The QR algorithm

The most common method for solving small (dense) eigenvalue problems. The basic algorithm:

**QR without shifts**

1. Until Convergence Do:
2. Compute the QR factorization $A = QR$
3. Set $A := RQ$
4. EndDo

“Until Convergence” means “Until $A$ becomes close enough to an upper triangular matrix”
Note: $A_{new} = RQ = Q^H(QR)Q = Q^H AQ$

$A_{new}$ is similar to $A$ throughout the algorithm.

Convergence analysis complicated – but insight: we are implicitly doing a QR factorization of $A^k$:

3 steps:

$A_0 = Q_0R_0$
$A_1 = Q_1R_1$
$A_2 = Q_2R_2$

$A_3 = R_2Q_2$

Then:

$[Q_0Q_1Q_2][R_2R_1R_0] = Q_0Q_1A_2R_1R_0$
$= Q_0Q_1R_1Q_1R_1R_0$
$= Q_0R_0Q_0R_0Q_0R_0$
$= A^3$

$[Q_0Q_1Q_2][R_2R_1R_0] == QR$ factorization of $A^3$
Above basic algorithm is never used in practice. Two variations:

1. Use shift of origin and
2. Start by transforming $A$ into an Hessenberg matrix
Practical QR algorithms: Shifts of origin

Observation: (from theory): Last row converges fastest. Convergence is dictated by $\frac{|\lambda_n|}{|\lambda_{n-1}|}$

We will now consider only the real symmetric case.

- Eigenvalues are real.
- $A^{(k)}$ remains symmetric throughout process.
- As $k$ goes to infinity the last column and row (except $a_{nn}^{(k)}$) converge to zero quickly.,
- and $a_{nn}^{(k)}$ converges to lowest eigenvalue.
\[ A^{(k)} = \begin{pmatrix}
\ddots & \ddots & \ddots & \ddots & \ddots \\
\ddots & \ddots & \ddots & \ddots & \ddots \\
\ddots & \ddots & \ddots & \ddots & \ddots \\
\ddots & \ddots & \ddots & \ddots & \ddots \\
a & a & a & a & a
\end{pmatrix} \]

Idea: Apply QR algorithm to \( A^{(k)} - \mu I \) with \( \mu = a^{(k)}_{nn} \). Note: eigenvalues of \( A^{(k)} - \mu I \) are shifted by \( \mu \), and eigenvectors are the same.
QR with shifts

1. Until row $a_{in}, 1 \leq i < n$ converges to zero DO:
2. Obtain next shift (e.g. $\mu = a_{nn}$)
3. $A - \mu I = QR$
4. Set $A := RQ + \mu I$
5. EndDo

» Convergence (of last row) is cubic at the limit! [for symmetric case]
Result of algorithm:

\[ A^{(k)} = \begin{pmatrix} . & . & . & . & . & 0 \\ . & . & . & . & . & 0 \\ . & . & . & . & . & 0 \\ . & . & . & . & . & 0 \\ . & . & . & . & . & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_n \end{pmatrix} \]

Next step: deflate, i.e., apply above algorithm to \((n - 1) \times (n - 1)\) upper triangular matrix.
Practical algorithm: Use the Hessenberg Form

Recall: Upper Hessenberg matrix is such that

\[ a_{ij} = 0 \text{ for } j < i - 1 \]

Observation: The QR algorithm preserves Hessenberg form (tridiagonal form in symmetric case). Results in substantial savings.

Transformation to Hessenberg form

- Want \( H_1 A H_1^T = H_1 A H_1 \) to have the form shown on the right

- Consider the first step only on a 6 \( \times \) 6 matrix
Choose a $w$ in $H_1 = I - 2ww^T$ to make the first column have zeros from position 3 to $n$. So $w_1 = 0$.

Apply to left: $B = H_1A$

Apply to right: $A_1 = BH_1$.

**Main observation:** the Householder matrix $H_1$ which transforms the column $A(2:n, 1)$ into $e_1$ works only on rows 2 to $n$. When applying the transpose $H_1$ to the right of $B = H_1A$, we observe that only columns 2 to $n$ will be altered. So the first column will retain the desired pattern (zeros below row 2).

Algorithm continues the same way for columns 2, ..., $n - 2$. 

---

13-9

TB: 28-30; AB: 1.3.3, 3.2.3, 3.4.2, 3.5, 3.6.2; GvL 8.1-8.2.3 – Eigen2New
QR for Hessenberg matrices

- Need the “Implicit Q theorem”

Suppose that $Q^T AQ$ is an unreduced upper Hessenberg matrix. Then columns 2 to $n$ of $Q$ are determined uniquely (up to signs) by the first column of $Q$.

