Numerical Analysis Hilary Term 2025
Lectures 7–8: Computing eigenvalues: The Symmetric QR Algorithm

Last lecture, we began studying iterative algorithms for the eigenvalue problem $Ax = \lambda x$ with $A \in \mathbb{R}^{n\times n}$. The first method we analyzed was the **power method** in Algorithm 1. We saw that if A is diagonalizable and the eigenvalues of A satisfy $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \cdots \geq |\lambda_n|$, then x_k "converges" to a dominant eigenvector (corresponding to λ_1) at a rate $\mathcal{O}(|\lambda_1/\lambda_2|)$. Recall that we can also apply the power method to A^{-1} (called the **inverse power method**), in which case x_k converges an eigenvector corresponding to λ_n at a rate $\mathcal{O}(|\lambda_n/\lambda_{n-1}|)$.

Algorithm 1 Power Method

Require: $A \in \mathbb{R}^{n\times n}$ and $x_0 \in \mathbb{R}^n$
  for $k = 1, 2, \ldots$ **do**
    $y_k = Ax_k$
    $x_{k+1} = y_k/\|y_k\|_2$
  end for

Although the power method for eigenvalue computation is the basis for effective methods, the current standard method is the **QR Algorithm** which was invented by John Francis in London in 1959/60. As we shall see, the mechanics of QR algorithm is very much related to the power method.

Simplifying assumptions: For the remainder of these notes, we will assume that $A \in \mathbb{R}^{n\times n}$ is symmetric and that the eigenvalues of A satisfy $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > 0$. The algorithm and the variants presented here can be extended to nonsymmetric matrices, but that is outside the scope of this course.

The QR algorithm

We now describe the QR algorithm, a magical algorithm that can solve eigenvalue problems $Ax = \lambda x$ and finds all eigenvalues (and eigenvectors). The basic **QR algorithm** is in Algorithm 2:

Algorithm 2 QR algorithm for symmetric matrix

Require: $A \in \mathbb{R}^{n\times n}$ symmetric.
  Set $A_1 := A$
  for $k = 1, 2, \ldots$ **do**
    Form the QR factorization $A_k = Q_k R_k$
    Set $A_{k+1} := R_k Q_k$
  end for

As you can see, the QR algorithm is remarkably simple: at each iteration, we compute one QR factorization and then reverse the order of the factors. We note that Algorithm 2 is well-defined even if A is not symmetric.

We now turn to the analysis of the QR algorithm. We first collect some preliminary properties.

Proposition. The iterates $A_1, A_2, \ldots, A_k, \ldots$ are symmetric and all similar to A. Thus, all iterates have the same eigenvalues.

Proof. The proposition is clearly true for $k = 1$, so suppose that it holds for a generic $k \geq 1$. Using that Q_k is orthogonal, we have

$$A_{k+1} = R_k Q_k = (Q_k^\top Q_k) R_k Q_k = Q_k^\top (Q_k R_k) Q_k = Q_k^\top A_k Q_k, \tag{1}$$

and so A_{k+1} is symmetric since we have assumed that A_k is symmetric. Moreover, $A_{k+1} = Q_k^\top A_k Q_k = Q_k^{-1} A_k Q_k$, and so A_{k+1} is similar to A_k and thus similar to A. The result now follows by induction. □

Lemma. Let

$$Q^{(k)} := Q_1 Q_2 \cdots Q_k \quad \text{and} \quad R^{(k)} := R_k R_{k-1} \cdots R_1. \tag{2}$$

Then, there holds

$$A_{k+1} = (Q^{(k)})^\top A Q^{(k)} \tag{3}$$

and

$$A^k = Q^{(k)} R^{(k)} \tag{4}$$

is the QR factorization of A^k (the k-th power of k).

