QUANTUM KICKSTART · SAMPLE TOPICS

Five Topics, Carried Through

Sample pages from the Quantum Kickstart extended handouts

One topic from each of the five sessions. Each runs from the slides that build it through to the companion-notebook page that makes it executable.

qubit-lab.ch

qubit-lab.ch
contact@qubit-lab.ch

What this is

One topic from each of the five Kickstart sessions, each carried through in full: the consecutive slides that build the idea, followed by the companion-notebook page where it becomes executable.

Single pages show a house style. Complete topics show whether the material actually teaches — whether a claim is set up, worked through, and then checked in code. Taking one topic from every session also shows the range: the first three assume no physics and can be followed line by line, while the last two are the judgement a team needs before spending money on hardware or believing a machine-learning result.

Quantum Kickstart is a five-session introduction for quantitative and data-engineering teams. The prerequisites are Python and the ability to multiply matrices; no physics is assumed. Each session runs three to four hours and pairs a deck with a runnable Jupyter notebook, and each participant receives an extended handout of the kind sampled here — every slide that carries an argument, reproduced with the explanation that accompanied it in the room.

The three devices used throughout

Check yourself. A question the reader should be able to answer before moving on — used to convert a passive page into an active one.

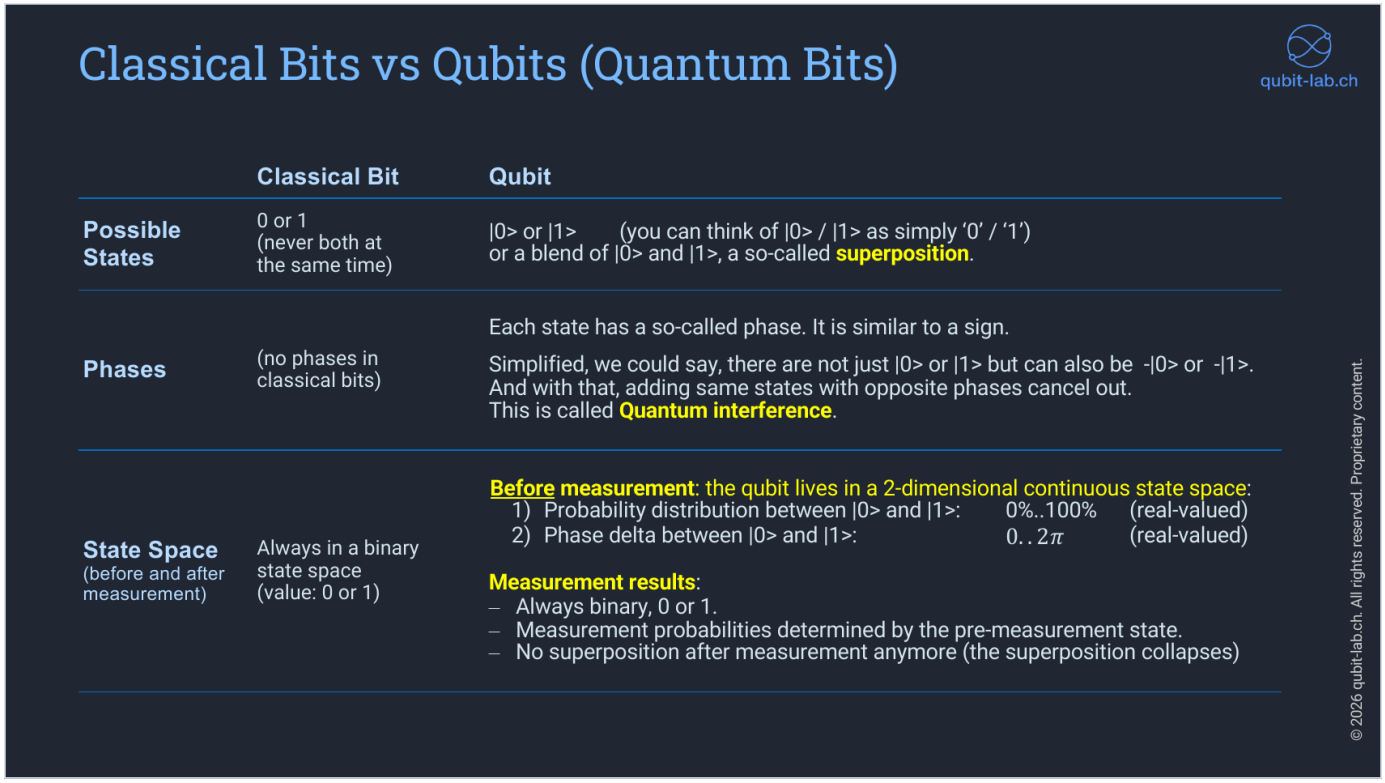
Common pitfall. A mistake that is made reliably. Naming the error explicitly is more durable than stating the correct version once.

In the notebook. The pointer into the companion notebook, so every claim on the page can be executed rather than believed. Each topic here ends on the page it points at.

The standard applied throughout

Every quantitative claim is stated against a **classical referee** — the exact optimum where one can be computed, a held-out test set and the incumbent method where it cannot. Two of the five notebook pages here exist mainly to apply that standard: one checks a textbook formula against a noisy simulation, and one puts a quantum classifier next to logistic regression, an SVM and a random forest. Where there is no advantage to report, the material reports that.

TOPIC 1 OF 5 · WHAT IS A QUBIT · SESSION 1



A bit is a value; a qubit is a direction

A classical bit is in one of two states, and that is all it can be. A qubit has the same two basis states, but it can also occupy a weighted combination of them — and the weights carry a phase, which is why the continuum here is a circle rather than a line segment.

The practical consequence: to describe a bit you need one binary digit; to describe a qubit before measurement you need two complex numbers. That is a far richer object, and the rest of this session is about handling it precisely.

> **Common pitfall.** "A qubit is both 0 and 1 at the same time." Prefer: a qubit is in a definite state that is not one of the two you are able to read out. Measurement is what forces it into one of them.

TOPIC 1 OF 5 · WHAT IS A QUBIT · SESSION 1

Mathematical Representation of a Qubit State

qubit-lab.ch

A single qubit can be in state $|0\rangle$, state $|1\rangle$, or any superposition of the two, as long as the measurement probabilities add up to 1 (100%).

The general state $|\psi\rangle$ of a qubit can be expressed as:

$$|\psi\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle, \quad \alpha, \beta \in \mathbb{C}, \quad |\alpha|^2 + |\beta|^2 = 1$$

where α and β are **probability amplitudes** for the basis states $|0\rangle$ and $|1\rangle$, and are complex numbers.

N.B. probability amplitudes are – despite the name – not probabilities yet. See next slide!

The definition everything rests on

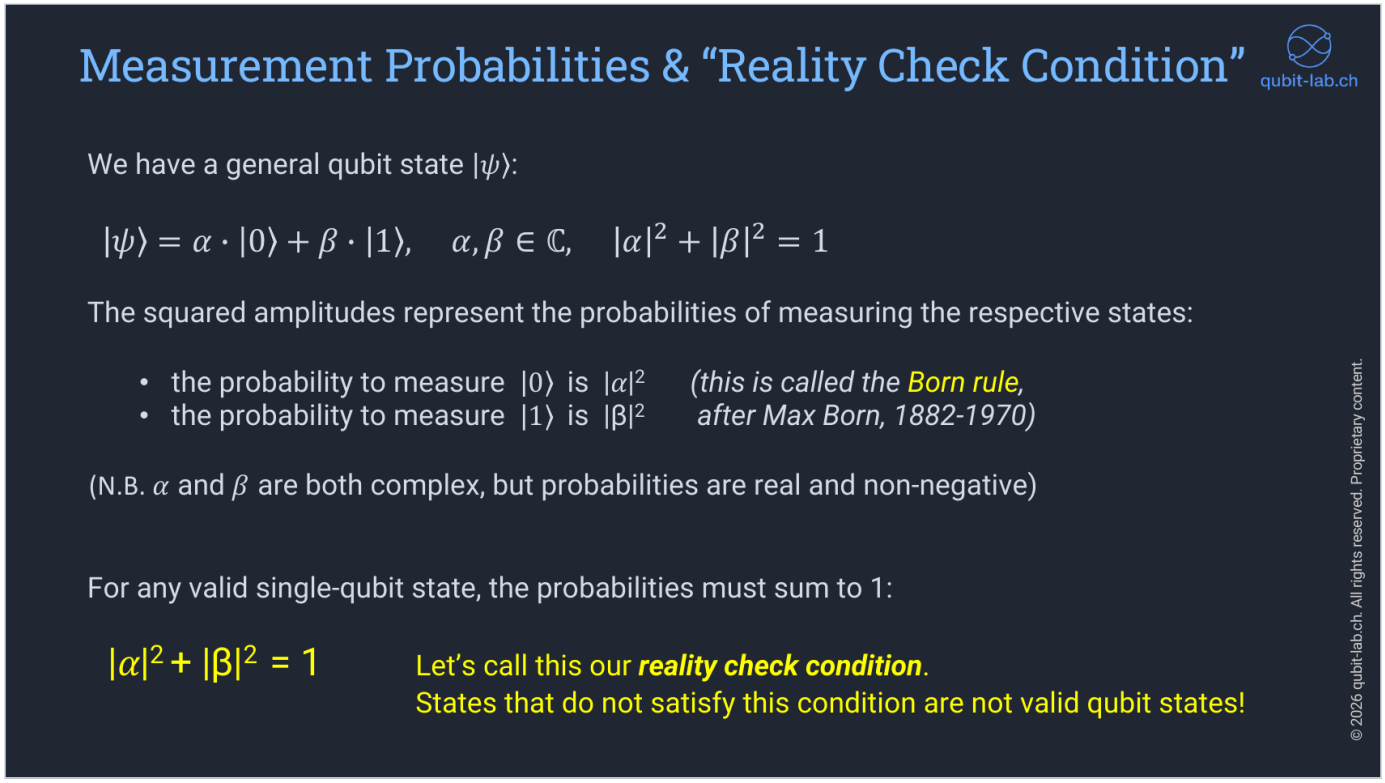
This is the single most important line of the course. A qubit state is a weighted sum of the two basis states, where the weights — the **probability amplitudes** — are complex numbers.