- In other words if $V^T AV = G$ and $Q^T AQ = H$ are both Hessenberg and $V(:, 1) = Q(:, 1)$ then $V(:, i) = \pm Q(:, i)$ for $i = 2 : n$.

**Implication:** To compute $A_{i+1} = Q_i^T AQ_i$ we can:

- Compute 1st column of $Q_i$ [= scalar $\times A(:, 1)$]
- Choose other columns so $Q_i = $ unitary, and $A_{i+1} = $ Hessenberg.
Wll do this with Givens rotations

Example: With \( n = 6 \):

\[
A = \begin{pmatrix}
* & * & * & * & * \\
* & * & * & * & * \\
0 & * & * & * & * \\
0 & 0 & * & * & * \\
0 & 0 & 0 & * & * \\
\end{pmatrix}
\]

1. Choose \( G_1 = G(1, 2, \theta_1) \) so that \( (G_1^T A_0)_{21} = 0 \)

\[
A_1 = G_1^T A G_1 = \begin{pmatrix}
* & * & * & * & * \\
* & * & * & * & * \\
+ & * & * & * & * \\
0 & 0 & * & * & * \\
0 & 0 & 0 & * & * \\
\end{pmatrix}
\]

2. Choose \( G_2 = G(2, 3, \theta_2) \) so that \( (G_2^T A_1)_{31} = 0 \)
\[ A_2 = G_2^T A_1 G_2 = \begin{pmatrix}
* & * & * & * & * \\
* & * & * & * & * \\
0 & * & * & * & * \\
0 & 0 & + & * & * \\
0 & 0 & 0 & * & *
\end{pmatrix} \]

3. Choose \( G_3 = G(3, 4, \theta_3) \) so that \( (G_3^T A_2)_{42} = 0 \)

\[ A_3 = G_3^T A_2 G_3 = \begin{pmatrix}
* & * & * & * & * \\
* & * & * & * & * \\
0 & * & * & * & * \\
0 & 0 & 0 & * & * \\
0 & 0 & 0 & + & *
\end{pmatrix} \]

4. Choose \( G_4 = G(4, 5, \theta_4) \) so that \( (G_4^T A_3)_{53} = 0 \)
\[
A_4 = G_4^T A_3 G_4 = \begin{pmatrix}
* & * & * & * & * \\
* & * & * & * & * \\
0 & * & * & * & * \\
0 & 0 & * & * & * \\
0 & 0 & 0 & * & * 
\end{pmatrix}
\]

Process known as “Bulge chasing”

Similar idea for the symmetric (tridiagonal) case
The symmetric eigenvalue problem: Basic facts

Consider the Schur form of a real symmetric matrix $A$:

$$A = QRQ^H$$

Since $A^H = A$ then $R = R^H$.

Eigenvalues of $A$ are real

and

There is an orthonormal basis of eigenvectors of $A$

In addition, $Q$ can be taken to be real when $A$ is real.

$$(A - \lambda I)(u + iv) = 0 \rightarrow (A - \lambda I)u = 0 \& (A - \lambda I)v = 0$$

Can select eigenvector to be either $u$ or $v$
The min-max theorem (Courant-Fischer)

Label eigenvalues decreasingly:

\[ \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \]

The eigenvalues of a Hermitian matrix \( A \) are characterized by the relation

\[ \lambda_k = \max_{S, \dim(S) = k} \min_{x \in S, x \neq 0} \frac{(Ax, x)}{(x, x)} \]

**Proof:** Preparation: Since \( A \) is symmetric real (or Hermitian complex) there is an orthonormal basis of eigenvectors \( u_1, u_2, \cdots, u_n \). Express any vector \( x \) in this basis as \( x = \sum_{i=1}^{n} \alpha_i u_i \). Then:

\[(Ax, x)/(x, x) = [\sum \lambda_i |\alpha_i|^2]/[\sum |\alpha_i|^2].\]

(a) Let \( S \) be any subspace of dimension \( k \) and let \( W = \text{span}\{u_k, u_{k+1}, \cdots, u_n\} \). A dimension argument (used before) shows that \( S \cap W \neq \{0\} \). So there is a
non-zero \( x_w \) in \( S \cap W \). Express this \( x_w \) in the eigenbasis as \( x_w = \sum_{i=k}^{n} \alpha_i u_i \).

Then since \( \lambda_i \leq \lambda_k \) for \( i \geq k \) we have:

\[
\frac{(Ax_w, x_w)}{(x_w, x_w)} = \frac{\sum_{i=k}^{n} \lambda_i |\alpha_i|^2}{\sum_{i=k}^{n} |\alpha_i|^2} \leq \lambda_k
\]

So for any subspace \( S \) of dim. \( k \) we have \( \min_{x \in S, x \neq 0} (Ax, x)/(x, x) \leq \lambda_k \).

