Numerical Methods I Overview*

Filip Belik

August 30, 2023

1 Matrix-Vector Multiplication

Theorem 1. range(A) is the space spanned by the columns of A.

Theorem 2. A matrix $A \in \mathbb{C}^{m\times n}$ with $m \geq n$ has full rank if and only if it maps no two distinct vectors to the same vector.

Theorem 3. The following conditions are equivalent for $A \in \mathbb{C}^{m\times m}$

1. A has an inverse A^{-1},
2. rank$(A) = m$,
3. null$(A) = \{\vec{0}\}$,
4. 0 is not an eigenvalue of A,
5. 0 is not a singular value of A,
6. det$(A) \neq 0$.

2 Orthogonal Vectors and Matrices

A matrix A is **hermitian** if $A^* = A$.

A pair of vectors $x, y \in \mathbb{C}^m$ are **orthogonal** if $x^*y = 0$. A set of vectors is **orthogonal** if each pair of vectors in it are orthogonal and all vectors are non-zero. A set of vectors S is **orthonormal** if it is orthogonal, and $\|x\| = 1$ for each $x \in S$

*Sources: Trefethen and Bau Numerical Linear Algebra and Class

Theorem 4. The vectors in an orthogonal set are linearly independent.

Given a set of orthonormal vectors $\{q_1, q_2, \dots, q_n\}$, and v an arbitrary vector, the vector

$$r = v - (q_1^* v)q_1 - (q_2^* v)q_2 - \cdots - (q_n^* v)q_n$$

is orthogonal to each q_i. We are subtracting the components of v in the direction of each q_i. This also tells us that v can be decomposed into $n+1$ orthogonal elements

$$v = r + \sum_{i=1}^{n} (q_i^* v)q_i.$$

A square matrix is **unitary** if $Q^* = Q^{-1}$. Unitary matrices carry the following properties

1. Unitary matrices preserve inner products: $(Qx)^*(Qy) = x^* y$,
2. Unitary matrices preserve lengths: $\|Qx\| = \|x\|$ and $\|yQ\| = \|Q\|$,
3. Determinant of ± 1, so each eigenvalue has magnitude ± 1, causes rotation and reflection, not stretching.

3 Norms

A **vector norm** is a function $\|\cdot\| : \mathbb{C}^m \to R$ which satisfies the following properties ($\forall x, y \in \mathbb{C}^m, \alpha \in C$):

1. $\|x\| \geq 0$, and $\|x\| = 0$ iff $x = \vec{0}$,
2. $\|x + y\| \leq \|x\| + \|y\|$,
3. $\|\alpha x\| = |\alpha| \|x\|$.

The p-norm of a vector $x \in \mathbb{C}^m$ is given by

$$\|x\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{1/p}$$

for $1 \leq p < \infty$. In the limit as $p \to \infty$, the ∞-norm of x is given by

$$\|x\|_\infty = \max_{1 \leq i \leq m} |x_i|.$$

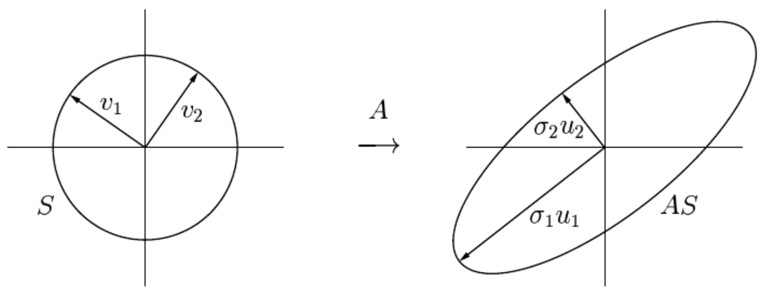


Figure 1: SVD of a 2×2 matrix.

Given $A \in \mathbb{C}^{m\times n}$, and vector norms $\|\cdot\|_{(m)}$ and $\|\cdot\|_{(n)}$, the **induced matrix norm** $\|A\|_{(m,n)}$ is given by

$$\|A\|_{(m,n)} = \sup_{x\in\mathbb{C}^n, \|x\|_{(n)}=1} \|Ax\|_{(m)}.$$

The **1-norm** of a matrix is equal to the maximum column sum of A. The ∞**-norm** is equal to the maximum row sum of A.

The **Hölder inequality** says that for any vectors x and y,

$$|x^* y| \leq \|x\|_{(p)} \|y\|_{(q)}$$

for $1 \leq p, q \leq \infty$ with $1/p + 1/q = 1$. The **Cauchy Schwartz inequality** is the special case of this for $p = q = 2$:

$$|x^* y| \leq \|x\|_{(2)} \|y\|_{(2)}$$

Theorem 5. For any $A \in \mathbb{C}^{m\times n}$ and unitary $Q \in \mathbb{C}^{m\times n}$, we have

$$\|QA\|_2 = \|A\|_2, \quad \|QA\|_F = \|A\|_F.$$

Similarly, it can be shown that

$$\|AQ\|_2 = \|A\|_2, \quad \|AQ\|_F = \|A\|_F.$$

4 The Singular Value Decomposition

The SVD is motivated by the fact that the image of the unit sphere under any $m \times n$ matrix is a hyperellipse. This is visualized in Figure 1.

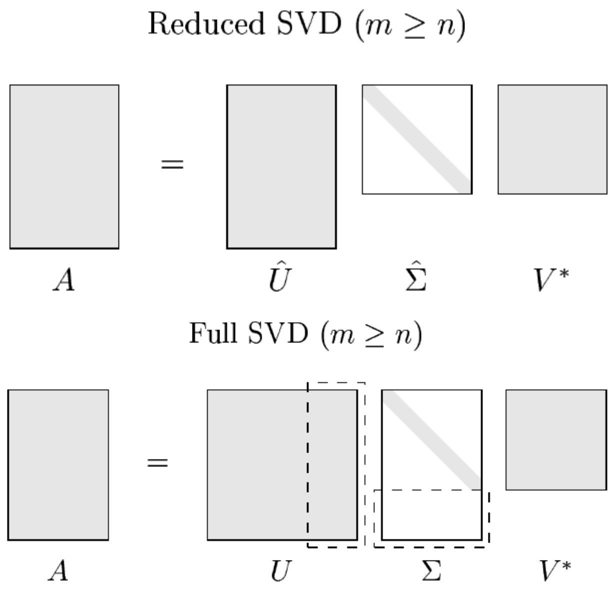


Figure 2: Visualization of full and reduced SVDs.

Given any $A \in \mathbb{C}^{m\times n}$, it can be decomposed into $A = U\Sigma V^*$ where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary, and $\Sigma \in \mathbb{C}^{m\times n}$ is diagonal with descending diagonal elements. This is called the **Singular Value Decomposition** or **SVD** of A. The diagonal elements of Σ, denoted σ_i, are the **singular values**. The columns of V (or rows of V^*) are the **right singular vectors**. And the columns of U are the **left singular vectors**. The algebraic relation between these quantities is given by

$$Av_i = \sigma_i u_i$$

for $1 \leq i \leq n$.

The **reduced SVD** of A is given by $A = \hat{U}\hat{\Sigma}\hat{V}^*$ where $\hat{U} \in \mathbb{C}^{m\times n}$, $\hat{\Sigma} \in \mathbb{C}^{n\times n}$, and $\hat{V} \in \mathbb{C}^{n\times n}$. To get from the reduced SVD to the full SVD, we extend $\hat{U}$ with $m-n$ orthonormal columns, and $\hat{\Sigma}$ by $m-n$ zero-rows. This is displayed in Figure 2.

Theorem 6. Every matrix $A \in \mathbb{C}^{m\times n}$ has a singular value decomposition. Furthermore, the singular values are uniquely determined, and if A is square, and each σ_i distinct, then the left and right singular values are uniquely determined up to complex signs.

