Chapter III

Numerical Linear Algebra

Many problems in mathematics are linear: for example, polynomial regression and differential equations. Numerical methods for such applications invariably result in (finite-dimensional) linear systems that must be solved numerically on a computer: the dimensions of the problems are often in the 1000s, millions, or even billions. One would certainly not want to tackle that with Gaussian elimination by hand! In this chapter we discuss algorithms, and in particular matrix factorisations, that are computed using floating point operations. We also introduce some basic applications.

In particular we discuss:

1. III.1 Structured Matrices: we discuss special structured matrices such as triangular and tridiagonal.
2. III.2 LU and PLU Factorisations: we see that Gaussian elimination can be recast as computing a factorisation of a square matrix as a product of a lower and upper triangular matrix, potentially with a permutation matrix corresponding to the case where row pivoting is required.
3. III.3 Cholesky Factorisation: In the special case where the matrix is symmetric positive definite the LU factorisation has a special form. Hidden in this is an algorithm to prove positive definiteness.
4. III.4 Orthogonal Matrices: we discuss different types of orthogonal matrices, which will be used to simplify rectangular least squares problems.
5. III.5 QR Factorisation: we introduce an algorithm to compute a factorisation of a rectangular matrix as a product of an orthogonal and upper triangular matrix, thereby solving least squares problems.

Here we are constructing underlying computational tools that are important in applications, such as solving differential equations and data regression, which we discuss later.

III.1 Structured Matrices

We have seen how algebraic operations (`+`, `-`, `*`, `/`) are defined exactly in terms of rounding ($\oplus$, $\ominus$, $\otimes$, $\oslash$) for floating point numbers. Now we see how this allows us to do (approximate) linear algebra operations on matrices.

A matrix can be stored in different formats, in particular it is important for large scale simulations that we take advantage of *sparsity*: if we know a matrix has entries that are guaranteed to be zero we can implement faster algorithms. We shall see that this comes up naturally in numerical methods for solving differential equations.

In particular, we will discuss some basic types of structure in matrices:

1. *Dense*: This can be considered unstructured, where we need to store all entries in a vector or matrix. Matrix-vector multiplication reduces directly to standard algebraic operations. Solving linear systems with dense matrices will be discussed later.
2. *Triangular*: If a matrix is upper or lower triangular, multiplication requires roughly half the number of operations. Crucially, we can apply the inverse of a triangular matrix using forward- or back-substitution.
3. *Banded*: If a matrix is zero apart from entries a fixed distance from the diagonal it is called banded and matrix-vector multiplication has a lower *complexity*: the number of operations scales linearly with the dimension (instead of quadratically). We discuss three cases: diagonal, tridiagonal and bidiagonal matrices.

Remark For those who took the first half of the module, there was an important emphasis on working with *linear operators* rather than *matrices*. That is, there was an emphasis on basis-independent mathematical techniques, which is critical for extension of results to infinite-dimensional spaces (which might not have a complete basis). However, in terms of practical computation we need to work with some representation of an operator and the most natural is a matrix. And indeed we will see in the next section how infinite-dimensional differential equations can be solved by reduction to finite-dimensional matrices. (Restricting attention to matrices is also important as some of the students have not taken the first half of the module.)

III.1.1 Dense matrices

A basic operation is matrix-vector multiplication. For a field $\mathbb{F}$ (typically $\mathbb{R}$ or $\mathbb{C}$, or this can be relaxed to be a ring), consider a matrix and vector whose entries are in $\mathbb{F}$:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1 | \cdots | \boldsymbol{a}_n \end{bmatrix} \in \mathbb{F}^{m\times n}, \qquad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{F}^n.$$

where $\boldsymbol{a}_j = A\boldsymbol{e}_j \in \mathbb{F}^m$ are the columns of A. Recall the usual definition of matrix multiplication:

$$A\boldsymbol{x} := \begin{bmatrix} \sum_{j=1}^n a_{1j}x_j \\ \vdots \\ \sum_{j=1}^n a_{mj}x_j \end{bmatrix}.$$

When we are working with floating point numbers $A \in F^{m\times n}$ we obtain an approximation:

$$A\boldsymbol{x} \approx \begin{bmatrix} \bigoplus_{j=1}^n (a_{1j} \otimes x_j) \\ \vdots \\ \bigoplus_{j=1}^n (a_{mj} \otimes x_j) \end{bmatrix}.$$

This actually encodes an algorithm for computing the entries.

This algorithm uses $O(mn)$ floating point operations (see the appendix if you are unaware of Big-O notation, here our complexities are implicitly taken to be when m or n tends to ∞): each of the m entries consists of n multiplications and $n-1$ additions, hence we have a total of $2n-1 = O(n)$ operations per row for a total of $m(2n-1) = O(mn)$ operations. For a square matrix this is $O(n^2)$ operations which we call *quadratic complexity*. In the problem sheet we see how the floating point error can be bounded in terms of norms, thus reducing the problem to a purely mathematical concept.

Sometimes there are multiple ways of implementing numerical algorithms. We have an alternative formula where we multiply by columns:

$$A\boldsymbol{x} = x_1\boldsymbol{a}_1 + \cdots + x_n\boldsymbol{a}_n.$$

The floating point formula for this is exactly the same as the previous algorithm and the number of operations is the same. Just the order of operations has changed. Suprisingly, this latter version is significantly faster.

Remark Floating point operations are sometimes called FLOPs, which are a standard measurement of speed of CPUs. However, FLOP sometimes uses an alternative definitions that combines an addition and multiplication as a single FLOP. In the lab we give an example showing that counting the precise number of operations is somewhat of a fools errand: algorithms such as the two approaches for matrix multiplication with the exact same number of operations can have wildly different speeds. We will therefore only be concerned with *complexity*; the asymptotic growth (Big-O) of operations as $n \to \infty$, in which case the difference between FLOPs and operations is immaterial.

III.1.2 Triangular matrices

The simplest sparsity case is being triangular: where all entries above or below the diagonal are zero. We consider upper and lower triangular matrices:

$$U = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ & \ddots & \vdots \\ & & u_{nn} \end{bmatrix}, \qquad L = \begin{bmatrix} \ell_{11} & & \\ \vdots & \ddots & \\ \ell_{n1} & \cdots & \ell_{nn} \end{bmatrix}.$$

Matrix multiplication can be modified to take advantage of the zero pattern of the matrix. Eg., if $L \in \mathbb{F}^{n\times n}$ is lower triangular we have:

$$L\boldsymbol{x} = \begin{bmatrix} \ell_{1,1}x_1 \\ \sum_{j=1}^2 \ell_{2j}x_j \\ \vdots \\ \sum_{j=1}^n \ell_{nj}x_j \end{bmatrix}.$$

When implemented in floating point this uses roughly half the number of multiplications: $1 + 2 + \ldots + n = n(n+1)/2$ multiplications. (It is also about twice as fast in practice.) The complexity is still quadratic: $O(n^2)$ operations.

Triangularity allows us to also invert systems using forward- or back-substitution. In particular if $\boldsymbol{x}$ solves $L\boldsymbol{x} = \boldsymbol{b}$ then we have:

$$x_k = \frac{b_k - \sum_{j=1}^{k-1} \ell_{kj}x_j}{\ell_{kk}}$$

Thus we can compute $x_1, x_2, \ldots, x_n$ in sequence.

III.1.3 Banded matrices

A *banded matrix* is zero off a prescribed number of diagonals. We call the number of (potentially) non-zero diagonals the *bandwidths*:

Definition 12 (bandwidths)**.** A matrix A has *lower-bandwidth* l if $a_{kj} = 0$ for all $k-j > l$ and *upper-bandwidth* u if $a_{kj} = 0$ for all $j - k > u$. We say that it has *strictly lower-bandwidth* l if it has lower-bandwidth l and there exists a j such that $a_{j+l,j} \neq 0$. We say that it has *strictly upper-bandwidth* u if it has upper-bandwidth u and there exists a k such that $a_{k,k+u} \neq 0$.

A square banded matrix has the sparsity pattern:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1,u+1} & & & \\ \vdots & a_{22} & \ddots & a_{2,u+2} & & \\ a_{1+l,1} & \ddots & \ddots & \ddots & \ddots & \\ & a_{2+l,2} & \ddots & \ddots & \ddots & a_{n-u,n} \\ & & \ddots & \ddots & \ddots & \vdots \\ & & & a_{n,n-l} & \cdots & a_{nn} \end{bmatrix}$$

A banded matrix has better complexity for matrix multiplication and solving linear systems: we can multiply square banded matrices in linear complexity: $O(n)$ operations. We consider two cases in particular (in addition to diagonal): bidiagonal and tridiagonal.

