

Introduction to Quantum Programming

Jarosław Miszczak
IITiS PAN

April 27, 2018
QIPLSIGML—Machine Learning meets Quantum Computation

Our goals

Quantum programming

Manipulation of quantum gates

Programming QRAM

High-level programming

What next?

Q?



Our goals

- ▶ Understand the difference between quantum and classical programming.



Our goals

- ▶ Understand the difference between quantum and classical programming.
- ▶ Introduce various approaches to quantum programming.



Our goals

- ▶ Understand the difference between quantum and classical programming.
- ▶ Introduce various approaches to quantum programming.
- ▶ Write some code.



Our goals

- ▶ Understand the difference between quantum and classical programming.
- ▶ Introduce various approaches to quantum programming.
- ▶ Write some code.
(\equiv *Talk is cheap. Show me the quantum code.*)

Quantum programming

➤ Quantum programming

What is quantum programming?

Quantum programming is a process that leads from an original formulation of a computing problem to executable quantum computer programs.

➤ Quantum programming

What is quantum programming?

- ▶ *The only way to learn a new quantum programming language is by writing programs in it.*

➤ Quantum programming

What is quantum programming?

- ▶ *The only way to learn a new quantum programming language is by writing programs in it.*
- ▶ *The process of preparing programs for a quantum computer is especially attractive because it not only can be economically and scientifically rewarding, it can also be an aesthetic experience much like composing poetry or music.*

➤ Quantum programming

What is quantum programming?

- ▶ *The only way to learn a new quantum programming language is by writing programs in it.*
- ▶ *The process of preparing programs for a quantum computer is especially attractive because it not only can be economically and scientifically rewarding, it can also be an aesthetic experience much like composing poetry or music.*
- ▶ *Only the modern quantum computer has made quantum programming both challenging and relevant.*

➤ Quantum programming

Why bother?

- ▶ Use real quantum computers.

➤ Quantum programming

Why bother?

- ▶ Use real quantum computers.
- ▶ Play with quantum mechanics.

➤ Quantum programming

Why bother?

- ▶ Use real quantum computers.
- ▶ Play with quantum mechanics.
- ▶ Stretch your imagination by creating a new programming language with quantum elements...

➤ Quantum programming

Why bother?

- ▶ Use real quantum computers.
- ▶ Play with quantum mechanics.
- ▶ Stretch your imagination by creating a new programming language with quantum elements...
- ▶ ...or a language for describing quantum mechanics.

➤ Quantum programming

How to do quantum programming?

- ▶ Level 0: Manipulation of quantum gates.

➤ Quantum programming

How to do quantum programming?

- ▶ Level 0: Manipulation of quantum gates.
- ▶ Level 1: Programming QRAM.

➤ Quantum programming

How to do quantum programming?

- ▶ Level 0: Manipulation of quantum gates.
- ▶ Level 1: Programming QRAM.
- ▶ Level 2: High-level programming.

Manipulation of quantum gates

➤ Manipulation of quantum gates

Level 0

Direct usage of quantum gates.

- ▶ Visual manipulation of gates and circuits.
- ▶ Multiplication of matrices and vectors.

➤ Manipulation of quantum gates

Visual manipulation of gates and circuits

The screenshot shows the IBM Quantum Experience circuit editor interface. At the top, there are two tabs: "IBM Q 5.1 [ibmqx4]" with "ACTIVE USERS" and "IBM Q 5 [ibmqx2]" with "ACTIVE DEDICATED". Below the tabs, the circuit name is "FourQubitsTest" with an edit icon. To the right are buttons for "Add a description", "New", "Save", and "Save as".

The main editor area includes a toolbar with "Switch to Qasm Editor", "Backend: ibmqx4", "My Units: 18", and "Experiment Units: 3". On the right side of the toolbar are "Run" and "Simulate" buttons with dropdown menus.

The circuit diagram shows five qubits: q[0], q[1], q[2], q[3], and q[4]. Qubit q[0] starts with a Hadamard (H) gate. Qubits q[1] and q[2] are connected by a CNOT gate with q[1] as the control and q[2] as the target. Qubit q[3] starts with an H gate. Qubits q[2] and q[3] are connected by a CNOT gate with q[2] as the control and q[3] as the target. The circuit ends with two measurement operations on q[0] and q[3].

On the right side, there is a palette of quantum gates and barrier operations. The gates include Z, H, S, S†, T, T†, and a CNOT gate. The barrier operations include a vertical dashed line and a pink square with a white circle.