5 More on the SVD

Theorem 7. The rank of A is equal to the number of nonzero singular values. Or $\text{rank}(A) = \text{rank}(\Sigma) = r$.

Theorem 8. The range of A is given by $\langle u_1, \dots, u_r \rangle$, the span of the first r left singular values, and the nullspace is given by $\langle v_{r+1}, \dots, v_n \rangle$, the last $n - r$ right singular values.

Theorem 9. $\|A\|_2 = \sigma_1$ and $\|A\|_F = \sqrt{\sigma_1^2 + \cdots \sigma_r^2}$.

Theorem 10. The nonzero singular values of A are the square roots of the nonzero eigenvalues of A^*A or AA^*.

Theorem 11. If A is hermitian, $A = A^*$, then the singular values of A are the absolute values of the eigenvalues of A.

Theorem 12. For $A \in \mathbb{C}^{m \times m}$, then the absolute value of the determinant of A is given by

$$|\det(A)| = \prod_{i=1}^{m} \sigma_i.$$

Theorem 13. A can be written as the sum of r rank-one matrices

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^*.$$

Theorem 14. For any ν with $0 \leq \nu \leq r$, define

$$A_\nu = \sum_{j=1}^{\nu} \sigma_j u_j v_j^*.$$

Then

$$\|A - A_\nu\|_2 = \inf_{B \in \mathbb{C}^{m \times n}, \text{rank}(B) \leq \nu} \|A - B\|_2 = \sigma_{\nu+1}.$$

Theorem 15. For any ν with $0 \leq \nu \leq r$, the matrix A_ν also satisfies

$$\|A - A_\nu\|_F = \inf_{B \in \mathbb{C}^{m \times n}, \text{rank}(B) \leq \nu} \|A - B\|_F = \sqrt{\sigma_{\nu+1}^2 + \cdots + \sigma_r}.$$

The methods for computing the SVD is a complicated subject, but once it is computed, it is very helpful. For example, we can then easily determine the rank, an orthonormal basis of the range or nullspace, the two-norm, the Frobenius norm, and more.

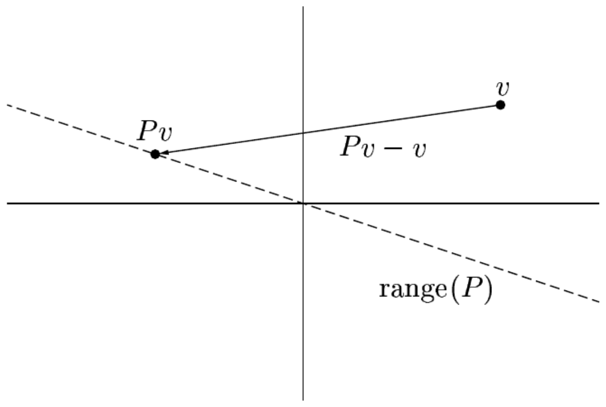


Figure 3: An oblique projector.

6 Projectors

A **projector** is a square matrix P that satisfies $P^2 = P$. It is also called **indepotent**. It is an **oblique projector** if it is not orthogonal. This is visualized in Figure 3.

The **complementary projector** to a projector P is another projector given by $(I - P)$. It satisfies that

$$\text{range}(I - P) = \text{null}(P), \quad \text{null}(I - P) = \text{range}(P).$$

This gives us the helpful fact that

$$\text{range}(P) \cap \text{null}(P) = \{0\}.$$

Geometrically, a projector separates $\mathbb{C}^m$ into two spaces. Calling them S_1 and S_2, we say that P projects onto S_1 along S_2.

An **orthogonal projector** is a projector that projects onto a space S_1 along a space S_2 where S_1 and S_2 are orthogonal.

Theorem 16. A projector P is orthogonal if and only if P is hermitian, $P = P^*$.

Denoting unit length basis vectors for S_1 as $s_1, s_2, \ldots, s_n$, and unit length basis vectors for S_2 as $s_{n+1}, s_{n+2}, \ldots, s_m$, we can construct the matrix

$$Q = \begin{bmatrix} s_1 & s_2 & \cdots & s_n & s_{n+1} & s_{n+2} & \cdots & s_m \end{bmatrix}.$$

Then, applying P to it results in

$$PQ = \begin{bmatrix} s_1 & s_2 & \cdots & s_n & \vec{0} & \vec{0} & \cdots & \vec{0} \end{bmatrix}.$$

And since the columns of Q are orthonormal, we have that

$$Q^*PQ = \Sigma = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Hence, we have an SVD for P as

$$P = Q\Sigma Q^*.$$

We can arbitrarily define an orthogonal rank one projector onto the span of any vector a by

$$P = \frac{aa^*}{a^*a}.$$

Similarly, given linearly independent vectors $a_1, a_2, \ldots, a_n$, defining the matrix A with column i as a_i, we can define the orthogonal projector onto $\langle a_1, a_2, \ldots, a_n \rangle$ by

$$P = A(A^*A)^{-1}A^*.$$

7 QR Factorization

The **reduced QR factorization** of a matrix A is $A = \hat{Q}\hat{R}$ where $\hat{Q} \in \mathbb{C}^{m \times n}$ has orthonormal columns and $\hat{R} \in \mathbb{C}^{n \times n}$ is upper triangular. In the **full QR factorization**, we append $m - n$ orthonormal columns to $\hat{Q}$ to form Q, and we append $m - n$ rows of zeros to $\hat{R}$, to form $A = QR$ with $Q \in \mathbb{C}^{m \times m}$ and $R \in \mathbb{C}^{m \times n}$. This is visualized in Figure 4.

The Classical Gram-Schmidt Algorithm is an unstable algorithm that provides existence and uniqueness for QR factorizations. It is detailed in Figure 5.

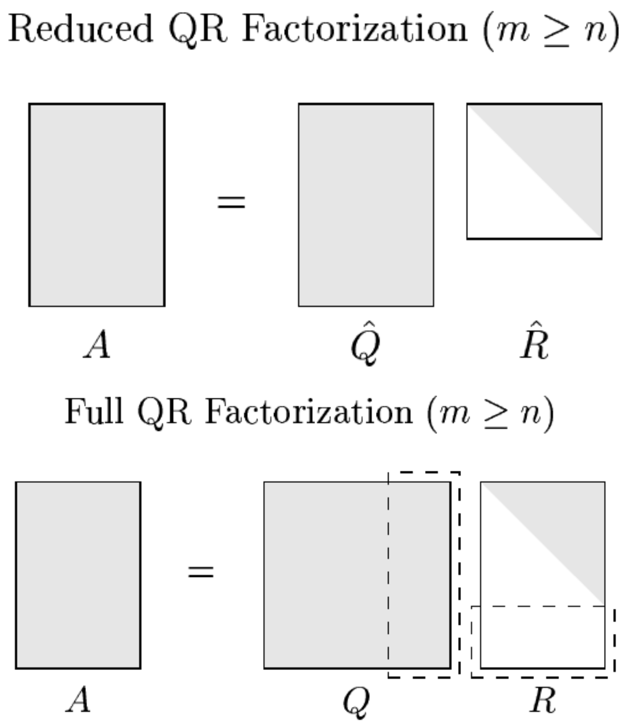

Reduced QR Factorization $(m \geq n)$
=
A
$\hat{Q}$
$\hat{R}$
Full QR Factorization $(m \geq n)$
=
A
Q
R

Figure 4: Visualization of full and reduced QRs.

Algorithm 7.1. Classical Gram–Schmidt (unstable)

```
for $j = 1$ to $n$
    $v_j = a_j$
    for $i = 1$ to $j - 1$
        $r_{ij} = q_i^* a_j$
        $v_j = v_j - r_{ij} q_i$
    $r_{jj} = \|v_j\|_2$
    $q_j = v_j / r_{jj}$
```

Figure 5: Classial Gram-Schmidt Algorithm.

