JUSTC

Article

http://justc.ustc.edu.cn

Received: August 29, 2022; Accepted: November 03, 2022

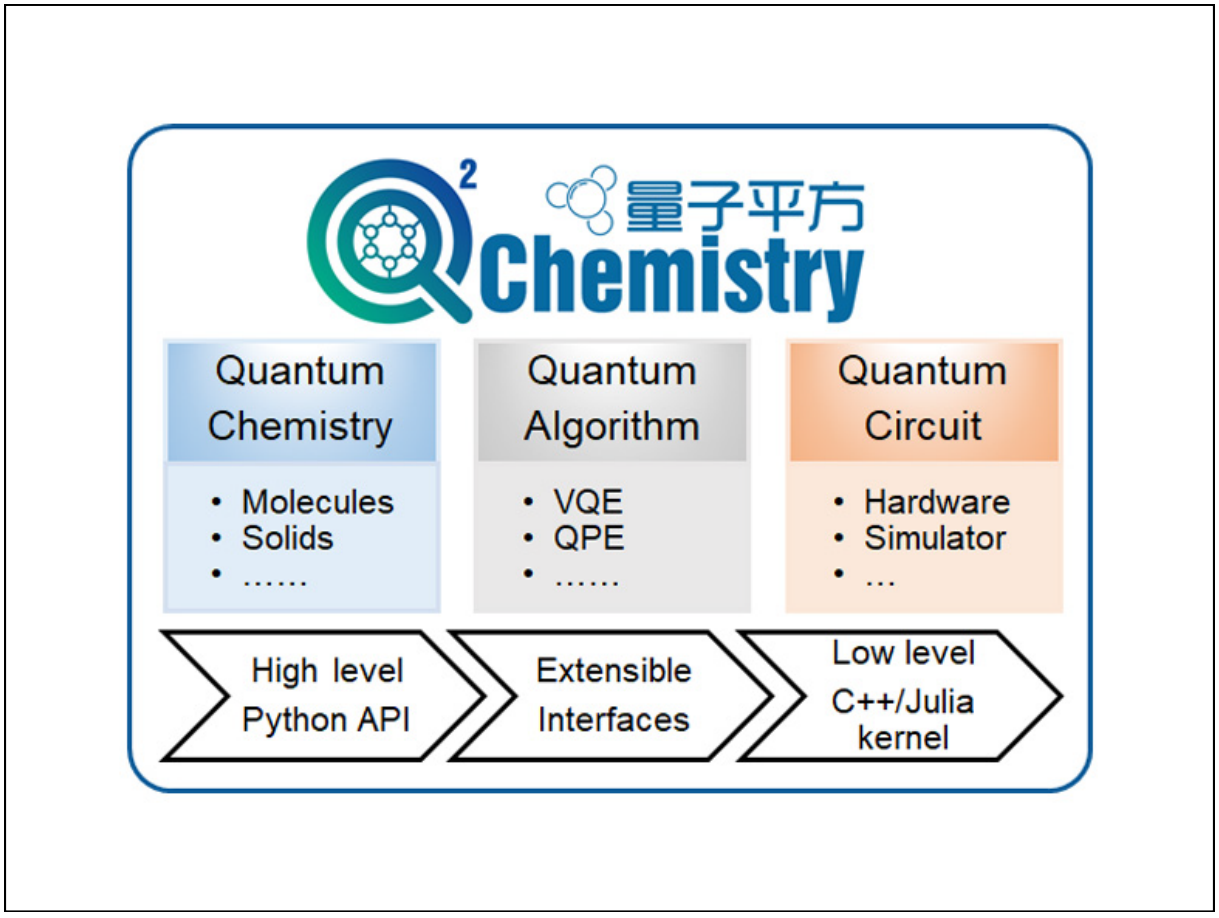
Q²Chemistry: A quantum computation platform for quantum chemistry

Yi Fan, Jie Liu, Xiongzhi Zeng, Zhiqian Xu, Honghui Shang, Zhenyu Li ✉, and Jinlong Yang

Hefei National Research Center for Physical Sciences at the Microscale, University of Science and Technology of China, Hefei 230026, China

✉Correspondence: Zhenyu Li, E-mail: zyli@ustc.edu.cn

Graphical abstract



The main framework of our newly developed quantum computation software package.

Public summary

■ We developed a quantum computation platform for quantum chemistry applications.

■ A modular structure was implemented with a mixed-language programming model for easy extension and high performance.

■ Excellent performance is achieved as demonstrated by simulating medium-scale quantum circuits up to 72 qubits.