$$|\psi\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle \qquad \alpha, \beta \in \mathbb{C}$$

Two things to notice. The amplitudes are not probabilities; their squared magnitudes are, which is what the next slide makes precise. And their being complex is not decoration — it is precisely the freedom that gives us phase, and phase is what makes interference possible.

If you take one formula out of Session 1, take this one.

TOPIC 1 OF 5 · WHAT IS A QUBIT · SESSION 1



The reality check condition

The squared magnitudes of the amplitudes are the probabilities of the two possible outcomes — the **Born rule**. Since 0 and 1 are the only things that can happen, those probabilities must sum to one.

$$P(0) = |\alpha|^2 \quad P(1) = |\beta|^2 \quad |\alpha|^2 + |\beta|^2 = 1$$

We will call this the **reality check condition** and use it constantly. Any time you produce a state vector, check it: it is the fastest way to catch an arithmetic slip, and later it is what forces quantum gates to be unitary rather than arbitrary matrices.

It is not a physical mystery. Probabilities of exhaustive alternatives sum to one, exactly as they do in any risk model.

TOPIC 1 OF 5 · WHAT IS A QUBIT · SESSION 1

Session 1 Companion Notebook — what is a qubit

The definition, and the counter-example. The reality-check condition from the slides, tested in code: a state is only a qubit if its squared amplitudes sum to one.

In [2]:

```
# The computational basis states, as plain column vectors
ket0 = np.array([1, 0], dtype=complex)
ket1 = np.array([0, 1], dtype=complex)

# The two balanced superpositions from the slides
ket_plus  = (ket0 + ket1) / np.sqrt(2)     # |+>
ket_minus = (ket0 - ket1) / np.sqrt(2)     # |->

def probabilities(psi):
    """Born rule: probability of measuring 0 and 1."""
    return np.abs(psi[0])**2, np.abs(psi[1])**2

def reality_check(psi, name="state"):
    p0, p1 = probabilities(psi)
    total = p0 + p1
    ok = np.isclose(total, 1.0)
    flag = "valid" if ok else "NOT valid"
    print(f"{name:10s} amplitudes = {np.round(psi, 3)}   "
          f"P(0) = {p0:6.1%}   P(1) = {p1:6.1%}   sum = {total:5.1%}  ->  {flag}")
    return ok

for name, psi in [("|0>", ket0), ("|1>", ket1), ("|+>", ket_plus), ("|->", ket_minus)]:
    reality_check(psi, name)
```

```
|0>        amplitudes = [1.+0.j 0.+0.j]   P(0) = 100.0%   P(1) =   0.0%   sum = 100.0%  ->  valid
|1>        amplitudes = [0.+0.j 1.+0.j]   P(0) =   0.0%   P(1) = 100.0%   sum = 100.0%  ->  valid
|+>        amplitudes = [0.707+0.j 0.707+0.j]   P(0) =  50.0%   P(1) =  50.0%   sum = 100.0%  ->  valid
|->        amplitudes = [ 0.707+0.j -0.707+0.j]   P(0) =  50.0%   P(1) =  50.0%   sum = 100.0%  ->  valid
```

In [3]:

```
broken = np.array([0.5, 0.5], dtype=complex)          # P(0) = P(1) = 25%
_ = reality_check(broken, "broken")
```

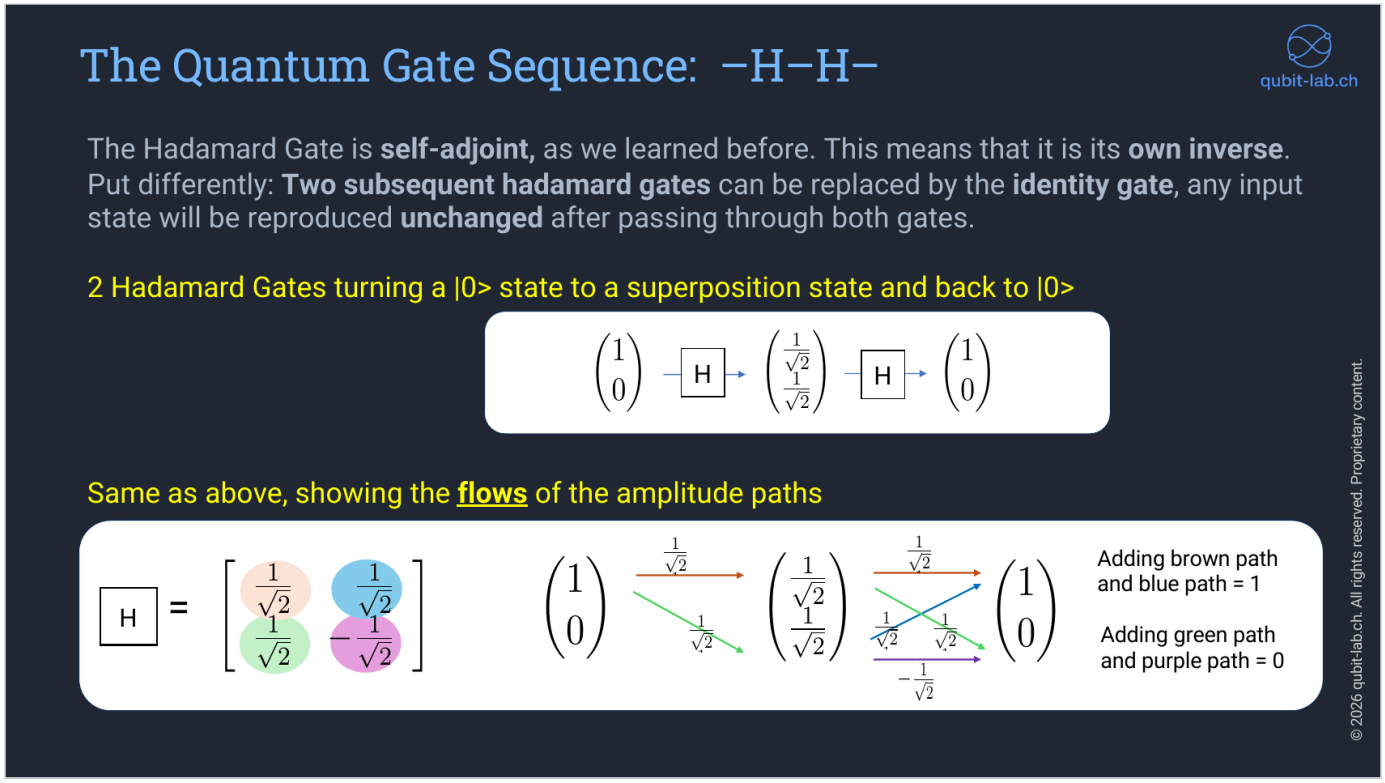
```
broken     amplitudes = [0.5+0.j 0.5+0.j]   P(0) =  25.0%   P(1) =  25.0%   sum = 50.0%  ->  NOT valid
```

…and the same condition, tested in code

The three pages before this one build to a single condition: a state is a qubit only if its squared amplitudes sum to one. This is the notebook cell that applies it — the four states from the slides, each checked, and then the counter-example, which fails the check and says so.

> **Check yourself.** The notebooks run on a laptop with no internet connection and no hardware account. Nothing in the first four sessions needs queue time or a vendor relationship, so a participant can re-run any of it the following week.

