Lecture 16: Eigenvectors / Eigenvalues - Practical QR

June 30, 2025

Outline

Simultaneous (aka Block Power) Iteration

- **Motivating Question:** How can a simple algorithm possibly work to give us all eigenvector/eigenvalue pairs?
- We first consider the simpler-to-analyze **simultaneous iteration**.
- Then, we argue that the simultaneous iteration is **equivalent** to the QR Iteration.

Simultaneous (aka Block Power) Iteration

- Simultaneous iteration (aka **block** power iteration) is when we apply power iteration to **several** vectors at once, while maintaining linear independence among them.
- Start with a set of p orthonormal vectors, $v_1^{(0)}, v_2^{(0)}, \ldots, v_p^{(0)}$.
- The matrix multiplication $A^k v_1^{(0)}$ converges to q_1 as $k \to \infty$, where $|\lambda_1|$ is the largest (as seen in Lecture 14).
- However, span $\left\{ A^k v_1^{(0)}, \ldots, A^k v_p^{(0)} \right\}$ also converges to span $\{q_1, q_2, \ldots, q_p\}$, where $\lambda_1, \ldots, \lambda_p$ are the p largest magnitude eigenvalues.

Simultaneous (aka Block Power) Iteration

- In matrix form, denote $V^{(0)} = \begin{bmatrix} v_1^{(0)} & v_2^{(0)} & \cdots & v_p^{(0)} \end{bmatrix}$ and $V^{(k)} = A^k V^{(0)}$.
- With the multiplication of $A^k V^{(0)}$ all the vectors are ultimately converging to (multiples of) q_1.
- This provides a **very** ill-conditioned basis for the space of eigenvectors.
- The solution is to **orthonormalize** the vectors at each step using QR factorization.

Remarks:

1. Our algorithm would fall apart if we ever had $A^k v_\ell^{(0)} = A^k v_m^{(0)}$, where $\ell \neq m$. Why can this not happen? This would mean that A maps different input vectors to the same output vector. In other words, A is not of full rank.

Simultaneous (aka Block Power) Iteration

Algorithm 1 gives pseudocode for the simultaneous iteration.

Algorithm 1 Simultaneous Iteration Algorithm

Pick initial $\hat{Q}^{(0)} \in \mathbb{R}^{n \times p}$ with orthonormal columns
for $k = 1, 2, \ldots$
  $Z^{(k)} = A\hat{Q}^{(k-1)}$ ▷ (Block) power iteration step
  $Z^{(k)} = \hat{Q}^{(k)}\hat{R}^{(k)}$ ▷ (Reduced) QR factorization
end for

The column spaces of $\hat{Q}^{(k)}$ and $Z^{(k)}$ are the same, and they both are equal to the column space of $A^k\hat{Q}^{(0)}$ (could prove by induction on k).

Simultaneous (aka Block Power) Iteration

Similar to power iteration, the simultaneous iteration relies on two assumptions:

1. The leading $p+1$ eigenvalues are distinct in absolute value, i.e.

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_p| > |\lambda_{p+1}| \geq |\lambda_{p+2}| \geq \cdots \geq |\lambda_n|, \text{ and}$$

2. all of the leading principal submatrices of $(\hat{Q}^{(0)})^T V^{(0)}$ are non-singular (i.e. none of the p-many initial guess vectors comprising $V^{(0)}$ is orthogonal to the subspace generated by the first p-many eigenvectors).

Simultaneous (aka Block Power) Iteration

With the above assumptions we have the following theorem that states that the simultaneous iteration converges linearly.

Theorem 1 (Simultaneous iteration convergence)

Suppose the simultaneous iteration is applied and the preceding two assumptions are satisfied. Then as $k \to \infty$,

$$\left\| q_j^{(k)} - (\pm q_j) \right\| = O(c^k), \quad j = 1, 2, \cdots, p,$$

where $c = \max_{1 \le k \le p} \left| \frac{\lambda_{k+1}}{\lambda_k} \right| < 1.$

Remarks:

1. Why we need $\pm$ in front of the true eigenvector, q_j, here: We might converge to the negative of the true eigenvector, which is fine. In this case, without $\pm$, the convergence would not work as stated.
2. The ratio $\left| \frac{\lambda_{k+1}}{\lambda_k} \right|$ is present for the same reasons as in Power Iteration.

