

Introduction to Quantum Computing

Report

Qubit code / measurements

Mihai-Tiberiu Dima

Hyperion University, Bucharest – Romania



Contents

- 1 *Quantum mechanics; TRANSMON qubits; read/set*
- 2 *ROOT package*
- 3 *HYBRILIT experience; SU2 package*
- 4 *Qubit measurements*
- 5 *Quant-gates; Groover algorithm*



Contents

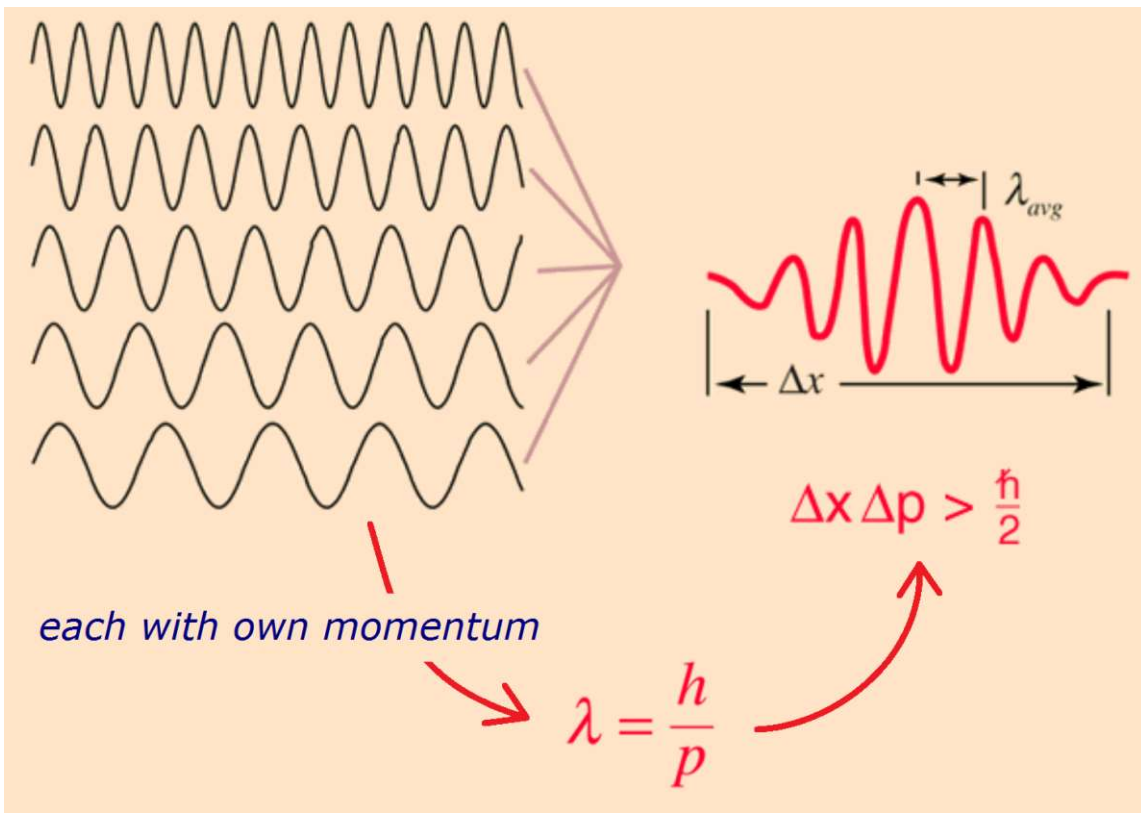
- 1 *Quantum mechanics; TRANSMON qubits; read/set*
- 2 *ROOT package*
- 3 *HYBRILIT experience; SU2 package*
- 4 *Qubit measurements*
- 5 *Quant-gates; Groover algorithm*



Ondulatory behaviour

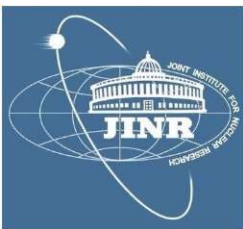
particles have wavelength : $\lambda = h / p$
... and a wavefunction : $|\psi\rangle = \text{Hilbert-space vec}$

Superposition of states



overlap of state ϕ onto ψ :
prob% = $|\langle \phi | \psi \rangle|^2$

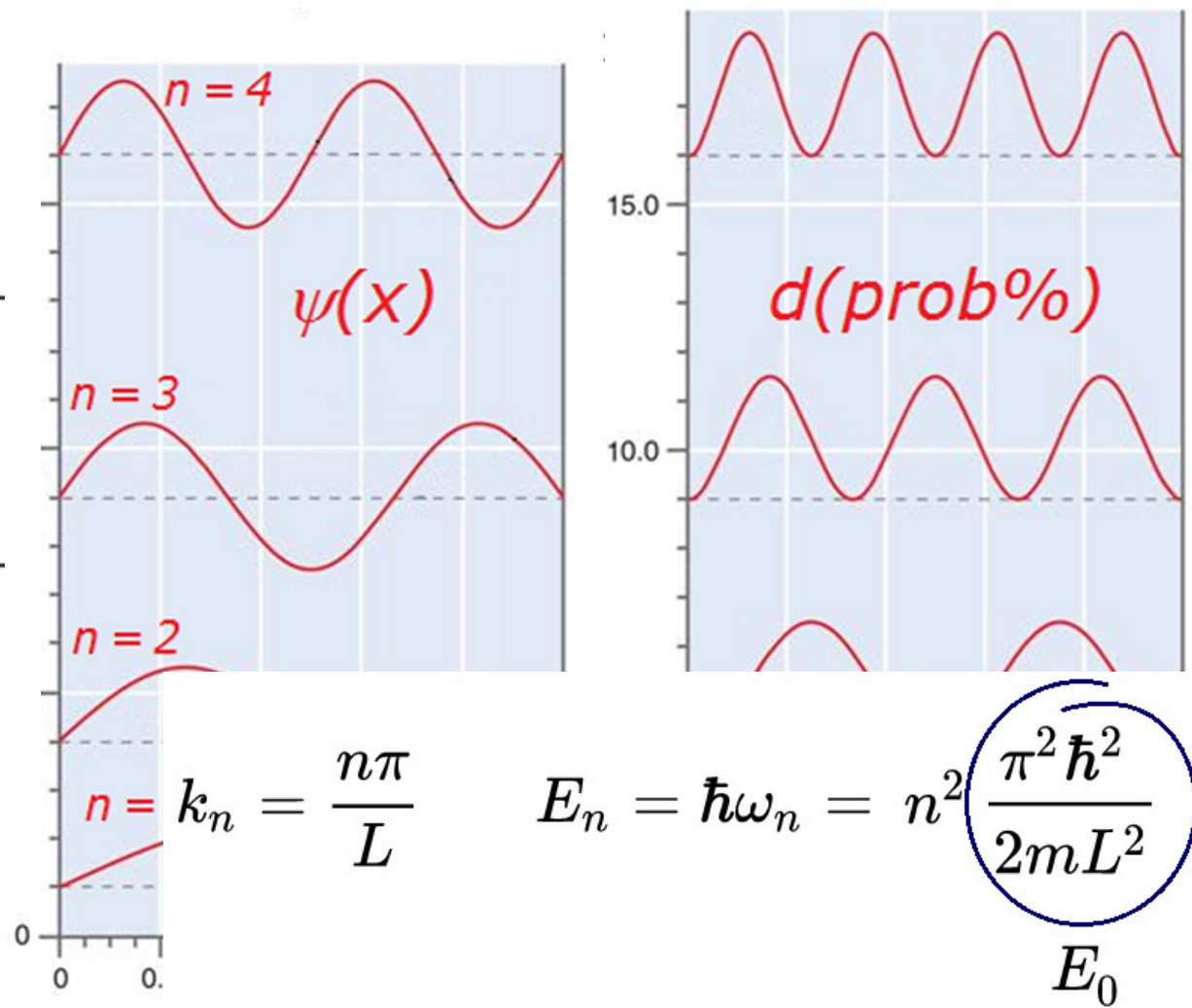
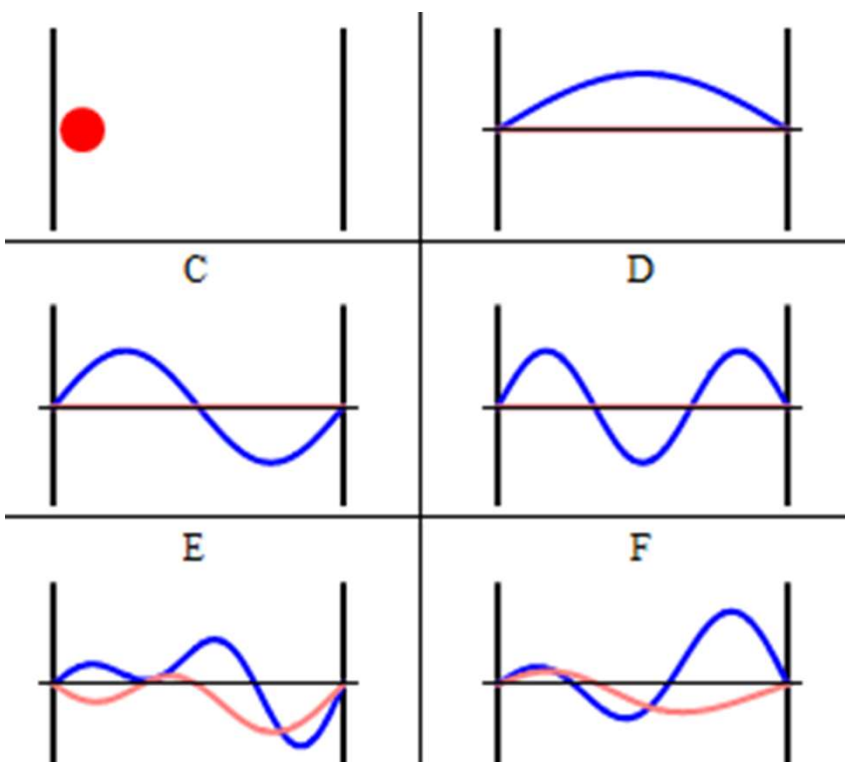
- of uncertain momentum and location
- Heisenberg uncertainty



Quantisation

Schrödinger equation

$$i\hbar \frac{\partial}{\partial t} \psi(x, t) = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \psi(x, t) + V(x)\psi(x, t)$$