<https://quantumexperience.ng.bluemix.net/qx/editor>

Manipulation of quantum gates

Visual manipulation of gates and circuits

The screenshot displays the QUIDE Quantum Integrated Development Environment. The main window is titled "Telepotation.cs" and contains the following C# code:

```
11 public static void Main() {  
12     QuantumComputer comp = QuantumComputer.GetInstance();  
13  
14     // nontrivial state to be teleported  
15     var x_initStates = new Dictionary<ulong, Complex>() {  
16         {0, new Complex(-0.6, 0)},  
17         {1, new Complex(0, 0.8)}  
18     };  
19     Register x = comp.NewRegister(x_initStates, 1);
```

The interface includes a "Circuit Designer" panel with various quantum gates (H, X, Y, Z, \sqrt{X} , R_x , R_y , R_z , R, θ , R_k , R_k^{-1}) and a "Select Composite Gate..." dropdown. The circuit diagram shows three qubits: $x_0 = |+\rangle$, $y_0 = |0\rangle$, and $z_0 = |0\rangle$. The circuit includes H gates, CNOT gates, and a Z gate.

The "Output" panel shows the following table:

Value	Qubits	Probability	Amplitude
0>	000>	P = 0.18	-0.42 + 0.00i
3>	011>	P = 0.18	-0.42 + 0.00i
5>	101>	P = 0.32	0.00 + 0.57i
6>	110>	P = 0.32	0.00 + 0.57i

The "Properties" panel displays a Bloch sphere visualization.

<http://www.quide.eu/> and <https://bitbucket.org/quide/>

➤ Manipulation of quantum gates

Visual manipulation of gates and circuits

- ▶ Direct manipulation of quantum gates, measurement.

➤ Manipulation of quantum gates

Visual manipulation of gates and circuits

- ▶ Direct manipulation of quantum gates, measurement.
- ▶ Integration with text-based description of circuits: QASM (IBM Q) or C# (QulIDE).

➤ Manipulation of quantum gates

Visual manipulation of gates and circuits

- ▶ Direct manipulation of quantum gates, measurement.
- ▶ Integration with text-based description of circuits: QASM (IBM Q) or C# (QulIDE).
- ▶ Useful for testing small circuits.

➤ Manipulation of quantum gates

Visual manipulation of gates and circuits

- ▶ Direct manipulation of quantum gates, measurement.
- ▶ Integration with text-based description of circuits: QASM (IBM Q) or C# (QulIDE).
- ▶ Useful for testing small circuits.
- ▶ Not so much for real algorithms.

➤ Manipulation of quantum gates

Visual manipulation of gates and circuits

- ▶ Direct manipulation of quantum gates, measurement.
- ▶ Integration with text-based description of circuits: QASM (IBM Q) or C# (QulIDE).
- ▶ Useful for testing small circuits.
- ▶ Not so much for real algorithms.
- ▶ You have to live with connectivity limitations (IBM Q).

➤ Manipulation of quantum gates

Visual manipulation of gates and circuits

- ▶ Direct manipulation of quantum gates, measurement.
- ▶ Integration with text-based description of circuits: QASM (IBM Q) or C# (QulIDE).
- ▶ Useful for testing small circuits.
- ▶ Not so much for real algorithms.
- ▶ You have to live with connectivity limitations (IBM Q).
- ▶ But you can use a real quantum computer!

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

You already know that

- ▶ quantum states are just vectors (at least we would like them to be)

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

You already know that

- ▶ quantum states are just vectors (at least we would like them to be)
- ▶ ...quantum gates are just matrices (at most 4×4 , and don't mention the decoherence)...

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

You already know that

- ▶ quantum states are just vectors (at least we would like them to be)
- ▶ ...quantum gates are just matrices (at most 4×4 , and don't mention the decoherence)...
- ▶ ...only the final step is somehow strange.

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

You already know that

- ▶ quantum states are just vectors (at least we would like them to be)
- ▶ ...quantum gates are just matrices (at most 4×4 , and don't mention the decoherence)...
- ▶ ...only the final step is somehow strange.
- ▶ *Real Programmers do Quantum Computing in FORTRAN.*

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

For example, in Julia...

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

For example, in Julia...

Examples in Wolfram can be found in the GitHub repo

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

- ▶ Actually this is almost as good as it gets!

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

- ▶ Actually this is almost as good as it gets!
- ▶ Because...

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

- ▶ Actually this is almost as good as it gets!
- ▶ Because...
 - ▶ ...you already know the language.

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

- ▶ Actually this is almost as good as it gets!
- ▶ Because...
 - ▶ ...you already know the language.
 - ▶ ...it is very easy to implement classical control.