Definition 13 (Bidiagonal)**.** If a square matrix has bandwidths $(l, u) = (1, 0)$ it is *lower-bidiagonal* and if it has bandwidths $(l, u) = (0, 1)$ it is *upper-bidiagonal*.

For example, if

$$L = \begin{bmatrix} \ell_{11} & & & \\ \ell_{21} & \ell_{22} & & \\ & \ddots & \ddots & \\ & & \ell_{n,n-1} & \ell_{nn} \end{bmatrix}$$

then lower-bidiagonal multiplication becomes

$$L\boldsymbol{x} = \begin{bmatrix} \ell_{1,1}x_1 \\ \ell_{21}x_1 + \ell_{22}x_2 \\ \vdots \\ \ell_{n,n-1}x_{n-1} + \ell_{nn}x_n \end{bmatrix}.$$

This requires $O(1)$ operations per row (at most 2 multiplications and 1 addition) and hence the total is only $O(n)$ operations. A bidiagonal matrix is always triangular and we can also invert in $O(n)$ operations: if $L\boldsymbol{x} = \boldsymbol{b}$ then $x_1 = b_1/\ell_{11}$ and for $k = 2, \ldots, n$ we can compute

$$x_k = \frac{b_k - \ell_{k-1,k}x_{k-1}}{\ell_{kk}}.$$

Definition 14 (Tridiagonal)**.** If a square matrix has bandwidths $l = u = 1$ it is *tridiagonal*.

For example,

$$A = \begin{bmatrix} a_{11} & a_{12} & & & \\ a_{21} & a_{22} & a_{23} & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ & & & a_{n,n-1} & a_{nn} \end{bmatrix}$$

is tridiagonal. Matrix multiplication is clearly $O(n)$ operations: each row has $O(1)$ non-zeros and there are n rows. But so is solving linear systems, which we shall see later.

III.2 LU and PLU factorisations

One of the most fundamental problems in linear algebra is solving linear systems. For a field $\mathbb{F}$ (for us either $\mathbb{R}$ or $\mathbb{C}$), given invertible matrix $A \in \mathbb{F}^{n\times n}$ and vector $\boldsymbol{b} \in \mathbb{F}^n$, find $\boldsymbol{x} \in \mathbb{F}^n$ such that

$$A\boldsymbol{x} = \boldsymbol{b}.$$

This can of course be done via Gaussian elimination, uing row swaps (or *pivoting*) if a zero is encountered on the diagonal, which can be viewed as an algorithm that can be implemented on a computer. However, a basic observation makes the practical implementation more straightforward and easier to apply to multiple right-hand sides, and connects with fundamental aspects in matrix analysis.

In particular, Gaussian elimination is equivalent to computing an *LU factorisation*:

$$A = LU$$

where L is lower triangular and U is upper triangular. Thus if we compute L and U we can deduce

$$\boldsymbol{x} = A^{-1}\boldsymbol{b} = U^{-1}L^{-1}\boldsymbol{b}$$

where $\boldsymbol{c} = L^{-1}\boldsymbol{b}$ can be computed using forward-substitution and $U^{-1}\boldsymbol{c}$ using back-substitution.

On the other hand, Gaussian elemination with pivoting (row-swapping) is equivalent to a *PLU factorisation*:

$$A = P^\top LU$$

where P is a permutation matrix (see appendix). Thus if we can compute P, L and U we can deduce

$$\boldsymbol{x} = A^{-1}\boldsymbol{b} = U^{-1}L^{-1}P\boldsymbol{b}$$

where multiplication by P is a simple swap of entries of $\boldsymbol{b}$ and L and U are again invertible via forward- and back-substitution.

III.2.1 Outer products

In what follows we will use outer products extensively:

Definition 15 (outer product)**.** Given $\boldsymbol{x} \in \mathbb{F}^m$ and $\boldsymbol{y} \in \mathbb{F}^n$ the *outer product* is:

$$\boldsymbol{x}\boldsymbol{y}^\top := [\boldsymbol{x}y_1|\cdots|\boldsymbol{x}y_n] = \begin{bmatrix} x_1y_1 & \cdots & x_1y_n \\ \vdots & \ddots & \vdots \\ x_my_1 & \cdots & x_my_n \end{bmatrix} \in \mathbb{F}^{m\times n}.$$

Note this is equivalent to matrix-matrix multiplication if we view $\boldsymbol{x}$ as a $m \times 1$ matrix and $\boldsymbol{y}^\top$ as a $1 \times n$ matrix.

Proposition 4 (rank-1)**.** *A matrix $A \in \mathbb{F}^{m\times n}$ has rank 1 if and only if there exists $\boldsymbol{x} \in \mathbb{F}^m$ and $\boldsymbol{y} \in \mathbb{F}^n$ such that*

$$A = \boldsymbol{x}\boldsymbol{y}^\top.$$

Proof If $A = \boldsymbol{x}\boldsymbol{y}^\top$ then all columns are multiples of $\boldsymbol{x}$, that is the column span has dimension 1. On the other hand, if A has rank-1 then its columns span a one-dimensional subspace: there exists $\boldsymbol{x} \in \mathbb{F}^m$

$$\operatorname{span}(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n) = \{c\boldsymbol{x} : c \in \mathbb{F}\}.$$

Thus there exist $y_k \in \mathbb{F}$ such that $\boldsymbol{a}_k = y_k \boldsymbol{x}$ and we have

$$A = \boldsymbol{x} \underbrace{\begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix}}_{\boldsymbol{y}^\top}.$$

■

III.2.2 LU factorisation

Gaussian elimination can be interpreted as an LU factorisation. Write a matrix $A \in \mathbb{F}^{n\times n}$ as follows:

$$A = \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ \boldsymbol{v} & K \end{bmatrix}$$

where $\alpha = a_{11}$, $\boldsymbol{v} = A[2:n,1]$ and $\boldsymbol{w} = A[1,2:n]$ (that is, $\boldsymbol{v} \in \mathbb{F}^{n-1}$ is a vector whose entries are the 2nd through last row of the first column of A whilst $\boldsymbol{w} \in \mathbb{F}^{n-1}$ is a vector containing the 2nd through last column of the first row of A). Gaussian elimination consists of taking the first row, dividing by α and subtracting from all other rows. That is equivalent to multiplying by a lower triangular matrix:

$$\underbrace{\begin{bmatrix} 1 & \\ -\boldsymbol{v}/\alpha & I \end{bmatrix}}_{L_1^{-1}} A = \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & K - \boldsymbol{v}\boldsymbol{w}^\top/\alpha \end{bmatrix}$$

where $A_2 := K - \boldsymbol{v}\boldsymbol{w}^\top/\alpha$ happens to be a rank-1 perturbation of K. We can write this another way:

$$A = \underbrace{\begin{bmatrix} 1 & \\ \boldsymbol{v}/\alpha & I \end{bmatrix}}_{L_1} \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & A_2 \end{bmatrix}$$

Now assume we continue this process and manage to deduce an LU factorisation $A_2 = \tilde{L}\tilde{U}$. Then

$$A = L_1 \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & \tilde{L}\tilde{U} \end{bmatrix} = \underbrace{L_1 \begin{bmatrix} 1 & \\ & \tilde{L} \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & \tilde{U} \end{bmatrix}}_{U}$$

Note we can multiply through to find

$$L = \begin{bmatrix} 1 & \\ \boldsymbol{v}/\alpha & \tilde{L} \end{bmatrix}.$$

Noting that if $A \in \mathbb{F}^{1\times 1}$ then it has a trivial LU factorisation we can use the above construction to proceed recursively until we arrive at the trivial case.