Stern-Gerlach experiment

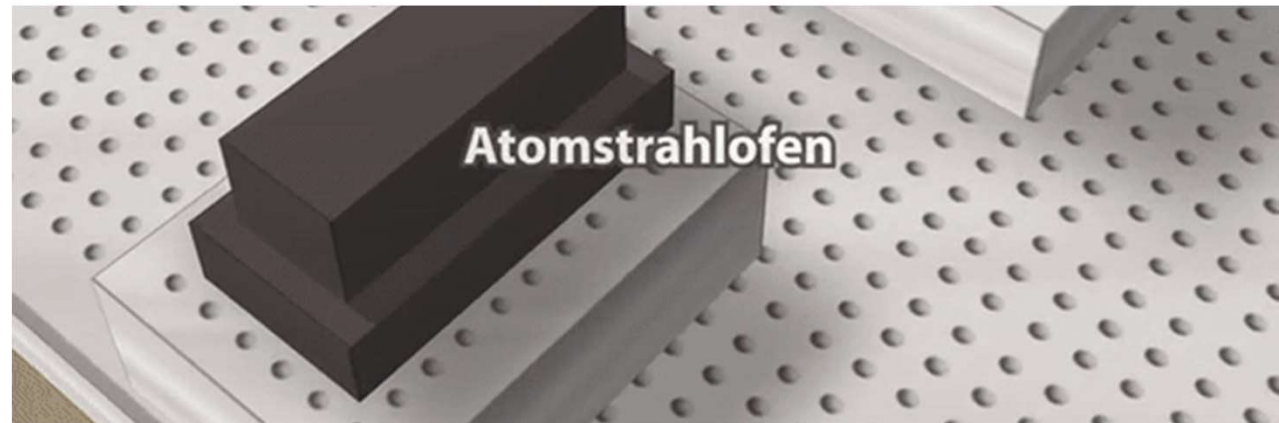
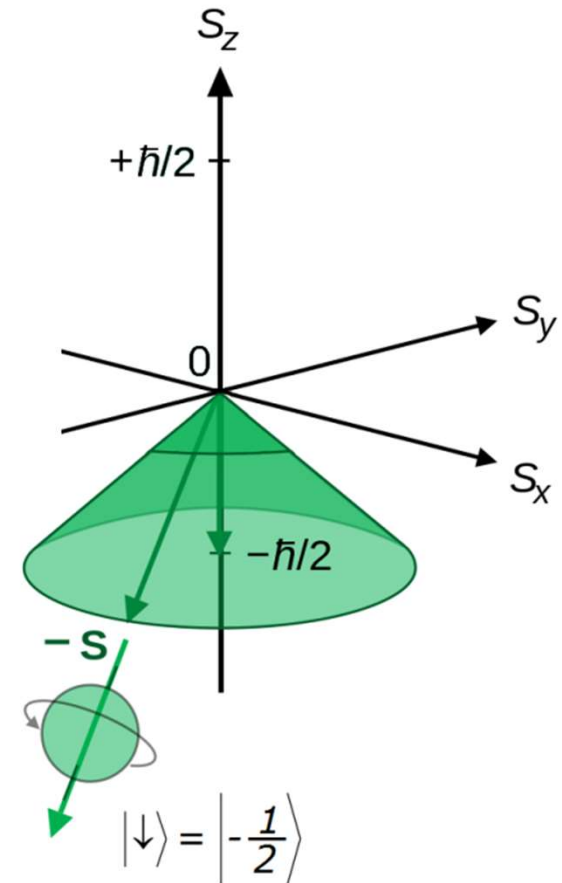
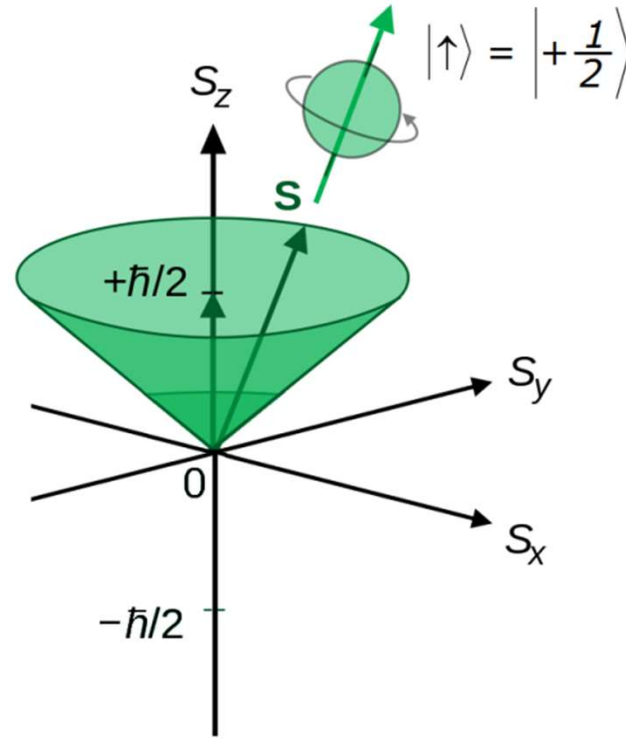
- electron has intrinsic spin
- that is quantised \uparrow or \downarrow

$$H = -\vec{\mu} \cdot \vec{B} = -\mu\vec{\sigma} \cdot \vec{B}$$

$$\vec{\sigma} \times \vec{\sigma} = 2i\vec{\sigma}$$

$$-|\leftarrow\rangle + |\rightarrow\rangle = \sqrt{2}|\uparrow\rangle$$

pure state in one base is superposition in another



Transmon qubits

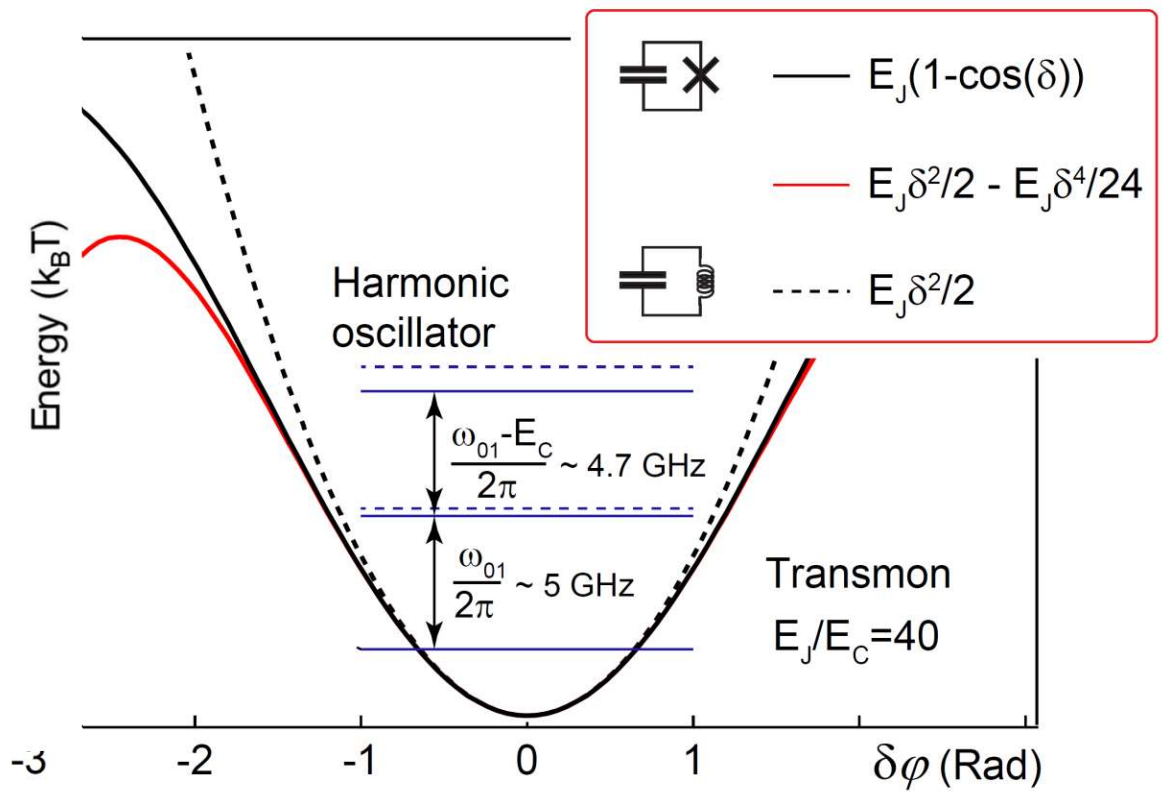
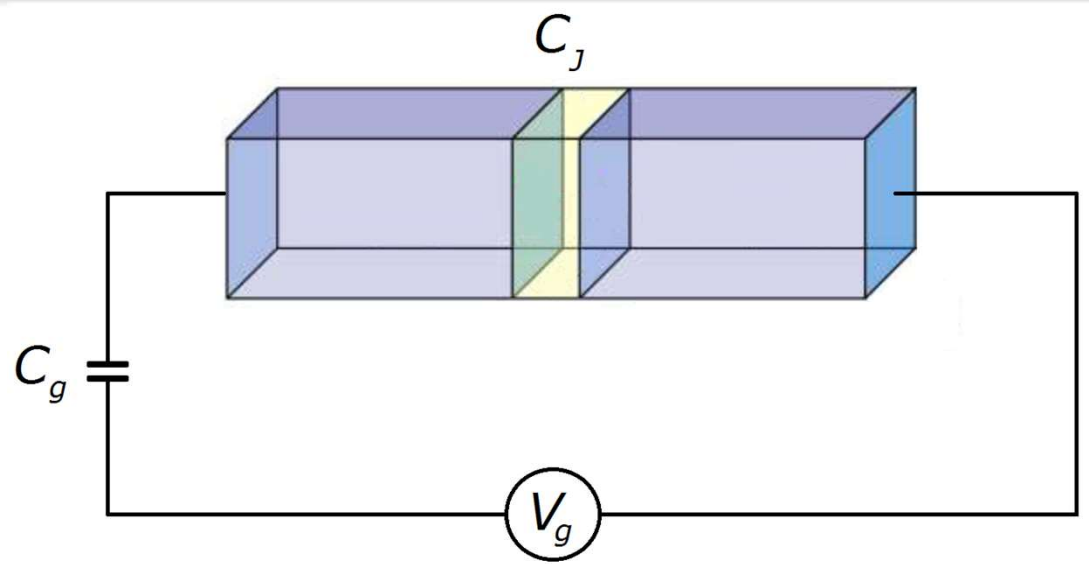
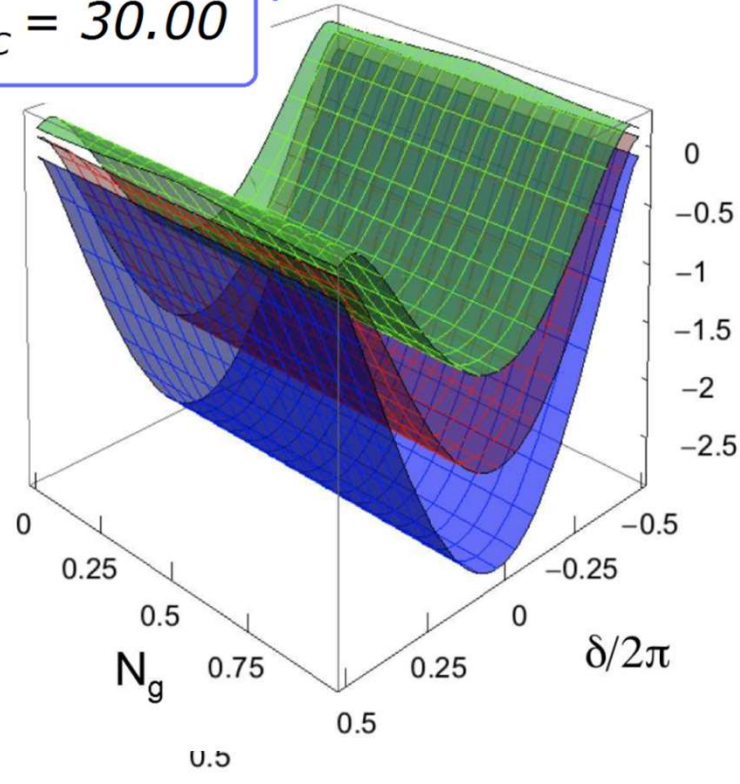
Cooper-pair Box

