Quantum Chemistry and Materials Algorithms

Dave Wecker
QuArC Chief Architect
Microsoft Research
Microsoft QuArC and StationQ
LIQUi|> Goals

- Simulation:
  - High enough level language to easily implement large quantum algorithms
  - Allow as large a simulation on classical computers as possible
  - Support abstraction and visualization to help the user
  - Implement as an extensible platform so users can tailor to their own requirements

- Compilation:
  - Multi-level analysis of circuits to allow many types of optimization
  - Circuit re-writing for specific needs (e.g., different gate sets, noise modeling)
  - Compilation into real target architectures
LIQUi|> is a modular software architecture designed to control quantum hardware. It enables easy programming, compilation, and simulation of quantum algorithms and circuits, and is independent of a specific quantum architecture. LIQUi|> contains an embedded, domain-specific language designed for programming quantum algorithms, with F# as the host language. It also allows the extraction of a circuit data structure that can be used for optimization, rendering, or translation. The circuit can also be exported to external hardware and software environments. Two different simulation environments are available to the user which allow a trade-off between number of qubits and class of operations. LIQUi|> has been implemented on a wide range of runtimes as back-ends with a single user front-end. We describe the significant components of the design architecture and how to express any given quantum algorithm.

http://arxiv.org/abs/1402.4467
Teleport: User Code

- Define a function to perform entanglement:

  ```
  let EPR (qs:Qubits) = H qs; CNOT qs
  ```

- The rest of the algorithm:

  ```
  let teleport (qs:Qubits) = 
    let qs' = qs.Tail 
    EPR qs'; CNOT qs; H qs 
    M qs'; BC X qs' 
    M qs ; BC Z !!(qs,0,2)
  ```
Full Teleport Circuit in a Steane7 Code

3 qubits go to 27
Advanced Noise Modeling

Noise(circ: Circuit, ket: Ket, models: NoiseModels)

```go
type NoiseModel = {
    gate: string, // Gate name (ending with "*" for wildcard match)
    maxQs: int, // Max qubits that gate uses
    time: float, // Floating duration of gate (convention Idle = 1.0)
    func: NoiseFunc, // Noise Model to execute
    gateEvents: NoiseEvents, // Stats for normal gates
    ecEvents: NoiseEvents, // Stats for EC gates
}

member n.DampProb // Get/Set damping probability on a qubit
```

Two Qubits H01N01

![Graph showing the performance of different teleportation schemes.](image)

Teleport, Steane7

- P_idle = P/10
- P_damp = P/10

3e-3
Shor’s algorithm: Full Circuit:
4 bits $\cong 8200$ gates

Largest we’ve done:
14 bits (factoring 8193)
14 Million Gates
30 days
Shor’s algorithm: Modular Adder

As defined in:
Circuit for Shor’s algorithm using 2n+3 qubits
– Stéphane Beauregard

let op (qs:Qubits) =
    CCAdd a cb

AddA’ N bs
QFT’ bs
CNOT [bMx;anc]
QFT bs
CAAddA N (anc :: bs)
CCAdd’ a cb

// Add a to Φ|b⟩
// Sub N from Φ|a + b⟩
// Inverse QFT of Φ|a + b – N⟩
// Save top bit in Ancilla
// QFT of a+b-N
// Add back N if negative
// Subtract a from Φ|a + b mod N⟩
QFT’ bs
X [bMx]
CNOT [bMx;anc]
X [bMx]
QFT bs
CCAdd a cb

// Inverse QFT
// Flip top bit
// Reset Ancilla to |0⟩
// Flip top bit back
// QFT back
// Finally get Φ|a + b mod N⟩
Shor’s algorithm results

