AMS526: Numerical Analysis I (Numerical Linear Algebra)

Lecture 20: QR Algorithm with Shifts

Xiangmin Jiao

SUNY Stony Brook

November 20, 2008

Simultaneous Iteration $\Longleftrightarrow$ QR Algorithm

```
Algorithm: Simultaneous Iteration
    Pick $\hat{Q}^{(0)} \in \mathbb{R}^{m\times n}$
    for $k = 1, 2, \ldots$
        $Z = A\hat{Q}^{(k-1)}$
        $\hat{Q}^{(k)}\hat{R}^{(k)} = Z$
```

```
Algorithm: "Pure" QR Algorithm
    $A^{(0)} = A$
    for $k = 1, 2, \ldots$
        $Q^{(k)}R^{(k)} = A^{(k-1)}$
        $A^{(k)} = R^{(k)}Q^{(k)}$
```

- QR algorithm is equivalent to simultaneous iteration with $\hat{Q}^{(0)} = I$
- Replace $\hat{R}^{(k)}$ by $R^{(k)}$ and $\hat{Q}^{(k)}$ by $\underline{Q}^{(k)}$, and introduce new statement $A^{(k)} = \left(\underline{Q}^{(k)}\right)^T A\underline{Q}^{(k)}$ in simultaneous iteration

```
Simultaneous iteration
    $\underline{Q}^{(0)} = I$
    $Z = A\underline{Q}^{(k-1)}$
    $\underline{Q}^{(k)}R^{(k)} = Z$
    $A^{(k)} = \left(\underline{Q}^{(k)}\right)^T A\underline{Q}^{(k)}$
```

```
QR algorithm
    $A^{(0)} = A$
    $Q^{(k)}R^{(k)} = A^{(k-1)}$
    $A^{(k)} = R^{(k)}Q^{(k)}$
    $\underline{Q}^{(k)} = Q^{(1)}Q^{(2)}\ldots Q^{(k)}$
```

Simultaneous Iteration $\Longleftrightarrow$ QR Algorithm

- $\underline{\boldsymbol{Q}}^{(k)} = \boldsymbol{Q}^{(1)}\boldsymbol{Q}^{(2)}\cdots\boldsymbol{Q}^{(k)}$. Let $\underline{\boldsymbol{R}}^{(k)} = \boldsymbol{R}^{(k)}\boldsymbol{R}^{(k-1)}\cdots\boldsymbol{R}^{(1)}$
- Both schemes generate QR factorization $\boldsymbol{A}^{k} = \underline{\boldsymbol{Q}}^{(k)}\underline{\boldsymbol{R}}^{(k)}$ and projection $\boldsymbol{A}^{(k)} = \left(\underline{\boldsymbol{Q}}^{(k)}\right)^{T}\boldsymbol{A}\underline{\boldsymbol{Q}}^{(k)}$

Proof by induction. For $k = 0$ it is trivial for both algorithms.
For $k \geq 1$ with simultaneous iteration, $\boldsymbol{A}^{(k)}$ is given by definition, and

$$\boldsymbol{A}^{k} = \boldsymbol{A}\underline{\boldsymbol{Q}}^{(k-1)}\underline{\boldsymbol{R}}^{(k-1)} = \underline{\boldsymbol{Q}}^{(k)}\boldsymbol{R}^{(k)}\underline{\boldsymbol{R}}^{(k-1)} = \underline{\boldsymbol{Q}}^{(k)}\underline{\boldsymbol{R}}^{(k)}$$

For $k \geq 1$ with QR algorithm,

$$\boldsymbol{A}^{k} = \boldsymbol{A}\underline{\boldsymbol{Q}}^{(k-1)}\underline{\boldsymbol{R}}^{(k-1)} = \underline{\boldsymbol{Q}}^{(k-1)}\boldsymbol{A}^{(k-1)}\underline{\boldsymbol{R}}^{(k-1)} = \underline{\boldsymbol{Q}}^{(k)}\underline{\boldsymbol{R}}^{(k)}$$

and

$$\boldsymbol{A}^{(k)} = \left(\boldsymbol{Q}^{(k)}\right)^{T}\boldsymbol{A}^{(k-1)}\boldsymbol{Q}^{(k)} = \left(\underline{\boldsymbol{Q}}^{(k)}\right)^{T}\boldsymbol{A}\underline{\boldsymbol{Q}}^{(k)}$$