- 2 superconductors
- ca. 1 nm insulator

$$H = 4E_C(n - n_g)^2 - E_J \cos \varphi$$

$E_J / E_C = 30.00$

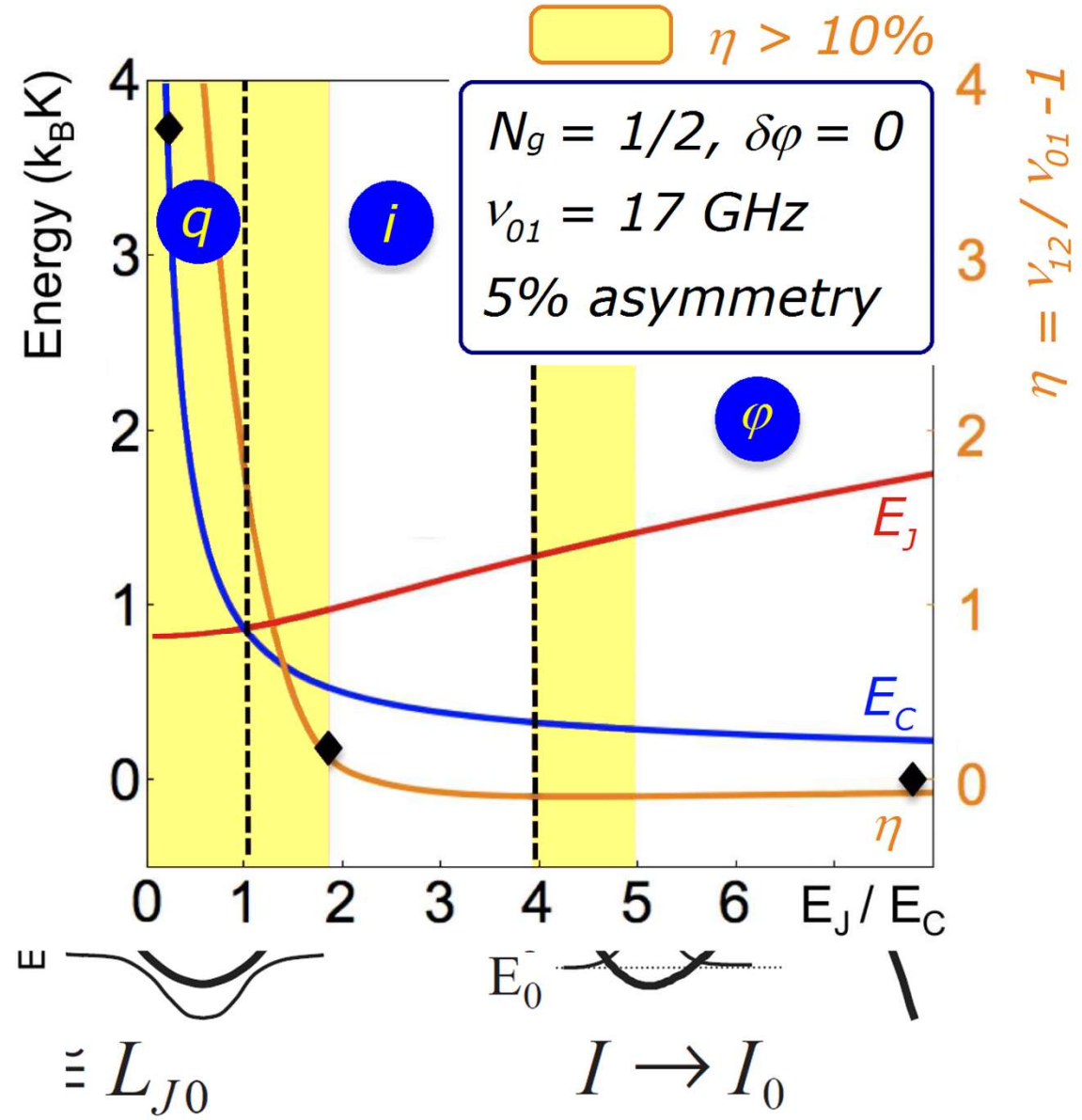
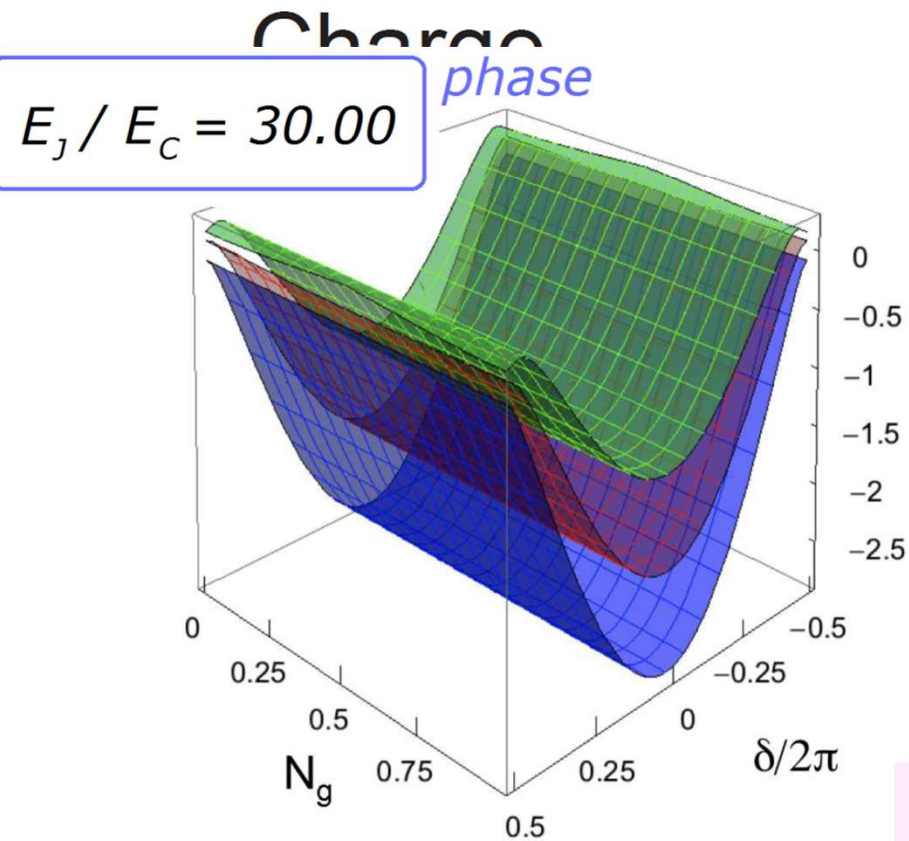
phase



Transmon qubits

Transmon qubit

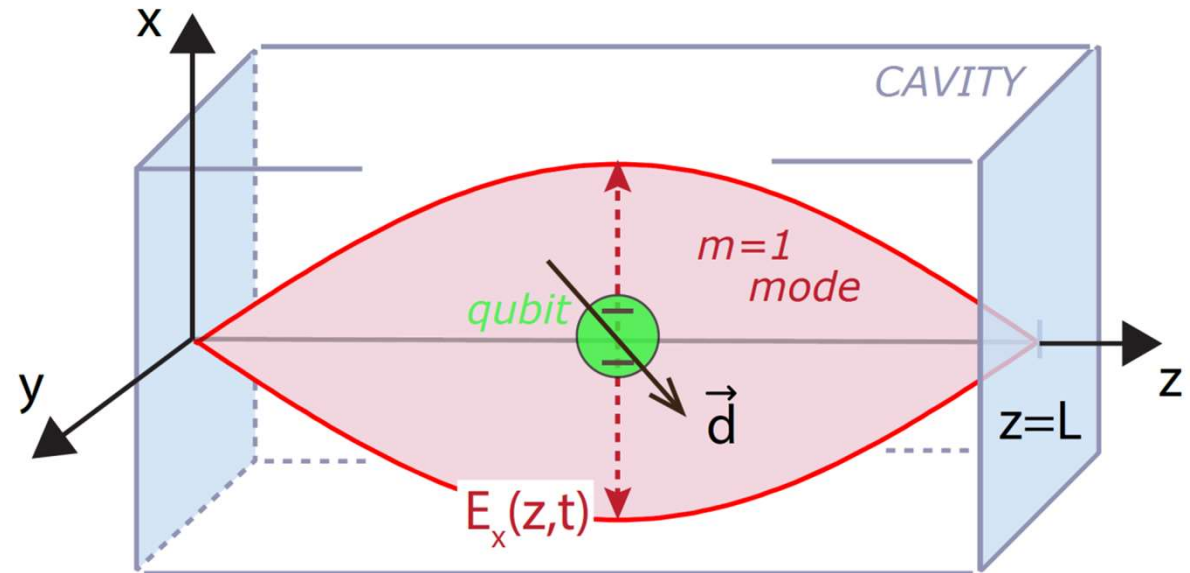
- anharmonicity engineered
- immune to V_g variations
- phase-state qubit