Rather than a recursive definition, we can view the above as an inductive procedure:

$$\begin{aligned}
A &= L_1 \begin{bmatrix} \alpha_1 & \boldsymbol{w}_1^\top \\ & A_2 \end{bmatrix} = L_1 \begin{bmatrix} \alpha_1 & \boldsymbol{w}_1^\top \\ & L_2 \begin{bmatrix} \alpha_2 & \boldsymbol{w}_2^\top \\ & A_3 \end{bmatrix} \end{bmatrix} \\
&= L_1 \begin{bmatrix} 1 & \\ & L_2 \end{bmatrix} \begin{bmatrix} \alpha_1 & \boldsymbol{w}_1^\top \\ & \begin{bmatrix} \alpha_2 & \boldsymbol{w}_2^\top \\ & L_3 \begin{bmatrix} \alpha_3 & \boldsymbol{w}_3^\top \\ & A_4 \end{bmatrix} \end{bmatrix} \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} 1 & \\ \boldsymbol{v}_1/\alpha_1 & \begin{bmatrix} 1 & \\ \boldsymbol{v}_2/\alpha_2 & \begin{bmatrix} 1 & \\ \boldsymbol{v}_3/\alpha_3 & \ddots \end{bmatrix} \end{bmatrix} \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \alpha_1 & \boldsymbol{w}_1^\top \\ & \begin{bmatrix} \alpha_2 & \boldsymbol{w}_2^\top \\ & \begin{bmatrix} \alpha_3 & \boldsymbol{w}_3^\top \\ & \ddots \end{bmatrix} \end{bmatrix} \end{bmatrix}}_{U}.
\end{aligned}$$

We can see this procedure clearer in the following example.

Example 12 (LU by-hand)**.** Consider the matrix

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 8 \\ 1 & 4 & 9 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & & \\ 2 & 1 & \\ 1 & & 1 \end{bmatrix}}_{L_1} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 6 \\ 0 & 3 & 8 \end{bmatrix}$$

In more detail, for $\alpha_1 := a_{11} = 1$, $\boldsymbol{v}_1 := A[2:3,1] = [2,1]^\top$, $\boldsymbol{w}_1 = A[1,2:3] = [1,1]^\top$ and

$$K_1 := A[2:3,2:3] = \begin{bmatrix} 4 & 8 \\ 4 & 9 \end{bmatrix}$$

we have

$$A_2 := K_1 - \boldsymbol{v}_1\boldsymbol{w}_1^\top/\alpha_1 = \begin{bmatrix} 4 & 8 \\ 4 & 9 \end{bmatrix} - \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 6 \\ 3 & 8 \end{bmatrix}.$$

We then repeat the process and determine (with $\alpha_2 := A_2[1,1] = 2$, $\boldsymbol{v}_2 := A_2[2:2,1] = [3]$, $\boldsymbol{w}_2 := A_2[1,2:2] = [6]$ and $K_2 := A_2[2:2,2:2] = [8]$):

$$A_2 = \begin{bmatrix} 2 & 6 \\ 3 & 8 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \\ 3/2 & 1 \end{bmatrix}}_{L_2} \begin{bmatrix} 2 & 6 \\ & -1 \end{bmatrix}$$

The last "matrix" is 1 x 1 so we get the trivial decomposition:

$$A_3 := K_2 - \boldsymbol{v}_2\boldsymbol{w}_2^\top/\alpha_2 = [-1] = \underbrace{[1]}_{L_3}[-1]$$

Putting everything together and placing the j-th column of L_j inside the j-th column of L we have

$$A = \underbrace{\begin{bmatrix} 1 & & \\ 2 & 1 & \\ 1 & 3/2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ & 2 & 6 \\ & & -1 \end{bmatrix}}_{U}$$

III.2.3 PLU factorisation

We learned in first year linear algebra that if a diagonal entry is zero when doing Gaussian elimination one has to *row pivot*. For stability, in implementation one may wish to pivot even if the diagonal entry is nonzero: swap the largest in magnitude entry for the entry on the diagonal turns out to be significantly more stable than standard LU.

This is equivalent to a PLU decomposition. Here we use a *permutation matrix*, whose action on a vector permutes its entries, as discussed in the appendix. That is, consider a permutation which we identify with a vector $\sigma = [\sigma_1, \ldots, \sigma_n]$ containing the integers $1, \ldots, n$ exactly once. The permutation operator represents the action of permuting the entries in a vector:

$$P_\sigma(\boldsymbol{v}) := \boldsymbol{v}[\sigma] = \begin{bmatrix} v_{\sigma_1} \\ \vdots \\ v_{\sigma_n} \end{bmatrix}$$

This is a linear operator, and hence we can identify it with a *permutation matrix* $P_\sigma \in \mathbb{R}^{n \times n}$ (more precisely the entries of P_σ are either 1 or 0). Importantly, products of permutation matrices are also permutation matrices and permutation matrices are orthogonal, that is, $P_\sigma^{-1} = P_\sigma^\top$.

Theorem 5 (PLU)**.** *A matrix $A \in \mathbb{C}^{n \times n}$ is invertible if and only if it has a PLU decomposition:*

$$A = P^\top L U$$

where the diagonal of L are all equal to 1 and the diagonal of U are all non-zero, and P is a permutation matrix.

Proof

If we have a PLU decomposition of this form then L and U are invertible and hence the inverse is simply $A^{-1} = U^{-1} L^{-1} P$. Hence we consider the orther direction.

If $A \in \mathbb{C}^{1 \times 1}$ we trivially have an LU decomposition $A = [1] * [a_{11}]$ as all 1×1 matrices are triangular. We now proceed by induction: assume all invertible matrices of lower dimension have a PLU factorisation. As A is invertible not all entries in the first column are zero. Therefore there exists a permutation P_1 so that $\alpha := (P_1 A)[1,1] \neq 0$. Hence we write

$$P_1 A = \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ \boldsymbol{v} & K \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \\ \boldsymbol{v}/\alpha & I \end{bmatrix}}_{L_1} \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & K - \boldsymbol{v}\boldsymbol{w}^\top/\alpha \end{bmatrix}$$

We deduce that $A_2 := K - \boldsymbol{v}\boldsymbol{w}^\top/\alpha$ is invertible because A and L_1 are invertible (Exercise).

By assumption we can write $A_2 = \tilde{P}^\top \tilde{L}\tilde{U}$. Thus we have:

$$\begin{aligned}
\underbrace{\begin{bmatrix} 1 & \\ & \tilde{P} \end{bmatrix} P_1}_{P} A &= \begin{bmatrix} 1 & \\ & \tilde{P} \end{bmatrix} L_1 \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & A_2 \end{bmatrix} = \begin{bmatrix} 1 & \\ & \tilde{P} \end{bmatrix} L_1 \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & \tilde{P}^\top \tilde{L}\tilde{U} \end{bmatrix} \\
&= \begin{bmatrix} 1 & \\ \tilde{P}\boldsymbol{v}/\alpha & \tilde{P} \end{bmatrix} \begin{bmatrix} 1 & \\ & \tilde{P}^\top \tilde{L} \end{bmatrix} \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & \tilde{U} \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} 1 & \\ \tilde{P}\boldsymbol{v}/\alpha & \tilde{L} \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & \tilde{U} \end{bmatrix}}_{U}.
\end{aligned}$$

■

For stability one uses the permutation that always puts the largest in magnitude entry in the top row, eg., by a simple swap with the row corresponding to the diagonal. One could try to justify this by considering floating point rounding, but actually there is no guaranteed this will produce accurate results and indeed in the lab we given an example of a 'bad matrix′ where large errors are still produced. However, it is observed in practice that the probability of encountering a 'bad matrix′ is extremely small. The biggest open problem in numerical linear algebra is proving this observation rigorously.

Again, the above recursive proof encodes an inductive procedure, which we see in the following example.

