

Einführung in das anwendungsbezogene Quantencomputing

GI-Rhein-Main / 25.06.2024

Dr. Matthias Heller, Dr.-Ing. Johannes Mueller-Roemer

Zentrum für Angewandtes Quantencomputing

Projektvorstellung

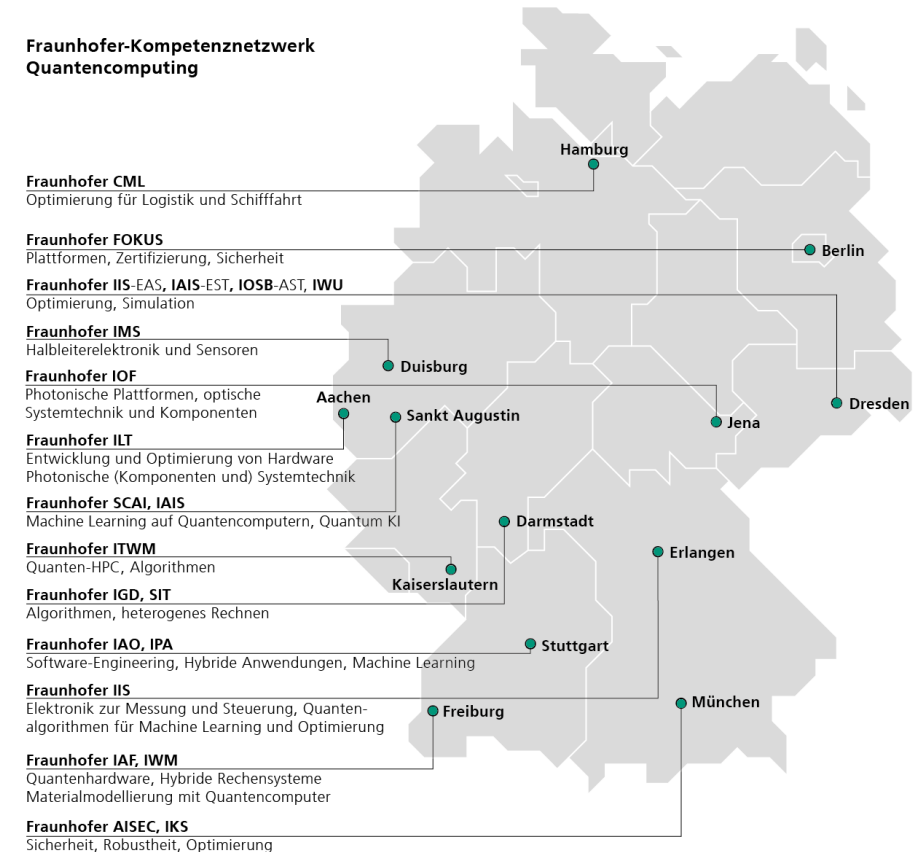
Was ist das ZAQC?

- Projekt des Fraunhofer IGD in Kooperation mit dem Fraunhofer SIT
- Gefördert vom Land Hessen (HMinD, HMWK) und Teil der hessischen Digitalstrategie
- Ansprechpartner für deutsche, insb. hessische, Wirtschaft und Industrie
- Ziel: Anwendungsmöglichkeiten von Quantencomputing identifizieren, bewerten und priorisieren
- Initialer Fokus auf chemisch/pharmazeutische Anwendungsbereiche
- Interdisziplinäres Team aus...
 - Informatikern
 - Physikern
 - Chemikern

Expertisen und aktuelle Arbeiten

Vorhandene Expertisen und aktuelle Arbeiten am ZAQC

- Langjährige Erfahrung im heterogenen Rechnen und physikalischer Simulation
- Teil des Fraunhofer-Netzwerks Quantencomputing
- Enge Kooperation mit TU-Darmstadt, insb.
 - Prof. Krewald (Quantenchemie)
 - Prof. Gachechiladze (Quantencomputing)
- Aktuelle Arbeiten
 - Measurement-based Quantum Computing
 - Benchmarking
 - Effizientere klassische und hybride Methoden



Grundlagen des Quantencomputing

Wie funktionieren Quantencomputer?

Grundlagen

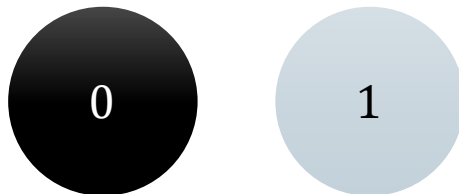
- *Quantencomputer* nutzen sogenannte *Quantum Bits*, kurz **Qubits**, für Berechnungen bzw. zum Lösen von Problemen
- Realisiert durch mikroskopische, physikalische Systeme, die den Gesetzen der Quantenphysik gehorchen
- Wie funktionieren Qubits und wie wird mit ihnen gerechnet?
- Muss man Quantenphysik verstehen, um Quantencomputer zu programmieren?
 - Nein, lineare Algebra, komplexe Zahlen und Wahrscheinlichkeitsrechnung reichen aus!

Bits und Qubits im Vergleich

Superposition

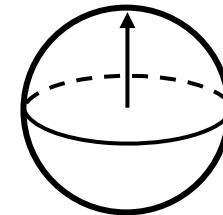
Bit

- Entweder 0 oder 1



Qubit

- 0 oder 1 oder eine beliebige Überlagerung beider Zustände (**superposition**)
- Bra-Ket-Notation
 $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $\langle\psi| = |\psi\rangle^H$, $\langle\psi|\varphi\rangle = |\psi\rangle^H|\varphi\rangle$
- Zustand ist eine „Mischung“ aus 0 und 1 mit den (komplexwertigen) Koeffizienten α und β
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

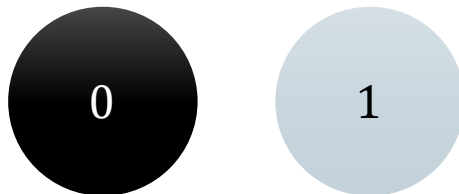


Bits und Qubits im Vergleich

Verschränkung

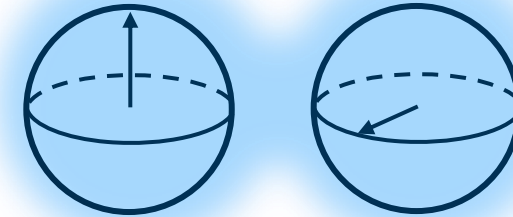
Bit

- Jedes Bit kann unabhängig geändert werden
- n Bits sind in *einem* von 2^n Zuständen



Qubit

- Es ist möglich, Qubits zu verschränken (**entanglement**)
 - Veränderung eines Qubits kann *sofort* den Zustand eines anderen ändern
- n Qubits sind in einer Überlagerung von 2^n Zuständen
$$\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$
 - Quantenparallelität steigt exponentiell!

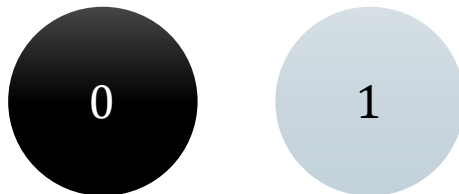


Bits und Qubits im Vergleich

Messung und No-Cloning

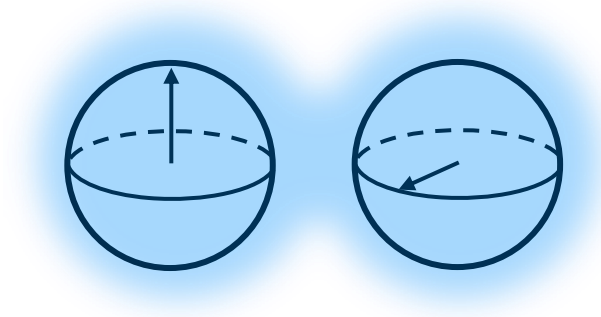
Bit

- Zustand
 - ist jederzeit eindeutig definiert
 - kann ausgelesen werden, ohne ihn zu ändern
- kann kopiert werden



Qubit

- Zustand
 - ist bei Messung 0 oder 1 mit Wahrscheinlichkeit
 $p_0 = |\alpha|^2, p_1 = |\beta|^2$
 - ändert sich durch Messung
 - betrifft alle verschränkten Qubits!
→ Zustand beschreibt Gesamtsystem
- kann nicht kopiert werden



Qubits verändern

Unitäre Operatoren

Wie beeinflusst man den Zustand von Qubit-Registern?

Physikalisch betrachtet gibt es verschiedene Möglichkeiten, abhängig von der physikalischen Realisierung. Konzeptionell entsprechen sie alle *unitären Operatoren*.