Algorithm 8.1. Modified Gram–Schmidt

```
for $i = 1$ to $n$
    $v_i = a_i$
for $i = 1$ to $n$
    $r_{ii} = \|v_i\|$
    $q_i = v_i / r_{ii}$
    for $j = i + 1$ to $n$
        $r_{ij} = q_i^* v_j$
        $v_j = v_j - r_{ij} q_i$
```

Figure 6: Modified Gram-Schmidt Algorithm.

Theorem 17. Every $A \in \mathbb{C}^{m \times n}$ with $(m \geq n)$ has a full QR factorization, hence also a reduced QR factorization.

Theorem 18. Every $A \in \mathbb{C}^{m \times n}$ with $(m \geq n)$ of full rank has a unique reduced QR factorization with nonnegative diagonal entries.

Note that equations of the form $Ax = b$ can be solved much easier after a QR factorization for A has been determined. Given $A = QR$. we first solve $y = Q^* b$, and then $Rx = y$ using back substitution.

8 Gram-Schmidt Orthogonalization

The modified Gram-Schmidt algorithm is more stable than the classical Gram-Schmidt algorithm and produces the same result by performing the operations in a different order. The algorithm is provided in Figure 6.

Theorem 19. Both the classical Gram-Schmidt and modified Gram-Schmidt algorithms require $\sim 2mn^2$ flops to compute a QR factorization of a $m \times n$ matrix.

9 MATLAB

This section is skipped.

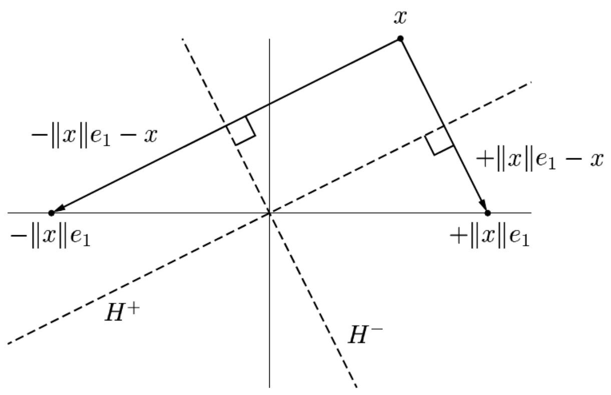


Figure 7: Two possible Householder reflectors such that x is mapped onto the span of e_1.

10 Householder Triangularization

The Householder triangularization method uses unitary matrices to turn a matrix A upper-triangular. Algebraically,

$$Q_n Q_{n-1} \cdots Q_1 A = R \implies A = Q_1^* \cdots Q_{n-1}^* Q_n^* R.$$

Each Q is of the form of $(I - 2P)$ where P is some orthogonal projector, $P = \frac{vv^*}{v^*v}$. It can be shown fairly easily that $(I - 2P)$ is unitary as it simply reflects across the span of v. As can be seen in Figure 7, there are two possible such reflectors. Choosing $v = (x_1)\|x\|e_1 + x$ satisfies the desired properties and chooses the reflector that moves x further, which allows for more stability.

We first choose x_1 to be the first row of A, and applying $Q_1 = (I - 2P)$ to A results in a matrix where the first column has all zeros below the first entry. We then choose x_2 to be the second column of $Q_1 A$ from entries two down, choose Q_2 similarly, and pad Q_2 with the identity column and row above and behind it, to get $Q_2 Q_1 A$ having a "diagonal" first and second columns. This process repeats until the product is fully diagonal. The algorithm is provided in Figure 8.

Note that the Householder algorithm in Figure 8 constructs (in-place) the upper-triangular matrix R, and vectors v_k. The vectors v_k are the same as those used for the Householder reflectors. They can be used to implicitly calculate the product Q^*b or Qx. Those are provided in Figure 9. Note that

```
Algorithm 10.1. Householder QR Factorization

for $k = 1$ to $n$
    $x = A_{k:m,k}$
    $v_k = \text{sign}(x_1)\|x\|_2 e_1 + x$
    $v_k = v_k / \|v_k\|_2$
    $A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^* A_{k:m,k:n})$
```

Figure 8: Householder QR Factorization algorithm.

```
Algorithm 10.2. Implicit Calculation of a Product $Q^*b$

for $k = 1$ to $n$
    $b_{k:m} = b_{k:m} - 2v_k(v_k^* b_{k:m})$
```

```
Algorithm 10.3. Implicit Calculation of a Product $Qx$

for $k = n$ downto $1$
    $x_{k:m} = x_{k:m} - 2v_k(v_k^* x_{k:m})$
```

Figure 9: Compute Q^*b or Qx from v vectors.

from Figure 9, the matrix Q can be found explicitly by multiplying Q by each of the identity vectors to produce each column of Q.

The work for Householder orthogonalization is $\sim 2mn^2 - \frac{2}{3}n^3$ flops.

11 Least Squares Problems

In general, equations of the form $Ax = b$ where $A \in \mathbb{C}^{m \times n}$ with $m > n$ and $b \in \mathbb{C}^m$ have no solutions. Such systems of equations are called **overdetermined**. Given a "solution" $x \in \mathbb{C}^n$, the **residual** is $r = b - Ax \in \mathbb{C}^m$. The general (linear) **least squares problem** involves finding a solution $x \in \mathbb{C}^n$ such that $\|b - Ax\|_2$ is minimized.

Theorem 20. Let $A \in \mathbb{C}^{m \times n}$ $(m \geq n)$ and $b \in \mathbb{C}^m$ be given. A vector $x \in \mathbb{C}^n$ minimizes the residual norm $\|r\|_2 = \|b - Ax\|_2$ if and only if $r \perp \text{range}(A)$, this is equivalent to

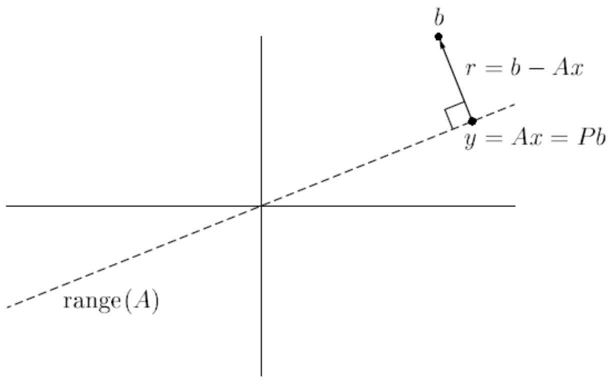


Figure 10: Illustration of least squares in $\mathbb{R}^2$.

1. $A^*r = 0$,
2. or $A^*Ax = A^*b$,
3. or $Pb = Ax$

where $P \in \mathbb{C}^{m\times m}$ is the orthogonal projector onto range(A). This is illustrated in Figure 10. The $n \times n$ system of equations in $A^*Ax = A^*b$, known as the normal equations, is nonsingular if and only if A has full rank. Consequently, the solution is unique if and only if A has full rank.