Simultaneous Iteration vs. QR Iteration

- In this section we show that the QR Iteration is identical to the simultaneous iteration when $\hat{Q}^{(0)} = I$ and $p = n$.
- Since the matrices are square, we will drop the hats on $\hat{Q}$ and $\hat{R}$.
- We will use the following notation:
  - $Q^{(k)}$ for Q's from QR Iteration,
  - $\underline{Q}^{(k)}$ for Q's from simultaneous iteration, i.e. $\underline{Q}^{(k)} = Q^{(1)}Q^{(2)}\cdots Q^{(k)}$, similar to the Householder notation.

Simultaneous Iteration vs. QR Iteration

Consider the QR Iteration and simultaneous iteration algorithms shown side by side below. We add steps (C) and (D) just for the upcoming proof of Theorem 2.

Algorithm 2 Simultaneous Iteration

$\underline{Q}^{(0)} = I$
for $k = 1, 2, \ldots$
$Z^{(k)} = A\underline{Q}^{(k-1)}$ ▷ (A)
$Z^{(k)} = \underline{Q}^{(k)} R^{(k)}$ ▷ (B)
$A^{(k)} = \left(\underline{Q}^{(k)}\right)^T A\underline{Q}^{(k)}$ ▷ (C)
$\underline{R}^{(k)} = R^{(k)} R^{(k-1)} \cdots R^{(1)}$ ▷ (D)
end for

Algorithm 3 QR Iteration

$A^{(0)} = A$
for $k = 1, 2, \ldots$
$A^{(k-1)} = Q^{(k)} R^{(k)}$ ▷ (A)
$A^{(k)} = R^{(k)} Q^{(k)}$ ▷ (B)
$\underline{Q}^{(k)} = Q^{(1)} Q^{(2)} \cdots Q^{(k)}$ ▷ (C)
$\underline{R}^{(k)} = R^{(k)} R^{(k-1)} \cdots R^{(1)}$ ▷ (D)
end for

Simultaneous Iteration vs. QR Iteration

Q & A

1. In the Simultaneous Iteration algorithm, should line (A) be corrected to
$$Z^{(k)} = A^{(k-1)}\underline{Q}^{(k-1)}?$$
A: No. $\underline{Q}^{(k)}$ changes at ever iteration. The original A is used to compute $Z^{(k)}$ from $\underline{Q}^{(k-1)}$. The given notation is correct.
2. Then why do we need $A^{(k)}$ here at all?
A: We don't need it for the computation itself. We will need it later, for convergence purposes.

Simultaneous Iteration vs. QR Iteration

Theorem 2

The QR Iteration and simultaneous iteration algorithms generate identical sequences of matrices, $\underline{R}^{(k)}, \underline{Q}^{(k)}, A^{(k)}$ *satisfying*

$$\begin{aligned} A^k &= \underline{Q}^{(k)}\underline{R}^{(k)}, \text{ (QR factorization of } \mathbf{k^{th}} \text{ power of } A) \\ A^{(k)} &= \left(\underline{Q}^{(k)}\right)^T A\underline{Q}^{(k)}. \text{ (Similarity transform of } A) \end{aligned}$$

Remark: As a consequence, convergence for simultaneous iteration will work the same as it does for QR iteration.

Proof. We will show

1. $A^k = \underline{Q}^{(k)}\underline{R}^{(k)}$, and
2. $A^{(k)} = (\underline{Q}^{(k)})^T A Q^{(k)}$,

separately for both algorithms by induction on k. Note the distinction that A^k is a matrix exponential ($A^k = \underbrace{AA\cdots A}_{k \text{ times}}$), whereas, $A^{(k)}$ is a matrix on the kth iteration.

Simultaneous Iteration vs. QR Iteration

The base case $k = 0$ is trivial.

1. $A^0 = \underline{Q}^{(0)} = \underline{R}^{(0)} = I$,
2. $A^{(0)} = A$.