Citation: Fan Y, Liu J, Zeng X Z, et al. Q²Chemistry: A quantum computation platform for quantum chemistry. *JUSTC*, **2022**, 52(12): 2. DOI: 10.52396/JUSTC-2022-0118

JUSTC

Article

http://justc.ustc.edu.cn

Received: August 29, 2022; Accepted: November 03, 2022

Q²Chemistry: A quantum computation platform for quantum chemistry

Yi Fan, Jie Liu, Xiongzhi Zeng, Zhiqian Xu, Honghui Shang, Zhenyu Li ✉, and Jinlong Yang

Hefei National Research Center for Physical Sciences at the Microscale, University of Science and Technology of China, Hefei 230026, China

✉Correspondence: Zhenyu Li, E-mail: zyli@ustc.edu.cn

Cite This: *JUSTC*, **2022**, 52(12): 2 (11pp)

Read Online

SI Supporting Information

Abstract: Quantum computers provide new opportunities for quantum chemistry. In this article,we present a versatile, extensible, and efficient software package, named Q²Chemistry, for developing quantum algorithms and quantum inspired classical algorithms in the field of quantum chemistry. In Q²Chemistry, the wave function and Hamiltonian can be conveniently mapped into the qubit space, then quantum circuits can be generated corresponding to a specific quantum algorithm already implemented in the package or newly developed by the users. The generated circuits can be dispatched to either a physical quantum computer, if available, or to the internal virtual quantum computer realized by simulating quantum circuits on classical computers. As demonstrated by our benchmark simulations, Q²Chemistry achieves excellent performance in simulating medium scale quantum circuits using the matrix product state algorithm. Applications of Q²Chemistry to simulate molecules and periodic systems are given with performance analysis.

Keywords: quantum algorithm; quantum circuit; electronic structure; matrix product state

CLC number: O641.12⁺1 **Document code:** A

1 Introduction

As the application of accurate classical methods is severely limited by the fast growing computational cost, quantum computation provides a promising pathway to solve quantum chemistry problems[1−3]. By encoding wave functions into the Hilbert space of qubits, the Schrödinger equation for molecular systems can be solved on a quantum computer. In recent years, various quantum algorithms, such as quantum phase estimation (QPE) and variational quantum eigensolver (VQE)[2−19], are developed for quantum chemistry. There are some important issues that should be studied for these algorithms. For example, the basis set used to construct the electronic Hamiltonian and the fermion-to-qubit encoding technique may lead to significant gate complexity and measurement overhead in QPE[20]. The optimization of a typical wave function ansatzes using VQE can suffer from "barren plateaus"[21,22] or local minimum traps[23]. A quantum computation platform that can provide extensive functionalities and step-by-step cross verification is therefore desirable for designing novel quantum algorithms for chemistry applications.

Noisy intermediate quantum (NISQ) devices have been used to demonstrate the possibility of studying the ground and excited states of molecular systems using currently available algorithms[10−15,24,25]. However, such experimental demonstrations are limited to tiny molecules with an artificially small basis set. This is because NISQ experiments are limited by the available quantum resources and the error associated with each quantum gate. Therefore, low gate fidelity, short coherence time, and insufficient qubit resource prohibit a systematic study of quantum chemistry oriented quantum algorithms on NISQ devices. The largest quantum computation experiment for chemistry to date uses 16 qubits with 160 two-qubit gates[25], while a simple VQE circuit of the commonly used unitary coupled-cluster (UCC)[26,27] ansatz for a small molecule has millions of CNOT gates (Table 1), which is far beyond the capability of NISQ devices. Therefore, it is important to have the capability to simulate the quantum circuit on a classical computer at this stage. Even beyond the NISQ era, such a capability can help us to develop quantum-inspired classical algorithms.

Several quantum computation packages have been reported, for example, C++ based ProjectQ[28] and Qiskit[29], GPU-enabled Qulacs[30], and the differentiable simulator Yao[31] implemented in Julia. Most of these codes are developed as standalone quantum circuit simulators or compilers, which are not dedicated to quantum chemistry applications. At the same time, the performance of simulating the quantum circuit is not very satisfactory in these packages. Most of these packages only implement the brute-force simulating method, which leads to an exponential computational cost. Tensor-based methods have been implemented in packages such as Qiskit and PennyLane[32], however, without an efficient distributed parallelization algorithm. As a reference, the largest simulated quantum circuit to date contains 28 qubits, which is used to study the ground state of ethylene molecules using VQE[33].

Based on the above considerations, we develop a versatile and extensible quantum computation platform for quantum chemistry, named Q²Chemistry (pronounced as "Q squared chemistry") to highlight the two quantum dimensions (the

DOI: 10.52396/JUSTC-2022-0118

JUSTC, **2022**, 52(12): 2

Table 1. Computational resources required to perform VQE simulations for a number of molecules using the unitary coupled-cluster ansatz truncated up to double excitations (UCCSD) and the minimum basis set STO-3G. UCCGSD means that generalized excitation operators (not distinguishing occupied and virtual orbitals) are used. The way to count CNOT gates is described in the supporting information.