Example 13 (PLU by-hand)**.** Consider the matrix

$$A = \begin{bmatrix} 0 & 2 & 1 \\ 2 & 6 & 2 \\ 1 & -1 & 5 \end{bmatrix}$$

The largest entry in the first column is 2 in the second row, hence we swap these rows then factor:

$$\underbrace{\begin{bmatrix} 0 & 1 & \\ 1 & 0 & \\ & & 1 \end{bmatrix}}_{P_1} A = \begin{bmatrix} 2 & 6 & 2 \\ 0 & 2 & 1 \\ 1 & -1 & 5 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & & \\ 0 & 1 & \\ 1/2 & 0 & 1 \end{bmatrix}}_{L_1} \begin{bmatrix} 2 & 6 & 2 \\ 0 & 2 & 1 \\ 0 & -4 & 4 \end{bmatrix}$$

Even though

$$A_2 := \begin{bmatrix} 2 & 1 \\ -4 & 4 \end{bmatrix}$$

is non-singular, we still permute the largest entry to the diagonal (this is helpful on a computer for stability). So we permute again to get:

$$\underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}_{P_2} A_2 = \begin{bmatrix} -4 & 4 \\ 2 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \\ -1/2 & 1 \end{bmatrix}}_{L_2} = \underbrace{\begin{bmatrix} -4 & 4 \\ & 3 \end{bmatrix}}_{U_2}$$

Putting it together we have

$$\begin{aligned} A &= P_1^\top L_1 \begin{bmatrix} \alpha_1 & \boldsymbol{w}_1^\top \\ & A_2 \end{bmatrix} = P_1^\top L_1 \begin{bmatrix} \alpha_1 & \boldsymbol{w}_1^\top \\ & P_2^\top L_2 U_2 \end{bmatrix} \\ &= P_1^\top \begin{bmatrix} 1 & \\ \boldsymbol{v}_1/\alpha_1 & I \end{bmatrix} \begin{bmatrix} 1 & \\ & P_2^\top L_2 \end{bmatrix} \begin{bmatrix} \alpha_1 & \boldsymbol{w}_1^\top \\ & U_2 \end{bmatrix} = P_1^\top \begin{bmatrix} 1 & \\ & P_2^\top \end{bmatrix} \begin{bmatrix} 1 & \\ P_2\boldsymbol{v}_1/\alpha_1 & L_2 \end{bmatrix} \begin{bmatrix} \alpha_1 & \boldsymbol{w}_1^\top \\ & U_2 \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{P^\top} \underbrace{\begin{bmatrix} 1 & & \\ 1/2 & 1 & \\ 0 & -1/2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 6 & 2 \\ & -4 & 4 \\ & & 3 \end{bmatrix}}_{U}. \end{aligned}$$

III.3 Cholesky factorisation

In the special case where A is a real square *symmetric positive definite* (SPD, that is $A \in \mathbb{R}^{n\times n}$ such that $A^\top = A$ and $\boldsymbol{x}^\top A\boldsymbol{x} > 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x} \neq 0$) matrix the LU factorisation has a special form called the *Cholesky factorisation*:

$$A = LL^\top,$$

i.e., $U = L^\top$, but now L does not necessarily have 1 on the diagonal. This provides an algorithmic way to *prove* that a matrix is symmetric positive definite, and is roughly twice as fast as the LU factorisation to compute.

Definition 16 (positive definite)**.** A square matrix $A \in \mathbb{R}^{n\times n}$ is *positive definite* if for all $\boldsymbol{x} \in \mathbb{R}^n, x \neq 0$ we have

$$\boldsymbol{x}^\top A \boldsymbol{x} > 0$$

First we establish some basic properties of positive definite matrices:

Proposition 5 (conjugating positive definite)**.** *If $A \in \mathbb{R}^{n\times n}$ is positive definite and $V \in \mathbb{R}^{n\times n}$ is non-singular then*

$$V^\top A V$$

is positive definite.

Proof

For all $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x} \neq 0$, define $\boldsymbol{y} = V\boldsymbol{x} \neq 0$ (since V is non-singular). Thus we have

$$\boldsymbol{x}^\top V^\top A V \boldsymbol{x} = \boldsymbol{y}^\top A \boldsymbol{y} > 0.$$

■

Proposition 6 (diag positivity)**.** *If $A \in \mathbb{R}^{n\times n}$ is positive definite then its diagonal entries are positive: $a_{kk} > 0$.*

Proof

$$a_{kk} = \boldsymbol{e}_k^\top A \boldsymbol{e}_k > 0.$$

■

Lemma 4 (subslice positive definite)**.** *If $A \in \mathbb{R}^{n\times n}$ is positive definite then $A[2:n, 2:n] \in \mathbb{R}^{(n-1)\times(n-1)}$ is also positive definite.*

Proof For all $\boldsymbol{x} \in \mathbb{R}^{n-1}$, define $\boldsymbol{y} := [0, \boldsymbol{x}]$. Then we have

$$\boldsymbol{x}^\top A[2:n, 2:n]\boldsymbol{x} = \boldsymbol{y}^\top A \boldsymbol{y} > 0.$$

■

Here is the key result:

Theorem 6 (Cholesky and SPD)**.** *A matrix A is symmetric positive definite if and only if it has a Cholesky factorisation*

$$A = LL^\top$$

where L is lower triangular with positive diagonal entries.

Proof If A has a Cholesky factorisation it is symmetric ($A^\top = (LL^\top)^\top = A$) and for $\boldsymbol{x} \neq 0$ we have

$$\boldsymbol{x}^\top A\boldsymbol{x} = (L^\top \boldsymbol{x})^\top L^\top \boldsymbol{x} = \|L^\top \boldsymbol{x}\|^2 > 0$$

where we use the fact that L is non-singular.

For the other direction we will prove it by induction, with the 1×1 case being trivial. Assume all lower dimensional symmetric positive definite matrices have Cholesky decompositions. Modifying the LU factorisation slightly we write

$$A = \begin{bmatrix} \alpha & \boldsymbol{v}^\top \\ \boldsymbol{v} & K \end{bmatrix} = \underbrace{\begin{bmatrix} \sqrt{\alpha} & \\ \frac{\boldsymbol{v}}{\sqrt{\alpha}} & I \end{bmatrix}}_{L_1} \begin{bmatrix} 1 & \\ & K - \frac{\boldsymbol{v}\boldsymbol{v}^\top}{\alpha} \end{bmatrix} \underbrace{\begin{bmatrix} \sqrt{\alpha} & \frac{\boldsymbol{v}^\top}{\sqrt{\alpha}} \\ & I \end{bmatrix}}_{L_1^\top}.$$

Note that $A_2 := K - \frac{\boldsymbol{v}\boldsymbol{v}^\top}{\alpha}$ is a subslice of $L_1^{-1} A L_1^{-\top}$, hence by combining the previous propositions is itself SPD. Thus we can write

$$A_2 = K - \frac{\boldsymbol{v}\boldsymbol{v}^\top}{\alpha} = L_2 L_2^\top$$

and hence $A = LL^\top$ for

$$L = L_1 \begin{bmatrix} 1 & \\ & L_2 \end{bmatrix} = \begin{bmatrix} \sqrt{\alpha} & \\ \frac{\boldsymbol{v}}{\sqrt{\alpha}} & L_2 \end{bmatrix}.$$

■

Example 14 (Cholesky by hand)**.** Consider the matrix

$$A = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix}$$

Then $\alpha_1 = 2$, $\boldsymbol{v}_1 = [1, 1, 1]$, and

$$A_2 = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}.$$

Continuing, we have $\alpha_2 = 3/2$, $\boldsymbol{v}_2 = [1/2, 1/2]$, and

$$A_3 = \frac{1}{2} \left(\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} - \frac{1}{3} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} \right) = \frac{1}{3} \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix}$$

Next, $\alpha_3 = 4/3$, $\boldsymbol{v}_3 = [1/3]$, and

$$A_4 = [4/3 - 3/4 * (1/3)^2] = [5/4]$$

i.e. $\alpha_4 = 5/4$.

Thus we get

$$L = \begin{bmatrix} \sqrt{\alpha_1} & & & \\ \frac{\boldsymbol{v}_1[1]}{\sqrt{\alpha_1}} & \sqrt{\alpha_2} & & \\ \frac{\boldsymbol{v}_1[2]}{\sqrt{\alpha_1}} & \frac{\boldsymbol{v}_2[1]}{\sqrt{\alpha_2}} & \sqrt{\alpha_3} & \\ \frac{\boldsymbol{v}_1[3]}{\sqrt{\alpha_1}} & \frac{\boldsymbol{v}_2[2]}{\sqrt{\alpha_2}} & \frac{\boldsymbol{v}_3[1]}{\sqrt{\alpha_3}} & \sqrt{\alpha_4} \end{bmatrix} = \begin{bmatrix} \sqrt{2} & & & \\ \frac{1}{\sqrt{2}} & \sqrt{\frac{3}{2}} & & \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{3}} & \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{12}} & \frac{\sqrt{5}}{2} \end{bmatrix}$$

III.4 Orthogonal and Unitary Matrices

PLU factorisations are an effective scheme for inverting systems, however, we saw in the lab that for very special matrices it can fail to be accurate. In the next two sections we introduce an alternative approach that is guaranteed to be stable: factorise a matrix as

$$A = QR$$

where Q is an orthogonal/unitary matrix and R is a *right-triangular matrix*, which for square matrices is another name for upper-triangular.