Now, assuming the equations hold for $k - 1$, we will now show they hold for k.

Simultaneous Iteration vs. QR Iteration

Simultaneous Iteration:

1. $$\begin{aligned} & A^k \\ = \; & AA^{k-1}, \\ = \; & A\underline{Q}^{(k-1)}\underline{R}^{(k-1)}, \\ & \text{by inductive hyp. (1)}, A^{k-1} = \underline{Q}^{(k-1)}\underline{R}^{(k-1)} \\ = \; & \underline{Q}^{(k)}R^{(k)}\underline{R}^{(k-1)}, \\ & \text{by alg. (A) and (B)}, A\underline{Q}^{(k-1)} = Z^{(k)} = \underline{Q}^{(k)}R^{(k)} \\ = \; & \underline{Q}^{(k)}\underline{R}^{(k)}. \\ & \text{by def. } R^{(k)} \text{ as in alg. (D)} \end{aligned}$$

2. holds directly by (C).

Simultaneous Iteration vs. QR Iteration

QR Iteration:

1.

$$
\begin{aligned}
& A^k \\
= \; & AA^{k-1}, \\
= \; & A\underline{Q}^{(k-1)}\underline{R}^{(k-1)}, \\
& \text{by inductive hyp. (1)} \\
= \; & \underline{Q}^{(k-1)}A^{(k-1)}\underline{R}^{(k-1)}, \\
& \text{by inductive hyp. (2), i.e., } A\underline{Q}^{(k-1)} = \underline{Q}^{(k-1)}A^{(k-1)} \\
= \; & \underline{Q}^{(k-1)}\left(Q^{(k)}R^{(k)}\right)\underline{R}^{(k-1)}, \\
& \text{by alg. (A)} \\
= \; & \underline{Q}^{(k)}\underline{R}^{(k)}. \\
& \text{by def. } \underline{Q}, \underline{R}^{(k)} \text{ in (C), (D)}
\end{aligned}
$$

Simultaneous Iteration vs. QR Iteration

2.

$$
\begin{aligned}
& A^{(k)} \\
= \quad & R^{(k)} Q^{(k)}, \text{ by alg. (B)} \\
= \quad & \left(Q^{(k)}\right)^T A^{(k-1)} Q^{(k)}, \\
& \text{by alg. (A), i.e., } A^{(k-1)} = Q^{(k)} R^{(k)} \\
= \quad & \left(Q^{(k)}\right)^T \left(\underline{Q}^{(k-1)}\right)^T A \underline{Q}^{(k-1)} Q^{(k)}, \\
& \text{by inductive hyp. (2)} \\
= \quad & \left(Q^{(k)}\right)^T A \left(Q^{(k)}\right). \\
& \text{by def. } \underline{Q}^{(k)} \text{ in (C)}
\end{aligned}
$$

Therefore the two algorithms yield the same matrix sequences for $\underline{R}^{(k)}$, $\underline{Q}^{(k)}$ and $A^{(k)}$. $\square$

Simultaneous Iteration vs. QR Iteration - Convergence of QR Iteration

- Observe that $A^k = \underline{Q}^{(k)} \underline{R}^{(k)}$ implies the QR Iteration is computing QR factors of A^k, i.e., an orthonormal basis of A^k.
- Also, $A^{(k)} = \left(\underline{Q}^{(k)}\right)^T A \underline{Q}^{(k)}$ implies that diagonal entries of $A^{(k)}$ are the **Rayleigh quotients** for column vectors in $\underline{Q}^{(k)}$.
- Recall the Rayleigh quotient is $r(x) = \frac{x^T A x}{x^T x}$.
- As the columns of $\underline{Q}^{(k)}$ approach eigenvectors, these Rayleigh quotients approach the corresponding eigenvalues.

Simultaneous Iteration vs. QR Iteration - Convergence of QR Iteration

- What about off-diagonal entries of $A^{(k)}$? That is, with $i \neq j$

$$A_{ij}^{(k)} = \left(\underline{q_i}^{(k)}\right)^T A\underline{q_j}^{(k)},$$

  where $\underline{q_i}^{(k)}, \underline{q_j}^{(k)}$ are columns of $\underline{Q}^{(k)}$.