Molecule	Qubits	Parameters		CNOT gates	
		UCCSD	UCCGSD	UCCSD	UCCGSD
H_2	4	2	5	64	144
LiH	8	14	72	1632	10720
H_2O	14	90	630	26272	2.1×10^5
NH_3	16	135	1064	46480	4.4×10^5
CH_4	18	230	1692	95200	8.2×10^5
C_2H_4	28	1224	9737	8.6×10^5	7.9×10^6
C_3H_6	42	5994	48930	6.6×10^6	6.3×10^7

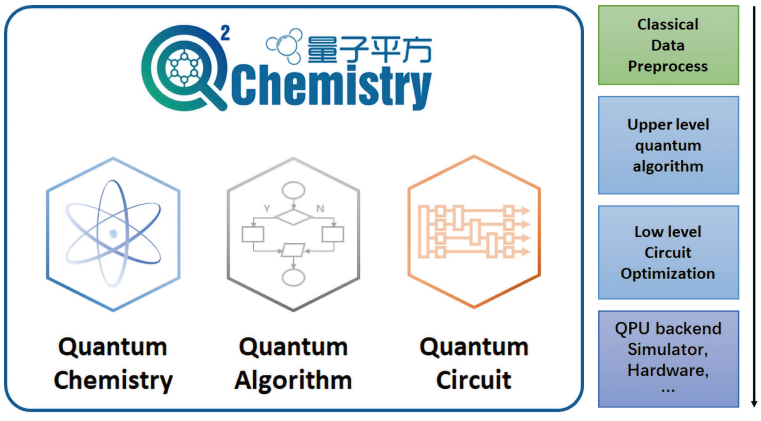
systems to be studied and the tools used to study them). Q²Chemistry adopt a modular design and a mixed-language programming model to achieve versatility together with high performance, where Python is used as its application program interface (API) while C++ and Julia[34] are used for computationally intensive tasks such as quantum circuit simulators. Q²Chemistry provides interfaces to quantum chemistry packages such as PySCF[35] to generate the required parameters for the qubit Hamiltonian. The quantum algorithms for solving the eigenstates of the electronic Hamiltonian are programmable at a high level of abstraction by adopting internal templates and/or defining custom circuit generation procedures. Some popular VQE and post-VQE algorithms have already been implemented for ground and excited states of molecular and periodic systems. Q²Chemistry provides native modules to run the circuit generated by a specific quantum algorithm, either via various high performance classical circuit simulation algorithms for simulating the circuit on a classical computer or via extensible interfaces reserved for the upcoming actual quantum processors. An efficient matrix product state (MPS)[36,37] based circuit simulation engine is implemented to perform quantum simulations for systems with 50–100 qubits.

The remainder of this article is organized as follows. After introducing the framework of Q²Chemistry in Section 2, we present functionalities for handling ab-initio chemistry quantities in Section 3. In Section 4 we explain the implementation of the circuit simulation on a classical computer or running on the upcoming quantum hardware. Section 5 provides a general introduction to the natively implemented VQE-based algorithms. Finally, in Section 6, we provide some examples to demonstrate the power of Q²Chemistry in chemistry applications, and provide the road map for further extensions including the support for more quantum algorithms, classical circuit simulation backends, and circuit optimization algorithms.

2 The framework of Q²Chemistry

As shown in Fig. 1a, Q²Chemistry contains three modules. (i) `q2chem.qchem`: the quantum chemistry module that defines quantum chemistry problems in the qubit space typically with the help of external classical quantum chemistry packages; (ii) `q2chem.qcirc`: the quantum circuit module that provides the quantum circuit related functionalities, including circuit construction, visualization, optimization, and execution on a virtual or real quantum computer; (iii) `q2chem.qalgo`: the quantum algorithm module, which includes native quantum algorithms and provides tools to implement new algorithms to solve chemistry problems.

High-level modules in Q²Chemistry are implemented using the scripting language Python and provide base implementations for classes in submodules. Benefiting from the modular design and Python's extensibility, extending existing submodules are easily achieved by constructing derived



(a) (b)

Fig. 1. (a) The framework of Q²Chemistry. (b) A typical workflow of solving a chemical problem using a quantum algorithm.

