Numerical Matrix Analysis

Notes #21 — Eigenvalues
The QR-Algorithm with Shifts

Peter Blomgren
⟨blomgren@sdsu.edu⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Spring 2024
(Revised: April 9, 2024)

Outline

1. The QR-Algorithm
    - Recap
2. Connections with Other Iterative Schemes
    - Inverse Iteration
    - Shifted Inverse Iteration
    - Rayleigh Quotient Iteration
3. Stability and Accuracy

Last Time: The QR-Algorithm

We introduced and discussed

Algorithm (The "Pure" QR-Algorithm)

$\mathbf{A}_{(0)} = \mathbf{A}$

```
while( ... )
    [Q_(k), R_(k)] = qr(A_(k-1))
    A_(k) = R_(k) Q_(k)
endwhile
```

Which iteratively transforms a general matrix to upper triangular form, and a Hermitian matrix to diagonal form.

Last Time: The QR-Algorithm

Through a rather long-winded argument using the simultaneous iteration we were able to argue that the following theorem is true

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* $\mathrm{diag}(\lambda_1, \ldots, \lambda_m)$ *and* $\underline{Q}_{(k)}$ *converges at the same rate to Q* $(\mathrm{mod} \pm 1 \cdot \underline{\vec{q}}_j^{(k)})$.

Where $\underline{Q}_{(k)} = Q_{(1)} Q_{(2)} \cdots Q_{(k)}$.

This time we look at introducing shifts into the QR-algorithm in order to speed up the convergence rate.

Connections with Other Iterative Schemes

We maintain the assumption that $A \in \mathbb{R}^{m\times m}$ is real and symmetric; with real eigenvalues $\lambda(A)$ and orthonormal eigenvectors $\{\vec{q}_j\}_{j=1,\dots,m}$.

We now make connections between the QR-algorithm and the three other iterative schemes we explored in our previous "detour" —

1. Inverse Iteration
2. Shifted Inverse Iteration
3. Rayleigh Quotient Iteration

With all these pieces in place, combined with the right shifting strategy, we can define a QR-algorithm with shifts, which generally **converges cubically**, and at least quadratically in the worst case.

Connection with Inverse Iteration

The "Pure" QR-algorithm is equivalent to the simultaneous iteration applied to the identity matrix (see [NOTES#20]), and **in particular**, the first column of the result evolves according to the power iteration applied to $\vec{e}_1$, the first standard unit vector.

There is a **dual** to this observation: The pure QR-algorithm is also equivalent to **simultaneous inverse iteration** applied to a particular permutation matrix P

$$P = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & \iddots & & \\ 1 & & & \end{bmatrix}.$$

In particular the m^{th} column of the QR-algorithm evolves according to inverse iteration applied to $\vec{e}_m$.

This is "less than" obvious at first glance...

Connection with Inverse Iteration

Let $Q_{(k)}$ be the orthogonal factor at the k^{th} step of the QR-algorithm, and let

$$\underline{Q}_{(k)} = \prod_{j=1}^{k} Q_{(j)} = \left[\ \vec{q}_1^{\,(k)} \mid \vec{q}_2^{\,(k)} \mid \cdots \mid \vec{q}_m^{\,(k)}\ \right].$$

This is the same orthogonal matrix that appears in the k^{th} step of simultaneous iteration, *i.e.* $\underline{Q}_{(k)}$ is the orthogonal factor in the QR-factorization

$$\underline{Q}_{(k)}\underline{R}_{(k)} = A^k.$$

Now, using the symmetry of A (and therefore of A^{-1}), we have

$$A^{-k} = (\underline{R}_{(k)})^{-1}(\underline{Q}_{(k)})^* \stackrel{\text{sym}}{=} \underline{Q}_{(k)}(\underline{R}_{(k)})^{-*}.$$

Connection with Inverse Iteration

Define the $(m \times m)$ permutation matrix P, which reverses the row (PA) or column (AP) order

$$P = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & \cdot^{\cdot^{\cdot}} & & \\ 1 & & & \end{bmatrix}, \quad P^2 = I, \quad P^* = P.$$

