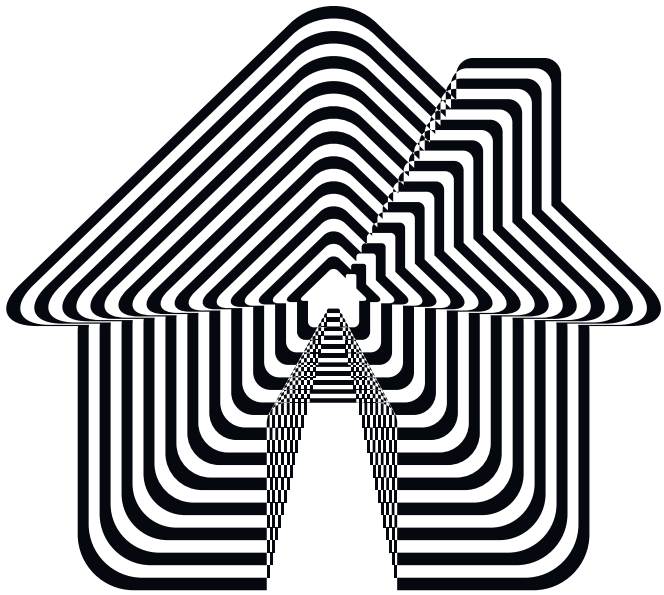
Quantenspiele, IBMs Qiskit und RasQberry

Eigenbau

David Jacob Drexlin, Jana Foehlisch, Isabell Heider, Dr. Jan-Rainer Lahmann, Lennart Schulze



Eigene Erfahrungen mit Quantenalgorithmen und -computern können Interessierte mit IBMs Qiskit-Umgebung sammeln. Das geht nicht nur im Web und auf einem klassischen Computer, sondern sogar mit einem Raspberry Pi.

Das auf der Quantenmechanik basierende Quantencomputing bietet das Potenzial für Anwendungen, die mit klassischen Computern nicht realisierbar sind. Allerdings ist die Technik selbst für viele IT-Experten schwierig zu verstehen. Neue Algorithmen und eine neue Art des Denkens sind erforderlich, um die mögliche Leistung der kommenden Quantencomputer auszuschöpfen. Dies verlangt neue Ansätze, Quantencomputing anschaulich und verständlich zu erklären.

Möglichkeiten gibt es viele, sich der neuen Technik zu nähern, sie zu verstehen und dafür zu entwickeln: angefangen bei Serious Games, die an das Thema spielerisch heranführen, über vielfältige Möglichkeiten mit IBMs freiem Quantum Software Development Kit Qiskit bis hin zum Bau eines Quantencomputermodells. Für diesen Artikel fiel die Wahl auf Qiskit, da es neben Simulatoren einen kostenfreien Zugang zu realen Quantencomputern in der Cloud bietet.

So spielen, dass man immer gewinnt

Als ein erstes Beispiel für Serious Games sei das Quanten-Münzspiel betrachtet, das die quantenphysikalischen Prinzipien Superposition und Interferenz nutzt. Zwei Spieler Alice und Bob drehen hierbei nacheinander ein und dieselbe Münze um oder belassen sie in der aktuellen Position. Die Anfangsposition ist „Kopf“. Zeigt die Münze nach insgesamt drei Zügen Kopf, gewinnt Alice; liegt Zahl oben, siegt Bob. Alle Züge erfolgen verdeckt, und die Zugreihenfolge ist immer gleich: Alice beginnt, dann ist Bob an der Reihe und zuletzt spielt Alice ein zweites Mal. Weitere Regeln gibt es nicht.

Übersetzt in die Quantenwelt repräsentiert ein Qubit die Münze, wobei Kopf dem Zustand $|0\rangle$ entspricht und Zahl für $|1\rangle$ steht. Zum Wenden der Münze dient ein Pauli-X-Gatter, das den Zustand des Qubits umkehrt. Soll die Lage der Münze gleich bleiben, kommt ein Identitätsgatter zum Einsatz, das den Zustand unverändert lässt. Da weder Alice noch Bob wissen, wie der andere gerade entschieden hat, gewinnen beide mit einer Wahrscheinlichkeit von 50 Prozent.

Durch die Besonderheiten eines Quantencomputers, in diesem Fall die Superposition, lässt sich die Gewinnwahrscheinlichkeit beeinflussen und verändern. Dazu darf Alice zusätzlich ein Hadamard-Gatter benutzen, das das Qubit in eine Superposition versetzt: Die Münze steht dadurch zwischen Kopf und Zahl. Wendet Alice das Hadamard-Gate in beiden Zügen an, beträgt ihre Gewinnwahrscheinlichkeit nun 100 %, weil sich das Qubit während Bobs Zug in der Superposition befindet, die weder ein X- noch ein Identitätsgatter beeinflusst. Gleichgültig welchen Spielzug Bob also wählt, er hat keine Chance, zu gewinnen. Das Spiel lässt sich online spielen (siehe ix.de/zfuw).