This factorisation is valid for rectangular matrices $A \in \mathbb{C}^{m\times n}$, where now *right-triangular* is a rectangular version of upper-triangular. For rectangular systems we can no longer solve linear systems of the form $A\boldsymbol{x} = \boldsymbol{b}$ (unless $\boldsymbol{b}$ lies in the column span of A) but instead we want to solve $A\boldsymbol{x} \approx \boldsymbol{b}$, where $\boldsymbol{x} \in \mathbb{C}^n$ and $\boldsymbol{b} \in \mathbb{C}^m$. More precisely, we can use a QR factorisation to solve *least squares* problems, find $\boldsymbol{x}$ that minimises the 2-norm:

$$\|A\boldsymbol{x} - \boldsymbol{b}\|$$

Before we discuss the computation of a QR factorisation and its role in solving least-squares problems, we introduce orthogonal and unitary matrices. In particular we will discuss reflections and rotations, which can be used to represent more general orthogonal matrices.

Definition 17 (orthogonal/unitary matrix)**.** A square real matrix is *orthogonal* if its inverse is its transpose:

$$O(n) = \{Q \in \mathbb{R}^{n\times n} : Q^\top Q = I\}$$

A square complex matrix is *unitary* if its inverse is its adjoint:

$$U(n) = \{Q \in \mathbb{C}^{n\times n} : Q^\star Q = I\}.$$

Here the adjoint is the same as the conjugate-transpose: $Q^\star := \bar{Q}^\top$.

Note that $O(n) \subset U(n)$ as for real matrices $Q^\star = Q^\top$. Because in either case $Q^{-1} = Q^\star$ we also have $QQ^\star = I$ (which for real matrices is $QQ^\top = I$). These matrices are particularly important for numerical linear algebra for a number of reasons (we'll explore these properties in the problem sheets):

1. They are norm-preserving: for any vector $\boldsymbol{x} \in \mathbb{C}^n$ and $Q \in U(n)$ we have $\|Q\boldsymbol{x}\| = \|\boldsymbol{x}\|$ where $\|\boldsymbol{x}\|^2 := \sum_{k=1}^n x_k^2$ (i.e. the 2-norm).
2. All eigenvalues have absolute value equal to 1.
3. For $Q \in O(n)$, $\det Q = \pm 1$.
4. They are trivially invertible (just take the adjoint).
5. They are generally "stable": errors due to rounding when multiplying a vector by Q are controlled.
6. They are *normal matrices*: they commute with their adjoint ($QQ^\star = QQ^\star$).
7. Both $O(n)$ and $U(n)$ are groups, in particular, they are closed under multiplication.

On a computer there are multiple ways of representing orthogonal/unitary matrices. The obvious way is to store entries as a dense matrix, however, this is very inefficient. In the appendices we have seen permutation matrices, which are a special type of orthogonal matrices where we can store only the order the entries are permuted as a vector.

More generally, we will use the group structure: represent general orthogonal/unitary matrices as products of simpler elements of the group. In partular we will use two building blocks:

1. *Rotations*: Rotations are equivalent to special orthogonal matrices $SO(2)$ and correspond to rotations in 2D.
2. *Reflections*: Reflections are elements of $U(n)$ that are defined in terms of a single unit vector $\boldsymbol{v} \in \mathbb{C}^n$ which is reflected.

We remark a related concept to orthogonal/unitary matrices are rectangular matrices with orthonormal columns, e.g.

$$U = [\boldsymbol{u}_1 | \cdots | \boldsymbol{u}_n] \in \mathbb{C}^{m \times n}$$

where $m \geq n$ such that $U^\star U = I_n$ (the $n \times n$ identity matrix). In the case where $m > n$ we must have $UU^\star \neq I_m$ as the rank of U is $n < m$.

III.4.1 Rotations

We begin with a general definition:

Definition 18 (Special Orthogonal and Rotations). *Special Orthogonal Matrices* are

$$SO(n) := \{Q \in O(n) | \det Q = 1\}$$

And (simple) *rotations* are $SO(2)$.

In what follows we use the following for writing the angle of a vector:

Definition 19 (two-arg arctan). The two-argument arctan function gives the angle θ through the point $[a, b]^\top$, i.e.,

$$\sqrt{a^2 + b^2} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}.$$

It can be defined in terms of the standard arctan as follows:

$$\operatorname{atan}(b, a) := \begin{cases} \operatorname{atan}\frac{b}{a} & a > 0 \\ \operatorname{atan}\frac{b}{a} + \pi & a < 0 \text{ and } b > 0 \\ \operatorname{atan}\frac{b}{a} - \pi & a < 0 \text{ and } b < 0 \\ \pi/2 & a = 0 \text{ and } b > 0 \\ -\pi/2 & a = 0 \text{ and } b < 0 \end{cases}$$

We show $SO(2)$ are exactly equivalent to standard rotations:

Proposition 7 (simple rotation). *A 2×2* rotation matrix *through angle θ is*

$$Q_\theta := \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

We have $Q \in SO(2)$ if and only if $Q = Q_\theta$ for some $\theta \in \mathbb{R}$.

Proof

First assume Q_θ is of that form and write $c = \cos\theta$ and $s = \sin\theta$. Then we have

$$Q_\theta^\top Q_\theta = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} c & -s \\ s & c \end{pmatrix} = \begin{pmatrix} c^2+s^2 & 0 \\ 0 & c^2+s^2 \end{pmatrix} = I$$

and $\det Q_\theta = c^2 + s^2 = 1$ hence $Q_\theta \in SO(2)$.

Now suppose $Q = [\boldsymbol{q}_1, \boldsymbol{q}_2] \in SO(2)$ where we know its columns have norm 1, i.e. $\|\boldsymbol{q}_k\| = 1$, and are orthogonal. Write $\boldsymbol{q}_1 = [c, s]$ where we know $c = \cos\theta$ and $s = \sin\theta$ for $\theta = \text{atan}(s, c)$. Since $\boldsymbol{q}_1 \cdot \boldsymbol{q}_2 = 0$ we can deduce $\boldsymbol{q}_2 = \pm[-s, c]$. The sign is positive as $\det Q = \pm(c^2+s^2) = \pm 1$.

■

We can rotate an arbitrary vector in $\mathbb{R}^2$ to the unit axis using rotations, which are useful in linear algebra decompositions. Interestingly it only requires basic algebraic functions (no trigonometric functions):

Proposition 8 (rotation of a vector)**.** *The matrix*

$$Q = \frac{1}{\sqrt{a^2+b^2}} \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$$

is a rotation matrix ($Q \in SO(2)$) satisfying

$$Q \begin{bmatrix} a \\ b \end{bmatrix} = \sqrt{a^2+b^2} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Proof

The last equation is trivial so the only question is that it is a rotation matrix. This follows immediately:

$$Q^\top Q = \frac{1}{a^2+b^2} \begin{bmatrix} a^2+b^2 & 0 \\ 0 & a^2+b^2 \end{bmatrix} = I$$

and $\det Q = 1$.

■

Example 15 (rotating a vector)**.** Consider the vector

$$\boldsymbol{x} = \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix}.$$

We can use the proposition above to deduce the rotation matrix that rotates this vector to the positive real axis is:

$$\frac{1}{\sqrt{1+3}} \begin{bmatrix} -1 & -\sqrt{3} \\ \sqrt{3} & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -1 & -\sqrt{3} \\ \sqrt{3} & -1 \end{bmatrix}.$$

Alternatively, we could determine the matrix by computing the angle of the vector via:

$$\theta = \text{atan}(-\sqrt{3}, -1) = \text{atan}(\sqrt{3}) - \pi = -\frac{2\pi}{3}.$$

We thus compute:

$$Q_{-\theta} = \begin{bmatrix} \cos(2\pi/3) & -\sin(2\pi/3) \\ \sin(2\pi/3) & \cos(2\pi/3) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -1 & -\sqrt{3} \\ \sqrt{3} & -1 \end{bmatrix}.$$

More generally, we can consider rotations that operate on two entries of a vector at a time. This will be explored in the problem sheet/lab.

III.4.2 Reflections

In addition to rotations, another type of orthogonal/unitary matrix are reflections. These are specified by a single vector which is reflected, with everything orthogonal to the vector left fixed.