DOI: 10.52396/JUSTC-2022-0118
JUSTC, **2022**, 52(12): 2

classes and implementing only a small number of virtual functions, which requires no modifications to higher level modules. Core functions in the backends are implemented using programming languages including C++ and Julia. These functions are integrated into the Python interfaces using just-in-time (JIT) technology, which compiles the code at runtime to provide machine-specific optimizations and deliver outstanding performance. Most of the low-level data structures accessible from Python are stored using NumPy's[38] `ndarray`. Therefore, auxiliary operations, such as exact diagonalization for the qubit Hamiltonian, can be realized by using NumPy-compatible packages such as SciPy[39] or PyTorch[40].

Using such a modular framework, a general workflow to solve a quantum chemistry problem on a quantum computer is briefly illustrated in Fig. 1b. (i) Collect classical data such as electron integrals and mean-field orbital coefficients to generate Hamiltonian and an initial quantum state. (ii) Choose or develop a suitable quantum algorithm and generate corresponding quantum circuits, which may depend on the Hamiltonian, such as in the QPE algorithm, or be system independent, such as in some VQE algorithms. (iii) Perform lower-level circuit optimizations, including eliminating redundant quantum gates to reduce circuit depth or reconstructing the circuit to fit a specific quantum processor. (iv) Execute the quantum circuits and perform some measurements to extract necessary information, using either a virtual or real quantum computer.

Currently, Q^{2}Chemistry can be routinely used for performing VQE simulations on a classical computer, which is powered by Hamiltonian generation for molecular and periodic systems, hardware-efficient and UCC-based ansatzes for parametric circuit construction, and a scalable noise-free tensor network backend for classical simulations of large quantum circuits.

3 The quantum chemistry module

The `q2chem.qchem` module mainly handles the system Hamiltonian and wave function mapping. The second-quantized Hamiltonian is constructed from classically calculated quantities such as molecular orbital coefficients. Wave function mapping determines how the orbitals of the simulated wave function are mapped onto qubits therefore influences the measurement strategy.

3.1 Hamiltonian

For a Hamiltonian that is expressed as the linear combination of Pauli strings

$$\hat{H} = \sum_i c_i \hat{P}_i \quad , \tag{1}$$

where $\hat{P}_i$ is the product of Pauli operators $\{\hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z\}^{\otimes}$, and the expectation value E can be obtained through quantum measurement techniques such as the Hadamard test as

$$\begin{aligned} E =& \langle\Psi|\hat{H}|\Psi\rangle = \\ & \langle\Psi|\sum_i c_i \hat{P}_i|\Psi\rangle = \\ & \sum_i c_i \langle\Psi|\hat{P}_i|\Psi\rangle. \end{aligned} \tag{2}$$

Given the Hartree-Fock orbitals, the second-quantized electronic Hamiltonian can be written as

$$\hat{H} = \sum_{p,q} h_{pq} \hat{T}^p_q + \sum_{\substack{p,q \\ r,s}} g^{pq}_{rs} \hat{T}^{pq}_{rs}, \tag{3}$$

with

$$\begin{aligned} \hat{T}^p_q &= a^\dagger_p a_q, \\ \hat{T}^{pq}_{rs} &= a^\dagger_p a^\dagger_q a_r a_s. \end{aligned} \tag{4}$$

h_{pq} and g^{pq}_{rs} are one- and two-electron integrals. To convert the second-quantized Hamiltonian Eq. (3) into the qubit form as given in Eq. (1), a fermion-to-qubit mapping such as the Jordan-Wigner or Bravyi-Kitaev mapping is required[41–43].

In Q^{2}Chemistry, an interface with the PySCF package is provided to calculate one- and two-electron integrals for molecular and periodic systems[44,45]. Orbital optimization is also supported by linking to the `pyscf.lo` module or using a custom cost function. A unitary matrix is then obtained to transform the integrals as follows:

$$\begin{aligned} \tilde{h}_{\bar{p}\bar{q}} &= \sum_{p,q} U^*_{p\bar{p}} h_{pq} U_{q\bar{q}}, \\ \tilde{g}^{\bar{p}\bar{q}}_{\bar{r}\bar{s}} &= \sum_{\substack{p,q \\ r,s}} g^{pq}_{rs} U^*_{p\bar{p}} U_{q\bar{q}} U^*_{r\bar{r}} U_{s\bar{s}}, \end{aligned} \tag{5}$$

where U^* is the elementwise complex conjugate of U. This step is carried out efficiently by calling the optimized tensor contraction package `opt_einsum`[46]. For fermion-to-qubit mapping, Q^{2}Chemistry implements an efficient Jordan-Wigner transformation written in pure Julia. Other methods, such as Bravyi-Kitaev, are currently provided through the interface of OpenFermion[47]. The qubit Hamiltonian can then be used to construct quantum gates for measurement in quantum algorithms such as VQE or QPE.