TOPIC 2 OF 5 · QUANTUM INTERFERENCE · SESSION 2



The Hadamard as an amplitude-path splitter

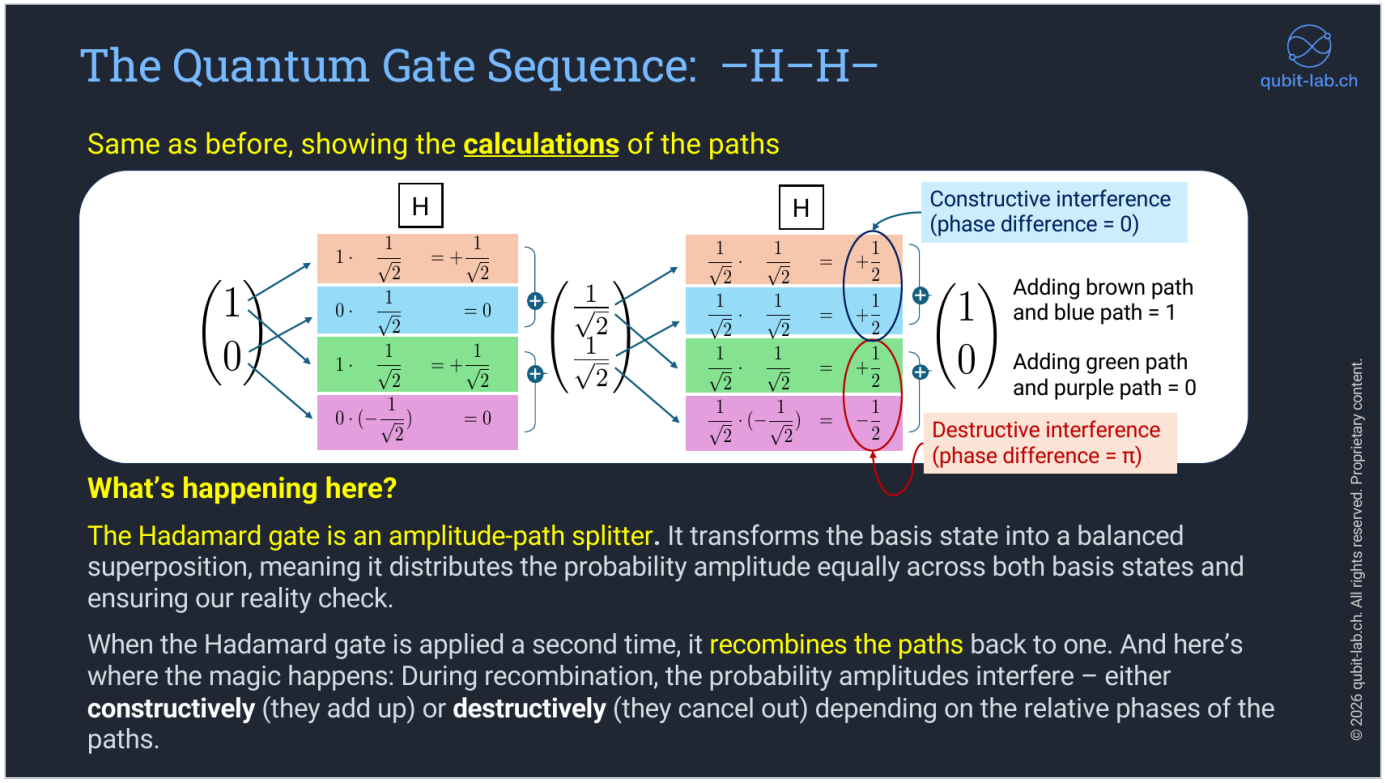
Start from what we know. H is self-adjoint, so it is its own inverse, so two Hadamards in a row are an identity. Nothing happens. That is the boring statement — and the slide asks us to look at how nothing happens.

The picture on the right is the useful one. The first Hadamard splits the amplitude into two paths, one via zero and one via one. The second recombines them. Follow the coloured flows: the paths arriving at the zero outcome add up, and the paths arriving at the one outcome cancel.

The identity is not the absence of activity. It is two things happening and cancelling exactly. Nothing happened only because the paths were precisely out of step at the one outcome — and that is a fact we are about to change.

> **In the notebook.** Code Bite 2.5 traces both paths numerically.

TOPIC 2 OF 5 · QUANTUM INTERFERENCE · SESSION 2



−H−H− with the arithmetic

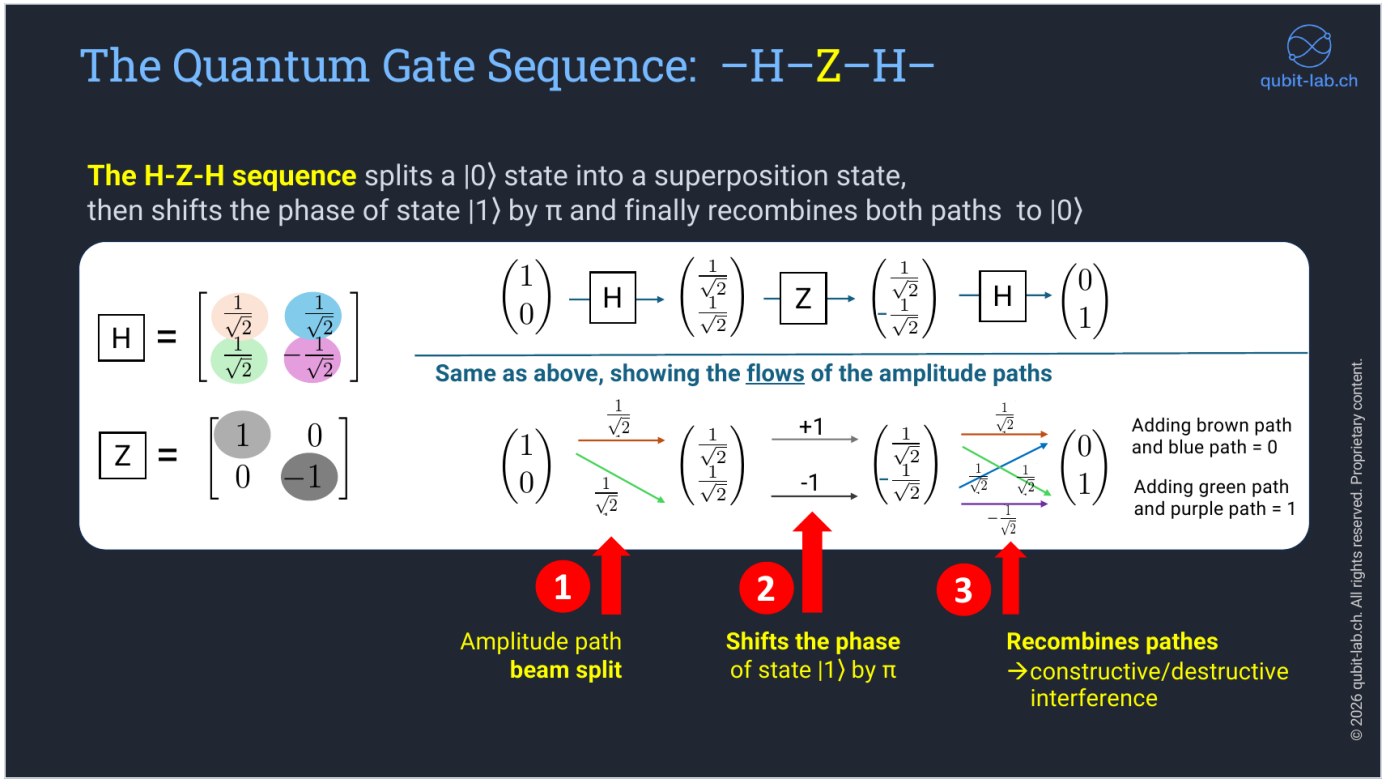
The same circuit, now with numbers. After the first H both paths carry 1/√2. The second H sends zero to a balanced sum and one to a balanced difference — that minus sign is the whole story.

$$\text{onto } |0\rangle : \ \tfrac{1}{2} + \tfrac{1}{2} = 1 \qquad \text{onto } |1\rangle : \ \tfrac{1}{2} - \tfrac{1}{2} = 0$$

So the probability of measuring one is zero — not because nothing was there, but because two contributions of equal size arrived with opposite sign. Constructive interference at the zero outcome, destructive at the one outcome.

Check yourself. Four numbers and two additions. Do them yourself. Having done the arithmetic once, you will trust the next page rather than take it on faith.

TOPIC 2 OF 5 · QUANTUM INTERFERENCE · SESSION 2



–H–Z–H– : one sign, opposite result

Now insert a Z between the two Hadamards. Z does exactly one thing: it flips the sign of the |1⟩ amplitude — a phase shift of π on that path, and nothing else.