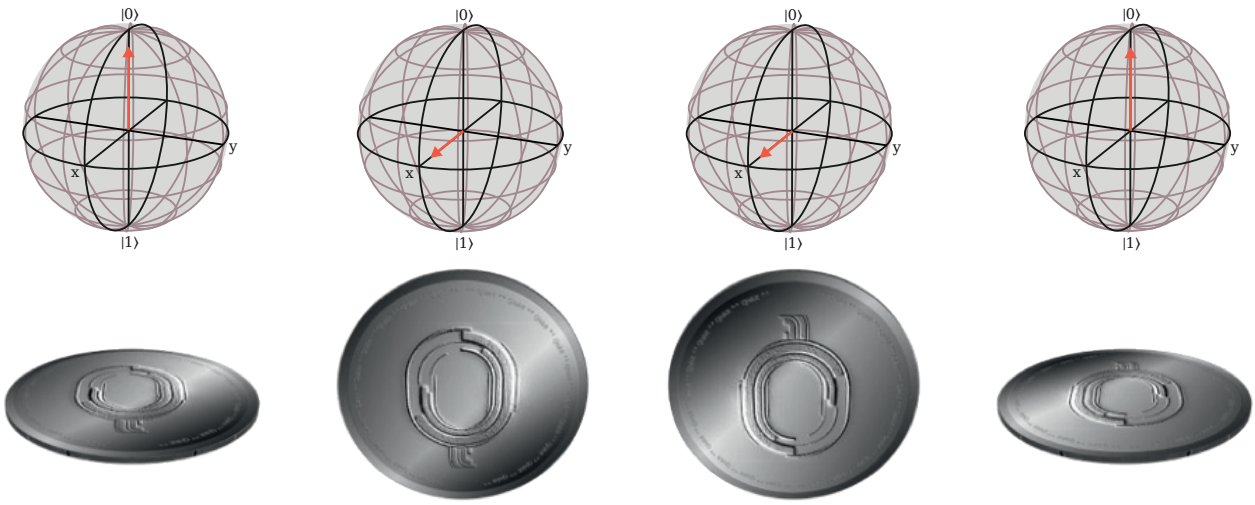
Anhand der Bloch-Sphäre lässt sich der Spielverlauf veranschaulichen: Vom Ausgangszustand $|0\rangle$ aus erzeugt das Hadamard-Gatter eine Superposition, die auf dem Äquator der Kugel liegt (siehe Abbildung 1). Durch das Anwenden eines cNOT-Gatters ändert sich am tatsächlichen Zustand der Superposition nichts; in der Veranschaulichung dreht sich die Münze auf der Kante. Eine genauere Erklärung sowie die exakten Formeln dazu finden sich am Ende des Online-Notebooks (siehe ix.de/zfuw).

Durch verschränkte Qubits in die Freiheit

Ein weiteres Quantenspiel, das nach Greenberger, Horne und Zeilinger benannte GHZ, veranschaulicht und nutzt das quantenmechanische Prinzip der Verschränkung. Dabei befinden sich drei Spieler in unterschiedlichen Räumen, die sie erst ver-

iX-TRACT

- Der Einstieg in das Rechnen mit Qubits und das Schreiben von Quantenalgorithmen ist schwierig. Helfen sollen dabei IBMs SDK Qiskit und diverse Onlinequellen.
- Damit lassen sich unter anderem Spiele bauen, bei denen man dank Quantenoperationen und -verschränkung zuverlässig gewinnt.
- Wer gerne bastelt, kann sogar einen Raspberry mit Qiskit ausrüsten und darauf einen Quantencomputer simulieren – samt LEDs zur Anzeige gemessener Qubits.



Die Kombination der Gatter H, cNOT und H stellt sicher, dass Alice immer als Gewinnerin aus dem Quanten-Münzspiel hervorgeht. Das liegt daran, dass cNOT das Qubit auf dem Äquator der Bloch-Sphäre belässt (Abb. 1).

lassen dürfen, wenn sie jeweils eine Frage beantwortet haben: Entweder bekommen alle die Frage nach einer Farbe gestellt oder zwei die nach einer Form und einer die nach einer Farbe. Die Farben sind Rot und Blau, die Formen Stern oder Rechteck. Sollen alle Spieler eine Farbe wählen, kommen sie frei, wenn eine gerade Anzahl von ihnen mit „Rot“ antwortet. Die erfolgreichen Kombinationen sind also Rot – Rot – Blau und Blau – Blau – Blau. Im anderen Fall muss die Summe der mit „Rot“ oder „Stern“ Antwortenden ungerade sein, sodass sie Rechteck – Rechteck – Rot, Rechteck – Stern – Blau und Stern – Stern – Rot in die Freiheit führen. Die Reihenfolge der Antworten ist irrelevant.

Die Spieler dürfen nicht miteinander kommunizieren, aber sich vorher auf eine Gewinnstrategie einigen. Auch bei diesem Spiel ist es nicht möglich, eine klassische Strategie zu finden, die immer zum Sieg führt. Die Spieler können nämlich mit einer abgesprochenen Strategie in maximal drei der vier möglichen Fälle korrekt antworten (siehe Abbildung 2). Um Genaueres dazu zu erfahren und das Spiel selber auszuprobieren, lohnt sich ein Blick auf das online verfügbare Jupyter-Notebook (siehe ix.de/zfuw).

