iterates A_j converge is

- upper triangular if either A is complex or A is real with only real eigenvalues;
- quasi-upper triangular if A is real with complex eigenvalues;
- diagonal if A is normal;
- real diagonal if A is Hermitian or real symmetric.

The iterates A_j are symmetric tridiagonal if A is Hermitian or real symmetric and upper Hessenberg otherwise.

Computing eigenvectors via QR algorithm

Let A be a $n \times n$ real symmetric or Hermitian matrix.

Computing eigenvectors via QR algorithm

Let A be a $n \times n$ real symmetric or Hermitian matrix.

- Each step of implicit QR costs $O(n)$ flops if only eigenvalues are computed;

Computing eigenvectors via QR algorithm

Let A be a $n \times n$ real symmetric or Hermitian matrix.

- Each step of implicit QR costs $O(n)$ flops if only eigenvalues are computed;
- The eigenvectors are the columns of Q where $Q^* AQ$ gives the limiting (diagonal) Schur form.

Computing eigenvectors via QR algorithm

Let A be a $n \times n$ real symmetric or Hermitian matrix.

- Each step of implicit QR costs $O(n)$ flops if only eigenvalues are computed;
- The eigenvectors are the columns of Q where $Q^* AQ$ gives the limiting (diagonal) Schur form.
  $\hookrightarrow$ Forming Q raises the flop count to $O(n^2)$ per iteration!

Computing eigenvectors via QR algorithm

Let A be a $n \times n$ real symmetric or Hermitian matrix.

- Each step of implicit QR costs $O(n)$ flops if only eigenvalues are computed;
- The eigenvectors are the columns of Q where Q^*AQ gives the limiting (diagonal) Schur form.
  $\hookrightarrow$ Forming Q raises the flop count to $O(n^2)$ per iteration!

For other matrices,

Computing eigenvectors via QR algorithm

Let A be a $n \times n$ real symmetric or Hermitian matrix.

- Each step of implicit QR costs $O(n)$ flops if only eigenvalues are computed;
- The eigenvectors are the columns of Q where Q^*AQ gives the limiting (diagonal) Schur form.
  $\hookrightarrow$ Forming Q raises the flop count to $O(n^2)$ per iteration!

For other matrices,

- Each step of implicit QR costs $O(n^2)$ when finding eigenvalues and also assimilating Q;

Computing eigenvectors via QR algorithm

Let A be a $n \times n$ real symmetric or Hermitian matrix.

- Each step of implicit QR costs $O(n)$ flops if only eigenvalues are computed;
- The eigenvectors are the columns of Q where $Q^* AQ$ gives the limiting (diagonal) Schur form.
  $\hookrightarrow$ Forming Q raises the flop count to $O(n^2)$ per iteration!

For other matrices,

- Each step of implicit QR costs $O(n^2)$ when finding eigenvalues and also assimilating Q;
- the eigenvectors are Qv where $T := Q^* AQ$ gives the limiting (upper triangular or quasi-upper triangular) Schur form, and v is an eigenvector of T.

Computing eigenvectors via QR algorithm

Let A be a $n \times n$ real symmetric or Hermitian matrix.

- Each step of implicit QR costs $O(n)$ flops if only eigenvalues are computed;
- The eigenvectors are the columns of Q where Q^*AQ gives the limiting (diagonal) Schur form.
  $\hookrightarrow$ Forming Q raises the flop count to $O(n^2)$ per iteration!

For other matrices,

- Each step of implicit QR costs $O(n^2)$ when finding eigenvalues and also assimilating Q;
- the eigenvectors are Qv where $T := Q^*AQ$ gives the limiting (upper triangular or quasi-upper triangular) Schur form, and v is an eigenvector of T.
  $\hookrightarrow$ Finding v requires forming T!

Computing eigenvectors via QR algorithm

Strategy to find eigenvectors:

Computing eigenvectors via QR algorithm

Strategy to find eigenvectors:

- Run the QR algorithm twice! First time find only eigenvalues and second time form the Q using the computed eigenvalues as shifts.

Computing eigenvectors via QR algorithm

Strategy to find eigenvectors:

- Run the QR algorithm twice! First time find only eigenvalues and second time form the Q using the computed eigenvalues as shifts.
- If A is Hermitian, the columns of Q are the eigenvectors.

Computing eigenvectors via QR algorithm

Strategy to find eigenvectors:

- Run the QR algorithm twice! First time find only eigenvalues and second time form the Q using the computed eigenvalues as shifts.
- If A is Hermitian, the columns of Q are the eigenvectors.
- If A is not Hermitian, solve upper triangular systems $Tv = \lambda v$ for v and compute Qv.

Computing eigenvectors via QR algorithm

Strategy to find eigenvectors:

- Run the QR algorithm twice! First time find only eigenvalues and second time form the Q using the computed eigenvalues as shifts.
- If A is Hermitian, the columns of Q are the eigenvectors.
- If A is not Hermitian, solve upper triangular systems $Tv = \lambda v$ for v and compute Qv.
- If A is real, arrange all computations to stay in real arithmetic till the very end.

Implicit QR: One of the top 10 algorithms of the 20th century