Minutes to factor

- 19 years
- 1 day
- 2 years
- 3 hours
- 1 month
- 9 mins

bits to factor

4 5 6 7 8 9 10 11 12 13 14
A Little Motivation

- **Nitrogen Fixation:**
  - Making fertilizer uses a process from 1909 and uses lots of energy (400°C/200 atm)
  - Cost: 3-5% of the world’s natural gas production (1-2% of the world’s annual energy)
  - Design of a new catalyst would take ~100-200 qubits (inexpensive fertilizer)

- **Carbon Capture:**
  - Cost: Capturing at point sources will result in 21-90% increase in energy cost
  - Design a new catalyst to extract $CO_2$ from the air would take ~200-400 qubits

- **Design of new chemicals and materials:**
  - Today 1/3 of all supercomputing time is spent on chemistry and materials modeling
  - Designs that can never be done classically are solvable with a few hundred qubits
  - Pharmaceuticals, High temperature Super Conductors (energy, transportation...)
    - Example: gain back current 6.5% transmission loss in power lines
Can quantum chemistry be performed on a small quantum computer? 

As quantum computers with small but nontrivial numbers of qubits appear feasible, the question of possible applications of small quantum computers gains importance. One frequently mentioned application is Feynman's original proposal of simulating quantum mechanics on a classical computer. However, simulating quantum systems, and in particular the electronic structure of molecules and materials, is a widely anticipated application of quantum computers. However, recent studies\cite{WBCH13a,HWBT14a} have cast a shadow on this hope by revealing that the scaling of known upper bounds on the complexity of these algorithms used in the simulation of quantum chemistry on a quantum computer depends crucially on the details of the simulated molecule. We present several improvements to the standard Trotter-Suzuki decomposition\cite{Suzuki1} scheme and show that it can sometimes outperform existing schemes, but that this possibility can be decisive for determining the cost of a quantum simulation. Our analysis motivates several potential avenues for reducing the quantum resources required to simulate quantum chemistry on small quantum computers.

The simulation of molecules is a widely anticipated application of quantum computers. However, as quantum computers with small but nontrivial numbers of qubits appear feasible, the question of possible applications of small quantum computers gains importance. One frequently mentioned application is Feynman's original proposal of simulating quantum mechanics on a classical computer. However, simulating quantum systems, and in particular the electronic structure of molecules and materials, is a widely anticipated application of quantum computers. However, recent studies\cite{WBCH13a,HWBT14a} have cast a shadow on this hope by revealing that the scaling of known upper bounds on the complexity of these algorithms used in the simulation of quantum chemistry on a quantum computer depends crucially on the details of the simulated molecule. We present several improvements to the standard Trotter-Suzuki decomposition\cite{Suzuki1} scheme and show that it can sometimes outperform existing schemes, but that this possibility can be decisive for determining the cost of a quantum simulation. Our analysis motivates several potential avenues for reducing the quantum resources required to simulate quantum chemistry on small quantum computers.

Ferredoxin ($Fe_2S_2$) used in many metabolic reactions including energy transport in photosynthesis

- Intractable on a classical computer
- Assumed quantum scaling: ~24 billion years ($N^{11}$ scaling)
- First paper: ~850 thousand years to solve ($N^9$ scaling)
- Second paper: ~30 years to solve ($N^7$ scaling)
- Third paper: ~5 days to solve ($N^{5.5}$ scaling)
- Fourth paper: ~1 hour to solve ($N^3, Z^{2.5}$ scaling)
$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrst} h_{pqrst} a_p^\dagger a_q^\dagger a_r a_s$
Simulation of Materials (on-going work)

\[
H = - \sum_{\langle i,j \rangle} \sum_{\sigma} t_{ij}(c_{i,\sigma}^\dagger c_{j,\sigma} + c_{j,\sigma}^\dagger c_{i,\sigma}) + U \sum_i n_i^\uparrow n_i^\downarrow + \sum_i \epsilon_i \eta_i
\]

See: d-wave resonating valence bond states of fermionic atoms in optical lattices
Quantum Algorithms for Quantum Impurity Problems

