GETTING QUANTUM RIGHT

Successfully creating quantum computers means scientists have to overcome unique forms of noise and mistakes. How can they use ideas from classical information theory to do this?

Quantum computers have been the next big thing in computing for over two decades now. Capable of much more powerful performance than classical systems by utilising the concept of quantum superposition (the idea that an electron, for example, is in different states at the same time until observed), and therefore useful for number-crunching in cryptography and complex modelling, the main issue that implementing these systems in the real world is noise – from heat affecting superconducting components, to the fact that setting up circuits for algorithms often relies on probabilistic assumptions. Overcoming noise and errors in data is nothing new in the field of information, with Claude Shannon outlining his noisy-channel coding theorem in 1948.: "*In a binary communication channel, it is possible to find an error correcting code with rate arbitrarily close to the channel's capacity, which has an error probability* $< \delta$ *for any* $\delta > 0$."

The lessons learnt from ensuring that messages sent through a binary communication channel are still readable at the other end are certainly applicable to work with quantum systems – but there are still unique challenges to be overcome.

The units of quantum computing are known as qubits - 2-level quantum mechanical systems, where the levels are used to represent the 0 and 1 states, commonly written using the Dirac Bra-Ket notation as $|0\rangle$ and $|1\rangle$. In the same way as classical computers, they are linked together in quantum circuits and controlled using logic gates to run algorithms.

The computers often rely on delicately balanced lasers or magnetic fields for these gates. It's also almost inevitable that the qubits will become intertwined in some way with the environment, which causes the state to change in ways that aren't predictable or wanted. All this means that it's necessary to be able to find errors as efficiently as possible, especially as the number of qubits in a computer increases and there is more scope for errors.

Error-correcting codes

The aim of an error-correcting code is ensuring that a mistake in data can be reversed. Maybe the simplest way of doing this for a binary string is to repeat each bit several times –

How Do You Make a Qubit?

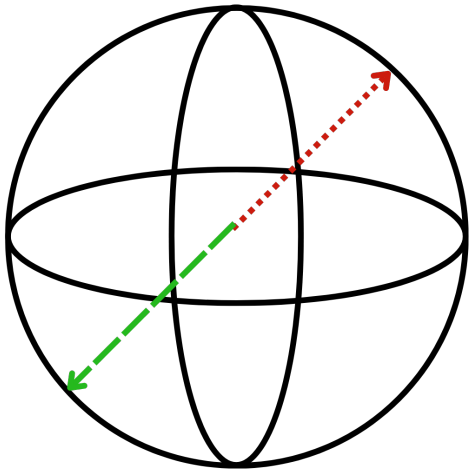
Physically there are many different options for making the qubits for a quantum computer: all that is needed is a 2-level quantum system

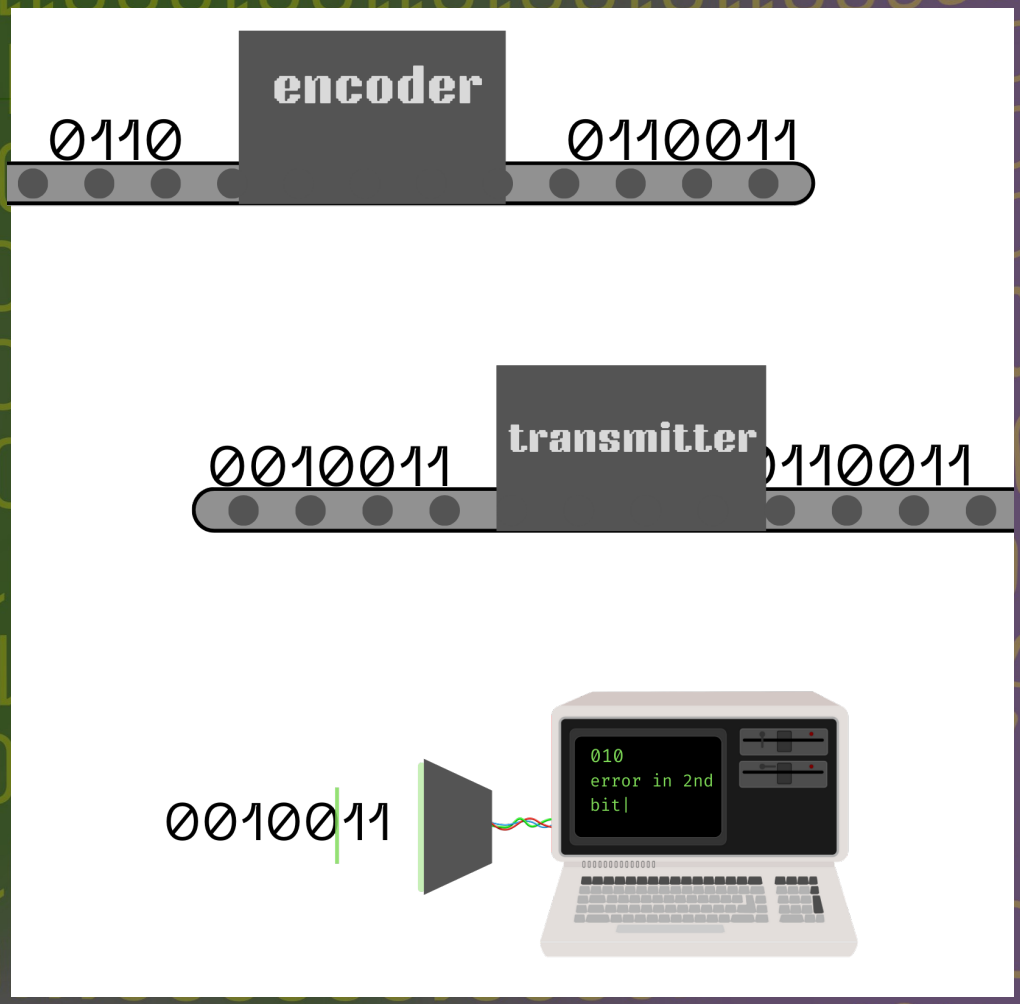
Superconducting computers use Josephson Junctions, where a thin strip of non-conducting material is placed between 2 superconductors, causing a current to flow. Different energy levels are used to represent the 0 and 1 states. However, superconductors typically need very cold temperatures to work, which can be impractical.