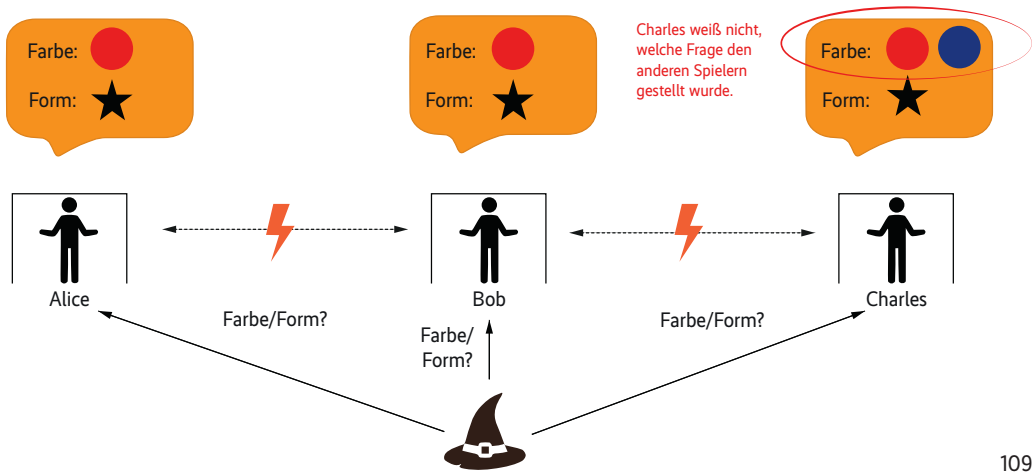
Mit einer Quantenstrategie lässt sich diese zunächst unlösbar erscheinende Aufgabe jedoch bewältigen, sodass die Spieler immer freikommen. Zunächst bringt man dafür drei Qubits in einen verschränkten Zustand mit starken Abhängigkeiten und Korrelationen zwischen den Qubits (GHZ-Zustand) und gibt jedem Spieler eines davon. Des Weiteren müssen die Spieler mit „Blau“ oder „Rechteck“ antworten, wenn ihr Qubit im Zustand $|0\rangle$ gemessen wird, und mit „Rot“ oder „Stern“ im Zustand $|1\rangle$. Im GHZ-Zustand können die Qubits nur noch als zusammenhängendes System und nicht mehr voneinander unabhängig betrachtet werden. Das entspricht aber keiner Kommunikation und verstößt deshalb nicht gegen die Regeln. Um den GHZ-Zustand zu erreichen, wendet man auf ein Qubit zunächst ein H-Gatter an. Die anderen beiden Qubits verschränkt dann jeweils ein cNOT-Gatter mit dem ersten. Je nach Frage wendet man danach unterschiedliche Gatter auf das System an. Sie lassen in der anschließenden Messung das System so zusammenfallen, dass nur die gewinnführenden Kombinationen als Ergebnisse abgebildet werden können. Um alle Einzelheiten des Spiels zu verstehen und es selbst auf echten Quantencomputern zu spielen, empfiehlt sich ein Blick in das Online-Repository „Fun with Quantum“ (siehe ix.de/zfuw). Hier finden sich auch weitere Notebooks zum Thema Quantencomputing.

Schon diese beiden überschaubaren Beispiele verdeutlichen, welches Potenzial in Quantencomputing steckt: Beide Spiele bieten plötzlich zuverlässige Gewinnstrategien, obwohl das im klassischen Sinne unmöglich ist.

Algorithmen für Spiele und anderes implementieren

IBMs Open-Source-Framework Qiskit enthält Werkzeuge zum Erstellen und Ausführen von Algorithmen auf realen Quantencomputern in der Cloud oder auf lokalen Simulatoren. Mit der grafischen Oberfläche des IBM Quantum Composer lassen sich Algorithmen entwerfen, und das IBM Quantum Lab dient zum Entwickeln und Ausführen von Qiskit-Code in Jupyter-Notebooks.

Das Programmieren mit Quantencomputern folgt mehreren Paradigmen. Als Modell der physikalischen Steuerung ist das Quantenschaltkreismodell am gebräuchlichsten und arbeitet in IBMs Quantencomputern. Daher basieren die im Folgenden vorgestellten Verfahren auf diesem Prinzip, das Schaltkreise aus Quantengattern und Messungen bildet. Diese Gatter repräsentieren unitäre Operationen, die auf Qubits angewendet deren Quantenzustände verändern.



Mit einer abgesprochenen Strategie können Spieler das GHZ-Spiel in 75 Prozent der Fälle gewinnen (Abb. 2).

Programmieren mit OpenQASM

Quantenprogramme lassen sich auch in der Open Quantum Assembly Language (OpenQASM) schreiben. IBM nutzt die Sprache für ihre „Quantum Experience“. Dieses Angebot erfordert eine Anmeldung und lässt sich dann komfortabel direkt im Browser nutzen (siehe ix.de/zfuw). OpenQASM orientiert sich nicht an Sprachen wie Python, sondern eher an Beschreibungssprachen für klassische Hardware wie Verilog. Nach einer Präambel mit Versionsinformationen importieren OpenQASM-Programme die nötigen Bibliotheken:

```
OPENQASM 2.0;
Include "qelib1.inc";
```

Als einfaches Beispiel diene der Bell-Zustand, bei dem zwei Qubits verschränkt sind. Um ihn herzustellen, definiert man ein Quantenregister mit zwei Qubits und ein klassisches Register mit zwei normalen Bits, das die Messergebnisse aufnehmen wird:

```
qreg q[2];
creg c[2];
```