- Mott Insulators
- Transition Metal Compounds
  - Cuprates (e.g., High Tc SC)
- Lanthanides and Actinides
- Kondo Physics (Low temperature Resistance) from Magnetic Impurities
  - Quantum Dots
Materials Modeling

- Solids have regular structure that can be modeled as lattices.
- The Hubbard model only implements $H_{pp}$ and $H_{pqqp}$ terms.
- This doesn’t cover many of the materials we’re interested in.
- One can choose a single site in the lattice to model.
- The effect of the rest of the lattice can be modeled in terms of its effect on this site.

$$H_{hub} = U \sum_i n_{i\uparrow} n_{i\downarrow} - t \sum_{<i,j>,\sigma} c_{i\sigma}^\dagger c_{j\sigma}$$

$$H_{imp} = U n_{\uparrow} n_{\downarrow} - \sum_{k,\sigma} (t_k c_{k\sigma}^\dagger a_{k,\sigma}^{bath} + h. c.) + H_{bath}$$
Anderson Impurity Model

- Choose a single place in the lattice to model (the impurity). This may contain a collection of local sites.
- The impurity is typically a full two-body model.
- The effect of the rest of the lattice can be modeled in terms of its effect on the impurity (the bath) via a Dynamic Green’s function $G(\omega)$.
- The bath may have many sites and interconnections.

$$H = \sum_{ij} t_{ij} a_i^\dagger a_j + \frac{1}{2} \sum_{ijkl} w_{ijkl} a_i^\dagger a_j^\dagger a_k a_l + \sum_{ip} V_{ip} (a_i^\dagger a_p + a_p^\dagger a_i) + \sum_{pq} \epsilon_p a_p^\dagger a_q$$
We can posit an initial model for a material.

Measure quantum simulations at many sites and frequencies deriving a dynamical Green’s function.

Use feedback to update model.

Repeat until converged.

The resulting model is defined classically and may be used to efficiently investigate many questions about the material.

\[ G_{\text{solver}}(\omega) \rightarrow \Sigma(\omega) \rightarrow G(k, \omega) \rightarrow G_n(\omega) \rightarrow \Delta_n(\omega) \]

Dynamical Mean Field Theory (DMFT)

\[ G_{\text{solver}}(\omega) = \langle c_i^\dagger(\omega) c_j(-\omega) \rangle \]

http://arxiv.org/abs/1012.3609
Cubic Hydrogen ($H_2$ crystal structure)

- Simulate at 2.5 Å with a bath of 9 orbitals
- Density of States plot for different frequencies. Red curve is the Hartree-Fock solution (used as initial guess). DMFT Converges after 7 iterations
- 5-7 total orbitals for a single site is state-of-the-art.
- Example from Gull et al, the diagram shows a 3 shell degenerate solution (need 5 for D and 7 for F)
- A Quantum Computer could do a 4x larger impurity or 4x more orbitals than state-of-the-art with 200 qubits

\[ G_{\text{imp}}(\omega)^{-1} = (\omega + \mu + i0_{\pm})S - h_{\text{imp}} - \Sigma(\omega) - \Delta(\omega) \]

http://arxiv.org/abs/1012.4474
SoLi|⟩ - Son of LIQUi|⟩

300 Kelvin - Room

CMOS
CPU
Memory

77K-Nitrogen

Superconducting
CPU
Memory

4K-Helium

Quantum
Control
Qubits

0.02K-He$_3$/He$_4$
Compilation for Hardware Execution

Algorithm

- Mixed Classical and Quantum
- F# Syntax, Domain Specific Language

```fsharp
let pkg = Package(<@ QFT @>)
let QFT (qs:Qubits) = ...
```

```fsharp
let sim = CircSimModel()
let impl = sim.Target(pkg)
let QFT (qs:Qubits) = ...
```
Thank You

Dave Wecker
QuArC Chief Architect
Microsoft Corporation

Referenced papers may be found at:
http://arxiv.org (Search: wecker_d)