Given by the normal equations, if A is full rank, then the solution is unique and given by

$$x = (A^*A)^{-1}A^*b.$$

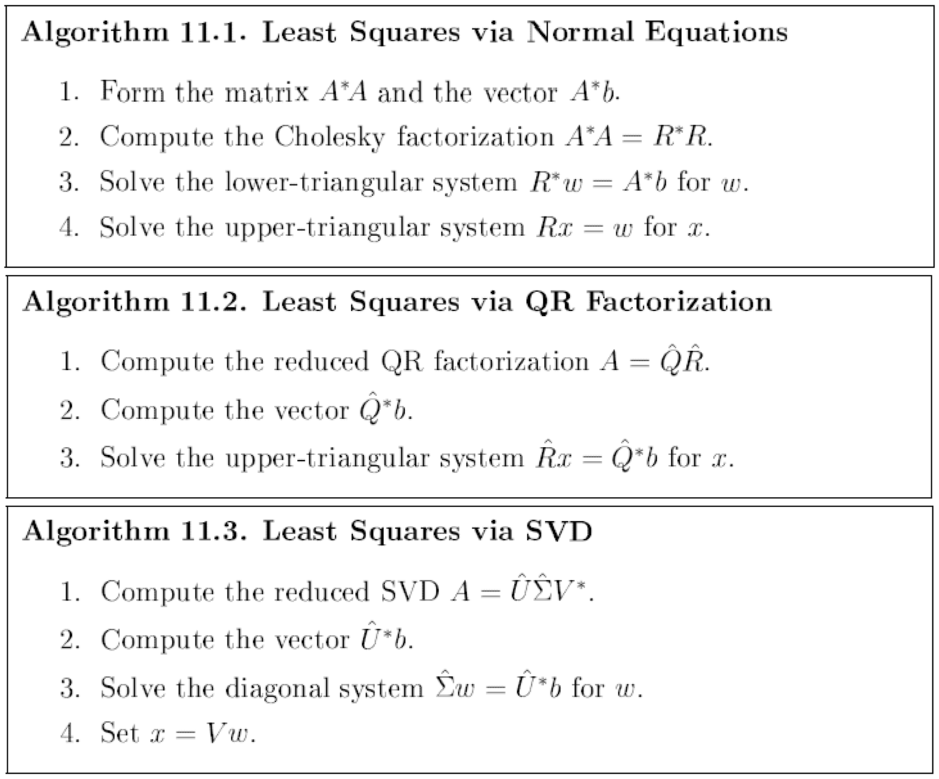
We then denote the **pseudoinverse** of A as $A^+ \in \mathbb{C}^{n\times m}$ where

$$A^+ = (A^*A)^{-1}A^*.$$

There are three primary algorithms described for solving such problems. They are all presented in Figure 11.

The **method of normal equations** essentially involves solving for the pseudoinverse, but utilizes Choelsky factorization to take advantage of using back-substitution to solve triangular systems. This method requires $\sim mn^2 + \frac{1}{3}n^3$ flops.

The **method of QR factorization** involves solving after a reduced QR decomposition has been computed. This is because then $A^*Ax = A^*b \implies$



Algorithm 11.1. Least Squares via Normal Equations

1. Form the matrix A^*A and the vector A^*b.
2. Compute the Cholesky factorization $A^*A = R^*R$.
3. Solve the lower-triangular system $R^*w = A^*b$ for w.
4. Solve the upper-triangular system $Rx = w$ for x.

Algorithm 11.2. Least Squares via QR Factorization

1. Compute the reduced QR factorization $A = \hat{Q}\hat{R}$.
2. Compute the vector $\hat{Q}^*b$.
3. Solve the upper-triangular system $\hat{R}x = \hat{Q}^*b$ for x.

Algorithm 11.3. Least Squares via SVD

1. Compute the reduced SVD $A = \hat{U}\hat{\Sigma}V^*$.
2. Compute the vector $\hat{U}^*b$.
3. Solve the diagonal system $\hat{\Sigma}w = \hat{U}^*b$ for w.
4. Set $x = Vw$.

Figure 11: Three methods for solving the least squares problem.

$\hat{R}x = \hat{Q}^*b$ which involves multiplication by a unitary matrix followed by back-substituting to solve for x. This method requires $\sim 2mn^2 + \frac{2}{3}n^3$ flops.

The **method of SVD** is nearly the same as in the QR case, but instead the equation simplifies to $\hat{\Sigma}V^*x = \hat{U}^*b$. This method requires $\sim 2mn^2 + 11n^3$ flops.

If speed is the sole consideration, then the normal equations method may be best. If stability is important, then QR factorization is recommended. And if A is near rank-deficient, then it turns out that the SVD method may be best.

12 Conditioning and Condition Numbers

Define a **problem** as a function $f : X \to Y$ where X is our state of data and Y is our set of solutions. Define a **problem instance** as the behavior of f on a specific input $x \in X$.

A **well-conditioned problem** (instance) is one with the property that small perturbations to x yield small changes in $f(x)$. Otherwise it is called **ill-conditioned**.

Let $\delta x \in X$ denote a small perturbation (small norm in X). And denote $\delta f = f(x + \delta x) - f(x)$. Then we define the **absolute condition number**, $\hat{\kappa}(x)$ or just $\hat{\kappa}$, of f at x by

$$\hat{\kappa} = \lim_{\delta \to 0} \sup_{\|\delta x\| < \delta} \frac{\|\delta f\|}{\|\delta x\|}.$$

Define the **relative condition number**, $\kappa(x)$ or just κ, of f at x by

$$\kappa = \lim_{\delta \to 0} \sup_{\|\delta x\| < \delta} \left(\frac{\|\delta f\|}{\|f(x)\|} \Big/ \frac{\|\delta x\|}{\|x\|} \right) = \lim_{\delta \to 0} \sup_{\|\delta x\| < \delta} \hat{\kappa} \Big/ \frac{\|f(x)\|}{\|x\|}.$$

In the case that f is differentiable, we use the first-order approximation $\delta f(x) \approx J(x)\delta x$ where $J(x)$ is the Jacobian of f. This simplifies our equations to

$$\hat{\kappa} = \|J(x)\|, \quad \kappa = \|J(x)\| \Big/ \frac{\|f(x)\|}{\|x\|}.$$

Typically we focus on the relative condition number, and wish for it to be small.

A frequently appearing value is the relative norm of a square, nonsingular matrix

$$\kappa(A) := \|A\| \left\|A^{-1}\right\|.$$

We call this the **condition number** of A.

Theorem 21. Let $A \in \mathbb{C}^{m \times m}$ be nonsingular and consider the equation $Ax = b$. The problem of computing b, given x, has condition number

$$\kappa = \|A\| \frac{\|x\|}{\|b\|} \leq \kappa A$$

with respect to perturbing x. The problem of computing x, given b, has condition number

$$\kappa = \left\|A^{-1}\right\| \frac{\|b\|}{\|x\|} \leq \kappa(A)$$

with respect to perturbing b.

We will typically only focus on $\|\cdot\| = \|\cdot\|_2$, in which case

$$\kappa(A) = \frac{\sigma_1}{\sigma_n},$$

where σ_1 is the largest singular value and σ_n is the smallest singular value. Note that if A is not full-rank and $\sigma_n = 0$, then we say $\kappa(A) = \infty$.

Theorem 22. Let b be fixed and consider the problem of computing $x = A^{-1}b$ where A is square and nonsingular. The condition number of this problem with respect to perturbations in A is $\kappa(A)$.

13 Floating Point Arithmetic

IEEE stores decimal numbers in a way such that the number of possible floating point numbers between $[1, 2]$ is the same as the number between $[2, 4]$ and $[2^n, 2^{n+1}]$. Gaps between numbers are never larger than $2^{-52} \approx 2.22 \times 10^{-16}$.

We define $\epsilon_{\text{machine}}$ as a sort of resolution or best accuracy of our computer. This is typically equal to $2^{-53} \approx 1.11 \times 10^{-16}$. The distance between a real number and its closest floating point approximation is always smaller than $\epsilon_{\text{machine}}$ in relative terms.

$$\text{fl}(x) = x(1 + \epsilon), \quad |\epsilon| \leq \epsilon_{\text{machine}}.$$

For any floating point operation $\cdot_c$ (could represent addition, subtraction, multiplication, or division), and for any x, y that are floating point numbers, there exists ϵ with $|\epsilon| \leq \epsilon_{\text{machine}}$ such that

$$x \cdot_c y = (x \cdot y)(1+\epsilon),$$

where $\cdot$ is the true mathematical operation. This is called the **fundamental axiom of floating point arithmetic**.