transm.-line shunted plasma oscillation qubit

Microwave cavity

- fundamental mode
- interaction w/ qubit dipole



$$H_{int} = -d \cdot E_x$$

$$= -d_x \mathcal{E}_0 (\hat{a} + \hat{a}^\dagger)(\sigma_+ + \sigma_-)$$

DRESSED states

$$|0, -\rangle = |g, 0\rangle$$

$$|n, -\rangle = \cos(\theta_n) |g, n+1\rangle - \sin(\theta_n) |e, n\rangle$$

$$|n, +\rangle = \sin(\theta_n) |g, n+1\rangle + \cos(\theta_n) |e, n\rangle$$

ground state

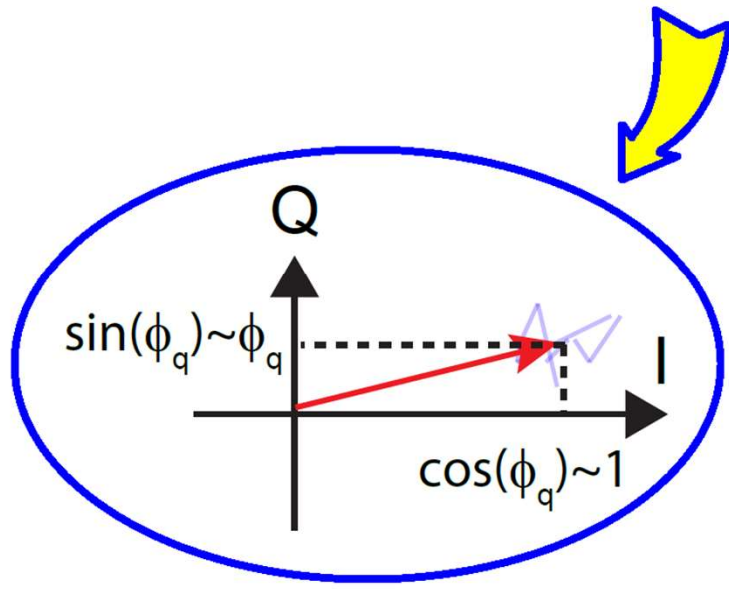
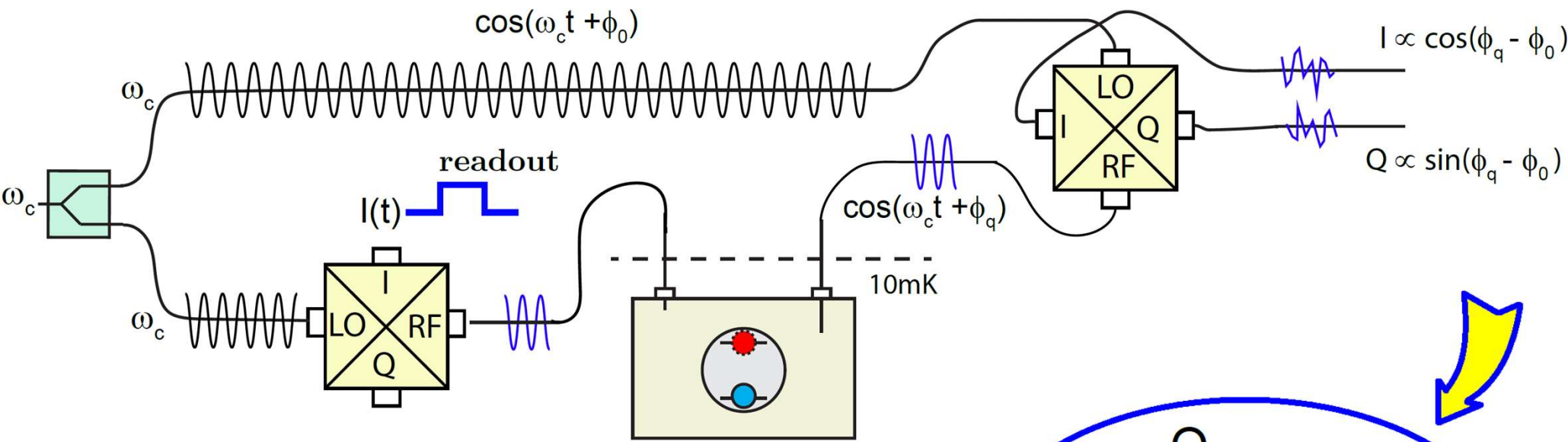
excited

cavity

Qubit readout

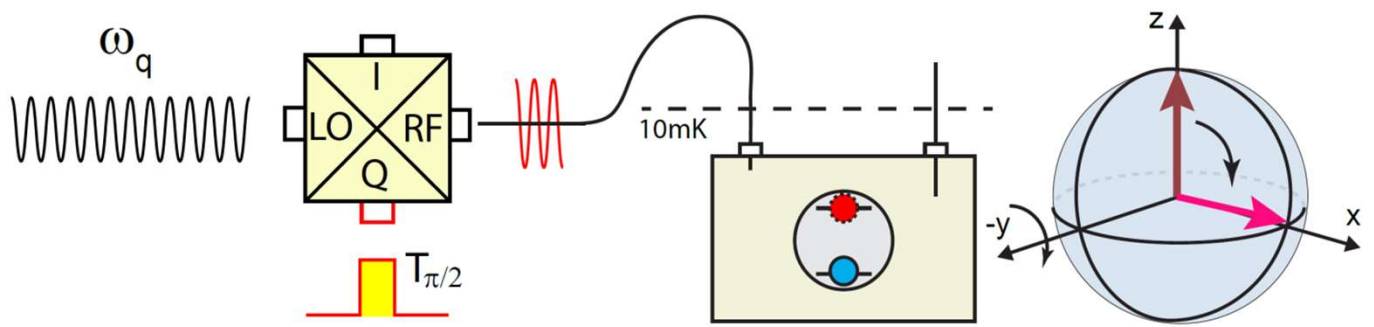
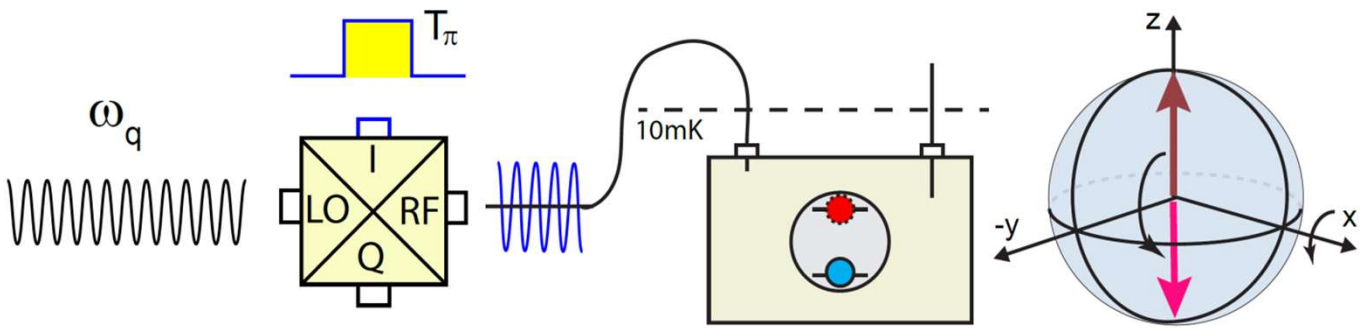
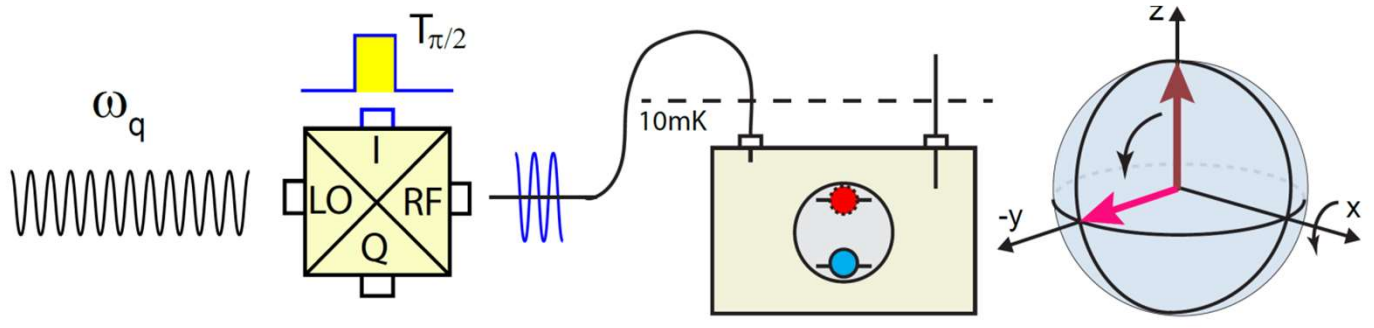
Readout pulse

- homodyne measurement
- dressed-state frequency



Qubit manipulation

Manipulation pulses



Contents

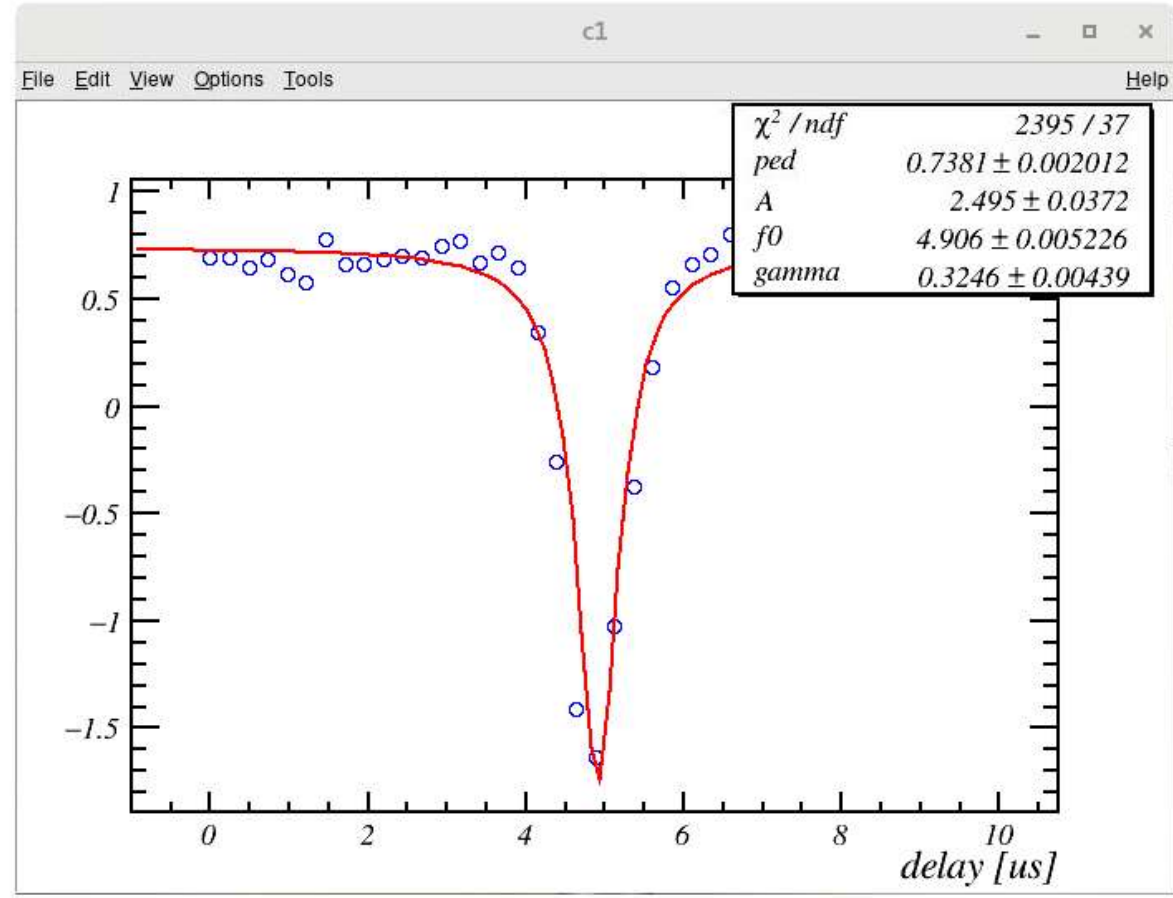
- 1 *Quantum mechanics; TRANSMON qubits; read/set*
- 2 *ROOT package*
- 3 *HYBRILIT experience; SU2 package*
- 4 *Qubit measurements*
- 5 *Quant-gates; Groover algorithm*



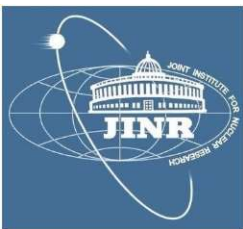
ROOT package

- I downloaded from CERN the ROOT-5.34 (Windows)
- I learned how to write my own macro and do fits

```
// _____ ROOT FITS _____  
  
void myfit() {  
  
// TGraph gr ("data.txt", "%lg %lg");  
// TGraph grr ("test.txt", "%lg %*lg %lg");  
// TGraph grrr("test.txt", "%lg %*lg %*lg %lg")  
  
gStyle->SetOptFit (1)  
gStyle->SetLinewidth(2)  
  
TGraphErrors* gr = new TGraphErrors("z1.txt")  
  
Int_t N = gr->GetN()  
Double_t x,y  
for (Int_t i=0; i<N; i++) {  
    gr->GetPoint (i, x, y)  
    gr->SetPointError(i, 0.01, 0.01)  
    gr->SetPoint (i, x/4.1, y)  
  
TF1 fit("fit", "([0]-[1]/(1+(x-[2])*(x-[2])/[3]/[3]))")  
  
    fit.SetParName (0, "ped" )  
    fit.SetParName (1, "A" )  
    fit.SetParName (2, "f0" )  
    fit.SetParName (3, "gamma")  
  
    fit.SetParameter(0, 0.500 )  
    fit.SetParameter(1, 2.500 )  
    fit.SetParameter(2, 4.700 )  
    fit.SetParameter(3, 0.500 )  
  
    gr->Fit("fit")  
}
```



```
Terminal  
File Edit View Search Terminal Help  
3 f0 4.90565e+00 5.22569e-03 5.71770e-05 -1.89307e-02  
4 gamma 3.24610e-01 4.39030e-03 5.10759e-05 -6.67857e-02  
root [3]
```