Trapped ion qubits hold ions in place using electrodes and radio frequencies. Different energy levels of the ion are used to represent the different states. Control is done through lasers, meaning that instability in the lasers could cause gate errors.

Photonics uses light in order to create qubits, utilising polarisation or the path a photon takes after a beam splitter to represent different states.

Systems that trap separate electrons and use spin-up and spin-down are also commonly used as qubits.





so if the message 00010 is received, it's safe to assume that the original message was 0 – but this isn't very efficient. It's better to break up the message into blocks of a few bits and find a way to encode each of these, in such a way that even if a 0 is changed to a 1, the block is still closer to the correct coded message than any other possibility. When the message is received, a parity check operation is run on it, which returns an *error syndrome* telling the receiver what mistakes have occurred. The easiest way to discuss these is using algebra: the parity checks and encoding are each done by a matrix, and the message and codewords are vectors.

Creating a way of protecting data in quantum computing starts with similar ideas. But this is where a key difference between classical and quantum data arises: rather than the familiar 2 possible values, it is easier to think of each qubit taking values from the surface of a unit sphere. The north and south poles are $|0\rangle$ and $|1\rangle$, and the ends of the x-axis are $|+\rangle$ and $|-\rangle$. Outside interference could move it to anywhere else on the sphere, so a code needs to detect more than one simple type of error. A further difficulty comes from the "No-Cloning" theorem, which prevents us from repeating an arbitrary quantum state to protect against errors – creating an identical copy is only possible for *orthogonal* states.

One common way of constructing a code to account for multiple types of error uses two classical codes – one of which contains all the code-words of the other as well as additional words, combining them to correct the effect of a qubit being rotated around the y axis – a "bit-flip" and being rotated around the z axis- a "phase-flip". Codes made in this way are known as Calderbank-Shor-Steane codes.

Firstly, the possible code vectors from the first code are split into groups of n according to their relation to the vectors from the second group; then each group is used to define a unique code state:

$$|c\rangle = \frac{1}{2^n} \sum code\ states\ in\ group$$

When the message is received, bit and phase flip errors are checked for in turn. The parity check operation for the first code tells us whether any bits were flipped. Checking for a phase-flip takes a bit longer because it is modelled as the exponent of a factor of -1. But applying the correct quantum logic gate switches the exponents and linear parts and once again, a parity check operation can be used – this time, it's for the *dual* of the second code, which is obtained by using the parity check matrix as an encoder and vice versa. If there is an error, it is corrected, and then the same logic gate returns the message to the original state.

These codes aren't guaranteed to get the right error every time: usually, 2 or more potential errors will share the same syndrome. Considering probabilities lets us makes assumptions about which error happened to cause a particular syndrome. If there is only a small chance p of an error happening on a qubit, then it is most likely that no error occurred, and that only 1 error occurred rather than 2 or more. So most of the time, assuming that an error syndrome was caused by the smallest number of errors possible will be correct.