- Zustandsvektor $|\varphi\rangle \in \mathbb{C}^{2^n}$ muss Länge $\langle\varphi|\varphi\rangle = 1$ haben, da Summe der Wahrscheinlichkeitsamplituden 1 sein muss
- Operatoren (Matrizen!) U müssen die Länge erhalten, d.h.

$$\langle\varphi|U^H U|\varphi\rangle = 1$$

- Dies ist genau dann gegeben wenn die Operatoren unitär sind:

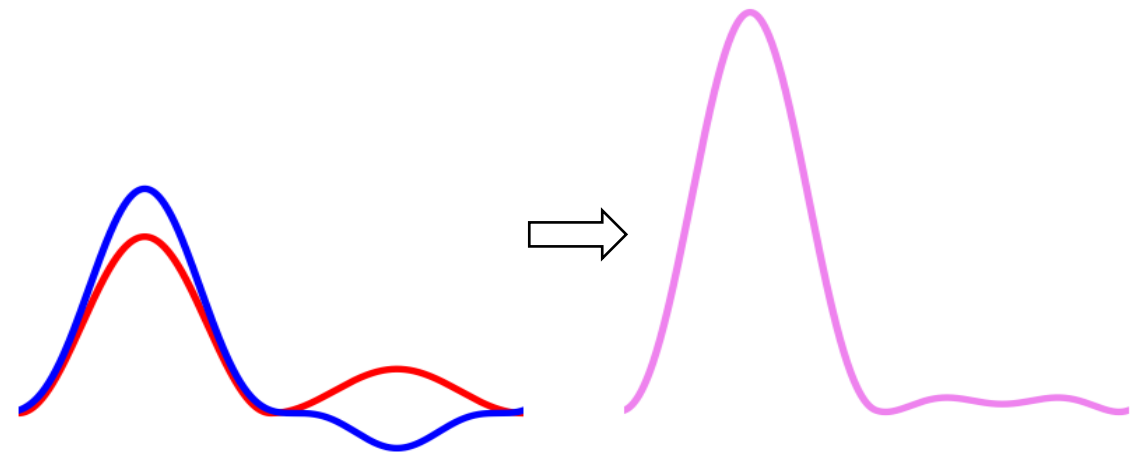
$$UU^H = U^H U = I$$

- Alle Änderungen, die nicht von einer Messung ausgehen, sind umkehrbar!
 - Es geht keine Information verloren
- Beispiele zur Veranschaulichung:
 - Rotationen
 - Spiegelungen

Quantencomputing

Wie Qubits für Computing nutzen?

- Wie kann man Qubits nutzen, wenn
 - sie nicht ausgelesen werden können, ohne ihren Zustand zu ändern?
 - nur eine probabilistische Messung durchgeführt werden kann?
- Wahrscheinlichkeitsamplituden $\alpha, \beta, \dots$ des Zustands $(\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle)$ beschreiben sog. *Wellenfunktion*, bei verschränkten Qubits „überlagern“ sich die Wellenfunktionen
 - Maximierung der Wahrscheinlichkeit des richtigen Ergebnisses durch *konstruktive* bzw. *destruktive* Interferenz (**interference**)
- ABER: Quantencomputer können nicht jedes Problem *effizient* lösen

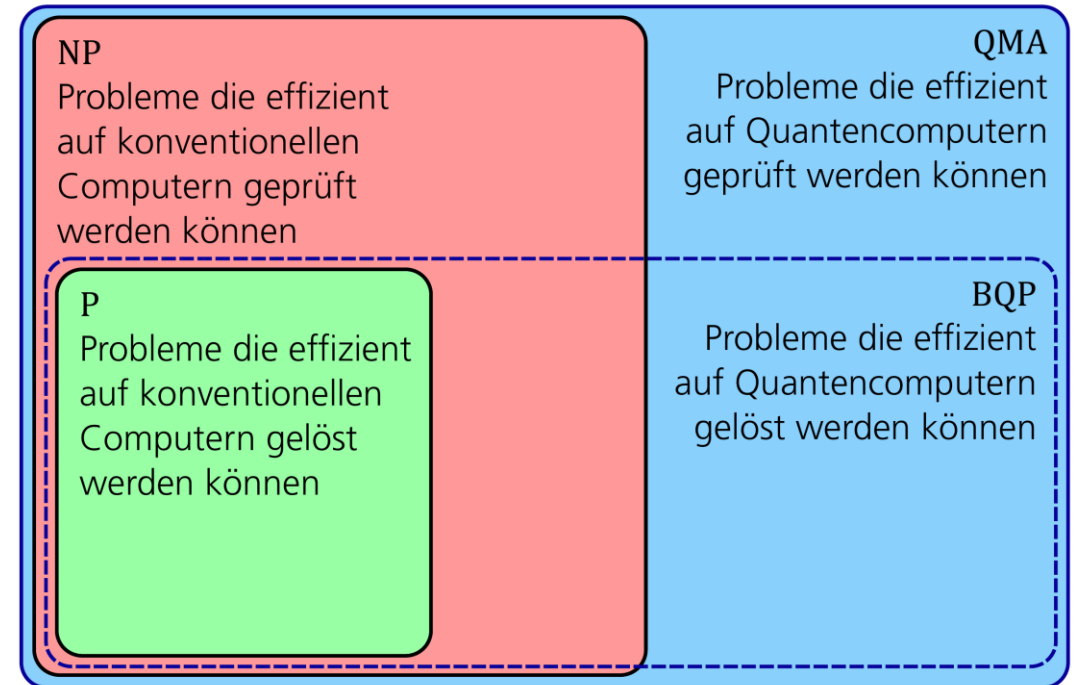


Was ist Effizienz? – Komplexitätsklassen

Komplexitätsklassen

Was ist Effizienz ?

- Wie effizient ein Problem lösbar ist, wird durch das Konzept der *Komplexität* ausgedrückt
- Komplexität wird durch den effizientesten theoretisch möglichen Algorithmus für die Lösung bestimmt
- *Effizient berechenbar* bedeutet, es existiert ein Algorithmus, der ein Problem in polynomieller Zeit $O(n^k)$ löst
- Jedoch ist nicht jedes Problem in $O(n^k)$ -Schritten lösbar
 - Komplexitätsklassen gruppieren Probleme bzgl. ihrer Komplexität
- Welche Klassen sind im Quantencomputing besonders relevant?



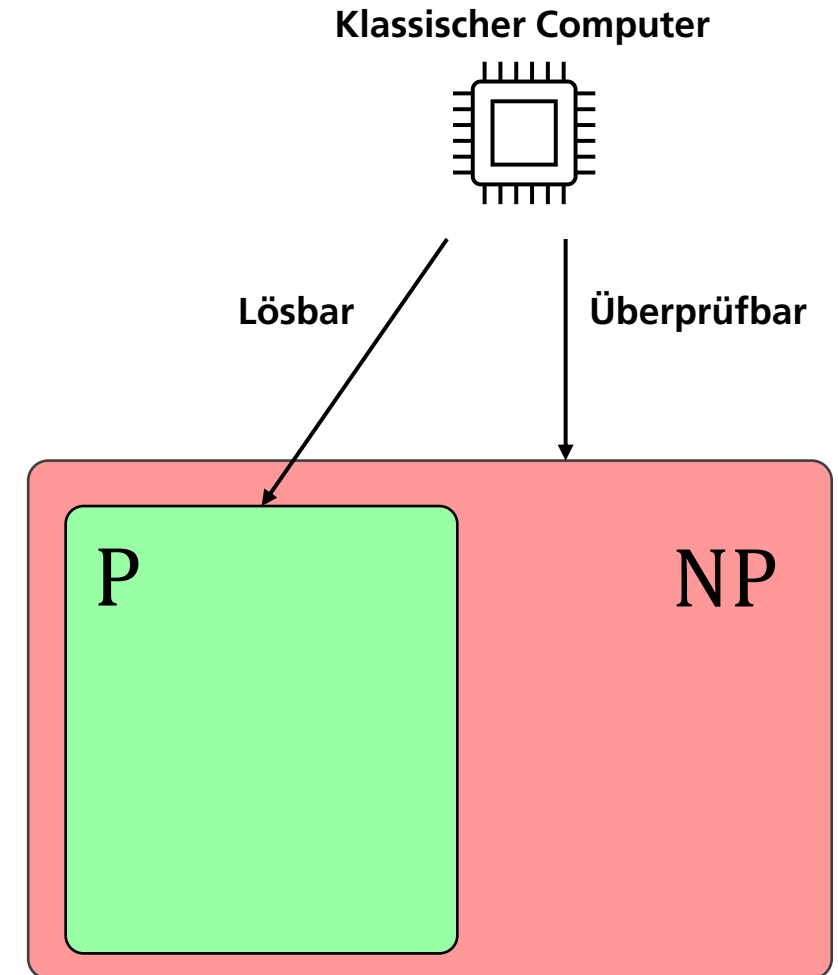
Komplexitätsklassen bilden eine hierarchische Struktur

Komplexitätsklassen

Die wichtigsten Komplexitätsklassen – P vs. NP

- Wichtigste Klasse der klassischen Informatik **P**; Menge aller Probleme, die in $O(n^k)$ auf einem herkömmlichen Rechner gelöst werden können
 - Bspw. Sortierungsprobleme, Lösen von linearen Gleichungssystemen, etc.
- Ebenfalls relevant **NP**: alle Probleme, die in polynomieller Laufzeit auf einem klassischen Rechner überprüft werden können
 - Berechnung der *exakten* Lösung hat exponentiellen Aufwand
 - Bspw. Traveling Salesman Problem (Logistik), Maximum Cut, etc.
- Allgemeiner: **NP**-Probleme sind in polynomieller Laufzeit auf einem *nicht-deterministischen* Computer lösbar

