Recap: Last Time

We noted that eigenvalue revealing computations are generally divided into 2 phases; in phase 1 we transform the matrix in into Hessenberg form in a finite number of steps, and in phase 2 we apply a (possibly infinite) number of transformations to transform the Hessenberg matrix into upper triangular form

\[
\begin{bmatrix}
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & *
\end{bmatrix}
\quad \text{Phase 1}
\quad \begin{bmatrix}
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & *
\end{bmatrix}
\quad \text{Phase 2}
\]

Where the “+”-entries are zeros if \(A\) is Hermitian (\(A = A^*\)).
The QR-Algorithm

The QR-algorithm can be viewed as a stable procedure for computing
QR-factorizations of the matrix powers $A$, $A^2$, $A^3$, ... 

Algorithm (The “Pure” QR-Algorithm)

$A(0) = A$

$k = 1$

while(...)

$[Q(k), R(k)] \leftarrow \text{qr}(A(k-1))$

$A(k) \leftarrow R(k)Q(k)$

$k \leftarrow k+1$

endwhile

Under suitable (non-restrictive) assumptions, this simple algorithm
converges to a Schur form for the matrix $A$ — Upper triangular if $A$ is
arbitrary, and diagonal if $A$ is Hermitian.

Before we go any further, let us illustrate this numerically...

Improvements... for Cubic Convergence
Detour: Simultaneous Iteration

The QR-Algorithm: Applied to $A = A^*$

Just a quick sanity-check... If $QR = A$, then

$$Q^*AQ = Q^*(QR)Q = (Q^*Q)RQ = RQ$$

Hence the matrices that we form are unitarily similar

$$A(k) = Q(1) \cdots Q(k)A(0)Q(k)^* \cdots Q(1)^*$$

In fact, this is the idea we had to reject in our effort to compute
the Hessenberg form of $A$. Even though it has to be rejected as a
finite-step method for transforming $A$, it turns out to be quite
powerful as the basis of an iterative scheme.

As in the last lecture, in order to keep the discussion simple(r), we
assume that $A \in \mathbb{R}^{m \times m}$, and $A = A^*$ so that $\lambda_k(A) \in \mathbb{R}$, and the
set of eigenvectors is orthonormal.
The QR-Algorithm: Modifications

Since we will be applying the QR-algorithm to real symmetric matrices, we are looking for the diagonalization $\Lambda(A)$.

Like the Rayleigh quotient algorithm, the QR-algorithm (for real symmetric matrices) can be made to converge cubically. In order to achieve this, we must introduce three modifications:

1. Before entering the iteration, $A$ must be reduced to tri-diagonal form (using the "Hessenberg algorithm" (Phase#1)).
2. Instead of $A(k)$, the shifted matrix $(A(k) - \mu(k)I)$ is factored at each step, where $\mu(k)$ is an eigenvalue estimate.
3. Whenever possible, and in particular whenever an eigenvalue is found, the problem is “deflated” by breaking $A(k)$ into sub-matrices.

The Modified QR-Algorithm: Components

1. We have already discussed reduction to Hessenberg form.
2. We will return to a discussion on selecting the shifts $\mu(k)$.
3. We leave the discussion on deflation as “an exercise for the motivated student.” (There are many details, e.g. pivoting to split the matrix exactly in “half,” to be taken care of to make this step maximally efficient)

For now, we focus the discussion on the “pure” form of the QR-algorithm... We relate the QR-algorithm to another method — simultaneous iteration — whose behavior is more intuitive.

The Modified QR-Algorithm

Algorithm (Modified QR-Algo)

\[ A(0) \leftarrow \text{hessenberg\_form}(A) \]
\[ \delta \leftarrow \text{small tolerance} \sim \sqrt{\varepsilon_{\text{mach}}} \]
\[ k \leftarrow 0 \]
while(...)  
\[
\text{Select } \mu(k)  
\[ [Q(k) \cdot R(k)] \leftarrow qr(\{A(k-1) - \mu(k)I\}) \]
\[ A(k) \leftarrow R(k)Q(k) + \mu(k)I \]
\]
if any 1st super-diagonal $|A(k)_{jj+1}| \leq \delta$ then
\[
\text{Set } A(k)_{jj+1} = A(k)_{j+1j} = 0, \text{ so that} 
\]
\[
\begin{bmatrix} A(k)[1] & 0 \\ 0 & A(k)[2] \end{bmatrix} = A(k) 
\]
\]
recursively apply QR-algorithm to $A(k)[1]$ and $A(k)[2]$  
endif  
\[ k \leftarrow k + 1 \]
while

We will see that the bottom right entry is where we get fastest (cubic) convergence. A naive pivoting strategy which tries to split the matrix into two blocks of equal size fails in that the upper-right and lower-left blocks are not empty. “Some” more work is required.
Unnormalized Simultaneous Iteration 1 of 4