Definition 20 (reflection matrix)**.** Given a unit vector $\boldsymbol{v} \in \mathbb{C}^n$ (satisfying $\|\boldsymbol{v}\| = 1$), define the corresponding *reflection matrix* as:

$$Q_{\boldsymbol{v}} := I - 2\boldsymbol{v}\boldsymbol{v}^\star$$

These are indeed reflections in the direction of $\boldsymbol{v}$. We can show this as follows:

Proposition 9 (Householder properties)**.** *$Q_{\boldsymbol{v}}$ satisfies:*

1. *Symmetry: $Q_{\boldsymbol{v}} = Q_{\boldsymbol{v}}^\star$*
2. *Orthogonality: $Q_{\boldsymbol{v}} \in U(n)$*
3. *The vector $\boldsymbol{v}$ is an eigenvector of $Q_{\boldsymbol{v}}$ with eigenvalue -1*
4. *For the dimension $n-1$ space $W := \{\boldsymbol{w} : \boldsymbol{w}^\star\boldsymbol{v} = 0\}$, all vectors $\boldsymbol{w} \in W$ satisfy $Q_{\boldsymbol{v}}\boldsymbol{w} = \boldsymbol{w}$.*
5. *Not a rotation:* $\det Q_{\boldsymbol{v}} = -1$

Proof

Property 1 follows immediately. Property 2 follows from

$$Q_{\boldsymbol{v}}^\star Q_{\boldsymbol{v}} = Q_{\boldsymbol{v}}^2 = I - 4\boldsymbol{v}\boldsymbol{v}^\star + 4\boldsymbol{v}\boldsymbol{v}^\star\boldsymbol{v}\boldsymbol{v}^\star = I.$$

Property 3 follows since

$$Q_{\boldsymbol{v}}\boldsymbol{v} = \boldsymbol{v} - 2\boldsymbol{v}(\boldsymbol{v}^\star\boldsymbol{v}) = -\boldsymbol{v}.$$

Property 4 follows from:

$$Q_{\boldsymbol{v}}\boldsymbol{w} = \boldsymbol{w} - 2\boldsymbol{v}(\boldsymbol{w}^\star\boldsymbol{v}) = \boldsymbol{w}$$

Property 5 then follows: Property 4 tells us that 1 is an eigenvalue with multiplicity $n-1$. Since -1 is an eigenvalue with multiplicity 1, the determinant, which is product of the eigenvalues, is -1.

■

Example 16 (reflection through 2-vector)**.** Consider reflection through $\boldsymbol{x} = [1, 2]^\top$. We first need to normalise $\boldsymbol{x}$:

$$\boldsymbol{v} = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix}$$

The reflection matrix is:

$$Q_{\boldsymbol{v}} = I - 2\boldsymbol{v}\boldsymbol{v}^\top = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} - \frac{2}{5}\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} = \frac{1}{5}\begin{bmatrix} 3 & -4 \\ -4 & -3 \end{bmatrix}$$

Indeed it is symmetric, and orthogonal. It sends $\boldsymbol{x}$ to $-\boldsymbol{x}$:

$$Q_{\boldsymbol{v}}\boldsymbol{x} = \frac{1}{5}\begin{bmatrix} 3-8 \\ -4-6 \end{bmatrix} = -\boldsymbol{x}$$

Any vector orthogonal to $\boldsymbol{x}$, like $\boldsymbol{y} = [-2, 1]^\top$, is left fixed:

$$Q_{\boldsymbol{v}}\boldsymbol{y} = \frac{1}{5}\begin{bmatrix} -6-4 \\ 8-3 \end{bmatrix} = \boldsymbol{y}$$

Note that *building* the matrix $Q_{\boldsymbol{v}}$ will be expensive ($O(n^2)$ operations), but we can *apply* $Q_{\boldsymbol{v}}$ to a vector in $O(n)$ operations using the expression:

$$Q_{\boldsymbol{v}}\boldsymbol{x} = \boldsymbol{x} - 2\boldsymbol{v}(\boldsymbol{v}^\star\boldsymbol{x}) = \boldsymbol{x} - 2\boldsymbol{v}(\boldsymbol{v}\cdot\boldsymbol{x}).$$

Householder reflections

Just as rotations can be used to rotate vectors to be aligned with coordinate axes, so can reflections, but in this case it works for vectors in $\mathbb{C}^n$, not just $\mathbb{R}^2$. We begin with the real case:

Definition 21 (Householder reflection, real case)**.** For a given vector $\boldsymbol{x} \in \mathbb{R}^n$, define the Householder reflection

$$Q_{\boldsymbol{x}}^{\pm,\mathrm{H}} := Q_{\boldsymbol{w}}$$

for $\boldsymbol{y} = \mp\|\boldsymbol{x}\|\boldsymbol{e}_1 + \boldsymbol{x}$ and $\boldsymbol{w} = \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|}$. The default choice in sign is:

$$Q_{\boldsymbol{x}}^{\mathrm{H}} := Q_{\boldsymbol{x}}^{-\mathrm{sign}(x_1),\mathrm{H}}.$$

Lemma 5 (Householder reflection maps to axis)**.** *For* $\boldsymbol{x} \in \mathbb{R}^n$,

$$Q_{\boldsymbol{x}}^{\pm,\mathrm{H}}\boldsymbol{x} = \pm\|\boldsymbol{x}\|\boldsymbol{e}_1$$

Proof Note that

$$\begin{aligned}\|\boldsymbol{y}\|^2 &= 2\|\boldsymbol{x}\|^2 \mp 2\|\boldsymbol{x}\|x_1,\\ \boldsymbol{y}^\top\boldsymbol{x} &= \|\boldsymbol{x}\|^2 \mp \|\boldsymbol{x}\|x_1\end{aligned}$$

where $x_1 = \boldsymbol{e}_1^\top\boldsymbol{x}$. Therefore:

$$Q_{\boldsymbol{x}}^{\pm,\mathrm{H}}\boldsymbol{x} = (I - 2\boldsymbol{w}\boldsymbol{w}^\top)\boldsymbol{x} = \boldsymbol{x} - 2\frac{\boldsymbol{y}\|\boldsymbol{x}\|}{\|\boldsymbol{y}\|^2}(\|\boldsymbol{x}\| \mp x_1) = \boldsymbol{x} - \boldsymbol{y} = \pm\|\boldsymbol{x}\|\boldsymbol{e}_1.$$

■

Remark Why do we choose the the opposite sign of x_1 for the default reflection? For stability, but we won't discuss this in more detail.

We can extend this definition for complex vectors. In this case the choice of the sign is delicate and so we only generalise the default choice using a complex-analogue of the sign fuunction.

Definition 22 (Householder reflection, complex case)**.** For a given vector $\boldsymbol{x} \in \mathbb{C}^n$, define the Householder reflection as

$$Q_{\boldsymbol{x}}^{\mathrm{H}} := Q_{\boldsymbol{w}}$$

for $\boldsymbol{y} = \text{csign}(x_1)\|\boldsymbol{x}\|\boldsymbol{e}_1 + \boldsymbol{x}$ and $\boldsymbol{w} = \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|}$, for $\text{csign}(z) = \mathrm{e}^{\mathrm{i}\arg z}$.