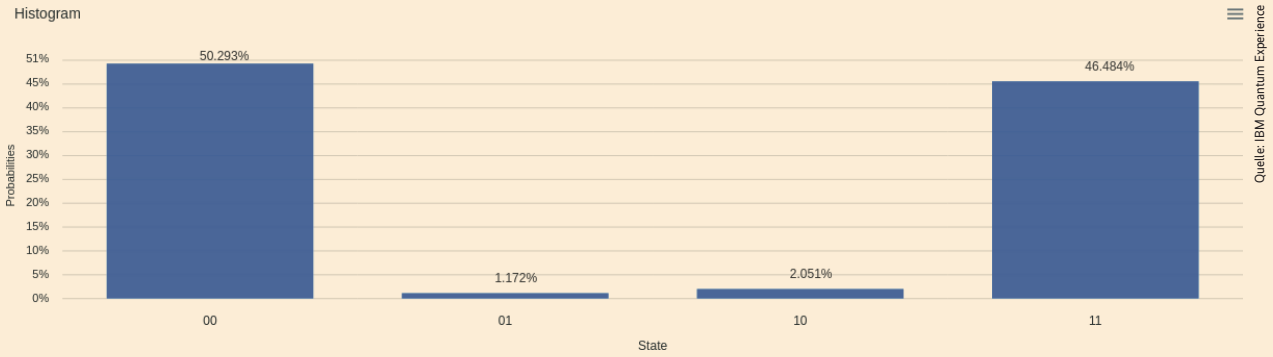
Dann wird die Hadamard-Transformation (`h` in OpenQASM) auf das Qubit 0 angewendet und danach cNOT (`cx`) auf die Qubits 0 und 1:

```
h q[0];
cx q[0],q[1];
```

Bei der anschließenden Messung beider Qubits landen die Ergebnisse in den zugehörigen klassischen Bits:

```
measure q[0] -> c[0]
measure q[1] -> c[1]
```

Auf welcher Hardware das Programm läuft, also ob auf einem echten oder simulierten Quantenchip, stellt man im Interface von Quantum Experience ein. Schaltkreise kann man dort auch rein grafisch definieren. Führt man das kleine Beispiel viele Male aus, ergibt wie erwartet ungefähr je die Hälfte der Messungen den Zustand |00⟩ und den Zustand |11⟩. Auf einem echten Quantenchip gibt es auch ein paar fehlerhafte Ergebnisse für |01⟩ und |10⟩. *Dr. Florian Neukart*



Dieses Ergebnis hat ein echter Quantencomputer produziert. Wie erwartet hat er die Qubits ungefähr zur Hälfte in den Zuständen |00⟩ und |11⟩ gemessen. Aber weil sich Qubits noch lange nicht so zuverlässig wie klassische Bits manipulieren und lesen lassen, haben ein paar Messungen |01⟩ und |10⟩ ergeben (Abb. 3).

Im grafischen Editor des Quantum Composer platziert man Quantengatter per Drag-and-Drop auf den horizontalen Qubit-Bahnen und wendet sie so auf die Qubits an (siehe Abbildung 4). Außerdem lassen sich benutzerdefinierte Gatter erstellen. Eine Veränderung im Quantenzustand der Qubits durch Gatter beeinflusst ihre Messwahrscheinlichkeiten, das verwertbare Ergebnis eines Quantenalgorithmus. Balkendiagramme und die sogenannte Q-Sphere visualisieren diese Messwahrscheinlichkeiten (siehe Abbildung 5). Die Q-Sphere kann anders als die Bloch-Kugel den Quantenzustand von mehr als einem Qubit darstellen.

Den GHZ-Zustand erzeugt man im Quantum Composer so: Zunächst platziert man ein Hadamard-Gatter, das das erste Qubit in eine Superposition bringt, also eine Linearkombination aus den Basiszuständen |0⟩ und |1⟩. Es folgen zwei cNOT-Gatter von Qubit 1 zu 2 und von 1 zu 3 (siehe Abbildung 6). Sie stellen sicher, dass eine Messung von Qubit 1 immer mit Messungen von Qubit 2 und 3 übereinstimmt. Damit erhält man die zulässigen Ergebnisse |000⟩ und |111⟩ mit gleicher Wahrscheinlichkeit.

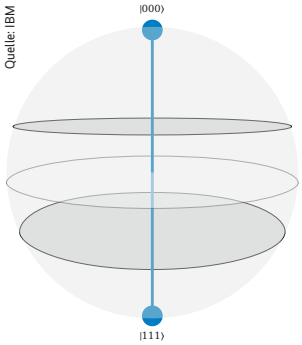
Ähnlich lässt sich dieser Quantenschaltkreis mit Python-Code in Qiskit abbilden (siehe Listing). Er beginnt mit dem Import der benötigten Bibliotheken, gefolgt vom Erstellen eines Quanten- und eines klassischen Registers mit je drei Elementen sowie dem Definieren eines lokalen Simulators als Backend für die Ausführung. Den Schaltkreis erzeugen Aufrufe von `qc()`, die die erwähnten Gatter als Parameter übergeben bekommen. Die Messung des Zustandes und ihre Darstellung beenden das Programm. Statt des hier verwendeten fehlerfrei arbeitenden Simulators lassen sich reale Quantencomputer in der Cloud verwenden, die Details sind online beschrieben.