(b) We now take \( S_* = \text{span}\{u_1, u_2, \ldots, u_k\} \). Since \( \lambda_i \geq \lambda_k \) for \( i \leq k \), for this particular subspace we have:

\[
\min_{x \in S_*, x \neq 0} \frac{(Ax, x)}{(x, x)} = \min_{x \in S_*, x \neq 0} \frac{\sum_{i=1}^{k} \lambda_i |\alpha_i|^2}{\sum_{i=k}^{n} |\alpha_i|^2} = \lambda_k.
\]

(c) The results of (a) and (b) imply that the max over all subspaces \( S \) of dim. \( k \) of \( \min_{x \in S, x \neq 0} (Ax, x)/(x, x) \) is equal to \( \lambda_k \). 

\[\square\]
Consequences:

\[
\lambda_1 = \max_{x \neq 0} \frac{(Ax, x)}{(x, x)} \quad \lambda_n = \min_{x \neq 0} \frac{(Ax, x)}{(x, x)}
\]

Actually 4 versions of the same theorem. 2nd version:

\[
\lambda_k = \min_{S, \dim(S) = n-k+1} \max_{x \in S, x \neq 0} \frac{(Ax, x)}{(x, x)}
\]

Other 2 versions come from ordering eigenvalues increasingly instead of decreasingly.

Write down all 4 versions of the theorem

Use the min-max theorem to show that \( \|A\|_2 = \sigma_1(A) \) - the largest singular value of \( A \).
Interlacing Theorem: Denote the $k \times k$ principal submatrix of $A$ as $A_k$, with eigenvalues $\{\lambda_i^{[k]}\}_{i=1}^k$. Then

$$\lambda_1^{[k]} \geq \lambda_1^{[k-1]} \geq \lambda_2^{[k]} \geq \lambda_2^{[k-1]} \geq \cdots \lambda_{k-1}^{[k]} \geq \lambda_k^{[k]}$$

**Example:** $\lambda_i$’s = eigenvalues of $A$, $\mu_i$’s = eigenvalues of $A_{n-1}$:

Many uses.

For example: interlacing theorem for roots of orthogonal polynomials.
The Law of inertia

Inertia of a matrix \( [m, z, p] \) with \( m = \) number of \(< 0\) eigenvalues, \( z = \) number of zero eigenvalues, and \( p = \) number of \( > 0\) eigenvalues.

\[ \mathbf{X} \in \mathbb{R}^{n \times n} \text{ is nonsingular, then} \mathbf{A} \text{ and} \mathbf{X}^T \mathbf{A} \mathbf{X} \text{ have the same inertia.} \]

Suppose that \( \mathbf{A} = \mathbf{LDL}^T \) where \( \mathbf{L} \) is unit lower triangular, and \( \mathbf{D} \) diagonal. How many negative eigenvalues does \( \mathbf{A} \) have?

Assume that \( \mathbf{A} \) is tridiagonal. How many operations are required to determine the number of negative eigenvalues of \( \mathbf{A} \)?
Devise an algorithm based on the inertia theorem to compute the \(i\)-th eigenvalue of a tridiagonal matrix.

What is the inertia of the matrix

\[
\begin{pmatrix}
I & F \\
F^T & 0
\end{pmatrix}
\]

where \(F\) is \(m \times n\), with \(n < m\), and of full rank?

[Hint: use a block LU factorization]
Bisection algorithm for tridiagonal matrices:

- Goal: to compute \( i \)-th eigenvalue of \( A \) (tridiagonal)

- Get interval \([a, b]\) containing spectrum [Gershgorin]: \( a \leq \lambda_n \leq \cdots \leq \lambda_1 \leq b \)

- Let \( \sigma = (a + b)/2 \) = middle of interval

- Calculate \( p = \) number of positive eigenvalues of \( A - \sigma I \)

  - If \( p \geq i \) then \( \lambda_i \in (\sigma, b) \) → set \( a := \sigma \)
  
  ![Diagram of eigenvalues and interval]

  - Else then \( \lambda_i \in [a, \sigma] \) → set \( b := \sigma \)

- Repeat until \( b - a \) is small enough.
The QR algorithm for symmetric matrices

- Most important method used: reduce to tridiagonal form and apply the QR algorithm with shifts.
- Householder transformation to Hessenberg form yields a tridiagonal matrix because

\[ HAH^T = A_1 \]

is symmetric and also of Hessenberg form, it is tridiagonal symmetric.

Tridiagonal form preserved by QR similarity transformation
Practical method

How to implement the QR algorithm with shifts?

It is best to use Givens rotations – can do a shifted QR step without explicitly shifting the matrix.