14 Stability

Given a problem $f : X \to Y$, we define an **algorithm** as another map $\tilde{f} : X \to Y$ where $f(x)$ is the true solution given data x, and $\tilde{f}(x)$ is the computed solution.

We define the **absolute error** of a computation as $\left\|\tilde{f}(x) - f(x)\right\|$, and we define the **relative error** as

$$\frac{\left\|\tilde{f}(x) - f(x)\right\|}{\|f(x)\|}.$$

We call $\tilde{f}$ accurate for a problem f if

$$\frac{\left\|\tilde{f}(x) - f(x)\right\|}{\|f(x)\|} = O(\epsilon_{\text{machine}}).$$

We call an algorithm **stable** if for each $x \in X$, there exists $\tilde{x} \in X$ with

$$\frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_{\text{machine}})$$

such that

$$\frac{\left\|\tilde{f}(x) - f(\tilde{x})\right\|}{\|f(\tilde{x})\|} = O(\epsilon_{\text{machine}}).$$

In words, a stable algorithm gives nearly the right answer to nearly the right question.

More strictly, we call an algorithm **backward stable** if for each $x \in X$, there exists $\tilde{x} \in X$ with

$$\frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_{\text{machine}})$$

such that

$$\tilde{f}(x) = f(\tilde{x}).$$

In words, a backward stable algorithm gives exactly the right answer to nearly the right question.

Strictly, a function $f(\epsilon_{\text{machine}}) = O(\epsilon_{\text{machine}})$ means that in the limit as $\epsilon_{\text{machine}} \to 0$, there exists a positive constant C such that

$$|f(\epsilon_{\text{machine}})| \leq C\epsilon_{\text{machine}}.$$

Theorem 23. For problems f and algorithms $\tilde{f}$ defined on finite-dimensional spaces X and Y, the properties of accuracy, stability, and backward stability all hold or fail to hold regardless of choice of norms on X and Y.

15 More on Stability

This section includes examples of stable and unstable algorithms. They find the dot product to be backward stable, the outer product to be stable but not backward stable, and addition $x + 1$ to not be backward stable when $x \approx 0$ (as errors are large relative to $|x|$). But addition $x + y$ is backward stable. Finding the roots of a polynomial is unstable.

Theorem 24. Suppose a backward stable algorithm is applied to solve a problem $f : X \to Y$ with condition number κ on a computer satisfying the above axioms. Then the relative errors satisfy

$$\frac{\left\|\tilde{f}(x) - f(x)\right\|}{\|f(x)\|} = O(\kappa\epsilon_{\text{machine}}).$$

16 Stability of Householder Triangularization

An experiment was done where random matrices were computed, and Householder QR factorizations were done. It was found that the errors $\left\|\tilde{Q} - Q\right\|$ and $\left\|\tilde{R} - R\right\|$ were quite large, but the error $\left\|A - \tilde{Q}\tilde{R}\right\|$ was quite small. It turns out that the first two were examples of **forward errors**, which is due to the problem being ill-conditioned rather than the algorithm being unstable. But the error in $\tilde{Q}\tilde{R}$ is the **backward error** or the **residual**. The smallness of this value suggests that the algorithm itself is backward stable.

Theorem 25. Let the QR factorization $A = QR$ of a matrix $A \in \mathbb{C}^{m\times n}$ be computed by Householder factorization on a computer satisfying the above axioms, and let $\tilde{Q}$ and $\tilde{R}$ be the resulting computed terms. Then we have

$$\tilde{Q}\tilde{R} = A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = O(\epsilon_{\text{machine}}$$

for some $\delta A \in \mathbb{C}^{m\times n}$. In words, the solution $\tilde{Q}\tilde{R}$ is backward stable relative to perturbations in A.

Using that Householder QR factorization is backward stable, and that both matrix multiplication by a unitary matrix and back-substitution are also backward stable, we get the following theorem.

Theorem 26. The algorithm for solving $Ax = b$ where $A \in \mathbb{C}^{m\times m}$ for x by first finding $A = QR$ by Householder Triangularization, and then solving $y = Q^*b$, and finally $Rx = y$ by back-substitution is backward stable, satisfying

$$(A + \Delta A)\tilde{x} = b, \quad \frac{\|\Delta A\|}{\|A\|} = O(\epsilon_{\text{machine}})$$

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

Theorem 27. The solution $\tilde{x}$ as computed above satisfies

$$\frac{\|\tilde{x} - x\|}{\|x\|} = O(\kappa(A)\epsilon_{\text{machine}}).$$

17 Stability of Back Substitution

Solving $Rx = b$ where R is upper (or lower) triangular is fairly straightforward through the method of **back subtitution**, in which the last (first) element of x is able to be directly solved for, and then each following element is able to be solved for iteratively using knowledge of already calculated elements of x. This algorithm is provided in Figure 12, and it requires $\sim m^2$ flops for $R \in \mathbb{C}^{m\times m}$.

Theorem 28. Let the back subtitution algorithm in Figure 12 be used on a computer satisfying the desired axioms. This algorithm is backward stable in the sense that it generates a computed solution $\hat{x} \in \mathbb{C}^m$ satisfying

$$(R + \delta R)\tilde{x} = b$$

Algorithm 17.1. Back Substitution

$$\begin{aligned}
x_m &= b_m / r_{mm} \\
x_{m-1} &= (b_{m-1} - x_m r_{m-1,m}) / r_{m-1,m-1} \\
x_{m-2} &= (b_{m-2} - x_{m-1} r_{m-2,m-1} - x_m r_{m-2,m}) / r_{m-2,m-2} \\
&\vdots \\
x_j &= \left(b_j - \sum_{k=j+1}^{m} x_k r_{jk} \right) \Big/ r_{jj}
\end{aligned}$$

Figure 12: Back substitution algorithm

for some upper-triangular $\delta R \in \mathbb{C}^{m \times m}$ with

$$\frac{\|\delta R\|}{\|R\|} = O(\epsilon_{\text{machine}}).$$

Specifically, for each i, j,

$$\frac{|\delta r_{i,j}|}{|r_{i,j}|} \leq m\epsilon_{\text{machine}} + O(\epsilon_{\text{machine}}^2).$$

So we can see that for fixed m, computing x is backward stable in perturbing R. The larger R's dimension is, the more instability we can expect.

18 Conditioning of Least Squares Problems

For the least squares problem $Ax = b$, we take A and b to be data for our problem, while x and $y := Ax$ are said to be the solutions. This means we can define conditioning numbers for the least squares problem in terms of perturbing A or b, and the resulting effect on x or y. To do this we require additional parameters.

First, we will use the the conditioning number of the matrix A, $\kappa(A)$. If A is square, it is given by $\|A\|\|A^{-1}\| = \frac{\sigma_1}{\sigma_m}$, and in the rectangular case it is given by $\|A\|\|A^{+}\| = \frac{\sigma_1}{\sigma_n}$. Recall that the pseudoinverse of A is given by $A^{+} = (A^*A)^{-1}A^*$. We also define the parameter θ that measures the closeness of the fit

$$\theta = \arccos \frac{\|Ax\|}{\|b\|}.$$

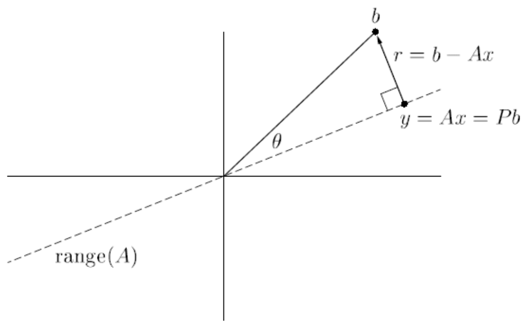


Figure 18.1. *The least squares problem (repetition of Figure 11.3).*

Figure 13: Least squares illustration.