Lemma 6 (Householder reflection maps to axis, complex case)**.** *For* $\boldsymbol{x} \in \mathbb{C}^n$*,*

$$Q_{\boldsymbol{x}}^{\mathrm{H}}\boldsymbol{x} = -\text{csign}(x_1)\|\boldsymbol{x}\|\boldsymbol{e}_1$$

Proof Denote $\alpha := \text{csign}(x_1)$. Note that $\bar{\alpha}x_1 = \mathrm{e}^{-\mathrm{i}\arg x_1}x_1 = |x_1|$. Now we have

$$\begin{aligned}
\|\boldsymbol{y}\|^2 &= (\alpha\|\boldsymbol{x}\|\boldsymbol{e}_1 + \boldsymbol{x})^\star(\alpha\|\boldsymbol{x}\|\boldsymbol{e}_1 + \boldsymbol{x}) = |\alpha|\|\boldsymbol{x}\|^2 + \|\boldsymbol{x}\|\alpha\bar{x}_1 + \bar{\alpha}x_1\|\boldsymbol{x}\| + \|\boldsymbol{x}\|^2 \\
&= 2\|\boldsymbol{x}\|^2 + 2|x_1|\|\boldsymbol{x}\| \\
\boldsymbol{y}^\star\boldsymbol{x} &= \bar{\alpha}x_1\|\boldsymbol{x}\| + \|\boldsymbol{x}\|^2 = \|\boldsymbol{x}\|^2 + |x_1|\|\boldsymbol{x}\|
\end{aligned}$$

Therefore:

$$Q_{\boldsymbol{x}}^{\mathrm{H}}\boldsymbol{x} = (I - 2\boldsymbol{w}\boldsymbol{w}^\star)\boldsymbol{x} = \boldsymbol{x} - 2\frac{\boldsymbol{y}}{\|\boldsymbol{y}\|^2}(\|\boldsymbol{x}\|^2 + |x_1|\|\boldsymbol{x}\|) = \boldsymbol{x} - \boldsymbol{y} = -\alpha\|\boldsymbol{x}\|\boldsymbol{e}_1.$$

■

III.5 QR Factorisation

Let $A \in \mathbb{C}^{m\times n}$ be a rectangular or square matrix such that $m \geq n$ (i.e. more rows then columns). In this section we consider two closely related factorisations:

Definition 23 (QR factorisation)**.** The *QR factorisation* is

$$A = QR = \underbrace{\left[\boldsymbol{q}_1|\cdots|\boldsymbol{q}_m\right]}_{Q\in U(m)} \underbrace{\begin{bmatrix} \times & \cdots & \times \\ & \ddots & \vdots \\ & & \times \\ & & 0 \\ & & \vdots \\ & & 0 \end{bmatrix}}_{R\in\mathbb{C}^{m\times n}}$$

where Q is unitary (i.e., $Q \in U(m)$, satisfying $Q^\star Q = I$, with columns $\boldsymbol{q}_j \in \mathbb{C}^m$) and R is *right triangular*, which means it is only nonzero on or to the right of the diagonal ($r_{kj} = 0$ if $k > j$).

Definition 24 (Reduced QR factorisation)**.** The *reduced QR factorisation*

$$A = \hat{Q}\hat{R} = \underbrace{\left[\boldsymbol{q}_1|\cdots|\boldsymbol{q}_n\right]}_{\hat{Q}\in\mathbb{C}^{m\times n}} \underbrace{\begin{bmatrix} \times & \cdots & \times \\ & \ddots & \vdots \\ & & \times \end{bmatrix}}_{\hat{R}\in\mathbb{C}^{n\times n}}$$

where $\hat{Q}$ has orthonormal columns ($\hat{Q}^\star\hat{Q} = I$, $\boldsymbol{q}_j \in \mathbb{C}^m$) and $\hat{R}$ is upper triangular.

Note for a square matrix the reduced QR factorisation is equivalent to the QR factorisation, in which case R is *upper triangular*. The importance of these factorisation for square matrices is that their component pieces are easy to invert:

$$A = QR \qquad \Rightarrow \qquad A^{-1}\boldsymbol{b} = R^{-1}Q^\top \boldsymbol{b}$$

and we saw previously that triangular and orthogonal matrices are easy to invert when applied to a vector $\boldsymbol{b}$.

For rectangular matrices we will see that the QR factorisation leads to efficient solutions to the *least squares problem*: find $\boldsymbol{x}$ that minimizes the 2-norm $\|A\boldsymbol{x}-\boldsymbol{b}\|$. Note in the rectangular case the QR factorisation contains within it the reduced QR factorisation:

$$A = QR = \left[\hat{Q}|\boldsymbol{q}_{n+1}|\cdots|\boldsymbol{q}_m\right]\begin{bmatrix}\hat{R}\\ \boldsymbol{0}_{m-n\times n}\end{bmatrix} = \hat{Q}\hat{R}.$$

In this section we discuss the following:

1. Reduced QR and Gram–Schmidt: We discuss computation of the Reduced QR factorisation using Gram–Schmidt.
2. Householder reflections and QR: We discuss computing the QR factorisation using Householder reflections. This is a more accurate approach for computing QR factorisations.
3. QR and least squares: We discuss the QR factorisation and its usage in solving least squares problems.

III.5.1 Reduced QR and Gram–Schmidt

How do we compute the QR factorisation? We begin with a method you may have seen before in another guise. Write

$$A = \left[\boldsymbol{a}_1|\cdots|\boldsymbol{a}_n\right]$$

where $\boldsymbol{a}_k \in \mathbb{C}^m$ and assume they are linearly independent (A has full column rank).

Proposition 10 (Column spaces match)**.** *Suppose $A = \hat{Q}\hat{R}$ where $\hat{Q} = [\boldsymbol{q}_1|\ldots|\boldsymbol{q}_n]$ has orthonormal columns and $\hat{R}$ is upper-triangular, and A has full rank. Then the first j columns of $\hat{Q}$ span the same space as the first j columns of A:*

$$span(\boldsymbol{a}_1,\ldots,\boldsymbol{a}_j) = span(\boldsymbol{q}_1,\ldots,\boldsymbol{q}_j).$$

Proof

Because A has full rank we know $\hat{R}$ is invertible, i.e. its diagonal entries do not vanish: $r_{jj} \neq 0$. If $\boldsymbol{v} \in \mathrm{span}(\boldsymbol{a}_1,\ldots,\boldsymbol{a}_j)$ we have for $\boldsymbol{c} \in \mathbb{C}^j$

$$\boldsymbol{v} = \left[\boldsymbol{a}_1|\cdots|\boldsymbol{a}_j\right]\boldsymbol{c} = \left[\boldsymbol{q}_1|\cdots|\boldsymbol{q}_j\right]\hat{R}[1:j,1:j]\boldsymbol{c} \in \mathrm{span}(\boldsymbol{q}_1,\ldots,\boldsymbol{q}_j)$$

while if $\boldsymbol{w} \in \mathrm{span}(\boldsymbol{q}_1,\ldots,\boldsymbol{q}_j)$ we have for $\boldsymbol{d} \in \mathbb{R}^j$

$$\boldsymbol{w} = \left[\boldsymbol{q}_1|\cdots|\boldsymbol{q}_j\right]\boldsymbol{d} = \left[\boldsymbol{a}_1|\cdots|\boldsymbol{a}_j\right]\hat{R}[1:j,1:j]^{-1}\boldsymbol{d} \in \mathrm{span}(\boldsymbol{a}_1,\ldots,\boldsymbol{a}_j).$$

■

It is possible to find $\hat{Q}$ and $\hat{R}$ using the *Gram–Schmidt algorithm*. We construct it column-by-column. For $j = 1, 2, \ldots, n$ define

$$\begin{aligned} \boldsymbol{v}_j &:= \boldsymbol{a}_j - \sum_{k=1}^{j-1} \underbrace{\boldsymbol{q}_k^\star \boldsymbol{a}_j}_{r_{kj}} \boldsymbol{q}_k, \\ r_{jj} &:= \|\boldsymbol{v}_j\|, \\ \boldsymbol{q}_j &:= \frac{\boldsymbol{v}_j}{r_{jj}}. \end{aligned}$$

Theorem (Gram–Schmidt and reduced QR) Define $\boldsymbol{q}_j$ and r_{kj} as above (with $r_{kj} = 0$ if $k > j$). Then a reduced QR factorisation is given by:

$$A = \underbrace{\begin{bmatrix} \boldsymbol{q}_1 | \cdots | \boldsymbol{q}_n \end{bmatrix}}_{\hat{Q} \in \mathbb{C}^{m \times n}} \underbrace{\begin{bmatrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ & & r_{nn} \end{bmatrix}}_{\hat{R} \in \mathbb{C}^{n \times n}}$$

Proof

We first show that $\hat{Q}$ has orthonormal columns. Assume that $\boldsymbol{q}_\ell^\star \boldsymbol{q}_k = \delta_{\ell k}$ for $k, \ell < j$. For $\ell < j$ we then have

$$\boldsymbol{q}_\ell^\star \boldsymbol{v}_j = \boldsymbol{q}_\ell^\star \boldsymbol{a}_j - \sum_{k=1}^{j-1} \boldsymbol{q}_\ell^\star \boldsymbol{q}_k \boldsymbol{q}_k^\star \boldsymbol{a}_j = 0$$

hence $\boldsymbol{q}_\ell^\star \boldsymbol{q}_j = 0$ and indeed $\hat{Q}$ has orthonormal columns. Further: from the definition of $\boldsymbol{v}_j$ we find

$$\boldsymbol{a}_j = \boldsymbol{v}_j + \sum_{k=1}^{j-1} r_{kj} \boldsymbol{q}_k = \sum_{k=1}^{j} r_{kj} \boldsymbol{q}_k = \hat{Q} \hat{R} \boldsymbol{e}_j$$

■

III.5.2 Householder reflections and QR

As an alternative, we will consider using Householder reflections to introduce zeros below the diagonal. Thus, if Gram–Schmidt is a process of *triangular orthogonalisation* (using triangular matrices to orthogonalise), Householder reflections is a process of *orthogonal triangularisation* (using orthogonal matrices to triangularise).