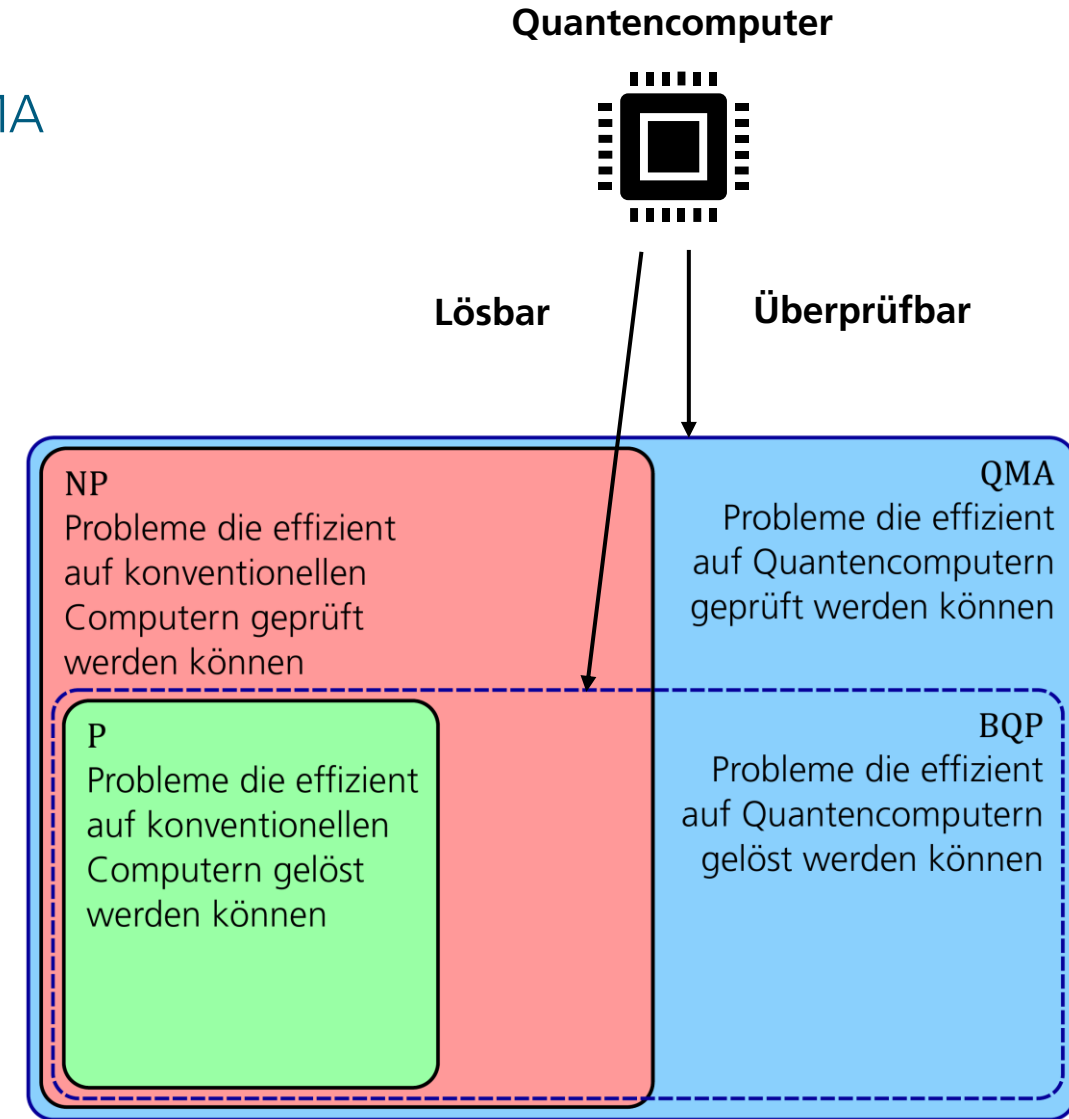
Für welche Probleme eignen sich Quantencomputer?



Komplexitätsklassen

Die wichtigsten Quanten-Komplexitätsklassen – BQP vs. QMA

- Bounded Error Quantum Polynomial (**BQP**) beschreibt in polynomieller Zeit auf einem Quantencomputer lösbare Probleme
 - Bspw. Shors Algorithmus, Grovers Algorithmus, etc.
- Es wird angenommen, dass Quantencomputer strikt mächtiger als klassische Computer sind $P \subset BQP$
- Analog zu P und NP: Quantum Merlin Arthur (**QMA**), beschreibt effizient überprüfbare Probleme auf einem QC
 - Bspw. überprüfen, ob ein Zustand ein Grundzustand eines Hamiltonians ist (physikalische Optimierungsprobleme)



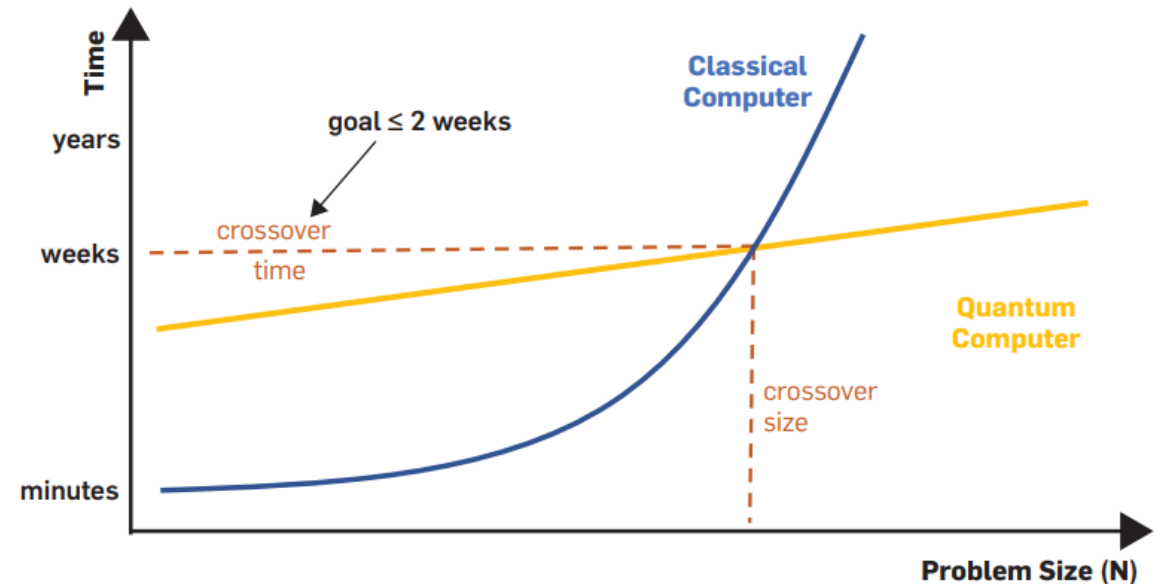
Ab welcher Problemgröße ist mit Quantum Advantage zu rechnen?

Komplexität und Laufzeit

Quantum Advantage: schneller richtige Ergebnisse für relevante Probleme

Aktuell

- Rechenoperationen/Qubitoperationen dauern einige 10–100 ns pro Operation – langsam ggü. CPUs
- Dauer des Einlesens der Daten in einen QC nicht vernachlässigbar, erhöht die effektive Laufzeit der Algorithmen
- QC-Systeme noch stark von Fehlern beeinträchtigt – wir leben „noch“ in der NISQ-Ära
- Konkurrenz mit klassischen Computern: erst bei großen Problemen ist ein Quantenvorteil zu erwarten



Quelle: TORSTEN HOEFLER, THOMAS HÄNER, AND MATTHIAS TROYER,

„Disentangling Hype from Practicality: On Realistically Achieving Quantum Advantage“

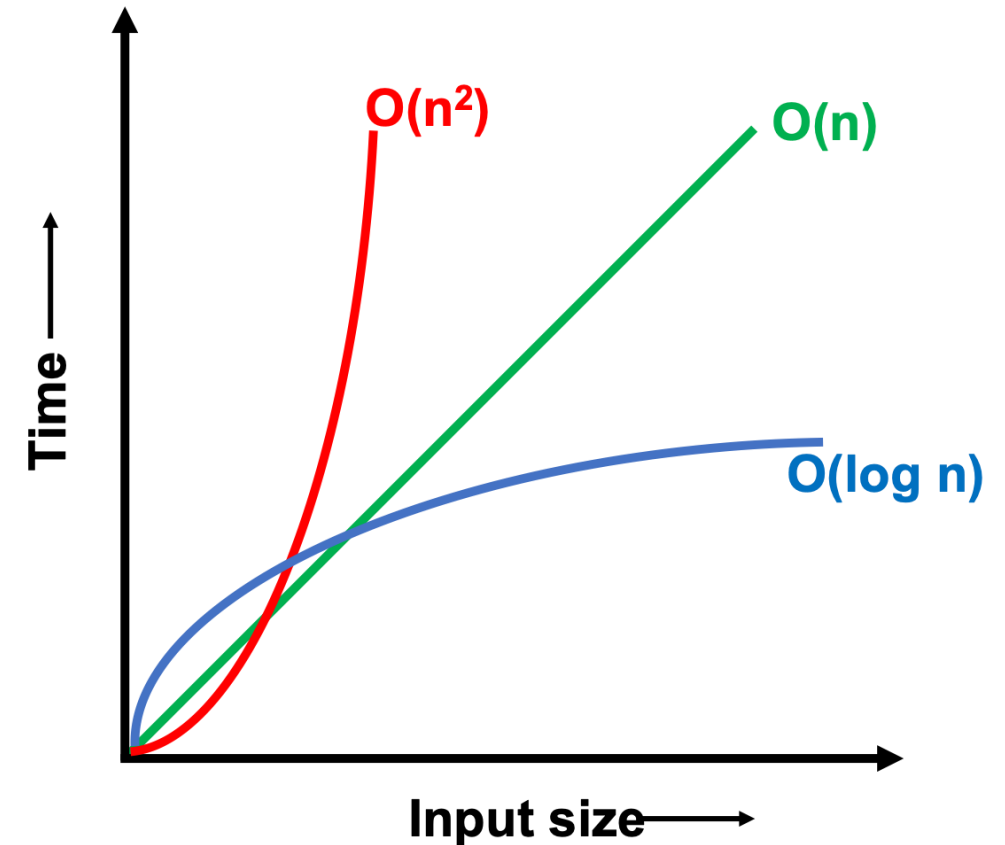
Wann wird sich Quantencomputing lohnen?