Finally we define the parameter η of how much $\|y\| = \|Ax\|$ falls short of its maximum possible value $\|A\|\|x\|$,

$$\eta = \frac{\|A\|\|x\|}{\|y\|}.$$

Note that the parameters lie in the ranges

$$1 \leq \kappa(A) \leq \infty, \quad 0 \leq \theta \leq \pi/2, \quad 1 \leq \eta \leq \kappa(A).$$

The least squares problem is again illustrated in Figure 13.

Theorem 29. Let $b \in \mathbb{C}^m$ and $A \in \mathbb{C}^{m \times m}$ of full rank be fixed. the least squares problem has 2-norm relative condition numbers describing sensitivities of x and y to perturbations in b and A in Figure 14. The results for the first row are exact and the results in the second row are upper bounds.

19 Stability of Least Squares Problems

Theorem 30. Let the full-rank least squares problem be solved by Householder triangularization on a computer satisfying the desired axioms. This algorithm is backward stable in the sense that the computed solution $\tilde{x}$ has the properties that $\|(A + \delta A)\tilde{x} - b\|$ is minimized and

$$\frac{\|\delta A\|}{\|A\|} = O(\epsilon_{\text{machine}})$$

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

	y	x
b	$\dfrac{1}{\cos\theta}$	$\dfrac{\kappa(A)}{\eta\cos\theta}$
A	$\dfrac{\kappa(A)}{\cos\theta}$	$\kappa(A) + \dfrac{\kappa(A)^2\tan\theta}{\eta}$

Figure 14: Conditioning numbers for least squares problem.

Theorem 31. The solution of the full-rank least squares problem via the normal equations is unstable. Stability can be achieved, however, by restriction to a class of problems in which $\kappa(A)$ is uniformly bounded above or $\tan(\theta)/\eta$ is uniformly bounded below.

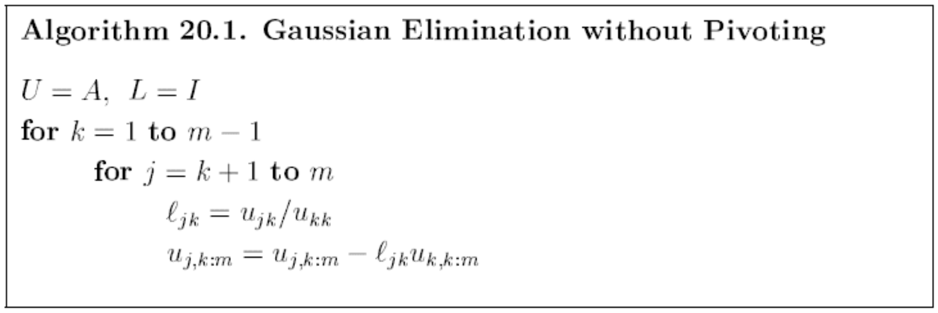
Theorem 32. The solution of the full-rank least squares problem by the SVD is backward stable.

20 Gaussian Elimination

Gaussian Elimination of **LU Factorization** is a process of generating $A = LU$ by triangular triangularization where L is unit lower triangular (1s on diagonal and 0s above), and U is lower triangular (0s below diagonal). The algorithm is provided in Figure 15, and it requires $\sim \frac{2}{3}m^3$ flops. In homework, we showed that a nonsingular matrix had an LU factorizaion from this algorithm if and only if the upper $k \times k$ block for $1 \leq k \leq m$ of A is nonsingular. Or else you can run into a divide by zero error.

21 Pivoting

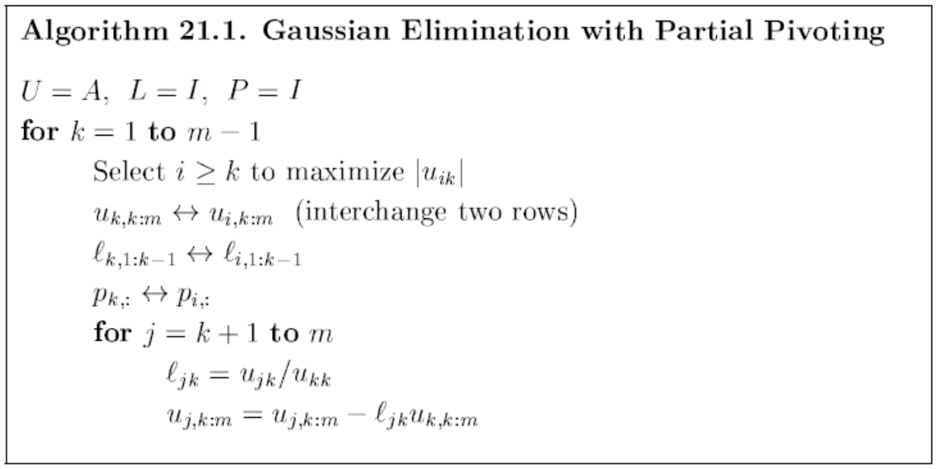
In **partial pivoting**, we follow the same algorithm as Gaussian Elimination, yet we perform row interchanges at each step so that in the case A is nonsingular, you are guaranteed to run to completion. This is still $\sim \frac{2}{3}m^3$ flops, and the algorithm is displayed in Figure 16. It results in a factorization $PA = LU$ where P is a permutation matrix.



Algorithm 20.1. Gaussian Elimination without Pivoting

```
$U = A,\ L = I$
for $k = 1$ to $m - 1$
    for $j = k + 1$ to $m$
        $\ell_{jk} = u_{jk}/u_{kk}$
        $u_{j,k:m} = u_{j,k:m} - \ell_{jk}u_{k,k:m}$
```

Figure 15: Gaussian elimination without pivoting algorithm.



Algorithm 21.1. Gaussian Elimination with Partial Pivoting

```
$U = A,\ L = I,\ P = I$
for $k = 1$ to $m - 1$
    Select $i \geq k$ to maximize $|u_{ik}|$
    $u_{k,k:m} \leftrightarrow u_{i,k:m}$   (interchange two rows)
    $\ell_{k,1:k-1} \leftrightarrow \ell_{i,1:k-1}$
    $p_{k,:} \leftrightarrow p_{i,:}$
    for $j = k + 1$ to $m$
        $\ell_{jk} = u_{jk}/u_{kk}$
        $u_{j,k:m} = u_{j,k:m} - \ell_{jk}u_{k,k:m}$
```

Figure 16: Gaussian elimination with partial pivoting algorithm.

In **complete pivoting**, instead of simply scanning the current row for the largest value in absolute value, you search the entire bottom right block, resulting in larger number of flops.

22 Stability of Gaussian Elimination

Theorem 33. Let the factorization $A = LU$ of a nonsingular matrix $A \in \mathbb{C}^{m\times m}$ be computed by Gaussian elimination without pivoting on a computer satisfying the desired axions. If A has an LU factorization, then for all sufficiently small $\epsilon_{\text{machine}}$, the factorization completes successfully in floating point arithmetic, and the computed matrices $\tilde{L}$ and $\tilde{U}$ satisfy

$$\tilde{L}\tilde{U} = A + \delta A, \quad \frac{\|\delta A\|}{\|L\|\|U\|} = O(\epsilon_{\text{machine}})$$

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

Note that we make no claims about the errors of $\tilde{L}$ or $\tilde{U}$ respectively, but their product.

Define the **growth factor** for a matrix A to be

$$\rho = \frac{\max_{i,j} |u_{ij}|}{\max_{i,j} |a_{ij}|}.$$

If ρ is of order 1, not much growth has taken place and the elimination process is stable. If larger, we expect instability.