3.2 Wave function

To represent a many-electron quantum state on a quantum computer, the most commonly used strategy is to map its molecular orbitals onto qubits, which can span the Fock space. Such a straightforward orbital-to-qubit approach can be written as

$$\begin{aligned} |\Psi_{\mathrm{HF}}\rangle &= |i_0 i_1 \cdots\rangle, \\ |\Psi_{\mathrm{CI}}\rangle &= \sum_{i_0,i_1,\cdots} c_{i_0 i_1 \cdots} |i_0 i_1 \cdots\rangle, \end{aligned} \tag{6}$$

where $i_j \in \{0, 1\}$ represents both the occupation of orbitals and the quantum state $|0\rangle$ or $|1\rangle$ of the corresponding qubit. Eq. (6) actually describes the quantum state corresponding to the eigenstates of a qubit Hamiltonian obtained from the Jordan-Wigner transformation. If other fermion-to-qubit mapping algorithms such as Bravyi-Kitaev are used for the Hamiltonian,

DOI: 10.52396/JUSTC-2022-0118
JUSTC, **2022**, 52(12): 2

such a correspondence does not necessarily exist. With a quantum state mapped to qubits, Q²Chemistry can provide a Hadamard test to evaluate the expectation value of the quantum state with respect to an operator.

In simple orbital-to-qubit mapping, the number of qubits required to simulate the wave function has a linear dependence on the number of basis functions, which prohibits the use of a large basis set on NISQ devices. Q²Chemistry provides another strategy that maps a classical tensor network (TN) state onto quantum circuits[48,49]. In this way, the qubits determine the classical bond dimension of the tensor network. For chemical systems that contain weak electron correlations or have a special symmetry, such a TN-based strategy provides a possible solution to effectively reduce the number of qubits at the expense of performing more measurements. Generally, different wave function mapping strategy leads to distinctive structures of the quantum circuits, therefore, may bring special restrictions to subsequent quantum algorithms and measurement protocols. The `q2chem.qchem` module passes the mapping strategy to quantum algorithms in `q2chem.qalgo` to generate an abstract circuit class for initialization. The abstract circuits are then extended according to the adopted quantum algorithm and instantiated to a common readable quantum circuit that can be used in `q2chem.qcirc`.

4 The quantum circuit module

The `q2chem.qcirc` module provides quantum circuit execution functionalities. It mainly contains two parts: an interface reserved for quantum computer manufacturers and a classical simulator that simulates state evolution determined by quantum circuits on a classical computer.

4.1 Interfaces with quantum devices

Functionalities in `q2chem.qcirc` are inherited from the `_Base` class. High-level Python APIs enable efficient extensions of Q²Chemistry, which can be used to interface with different quantum device operation systems. Currently, there are multiple competing technical routes of quantum computers, e.g., photonic qubits have long coherence time, while superconducting platforms have good scalability. To efficiently and accurately study quantum chemical problems, various quantum hardwares may be required depending on the characteristics of the quantum algorithm. Therefore, we provide a flexible interface for connecting to different quantum hardware platforms as illustrated in Code Example 1. In a general `_BaseHardware`, the circuit generated by Q²Chemistry is first represented by quantum assembly language such as OpenQASM via `_compile_circuit`(). This step generally should include circuit optimization and reconstruction to fit the architecture of specific hardware. Then, the sequence of gates is converted into signals or pulses and sent to the quantum devices through `_quantum_hardware_evolution`() using a compiler provided by the vendor of the hardware system. Finally, the results such as measurement statistics or bit strings are collected and postprocessed in Q²Chemistry to obtain the required quantities.

4.2 Classical quantum circuit simulator

Q²Chemistry implements several simulators to run quantum circuits on a classical computer. The quantum state of multiple qubits can be expanded using a certain basis set

$$|\Psi\rangle = \sum_{i_1 i_2 \ldots i_N} c_{i_1 i_2 \ldots i_N} |i_1 i_2 \ldots i_N\rangle, \tag{7}$$

where N is the number of qubits, and $|i_1 i_2 \ldots i_N\rangle$ is the basis state. The coefficients $c_{i_1 i_2 \ldots i_N}$ form an N-dimensional tensor that contains 2^N amplitudes. Eq. (7) is similar to the correlated wave function in quantum chemistry represented by a configuration interaction (CI) expansion.

Currently, there are two strategies to simulate the evolution of the quantum state on a classical computer. One is brute-force simulation which uses a $(2^N \times 1)$ state vector (SV) or a

Table 2. Backends for circuit simulations in present open-source quantum computation softwares. External indicates that one or more third-party packages are required to enable the functionality. SA stands for single-amplitude simulation.