Proof. Repeatedly applying equation eq. (1) gives

$$\begin{aligned} A_{k+1} = Q_k^\top A_k Q_k = Q_k^\top Q_{k-1}^\top A_{k-1} Q_{k-1} Q_k = \cdots = Q_k^\top Q_{k-1}^\top \cdots Q_1^\top A_1 Q_1 \cdots Q_{k-1} Q_k \\ = (Q^{(k)})^\top A Q^{(k)}. \end{aligned}$$

We now turn to eq. (4). We proceed by induction. The case $k = 1$ follows by definition, so suppose that $A^{k-1} = Q^{(k-1)} R^{(k-1)}$ for some $k \geq 2$. Then, there holds

$$A_k = R_{k-1} Q_{k-1} = (Q^{(k-1)})^T A Q^{(k-1)} = Q_k R_k.$$

Examining the last equality and using that $Q^{(k-1)}$ is orthogonal, we have

$$A Q^{(k-1)} = Q^{(k-1)} Q_k R_k = Q^{(k)} R_k,$$

and so

$$A^k = A A^{k-1} = A Q^{(k-1)} R^{(k-1)} = Q^{(k)} R_k R^{(k-1)} = Q^{(k)} R^{(k)}.$$

The result now follows by induction. □

Using the above result, we can, perhaps surprisingly, connect the QR algorithm to the power method.

Lemma. Let q_1 denote the first column of $Q^{(k)}$ defined in eq. (4) and let $e_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^\top$. Then, either

$$q_1 = \frac{A^k e_1}{\|A^k e_1\|_2} \quad \text{or} \quad q_1 = -\frac{A^k e_1}{\|A^k e_1\|_2}.$$

That is, the first column of $Q^{(k)}$ is the kth-iterate of the power method applied to A with e_1 as the starting vector.

Proof. Thanks to eq. (4), we have

$$A^k e_1 = Q^{(k)} R^{(k)} e_1 = Q^{(k)} \begin{bmatrix} r_{11}^{(k)} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = r_{11}^{(k)} q_1,$$

and so

$$\frac{A^k e_1}{\|A^k e_1\|_2} = \operatorname{sgn}(r_{11}^{(k)}) \frac{q_1}{\|q_1\|} = \pm q_1$$

since $\|q_1\|_2 = 1$. □

Perhaps even more surprisingly, there is also a connection to the inverse power method.

Lemma. Let q_n denote the first column of $Q^{(k)}$ defined in eq. (4) and let $e_n = \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix}^\top$. Then, either

$$q_n = \frac{A^{-k} e_n}{\|A^{-k} e_n\|_2} \quad \text{or} \quad q_n = -\frac{A^{-k} e_n}{\|A^{-k} e_n\|_2}.$$

That is, the final column of $Q^{(k)}$ is the kth-iterate of the inverse power method applied to A (power method applied to A^{-1}) with e_n as the starting vector.

Note: We assumed that all of the eigenvalues are nonzero, so A^{-1} is well-defined.

Proof. The proof is nearly the same as the previous lemma. Thanks to eq. (4), we have

$$A^{-k} = (R^{(k)})^{-1} (Q^{(k)})^\top \quad \text{and} \quad A^{-k} = (A^{-k})^\top = (Q^{(k)})(R^{(k)})^{-\top},$$

and using that used that $(R^{(k)})^{-\top}$ is lower triangular, we obtain

$$A^{-k} e_n = (Q^{(k)})(R^{(k)})^{-\top} e_n = Q^{(k)} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \alpha \end{bmatrix} = \alpha q_n,$$

where α is the (n, n) entry of $(R^{(k)})^{-\top}$. Consequently, there holds

$$\frac{A^{-k} e_n}{\|A^{-k} e_n\|_2} = \operatorname{sgn}(\alpha) \frac{q_n}{\|q_n\|} = \pm q_n$$

since $\|q_n\|_2 = 1$. □

Summarizing the previous two lemmas, the **first** column q_1 of $Q^{(k)}$ is the kth-iterate of the **power method** applied to A and the **final** column q_n of $Q^{(k)}$ is the kth-iterate of the **inverse power method** applied to A with e_n as the starting vector.

Applying the convergence results for the power method, we know that the first column of $Q^{(k)}$ converges to the dominant eigenvector of A at a rate $\mathcal{O}(|\lambda_1/\lambda_2|)$ and the last column of $Q^{(k)}$ converges to the eigenvector corresponding to λ_n at a rate $\mathcal{O}(|\lambda_n/\lambda_{n-1}|)$. We will not prove this, but one can show that as $k \to \infty$, A_k converges to a diagonal matrix whose entries are the eigenvalues of A and columns of $Q^{(k)}$ converge to the corresponding eigenvectors.