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

- ▶ Actually this is almost as good as it gets!
- ▶ Because...
 - ▶ ...you already know the language.
 - ▶ ...it is very easy to implement classical control.
 - ▶ ...it is relatively easy to take into account effects of decoherence.

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

- ▶ Actually this is almost as good as it gets!
- ▶ Because...
 - ▶ ...you already know the language.
 - ▶ ...it is very easy to implement classical control.
 - ▶ ...it is relatively easy to take into account effects of decoherence.
- ▶ Missing: the memory management.

➤ Manipulation of quantum gates

Alternative approach: Manipulation of vectors and matrices

- ▶ Actually this is almost as good as it gets!
- ▶ Because...
 - ▶ ...you already know the language.
 - ▶ ...it is very easy to implement classical control.
 - ▶ ...it is relatively easy to take into account effects of decoherence.
- ▶ Missing: the memory management.
- ▶ And in most cases you don't need all the power/libraries/etc coming with the host language.

➤ Manipulation of quantum gates

Where to go next?

- ▶ IBM Q Experience:

`https://quantumexperience.ng.bluemix.net/qx/editor`

➤ Manipulation of quantum gates

Where to go next?

- ▶ IBM Q Experience:
`https://quantumexperience.ng.bluemix.net/qx/editor`
- ▶ Packages/matrix manipulation libraries:

➤ Manipulation of quantum gates

Where to go next?

- ▶ IBM Q Experience:
<https://quantumexperience.ng.bluemix.net/qx/editor>
- ▶ Packages/matrix manipulation libraries:
 - ▶ quantum-octave (Octave/Matlab):
<https://github.com/ZKSI/quantum-octave>

➤ Manipulation of quantum gates

Where to go next?

- ▶ IBM Q Experience:
`https://quantumexperience.ng.bluemix.net/qx/editor`
- ▶ Packages/matrix manipulation libraries:
 - ▶ quantum-octave (Octave/Matlab):
`https://github.com/ZKSI/quantum-octave`
 - ▶ QuTiP (Python library): `http://qutip.org/`

➤ Manipulation of quantum gates

Where to go next?

- ▶ IBM Q Experience:
<https://quantumexperience.ng.bluemix.net/qx/editor>
- ▶ Packages/matrix manipulation libraries:
 - ▶ quantum-octave (Octave/Matlab):
<https://github.com/ZKSI/quantum-octave>
 - ▶ QuTiP (Python library): <http://qutip.org/>
- ▶ Many more at Quantiki
<https://quantiki.org/wiki/list-qc-simulators>

Programming QRAM

➤ Programming QRAM

Advantages of QRAM

- ▶ data abstraction

➤ Programming QRAM

Advantages of QRAM

- ▶ data abstraction \equiv allocation of quantum memory

➤ Programming QRAM

Advantages of QRAM

- ▶ data abstraction \equiv allocation of quantum memory
- ▶ compound quantum operations

➤ Programming QRAM

Advantages of QRAM

- ▶ data abstraction \equiv allocation of quantum memory
- ▶ compound quantum operations \equiv functions encapsulating sequence of quantum gates or quantum primitives

➤ Programming QRAM

Advantages of QRAM

- ▶ data abstraction \equiv allocation of quantum memory
- ▶ compound quantum operations \equiv functions encapsulating sequence of quantum gates or quantum primitives
- ▶ classical control of quantum operations

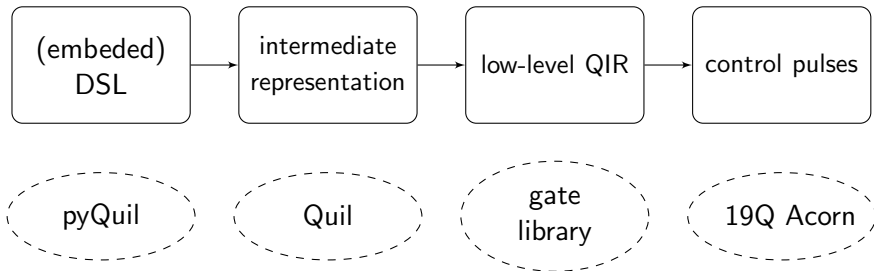
➤ Programming QRAM

Advantages of QRAM

- ▶ data abstraction \equiv allocation of quantum memory
- ▶ compound quantum operations \equiv functions encapsulating sequence of quantum gates or quantum primitives
- ▶ classical control of quantum operations \equiv loops, ifs etc. mixed with quantum code

➤ Programming QRAM

Software architecture



➤ Programming QRAM

Quantum middleware

- ▶ embedded domain specific language →
C/Python/Wolfram/Haskell with library of functions