Software	Backend			Parallelism		GPU	Automatic differentiation
	SV	DM	TN	Threaded	Distributed		
ProjectQ[28]	✓	Ongoing	×	✓	SV	×	×
HiQ[50]	✓	Ongoing	SA	✓	SV	Ongoing	×
MindQuantum[51]	✓	✓	×	✓	×	✓	✓
Qulacs[30]	✓	✓	×	✓	×	✓	×
Yao[31]	✓	✓	Ongoing	✓	×	✓	✓
Qiskit[29]	✓	✓	MPS	✓	SV	✓	×
Cirq[52]	✓	✓	External	✓	×	✓	×
PaddleQuantum[53]	✓	✓	×	✓	×	✓	✓
PennyLane[32]	✓	✓	MPS	✓	×	External	✓
QuEST[54]	✓	✓	×	✓	SV, DM	✓	×
TEQUILA[55]	✓	✓	×	✓	×	✓	✓
QFORTE[56]	✓	×	×	✓	×	×	×
XACC[57]	✓	✓	External	✓	✓	External	×
Q²Chemistry	✓	✓	MPS	✓	✓	✓	✓

DOI: 10.52396/JUSTC-2022-0118
JUSTC, **2022**, 52(12): 2

```python
class _BaseQPU(object):
  def __init__(self, options, ...):
    self.quantum_state = ...
    ...
  def evolve_circuit(self, ...):
    ...

class _BaseHardware(_BaseQPU):
  def evolve_circuit(self, ...):
    circuit_asm = self._compile_circuit(...)
    measurement_result = self._quantum_hardware_evolution(...)
    ...
  def _compile_circuit(self, ...):
    # Implement for specific hardware.
  def _quantum_hardware_evolution(self, ...):
    # Implement for specific hardware.

class _BaseSimulator(_BaseQPU):
  def evolve_circuit(self, ...):
    ...
    for op in quantum_circuit:
          if isinstance(op, QuantumGate):
              self._quantum_gate_evolution(...)
          elif isinstance(op, MeasurementOp):
              self._measure_qubit(...)
    ...
  def _quantum_gate_evolution(self, ...):
    # Implement in custom simulators.
  def _measure_qubit(self, ...):
    # Implement in custom simulators.
```

Code Example 1. Interface to quantum devices in the qcircuit module.

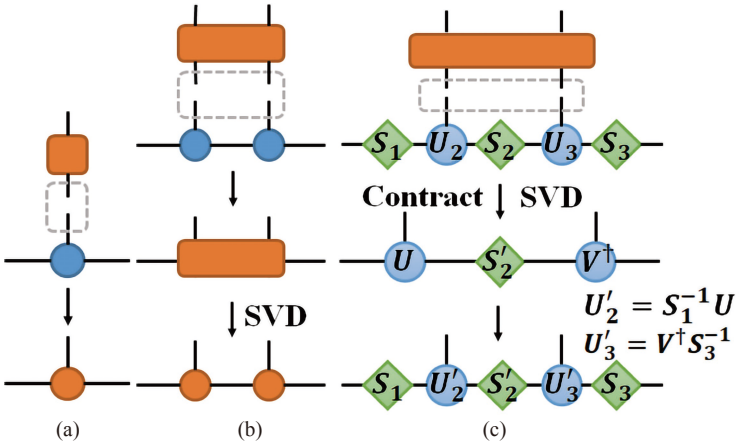


Fig. 2. (a) Applying a single qubit gate on the MPS quantum state is simulated by simply a local contraction. (b) Applying a two-qubit gate on neighboring qubits generally has 2 steps: (i) reshape the two-qubit gate into a 4-dimensional tensor and contract with the qubits to form a two-qubit tensor; (ii) perform a singular value decomposition to restore the two-qubit tensor back to the MPS formulation. Postprocessing is usually required to maintain normalization or canonicalization of MPS tensors. (c) Auxiliary matrices that contain truncated and normalized singular values are used to normalize the quantum state.

DOI: 10.52396/JUSTC-2022-0118
JUSTC, **2022**, 52(12): 2

$(2^N \times 2^N)$ density matrix (DM) to represent the quantum state. The other is tensor network methods which approximate the quantum state by a set of low-rank tensors. The brute-force methods are supported by most of the existing packages (Table 2). Q²Chemistry also supports the state vector and density matrix methods. These brute-force methods implemented in the `q2chem.qcirc` module use the compressed sparse row (CSR) format for sparse matrix storage and OpenMP for multithreaded calculation. OpenACC-enabled C++ code is also implemented and can be selected at compile time to utilize GPU, which can provide 2 – 10 times speed-up over the multithreaded CPU version.