Numerical example

To illustrate the performance of the QR algorithm, we look at an example. All of the code is in the `QRalg_demo.m` file. We select a matrix $B \in \mathbb{R}^{100\times 100}$ whose entries are uniformly distributed in $(0, 1)$ and take $A = BB^\top$ so that A is symmetric. In fact, A is positive definite, but this will not matter.

Using built-in functions, we compute the eigenvalues $\lambda_1, \ldots, \lambda_{100}$ and eigenvectors $v_1, \ldots, v_{100}$ of A for comparison. The eigenvalues of A are displayed in Figure 1. In particular, we see that there is a gap between the first two and final two eigenvalues, so we expect λ_1 and λ_{100} to converge rapidly. However, most of the remaining eigenvalues are clustered closer together, so the remaining eigenvalues may not converge as fast.

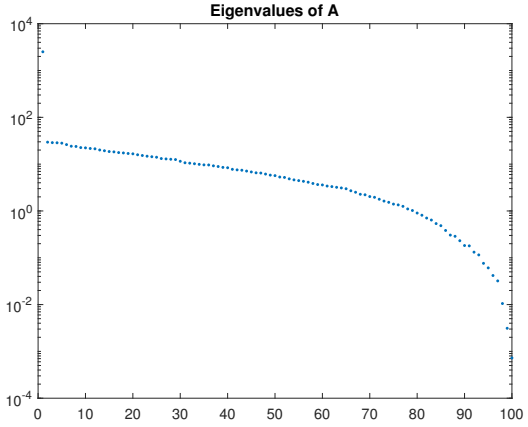

Figure 1: Eigenvalues of randomly generated A

We will also compute the approximate eigenvectors, which are the columns of $Q^{(k)}$. To do this, we slightly modify Algorithm 2 to also store $Q^{(k)}$ at each iteration, as displayed in Algorithm 3. After 300 iterations, we extract the diagonal of A_k as the approximate eigenvalues $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_{100}$, and the columns of $Q^{(k)}$ $q_1, \ldots, q_n$ as the approximate eigenvectors. The results are as follows.

- Eigenvalue error: $\left(\sum_{i=1}^{100} |\lambda_i - \tilde{\lambda}_i|^2\right)^{1/2} \sim 6.9 \times 10^{-2}$.
- Eigenvector errors:
  - First eigenvector $\|v_1 - q_1\|_2 \sim 1.3 \times 10^{-15}$.
  - Final eigenvector $\|v_n - q_n\|_2 \sim 5.5 \times 10^{-12}$.
  - $[\min, \text{median}, \max]$ of $\{\|v_i - q_i\|_2\}_{i=1}^{100}$: $[1.3 \times 10^{-15}, 1.0 \times 10^{-4}, 6.8 \times 10^{-1}]$.

Note that the eigenvector errors are both absolute and relative errors since $\|v_i\|_2 = 1$.
Observation: The convergence can be very slow (in terms of iterations) for some eigenvectors and eigenvalues. To quantify this statement, if we run the final algorithm Algorithm 6 at the end of the notes, then we get the following results.

- Eigenvalue error: $\left(\sum_{i=1}^{100} |\lambda_i - \tilde{\lambda}_i|^2\right)^{1/2} \sim 3.7 \times 10^{-12}$.
- Algorithm 6 does something different for each eigenvalue, so each eigenvalue can require a different number of iterations. The iteration counts are
  - Total iterations: 216.
  - [min, median, max] iterations over all eigenvalue: $[0, 2, 6]$.

With 84 fewer total iterations($\sim \frac{2}{3}$ iterations of standard QR), our final algorithm is 10 times more accurate. We now turn to modifications of the standard QR algorithm to achieve the results above. There are two improvements we will make: (i) make each iteration cheaper by first reducing A to tridiagonal form (which we did not measure in the numerical example), and (ii) accelerate convergence using shifts.