Follow the flows again. The split is the same. The recombination is the same. But because one path now carries a minus sign, the additions swap: the contributions that used to cancel now add, and the ones that used to add now cancel. **The circuit that did nothing at all now flips the qubit.**

Three gates, one sign, opposite result. That is quantum interference, and it is the entire trick of quantum computing.

TOPIC 2 OF 5 · QUANTUM INTERFERENCE · SESSION 2

Session 2 Companion Notebook — the interference fringe

Sweep the phase and watch the output move. The exact curve, with shot-based estimates on top, so sampling noise is visible next to the theory.

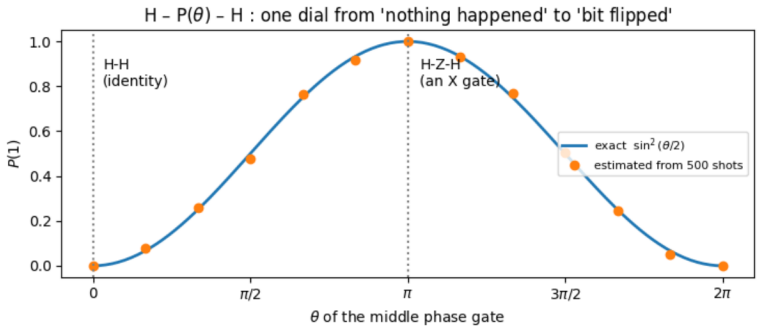
In [2]:

```
thetas = np.linspace(0, 2*np.pi, 200)
exact = []
for t in thetas:
    qc = QuantumCircuit(1); qc.h(0); qc.p(t, 0); qc.h(0)
    exact.append(probabilities(Statevector(qc).data)[1])

# a shot-based estimate at a handful of angles, to show sampling noise on top of the exact curve
sample_thetas = np.linspace(0, 2*np.pi, 13)
shots = 500
estimates = []
for i, t in enumerate(sample_thetas):
    qc = QuantumCircuit(1); qc.h(0); qc.p(t, 0); qc.h(0); qc.measure_all()
    counts = StatevectorSampler(seed=100+i).run([qc], shots=shots).result()[0].data.meas.get_counts()
    estimates.append(counts.get("1", 0) / shots)

fig, ax = plt.subplots(figsize=(7.5, 3.4))
ax.plot(thetas, exact, lw=2, label="exact  $\\sin^2(\\theta/2)$")
ax.plot(sample_thetas, estimates, "o", ms=6, label=f"estimated from {shots} shots")
ax.axvline(0, ls=":", c="grey"); ax.axvline(np.pi, ls=":", c="grey")
ax.annotate("H–H\n(identity)", xy=(0.1, 0.80)); ax.annotate("H–Z–H\n(an X gate)", xy=(np.pi+0.12, 0.80))
ax.set_xlabel(r"$\theta$ of the middle phase gate"); ax.set_ylabel(r"$P(1)$")
ax.set_xticks([0, np.pi/2, np.pi, 3*np.pi/2, 2*np.pi])
ax.set_xticklabels(["0", r"$\pi/2$", r"$\pi$", r"$3\pi/2$", r"$2\pi$"])
ax.set_title("H – P($\\theta$) – H : one dial from 'nothing happened' to 'bit flipped'")
ax.legend(loc="center right", fontsize=8)
plt.tight_layout(); plt.show()

print("Compare this curve with the one you plotted in Code Bite 2.2. It is the same physics.")
```



Compare this curve with the one you plotted in Code Bite 2.2. It is the same physics.

…and the whole dial, swept

The slides work two cases by hand: nothing in the middle, then a Z. The notebook sweeps the phase continuously and plots the result, so both cases turn out to be points on a single curve — H–H at the left, H–Z–H at the peak.

> **Check yourself.** The orange markers are 500-shot estimates laid over the exact curve. Sampling noise becomes visible as scatter around the theory, which is the first appearance of a question that runs through Sessions 4 and 5: how many shots is an answer worth?

TOPIC 3 OF 5 · ENTANGLEMENT · SESSION 3

CNOT & Entanglement: When Qubits Get Entangled

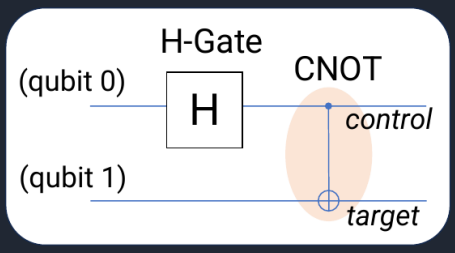
qubit-lab.ch

Besides single-qubit gates like H, X, Y, and Z, some quantum gates act on two qubits – with two inputs and two outputs.

(Remember: valid quantum gates always have equal numbers of input and output qubits)

A typical 2-qubit gate is the **CNOT or CX Gate** (Controlled-NOT)

It is a NOT gate (X-Gate) on the target qubit which will be activated only if the control qubit is in state |1>



CNOT Gate (Controlled NOT- or X-Gate)
qubit 0: control qubit, qubit 1: target qubit

- If the control qubit (qubit 0 here) is in state |1>, the target qubit's probability amplitudes will be swapped (NOT Gate)
- If the control qubit is in state |0>, the target qubit will not be changed

After this sequence, qubits 0 and 1 are entangled!

The CNOT, and the sequence that entangles

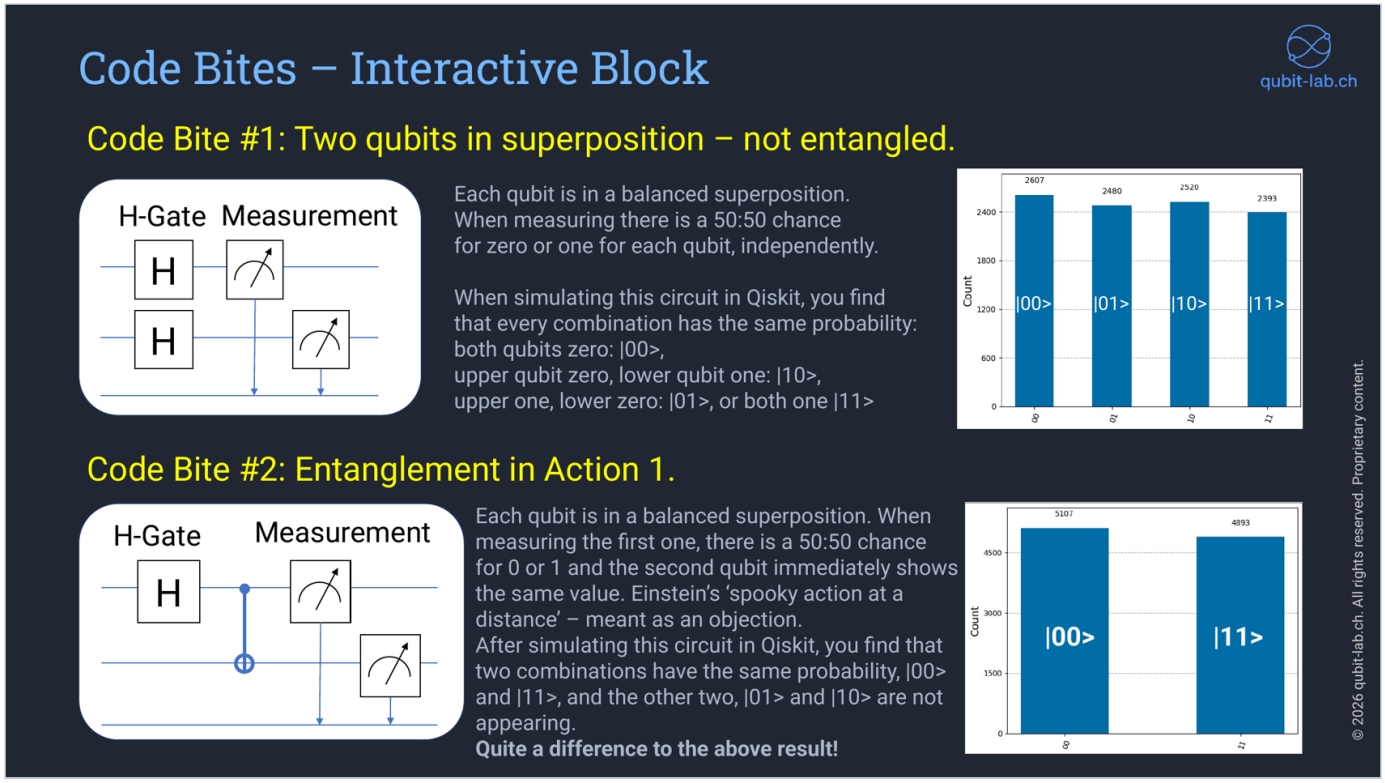
Until now every gate acted on one qubit. The **CNOT** — controlled-NOT, also written CX — acts on two. The rule is simple: if the control qubit is in state one, apply an X to the target; if the control is zero, do nothing. Since valid gates always have as many outputs as inputs, a two-qubit gate is a 4×4 matrix.