Two most popular shifts:

\[ s = a_{nn} \text{ and } s = \text{smallest e.v. of } A(n-1:n, n-1:n) \]
Main idea: Rotation matrices of the form

\[ J(p, q, \theta) = \begin{pmatrix}
1 & \cdots & 0 & \cdots & 0 & 0 \\
\vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & c & \cdots & s & \cdots & 0 \\
\vdots & \cdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & \cdots & -s & \cdots & c & \cdots & 0 \\
\vdots & \cdots & \vdots & \cdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & \cdots & \cdots & \cdots & 1
\end{pmatrix} \]

\( J(p, q, \theta)^T A J(p, q, \theta) \) has a zero in position \((p, q)\) (and also \((q, p)\))

**Frobenius norm of matrix is preserved – but diagonal elements become larger**

\( c = \cos \theta \) and \( s = \sin \theta \) are so that \( J(p, q, \theta)^T A J(p, q, \theta) \) has a zero in position \((p, q)\) (and also \((q, p)\)).
Let $B = J^T AJ$ (where $J \equiv J_{p,q,\theta}$).

Look at $2 \times 2$ matrix $B([p, q], [p, q])$ (matlab notation)

Keep in mind that $a_{pq} = a_{qp}$ and $b_{pq} = b_{qp}$

\[
\begin{pmatrix}
    b_{pp} & b_{pq} \\
    b_{qp} & b_{qq}
\end{pmatrix} = \begin{pmatrix}
    c & -s \\
    s & c
\end{pmatrix} \begin{pmatrix}
    a_{pp} & a_{pq} \\
    a_{qp} & a_{qq}
\end{pmatrix} \begin{pmatrix}
    c & s \\
    -s & c
\end{pmatrix}
\]

\[
= \begin{pmatrix}
    c & -s \\
    s & c
\end{pmatrix} \begin{pmatrix}
    ca_{pp} - sa_{pq} & sa_{pp} + ca_{pq} \\
    ca_{qp} - sa_{qq} & sa_{pq} + ca_{qq}
\end{pmatrix}
\]

\[
= \begin{pmatrix}
    c^2a_{pp} + s^2a_{qq} - 2sc a_{pq} & (c^2 - s^2)a_{pq} - sc(a_{qq} - a_{pp}) \\
    sc(a_{qq} - a_{pp}) & c^2a_{qq} + s^2a_{pp} + 2sc a_{pq}
\end{pmatrix} *\]

Want: \[(c^2 - s^2)a_{pq} - sc(a_{qq} - a_{pp}) = 0\]
\[
\frac{c^2 - s^2}{2sc} = \frac{a_{qq} - a_{pp}}{2a_{pq}} \equiv \tau
\]

Letting \( t = s/c \) (= \tan \theta) \quad \rightarrow \quad \text{quad. equation}

\[
t^2 + 2\tau t - 1 = 0
\]

\[
t = -\tau \pm \sqrt{1 + \tau^2} = \frac{1}{\tau \pm \sqrt{1 + \tau^2}}
\]

Select sign to get a smaller \( t \) so \( \theta \leq \pi/4 \).

Then:
\[
c = \frac{1}{\sqrt{1 + t^2}}; \quad s = c \times t
\]

Implemented in matlab script \texttt{jacrot(A,p,q)} – See HW6.
Define: \[ A_O = A - \text{Diag}(A) \equiv A \text{ ‘with its diagonal entries replaced by zeros’} \]

Observations: (1) Unitary transformations preserve \( \| . \|_F \). (2) Only changes are in rows and columns \( p \) and \( q \).

Let \( B = J^T A J \) (where \( J \equiv J_{p,q,\theta} \)). Then,

\[
a_{pp}^2 + a_{qq}^2 + 2a_{pq}^2 = b_{pp}^2 + b_{qq}^2 + 2b_{pq}^2 = b_{pp}^2 + b_{qq}^2
\]

because \( b_{pq} = 0 \). Then, a little calculation leads to:

\[
\| B_O \|_F^2 = \| B \|_F^2 - \sum b_{ii}^2 = \| A \|_F^2 - \sum b_{ii}^2
\]

\[
= \| A \|_F^2 - \sum a_{ii}^2 + \sum a_{ii}^2 - \sum b_{ii}^2
\]

\[
= \| A_O \|_F^2 + (a_{pp}^2 + a_{qq}^2 - b_{pp}^2 - b_{qq}^2)
\]

\[
= \| A_O \|_F^2 - 2a_{pq}^2
\]
\[ \|A_O\|_F \] will decrease from one step to the next.

Let \[ \|A_O\|_I = \max_{i \neq j} |a_{ij}|. \] Show that
\[ \|A_O\|_F \leq \sqrt{n(n - 1)} \|A_O\|_I \]

Use this to show convergence in the case when largest entry is zeroed at each step.