Komplexitätsvorteil

- Auch mit fehlerkorrigierten Qubits ist die geeignete Wahl der Probleme entscheidend für einen vermeintlichen Quantenvorteil
- Mindestens kubische, besser exponentielle Komplexität für Quantum Advantage
- Zeit für Ein- und Ausgabe nicht vernachlässigbar
 - Beispiel: HHL-Algorithmus für $n \times n$ LGS
$$O(n^3) \rightarrow O(\log n)$$
Aber: Input der Größe $O(n^2)$, Beschleunigung effektiv weniger als quadratisch!

Big Compute, nicht Big Data.

Hybride Algorithmen, die klassisches Supercomputing und Quantencomputing kombinieren

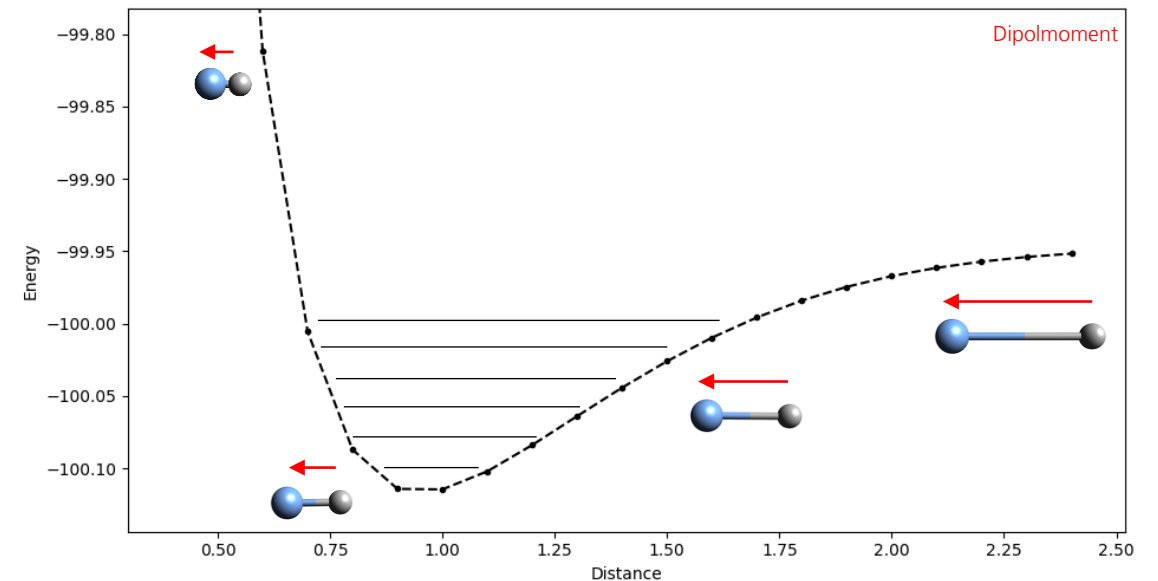


Anwendungsgebiete und Problemklassen

Beispiele

- Simulation chemischer Systeme im Rahmen der Quantenchemie
- Simulation quantenphysikalischer Systeme
- Anwendungsfälle, denen die Quantenphysik zugrunde liegt, haben ein „natürliches“ Mapping auf Quantencomputer
- Anwendungsfelder mit **theoretisch erwiesenem** Quantenvorteil

Ergebnisse einer quantenchemischen Rechnung am Beispiel Fluorwasserstoff



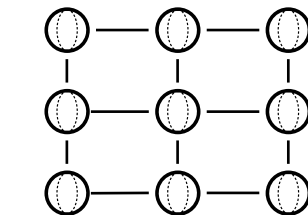
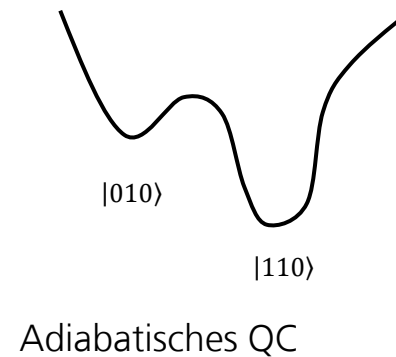
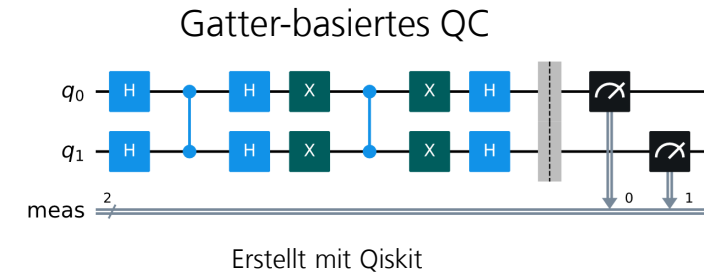
Wie programmiert man Quantencomputer?

Modelle des Quantencomputing

Gatter, Messungen oder analog?

Verschiedene mathematische Modelle des Quantencomputing

- Gatter-basiertes Quantencomputing
- Messungs-basiertes Quantencomputing
- Adiabatisches Quantencomputing
 - Quantenannealing
 - Analoges Quantencomputing



Messungs-basiertes QC

Quantengatter

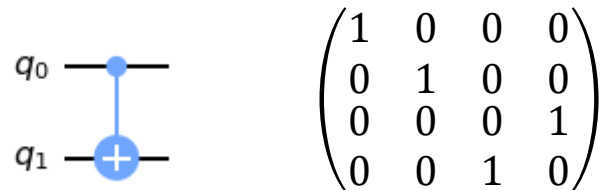
Quantencomputer werden über sogenannte Quantengatter (elementare Operationen) programmiert

- NOT-Gate



Input	Output
$ 0\rangle$	$ 1\rangle$
$ 1\rangle$	$ 0\rangle$

- CNOT-Gate



Input		Output	
q_0	q_1	q_0	q_1
$ 0\rangle$	$ 0\rangle$	$ 0\rangle$	$ 0\rangle$
$ 0\rangle$	$ 1\rangle$	$ 0\rangle$	$ 1\rangle$
$ 1\rangle$	$ 0\rangle$	$ 1\rangle$	$ 1\rangle$
$ 1\rangle$	$ 1\rangle$	$ 1\rangle$	$ 0\rangle$

- Hadamard-Gate



Input	Output
$ 0\rangle$	$\frac{1}{\sqrt{2}}(0\rangle + 1\rangle)$
$ 1\rangle$	$\frac{1}{\sqrt{2}}(0\rangle - 1\rangle)$

- Phase-Gate



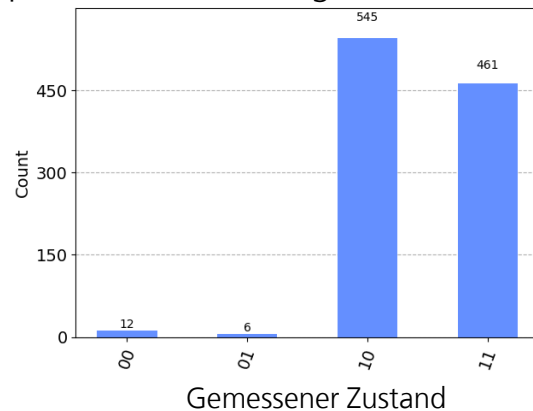
Input	Output
$ 0\rangle$	$ 0\rangle$
$ 1\rangle$	$e^{i\theta} 1\rangle$

Programmieren von Quantencomputern

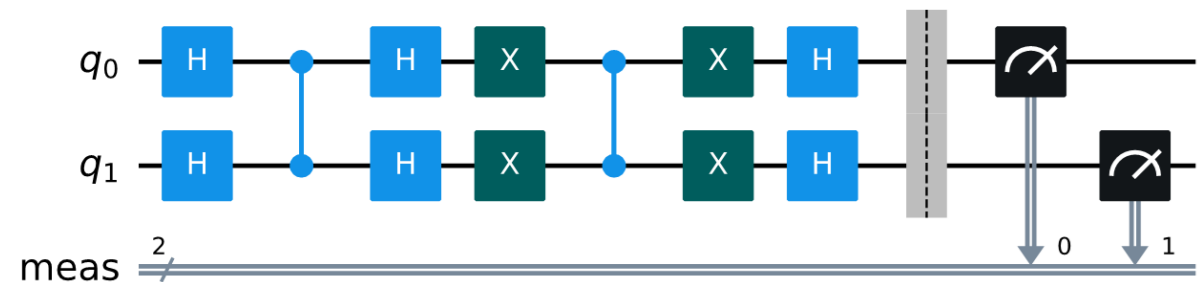
Was macht es so schwierig?