Now the sequence that matters: a Hadamard on the control, then a CNOT. After it, the two qubits are **entangled**.

Common pitfall. Resist defining entanglement as "spooky correlation". The operational definition is both correct and checkable: the state of the pair cannot be factored into a product of two single-qubit states. Neither half has a state of its own.

In the notebook. Code Bite 3.5 builds it; Code Bite 3.6 tests whether it is really entangled.

TOPIC 3 OF 5 · ENTANGLEMENT · SESSION 3



Two qubits, with and without a CNOT

The upper block is deliberately the boring one: a Hadamard on each of two qubits, no interaction. Each is independently 50/50, so all four combinations appear at 25% each. Nothing surprising — but it is the baseline you need.

The lower block adds the CNOT, and two outcomes vanish. Only $|00\rangle$ and $|11\rangle$ appear. Each qubit is still individually 50/50 — measure just one and you learn nothing — but the outcomes are now perfectly correlated.

Common pitfall. This histogram is exactly what two copies of a single coin flip would give. From this measurement alone you cannot tell that anything quantum is happening. The CHSH test in the notebook is how you actually prove it — and it needs measurements in other bases, which is why it could not appear any earlier.

Check yourself. Does this transmit information faster than light? No. The second qubit's own statistics are unchanged whether or not the first was measured. The correlation only appears once the two records are brought together.

In the notebook. Code Bites 3.1 and 3.5; the proof is Code Bite 3.8.

TOPIC 3 OF 5 · ENTANGLEMENT · SESSION 3

CNOT Gate – First Insights...

qubit-lab.ch

Using the CNOT gate, we cannot describe entangled qubits separately.
We need to see them as a joint system!

- In general, an entangled state cannot be written as a product of two single-qubit states.
- Need to introduce a new system basis:
  instead of |0> and |1>, use |00>, |01>, |10> and |11>
- Attach a probability amplitude to each of the new basis states:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

  and therefore, the state vector becomes: $|\psi\rangle \equiv \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$

- Consequently, multi-qubit quantum gates act on the joint state space and are represented by 4×4 matrices acting on the joint state vector

Writing the two-qubit state down

The state after that circuit, written properly: a superposition of the four basis states with four complex amplitudes, stacked into a column vector of length four.

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

For the Bell state just built, two of those amplitudes are 1/√2 and two are zero. That is the whole content of the histogram, in a form you can compute with.

> **Check yourself.** One qubit needed two amplitudes; two qubits need four. How many will three need? That doubling is the theme of the next four pages.

TOPIC 3 OF 5 · ENTANGLEMENT · SESSION 3

Session 3 Companion Notebook — the Bell state, built and measured

H then CNOT. The statevector, and 4,000 shots: only 00 and 11 ever appear.

In [2]:

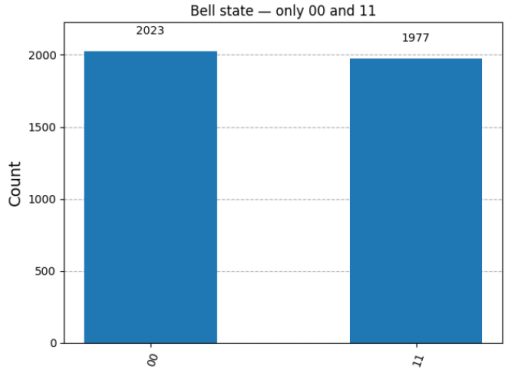
```
bell = QuantumCircuit(2)
bell.h(0)
bell.cx(0, 1)
show(bell, "Deck Code Bite #2 — H then CNOT:")

sv = Statevector(bell)
print("\nstatevector:", np.round(np.real(sv.data), 3))
for basis, p in sorted(sv.probabilities_dict().items()):
    print(f"  P({basis}) = {p:.3f}")

circ = bell.copy(); circ.measure_all()
counts = StatevectorSampler(seed=42).run([circ], shots=4000).result()[0].data.meas.get_counts()
display(plot_histogram(counts, title="Bell state — only 00 and 11"))
print("""|01> and |10> have vanished. The two qubits are still individually 50/50,
but their outcomes are now perfectly correlated.""")
```

```
Deck Code Bite #2 — H then CNOT:
q_0: ─H─●─
q_1: ───X─

statevector: [0.707 0.    0.    0.707]
  P(00) = 0.500
  P(11) = 0.500
```



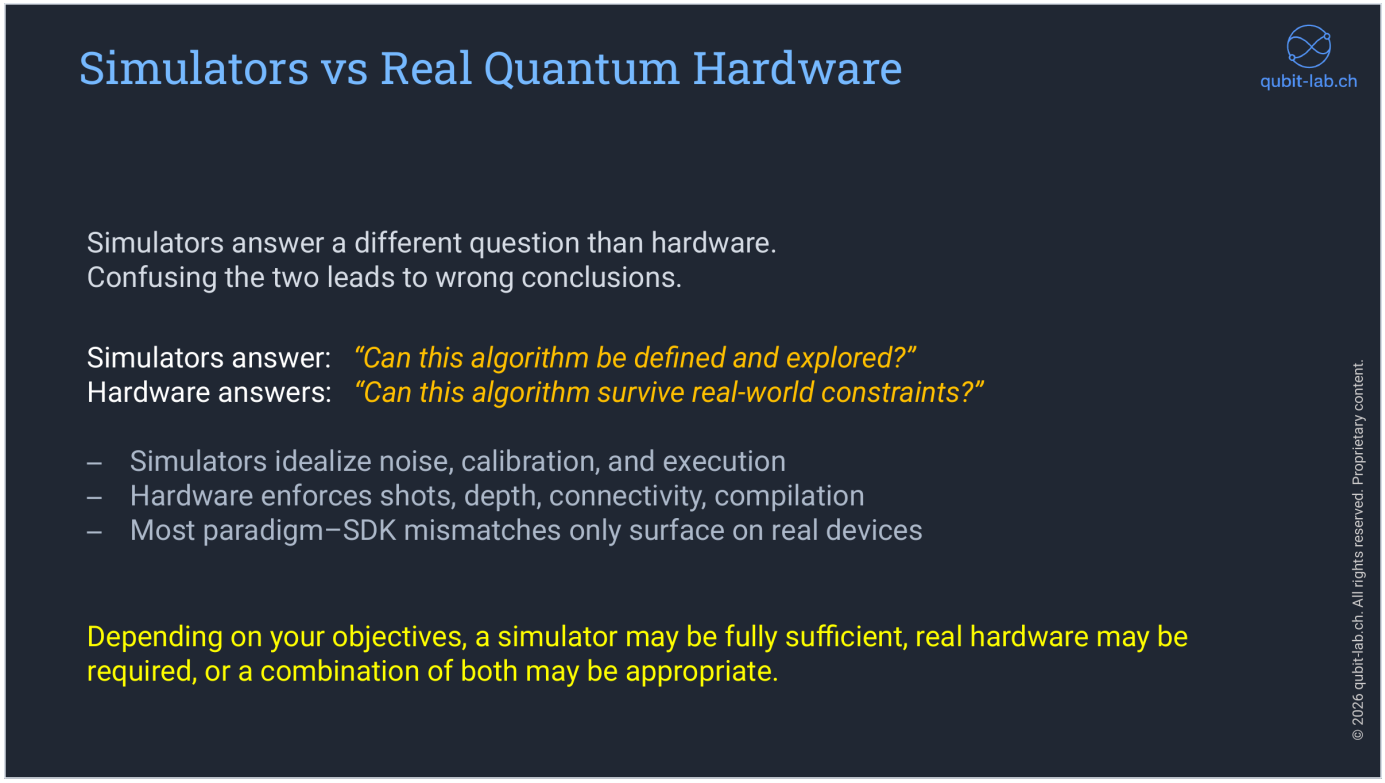

```
|01> and |10> have vanished. The two qubits are still individually 50/50,
but their outcomes are now perfectly correlated.
```

…and the Bell state, built and measured

A Hadamard, a CNOT, and 4,000 shots. The statevector is printed alongside the histogram, so the amplitudes and the measured frequencies can be compared directly: only 00 and 11 ever appear.

> **Common pitfall.** This histogram is also exactly what two copies of one coin flip would produce, which is why the notebook does not stop here. The CHSH test later in the same notebook is what actually separates entanglement from ordinary correlation.

