Eigenvalue algorithms: The QR algorithm with shifts

MATH 6610 Lecture 20

October 26, 2020

Trefethen & Bau: Lecture 29

The QR algorithm

Assume A is Hermitian. The QR algorithm for computing eigenvalues:

1. Compute $A = QR$, the QR decomposition of A
2. Replace A by the procedure $A \leftarrow RQ$
3. Return to step 1

We've seen that this is just simultaneous power iteration.

The QR algorithm

Assume A is Hermitian. The QR algorithm for computing eigenvalues:

1. Compute $A = QR$, the QR decomposition of A
2. Replace A by the procedure $A \leftarrow RQ$
3. Return to step 1

We've seen that this is just simultaneous power iteration.

....which also means that it converges as “quickly” as power iteration.

Can we instead develop a Rayleigh iteration type of algorithm?

QR algorithm and inverse iteration, I

(Unshifted) inverse iteration: power iteration on A^{-1}.

The QR algorithm performs power iteration.

QR algorithm and inverse iteration, I

(Unshifted) inverse iteration: power iteration on A^{-1}.

The QR algorithm performs power iteration.
The QR algorithm also performs unshifted inverse iteration.

QR algorithm and inverse iteration, I

(Unshifted) inverse iteration: power iteration on A^{-1}.

The QR algorithm performs power iteration.
The QR algorithm <u>also</u> performs unshifted inverse iteration.

Recall: if Q_k^{QR} is the QR factor computed at the kth iteration, then

$$A^k = Q_k R_k, \qquad Q_k = Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR},$$

and R_k is an upper-triangular matrix.

QR algorithm and inverse iteration, II

Let F be a permutation matrix that flips a vector, i.e., associated to the permutation map:

$$\{1, 2, \ldots, n-1, n\} \longrightarrow \{n, n-1, \ldots, 2, 1\}.$$

Then:

QR algorithm and inverse iteration, II

Let F be a permutation matrix that flips a vector, i.e., associated to the permutation map:

$$\{1, 2, \ldots, n-1, n\} \longrightarrow \{n, n-1, \ldots, 2, 1\}.$$

Then:

$$A^{-k} F = (Q_k F) \left(F R_k^{-T} F \right),$$

which is a QR factorization of $A^{-k} F$.

I.e., the last column(s) of Q^k (computed via the QR algorithm) are inverse iteration with starting vectors given by the first columns of F.

The QR algorithm with shifts

We can almost perform Rayleigh iteration with QR procedures.
We can perform inverse iteration; how to accomplish shifting?

The QR algorithm with shifts

We can almost perform Rayleigh iteration with QR procedures.
We can perform inverse iteration; how to accomplish shifting?

Before explaining the shift, we show the result:
The QR algorithm with shifts.

Set $A_0^{QR} = A$ and $\mu_0 = 0$. For $k = 1, 2, \ldots,$

- $Q_k^{QR} R_k^{QR} = A_{k-1}^{QR} - \mu_{k-1} I$
- $A_k^{QR} = R_k^{QR} Q_k^{QR} + \mu_{k-1} I$
- "Choose" μ_k

The QR algorithm with shifts

We can almost perform Rayleigh iteration with QR procedures.
We can perform inverse iteration; how to accomplish shifting?

Before explaining the shift, we show the result:
The QR algorithm with shifts.

Set $A_0^{QR} = A$ and $\mu_0 = 0$. For $k = 1, 2, \ldots,$

- $Q_k^{QR} R_k^{QR} = A_{k-1}^{QR} - \mu_{k-1} I$
- $A_k^{QR} = R_k^{QR} Q_k^{QR} + \mu_{k-1} I$
- "Choose" μ_k

For quite a time, the QR algorithm with (properly chosen) shifts was the gold standard for computing eigenvalues.
(Not quite so widely used today, though.)

QR with shifts is shifted inverse iteration

We can see why the QR algorithm with shifts is shifted inverse iteration:

QR with shifts is shifted inverse iteration

We can see why the QR algorithm with shifts is shifted inverse iteration:

$$A_k^{QR} = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR} \right)^* \; A \; \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR} \right),$$

so that this algorithm still produces a matrix unitarily equivalent to A.