Algorithm 3 QR algorithm for symmetric matrix (store eigenvectors)

Require: $A \in \mathbb{R}^{n\times n}$ symmetric.
  Set $A_1 := A$, $Q^{(0)} := I$
  for $k = 1, 2, \ldots$ **do**
    Form the QR factorization $A_k = Q_k R_k$
    Set $A_{k+1} := R_k Q_k$
    if we want eigenvectors **then**
      Set $Q^{(k)} := Q^{(k-1)} Q_k$.
    end if
  end for

Reduction to tridiagonal form

The dominant computational cost in Algorithm 3 is the computation of the QR factorization of A_k at $\mathcal{O}(n^3)$ flops. For the first improvement, we will reduce A to a tridiagonal matrix for which the QR factorization can be computed in only $\mathcal{O}(n)$ flops, as we will show later.

The idea is to find an explicit orthogonal matrix Q so that $B = QAQ^{-1}$ is tridiagonal and has the same eigenvalues of A. Recall from the QR factorization lecture that we can find a Householder reflector $H(w)$ so that

$$H(w)A = \begin{bmatrix} \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix}.$$

For general matrices, the two-sided product $H(w)AH(w)^\top$ is full, i.e., all zeros created by pre-multiplication are destroyed by the post-multiplication. However, we have a symmetric matrix so that

$$A = \begin{bmatrix} \gamma & u^\top \\ u & M \end{bmatrix}.$$

We then choose

$$w = \begin{bmatrix} 0 \\ \hat{w} \end{bmatrix} \quad \text{where} \quad H(\hat{w})u = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

which gives

$$H(w)A = \begin{bmatrix} \gamma & & u^\top & \\ \alpha & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix}.$$

The key difference from the QR lecture is that we have *two* nonzero elements in the first column so that the $u^\top$ part of the first row of A is unchanged. We can now right multiply by $H(w)^{-1} = H(w)^\top = H(w)$ and use that $u^\top H(\hat{w})^\top = (H(w)u)^\top = \begin{bmatrix} \alpha & 0 & \cdots & 0 \end{bmatrix}$ to obtain

$$H(w)AH(w) = H(w)AH(w)^{-1} = H(w)AH(w)^\top = \left[\begin{array}{c|ccc} \gamma & \alpha & 0 & \cdots & 0 \\ \hline \alpha & & & & \\ 0 & & & & \\ \vdots & & C_{n-1} & & \\ 0 & & & & \end{array}\right]$$

where $C_{n-1} = H(\hat{w})MH^\top(\hat{w})$. Also note that C_{n-1} and $H(w)AH(w)^\top$ is symmetric as A is. Now, we inductively apply this to the smaller matrix C_{n-1}, as described for the QR factorization but using post- as well as pre-multiplications. The result of $n-2$ such Householder similarity transformations is the matrix

$$B := H(w_{n-2}) \cdots H(w_2)H(w)AH(w)H(w_2) \cdots H(w_{n-2}),$$

which is symmetric and tridiagonal.

Exercise: Show that the cost of computing B is $\mathcal{O}(n^3)$ operations.

QR factorization of tridiagonal matrix

We now show how the QR factorization of a tridiagonal matrix A can be achieved in $\mathcal{O}(n)$ flops. Instead of using Householder reflections, we will use *Givens* rotation $J(i,j)$. The matrix $J(i,j) \in \mathbb{R}^{n \times n}$ is equal to the identity matrix I except in the $(i,i), (i,j), (j,i), (j,j)$

entries which have values $c, s, -s, c$ with $c^2 + s^2 = 1$ (corresponding to cos's and sin's of an angle):

$$J(i,j) := \begin{bmatrix} I_{(i-1)} & & & & & \\ & c & & s & & \leftarrow i \\ & & I_{(j-i+1)} & & & \\ & -s & & c & & \leftarrow j \\ & & & & I_{(n-j)} & \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \\ \begin{matrix} & \uparrow & & \uparrow & \\ & i & & j & \end{matrix}$$

First note that we can choose c so that

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sqrt{a^2+b^2} \\ 0 \end{bmatrix},$$

and so

$$J(1,2)A = \begin{bmatrix} \times & \times & \times & 0 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & 0 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \times & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \times & \times & \times & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}$$

We can continue with $n-1$ total steps to obtain

$$\underbrace{J(n-1,n)\cdots J(2,3)J(1,2)}_{Q^\top} A = R, \qquad \text{upper triangular.}$$

Precisely, R has a diagonal and 2 super-diagonals,

$$R = \begin{bmatrix} \times & \times & \times & 0 & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \times & 0 & 0 & \cdots & 0 \\ 0 & 0 & \times & \times & \times & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \times \end{bmatrix}$$

(exercise: check!).