➤ Programming QRAM

Quantum middleware

- ▶ embedded domain specific language →
C/Python/Wolfram/Haskell with library of functions
- ▶ data abstraction → allocation of classical and quantum registers based on $qu(b|d)its$

➤ Programming QRAM

Quantum middleware

- ▶ embedded domain specific language → C/Python/Wolfram/Haskell with library of functions
- ▶ data abstraction → allocation of classical and quantum registers based on $qu(b|d)its$
- ▶ quantum functions → custom elementary gates defined by matrices or compound statements

➤ Programming QRAM

Quantum middleware

- ▶ embedded domain specific language → C/Python/Wolfram/Haskell with library of functions
- ▶ data abstraction → allocation of classical and quantum registers based on $qu(b|d)its$
- ▶ quantum functions → custom elementary gates defined by matrices or compound statements
- ▶ classical control of quantum memory → by using host language

➤ Programming QRAM

...its advantages...

- ▶ easy to learn and use

➤ Programming QRAM

...its advantages...

- ▶ easy to learn and use
- ▶ auto-magic quantum memory management

➤ Programming QRAM

...its advantages...

- ▶ easy to learn and use
- ▶ auto-magic quantum memory management
- ▶ integration with classical machine

➤ Programming QRAM

...and its disadvantages

- ▶ very similar to low-level code

➤ Programming QRAM

...and its disadvantages

- ▶ very similar to low-level code
- ▶ lack of expressibility

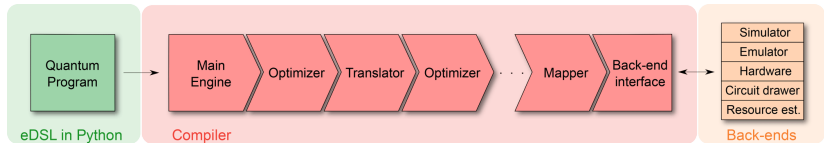
➤ Programming QRAM

Example 1: ProjectQ

- ▶ Python library developed by ETH (<https://projectq.ch/>)
- ▶ offers various targets
 - ▶ hardware (IBM Q Experience)
 - ▶ resource counter (???)
 - ▶ graphical circuit representation

➤ Programming QRAM

Example 1: ProjectQ



➤ Programming QRAM

Example 1: ProjectQ

Nice features

- ▶ Natural (for physicist) syntax for executing quantum gates.
- ▶ Meta instructions for quantum-controlled quantum operations and support for reverse call

➤ Programming QRAM

Example 1: ProjectQ

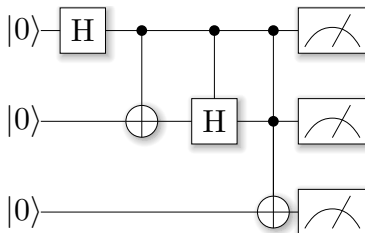
Some examples...

➤ Programming QRAM

Example 1: ProjectQ

Metainstruction Control

Execution of the code is based on the state of quantum register.

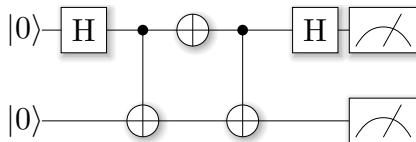


➤ Programming QRAM

Example 1: ProjectQ

Metainstruction Dagger

Reverse execution of the quantum code.



➤ Programming QRAM

Example 2: Rigetti Forest and pyQuil

- ▶ Quil \equiv quantum assembly language.

➤ Programming QRAM

Example 2: Rigetti Forest and pyQuil

- ▶ Quil \equiv quantum assembly language.
- ▶ pyQuil \equiv Python library for manipulating quantum programmes in Quil.

➤ Programming QRAM

Example 2: Rigetti Forest and pyQuil

- ▶ Quil \equiv quantum assembly language.
- ▶ pyQuil \equiv Python library for manipulating quantum programmes in Quil.
- ▶ Access to Rigetti 19Q-Acorn quantum computer!

➤ Programming QRAM

Example 2: Rigetti Forest and pyQuil

- ▶ Quil \equiv quantum assembly language.
- ▶ pyQuil \equiv Python library for manipulating quantum programmes in Quil.
- ▶ Access to Rigetti 19Q-Acorn quantum computer!
- ▶ More during the third day: Adam Szady, Quantum programming with (Py)Quil.

➤ High-level programming

Domain specific languages

Level 2

Domain specific language with data and function abstraction.