QR with shifts is shifted inverse iteration

We can see why the QR algorithm with shifts is shifted inverse iteration:

$$A_k^{QR} = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right)^* \; A \; \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right),$$

so that this algorithm still produces a matrix unitarily equivalent to A.

The second critical property is that

$$(A - \mu_0 I)(A - \mu_1 I) \cdots (A - \mu_{k-1} I) = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right)\left(R_k^{QR} R_{k-1}^{QR} \cdots R_1^{QR}\right).$$

QR with shifts is shifted inverse iteration

We can see why the QR algorithm with shifts is shifted inverse iteration:

$$A_k^{QR} = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right)^* \; A \; \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right),$$

so that this algorithm still produces a matrix unitarily equivalent to A.

The second critical property is that

$$\begin{aligned}(A - \mu_0 I)(A - \mu_1 I) \cdots (A - \mu_{k-1} I) = \\ \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right)\left(R_k^{QR} R_{k-1}^{QR} \cdots R_1^{QR}\right).\end{aligned}$$

I.e., the QR algorithm with shifts computes a QR decomposition for a type of shifted simultaneous iteration.

- The first column of $\prod_{j=1}^k Q_k^{QR}$ is "shifted" power iteration, on e_1.
- The last column of $\prod_{j=1}^k Q_k^{QR}$ is shifted inverse iteration, on e_n.

If the shifts are chosen well: the last column of $\prod_{j=1}^k Q_k^{QR}$ is an eigenvector of A.

QR with shifts

$$A_k^{QR} = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right)^* \; A \; \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right),$$

and the last column of $\prod_{j=1}^{k} Q_k^{QR}$ is an eigenvector.

QR with shifts

$$A_k^{QR} = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right)^* \; A \; \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right),$$

and the last column of $\prod_{j=1}^{k} Q_k^{QR}$ is an eigenvector.

This implies that the last, (n,n) entry of A_k^{QR} for large k, is an eigenvalue of A.

QR with shifts

$$A_k^{QR} = \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right)^* \;\; A \;\; \left(Q_1^{QR} Q_2^{QR} \cdots Q_k^{QR}\right),$$

and the last column of $\prod_{j=1}^{k} Q_k^{QR}$ is an eigenvector.

This implies that the last, (n, n) entry of A_k^{QR} for large k, is an eigenvalue of A.

This also reveals a (simple!) deflation technique: if the $(n, 1)$, $(n, 2), \cdots, (n, n-1)$ entries of A_k^{QR} are all close to zero, then:

- the $(n-1) \times (n-1)$ principal submatrix of A_k^{QR} is a matrix whose eigenvalues matches the remaining eigenvalues of A.
- The QR algorithm with shifts can now be applied to this principal submatrix.

Rayleigh shifts

We haven't discussed how to choose the shifts μ_k.

From previous experience: using Rayleigh quotients seems like a good idea.

Rayleigh shifts

We haven't discussed how to choose the shifts μ_k.

From previous experience: using Rayleigh quotients seems like a good idea.

We know the last column, call it q, of $\prod_{j=1}^{k} Q_k^{QR}$ is close to an eigenvector.

Then:

$$R_A(q) = \left(A_k^{QR}\right)_{n,n}$$

Thus, computing Rayleigh quotients is <u>easy</u>.

Setting $\mu_k = \left(A_k^{QR}\right)_{n,n}$ is called a Rayleigh shift.

QR with shifts and details

The QR algorithm with shifts.

Set $A_0^{QR} = A$ and $\mu_0 = 0$. For $k = 1, 2, \ldots,$

- $Q_k^{QR} R_k^{QR} = A_{k-1}^{QR} - \mu_{k-1} I$
- $A_k^{QR} = R_k^{QR} Q_k^{QR} + \mu_{k-1} I$
- $\mu_k = \left(A_k^{QR}\right)_{n,n}$ (Rayleigh shift)
- If the last row of A_k^{QR} is a multiple of e_n:
  - μ_k is an eigenvalue of A,
  - Run this algorithm on the $(n-1) \times (n-1)$ principal submatrix of A_k^{QR}