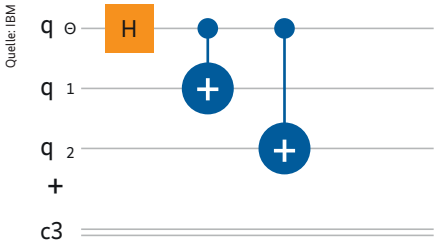
Tiefer einsteigen in Quantencomputing mit Qiskit Textbook

Quantencomputer ermöglichen nicht nur ungeahnte Dinge, sondern folgen auch anderen Paradigmen als die klassische Informatik. Da die Orientierung darin nicht gerade leicht ist, rief IBM das Qiskit Textbook ins Leben. Dieses frei zugäng-

Im IBM Quantum Composer sind klassische Gatter dunkelblau, spezielle Phasengatter hellblau und nicht unitäre Operationen grau (Abb. 4).

Die Q-Sphere veranschaulicht den Zustand eines Systems aus mehreren Qubits. Die Größe der Knoten ist proportional zur jeweiligen Messwahrscheinlichkeit. Beim GHZ-Zustand sind diese Werte für |000⟩ und |111⟩ jeweils gleich groß, also je 50 Prozent (Abb. 5).





Im grafischen Editor für Quantenschaltkreise kann man einen GHZ-Zustand erstellen (Abb. 6).

liche interaktive Onlinelehrbuch bietet praktische und wissenschaftliche Erläuterungen zum gesamten Quanten-Stack. Seinem Open-Source-Charakter entsprechend stellen Fachleute sicher, dass der kollaborativ erstellte Inhalt sowohl Grundlagen als auch neueste Forschungserkenntnisse umfasst. Besonders wegen seines niedrigschwelligen Zugangs und seiner Praxisorientierung – Code-Snippets lassen sich direkt im Browser ausprobieren – ist das Qiskit Textbook als Einstiegspunkt in das Quantencomputing und zur Vertiefung geeignet.

Es beschäftigt sich mit den Kernthemen der aktuellen Forschung zu Quantencomputern: Informationstheorie, Algorithmen und Anwendungen, Fehlerreduktion und Hardwareanalyse. Zur ersten Gruppe gehört, wie Quantensysteme Informationen speichern und darstellen. Algorithmen wiederum sind wiederverwendbare Bausteine oder

Listing: GHZ-Zustand in Qiskit erstellen

```
#Bibliotheken importieren
import qiskit

# Quantenschaltkreis mit drei Qubits erstellen
q = QuantumRegister(3)
# ein klassisches Register erstellen, das die Ergebnisse der Messung enthält
c = ClassicalRegister(3)
# Definition des Backends, hier ein Quantensimulator
backend = BasicAer.get_backend('qasm_simulator')

qc = QuantumCircuit(q, c)
# GHZ-Zustand erzeugen
qc.h(q[0])          # H-Gatter auf das erste Qubit anwenden
qc.cx(q[0],q[1]) # die ersten beiden Qubits verschränken
qc.cx(q[0],q[2]) # das erste und das letzte Qubit verschränken
qc.barrier()      # optische und inhaltliche Trennung (optional)
qc.draw(output='mpl')
```

Praktische Arbeit mit Qiskit

Mit Jupyter-Notebooks lassen sich Algorithmen und Abläufe in Qiskit schnell entwickeln und Codefragmente erarbeiten. Man führt sie vom Notebook aus entweder in einem der Simulatoren oder direkt auf einer von IBM bereitgestellten Quantencomputerinstanz aus. Qiskit-Bibliotheken können auch Zwischenschritte visualisieren, beispielsweise Quantenschaltungen mit den dazugehörigen Gattern (siehe Abbildung 7).

Mit Tools wie PyCharm oder Plug-ins für Eclipse stehen vollwertige IDEs bereit, die alle Unterstützungsmöglichkeiten für den Entwicklungsprozess bieten. Eine Momentaufnahme der Quantenprogrammierung für das Problem des Handlungsreisenden sieht etwa so aus:

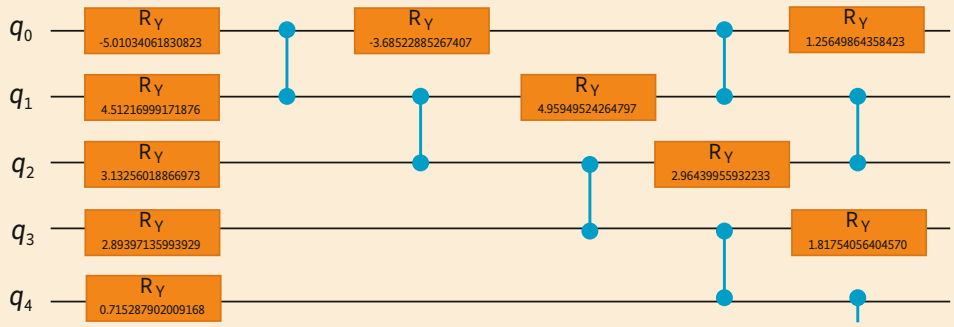
```
…
qp = QuadraticProgram()
qp.from_ising(qubitOp, offset, linear=True)
qp.to_docplex().prettyprint()

result = exact.solve(qp)
print(result)

ee = NumPyMinimumEigensolver(qubitOp)
result = ee.run()
x = sample_most_likely(result.eigenstate)
print('feasible:', tsp.tsp_feasible(x))
z = tsp.get_tsp_solution(x)
print('solution:', z)
print('solution objective:', tsp.tsp_value(z, ins.w))
```

In diesem Beispiel hilft PyTest als Unit-Testing-Framework, die Qualität der Implementierung in verschiedenen Umgebungen kontinuierlich zu überprüfen.