- ▶ QCL (<http://tph.tuwien.ac.at/~oemer/qcl.html>)
- ▶ LanQ (<http://lanq.sourceforge.net/>)
- ▶ QPL and cQPL
(<https://arxiv.org/abs/quant-ph/0511145>)
- ▶ Scaffold (<https://github.com/epiqc/ScaffCC>)

➤ High-level programming

Example 1: QCL – focus on quantum computing

- ▶ First release in 1998, last in 2014
(<http://tph.tuwien.ac.at/~oemer/qcl.html>).

➤ High-level programming

Example 1: QCL – focus on quantum computing

- ▶ First release in 1998, last in 2014
(<http://tph.tuwien.ac.at/~oemer/qcl.html>).
- ▶ Architecture independent programming language for quantum computers.

➤ High-level programming

Example 1: QCL – focus on quantum computing

Features

- ▶ Syntax for reversibility (uncomputing).

➤ High-level programming

Example 1: QCL – focus on quantum computing

Features

- ▶ Syntax for reversibility (uncomputing).
- ▶ Different types of quantum memory for better optimization.

➤ High-level programming

Example 1: QCL – focus on quantum computing

Features

- ▶ Syntax for reversibility (uncomputing).
- ▶ Different types of quantum memory for better optimization.
- ▶ Quantum conditions — quantum-controlled execution (generalization of controlled gates).

➤ High-level programming

Example 1: QCL – focus on quantum computing

Features

- ▶ Syntax for reversibility (uncomputing).
- ▶ Different types of quantum memory for better optimization.
- ▶ Quantum conditions — quantum-controlled execution (generalization of controlled gates).
- ▶ Various types of compound statements (related with memory management): quantum operators, quantum functions, procedures.

➤ High-level programming

Example 1: QCL – focus on quantum computing

Advanced quantum memory management using types:

- ▶ `qureg` — basic type for quantum registers,

➤ High-level programming

Example 1: QCL – focus on quantum computing

Advanced quantum memory management using types:

- ▶ `qureg` — basic type for quantum registers,
- ▶ `quconst` — cannot be modified,

➤ High-level programming

Example 1: QCL – focus on quantum computing

Advanced quantum memory management using types:

- ▶ `qureg` — basic type for quantum registers,
- ▶ `quconst` — cannot be modified,
- ▶ `quvoid` — has to be empty before the call,

➤ High-level programming

Example 1: QCL – focus on quantum computing

Advanced quantum memory management using types:

- ▶ `qureg` — basic type for quantum registers,
- ▶ `quconst` — cannot be modified,
- ▶ `quvoid` — has to be empty before the call,
- ▶ `quscratch` — has to be empty before and after the call.

➤ High-level programming

Example 1: QCL – focus on quantum computing

Types of quantum functions:

- ▶ procedure — classically controlled quantum computation,

➤ High-level programming

Example 1: QCL – focus on quantum computing

Types of quantum functions:

- ▶ `procedure` — classically controlled quantum computation,
- ▶ `qfunct` — used to implement irreversible functions,

➤ High-level programming

Example 1: QCL – focus on quantum computing

Types of quantum functions:

- ▶ `procedure` — classically controlled quantum computation,
- ▶ `qfunct` — used to implement irreversible functions,
- ▶ `operator` — compound quantum operation.

➤ High-level programming

Example 1: QCL – focus on quantum computing

Some examples...

➤ High-level programming

Example 2: cQPL – focus on quantum communication

- ▶ QPL (formal specification) and cQPL (implementation of QPL, <https://arxiv.org/abs/quant-ph/0511145>)

➤ High-level programming

Example 2: cQPL – focus on quantum communication

- ▶ QPL (formal specification) and cQPL (implementation of QPL, <https://arxiv.org/abs/quant-ph/0511145>)
- ▶ Functional paradigm.

➤ High-level programming

Example 2: cQPL – focus on quantum communication

- ▶ QPL (formal specification) and cQPL (implementation of QPL, <https://arxiv.org/abs/quant-ph/0511145>)
- ▶ Functional paradigm.
- ▶ No publicly available implementation.

➤ High-level programming

Example 2: cQPL – focus on quantum communication

- ▶ QPL (formal specification) and cQPL (implementation of QPL, <https://arxiv.org/abs/quant-ph/0511145>)
- ▶ Functional paradigm.
- ▶ No publicly available implementation.
- ▶ Syntax for creating quantum communication channels by sharing (entangled) qubits.

➔ What next?

- ▶ **IF D-Wave GOTO** Andy Mason and Sheir Yarkoni, Tutorial on programming the D-Wave system
- ▶ **IF IBM GOTO** Ram Dušić Hren, IBM Q Experience: Hands-on workshop
- ▶ **IF Rigetti GOTO** Adam Szady, Quantum programming with (Py)Quil



Q?

`https://github.com/jmiszczak/qprog-tutorial`