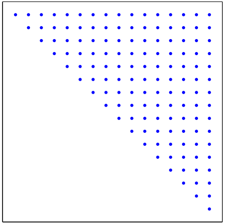
We can now rewrite

$$\textbf{[1]} \qquad A^{-k}\mathbf{P} = \underline{Q}_{(k)}\mathbf{P^2}(\underline{R}_{(k)})^{-*}\mathbf{P} = \left[\underline{Q}_{(k)}P\right]\left[P(\underline{R}_{(k)})^{-*}P\right].$$

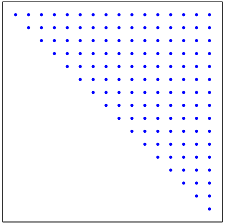
Where the first factor $\underline{Q}_{(k)}P$ is orthogonal, and the second $P(\underline{R}_{(k)})^{-*}P$ is upper triangular. Hence we can interpret **[1]** as a QR-factorization of $A^{-k}P$.

Connection with Inverse Iteration

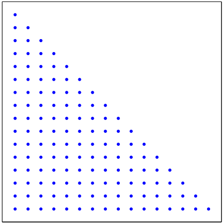
$\exists$ Movie



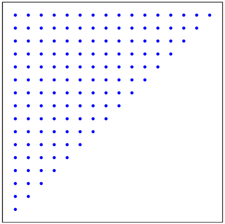
R



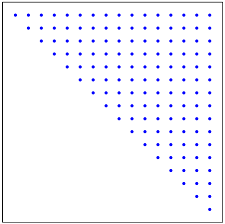
[Inverse] R^{-1}



[Transpose] R^{-*}



[Row-Flip] PR^{-*}



[Column-Flip] $PR^{-*}P$

Connection with Inverse Iteration

We are, in effect, carrying out simultaneous iteration on A^{-1} applied to the initial matrix P, *i.e* **simultaneous inverse iteration of A.**

The first column of $\underline{Q}_{(k)}P$, *i.e.* the last column of $\underline{Q}_{(k)}$ is the result of applying k steps of inverse iteration to $\vec{e}_m$.

Hence, the QR-algorithm is in some sense performing both simultaneous iteration and simultaneous inverse iteration — a sense of perfect symmetry!

From our previous discussion, we know that inverse iteration can be accelerated significantly by introducing appropriate shifts...

Connection with Shifted Inverse Iteration

We now consider the following modification to the QR-algorithm:

Algorithm (The QR-Algorithm with Shifts)

$\mathbf{A_{(0)}} = \texttt{hessenberg\_form}(\mathbf{A})$
`while( ... )`
    `Select` $\mu_{(\mathbf{k})}$
    $\left[\mathbf{Q_{(k)}}, \mathbf{R_{(k)}}\right]$ = $\texttt{qr}\left(\mathbf{A_{(k-1)}} - \mu_{(\mathbf{k})}\mathbf{I}\right)$
    $\mathbf{A_{(k)}} = \mathbf{R_{(k)}}\mathbf{Q_{(k)}} + \mu_{(\mathbf{k})}\mathbf{I}$
`endwhile`

The following relations still hold

$$A_{(k)} = (Q_{(k)})^* A_{(k-1)} Q_{(k)}, \quad A_{(k)} = (\underline{Q}_{(k)})^* A \underline{Q}_{(k)}.$$

However, the following relation is **no longer valid:** $A^k = \underline{Q}_{(k)} \underline{R}_{(k)}$.

Connection with Shifted Inverse Iteration

We recall that $A^k = \underline{Q}_{(k)}\underline{R}_{(k)}$ appears in the convergence theorem. Hence we may fear that we have lost convergence!

Fortunately, it turns out that the following holds

$$(A - \mu_{(k)}I)(A - \mu_{(k-1)}I)\ldots(A - \mu_{(1)}I) = \underline{Q}_{(k)}\underline{R}_{(k)}$$

and the proof of the theorem goes through with this modification.

The last column of $\underline{Q}_{(k)}$ is the result of applying k steps of shifted inverse iteration to $\vec{e}_m$ with the shift sequence $\{\mu_{(j)}\}_{j=1,\ldots,k}$.

If the shifts are good eigenvalue estimates, this last column converges quickly to an eigenvector.

The shifted inverse iteration is, in a manner of speaking, a **hidden treasure** inside the shifted QR-algorithm.

Connection with Rayleigh Quotient Iteration

To complete the argument, we must find a way of choosing the shifts so that we indeed achieve fast convergence in the last column of $\underline{Q}_{(k)}$.