Für die Entwicklungswerkzeuge erscheinen regelmäßig Updates und Erweiterungen. Man kann leicht mit den verschiedenen implementierten und bekannten Algorithmen wie Grover und Bernstein-Vazirani oder mit spezialisierten Verfahren aus dem Finanzwesen, der Chemie und der Graphentheorie experimentieren.

Zum Einstieg in die Quantenprogrammierung eignet sich der Grover-Algorithmus: Er sucht in unsortierten Datenbanken und hat dabei einen quadratischen Geschwindigkeitsvorteil gegenüber klassischen Algorithmen: Die Anzahl der Schritte zum Finden eines Eintrags in einer unsortierten Datenbank ist durch die Quadratwurzel der Einträge begrenzt. Eine lineare Suche auf einem klassischen Rechner benötigt hingegen im ungünstigsten Fall ebenso viele Schritte, wie es Einträge gibt (siehe Kasten „Grover-Algorithmus“ im Artikel „Ansichtssache“ auf Seite 14). Qiskit implementiert den Grover-Algorithmus und mehrere davon inspirierte Funktionen, etwa den Grover-Operator für Optimierungsprobleme.

Im Fall eines 3-Qubit-Grover reichen die codierten, zu durchsuchenden Werte von null bis sieben. Zum Vergleich des gesuchten Werts mit den



Qiskit-Bibliotheken können unter anderem Quantenschaltungen mit ihren Gattern visualisieren, hier ein Ausschnitt (Abb. 7).

zweckgebundene Applikationen. Zu den besonders bedeutsamen gehören die Quantum Phase Estimation und der HHL-Algorithmus, die unter anderem Eigenwerte einer Matrix berechnen und Lösungen linearer Gleichungssystemen finden können (siehe Artikel „Richtungsweisend“ auf Seite 86). Letzteres nutzen beispielsweise Optimierungsverfahren und maschinelles Lernen. Ergebnisse dieser Algorithmen sind besonders beachtlich, weil die verfügbare Hardware sich noch in einem frühen Stadium befindet.

Bislang bieten NISQ-Devices (Noisy Intermediate Scale Quantum) nur bis zu knapp 70 Qubits. Gleichzeitig unterliegen die Prozessoren hardwareinduzierten Fehlerquellen: Die fragilen Quantenzustände müsste man bei einer Temperatur nahe dem absoluten Nullpunkt komplett von der Umwelt isolieren, um sie aufrechtzuerhalten. Darüber hinaus können die physikalischen Operationen, die den Schritten des Algorithmus entsprechen, nicht akkurat durchgeführt werden, sodass das Ergebnis eines Algorithmus mit zunehmender Gatterzahl von seinem Erwartungswert abweicht. Es gibt zwar Verfahren, die diesen Effekten theoretisch entgegenwirken (siehe Artikel „Unumgänglich“ auf Seite 32). Sie benötigen jedoch viele zusätzliche Qubits, was die Umsetzung erschwert. Quantum Error Mitigation hingegen, eine Familie von klassischen und Quantenalgorithmen, erlaubt es heute schon, die Genauigkeit eines Quantencomputers zu verbessern. So unterliegt auch das GHZ-Spiel Einflüssen, die die Zuverlässigkeit reduzieren, mit der man es auf einem echten Quantencomputer gewinnt. Mit direkt in Qiskit umgesetzten Methoden lässt sich diese jedoch erheblich verbessern. Alle hier verwendeten Ansätze stehen online zum Ausprobieren bereit.

Als Basis zur Implementierung der Qubits in diesen neuartigen Rechnern eignen sich physikalische Bauelemente, die auf der Ebene der Quantenmechanik gezielt gesteuert werden können. So lassen sich damit Superposition, Verschränkung und Quantengatter realisieren. Weil dazu zahlreiche Möglichkeiten bestehen, befasst sich ein Teil der Hardwareforschung mit der Leistungsmessung von Quantencomputern. Neben der Anzahl der Qubits sind besonders die Aspekte wichtig, die die Fehleranfälligkeit bestimmen. Eine Maßzahl dafür ist das Quantenvolumen: Es fasst unter anderem die Konnektivität zwischen Qubits, die Kohärenzdauer eines Quantenzustands und die Fehleranfälligkeit in Operationen zusammen (siehe

Datenbankeinträgen dient ein „Orakel“: eine Funktion, die das gesuchte Element markiert und erkennt, falls es ihm als Eingabe präsentiert wird. Die Implementierung nutzt eine von Maslov und anderen beschriebene Schaltung, die mit IBMs Quantum Composer für eine Datenbank mit acht Elementen nachgebaut wurde (siehe Abbildung 8 und ix.de/zfuw). Aus diesem Schaltkreis generiert das Composer-Panel direkt den dazugehörigen OpenQASM-Code, den ein Klick in Qiskit überführt.