Theorem 34. Let the factorization $PA = LU$ of a matrix $A \in \mathbb{C}^{m\times m}$ be computed by Gaussian elimination with partial pivoting. Then the computed matrices $\tilde{P}$, $\tilde{L}$, and $\tilde{U}$ satisfy

$$\tilde{L}\tilde{U} = \tilde{P}A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = O(\rho\epsilon_{\text{machine}})$$

for some $\delta A \in \mathbb{C}^{m\times m}$, where ρ is the growth factor for A. If $|l_{ij}| < 1$ for each $i > j$, implying that there are no ties in the selection of pivots in exact arithmetic, then $\tilde{P} = P$ for all sufficiently small $\epsilon_{\text{machine}}$.

So in fact this algorithm is not backwards stable unless $\rho = O(1)$.

It can be shown that in the worst-case, $\rho = 2^{m-1}$ where A is $m \times m$. However, in practice, we have at most $\rho = O(\sqrt{m})$ meaning that nearly always, we can refer to this algorithm as backward stable.

23 Cholesky Factorization

A matrix $A \in \mathbb{R}^{m\times m}$ is **symmetric** if $A = A^T$. It also satisfies $x^TAy = y^TAx$ for all $x, y \in \mathbb{R}^m$. Equivalently, $A \in \mathbb{C}^{m\times m}$ is **hermitian** if $A = A^*$, and satisfies both $x^*Ay = \overline{y^*Ax}$ and $x^*Ax \in \mathbb{R}$ for all $x, y \in \mathbb{C}^m$.

A matrix $A \in \mathbb{C}^{m\times m}$ is **hermitian positive definite**, or just **positive definite** if it is hermitian and $x^*Ax > 0$ for all $x \in \mathbb{C}^m$ nonzero. A then satisfies the following properties

- If $X \in \mathbb{C}^{m\times n}$ is full rank with $m \geq n$, then X^*AX is also hermitian positive definite.
- Any principal submatrix of A is also hermitian positive definite (of the form P^*AP for some permutation matrix).
- Every diagonal element of A is a positive real number.
- The eigenvalues of A are positive real numbers (if and only if for hermitian matrices).
- A's eigenvectors form an orthogonal set.

Cholesky factorization is a method of writing $A = R^*R$ where A is positive definite, and R is upper triangular. One step of it looks like

$$A = \begin{bmatrix} a_{11} & w^* \\ w & K \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ w/\alpha & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & K - ww^*/a_{11} \end{bmatrix} \begin{bmatrix} \alpha & w^*/\alpha \\ 0 & I \end{bmatrix}$$

where $\alpha = \sqrt{a_{11}}$. It can be shown that $K - ww^*/a_{11}$ is also positive definite, hence we inductively perform the same prodedure on that lower submatrix until we are left with $R^*IR = A$ or $A = R^*R$.

Theorem 35. Every hermitian positive definite matrix $A \in \mathbb{C}^{m\times m}$ has a unique Cholesky factorization.

The full algorithm is displayed in Figure 17. In total, it requires $\sim \frac{1}{3}m^3$ flops.

Theorem 36. Let $A \in \mathbb{C}^{m\times m}$ be hermitian positive definite, and let a Cholesky factorization of A be computed by the algorithm in Figure 17 on a computer satisfying the assumed axioms. For sufficiently small $\epsilon_{\text{machine}}$, this

```
**Algorithm 23.1. Cholesky Factorization**
$R = A$
for $k = 1$ to $m$
    for $j = k+1$ to $m$
        $R_{j,j:m} = R_{j,j:m} - R_{k,j:m}\overline{R}_{kj}/R_{kk}$
    $R_{k,k:m} = R_{k,k:m}/\sqrt{R_{kk}}$
```

Figure 17: Algorithm for Cholesky Factorization.

process is guaranteed to run to completion generating a computed factor $\tilde{R}$ that satisfies

$$\tilde{R}^*\tilde{R} = A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = O(\epsilon_{\text{machine}})$$

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

Cholesky can then be used to solve $Ax = b$, and that is backward stable. It involves $R^*Rx = b$, solving $R^*y = b$ for y, then $Rx = y$ for x.

Theorem 37. The solution of hermitian positive definite systems $Ax = b$ via Cholesky factorization is backward stable, generating a computed solution $\tilde{x}$ that satisfies

$$(A + \Delta A)\tilde{x} = b, \quad \frac{\|\Delta A\|}{\|A\|} = O(\epsilon_{\text{machine}})$$

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

24 Iterative Methods

Suppose we wish to solve the matrix equation $Ax = b$. We perform matrix-splitting, writing $A = M - N$ where M is invertible, and then we can reformulate the problem as

$$x = M^{-1}Nx + M^{-1}b = Tx + c.$$

We then have an initial guess x_0, and generate the sequence

$$x^{(n)} = Tx^{(n-1)} + c.$$

We know that if x is a solution to $Ax = b$, then $x = Tx + c$. And

$$x - x^{(n)} = Tx - Tx^{(n-1)} = T(x - x^{(n-1)}) = T^n(x - x^{(0)}),$$

so $x^{(n)} \to x$ if and only if $T^n \to 0$. One sufficient condition to this is $\|T\| < 1$ as $\|T^n\| \le \|T\|^n \to 0$.

Define the **spectral radius** of a matrix T to be the largest eigenvalue of T in absolute value, denoted $\rho(T)$.

Theorem 38. *$T^n \to 0$ as $n \to \infty$ if and only if $\rho(T) < 1$.*

Assume $A \in \mathbb{C}^{m\times m}$ is a matrix with nonzero diagonals. Split A into $D + L + U$ where D is the diagonal component, L is the below-diagonal, and U is the above-diagonal.

In the **Jacobi Method**, we split $A = D - (-L - U)$, and then achieve the form

$$x^{(n)} = -D^{-1}(L + U)x^{(n-1)} + D^{-1}b = T_J x^{(n-1)} + c.$$

Note that D^{-1} is simply the inverted diagonals. Rewriting this component form we see

$$x_i^{(n)} = \frac{1}{a_{ii}}\left(-\sum_{j\neq i} A_{ij}x_j^{(n-1)} + b_i\right).$$

In the **Gauss-Seidel (G.S.) Method**, we split $A = L + D - (-U)$, and then achieve the form

$$\begin{aligned} x^{(n)} &= -(D + L)^{-1}(U)x^{(n-1)} + (D + L)^{-1}b \\ &= T_{GS}x^{(n-1)} + c \\ &= -D^{-1}Lx^{(n)} - D^{-1}Ux^{(n-1)} + D^{-1}b. \end{aligned}$$

Using the last equality and writing this component form we see

$$x_i^{(n)} = \frac{1}{a_{ii}}\left(-\sum_{j<i} A_{ij}x_j^{(n)} - \sum_{j>i} A_{ij}x_j^{(n-1)} + b_i\right)$$

where we must first solve for $x_1^{(n)}, x_2^{(n)}, \ldots, x_{i-1}^{(n)}$ before we can solve for $x_i^{(n)}$, but then we are using updated values in each calculation.

In the **Successive Over Relaxation (SOR) Method**, we use the same splitting as G.S., but introduce a parameter $\omega > 0$, to achieve

$$\begin{aligned} x^{(n)} &= (D+\omega L)^{-1}(D-\omega(D+U))x^{(n-1)} + \omega(D+\omega L)^{-1}b \\ &= T_{SOR}x^{(n-1)} + c \\ &= \omega D^{-1}(-Lx^{(n)} - Ux^{(n-1)} + b) + (1-\omega)x^{(n-1)}. \end{aligned}$$

Or component-wise,

$$x_i^{(n)} = \frac{\omega}{a_{ii}}\left(-\sum_{j<i} A_{ij}x_j^{(n)} - \sum_{j>i} A_{ij}x_j^{(n-1)} + b_i\right) + (1-\omega)x_i^{(n-1)}.$$

We call a matrix $A \in \mathbb{C}^{m\times m}$ **strictly row diagonal dominant (SDD)** if for all $1 \le i \le m$,

$$|A_{ii}| > \sum_{j\neq i} |A_{ij}|.$$

We call A **weakly row diagonally dominant (WDD)** if for all $1 \le i \le m$,

$$|A_{ii}| \ge \sum_{j\neq i} |A_{ij}|,$$

and for at least one i,

$$|A_{ii}| > \sum_{j\neq i} |A_{ij}|.$$

Theorem 39. Gershgorin's Theorem: Consider $A \in \mathbb{C}^{m\times m}$. Any eigenvalue of A is located in one of the closed disks of the complex plane centered at A_{ii} with radius $r_i = \sum_{j\neq i} |A_{ij}|$.