Consider multiplication by the Householder reflection corresponding to the first column, that is, for

$$Q_1 := Q_{\boldsymbol{a}_1}^{\mathrm{H}},$$

consider

$$Q_1 A = \begin{bmatrix} \times & \times & \cdots & \times \\ & \times & \cdots & \times \\ & \vdots & \ddots & \vdots \\ & \times & \cdots & \times \end{bmatrix} = \begin{bmatrix} \alpha_1 & \boldsymbol{w}_1^\top \\ & A_2 \end{bmatrix}$$

where

$$\alpha_1 := -\mathrm{csign}(a_{11}) \|\boldsymbol{a}_1\|, \boldsymbol{w}_1 = (Q_1 A)[1, 2:n] \qquad \text{and} \qquad A_2 = (Q_1 A)[2:m, 2:n],$$

where as before $\mathrm{csign}(z) := \mathrm{e}^{\mathrm{i} \arg z}$. That is, we have made the first column triangular. In terms of an algorithm, we then introduce zeros into the first column of A_2, leaving an A_3, and so-on. But we can wrap this iterative algorithm into a simple proof by induction, reminisicent of our proofs for the PLU and Cholesky factorisations:

Theorem 7 (QR)**.** *Every matrix $A \in \mathbb{C}^{m\times n}$ has a QR factorisation:*

$$A = QR$$

where $Q \in U(m)$ and $R \in \mathbb{C}^{m\times n}$ is right triangular.

Proof

First assume $m \geq n$. If $A = [\boldsymbol{a}_1] \in \mathbb{C}^{m\times 1}$ then we have for the Householder reflection $Q_1 = Q^{\mathrm{H}}_{\boldsymbol{a}_1}$

$$Q_1 A = \alpha \boldsymbol{e}_1$$

which is right triangular, where $\alpha = -\mathrm{csign}(a_{11})\|\boldsymbol{a}_1\|$. In other words

$$A = \underbrace{Q_1}_{Q} \underbrace{\alpha \boldsymbol{e}_1}_{R}.$$

For $n > 1$, assume every matrix with less columns than n has a QR factorisation. For $A = [\boldsymbol{a}_1|\dots|\boldsymbol{a}_n] \in \mathbb{C}^{m\times n}$, let $Q_1 = Q^{\mathrm{H}}_{\boldsymbol{a}_1}$ so that

$$Q_1 A = \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & A_2 \end{bmatrix}.$$

By assumption $A_2 = Q_2 R_2$. Thus we have (recalling that $Q_1^{-1} = Q_1^\star = Q_1$):

$$A = Q_1 \begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & Q_2 R_2 \end{bmatrix}$$
$$= \underbrace{Q_1 \begin{bmatrix} 1 & \\ & Q_2 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} \alpha & \boldsymbol{w}^\top \\ & R_2 \end{bmatrix}}_{R}.$$

If $m < n$, i.e., A has more columns then rows, write

$$A = \begin{bmatrix} \tilde{A} & B \end{bmatrix}$$

where $\tilde{A} \in \mathbb{C}^{m\times m}$. From above we know we can write $\tilde{A} = Q\tilde{R}$. We thus have

$$A = Q \underbrace{\begin{bmatrix} \tilde{R} & Q^\star B \end{bmatrix}}_{R}$$

where R is right triangular.

■

Example 17 (QR by hand)**.** We will now do an example by hand. Consider finding the QR factorisation where the diagonal of R is positive for the 4×3 matrix

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

For the first column, since the entry $a_{11} > 0$ on a computer we would want to choose the Householder reflection that makes this negative, but in this case we want R to have a positive diagonal (partly because the numbers involved become very complicated otherwise!). So instead we choose the "wrong" sign and leave it positive. Since $\|\boldsymbol{a}_1\| = 2$ we have

$$\boldsymbol{y}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \Rightarrow \boldsymbol{w}_1 = \frac{\boldsymbol{y}_1}{\|\boldsymbol{y}_1\|} = \frac{1}{2}\begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}.$$

Hence

$$Q_1 := I - \frac{1}{2}\begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}\begin{bmatrix} -1 & -1 & -1 & -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

so that

$$Q_1 A = \begin{bmatrix} 2 & 1 & 0 \\ & 0 & 0 \\ & -1 & -1 \\ & 0 & -1 \end{bmatrix}$$

For the second column we have a zero entry so on a computer we can either send it to positive or negative sign, but in this case we are told to make it positive. Thus we have

$$\boldsymbol{y}_2 := [0, -1, 0] - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix} \Rightarrow \boldsymbol{w}_2 = \frac{\boldsymbol{y}_2}{\|\boldsymbol{y}_2\|} = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}$$

Thus we have

$$Q_2 := I - \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}\begin{bmatrix} -1 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

so that

$$\tilde{Q}_2 Q_1 A = \begin{bmatrix} 2 & 1 & 0 \\ & 1 & 1 \\ & & 0 \\ & & -1 \end{bmatrix}$$

The final vector is

$$\boldsymbol{y}_3 := \begin{bmatrix} 0 \\ -1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \Rightarrow \boldsymbol{w}_3 = -\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Hence

$$Q_3 := I - \begin{bmatrix} 1 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$$

so that

$$\tilde{Q}_3 \tilde{Q}_2 Q_1 A = \begin{bmatrix} 2 & 1 & 0 \\ & 1 & 1 \\ & & 1 \\ & & 0 \end{bmatrix} =: R$$

and

$$Q := Q_1 \tilde{Q}_2 \tilde{Q}_3 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 \\ -1 & 1 & -1 & -1 \end{bmatrix}.$$

III.5.3 QR and least squares

We consider rectangular matrices with more rows than columns. Given $A \in \mathbb{C}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{C}^m$, a least squares problem consists of finding a vector $\boldsymbol{x} \in \mathbb{C}^n$ that minimises the 2-norm: $\|A\boldsymbol{x} - \boldsymbol{b}\|$. There is a lot of theory around least squares, however, we focus on a simple computational aspect: we can solve least squares problems using the QR factorisation.

Theorem 8 (least squares via QR)**.** *Suppose $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ has full rank and a QR factorisation $A = QR$ (which includes within it a reduced QR factorisation $A = \hat{Q}\hat{R}$). The vector*

$$\boldsymbol{x} = \hat{R}^{-1}\hat{Q}^\star \boldsymbol{b}$$

minimises $\|A\boldsymbol{x} - \boldsymbol{b}\|$.

Proof

The norm-preserving property ($\|Q\boldsymbol{x}\| = \|\boldsymbol{x}\|$) of unitary matrices tells us

$$\|A\boldsymbol{x} - \boldsymbol{b}\| = \|QR\boldsymbol{x} - \boldsymbol{b}\| = \|Q(R\boldsymbol{x} - Q^\star\boldsymbol{b})\| = \|R\boldsymbol{x} - Q^\star\boldsymbol{b}\| = \left\| \begin{bmatrix} \hat{R} \\ \mathbf{0}_{m-n \times n} \end{bmatrix} \boldsymbol{x} - \begin{bmatrix} \hat{Q}^\star \\ \boldsymbol{q}_{n+1}^\star \\ \vdots \\ \boldsymbol{q}_m^\star \end{bmatrix} \boldsymbol{b} \right\|$$

Now note that the rows $k > n$ are independent of $\boldsymbol{x}$ and are a fixed contribution. Thus to minimise this norm it suffices to drop them and minimise:

$$\|\hat{R}\boldsymbol{x} - \hat{Q}^\star\boldsymbol{b}\|$$

This norm is minimised if it is zero. Provided the column rank of A is full, $\hat{R}$ will be invertible.