- Quantenmechanik ist wenig intuitiv
- Probleme sind sehr spezifisch
- Entwicklung grundlegender Bausteine noch nicht abgeschlossen
- Konkurrenz mit klassischen Algorithmen ist groß
- Kein Zugriff auf Zustandsvektor, nur „counts“

Beispiel für die Messung eines Zustandes:



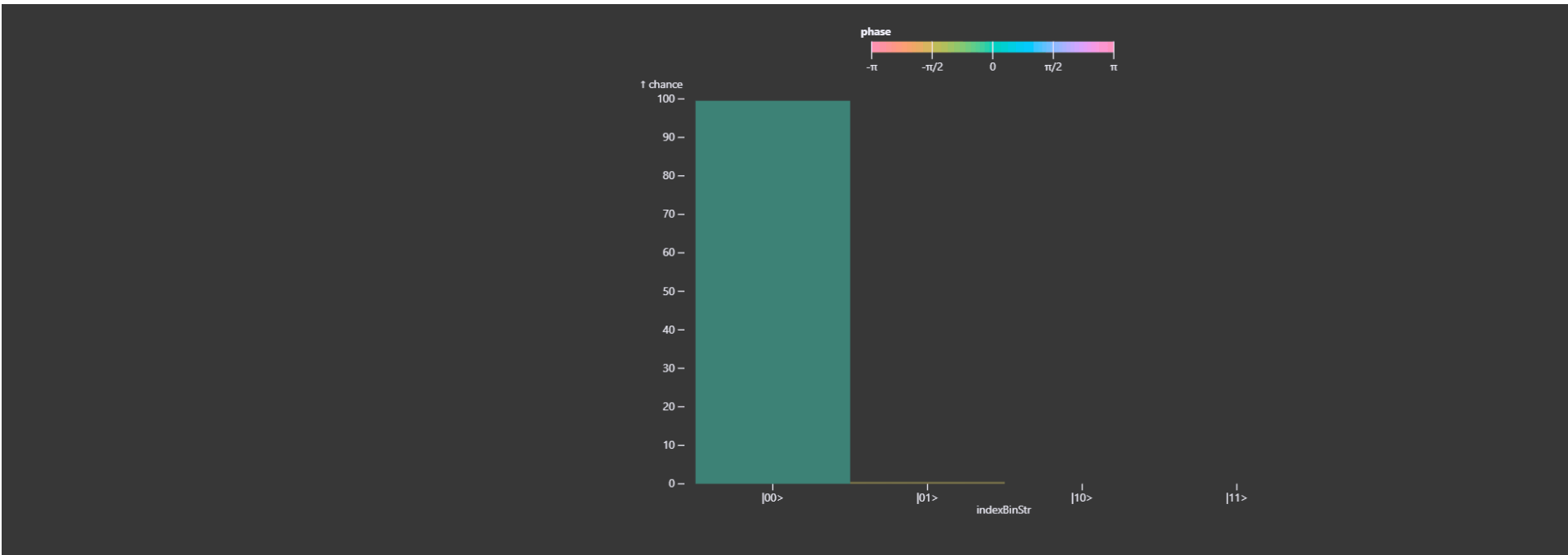
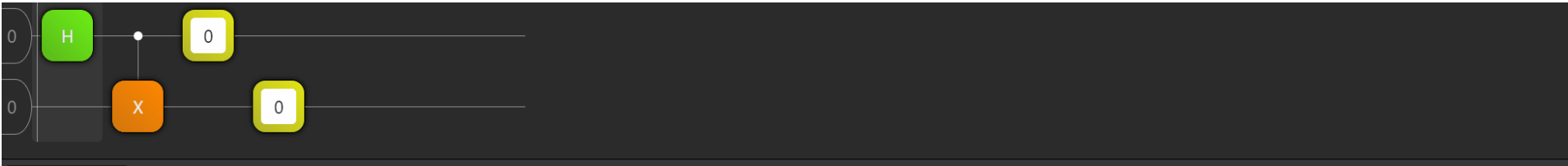
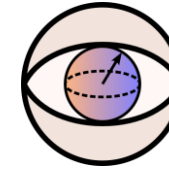
```
from qiskit import QuantumCircuit, Aer
qc = QuantumCircuit(2)
qc.h([0, 1])
qc.x([0, 1])
qc.cz(0, 1)
qc.x([0, 1])
qc.h([0, 1])
qc.measure_all()
sim = Aer.get_backend('aer_simulator')
sim.run(qc).result().get_counts()
```



Quantenschaltkreise

Visualisieren von QC-Algorithmen

Erstellt mit QCVIS
[QCVIS \(fh-igd-iet.github.io\)](https://fh-igd-iet.github.io/QCVIS)



Wie funktionieren Quantenalgorithmen?

Quantenalgorithmen

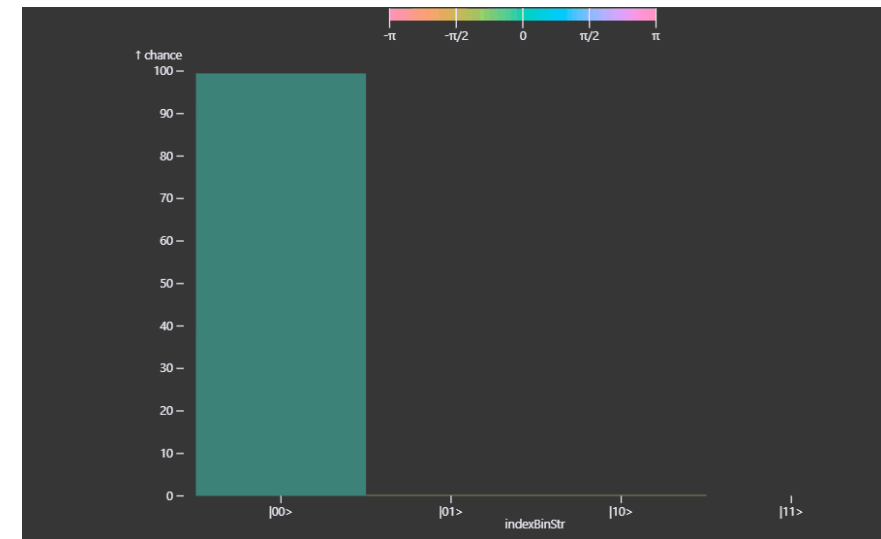
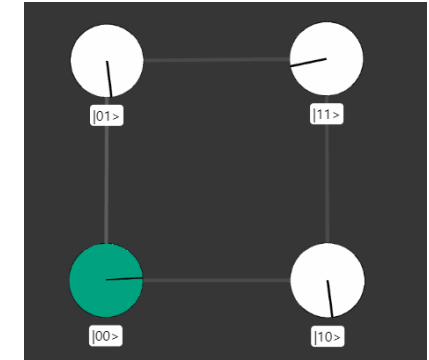
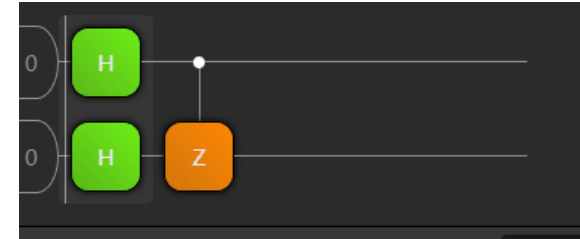
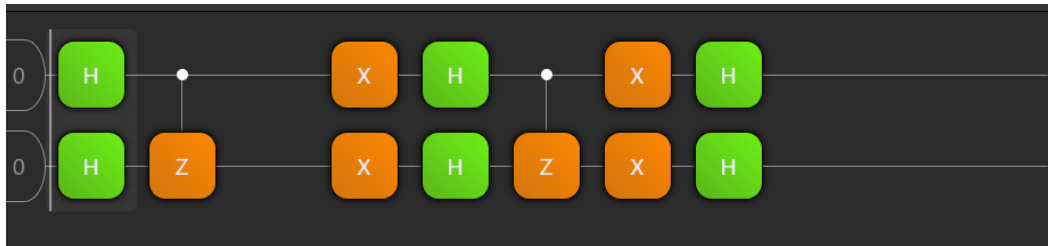
Grovers Algorithmus

- Algorithmus zum Durchsuchen unsortierter Daten
- Quadratischer Speed-Up gegenüber klassischen Algorithmen
- Schritt 1 (Initialisierung): Initialisiere Zustand als Superposition aller möglichen
- Schritt 2 (Orakel): Rufe das Orakel auf, welches den gesuchten Zustand durch einen Phasenflip markiert
- Schritt 3 (Amplifikation): Verstärkung der markierten Amplitude, Abschwächung aller anderen
- Findet Einsatz bei binären Optimierungsproblemen
 - Markiere die „besten“ Zustände (besser als gegebener Threshold)
 - Berechne Kostenfunktion der markierten Elemente
 - Minimum gibt neuen Threshold

Quantenalgorithmen

Grovers Algorithmus

- Schritt 1: Initialisiere Zustand als Superposition aller möglichen
- Schritt 2: Rufe das Orakel auf, welches den gesuchten Zustand durch einen Phasenflip markiert
- Schritt 3: Verstärkung der markierten Amplitude, Abschwächung aller anderen