Lemma. Suppose that A is a symmetric tridiagonal matrix and that we compute $A = QR$ using Givens rotations as above. Then, RQ is symmetric and tridiagonal.

Proof. We have already shown that if $A = QR$ is symmetric, then so is RQ. If $A = QR = J(1,2)^\top J(2,3)^\top \cdots J(n-1,n)^\top R$ is tridiagonal, then $RQ = RJ(1,2)^\top J(2,3)^\top \cdots J(n-$

$1, n)^\top$. Recall that post-multiplication of a matrix by $J(i, i+1)^\top$ replaces columns i and $i+1$ by linear combinations of the pair of columns, while leaving columns $j = 1, 2, \ldots, i-1, i+2, \ldots, n$ alone. Thus, since R is upper triangular, the only subdiagonal entry in $RJ(1,2)^\top$ is in position $(2,1)$. Similarly, the only subdiagonal entries in $RJ(1,2)^\top J(2,3)^\top = (RJ(1,2)^\top)J(2,3)^\top$ are in positions $(2,1)$ and $(3,2)$. Inductively, the only subdiagonal entries in

$$\begin{aligned} &RJ(1,2)^\top J(2,3)^\top \cdots J(i-2,i-1)^\top J(i-1,i)^\top \\ &\quad = (RJ(1,2)^\top J(2,3)^\top \cdots J(i-2,i-1)^\top)J(i-1,i)^\top \end{aligned}$$

are in positions $(j, j-1)$, $j = 2, \ldots i$. So, the lower triangular part of RQ only has nonzeros on its first subdiagonal. However, then since RQ is symmetric, it must be tridiagonal. $\square$

The above lemma means that the following QR algorithm for symmetric, tridiagonal matrices is well-defined:

Algorithm 4 QR algorithm for symmetric tridiagonal matrix

Require: $A \in \mathbb{R}^{n\times n}$ symmetric and tridiagonal.
  Set $A_1 := A$
  for $k = 1, 2, \ldots$ **do**
    Form the QR factorization $A_k = Q_k R_k$ using Givens rotations.
    Set $A_{k+1} := R_k Q_k$
  end for

Operation count: Note that $J(1,2)A$ only affects the first three columns of A and can be computed in $\mathcal{O}(1)$ operations since A has at most 3 entries per column. One can show that this property remains true throughout the reduction: $J(i-1,i)\cdot(J(i-2,i-1)\cdots J(1,2)A)$ can be computed in $\mathcal{O}(1)$ operations. Thus, computing R takes only $\mathcal{O}(n)$ operations. Similarly, the product RQ can be computed in $\mathcal{O}(n)$ operations. Thus, each iteration of the QR algorithm in Algorithm 4 can be computed in $\mathcal{O}(n)$ operations.

Accelerating convergence: Using shifts

One further and final step in making an efficient algorithm is the use of **shifts**:

Algorithm 5 QR algorithm for symmetric tridiagonal matrix with shifts

Require: $A \in \mathbb{R}^{n\times n}$ symmetric and tridiagonal.
  Set $A_1 := A$
  for $k = 1, 2, \ldots$ **do**
    Pick a shift $\mu_k \in \mathbb{R}$.
    Form the QR factorization $A_k - \mu_k I = Q_k R_k$ using Givens rotations.
    Set $A_{k+1} := R_k Q_k + \mu_k I$
  end for

For any choice of shifts $\{\mu_k\}_{k=1}^\infty \subset \mathbb{R}$, the iterates $\{A_k\}_{k=1}^\infty$ are symmetric, tridiagonal, and similar to A.

The simplest shift to use is the lower right entry $a_{n,n}$ of A_k which converges rapidly in almost all cases to

$$A_k = \left[\begin{array}{ccc|c} & & & 0 \\ & T_k & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & \lambda \end{array}\right],$$

where $T_k \in \mathbb{R}^{(n-1)\times(n-1)}$ is symmetric and tridiagonal and λ is an eigenvalue of A. Once we have converged, then we can run the QR algorithm on T_k, again using the lower right entry as the shift, and so on. This process is called **deflation** because we are deflating/removing the eigenvalue we have already found from the system. The overall algorithm is in Algorithm 6.