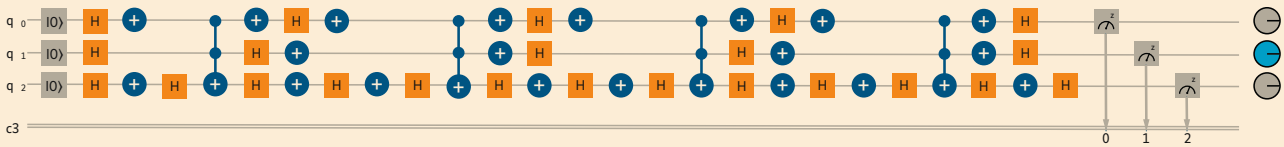
```
OPENQASM 2.0; include "qelib1.inc";
qreg q[3]; creg c[3];   // 3 Qubit Registers
h q[0]; h q[1]; h q[2]; // Initialization of Qbits in superposition
x q[0]; x q[2]; h q[2]; ccx q[0],q[1],q[2]; h q[2]; x q[0]; x q[2]; 7
                                                    // Oracle for 010
h q[0]; h q[1]; h q[2]; x q[0]; x q[1]; x q[2]; h q[2]; ccx q[0],q[1]7
                                  ,q[2]; h q[2]; //Amplification part
x q[0]; x q[1]; x q[2]; h q[0]; h q[1]; h q[2]; //Amplification part
x q[0]; x q[2]; h q[2]; ccx q[0],q[1],q[2]; h q[2]; x q[0]; x q[2]; 7
                                                    // Oracle for 010
h q[0]; h q[1]; h q[2]; x q[0]; x q[1]; x q[2]; h q[2]; ccx q[0],q[1]7
                                  ,q[2]; h q[2]; // Amplification part
x q[0]; x q[1]; x q[2]; h q[0]; h q[1]; h q[2]; // Amplification part
measure q[0] -> c[0]; measure q[1] -> c[1]; measure q[2] -> c[2]; 7
                                                    // Measurement
```

In diesem Beispiel ist das Element |010⟩ in den acht möglichen Kombinationen zu finden. Das Quantenorakel und Phasenverstärkungen machen es wahrscheinlicher, dass das vom Orakel markierte Element am Schaltkreisausgang zu sehen ist. Im Composer-Panel sind die simulierten erwarteten Ergebnisse dieses Algorithmus erkennbar – also das gesuchte Element |010⟩. Gerade für kleinere Programme ist es praktisch, das erwartete Ergebnis direkt zu sehen, sodass sich Fehler einfacher finden lassen.

Programmieren auf einem Quantencomputer ist nicht einfach und erst recht nicht intuitiv. OpenQASM und das Qiskit-Framework liefern viele Funktionen und hilfreiche Einführungen, die den Einstieg erleichtern. Erfahrenere Nutzer können Qiskit in etablierte Software und Entwicklungsumgebungen integrieren, wie am Beispiel des Handlungsreisenden gezeigt. Ob sich die verfügbaren Tools und Programmiersprachen auch für größere Produkte mit umfangreichem, teamübergreifend entwickeltem Code eignen, muss sich jedoch erst noch zeigen.

Die Quantenhardware jedenfalls wird sich auch weiterhin rasch weiterentwickeln. Beispielsweise wird die Zahl der Qubits steigen. Einige Prognosen sprechen, wie bei Moores Gesetz, von einem exponentiellen Wachstum mit einer Verdopplung alle ein bis zwei Jahre. Die Zuverlässigkeit der Berechnungen wird zunehmen und die Fehlerrate sinken. Es ist bereits erkennbar, dass diese Weiterentwicklung mit einer hohen Komplexität der Quantenalgorithmen und des Programmcodes einhergeht. Folglich ist das Entstehen neuer Programmiersprachen, Programmiermodelle und Entwicklungswerkzeuge zu erwarten, die diese erhöhte Komplexität handhabbar machen.

Ilie-Daniel Georghe-Pop, Nikolay Tcholtchev, Sebastian Bock



Aus dieser Quantenschaltung generiert das Composer-Panel direkt den dazugehörigen OpenQASM-Code für den Grover-Algorithmus (Abb. 8).

Artikel „Kaum verschränkt“ auf Seite 44). Genauer lässt sich die Leistungsfähigkeit von Quantencomputern durch Messen zufälliger Testschaltkreise, durch die informationstheoretische Analyse von Algorithmenergebnissen und durch Bestimmen physikalischer Hardwareeigenschaften beschreiben. In dieses Thema führt Kapitel 5 des Qiskit Textbook ein.

Quantencomputer aus supraleitenden Schaltkreisen werden mit ultrapräzisen Mikrowellenpulsen gesteuert, die den Zustand der Qubits manipulieren. Die Entwicklung und Kalibrierung dieser Pulse ist ein Forschungsgebiet, in dem bisher nur Labore mit Zugriff auf die Hardware aktiv waren. Qiskit Pulse bietet die Technik jedem an: Benutzer können dadurch nicht mehr nur die vorgegebenen Gatter verwenden, sondern die Operationen und Messungen auch direkt auf Pulsebene implementieren. Dies wiederum ermöglicht es ihnen, Quantenprogramme auf den verfügbaren Systemen zu optimieren.

Noch einen Schritt weiter geht Qiskit Metal. Die ebenfalls freie Software dient zum Automatisieren des Elektronikdesigns für Quantencomputer aus supraleitenden Qubits. Dafür lassen sich vorgefertigte und benutzerdefinierte Komponenten verwenden, die Qubits und Steuerungsleitungen zwischen ihnen verbessern oder zum Entwickeln neuer Quantentechnik beitragen. Ebenso trägt Qiskit Metal dazu bei, Quantenhardware besser zu verstehen. Nach dem Designen eigener Hardware kann man darauf Analysen durchführen und so etwa die elektromagnetischen Eigenschaften sowie das Energieeigenspektrum des Bauelements bestimmen, was bei weiteren Optimierungen hilft.

