Vectors and matrices: the operations the course uses

Everything the course does with states and gates is one of a handful of operations. Each is shown twice: first the structure with placeholders, then one worked example you can check by hand. The NumPy calls are at the end.

1 · The objects

A **vector** is a column of numbers; a qubit state is a vector with two entries. A **matrix** is a rectangle of numbers; a gate is a 2×2 matrix. Shapes are written rows × columns. Sizes must chain: a 2×2 matrix can multiply a 2-entry vector or another 2×2 matrix, a 4×4 matrix needs a 4-entry vector.

$$v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \qquad A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \qquad I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

2 · Scalar times vector or matrix

Multiply every entry by the number. Nothing else changes, not even the shape.

$$s \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} s\,v_1 \\ s\,v_2 \end{pmatrix}, \qquad s \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} s\,a & s\,b \\ s\,c & s\,d \end{pmatrix}$$

$$3 \begin{pmatrix} 1 \\ -2 \end{pmatrix} = \begin{pmatrix} 3 \\ -6 \end{pmatrix}, \qquad \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 0.707 & 0.707 \\ 0.707 & -0.707 \end{pmatrix}$$

3 · Matrix times vector

Each entry of the result is one **row** of the matrix combined with the vector: multiply entry by entry, add up. Applied to a basis vector, a matrix returns one of its columns.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot 1 + 1 \cdot 0 \\ 1 \cdot 1 + (-1) \cdot 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

4 · Matrix times matrix, and the order

Entry (row i, column j) of the product is row i of the left matrix combined with column j of the right one. The order matters: AB and BA are different matrices in general. When several matrices act on a vector, **the one next to the vector acts first**: in S Z v, Z acts before S. A circuit diagram draws gates left to right, so the matrix product reads in the opposite direction.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 4 & 3 \end{pmatrix}, \qquad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 3 & 4 \\ 1 & 2 \end{pmatrix}$$

5 · Length of a vector

Square the magnitude of every entry, add up, take the root. A state vector always has length 1; that is the reality check the course applies to every state.

$$\left\| \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \right\| = \sqrt{|v_1|^2 + |v_2|^2}$$

$$\left\| \begin{pmatrix} 3 \\ 4 \end{pmatrix} \right\| = \sqrt{9+16} = 5, \qquad \left\| \begin{pmatrix} 0.707 \\ 0.707 \end{pmatrix} \right\| = \sqrt{0.5+0.5} = 1$$

A four-entry vector, the state of two qubits, works the same way with four terms under the root.

$$\left\| \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \right\| = \sqrt{|v_1|^2 + |v_2|^2 + |v_3|^2 + |v_4|^2}$$

$$\left\| \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} \right\| = \sqrt{4 \cdot 0.25} = 1, \qquad \left\| \begin{pmatrix} 1 \\ 2 \\ 2 \\ 0 \end{pmatrix} \right\| = \sqrt{1+4+4+0} = 3$$

6 · Conjugate, transpose, dagger

The **conjugate** of a complex number z = x + iy flips the sign of its imaginary part; a real number stays as it is. Conjugating a vector or matrix means conjugating every entry.

$$\overline{x + i\,y} = x - i\,y, \qquad \overline{2+3i} = 2-3i, \qquad \overline{5} = 5, \qquad \overline{\begin{pmatrix} 1 & 2i \\ 3 & 1+i \end{pmatrix}} = \begin{pmatrix} 1 & -2i \\ 3 & 1-i \end{pmatrix}$$