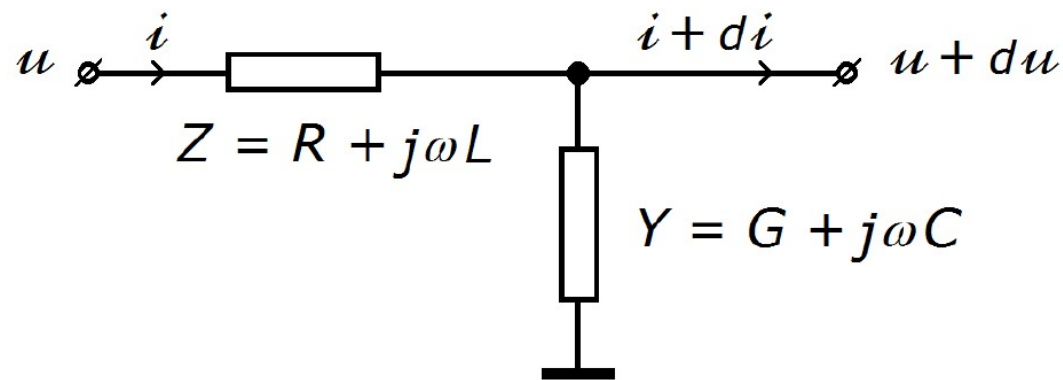
Contents

- 1 *Quantum mechanics; TRANSMON qubits; read/set*
- 2 *ROOT package*
- 3 *HYBRILIT experience; SU2 package***
- 4 *Qubit measurements*
- 5 *Quant-gates; Groover algorithm*



SU2 package

- model dispersion of a square wave on a transmission line:



$$-\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \partial_x \equiv \begin{pmatrix} 0 & L \\ C & 0 \end{pmatrix} \partial_t + \begin{pmatrix} 0 & R \\ G & 0 \end{pmatrix} \bigg| \begin{pmatrix} u \\ i \end{pmatrix}$$

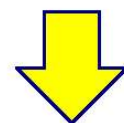
$$Z_0 = Y_0^{-1} = \sqrt{L/C}, \text{ line characteristic impedance}$$

$$\lambda_d^{-1} = (RY_0 - GZ_0)/2, \text{ dispersion length}$$

$$\lambda_a^{-1} = (RY_0 + GZ_0)/2, \text{ attenuation length}$$

$$c = 1/\sqrt{LC}, \text{ signal propagation speed}$$

- *equation:* $\partial_x + \sigma_1(\partial_{ct} + \lambda_a^{-1}) + j\sigma_2\lambda_d^{-1} = 0 \Big|_{\psi}$

 $\psi = e^{-ct/\lambda_a} \phi$

$$\partial_x + \sigma_1\partial_{ct} + j\sigma_2\lambda_d^{-1} = 0 \Big|_{\phi}$$

- *solution:*

$$\phi = e^{-\gamma^2(1+\sigma_1\beta)\frac{j\sigma_2}{\lambda_d}(x-vt)} \Big|_{\phi_0}$$



SU2 package

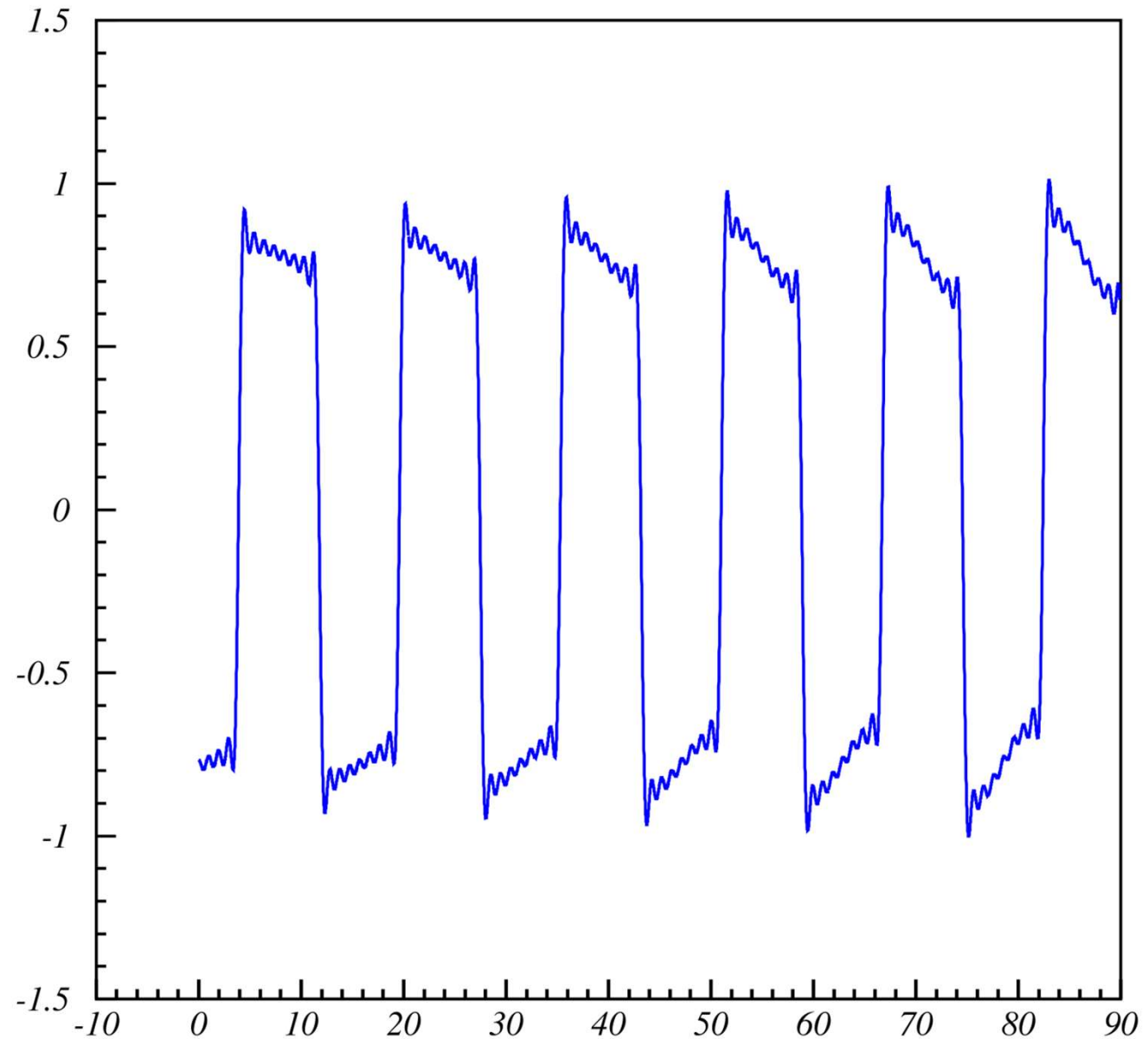
- I used the SU2 package to model the propagator:

```
auto propagator(real x, real t, real f){
{
  real gamma = sqrt(1+f*f*Ld*Ld/c/c)
  real beta  = sqrt(gamma*gamma-1) / gamma      ;
  return e^(-(1+sx*beta)*(j*sy)*(x-beta*c*t)   ;
            *gamma*gamma/Ld)                  ;}
```



SU2 package

- I obtained a very nice solution of square wave dispersion:



Contents

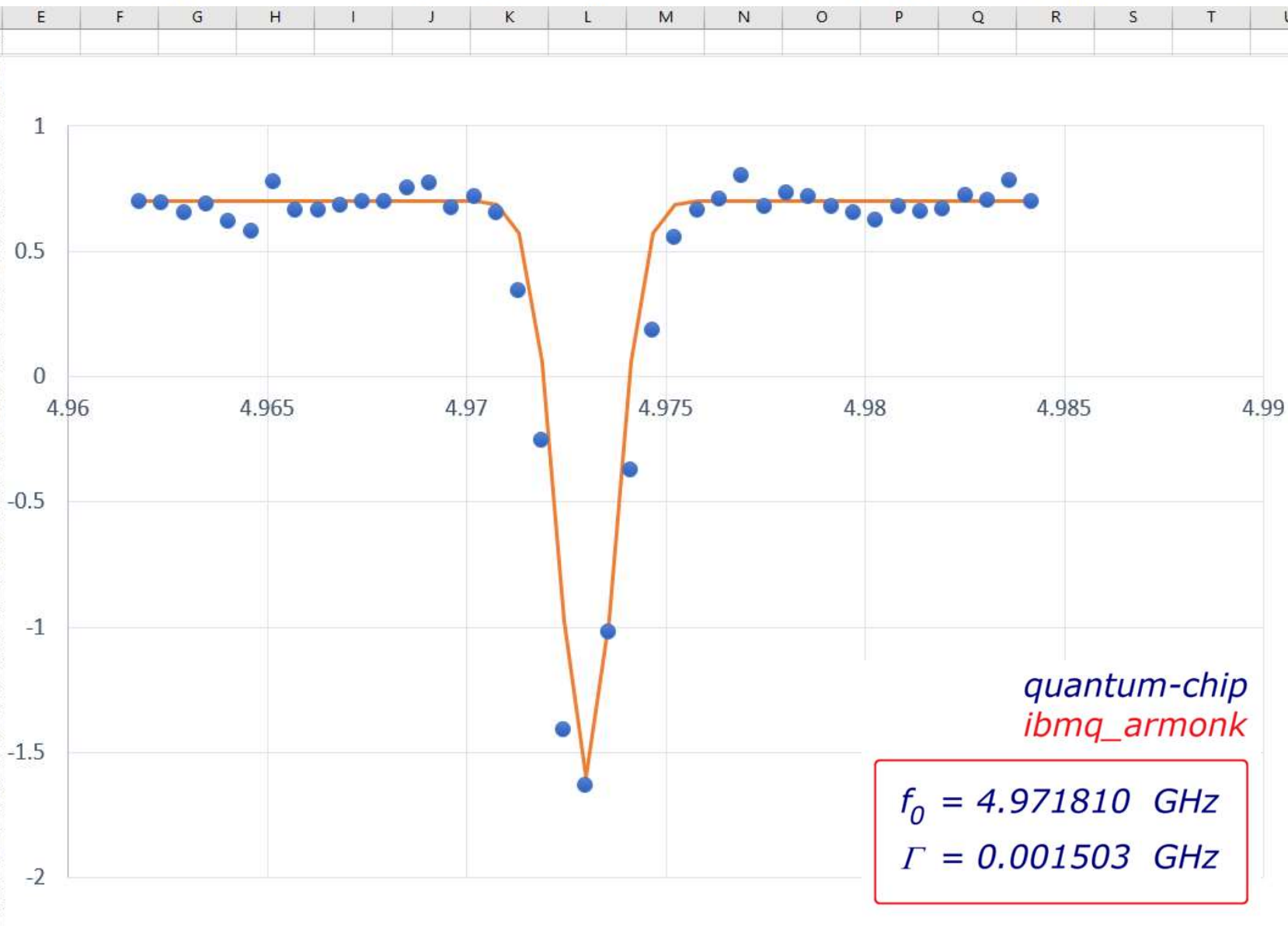
- 1 *Quantum mechanics; TRANSMON qubits; read/set*
- 2 *ROOT package*
- 3 *HYBRILIT experience; SU2 package*
- 4 *Qubit measurements***
- 5 *Quant-gates; Groover algorithm*