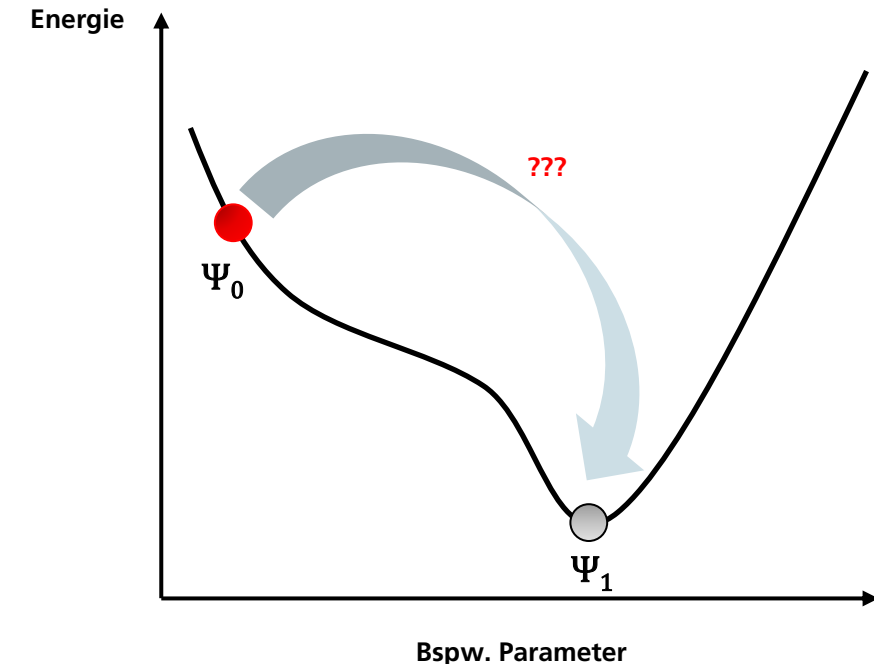


Simulation quantenphysikalischer Systeme

Mapping von Problemen auf einen QC

Typischerweise werden Probleme auf QC als Eigenwertgleichung des Hamiltonians H formuliert

- Eigenwertgleichung: $H\Psi = E\Psi$
 - Wellenfunktion Ψ beschreibt den Zustand des Quantencomputers
 - H ist hermitesche Matrix, dargestellt über Pauli-Matrizen X, Y und Z
- Problemstellung: finde Ψ , so dass E minimal ist
- Hamiltonian H kann ein chemisches oder physikalisches System oder auch ein Optimierungsproblem beschreiben
- Mit welchen Quantenalgorithmen kann die Wellenfunktion gefunden werden?



Simulation quantenchemischer Systeme

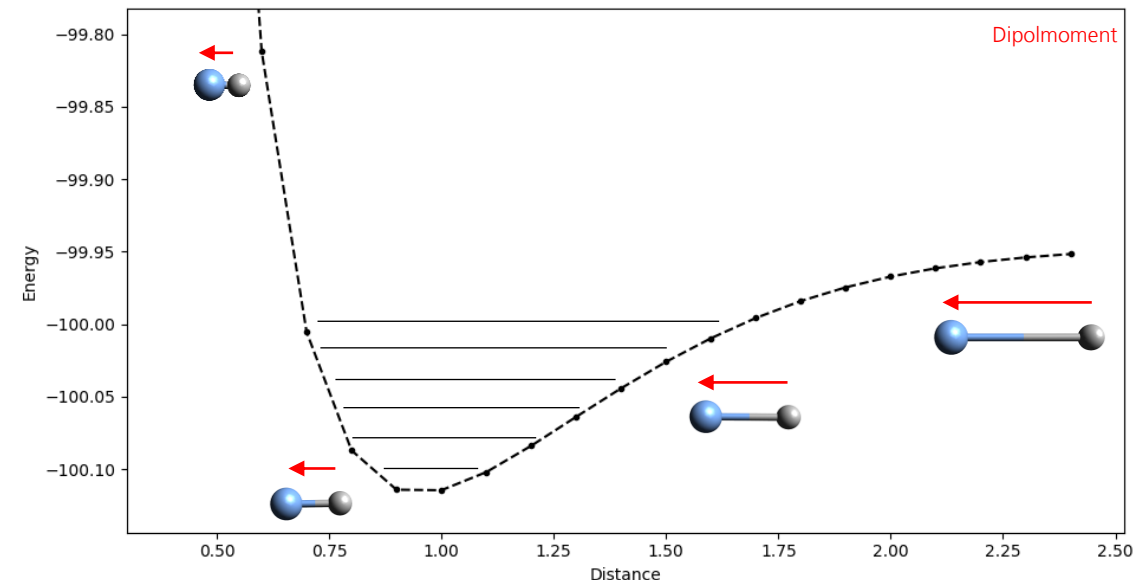
Überblick

- Allgemeines Ziel: Berechnung der molekularen Wellenfunktion
 - Lösung der zeitunabhängigen Schrödinger-Gleichung $H \Psi = E \Psi$
- Anwendung
 - Geometrieoptimierung von Molekülen
 - Voraussetzung von Reaktionsverläufen, thermodynamischen Eigenschaften, etc.
 - Entwicklung von Molekülen mit bestimmten Eigenschaften (bspw. Katalysatoren)

Ψ

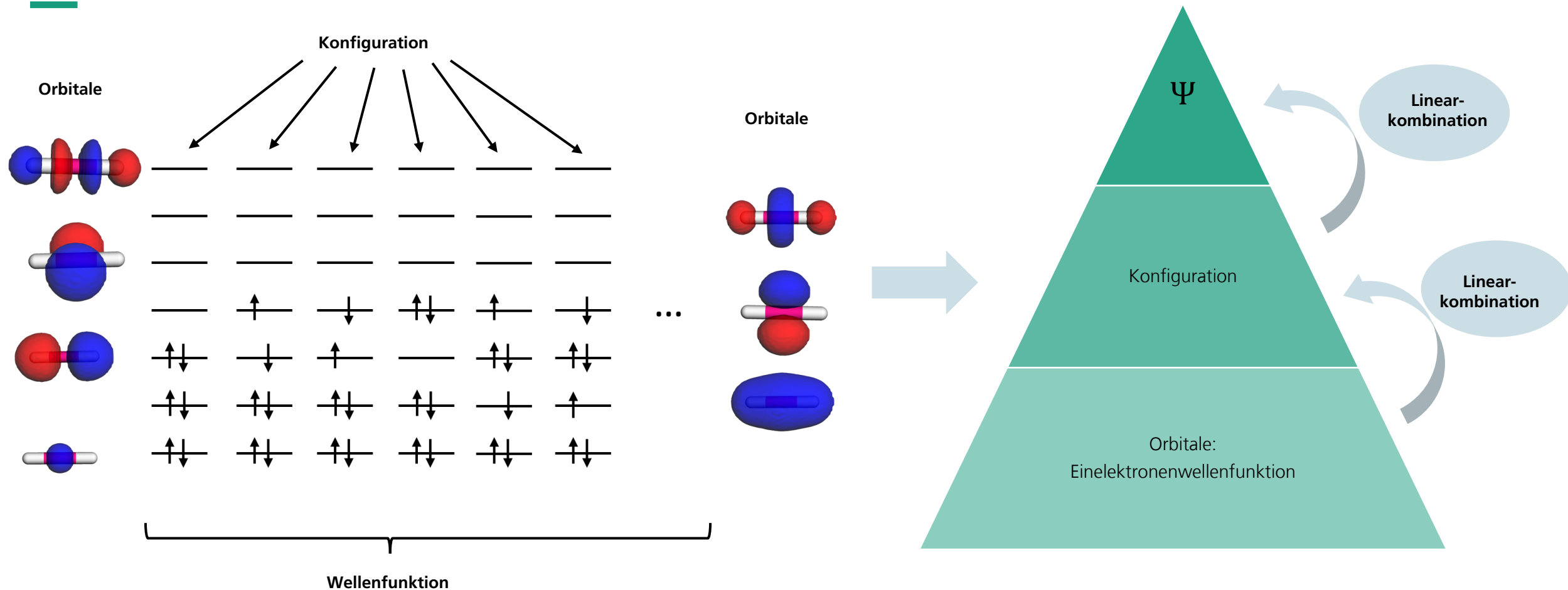
Die Wellenfunktion beschreibt
ein physikalisches System
vollständig

Ergebnisse einer quantenchemischen Rechnung am Beispiel Fluorwasserstoff



Simulation quantenchemischer Systeme

Elektronische Wellenfunktion



Mapping von Problemen auf einen QC

Optimierungsprobleme vs. Quantenphysikalische Simulation

Quantenphysikalische Simulation

- Mapping zwischen Hamiltonian des Quantensystems und Hamiltonian auf dem Quantencomputer
- Beispiel H_2 Molekül
 - $H = c_1 Z_1 + c_2 Z_2 + c_3 Z_1 \times Z_2 + c_4 X_1 \times X_2 + c_5 Y_1 \times Y_2$
- Grundzustand ist *keine* „klassische Lösung, sondern Superposition mehrerer Zustände