**The Idea:** Apply the power iteration to several vectors at once.

Suppose we have a set of *linearly independent* vectors \( \{ v_1^{(0)}, \ldots, v_n^{(0)} \} \), then the space spanned by the vectors \( \{ A^k v_1^{(0)}, \ldots, A^k v_n^{(0)} \} \) generated by simultaneous power iteration, converges to the space spanned by the \( n \) eigenvectors \( \bar{q}_k \) corresponding to the \( n \) abs-largest eigenvalues \( |\lambda_k| > 0 \), i.e.

\[
\lim_{k \to \infty} \text{span} \left( A^k v_1^{(0)}, \ldots, A^k v_n^{(0)} \right) = \text{span} \left( \bar{q}_1, \ldots, \bar{q}_n \right) .
\]

In matrix form

\[
V(0) = \begin{bmatrix} \bar{v}_1^{(0)} & \cdots & \bar{v}_n^{(0)} \end{bmatrix}, \quad V(k) = A^k V(0) = \begin{bmatrix} \bar{v}_1^{(k)} & \cdots & \bar{v}_n^{(k)} \end{bmatrix}
\]

Unnormalized Simultaneous Iteration 3 of 4

We need one further assumption before we can state a theorem — Let \( \tilde{Q} \) be the \( (m \times n) \) matrix whose columns are the eigenvectors \( \bar{q}_k \). We need the following to be true

**Assumption #2**

*All the leading principal sub-matrices of \( \tilde{Q}^* V(0) \) are non-singular.*

A leading principal sub-matrix is anchored in the upper left corner (the \( m_{11} \)-element) and is a square matrix of size \((1 \times 1), (2 \times 2), \ldots, (n \times n)\).

With these assumptions we can say something about how the vectors generated by the simultaneous iteration converge to the eigenvectors.
Simultaneous Iteration

We have a problem: As \( k \to \infty \), all the vectors \( \vec{v}_1^{(k)}, \ldots, \vec{v}_n^{(k)} \) in the unnormalized simultaneous iteration converge to the same dominant eigenvector \( \vec{q}_1(A) \).

Even though \( \text{span} \left( \vec{v}_1^{(k)}, \ldots, \vec{v}_n^{(k)} \right) \) converges to something useful, i.e. \( \text{span} \left( \vec{q}_1, \ldots, \vec{q}_n \right) \), these vectors constitute a highly ill-conditioned basis (nearly linearly dependent basis) for that space. For practical purposes, this approach is useless.

The fix is straight-forward:

**Necessary Improvement**

We must orthonormalize the basis in every iteration. Instead of forming the sequence \( V_k \), we form a sequence \( Z_k \) with the same column spaces / spans / images, but where \( Z_k \) is orthonormal.

---

Simultaneous Iteration ⇔ QR-Algorithm

The QR-algorithm is equivalent to simultaneous iteration applied to the full set \( (n = m) \) of initial vectors, i.e. \( Q_0 = I_{m \times m} \).

We are now dealing with the full QR-factorizations, so we drop the hats on \( Q_k \), and \( R_k \). Further, let \( Q_k \) denote the matrices generated by the simultaneous iteration, and \( Q_k \) be the matrices generated by the QR-algorithm...

<table>
<thead>
<tr>
<th>Simultaneous Iteration</th>
<th>Pure QR-Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q_0 ) = ( I )</td>
<td>( A_0 ) = ( A )</td>
</tr>
<tr>
<td>( Z(k) ) = ( A Z_{(k-1)} )</td>
<td>( Q(k) R_{(k)} ) = ( A^{(k-1)} )</td>
</tr>
<tr>
<td>( Q_k R_{(k)} ) = ( Z(k) )</td>
<td>( A(k) ) = ( R_{(k)} Q_{(k)} )</td>
</tr>
<tr>
<td>( A^{(k)} ) = ( (Q_{(k)})^* A Q_{(k)} )</td>
<td>( Q_{(k)} = Q_1 Q_2 \cdots Q_k )</td>
</tr>
</tbody>
</table>

Table: The operations and quantities that define the Simultaneous Iteration algorithm and Pure QR-Algorithm.

---

Simultaneous Iteration

**Algorithm (Simultaneous Iteration)**

Let \( \hat{Q}_0(0) \in \mathbb{R}^{m \times n} \) with orthonormal columns

\[
\begin{align*}
k &= 0 \\
\text{while} & (\ldots) \\
\hat{Z}_1(0) & \leftarrow A \hat{Q}_0(0) \\
[\hat{Q}_k, \hat{R}_k] & \leftarrow \text{qr} (\hat{Z}_1(0)) \\
k & \leftarrow k + 1
\end{align*}
\]

Clearly, the column spaces / spans / images of \( Q_k \) and \( Z_k \) are the same.