TOPIC 4 OF 5 · SIMULATORS VERSUS HARDWARE · SESSION 4



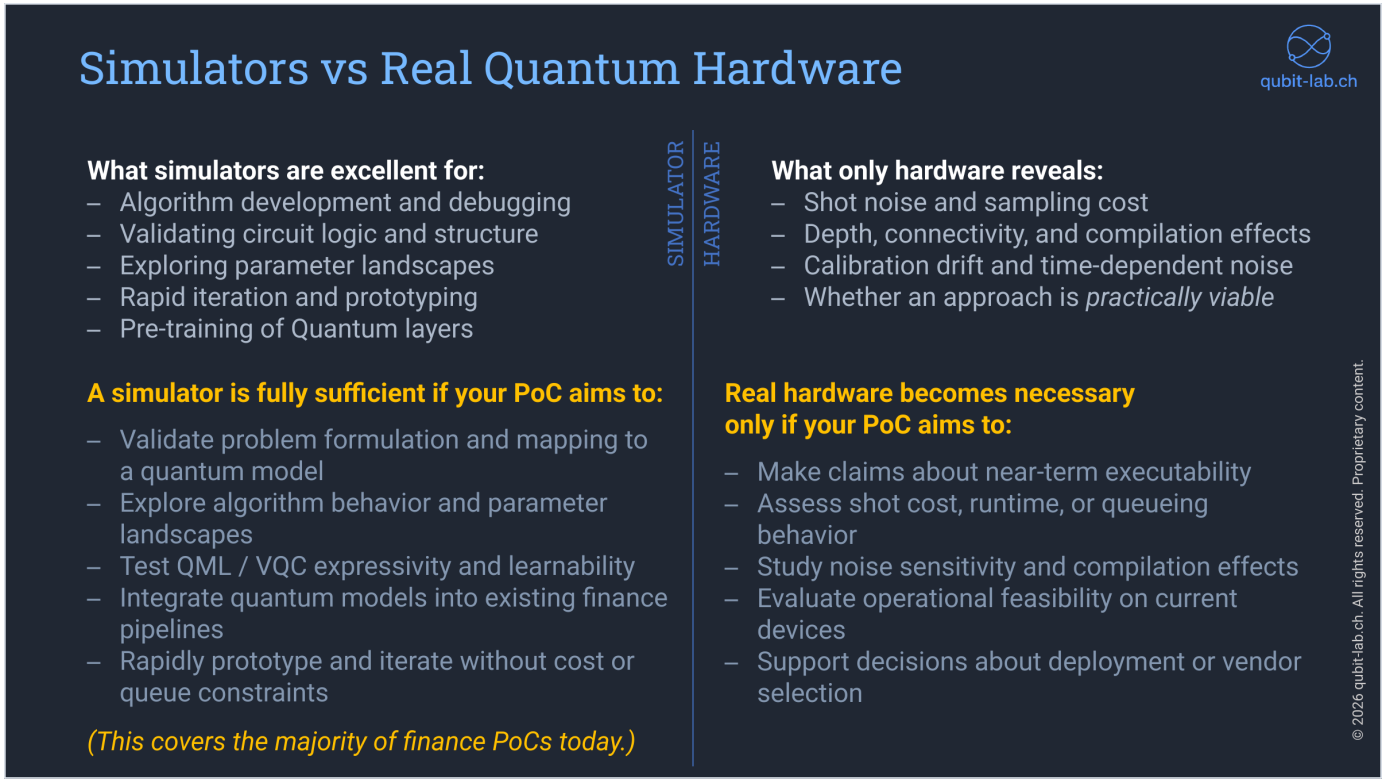
They answer different questions

Simulators answer: can this algorithm be defined and explored? Hardware answers: can this algorithm survive real-world constraints? Those are different questions, and confusing them is the most common error in the field.

> **Check yourself.** The sentence to use with management: **a simulator result is a statement about the algorithm; a hardware result is a statement about the machine.** Both are legitimate. Presenting the first as though it were the second is how quantum programmes lose credibility — usually by accident rather than intent.

Note the closing line of the slide and mean it: depending on your objective, a simulator may be entirely sufficient. That is not a consolation prize.

TOPIC 4 OF 5 · SIMULATORS VERSUS HARDWARE · SESSION 4



What each one is actually for

Simulators are excellent for development, debugging, validating circuit logic, exploring parameter landscapes and pre-training quantum layers. Only hardware reveals shot noise, compilation and connectivity effects, calibration drift, and whether an approach is practically viable.

The lower half is the part worth dwelling on: if your PoC is about problem formulation, algorithm behaviour, expressivity or pipeline integration, a simulator is sufficient — and that covers most finance PoCs being run today.

> **Common pitfall.** The slide leaves the cost dimension implicit. Simulators are free and instant; hardware has queues, per-shot pricing and calibration windows. A training loop that takes four minutes on a laptop can take days on a device — not because the device is slow per circuit, but because there are tens of thousands of circuits and a queue in front of each batch.

TOPIC 4 OF 5 · SIMULATORS VERSUS HARDWARE · SESSION 4

Typical Quantum PoC Objectives in Finance (2025/2026)

qubit-lab.ch

PoC type	Description	Typical objective	Simulator-based?	Hardware needed?	Comments
Concept feasibility PoC	Test whether a financial problem can be expressed as a quantum model (QUBO, circuit, feature map)	*Formulate the problem correctly*	✅ Yes	❌ No	Most common PoC today. Purely conceptual. No hardware claims.
Algorithm behavior PoC	Study convergence, expressivity, and failure modes of a quantum algorithm	*Understand algorithm dynamics*	✅ Yes	❌ No	Ideal use of simulators. Hardware adds little insight here.
QML expressivity PoC	Test whether a VQC/QNN can separate financial data	*Learnability and inductive bias*	✅ Yes	❌ No	Valid even without any hardware intent.
Baseline comparison PoC	Compare quantum vs classical models under controlled conditions	*Relative performance under ideal assumptions*	✅ Yes	❌ No	Must clearly label results as "idealized".
Pipeline integration PoC	Integrate quantum model into an existing ML / risk / pricing pipeline	*Workflow compatibility*	✅ Yes	❌ No	Often the real business value of QML PoCs.
Parameter transfer PoC	Train on simulator, evaluate on hardware	*Behavioral transfer test*	✅ Yes	⚠️ Optional	Valid exploration, **not** proof of readiness.
Noise sensitivity PoC	Assess how sensitive a model is to realistic noise	⚠️ Partial realism	⚠️ Sometimes	⚠️ Sometimes	Simulator noise models help, but hardware is more honest.
Hardware sanity PoC	Run a minimal version on real hardware	*Check executability*	❌ No	✅ Yes	Scope must be tiny. This is about survival, not performance.
Cost & scaling PoC	Estimate shots, depth, runtime, queueing cost	*Economic feasibility*	❌ No	✅ Yes	Impossible to do meaningfully on simulators alone.
Near-term viability PoC	Test whether a use case could plausibly run on current devices	*Operational realism*	❌ No	✅ Yes	Rare today. Often overclaimed.
FTQC readiness PoC	Study how an algorithm would scale under fault tolerance	*Long-term planning*	✅ Yes	❌ No	Uses logical-level simulation, not NISQ hardware.

PoC objectives, and what each one actually needs

A decision aid rather than a reference table. Note how many rows read simulator yes, hardware no — concept feasibility, algorithm behaviour, QML expressivity, baseline comparison and pipeline integration are all simulator work.

The parameter-transfer row is the honest middle: train on a simulator, evaluate on hardware, and call it a behavioural transfer test rather than proof of readiness.

> **Check yourself.** In most Swiss finance conversations the relevant row is either concept feasibility or pipeline integration — both firmly simulator-only. That is a far more comfortable conversation to have early than after someone has budgeted for hardware access they did not need.

TOPIC 4 OF 5 · SIMULATORS VERSUS HARDWARE · SESSION 4

Session 4 Companion Notebook — what a noisy simulator is for

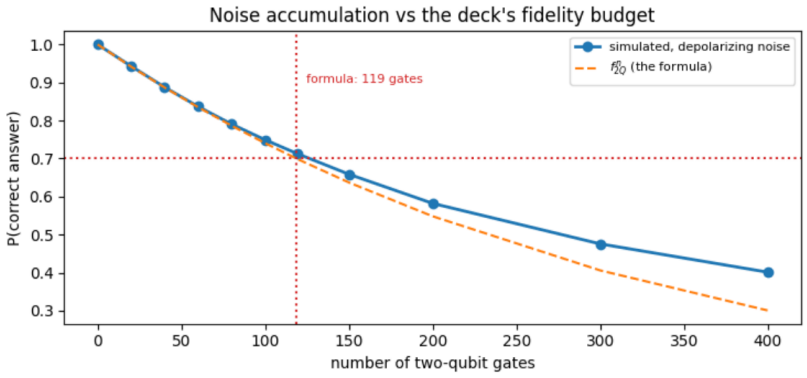
The same circuit, simulated with a realistic noise model. Measured points against the analytic prediction, so the deck's claim can be checked rather than believed.