The **transpose** swaps rows and columns: the first row becomes the first column. Doing both, conjugate and transpose, is the **conjugate transpose**, written with a dagger †. With placeholders p, q, r, s for the entries:

$$M = \begin{pmatrix} p & q \\ r & s \end{pmatrix}, \qquad M^T = \begin{pmatrix} p & r \\ q & s \end{pmatrix}, \qquad M^\dagger = \begin{pmatrix} \bar{p} & \bar{r} \\ \bar{q} & \bar{s} \end{pmatrix}$$

$$M = \begin{pmatrix} 1 & 2i \\ 3 & 1+i \end{pmatrix}, \qquad M^T = \begin{pmatrix} 1 & 3 \\ 2i & 1+i \end{pmatrix}, \qquad M^\dagger = \begin{pmatrix} 1 & 3 \\ -2i & 1-i \end{pmatrix}$$

A matrix is a valid gate exactly when its dagger undoes it: U†U = I. This check is the first homework of the course. Example with U = (1, i; 1, −i)/√2:

$$U^\dagger U = \tfrac{1}{2} \begin{pmatrix} 1 & 1 \\ -i & i \end{pmatrix} \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} = \tfrac{1}{2} \begin{pmatrix} 1+1 & i-i \\ -i+i & 1+1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

7 · Tensor product: two qubits become one system

The **tensor product** (Kronecker product, symbol ⊗) glues two vectors into one longer vector: every entry of the first times the whole second. Two single-qubit states become one four-entry state (Session 2).

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a\,c \\ a\,d \\ b\,c \\ b\,d \end{pmatrix}, \qquad \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

The same rule glues two matrices: every entry of the first times the whole second matrix, giving a 4×4 matrix. This is how a 2×2 gate on one qubit becomes a 4×4 gate on the pair: tensor it with the identity for the qubit it leaves alone.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \otimes B = \begin{pmatrix} a\,B & b\,B \\ c\,B & d\,B \end{pmatrix} = \begin{pmatrix} ab_{11} & ab_{12} & b\,b_{11} & b\,b_{12} \\ ab_{21} & ab_{22} & b\,b_{21} & b\,b_{22} \\ cb_{11} & cb_{12} & d\,b_{11} & d\,b_{12} \\ cb_{21} & cb_{22} & d\,b_{21} & d\,b_{22} \end{pmatrix}$$

$$X \otimes I = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

8 · A 4×4 matrix times a four-entry state

Same rule as in 3, one row at a time, now with four terms per row. A 4×4 gate acts on the four-entry state of two qubits; it cannot multiply a 2×2 matrix, the sizes do not chain. The example is the CNOT gate on the state from section 7: it flips the second qubit when the first is 1, so the entry moves from position 3 to position 4.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 \end{pmatrix}$$

$$\mathrm{CNOT} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

In NumPy

Operation	Call
Build	np.array([1, 0]) · np.array([[1, 1], [1, -1]]) · np.eye(2)
Scalar times	3 * v · A / np.sqrt(2)
Matrix times vector / matrix	A @ v · A @ B — never *, which is elementwise
Order	S @ Z @ v applies Z first, then S
Length	np.linalg.norm(v)
Conjugate, transpose, dagger	np.conj(A) · A.T · A.conj().T
Tensor product	np.kron(a, b) — vectors and matrices alike
Compare with tolerance	np.allclose(U.conj().T @ U, np.eye(2))

Two pitfalls that account for most bugs. `A * B` multiplies entry by entry and is not a matrix product; use `@`. And a real array silently drops imaginary parts, so build states and gates with `dtype=complex` as soon as an i appears.

Refresher, if any of this felt rusty

Two hours at most; or ask an LLM for the idea behind the operation you are unsure about, then redo the example above by hand.

Resource	By	Time	Link
NumPy: the absolute basics for beginners	numpy.org	1 h	numpy.org/doc/stable/user/absolute_beginners.html
Vectors, what even are they? (Essence of linear algebra, ch. 1)	3Blue1Brown	10 min	youtu.be/fNk_zzaMoSs
Linear transformations and matrices (ch. 3)	3Blue1Brown	11 min	youtu.be/kYB8IZa5AuE
Matrix multiplication as composition (ch. 4)	3Blue1Brown	10 min	youtu.be/XkY2DOUCWMU