- As $\underline{q_i}^{(k)}, \underline{q_j}^{(k)}$ converge to (orthonormal) eigenvectors, q_i, q_j, then

$$A_{ij}^{(k)} \approx q_i^T A q_j = q_i(\lambda q_j) \approx 0, \text{ for } i \neq j.$$

- Hence, $A^{(k)}$ converges to a diagonal matrix.

Simultaneous Iteration vs. QR Iteration - Convergence of QR Iteration

Theorem 3 (QR Iteration convergence)

Assume $|\lambda_1| > |\lambda_2| > \cdots |\lambda_n|$ *and the corresponding eigenvector matrix* Q *has nonsingular leading principal submatrices. Then, as* $k \to \infty$, $A^{(k)}$ *converges linearly to* $\text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ *with constant*

$$C = \max_k \left| \frac{\lambda_{k+1}}{\lambda_k} \right|.$$

The matrix $\underline{Q}^{(k)}$ *also converges linearly to* Q *with the same constant* C.

- For more details about the QR Iteration see Lecture 28 of Trefethen & Bau.

Simultaneous Iteration vs. QR Iteration - Eigenvalue Problems Recap

Here we give a recap of what we have discussed so far for eigenvalue problems.

- We used the Rayleigh quotient to recover an estimated eigenvalue, given an estimated eigenvector.
- We saw the power iteration, inverse iteration, shifted inverse iteration, and Rayleigh quotient iteration, as methods to recover **individual** eigenvectors.
- We introduced two schemes to find **multiple** eigenvectors/eigenvalues at once:
  - QR Iteration,
  - Simultaneous (block power) iteration.

Simultaneous Iteration vs. QR Iteration - Eigenvalue Problems Recap

- Dense QR factorization at **every single step** of the QR Iteration algorithm is costly.
- This takes approximately $\frac{4}{3}n^3$ flops.
- In the next section we look at a way to make the QR Iteration more practical (efficient).
- The idea is to first pre-process A (with another similarity transform) to increase sparsity!
- If A is non-symmetric, one can reduce to upper Hessenberg form.
- Upper Hessenberg matrices only require $\rightarrow O(n^2)$ flops for QR factorization.
- If A is symmetric we can reduce to a tridiagonal matrix, which requires only $\rightarrow O(n)$ flops for QR factorization.
- Exercise: derive efficient QR factorizations of UH and tridiagonal matrices.

Reduction to Upper Hessenberg

- In the general case A can be non-symmetric.
- One might ask why would we reduce to just upper Hessenberg form and not triangular?
- Wouldn't having a triangular matrix be even cheaper for QR factorization?
- Let's consider this by attempting to reduce to triangular instead.

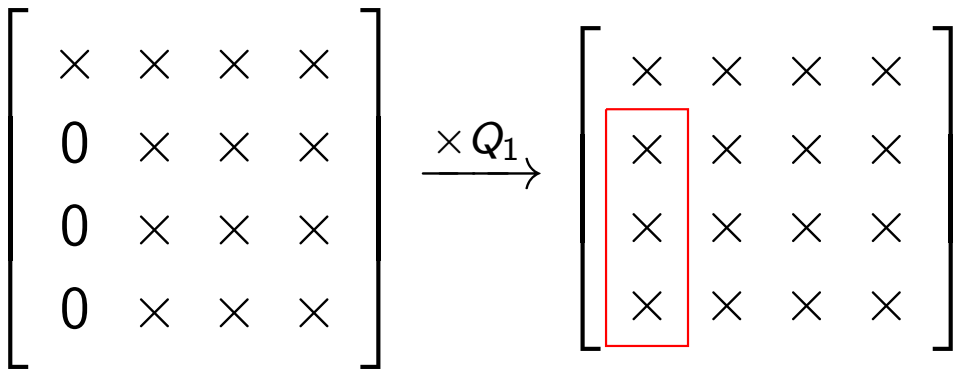
Reduction to Upper Hessenberg - First attempt

Try to reduce A to **triangular** via usual Householder. Apply Householder Q_1 to A.