In [2]:

```
fig, ax = plt.subplots(figsize=(7.5, 3.6))
ax.plot(depths, success, "o-", lw=2, label="simulated, depolarizing noise")
ax.plot(depths, [f_2q**n for n in depths], "--", lw=1.5, label=r"$f_{2Q}^{\,n}$ (the formula)")
ax.axhline(0.7, color="tab:red", ls=":", lw=1.5)
ax.axvline(predicted, color="tab:red", ls=":", lw=1.5)
ax.text(predicted+6, 0.9, f"formula: {predicted:.0f} gates", fontsize=8, color="tab:red")
ax.set_xlabel("number of two-qubit gates"); ax.set_ylabel("P(correct answer)")
ax.set_title("Noise accumulation vs the deck's fidelity budget")
ax.legend(fontsize=8); plt.tight_layout(); plt.show()

print("""
The simulation lands on the formula. That is the point of the slide made concrete: the
2Q gate budget is not a rule of thumb, it is multiplicative error compounding.

One caveat worth carrying into any vendor conversation: 'fidelity' is defined more than
one way (average gate fidelity, process fidelity, the depolarizing parameter used here).
The conversions differ by factors of order one, so quoted numbers from two vendors are
not automatically comparable — and neither are two papers.""")
```




```
The simulation lands on the formula. That is the point of the slide made concrete: the
2Q gate budget is not a rule of thumb, it is multiplicative error compounding.

One caveat worth carrying into any vendor conversation: 'fidelity' is defined more than
one way (average gate fidelity, process fidelity, the depolarizing parameter used here).
The conversions differ by factors of order one, so quoted numbers from two vendors are
not automatically comparable — and neither are two papers.
```

…and a noisy simulator, put to work

The slides argue that simulators and hardware answer different questions. This is a simulator answering one of them: the same circuit run at increasing depth under a realistic noise model, with the measured success rate plotted against the analytic prediction.

> **Check yourself.** The two curves meet where the formula says they should. That is what a noise simulator is for — not to imitate a particular machine, but to check whether a claim about one survives contact with arithmetic.

TOPIC 5 OF 5 · QUANTUM MACHINE LEARNING · SESSION 5

Kaggle Fraud Detection Data Sample

qubit-lab.ch

Real Data, 30 Features, 285k Transactions, 492 Fraud cases.

- Real transaction data; **28 features** were derived via Principal Component Analysis (PCA). *(The dataset has 31 columns in total: 28 PCA features + Time + Amount + Class)*
- **492 fraud cases** out of **284,807 transactions** (≈ **1 in 579**)
- **Highly imbalanced dataset.** For training the QNN, a **balanced subset of 984 transactions** (1:1 fraud vs. legitimate) will be created.
- Split the subset **80% / 20%** into training and test sets: 787 trainings & 197 test cases
- **Due to simulator limits, only the top 5 of the 28 PCA features** will be used (these five explain ~22% of the variance)

https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Data source used in Jupyter Notebook on qubit-lab.ch: Credit Card Fraud Detection dataset. Mirror via Zenodo: Liu, L. (2022). *creditcard.csv* [Data set]. Zenodo. https://doi.org/10.5281/zenodo.7395559 — **CC BY 4.0**. Original dataset: MLG–ULB, “Credit Card Fraud Detection” (Kaggle).

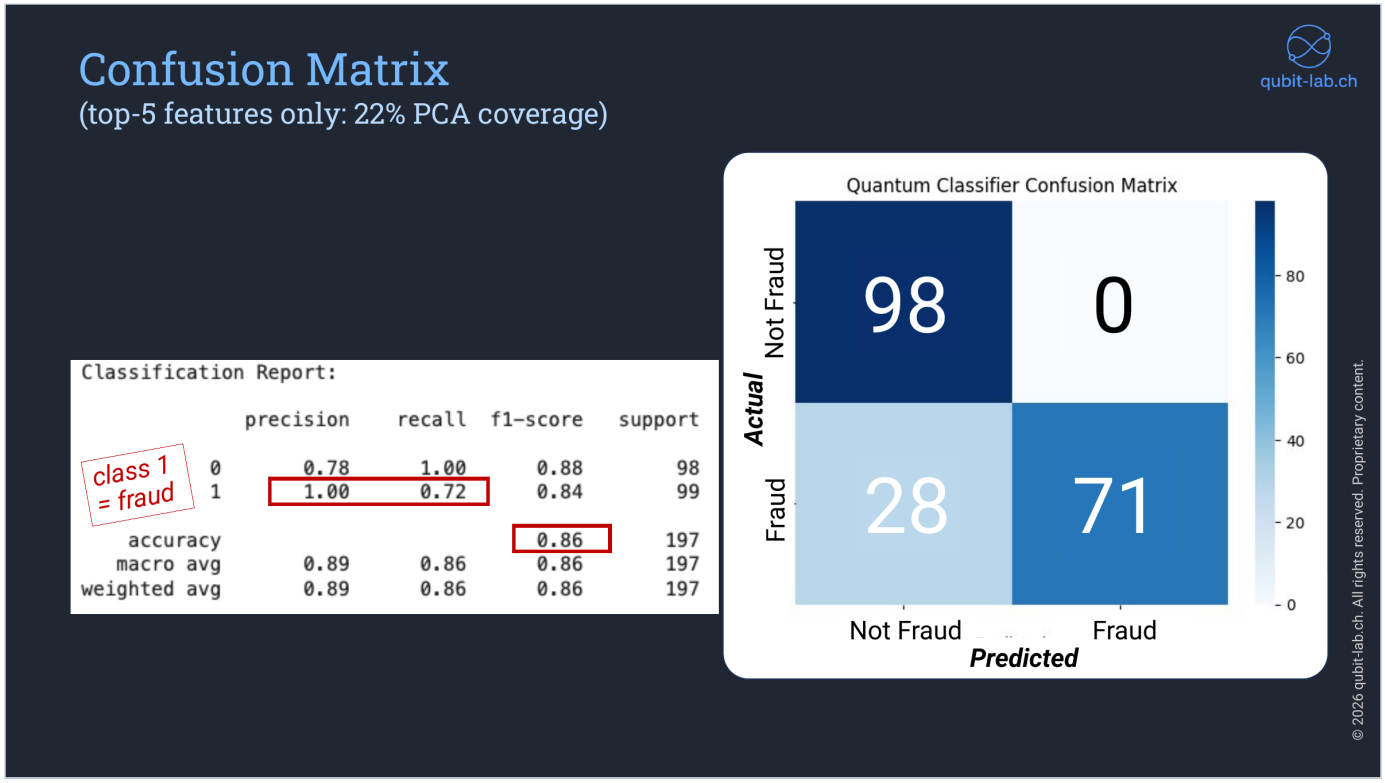
The dataset, and every compromise in it

Real transaction data: 28 PCA-derived features plus Time and Amount, 492 frauds in 284,807 transactions — roughly one in 579. A balanced subset of 984 transactions is built, split 80/20 into 787 training and 197 test cases. And because of simulator limits only the **top five** PCA features are used, explaining about 22 percent of the variance.

Common pitfall. That last constraint is severe, and every result must be read in its light. This is a demonstration of a pipeline, not a fraud model anyone should deploy.

Check yourself. The balanced-subset choice is a modelling decision with consequences worth naming. Training on a 1:1 subset makes learning tractable but means the model's calibration no longer reflects the real base rate, so its output probabilities are not directly usable. In production you would re-calibrate. Stating this is what a careful PoC write-up looks like.