It should come as no surprise that we use the Rayleigh quotient in order to generate our eigenvalue estimates $\mu_{(k)}$. We extract the last column of $\underline{Q}_{(k)}$, $\vec{q}_m^{(k)}$, and compute

$$\mu_{(k)} = \frac{(\vec{q}_m^{(k)})^* A \vec{q}_m^{(k)}}{\|\vec{q}_m^{(k)}\|_2^2} = (\vec{q}_m^{(k)})^* A \vec{q}_m^{(k)}.$$

If we use this shift, then the eigenvalue-eigenvector estimates $(\mu_{(k)}, \vec{q}_m^{(k)})$ are identical to the ones computed by the Rayleigh quotient iteration, starting with $\vec{e}_m$.

Hence, we inherit the **cubic convergence for** $(\mu_{(k)}, \vec{q}_m^{(k)})$.

Connection with Rayleigh Quotient Iteration

Good News: The Rayleigh quotient is "free." The (m, m)-entry of $A_{(k)}$ already contains the value

$$A_{(k),m,m} = \vec{e}_m^* A_{(k)} \vec{e}_m = \vec{e}_m^* (\underline{Q}_{(k)})^* A \underline{Q}_{(k)} \vec{e}_m = (\vec{q}_m^{(k)})^* A \vec{q}_m^{(k)},$$

and $\|\vec{q}_m^{(k)}\| = 1$.

Therefore, all we have to do is setting $\mu_{(k)} = A_{(k),m,m}$.

This strategy is known as the **Rayleigh Quotient Shift**.

Bad News: Although this strategy, in general, gives cubic convergence, there are matrices for which the strategy does not converge at all.

Connection with Rayleigh Quotient Iteration

Example of Rayleigh Quotient Shift Breakdown

$$A = \begin{bmatrix} 0 & 1 \\ 1 & \mathbf{0} \end{bmatrix}.$$

The Rayleigh-Shifted QR-algorithm gives, $\mu_{(1)} = \mathbf{0}$, and

$$\begin{aligned} Q_{(1)} R_{(1)} &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ A_{(1)} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = A. \end{aligned}$$

Rayleigh-shifting does not help since $\mu_{(k)} \equiv 0$.

Connection with Rayleigh Quotient Iteration

The problem with Rayleigh-shifting arises because of the symmetry of eigenvalues. In the example $\lambda(A) = \{-1, 1\}$.

With the initial estimate $\mu = 0$, we are "stuck in the middle" — there is no tendency to favor either eigenvalue, and hence the estimate is not improved.

We need a shifting strategy which can break the dead-lock...

Consider the lower-right corner of the matrix $A_{(k)}$, and let B denote the (2×2) sub-matrix anchored there, *i.e.*

$$B = \begin{bmatrix} a_{m-1} & b_{m-1} \\ b_{m-1} & a_m \end{bmatrix}.$$

Breaking the Deadlock: Wilkinson Shift

The **Wilkinson Shift** is the eigenvalue of B that is closest to a_m.

When there is a tie, the choice is arbitrary [But must be made!]. The shift can be implemented as

$$\mu_{\mathrm{W},(k)} = a_m - \frac{\mathrm{sign}(\delta) b_{m-1}^2}{|\delta| + \sqrt{\delta^2 + b_{m-1}^2}}, \quad \delta = \frac{a_{m-1} - a_m}{2}.$$

If $\delta = 0$, then $\mathrm{sign}(\delta)$ can arbitrarily be set to either 1 or -1.

The Wilkinson shift achieves **cubic convergence** in general, and quadratic convergence in the worst case. **In exact arithmetic** the QR-algorithm with the Wilkinson shift always converges.

For the example that "broke" the Rayleigh shift is $\mu_{\mathrm{W}} = \pm 1$, and we converge in one step.

A Comment on $\mathrm{sign}(x)$

Whereas the mathematical sign/signum function is

$$\forall x \in \mathbb{R} : \quad \mathrm{sign}(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

The "computational science" sign/signum function is usually (always?)

$$\forall x_{\mathbb{F}} \in \mathbb{F}_{64,128,\ldots} : \quad \mathrm{sign}(x_{\mathbb{F}}) = (-1)^s$$

where $s \in \{0, 1\}$ is the value of the sign-bit of the floating-point value $x_{\mathbb{F}}$.