Quantencomputing für zu Hause: RasQberry

Obwohl es noch nicht möglich ist, echte Quantencomputer zu Hause zu betreiben, kann man zumindest ein Modell selbst bauen, auf dem Demoprogramme laufen. Basis dafür ist das Open-Source-Projekt RasQberry. Der Name verweist auf einen Raspberry Pi als Basis. Darauf lässt sich Qiskit installieren, sodass dem Minirechner ein Quantencomputing-Simulator zur Verfügung steht. Der Zugriff auf die echten Quantencomputer in der IBM-Cloud ist ebenfalls möglich (siehe Artikel „Spielplatz“ auf Seite 60). Seinen besonderen Reiz erhält das Modell durch ein 3-D-gedrucktes Gehäuse,

Auf der Rückseite dieses RasQberry befindet sich der 4 Zoll große Touchscreen (Abb. 9).

das IBMs Quantum System One nachempfunden ist. Abgerundet wird das Ganze durch optionale Komponenten wie eine Acrylverglasung, LED-Beleuchtung, einen integrierten Akku und ein 4 Zoll großes Touchdisplay (siehe Abbildung 9). Alternativ zum Touchscreen lässt sich ein SenseHAT mit einem 8×8-LED-Display und diversen Sensoren verwenden.

Mehrere der hier vorgestellten Demoprogramme laufen bereits auf dem RasQberry, so die interaktive Visualisierung der Bloch-Kugel und der grafische Quantum Composer. Einige weitere Beispiele wie das Münz- und das GHZ-Spiel sind in Planung. Speziell für die Variante mit SenseHAT gibt es Demos, die auf den grundlegenden Quantenzuständen von Bell und Greenberger-Horne-Zeilinger (GHZ) beruhen oder eine Anzahl von Qubits in Superposition versetzen, den Zustand messen und anzeigen. Ist ein LED-Ring integriert, kann er nicht nur hübsch leuchten, sondern die Messung von bis zu 12 Qubits visualisieren, die sich in Superposition oder Verschränkung befinden können. Eine erweiterte Variante mit 27 LEDs ist in Planung. Im Internet finden sich erste Ideen, eine Kamera hinzuzufügen und deren Bilder per maschinellem Lernen mit Quantenalgorithmen auszuwerten.

Ein Bootstrap-Skript installiert die nötige Software auf dem RasQberry und stellt ein Konfigurationstool zur Verfügung. Es ähnelt dem des klassischen Raspberry Pi. Neben der Hardware – empfohlen wird das Modell 4 mit 2 GByte RAM, ist Raspberry Pi OS und eine SD-Karte mit mindestens 8 GByte erforderlich. Deren Initialisierung kann der Raspberry Pi Imager übernehmen. Ist eine SSH-Verbindung zum System hergestellt, genügen drei Kommandos für die initiale Installation:

```
pip install getgist
.local/bin/getgist JanLahmann RasQ-init.sh
. RasQ-init.sh
```

Alle für den 3-D-Druck des Gehäuses notwendigen Informationen sowie die erforderlichen STL-Dateien sind auf der GitHub-Seite des RasQberry-Projektes verfügbar (siehe ix.de/zfuw). Hier finden sich auch eine Installationsanleitung, Konfigurationsdateien und vieles mehr – und eine Open-Source-Community, die bei Fragen weiterhilft. (sun@ix.de)

Quellen

Onlinequellen zu Qiskit, RasQberry und mehr: ix.de/zfuw

David Jacob Drexlin

ist dualer Student der internationalen Wirtschaftsinformatik an der Dualen Hochschule Baden-Württemberg in Kooperation mit IBM.

Jana Foehlisch

studiert Digital Business Management an der Dualen Hochschule Baden-Württemberg und ist bei ihrem Partnerunternehmen IBM unter anderem als IBM Quantum Ambassador aktiv.

Isabell Heider

studiert IT-Sicherheit und Forensik im WINGS-Fernstudium an der Hochschule Wismar. Sie arbeitet als IT-Beraterin bei IBM und ist als Qiskit Advocate aktiv.

Dr. Jan-Rainer Lahmann

hat 1999 am KIT Karlsruhe in Mathematik promoviert und ist seitdem bei IBM Deutschland in der technischen Vertriebsunterstützung tätig. Er ist Qiskit Contributor und Gründer des RasQberry-Open-Source-Projektes.

Lennart Schulze

studiert internationale Wirtschaftsinformatik an der Dualen Hochschule Baden-Württemberg in Stuttgart. Als IBM Quantum Ambassador forscht er unter anderem zu Quantum Error Mitigation und Quantum Machine Learning.

Dr. Florian Neukart

ist Assistenzprofessor für Quantencomputing an der Universität Leiden und Direktor des Volkswagen Data:Lab in München.

Sebastian Bock

arbeitet als wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Offene Kommunikationssysteme FOKUS im Bereich Quantum Computing. Die Leidenschaft für den Fachbereich entwickelte er während seines Physikstudiums in Chemnitz und Siegen.

Ilie-Daniel Gheorghe-Pop

studierte Computerarchitekturen an der Polytechnischen Universität Bukarest. Seine Hauptaktivitäten im Bereich der angewandten Wissenschaften sind Leistungstests, Standardisierung und Entwicklung im IT-Bereich.

Dr. Nikolay Tcholtchev

ist Ingenieur und Diplom-Informatiker und arbeitet als Gruppenleiter beim Fraunhofer-Institut für Offene Kommunikationssysteme FOKUS unter anderem für Quantencomputing-Projekte.