Optimierungsprobleme

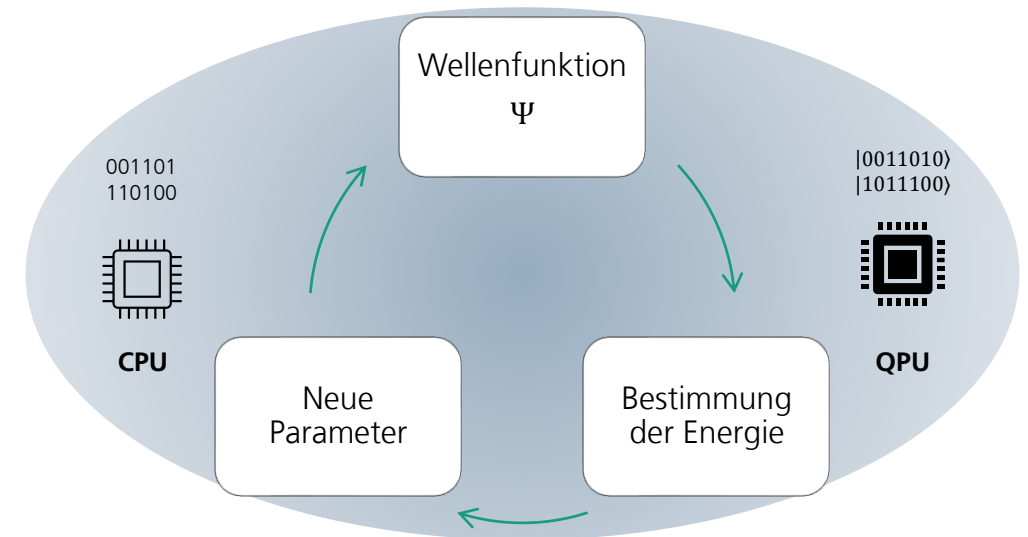
- Typischerweise binäre Optimierungsprobleme
- Mapping auf diagonalen Hamiltonian
- Beispiel QUBO:
 - $\min_{x \in X} x^T Q x$
 - $Q = \begin{pmatrix} 0 & 2 & 1 \\ & 0 & 0 \\ & & 0 \end{pmatrix} \rightarrow H = 0.5 Z_1 \times Z_2 + 0.25 Z_1 \times Z_3 + \dots$
- Algorithmus führt auf eine „klassische“ Lösung (keine Superposition!)

Quantenalgorithmen zur Simulation quantenphysikalischer Systeme

Quantenalgorithmus (1/3): Variational Quantum Eigensolver

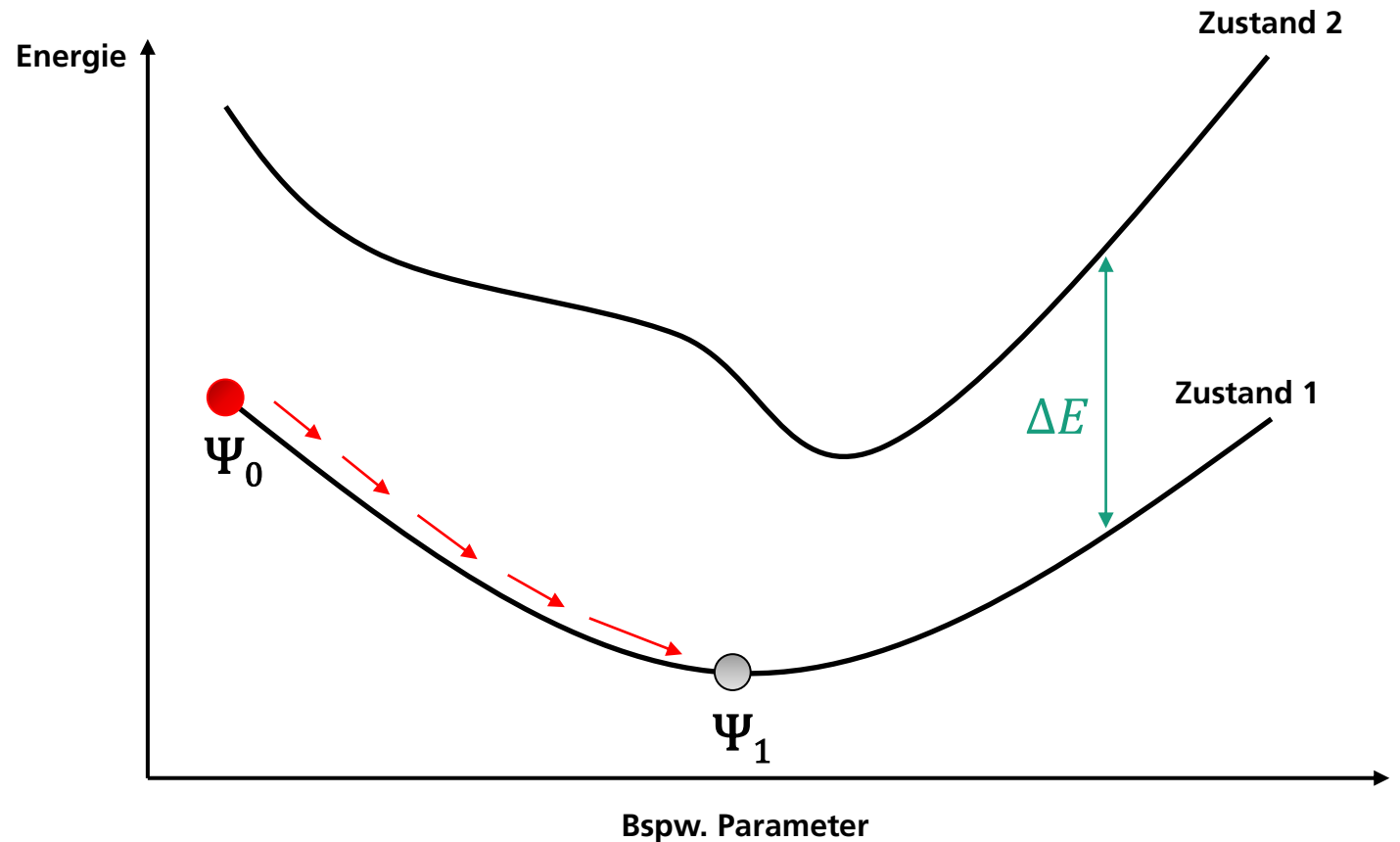
- Quantenalgorithmus mit kurzen Schaltkreisen
- VQE ist aufgrund der begrenzten QC-Ressourcen bzw. QC-Rechenkapazitäten derzeit beliebt
- Quantenvorteil bisher nicht nachgewiesen

- 
1. Parameter der Wellenfunktion werden von einem klassischen Computer berechnet
 2. Quantenschaltkreise, definiert durch die Parameter, werden an den Quantencomputer gesendet
 3. Wellenfunktion wird auf dem Quantencomputer aufgebaut und gemessen
 4. Übertragung des Resultats auf den klassischen Computer



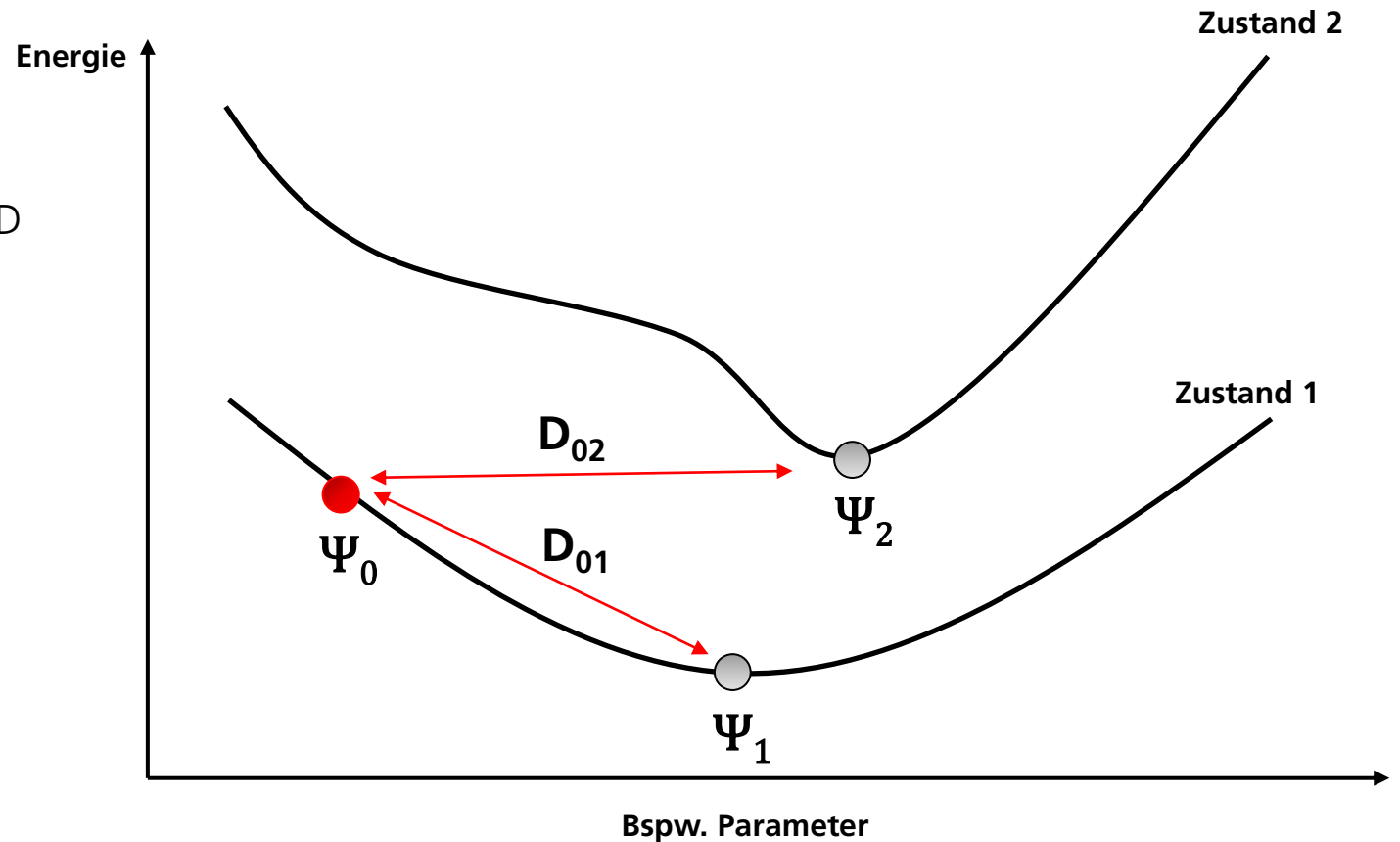
Quantenalgorithmus (2/3): Adiabatic State Evolution