A typical drawback of brute-force simulation is that the memory usage and computational complexity both scale as $O(c^N)$, where c is a constant between 2 and 4 and N is the number of qubits. This exponential scaling prevents quantum simulations for larger molecules with more than ~30 qubits. Recently, matrix product states (MPS) and projected entangled pair states (PEPS) have been used to simulate large scale random quantum circuits[58–61]. Since the tensor contraction pattern is mostly fixed and does not have NP-hard path optimization problems[62], the one-dimensional MPS is preferable as a high-performance tensor network backend for a general quantum circuit simulator.

The MPS ansatz factorizes the rank-N coefficients into lower rank tensors, which can be written as

$$c_{i_1 i_2 \cdots i_N} = \sum_{u_0 \cdots u_N} {}^{1}T^{i_1}_{u_0 u_1} {}^{2}T^{i_2}_{u_1 u_2} \cdots {}^{N}T^{i_N}_{u_{N-1} u_N} , \tag{8}$$

where ${}^{k}T^{i_k}_{u_{k-1} u_k}$ is a rank-3 tensor with i_k called the physical index and u_k the auxiliary index. The maximum size of the auxiliary indices is defined as the bond dimension of the MPS, which is denoted as $D = \max_{0 \le k \le N} \{u_k\}$. Algorithms based on MPS generally have a complexity of $O(ND^3)$. If the bond dimension D is allowed to grow exponentially, the MPS can exactly represent any quantum state using Eq. (8).

The `q2chem.qcirc` module implements the MPS simulation algorithm on top of the machine learning framework PyTorch[40] based on the algorithm proposed by Vidal et al.[63,64]. A demonstrative procedure for simulating quantum circuits using the MPS algorithm is presented in Fig. 2. Different from common classical MPS-based methods such as the density matrix renormalization group (DMRG)[37], a set of auxiliary matrices are inserted between the rank-3 tensors and stored to maintain a normalized quantum state after truncated SVD.

Classical quantum simulator backends can also be extended within the framework of Q²Chemistry in a similar style as the interfaces for hardware. It requires only a minimum effort to implement functions `_quantum_gate_evolution`() and `_measure_qubit`() for a derived class of `_BaseSimulator` and, if necessary, a custom data structure to store the quantum state. No modification to the upper level modules such as expectation value evaluation or higher level quantum algorithms is needed. In Section 6.2, we show the application of Q²Chemistry interfaced with an external MPS simulator QuantumSpins[59,65], which efficiently simulated a 40-qubit molecule with a high accuracy.

4.3 Reversible automatic differentiation

In many quantum algorithms, a parametric quantum circuit is constructed, and optimization of the circuit parameters is carried out iteratively. On a quantum computer, the gradients of energy or another target function with respect to circuit parameters can be calculated through the parameter-shift rule or finite difference steps, which will introduce an additional complexity factor of $O(N_p)$, where N_p is the number of parameters. Nevertheless, using a classical quantum circuit simulator the gradients can be evaluated efficiently with an approximately $O(1)$ complexity, which is extremely helpful just as the reverse-mode automatic differentiation (AD) algorithm used in classical machine learning if a large number of parameters are involved. Different from conventional machine learning, the computation graph of a quantum expectation value with respect to variational parameters is reversible:

$$|\Psi\rangle_n = U_n |\Psi\rangle_{n-1} \Leftrightarrow |\Psi\rangle_{n-1} = U_n^\dagger |\Psi\rangle_n, \tag{9}$$

which indicates that the intermediate quantum states $\{|\Psi\rangle_{n-1}, |\Psi\rangle_{n-2}, \cdots\}$ can be calculated iteratively and thus do not need to be stored explicitly in memory, enabling a significant reduction in memory consumption during the reverse-mode gradient evaluations[66–68]. This classical algorithm, which is termed reversible AD, is first implemented in the Julia package Yao.jl, and a detailed description is given in Algorithm 1.

Algorithm 1: Reversible AD algorithm to calculate $g_i = \frac{\partial E}{\partial \theta_i}$.

Data: $\hat{H}$, $\{U_0(\theta_0), U_0(\theta_0), \ldots\}$, $|\Psi\rangle$
Result: g: gradients of energy w.r.t. parameters $\{\theta_0, \theta_1, \ldots\}$
$N_p \leftarrow$ number of parameters, $g \leftarrow$empty array of length N_p;
for i=0; $i \leqslant N_p - 1$; i+=1 **do**
  $|\Psi\rangle \leftarrow U_i(\theta_i)|\Psi\rangle$;
end
$|\Psi_l\rangle \leftarrow \hat{H}|\Psi\rangle$, $|\Psi_r\rangle \leftarrow |\Psi\rangle$;
for i=$N_p - 1$; $i \geqslant 0$; i-=1 **do**
  $|\Psi_r\rangle \leftarrow U_i^\dagger(\theta_i)|\Psi_r\rangle$;
  $g[i] = 2 \times \mathrm{Re}(\langle\Psi_l| \frac{\partial U_i(\theta_i)}{\partial \theta_i} \Psi_r\rangle)$;
  $|\Psi_l\rangle \leftarrow U_i^\dagger(\theta_i)|\Psi_l\rangle$;
end