Qubit resonance frequency

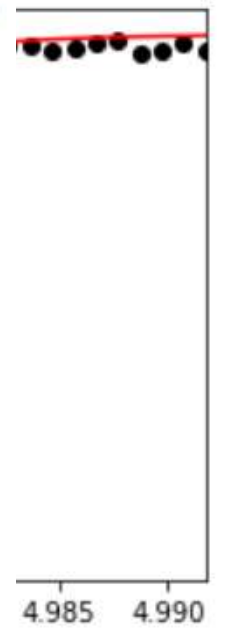
1. Qubit frequency scan

C	D
4.9646	0.578657
4.96516	0.775074
4.96572	0.663654
4.96628	0.662775
4.96684	0.682065
4.9674	0.697561
4.96796	0.693575
4.96852	0.749762
4.96908	0.770698
4.96964	0.670504
4.9702	0.717225
4.97076	0.649427
4.97132	0.342767
4.97188	-0.25658
4.97244	-1.41408
4.973	-1.63464
4.97356	-1.02303
4.97412	-0.37385
4.97468	0.184763
4.97524	0.554533
4.9758	0.659908
4.97636	0.708143
4.97692	0.800487
4.97748	0.675085
4.97804	0.732475
4.9786	0.717688
4.97916	0.675738
4.97972	0.65085
4.98028	0.623708
4.98084	0.678176
4.9814	0.654452
4.98196	0.66476
4.98252	0.718677
4.98308	0.700671
4.98364	0.777993
4.9842	0.694082

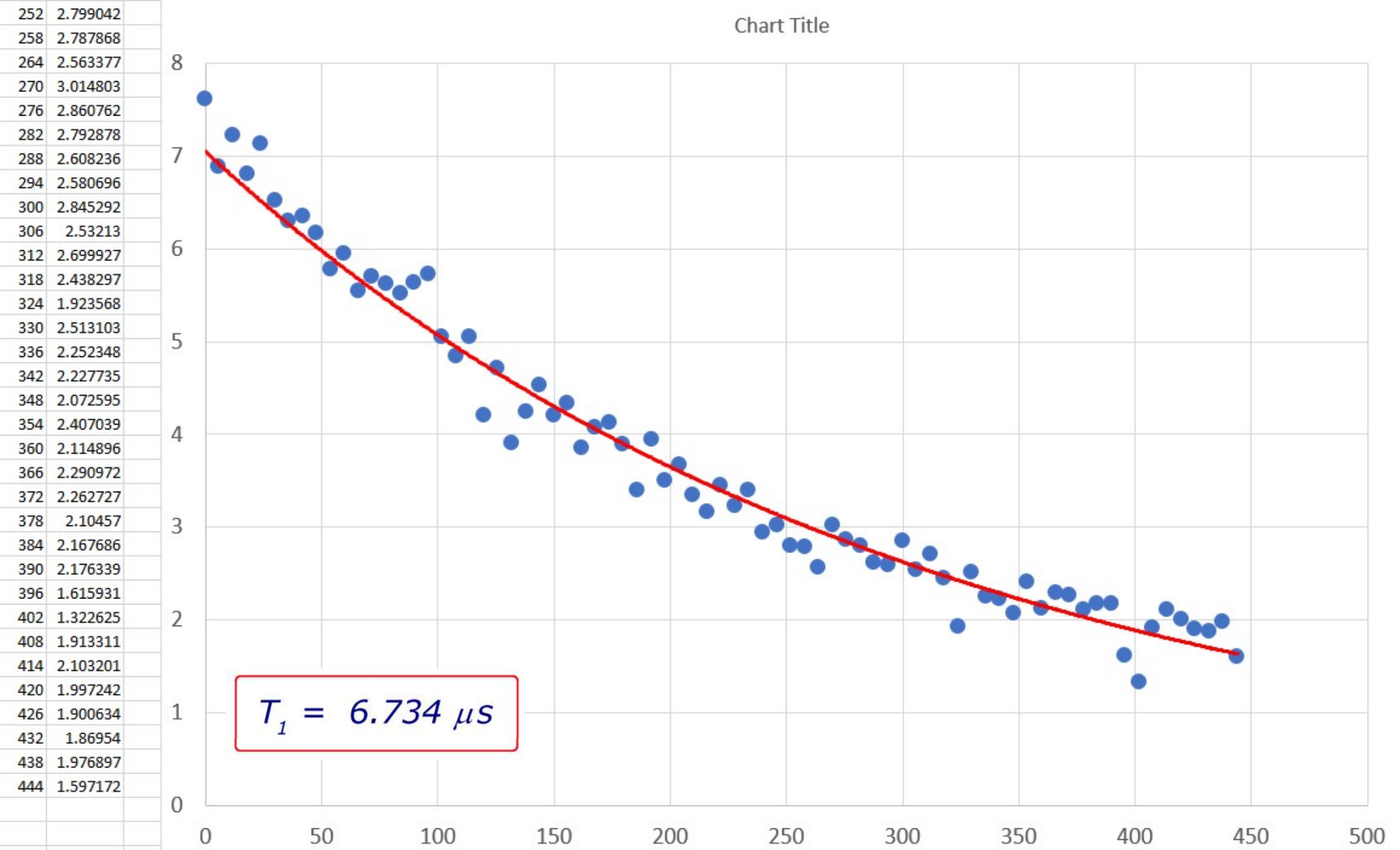


quantum-chip
ibmq_armonk
 $f_0 = 4.971810 \text{ GHz}$
 $\Gamma = 0.001503 \text{ GHz}$

on its ground
 red for non-
 qubit to an
 Ramsey puls
 increments.



3.3 T_1 determination via Inversion Recovery



Qubits on the Bloch sphere

Bloch sphere

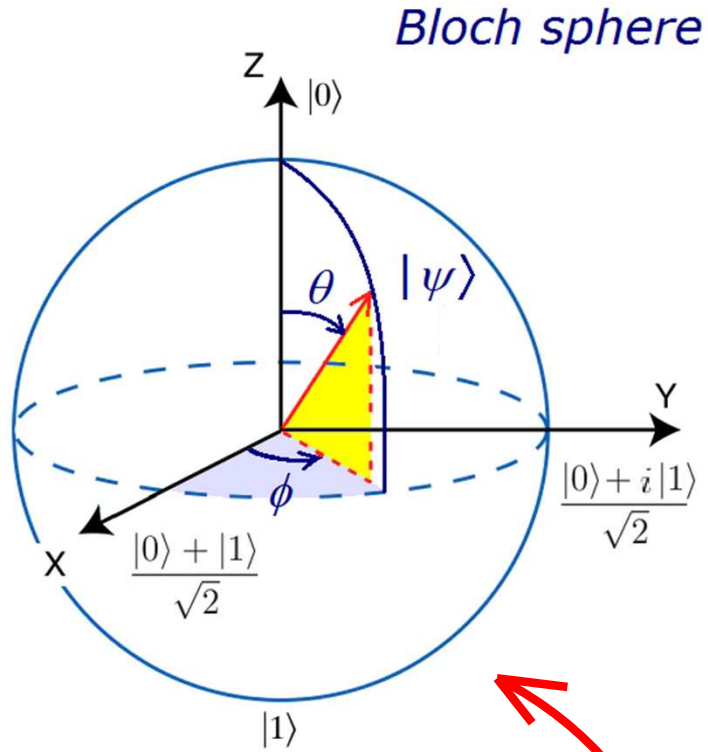
- 2 level system always equivalent to spin
- arbitrary wave-vector can be written as:

$$|\psi\rangle = \psi_{\uparrow}|\uparrow\rangle + \psi_{\downarrow}|\downarrow\rangle$$

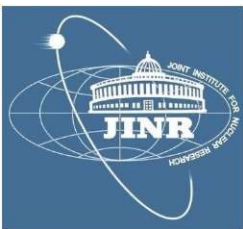
$$= e^{i\phi_{\uparrow}} \left(|\psi_{\uparrow}| \cdot |\uparrow\rangle + e^{i(\phi_{\downarrow}-\phi_{\uparrow})} |\psi_{\downarrow}| \cdot |\downarrow\rangle \right)$$

$$= e^{i\phi_{\uparrow}} \sqrt{|\psi_{\uparrow}|^2 + |\psi_{\downarrow}|^2} \left(\frac{|\psi_{\uparrow}|}{\sqrt{|\psi_{\uparrow}|^2 + |\psi_{\downarrow}|^2}} |\uparrow\rangle + \frac{|\psi_{\downarrow}|}{\sqrt{|\psi_{\uparrow}|^2 + |\psi_{\downarrow}|^2}} e^{i(\phi_{\downarrow}-\phi_{\uparrow})} |\downarrow\rangle \right)$$


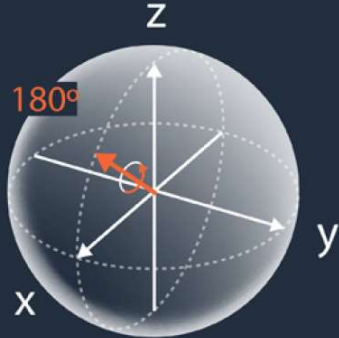
$$= e^{i\phi_{\uparrow}} \sqrt{|\psi_{\uparrow}|^2 + |\psi_{\downarrow}|^2} \left(\cos\frac{\theta}{2} |\uparrow\rangle + \sin\frac{\theta}{2} e^{i(\phi_{\downarrow}-\phi_{\uparrow})} |\downarrow\rangle \right)$$



represented on the Bloch sphere



Quantum logical gates

GATE	CIRCUIT REPRESENTATION	MATRIX REPRESENTATION	TRUTH TABLE	BLOCH SPHERE						
<p>H gate: rotates the qubit state by π radians (180°) about an axis diagonal in the x-z plane. This is equivalent to an X-gate followed by a $\frac{\pi}{2}$ rotation about the y-axis.</p>		$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	<table border="1"> <thead> <tr> <th>Input</th> <th>Output</th> </tr> </thead> <tbody> <tr> <td>$0\rangle$</td> <td>$\frac{ 0\rangle + 1\rangle}{\sqrt{2}}$</td> </tr> <tr> <td>$1\rangle$</td> <td>$\frac{ 0\rangle - 1\rangle}{\sqrt{2}}$</td> </tr> </tbody> </table>	Input	Output	$ 0\rangle$	$\frac{ 0\rangle + 1\rangle}{\sqrt{2}}$	$ 1\rangle$	$\frac{ 0\rangle - 1\rangle}{\sqrt{2}}$	
Input	Output									
$ 0\rangle$	$\frac{ 0\rangle + 1\rangle}{\sqrt{2}}$									
$ 1\rangle$	$\frac{ 0\rangle - 1\rangle}{\sqrt{2}}$									