Also, as long as the initial matrices \( (\hat{Q}_0(0)) \) are the same, this algorithm generates the same sequence \( Q_k \) as the unnormalized simultaneous iteration.

For the price of a QR-factorization per iteration we get a much better conditioned sequence of basis for the space; \( \text{span} (\vec{z}_1^{(k)}, \ldots, \vec{z}_n^{(k)}) \to \text{span} (\vec{q}_1, \ldots, \vec{q}_n) \).

**Theorem**

The Simultaneous Iteration algorithm and the Pure QR-algorithm generate identical sequences of matrices \( R_k, Q_k, A_k \), namely those defined by the QR-factorization of \( A^k \),

\[
Q_k R_k = A_k,
\]

together with the projection (similarity relation)

\[
A_k = (Q_k)^* A Q_k.
\]

This is not obvious at first glance, so let’s look at the proof...
Equivalence of Simultaneous Iteration and QR-Algorithm

**Proof:** [By Induction] The base case \( k = 0 \) is trivial, for both SI and QR-Alg we immediately see that
\[
A^0 = Q(0) = R(0) = I, \quad A^{(0)} = A,
\]
from which
\[
A^0 = Q(0)^*R(0); \quad A^{(0)} = (Q(0))^*AQ(0). \quad \sqrt{1}
\]
Now, consider \( k \geq 1 \) for SI: The second part of the theorem is valid by definition — \( A^{(k)} = (Q^{(k)})^*AQ^{(k)} \). The first part follows from
\[
A^k \overset{[1]}{=} AQ^{(k−1)}R^{(k−1)} \overset{[2]}{=} Q^{(k)}R^{(k)}R^{(k−1)} \overset{[3]}{=} Q^{(k)}R^{(k)}
\]
\[1\] Follows from the inductive hypothesis \( A^{k−1} = Q^{(k−1)}R^{(k−1)} \)
\[2\] From \( Q^{(k)}R^{(k)} = Z^{(k)} = AQ^{(k−1)} \)
\[3\] From \( R^{(k)} = R^{(k)}R^{(k−1)} \cdots R^{(1)} \)

\[3\] From \( Q^{(k)}R^{(k)} = Z^{(k)} = AQ^{(k−1)} \), \( Q^{(k)} = Q^{(1)}Q^{(2)} \cdots Q^{(k)}, \) and
\[
R^{(k)} = R^{(k)}R^{(k−1)} \cdots R^{(1)} \quad \square
\]

Equivalence of Simultaneous Iteration and QR-Algorithm

Finally, we verify the second part by the sequence
\[
A^{(k)} \overset{[1]}{=} (Q^{(k)})^*A^{(k−1)}Q^{(k)} \overset{[2]}{=} (Q^{(k)})^*AQ^{(k)}
\]
\[1\] Follows from \( Q^{(k)}R^{(k)} = A^{(k−1)} \), and \( A^{(k)} = R^{(k)}Q^{(k)} \)
\[2\] From the inductive hypothesis \( A^{(k−1)} = (Q^{(k−1)})^*AQ^{(k−1)} \), and \( Q^{(k)} = Q^{(1)}Q^{(2)} \cdots Q^{(k)} \) \( \square \)

Let’s put together the pieces of the “QR-Algorithm Jigsaw Puzzle”

- The relations (from the theorem)
  \[
  Q^{(k)}R^{(k)} = A^k, \text{ tell us why we expect to find the eigenvectors — the QR-Algorithm constructs orthonormal bases for successive powers of } A^k
  \]
  \[
  A^k = (Q^{(k)})^*AQ^{(k)} \text{ explain why we find the eigenvalues — the diagonal elements of } A^k \text{ are the Rayleigh coefficients of } A \text{ corresponding to the columns of } Q^{(k)}. \text{ As the columns converge to eigenvectors, the Rayleigh coefficients converge ("quadratically faster") to the corresponding eigenvalues. Since } Q^{(k)} \text{ converge to an orthonormal matrix, the off-diagonal elements in } A^k \text{ must converge to zero.}
  \]
Theorem

Let the pure QR-Algorithm be applied to a real symmetric matrix $A$ whose eigenvalues satisfy $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_m|$ and whose corresponding eigenvector matrix $Q$ has all non-singular leading principal sub-matrices. Then as $k \to \infty$, $A^{(k)}$ converges linearly with constant $\max_{1 \leq j < n} \left| \frac{\lambda_{j+1}}{\lambda_j} \right|$ to diag$(\lambda_1, \ldots, \lambda_m)$ and $Q^{(k)}$ converges at the same rate to $Q \ (mod \ 1 \cdot \vec{q}_j^{(k)})$.

Next, we look into adding shifts to the QR-Algorithm in order to speed up the convergence.