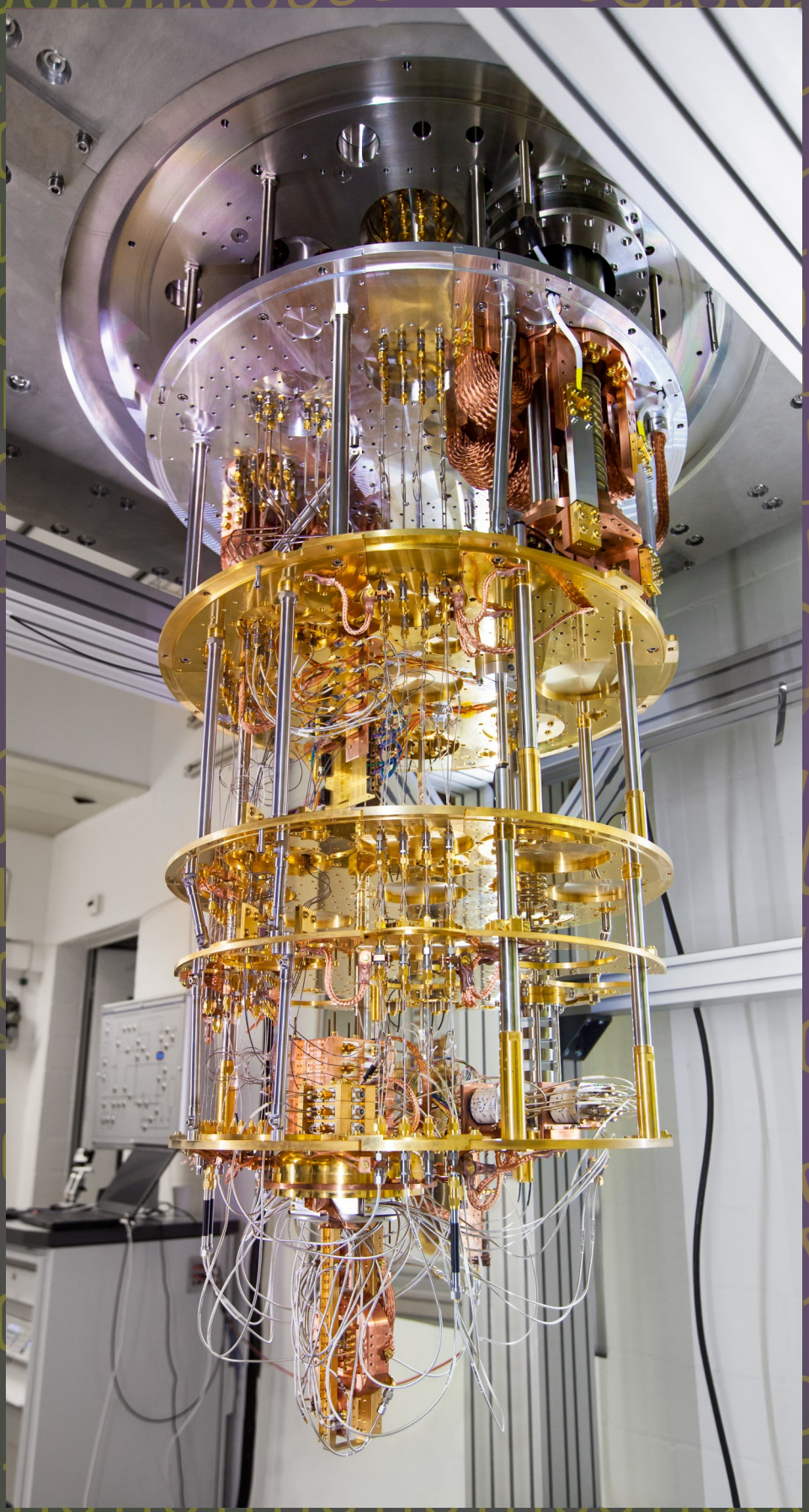
Applying codes

Encoding isn't just useful for sending quantum data through a communication device – it is also used to reduce the number of mistakes that occur as a program is running on a computer, in a process known as fault tolerance. The aim of fault tolerance is to stop errors spreading – if an error occurs in one collection – *block* - of qubits, it

should only cause at most one error on each block interacted with, preventing an exponential cascade in the number of mistakes. The No-Cloning theorem implies that it isn't possible to make back-up copies of data in the middle of a computation, which makes it even more important to prevent as many errors as possible.

One approach to fault tolerance is for quantum computers to use layers of encoded blocks of qubits to perform computations. For example, each qubit used could be represented by 5 physical qubits by encoding its state, using each physical qubit to store one part of the resulting data. Doing this requires all the logic gates for the computation to be encoded so that they match. How easy this is to do depends on the gate, and what codes were chosen for the fault-tolerance – sometimes, such as for the controlled-NOT gate, it's enough to just apply the gate to each physical qubit in turn, which is fault tolerant because each iteration of the gate only interacts with one physical qubit. For other gates, a more complicated set-up is required. Encoding the $\frac{\pi}{8}$ gate involves setting up a superposition of superimposed states, performing a measurement and then applying more gates depending on what the result was.

Current research in fault tolerance includes working specifically with the properties of different types of qubits, and finding ways of applying gates in parallel rather than sequentially. The faster a computer can work, the better, and overcoming the problem of noise will let the devices reach their full potential. Researchers have to consider which ways of running quantum computers are best for commercial use as well. Most forms of computer need the physical apparatus to be kept close to absolute zero, which isn't very practical for future off-the-shelf models. Photon based computers are an exception to this which has made them a popular choice for more current research.



A quantum computer at the IBM lab in Zurich (licensed under Creative Commons: IBM Zurich Lab Quantum Computer by IBM Reseearch under CC BY-ND 2.0.)