Algorithm 6 QR algorithm for symmetric matrix with shifts

Require: $A \in \mathbb{R}^{n\times n}$ symmetric and tolerance ϵ.
  Reduce A to tridiagonal form using (Householder) similarity transformations.
  for $m = n, n-1, \ldots, 2$ **do**
    while $a_{m-1,m} > \epsilon$ **do**
      Form the QR factorization of $A - a_{m,m}I = QR$ using Givens rotations.
      $A \leftarrow RQ + a_{m,m}I$.
    end while
    Record eigenvalue $\lambda_m = a_{m,m}$.
    $A \leftarrow$ leading $m-1$ by $m-1$ submatrix of A.
  end for
  Record eigenvalue $\lambda_1 = a_{1,1}$. ▷ The eigenvalues may be in an arbitrary order.

Why does introducing a shift μ help? Recall that the final column of Q is the result of applying the inverse power method to $A - \mu I$ with starting vector e_n. The eigenvalues of $A - \mu I$ are $\lambda_1 - \mu, \lambda_2 - \mu, \ldots, \lambda_n - \mu$, and so the eigenvalues of $(A - \mu I)^{-1}$ are $(\lambda_1 - \mu)^{-1}, (\lambda_2 - \mu)^{-1}, \ldots, (\lambda_n - \mu)^{-1}$. The inverse power method then converges at a rate

$$\left|\frac{\lambda_{\sigma(n)} - \mu}{\lambda_{\sigma(n-1)} - \mu}\right|,$$

where σ is a permutation such that $|\lambda_{\sigma(1)} - \mu| \geq |\lambda_{\sigma(2)} - \mu| \geq \cdots \geq |\lambda_{\sigma(n)} - \mu|$. If μ is close to an eigenvalue, this implies (potentially extremely) fast convergence; in fact by choosing the shift $\mu_k = a_{n,n}$, it can be shown that (proof omitted and non-examinable) $a_{m,m-1}$ converges *cubically*: $|a_{m,m-1,k+1}| = O(|a_{m,m-1,k}|^3)$, where $a_{m,m-1,j}$ is the $(m, m-1)$ entry of A_j.

Summary of NLA Material

- We can solve square systems $Ax = b$ using the LU factorization. Compute the factors takes $\mathcal{O}(n^3)$ operations, while forward and back substitution require $\mathcal{O}(n^2)$ operations each. The latter fact is particularly useful when solving $Ax = b$ for many different b's with the same A.

- Every rectangular matrix $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ admits a full QR factorization and a thin QR factorization. The case $m \leq n$ also has a full QR factorization. For a square matrix, computing a QR factorization is about twice as expensive as computing an LU factorization.

- Using QR factorizations, we can solve least squares problems of the two types: (i) overdetermined systems $m \geq n$ of the form $\min_x \|Ax - b\|_2$ and (ii) underdetermined systems $m \leq n$ of the form $\min_x \|x\|_2$ where we minimize over all x such that $Ax = b$. The first type uses the QR factorization of A, while the second type uses the QR factorization of $A^\top$. While the proof using the full QR factorizations, only the thin QR factorizations are needed for computing the solution.

- Every rectangular matrix $A \in \mathbb{R}^{m \times n}$ admits a full SVD and thin SVD. We can truncate the SVD of A by forming $A_r = \sum_{i=1}^{r} \sigma_i u_i v_i^\top$, where u_i and v_i are the left and right singular vectors of A. This truncation is the best rank-r approximation to A in the $\|\cdot\|_2$-norm.

- If $A \in \mathbb{R}^{n \times n}$, then the eigenvalue problem $Ax = \lambda x$ must be "solved" (approximated) using iterative methods if $n \geq 5$. The Gerschgorin theorems are sometimes useful in estimating the eigenvalues of A. The power method is an extremely simple method to approximate the largest eigenvalue and its corresponding eigenvector. The QR algorithm (with tridiagonal reduction and shifts) is very efficient method for finding all eigenvalues, in part because the algorithm implicitly performs the inverse power method on the shifted matrix.