$$2U_3(\theta, \phi, \lambda) = \cos\frac{\theta}{2} \left[(1 + e^{i(\lambda+\phi)}) \cdot \mathbf{1} + (1 - e^{i(\lambda+\phi)}) \cdot \sigma_z \right] + \sin\frac{\theta}{2} \left[e^{-i\lambda} \sigma_+ + e^{i\phi} \sigma_- \right]$$

controlled-U gates

if $q[0] = |1\rangle$ operation U is performed on $q[1]$
 else ID

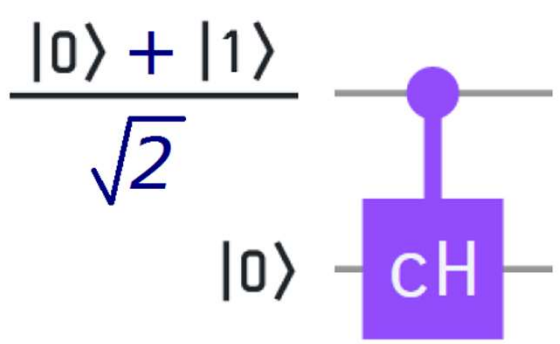


Entanglement

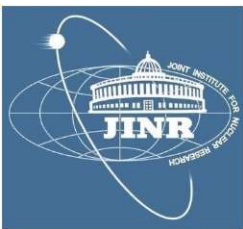
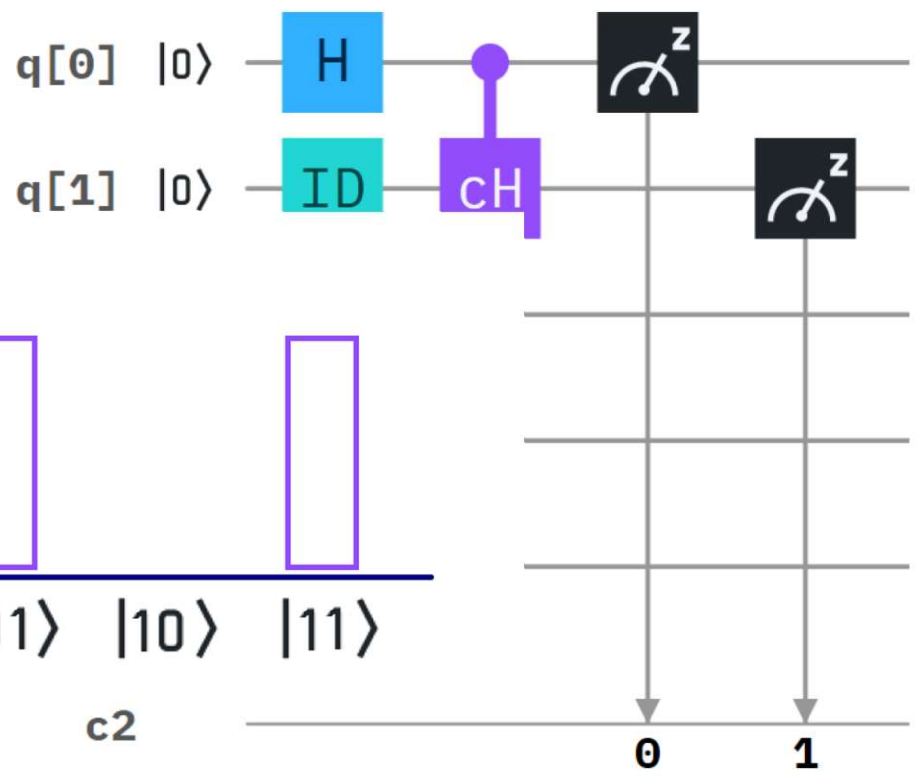
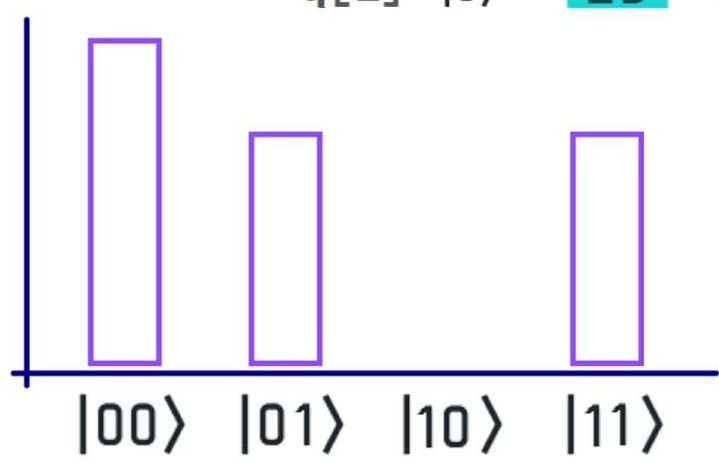
- 2 qubit states: $|\uparrow\uparrow\rangle, |\uparrow\downarrow\rangle, |\downarrow\uparrow\rangle$ and $|\downarrow\downarrow\rangle$

- entangled states: $|\psi\rangle = \frac{|\downarrow, \uparrow\rangle \pm |\uparrow, \downarrow\rangle}{\sqrt{2}}$

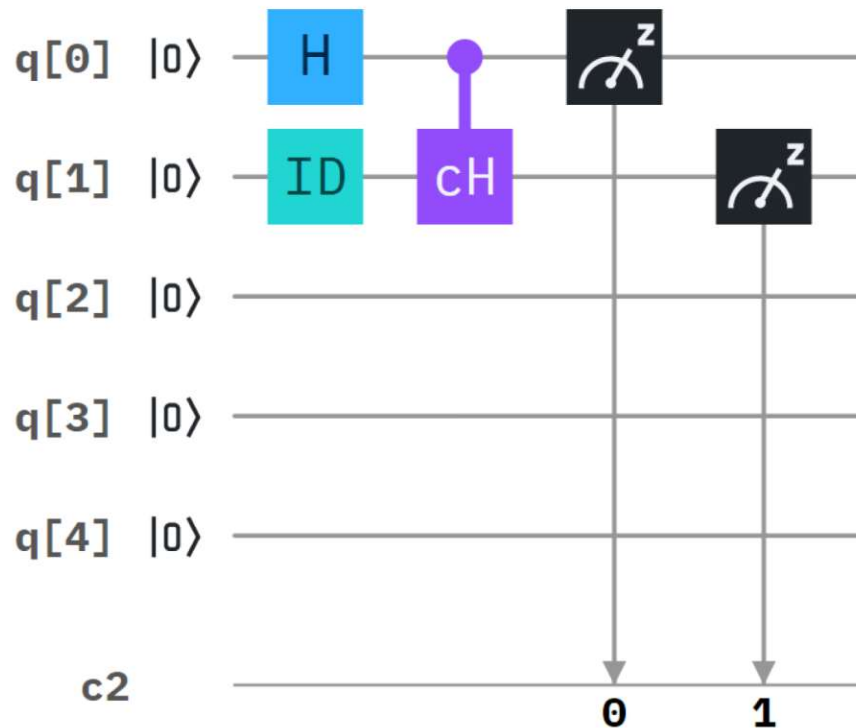
- c-Hadamard gate:



=



Circuit composer



Circuit editor

```
1 OPENQASM 2.0;  
2 include "qelib1.inc";  
3  
4 qreg q[5];  
5 creg c[2];  
6  
7 h q[0];  
8 id q[1];  
9 ch q[0],q[1];  
10 measure q[0] -> c[0];  
11 measure q[1] -> c[1];
```

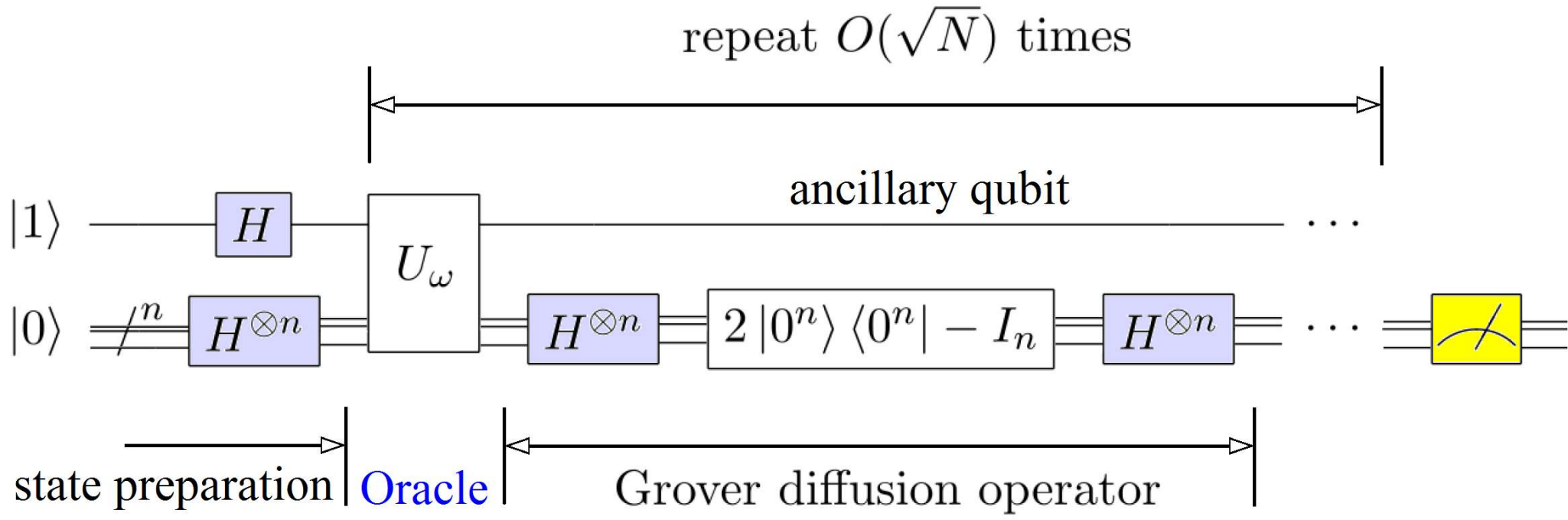
Create account

The screenshot displays the IBM Quantum Composer interface. On the left, a file explorer shows 'Composer files' with one file named 'Untitled circuit' updated 'an hour ago'. The main workspace shows a quantum circuit with two qubits (q 0 and q 1) and a classical register (c2). Q 0 starts in state $|0\rangle$, followed by an H gate. Q 1 starts in state $|0\rangle$, followed by an I gate and an H gate. A CNOT gate is controlled by q 0 and targets q 1. The circuit is followed by a measurement on q 0 and a classical control gate (C) that sets c2 based on the measurement result. The top toolbar includes gates like H, \oplus , \otimes , \otimes^+ , \otimes^- , I, T, S, Z, T^\dagger , S^\dagger , P, RZ, $|0\rangle$, and \otimes^z . Below the circuit, the 'Statevector' visualization shows a bar chart of amplitudes for computational basis states 00, 01, 10, and 11. The 'Q-sphere' visualization shows a Bloch sphere with a state vector pointing towards the top pole (100) and a phase angle of 0. The right sidebar shows the OpenQASM 2.0 code and a 'Setup and run' button.