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \xrightarrow{Q_1^T \times} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} = Q_1^T A$$

Reduction to Upper Hessenberg - First attempt

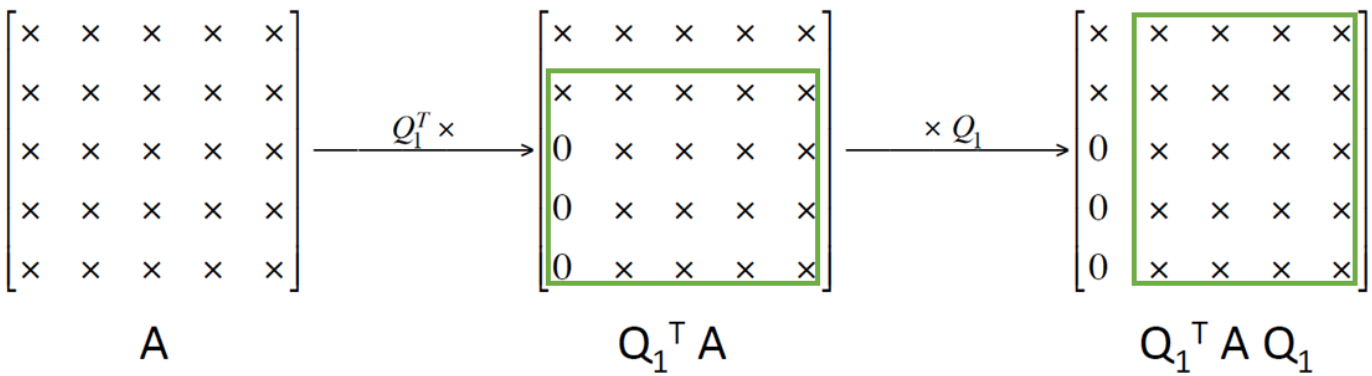
But to maintain similarity, also need to multiply by Q_1 on the **right**



So with Householder reflections our newly created zeros are just destroyed again!

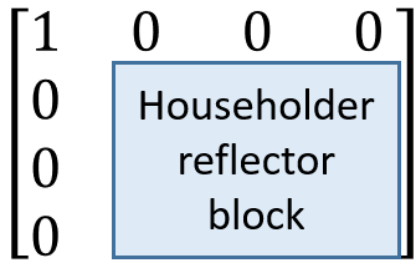
Reduction to Upper Hessenberg - Second attempt

We will be a little less ambitious and choose a different Q_1^T that leaves the **whole row untouched**. Then, when we multiply by Q_1 on the right, it won't destroy our progress (i.e., it will leave the 1^{st} column alone).



Reduction to Upper Hessenberg - Second attempt

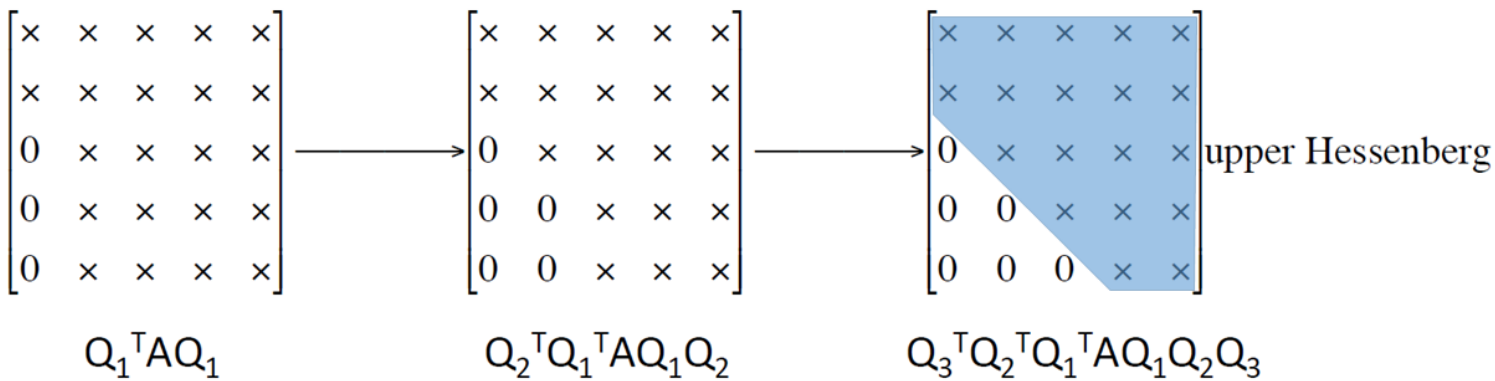
To achieve this, the first Q matrix will have a form like:



This leaves the desired first row/column alone after computing $Q_1^T A Q_1$, preserving the new zeros.

Reduction to Upper Hessenberg - Second attempt

We apply the same idea to subsequent columns (similar to Householder QR).



This gives $Q = Q_1 Q_2 \cdots Q_{n-2}$ and $Q^T AQ =$ an upper Hessenberg matrix. Algorithm 4 gives the pseudocode for the reduction to Hessenberg form.

Reduction to Upper Hessenberg - Second attempt

Algorithm 4 Reduction to Hessenberg

for $k = 1, 2, \ldots, n-2$
  $x = A(k+1:n, k)$
  $v_k = \text{sign}(x_1)||x||e_1 + x$ ▷ Householder reflection
  $v_k = \frac{v_k}{\|v_k\|}$ ▷ Normalize
  for $j = k, k+1, \ldots, n$ ▷ Left multiply, $Q_k^T \times$
    $A(k+1:n, j) = A(k+1:n, j) - 2v_k \left(v_k^T A(k+1:n, j)\right)$
  end for
  for $j = 1, 2, \ldots, n$ ▷ Right multiply, $\times Q_k$
    $A(i, k+1:n) = A(i, k+1:n) - 2\left(A(i, k+1:n)v_k\right) v_k^T$
  end for
end for

The cost is flops(Reduction to Hessenberg) $\approx \frac{10}{3}n^3$. However, this reduction to Hessenberg is done only **once**, before QR Iteration.

Reduction to Upper Hessenberg - Symmetric Matrices: Two-Phase Process

For the symmetric case, when $A = A^T$, then

$$(Q^T A Q)^T = Q^T A Q,$$

is also symmetric. A matrix that is both symmetric **and** upper Hessenberg is necessarily tridiagonal. Reduction will produce zeros above the diagonal as well. Sparsity and symmetry together reduce the cost of reduction from $\frac{10}{3}n^3$ to $\frac{4}{3}n^3$. The idea then to make the QR Iteration more practical is a two-phased process:

1. Reduce A to tridiagonal via Householder operations (direct).
2. Perform QR Iteration until convergence (iterative).

Reduction to Upper Hessenberg - Symmetric Matrices: Two-Phase Process

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \quad (A)$$

$$\xrightarrow[\text{Reduction}]{\text{Phase 1}} \quad \begin{bmatrix} \times & \times & & \\ \times & \times & \times & \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix} \quad (T = Q^T A Q)$$

$$\xrightarrow[\text{QR Iteration}]{\text{Phase 2}} \quad \begin{bmatrix} \times & & & \\ & \times & & \\ & & \times & \\ & & & \times \end{bmatrix} \quad (D)$$

Reduction to Upper Hessenberg - Symmetric Matrices: Two-Phase Process

QR Iteration can be additionally improved by:

- Applying shifting to achieve cubic convergence rates (similar to Rayleigh Quotient Iteration).
- Breaking $A^{(k)}$ into sub-matrices once an eigenvalue is found ("deflation").

Aside: The QR Iteration's Inventors

- John Francis published the (implicit, shifted) QR algorithm in 1961.
- It was named one of the ten "most important" algorithms of the 20th century.
- John Francis left the field of numerical analysis that same year.
- He was tracked down in 2007, and had no idea the huge influence of his work! **Concurrently**, the algorithm was invented by Vera Kublanovskaya, who continued to work in numerical analysis until passing away in 2012.

John Francis

Vera Kublanovskaya