This FORCES a choice ± 1 for all values of $x_{\mathbb{F}}$

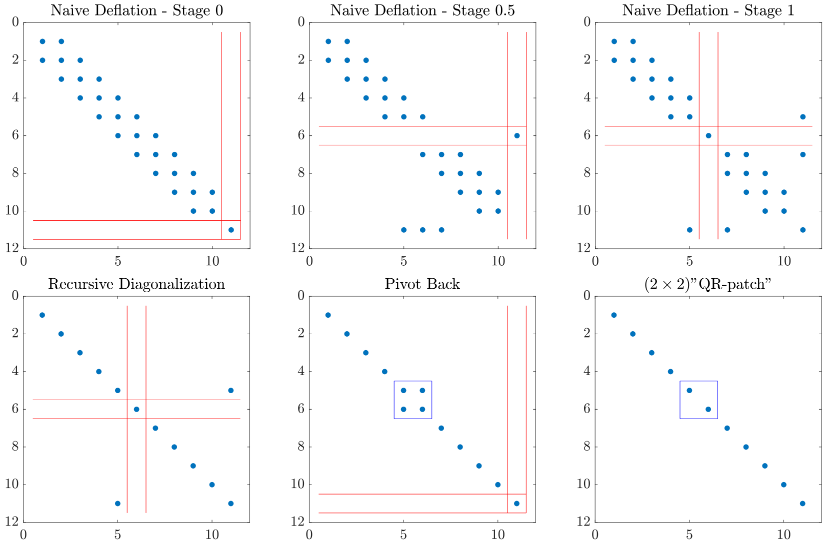
Cleaning Up...

We now have all but one of the main components of the QR-algorithm: Once we have found λ_m to desired accuracy, we should **deflate** the problem in an appropriate way in order to identify the remaining eigenvalues.

A full implementation, including a discussion of deflation strategies, may be a good project idea... for a dark and stormy night.

We conclude the discussion on the QR-algorithm with some comments regarding stability and accuracy.

The Modified QR-Algorithm: Naive Deflation, and Beyond



Stability and Accuracy

Since the QR-algorithm is built using orthogonal transformations, we expect the algorithm to be backward stable; with $\tilde{\Lambda}$ being the computed diagonalization, and $\tilde{Q}$ being the exactly orthogonal matrix assembled from all the numerically computed Householder reflections used along the way, the following result holds

Theorem

Let a real, symmetric, tridiagonal matrix $A \in \mathbb{R}^{m \times m}$ be diagonalized by the QR-algorithm with shifts and deflation in a floating point environment satisfying the usual axioms, then we have

$$\tilde{Q}\tilde{\Lambda}\tilde{Q}^* = A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\varepsilon_{mach}),$$

for some $\delta A \in \mathbb{C}^{m \times m}$.

It follows that $\dfrac{|\tilde{\lambda}_j - \lambda_j|}{\|A\|} = \mathcal{O}(\varepsilon_{\text{mach}})$.

Next... Computing the SVD

Building for the Final

Spring 2024

Final-Fragments:

- **QR-Algorithm with Wilkinson Shifts:** You are going to need an implementation of the QR-algorithm with Wilkinson shifts [LECTURE#21]. You are free to use a library / built-in call to compute the QR-factorization in the QR-algorithm.
- **Inverse Iteration:** You are going to need an implementation of the inverse iteration [LECTURE#19]. You are free to use a library / built-in call to solve the linear system in the inverse iteration.
- **Rayleigh Quotient:** You are going to need an implementaton of the Rayleigh quotient [LECTURE#19] (not to be confused with the Rayleigh quotient iteration).

Now is a good time to start building and testing...

Homework AI-Policy Spring 2024

AI-era Policies — SPRING 2024

AI-3 Documented: *Students can use AI in any manner for this assessment or deliverable, but they must provide appropriate documentation for all AI use.*

This applies to ALL MATH-543 WORK during the SPRING 2024 semester.

The goal is to leverage existing tools and resources to generate HIGH QUALITY SOLUTIONS to all assessments.

You MUST document what tools you use and HOW they were used (including prompts); AND how results were VALIDATED.

BE PREPARED to DISCUSS homework solutions and AI-strategies. **Participation in the in-class discussions will be an essential component of the grade for each assessment.**