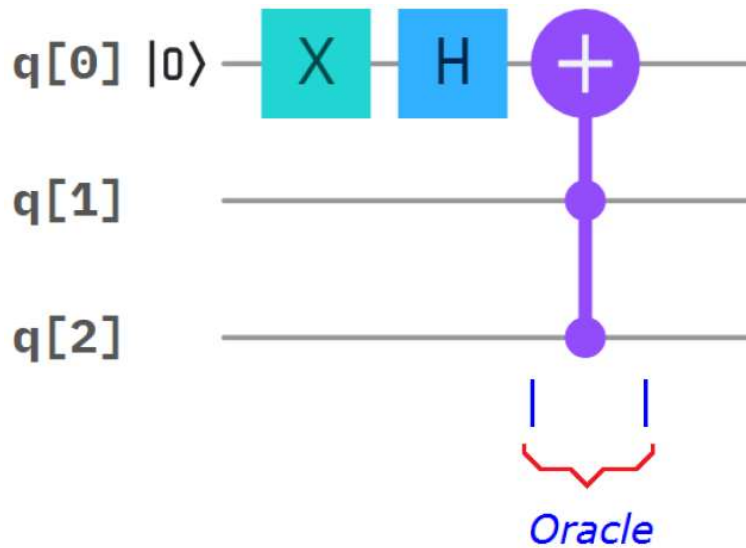
```
1 OPENQASM 2.0;
2 include "qelib1.inc";
3
4 qreg q[2];
5 creg c[2];
6
7 reset q[0];
8 reset q[1];
9 h q[0];
10 id q[1];
11 ch q[0],q[1];
```



Grover algorithm



Detect the $|1, 1\rangle$ state



$$cc\text{-NOT}(|q_0\rangle; q_1, q_2)$$

$$= \text{NOT}(|q_0\rangle) \dots \text{for } (q_1, q_2) = (1, 1)$$

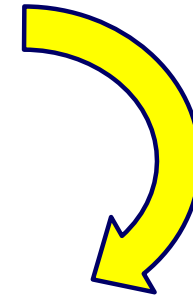
$$= \text{ID}(|q_0\rangle) \dots \text{for } (q_1, q_2) = (0, 0)$$

$$(0, 1)$$

$$(1, 0)$$

$$\text{NOT}(|0\rangle - |1\rangle) = |1\rangle - |0\rangle = -(|0\rangle - |1\rangle)$$

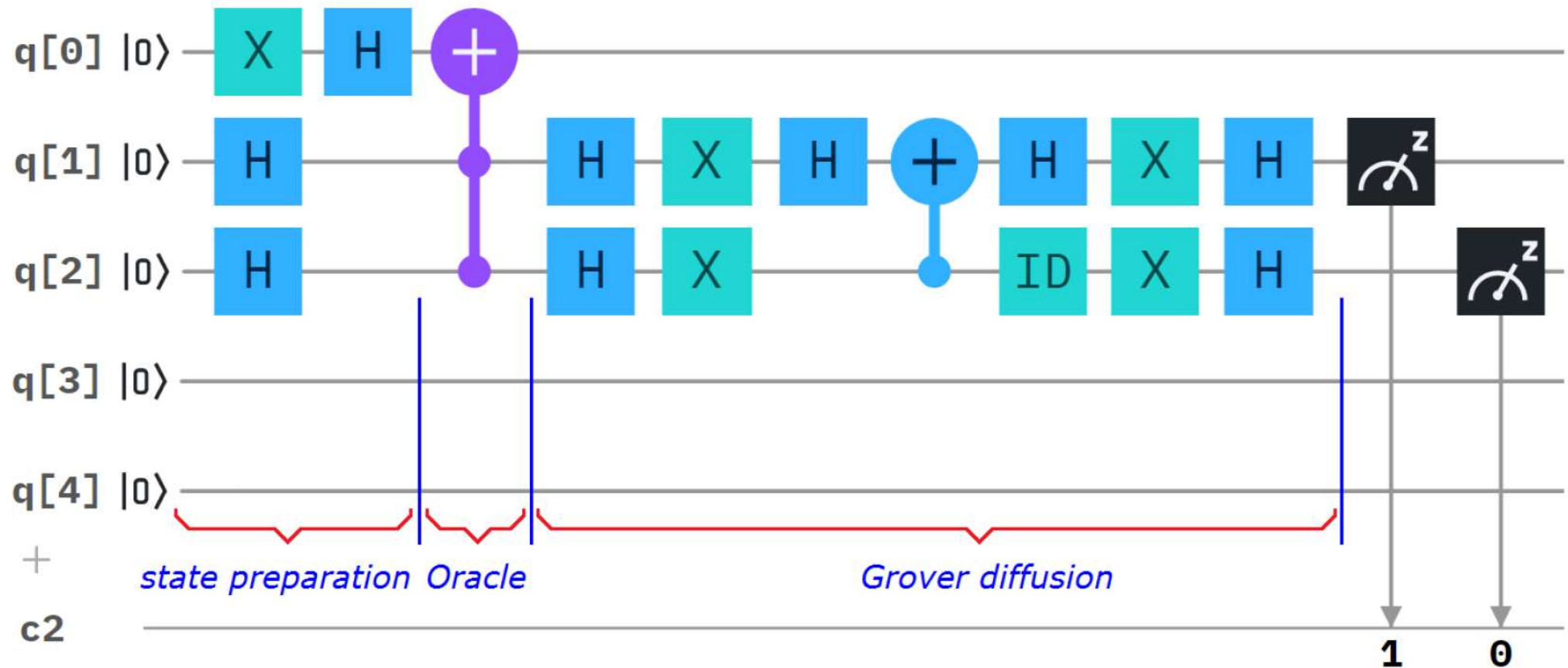
$$\text{ID}(|0\rangle - |1\rangle) = |0\rangle - |1\rangle = +(|0\rangle - |1\rangle)$$



Signals with a “ - ” the target state

IBM Q-Experience

Circuit composer



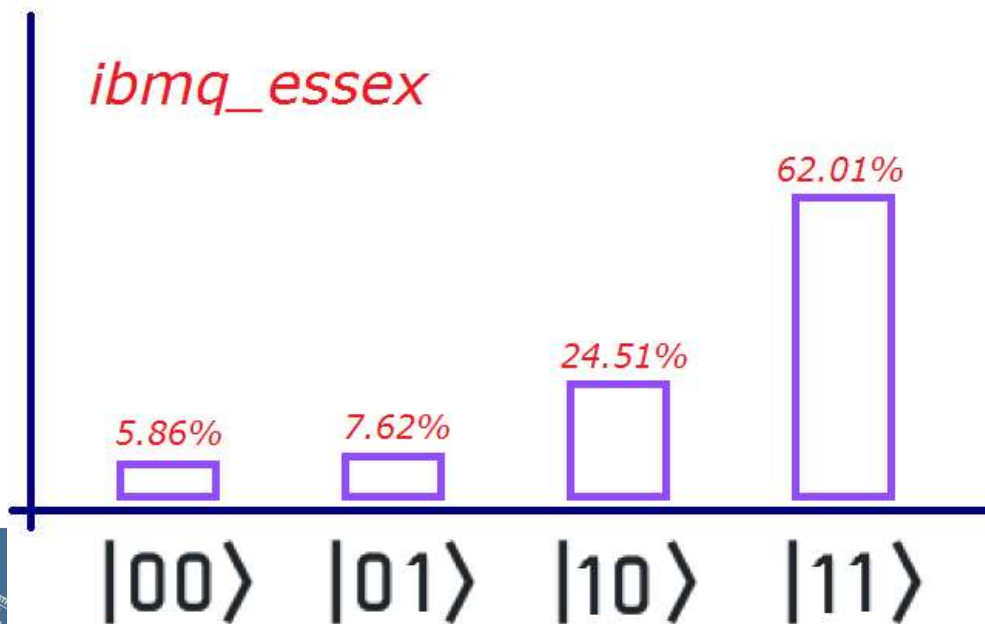
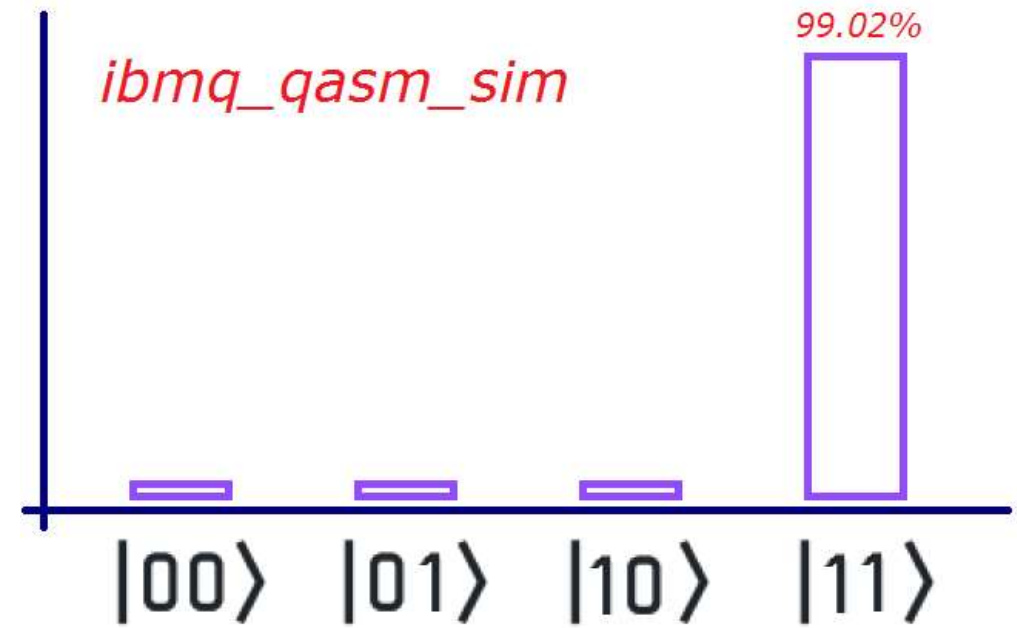
QASM-2

Circuit editor

```
1 OPENQASM 2.0;
2 include "qelib1.inc";
3
4 qreg q[5];
5 creg c[2];
6
7 x q[0];
8 h q[1];
9 h q[2];
10 h q[0];
11 ccx q[1],q[2],q[0];
12 h q[1];
13 h q[2];
14 x q[1];
15 x q[2];
16 h q[1];
17 cx q[2],q[1];
18 h q[1];
19 id q[2];
20 x q[1];
21 x q[2];
22 h q[1];
23 h q[2];
24 measure q[1] -> c[1];
25 measure q[2] -> c[0];
```



Results



Personal opinions

- I learned about the quantum physics fundamentals of qubits and did some interesting hands-on determinations (f_0 , T_1 , T_2) of the `ibmq_armonk` qubit system on IBM's Q-Experience site
- We had access to the supercomputing cluster HybriLIT of JINR, which was very cool – for an SU2 simulation package in C++
- I learned to use the ROOT package from CERN to process and do fits on data
- We learned how to process multiple-entry quantum gate output and walked through the Grover quantum search algorithm – and after implemented and ran it on IBM's Q-Experience site
- The professors were very good and friendly, I highly recommend this student training programme !

