Classes week of Nov. 2 (Nov. 2, 4, 6)

- cancel Mon. Nov. 2

- We won't really have lecture on Nov. 4, 6.
- I'll prepare slides for content
  + iterative methods (A)
  + nonlinear equations/optimization (B)
- But I won't deliver a lecture on Wed, Fri.
- Instead, I'll ask that if you decide to come to lecture, then review slides beforehand and we'll have an informal discussion.

- You are not ~~is~~ required to attend lecture on Nov. 4, 6.

- (A+B) are not on quals / final exam / HW.

Eigenvalue algorithms: The QR algorithm with shifts

MATH 6610 Lecture ~~20~~ 21

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.

Set $A_0^{QR} = A = V \Lambda V^*$

$k = 1, 2, \ldots$

$Q_k^{QR} R_k^{QR} = A_{k-1}^{QR}$

$A_k^{QR} = R_k^{QR} Q_k^{QR}$

1.) $A_{k-1}^{QR} \to A_k^{QR}$ is a unitary similarity transform.

2.) $A_k^{QR} \longrightarrow \Lambda$

Note: computationally, $A_{k-1}^{QR} \mapsto A_k^{QR}$ can be very efficient.

$A_{k-1}^{QR} \longrightarrow R_k^{QR}$ performs a sequence of Householder reflections.

$$R_k^{QR} = \underbrace{\left(\prod_{j=1}^{n} H_j\right)}_{(Q_k^{QR})^*} A_{k-1}^{QR}$$

$$A_k^{QR} = R_k^{QR} Q_k^{QR} = \left(\prod_{j=1}^{n} H_j\right) A_{k-1}^{QR} \left(\prod_{j=1}^{n} H_j\right)^*$$

$$= H_1 H_2 \cdots \underbrace{H_n A_{k-1}^{QR} H_n^*}_{\substack{\text{symmetric application of a} \\ \text{Householder reflector (efficient!)}}} H_{n-1}^* \cdots H_1^*$$

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 <u>also</u> 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 also performs unshifted inverse iteration.

from the QR algorithm

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

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

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:

$$\cancel{A^k F} \quad A^{-k} F = (A^k)^{-1} F$$

$$= (Q_k R_k)^{-1} F$$

$$= R_k^{-1} Q_k^* F \qquad \left(A \text{ Hermitian} \Rightarrow (A^{-k}) = (A^{-k})^* \right)$$

$$= (R_k^{-1} Q_k^*)^* F$$

$$= Q_k R_k^{-*} F$$

$$\underbrace{A^{-k}F}_{\text{simultaneous inverse iteration on starting matrix } F.} = \underbrace{Q_k F}_{\text{unitary}}\ \underbrace{F R_k^{-*} F}$$

R_k : upper triangular

R_k^{*} : lower triangular

R_k^{-*} : lower triangular

$F R_k^{-*} F$: upper triangular matrix

$\Rightarrow$ RHS is a QR decomposition of LHS.

Q factor $Q_k F$

$\Rightarrow$ $Q_k F$ is approximately an eigenvector matrix for $\cancel{A^{-k}}$. A^{-1}!

Since A is Hermitian, then $Q_k F$ is also an eigenvector matrix for A.

Q_k : computed (implicitly via QR algorithm)

$Q_k F$: first column of this is a good guess to a dominant eigenvector of A^{-1}

$\Rightarrow$ last column of Q_k is a good estimate to an eigenvector of A.

Q_k is computed via QR algorithm!

$\Rightarrow$ QR algorithm performs inverse iteration.

(as well as power iteration!)

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 $\cancel{Q^k}$ 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.

(or arbitrary)

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

- $Q_k^{QR} R_k^{QR} = A_{k-1}^{QR} - \mu_{k-1} I$ (QR decomp. of a shifted version of A_{k-1}^{QR})
- $A_k^{QR} = R_k^{QR} Q_k^{QR} + \mu_{k-1} I$
- "Choose" μ_k

Recall from Rayleigh iteration: chosen via Rayleigh quotient.

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.

Again, A_k^{QR} is unitarily equivalent to A.

Idea: $Q_k^{QR} R_k^{QR} = A_{k-1}^{QR} - \mu_{k-1} I$

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

$$A_k^{QR} = (Q_k^{QR})^* Q_k^{QR} R_k^{QR} Q_k^{QR} + \mu_{k-1} I$$

$$= (Q_k^{QR})^* [A_{k-1}^{QR} - \mu_{k-1} I] Q_k^{QR} + \mu_{k-1} I$$

$$= (Q_k^{QR})^* A_{k-1}^{QR} Q_k^{QR}$$

Induction yields the result.

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

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

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 $\prod_{j=1}^{k} Q_{\cancel{k}\, j}^{QR}$ is an eigenvector.

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

$$\tilde{V} = \prod_{j=1}^{k} Q_j^{QR} \qquad \tilde{V} = \begin{bmatrix} | & & | \\ \tilde{v}_1 & \cdots & \tilde{v}_n \\ | & & | \end{bmatrix}$$

$\tilde{v}_n$ is an eigenvector of A, $\|\tilde{v}_n\|_2 = 1$

$$A_k^{QR} = (\tilde{V})^* A \tilde{V} = \left(\langle A\tilde{v}_j, \tilde{v}_k \rangle \right)_{j,k=1}^{n}$$

$$A\tilde{v}_n = \lambda \tilde{v}_n$$

last row of $A_k^{QR} = \langle A\hat{v}_n, \hat{v}_k \rangle$, $k=1 \ldots n$

$= \lambda \langle \hat{v}_n, \hat{v}_k \rangle$, $k = 1 \ldots n$

$= \delta_{n,k} \lambda$ ($\hat{v}_k$ are orthogonal)

$$A_k^{QR} = \begin{pmatrix} & \boxed{\times} & \\ 0 & \cdots \; 0 & \lambda \end{pmatrix} \Longrightarrow (A_k^{QR})_{n,n} = \lambda$$

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.

$$A_k^{QR} = \begin{pmatrix} \boxed{\times} & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ 0 \cdots 0 & \lambda \end{pmatrix}$$

n-1 eigenvalues of this block match remaining n-1 eigenvalues of A.

Rayleigh shifts

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

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

(This is what Rayleigh iteration chooses.)

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 $\underbrace{\prod_{j=1}^{k} Q_k^{QR}}_{\widetilde{V}}$ is close to an eigenvector.

$\Rightarrow A_k^{QR} = (\widetilde{V})^* A (\widetilde{V})$

Then:

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

Thus, computing Rayleigh quotients is easy.

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.

or arbitrary.

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}

$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ $\mu_0 = 0 \Rightarrow A_k^{QR} = A \quad \forall k.$