- Entwicklung der Wellenfunktion in kleinen zeitlichen Schritten
- Schrittweite (rote Pfeile) proportional zu ΔE
- Quantenalgorithmus führt zu tiefen Schaltkreisen
- Eignet sich gut für analoges Quantencomputing



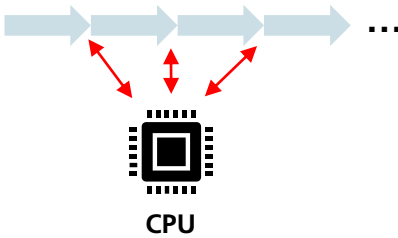
Quantenalgorithmus (3/3): Quantum Phase Estimation

- Anfangswellenfunktion Ψ_0 wird abhängig von D mit unterschiedlicher Wahrscheinlichkeit in die exakten Wellenfunktionen $\Psi_{1,2}$ überführt
- Ψ_0 muss hinreichend nah am gewünschten Endzustand liegen
- Quantenalgorithmus führt zu sehr tiefen Schaltkreisen



Quantenalgorithmen

Ein Vergleich

	Variational Quantum Eigensolver	Adiabatic State Evolution	Quantum Phase Estimation
hybrid	☑	☒	☒
Quantenschaltkreis			
Abhängigkeit vom Anfangszustand	weniger stark	weniger stark	stärker
Quantenvorteil (theoretisch)	fraglich	theoretisch bewiesen	theoretisch bewiesen

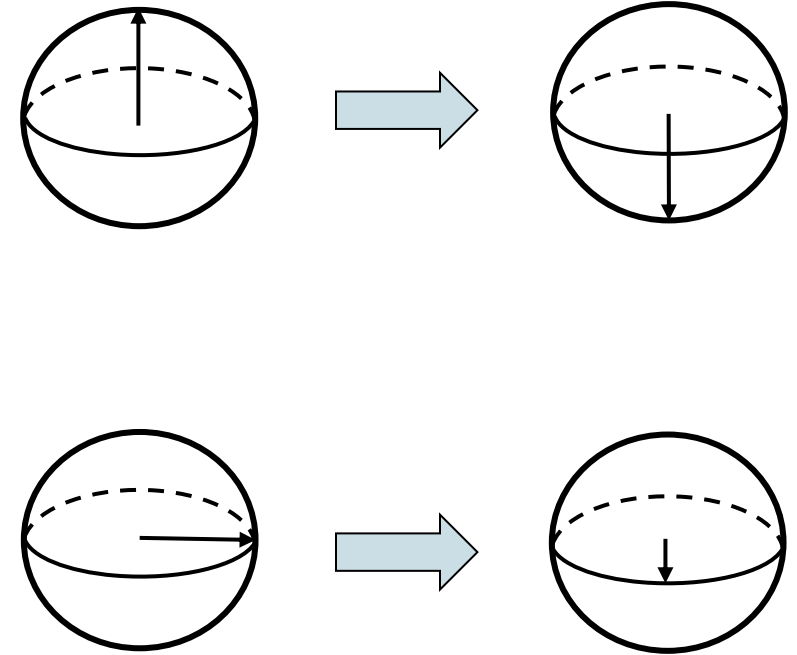
Der Weg zum „Quantum Advantage“

Was limitiert aktuelle Quantencomputer?

Typische Fehler von Quantencomputern

Beispielhafte Fehler eines Supraleitenden Quantencomputers

- Transverales Rauschen
 - Unbeabsichtigte Umwandlung zwischen $|0\rangle$ und $|1\rangle$
- Dephasierung
 - Auseinanderlaufen der individuellen Qubitphasen (Dekohärenz)
- Cross-Talk
 - Gate-Operation auf einem Qubit beeinflusst zweites Qubit
- Auslesefehler
 - Die physikalischen Zustände werden nicht korrekt den Messwerten 0 und 1 zugeordnet

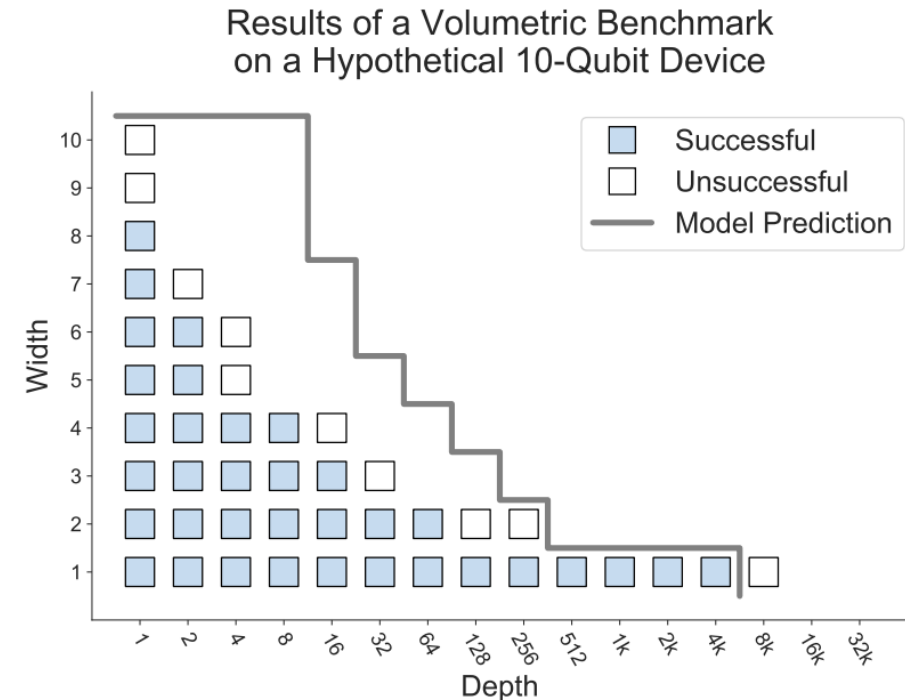


Benchmarking

Volumetrisches Benchmarking – Welcher Algorithmus funktioniert auf welchem QC wie gut?

Volumetrisches Benchmarking testet Quantencomputer auf Anzahl an Qubits und Länge der Schaltkreise.

- Teste verschiedene Algorithmen mit variierender Anzahl an Qubits und Schaltkreislänge
- Definiere, wann ein Algorithmus erfolgreich war
 - Beispielsweise durch Vergleich mit klassischer Simulation
- Trage Resultate in einen Plot „Tiefe vs. Qubits (Breite)“ auf
- Anwendungsorientierte Benchmarks
 - Lösen von QUBOs mit QAOA-Algorithmus
 - Digitale Simulation von Hamilton-Operatoren
 - VQE-Algorithmus



A volumetric framework for quantum computer benchmarks, Blume-Kohout, Robin and Young, Kevin C., Quantum (2020)

Der Weg zu „Quantum Advantage“

Wichtige Meilensteine

- Google 2019
 - Arute, F., Arya, K., Babbush, R. *et al.* Quantum supremacy using a programmable superconducting processor. *Nature* 574
 - Ausführen von Schaltkreisen mit bis zu 53 Qubits („Random Circuit Sampling“ mit Noise)
 - Neue Arbeiten zeigen, dass Simulation von Schaltkreisen mit viel Noise auch klassisch effizient lösbar ist
- IBM 2023
 - Kim, Y., Eddins, A., Anand, S. *et al.* Evidence for the utility of quantum computing before fault tolerance. *Nature* 618
Zeitentwicklung eines quantenmechanischen Systems mit 127 Qubits (mit Fehlermitigation)
 - Sehr spezielle Topologie / Modell
 - Oben genanntes Problem ist auch klassisch effizient lösbar
- QuEra 2023
 - Bluvstein, D., Evered, S.J., Geim, A.A. *et al.* Logical quantum processor based on reconfigurable atom arrays. *Nature* 626
 - Simulation von speziellen „IQP-Circuits“ mit bis zu 48 logischen Qubits (mit Fehlererkennung)
 - Neue Arbeit zeigt, diese Schaltkreise lassen sich auch klassisch effizient simulieren

Ausblick verschiedener Hersteller

QuEra

- Ab 2025: Realisierung von 30 logischen Qubits
- Ab 2026: 100 logische Qubits, tiefe Schaltkreise

[QuEra Computing Releases a Groundbreaking Roadmap for Advanced Error-Corrected Quantum Computers, Pioneering the Next Frontier in Quantum Innovation](#)

IonQ

- Ab 2025 64 logische Qubits (16:1 QEC)
- Ab 2028 1024 logische Qubits (32:1 QEC)

<https://ionq.com/posts/december-09-2020-scaling-quantum-computer-roadmap>

IBM

- Ab 2029 200 Qubits mit Fehlerkorrektur (Schaltkreise mit bis zu 100M Gates)

[IBM Debuts Next-Generation Quantum Processor & IBM Quantum System Two, Extends Roadmap to Advance Era of Quantum Utility - Dec 4, 2023](#)

Vielen Dank für Ihre
Aufmerksamkeit