- Therefore, diagonal entries of $\boldsymbol{A}^{(k)}$ are Rayleigh quotients of column vectors of $\underline{\boldsymbol{Q}}^{(k)}$

Simultaneous Inverse Iteration $\Longleftrightarrow$ QR Algorithm

- Similar to inverse iteration, QR algorithm can be sped-up by introducing shifts at each step
- QR algorithm is equivalent to *simultaneous inverse iteration*, applied to "flipped" identity matrix $\boldsymbol{P}$

$$\boldsymbol{P} = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & \vdots & & \\ 1 & & & \end{bmatrix}$$

```
Simultaneous inverse iteration
  $\hat{Q}^{(0)} = P$
  for $k = 1, 2, \ldots$
      $Z = A^{-1}\hat{Q}^{(k-1)}$
      $\hat{Q}^{(k)}\hat{R}^{(k)} = Z$
```

```
"Pure" QR Algorithm
  $A^{(0)} = A$
  for $k = 1, 2, \ldots$
      $Q^{(k)}R^{(k)} = A^{(k-1)}$
      $A^{(k)} = R^{(k)}Q^{(k)}$
```

Simultaneous Inverse Iteration $\Longleftrightarrow$QR Algorithm

- We establish the equivalence as follows
- $\boldsymbol{A}^k = \underline{\boldsymbol{Q}}^{(k)}\underline{\boldsymbol{R}}^{(k)}$, where $\underline{\boldsymbol{Q}}^{(k)} = \prod_{j=1}^{k} \boldsymbol{Q}^{(j)}$, $\underline{\boldsymbol{R}}^{(k)} = \prod_{j=k}^{1} \boldsymbol{R}^{(j)}$
- Invert $\boldsymbol{A}^k$, we have $\boldsymbol{A}^{-k} = \left(\underline{\boldsymbol{R}}^{(k)}\right)^{-1}\left(\underline{\boldsymbol{Q}}^{(k)}\right)^{T} = \underline{\boldsymbol{Q}}^{(k)}\left(\underline{\boldsymbol{R}}^{(k)}\right)^{-T}$, where second equality uses the fact that $\boldsymbol{A}^{-k}$ is symmetric
- Use "flipped" permutation matrix $\boldsymbol{P}$ and write that last expression as

  $$\boldsymbol{A}^{-k}\boldsymbol{P} = \left[\underline{\boldsymbol{Q}}^{(k)}\boldsymbol{P}\right]\left[\boldsymbol{P}\left(\underline{\boldsymbol{R}}^{(k)}\right)^{-T}\boldsymbol{P}\right],$$

  which is the QR factorization of $\boldsymbol{A}^{-k}\boldsymbol{P}$
- Therefore, simultaneous inverse iteration applied to $\hat{\boldsymbol{Q}}^{(0)} = \boldsymbol{P}$ is "equivalent" to QR algorithm, in the sense that the former produces

  $$\hat{\boldsymbol{Q}}^{(k)} = \underline{\boldsymbol{Q}}^{(k)}\boldsymbol{P} \text{ and } \hat{\boldsymbol{R}}^{(k)}\hat{\boldsymbol{R}}^{(k-1)}\cdots\hat{\boldsymbol{R}}^{(1)} = \boldsymbol{P}\left(\underline{\boldsymbol{R}}^{(k)}\right)^{-T}\boldsymbol{P}.$$

QR Algorithm with Shifts

- Introduce shifts $\mu^{(k)}$ to accelerate convergence

```
Algorithm: "Practical" QR Algorithm
  $\left(\mathbf{Q}^{(0)}\right)^T \mathbf{A}^{(0)} \mathbf{Q}^{(0)} = \mathbf{A}$
  for $k = 1, 2, \dots$
          Pick a shift $\mu^{(k)}$
          $\mathbf{Q}^{(k)} \mathbf{R}^{(k)} = \mathbf{A}^{(k-1)} - \mu^{(k)} \mathbf{I}$
          $\mathbf{A}^{(k)} = \mathbf{R}^{(k)} \mathbf{Q}^{(k)} + \mu^{(k)} \mathbf{I}$
          If any off-diagonal element $a_{j,j+1}^{(k)}$ is sufficiently close to zero
                set $a_{j,j+1} = a_{j+1,j} = 0$ to obtain $\begin{bmatrix} \mathbf{A}_1 & \\ & \mathbf{A}_2 \end{bmatrix} = \mathbf{A}^{(k)}$ and
  apply QR algorithm to $\mathbf{A}_1$ and $\mathbf{A}_2$
```