TOPIC 5 OF 5 · QUANTUM MACHINE LEARNING · SESSION 5



The result, read honestly

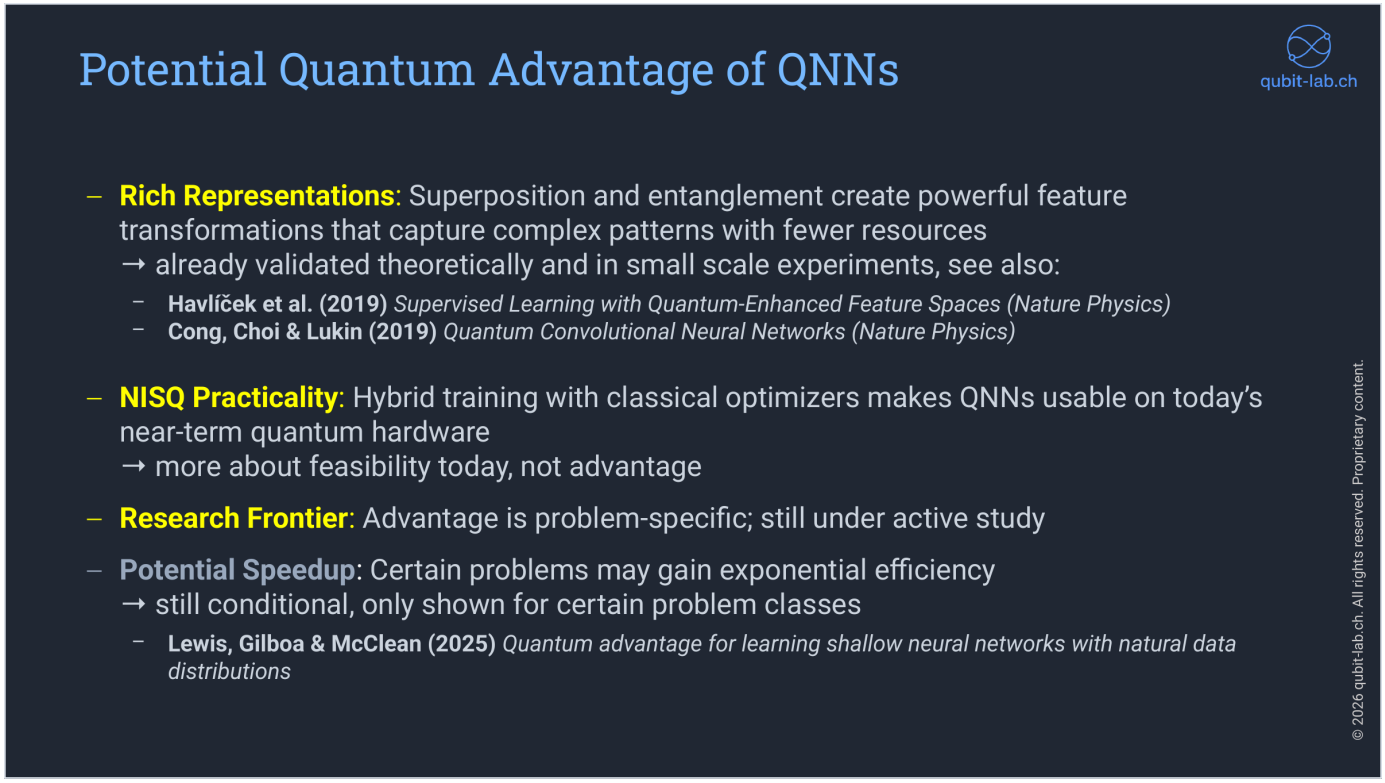
98 true negatives, zero false positives, 28 false negatives, 71 true positives. Precision is perfect — everything flagged as fraud was fraud. Recall is roughly 72 percent, so about a quarter of frauds were missed.

On five features covering 22 percent of the variance, that is a respectable demonstration that the pipeline learns something real. It is not a claim of advantage, and the missing number — the same five features through a classical classifier — is the first thing a reviewer will ask for.

Common pitfall. One further observation. Zero false positives out of 98 is suspiciously clean on a test set this small, and the right response is to say so and check it rather than celebrate it. A confidence interval on 197 cases is wide. **Being suspicious of your own good result** is the most valuable habit in the course.

In the notebook. Code Bite 5.9 runs the classical baseline on identical features.

TOPIC 5 OF 5 · QUANTUM MACHINE LEARNING · SESSION 5



Potential advantage — with the qualifiers intact

The annotations are the honest part of this slide, so read them rather than the headings. **Rich representations**: validated theoretically and in small-scale experiments (Havlíček 2019; Cong, Choi and Lukin 2019). **NISQ practicality**: about feasibility today, not advantage. **Potential speedup**: still conditional, shown only for certain problem classes (Lewis, Gilboa and McClean 2025). **Research frontier**: advantage is problem-specific and under active study.

> **Check yourself.** Asked directly whether QNNs are useful today, give the layered answer this slide supports: the mechanism is real and demonstrated at small scale, the training is feasible on current hardware, and the advantage is problem-specific, conditional and unproven on financial data. Anyone who compresses that into a yes or a no is selling something.

TOPIC 5 OF 5 · QUANTUM MACHINE LEARNING · SESSION 5

Session 5 Companion Notebook — the classifier, benchmarked

The same variational classifier against three classical baselines, on two controlled datasets: one generated by the quantum feature map itself, one ordinary tabular data.

In [2]:

```
# ---- Dataset A: labels generated by the quantum feature map ------------------
rng = np.random.default_rng(2)
XA = rng.uniform(-np.pi, np.pi, (240, n_qubits))
EA = encode(XA)
V  = Operator(real_amplitudes(n_qubits, reps=2, entanglement="full")
              .assign_parameters(rng.normal(0, 1, 12))).data
score = (np.abs(EA @ V.T)**2) @ Zall
keep  = np.abs(score) > 0.10                      # drop ambiguous points (Havlicek et al. do this too)
XA, EA, score = XA[keep], EA[keep], score[keep]
yA = (score > 0).astype(int)

# ---- Dataset B: ordinary tabular data, non-linear structure ------------------
XB = rng.uniform(-np.pi/2, np.pi/2, (240, n_qubits))
yB = ((XB[:, 0] > 0) ^ (XB[:, 1] > 0)).astype(int)     # XOR; features 2,3 are noise
EB = encode(XB)

datasets = {"A: quantum-generated": (XA, yA, EA), "B: ordinary tabular": (XB, yB, EB)}
scores = {}; trained = {}

for name, (Xd, yd, Ed) in datasets.items():
    idx = np.arange(len(yd))
    itr, ite = train_test_split(idx, test_size=0.3, random_state=7, stratify=yd)
    theta = train_vqc(Ed[itr], yd[itr])
    acc_q = float(np.mean(classify(theta, Ed[ite]) == yd[ite]))
    acc_lr = LogisticRegression(max_iter=2000).fit(Xd[itr], yd[itr]).score(Xd[ite], yd[ite])
    acc_svm = SVC().fit(Xd[itr], yd[itr]).score(Xd[ite], yd[ite])
    acc_rf = RandomForestClassifier(n_estimators=300, random_state=0).fit(
        Xd[itr], yd[itr]).score(Xd[ite], yd[ite])
    scores[name] = (acc_q, acc_lr, acc_svm, acc_rf)
    trained[name] = (itr, ite, theta)
    print(f"{name:24s} n={len(yd):3d}   VQC {acc_q:6.1%} | LogReg {acc_lr:6.1%} | "
          f"SVM {acc_svm:6.1%} | RandomForest {acc_rf:6.1%}")
```

```
A: quantum-generated     n=163   VQC  87.8% | LogReg  61.2% | SVM  53.1% | RandomForest  65.3%
B: ordinary tabular      n=240   VQC  48.6% | LogReg  61.1% | SVM  90.3% | RandomForest  95.8%
```

…and the classifier, put next to the classical field

The slides report a fraud-detection result and then qualify it carefully. The notebook runs the controlled version: the same variational classifier against logistic regression, an SVM and a random forest, on two datasets built for the purpose — one generated by the quantum feature map itself, one ordinary tabular data with an XOR structure.

Common pitfall. The two output lines are the honest summary of quantum machine learning in 2026. On data the feature map generated, the quantum classifier wins comfortably. On ordinary tabular data it is beaten by every classical baseline, a random forest most of all. Both lines are printed, and neither is hidden.

Check yourself. This is what a classical referee looks like when the answer is unflattering. A course that only ever shows the first line is selling something.

What the full programme contains

Five sessions of three to four hours each, delivered live. Session 1 — single-qubit states and gates. Session 2 — quantum interference. Session 3 — multi-qubit systems and entanglement. Session 4 — frameworks, simulators and hardware. Session 5 — key quantum algorithms for finance.

256 slides across the five decks, every one carrying presenter notes: 32,600 words written so that a second trainer can deliver the material without reverse-engineering the intent of each slide.

Five extended handouts, 201 pages in total, of which fifteen slide pages are sampled here. **Five companion notebooks**, 142 runnable code cells with exercises and worked solutions, of which five pages are sampled here.

What a sample cannot show

One topic per session is roughly a tenth of the programme, and the parts left out are the ones clients tend to quote back afterwards: what a circuit costs on real hardware and how deep it can be before noise wins; how to read a vendor specification sheet without being led by it; where quantum optimisation stands against the classical heuristics it actually competes with; how LLM-generated quantum code fails, silently and reliably; and what post-quantum cryptography requires of a bank, independent of when useful quantum computers arrive.

Those sections are what make a course decision-grade rather than merely interesting, and they are hard to judge from an excerpt. They are best walked through live.

About

qubit-lab.ch builds quantum prototypes and training for financial institutions, with an emphasis on verifiable results and classical baselines. **Evidence, not promises.**

contact@qubit-lab.ch · qubit-lab.ch