Theorem 40. Strictly diagonally dominant matrices are invertible. (Proof follows from Gershgorin)

Theorem 41. If A is SDD by rows, then Jacobi and Gauss-Seidel methods are convergent.

Theorem 42.
1. If A is symmetric positive definite, then the SOR method is convergent if and only if $0 < \omega < 2$.
2. If A is SDD by rows, then the SOR method is convergent if $0 < \omega \le 1$.

3. If A and $2D-A$ are symmetric positive definite, then the Jacobi method is convergent.

4. If A is SPD, then the Gauss-Seidel method is convergent.

An $m \times m$ matrix A is **irreducible** if there is no permutation matrix P such that

$$P^T AP = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}.$$

A **directed graph** is a finite collection of nodes connected by a finite collection of directed edges. For any $m \times m$ matrix A, we define it's **adjacency graph** $G(A)$ to be the graph with m nodes, and an edge from i to j iff $A_{ij} \neq 0$. A graph is **strongly connected** if for any two nodes, there exists a path from one to the other and visa-versa.

Theorem 43. A matrix A is irreducible if and only if $G(A)$ is strongly connected.

Theorem 44. If A is irreducible and WDD, then A is nonsingular.

Theorem 45. If A is symmetric, WDD, irreducible, and has $A_{ii} > 0$ for each i, then A is symmetric positive definite.

Theorem 46. Let A be an $m \times m$ irreducible matrix. If an eigenvalue of A lies on the boundary of one of the m Gershgorin disks of A, then it also lies on the boundary of all of the other Gershgorin disks.

Theorem 47. If A is irreducible and WDD, then both Jacobi and Gauss-Seidel converge for any initial guess and $\rho(T_{GS}) < \rho(T_J) < 1$.

25 Solving Nonlinear Equations

In general, we wish to find solutions of $f(x) = 0$.

A sequence of iterates (x_n) is said to **converge** to α with order $p \geq 1$ if

$$|\alpha - x_{n+1}| \leq C|\alpha - x_n|^p, \quad n \geq 0$$

for some constant $C > 0$. When $p = 1$, we call it **linear convergence**, in which case we require $C < 1$, which we call the **rate of convergence**.

In the **Bisection Algorithm**, we assume f is continuous on $[a, b]$, and that $f(a) \cdot f(b) < 0$ (signs differ). The algorithm then goes as follows

1. Define $c := (a+b)/2$.
2. If $b - c \leq \epsilon$, then return c.
3. If $f(b) \cdot f(c) < 0$, then set $a = c$, otherwise set $b = c$ and return to step 1.

Since we cut the interval in half each time, this is a linear method with rate of convergence $\frac{1}{2}$. In order to reduce the initial interval (increase the accuracy) by 10^{-d}, we require $n \approx \frac{d}{\log_{10}(2)} \approx \frac{d}{0.3}$ iterations.

In **Newton's Method**, we assume f is once differentiable everywhere, begin with some initial guess x_0, and then perform the iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

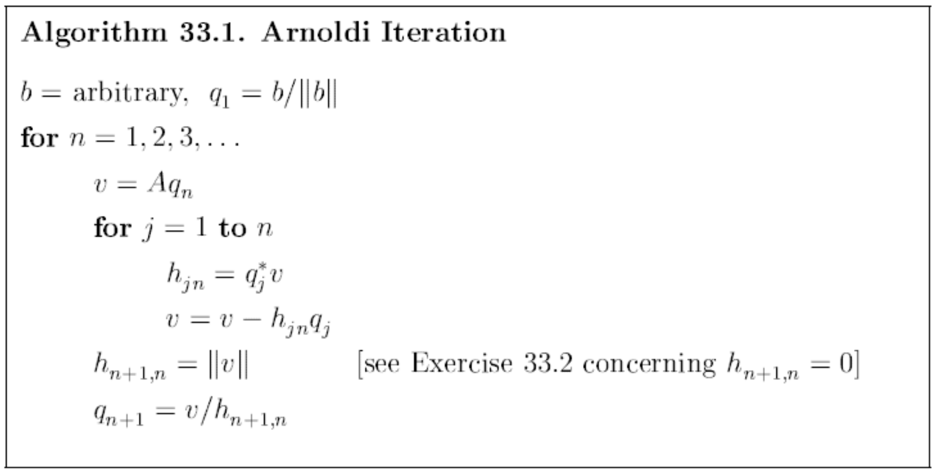
Using a Taylor Series analysis, we can show that if this method converges, it does so quadratically.

Theorem 48. Assume that $f \in C^2$ in some neighborhood of a root α of f and that $f'(\alpha) \neq 0$. Then given x_0 sufficiently close to α, Newton's method converges on α with

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{(\alpha - x_n)^2} = -\frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}.$$

Some facts about **fixed point iteration**, solving $x = g(x)$ with $x_{n+1} = g(x_n)$:

1. If g is continuous on $[a,b]$ and $g([a,b]) \subset [a,b]$, then $x = g(x)$ has at least one solution on $[a,b]$.
2. If g is also a contraction on $[a,b]$ with $|g(x) - g(y)| \leq \lambda|x-y|$ for $\lambda \in (0,1)$, then there is a unique solution on $[a,b]$, iteration will converge to it given any $x_0 \in [a,b]$, and $|\alpha - x_n| \leq \frac{\lambda^n}{1-\lambda}|x_1 - x_0|$.
3. If in addition, g is differentiable on $[a,b]$, then we have $\lambda = \max_{x\in[a,b]} |g'(x)|$.



Algorithm 33.1. Arnoldi Iteration

```
$b =$ arbitrary,  $q_1 = b/\|b\|$
for $n = 1, 2, 3, \dots$
    $v = Aq_n$
    for $j = 1$ to $n$
        $h_{jn} = q_j^* v$
        $v = v - h_{jn} q_j$
    $h_{n+1,n} = \|v\|$        [see Exercise 33.2 concerning $h_{n+1,n} = 0$]
    $q_{n+1} = v/h_{n+1,n}$
```

Figure 18: Algorithm for Arnoldi Iteration.

26 Krylov Subspace Methods

Suppose we are trying to solve $Ax = b$ where $A \in \mathbb{C}^{m\times m}$ is nonsingular and $b \in \mathbb{C}^m$.

We define **Kyrlov subspaces** as

$$\mathcal{K}_n = \langle b, Ab, \dots, A^{n-1}b\rangle$$

where the angle brackets denote span.

An **upper Hessenberg matrix** is a matrix that has zeros below the first subdiagonal, or A is upper Hessenberg if $a_{ij} = 0$ for $i > j + 1$.

Theorem 49. Given an $m \times m$ matrix A, there exists a unitary matrix Q such that $Q^*AQ = H$ where H is upper Hessenberg. And if A is real, then Q and H are also real.

One way of factoring a matrix into upper Hessenberg form is through **Arnoldi Iteration** which utilizes Gram-Schmidt. The algorithm is displayed in Figure 18. The algorithm essentially projects onto consecutive Krylov subspaces.