- We then get $\mathbf{A}^{(k)} = \left(\mathbf{Q}^{(k)}\right)^T \mathbf{A}^{(k-1)} \mathbf{Q}^{(k)} = \left(\underline{\mathbf{Q}}^{(k)}\right)^T \mathbf{A} \underline{\mathbf{Q}}^{(k)}$
- Furthermore, $\left(\mathbf{A} - \mu^{(k)} \mathbf{I}\right) \left(\mathbf{A} - \mu^{(k-1)} \mathbf{I}\right) \cdots \left(\mathbf{A} - \mu^{(1)} \mathbf{I}\right) = \underline{\mathbf{Q}}^{(k)} \underline{\mathbf{R}}^{(k)}$

Choosing $\mu^{(k)}$: Rayleigh Quotient Shift

- Natural choice of $\mu^{(k)}$ is Rayleigh quotient for last column of $\underline{\boldsymbol{Q}}^{(k)}$

$$\mu^{(k)} = r(\boldsymbol{q}_m^{(k)}) = \left(\boldsymbol{q}_m^{(k)}\right)^T \boldsymbol{A}\boldsymbol{q}_m^{(k)}$$

- As in Rayleigh quotient iteration, last column $\boldsymbol{q}_m^{(k)}$ converges cubically
- This Rayleigh quotient appears as (m,m) entry of $\boldsymbol{A}^{(k)}$ since $\boldsymbol{A}^{(k)} = \left(\underline{\boldsymbol{Q}}^{(k)}\right)^T \boldsymbol{A}\underline{\boldsymbol{Q}}^{(k)}$
- The Rayleigh quotient shift corresponds to setting $\mu^{(k)} = A_{mm}^{(k)}$

Choosing $\mu^{(k)}$: Wilkinson Shift

- QR algorithm with Rayleigh quotient shift might fail, e.g., for $\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, which has two eigenvalues with same magnitude but opposite signs
- Wilkinson breaks symmetry by considering lower-rightmost 2×2 submatrix of $\boldsymbol{A}^{(k)}$: $\boldsymbol{B} = \begin{bmatrix} a_{m-1} & b_{m-1} \\ b_{m-1} & a_m \end{bmatrix}$
- Choose eigenvalue of $\boldsymbol{B}$ closer to a_m, with arbitrary tie-breaking:

  $$\mu = a_m - \text{sign}(\delta) b_{m-1}^2 / \left(|\delta| + \sqrt{\delta^2 + b_{m-1}^2} \right)$$

  where $\delta = (a_{m-1} - a_m)/2$; if $\delta = 0$, set $\mathrm{sign}(\delta)$ to 1 (or -1)
- Always converge with this shift, quadratically in worst case, cubically in general

Stability and Accuracy

Theorem

QR algorithm is backward stable

$$\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{Q}} = \boldsymbol{A} + \delta\boldsymbol{A}, \quad \frac{\|\delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|} = O(\epsilon_{machine})$$

where $\tilde{\boldsymbol{\Lambda}}$ is computed $\boldsymbol{\Lambda}$ and $\tilde{\boldsymbol{Q}}$ is exactly orthogonal matrix

- Its combination with Hessenberg reduction is also backward stable
- Furthermore, eigenvalues are always well conditioned for **normal** matrices: it can be show that $|\tilde{\lambda}_j - \lambda_j| \leq \|\delta\boldsymbol{A}\|_2$, and therefore,

  $$\frac{|\tilde{\lambda}_j - \lambda_j|}{\|\boldsymbol{A}\|} = O(\epsilon_{\text{machine}})$$

  where $\tilde{\lambda}_j$ are the computed eigenvalues
- However, sensitivity of eigenvectors depends on distances between adjacent eigenvalues, so error in eigenvectors may be arbitrarily large