Q²Chemistry implements reversible AD for brute-force backends. It should be noted that the reversible AD algorithm is invalid on real quantum devices even if the original equation is used:

$$\frac{\partial E}{\partial \theta_i} = 2 \times \mathrm{Re}(\langle\Psi| U_0^\dagger \cdots U_{N_p-1}^\dagger \hat{H} U_{N_p-1} \cdots U_{i+1} \frac{\partial U_i}{\partial \theta_i} U_{i-1} \cdots U_0 |\Psi\rangle),$$

since the derivative gates $\left\{\frac{\partial U_i(\theta_i)}{\partial \theta_i}\right\}$ are generally nonunitary. At the same time, using reversible AD with the MPS simulator requires careful modifications to the algorithm and specially designed techniques due to SVD truncation during the simulation. A naive implementation of Algorithm 1 probably leads to suboptimal performance and numerical errors.

5 Quantum algorithms

In the current release, Q²Chemistry provides a couple of VQE-based algorithms. The module `q2chem.qalgo` adopted several variational wave function ansatzes that can be used indi-

DOI: 10.52396/JUSTC-2022-0118
JUSTC, **2022**, 52(12): 2

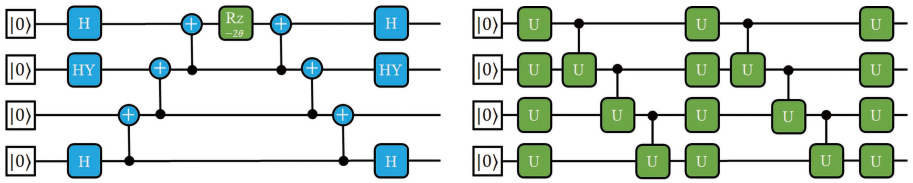


Fig. 3. (a) The quantum circuit corresponding to the operator $\exp(\mathrm{i}\theta\hat{\sigma}^x\hat{\sigma}^y\hat{\sigma}^z\hat{\sigma}^x)$ and (b) the two-layer HEA circuit that entangles all neighboring qubits using the controlled-U gate. Blue squares represent nonparametric gates, while green squares represent parametric quantum gates such as Rz and the three-parameter (controlled-)U gate.

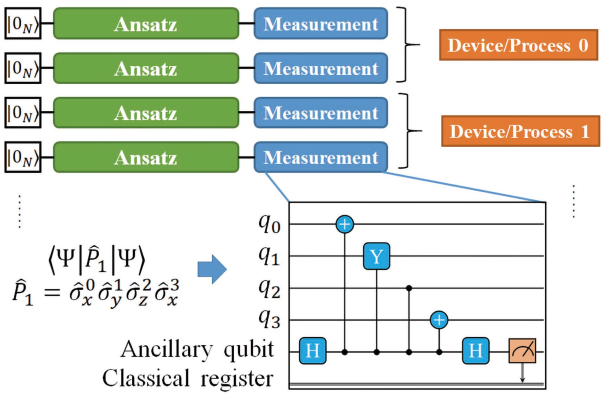


Fig. 4. Evaluating the expectation value of a linear combination of Pauli strings using multiple quantum devices or simulator processes. $|0_N\rangle$ represents an N-qubit quantum register with all qubits initialized to $|0\rangle$. In this example, the measurement parts are the Hadamard test circuits.

vidually or collectively to solve for the eigenstates of the given chemical system. New quantum algorithms can also be conveniently implemented by users.

5.1 Variational quantum circuit ansatz

Introducing a parametric wave function $|\Psi(\theta)\rangle$, the lowest eigenvalue E_0 can be obtained variationally:

$$E_0 = \min_\theta \langle\Psi(\theta)|\hat{H}|\Psi(\theta)\rangle. \tag{10}$$

Such a protocol is implemented in the `q2chem.qalgo` module to obtain the eigenstates of a given Hamiltonian. Properly constructing a parametric quantum circuit, the wave function ansatz $|\Psi(\theta)\rangle$ is encoded into the quantum state of qubits. Combining the measurements for expectation value evaluation on a quantum computer and a numerical optimization algorithm on a classical computer, the variational procedure can then be performed in a hybrid quantum-classical way.

Generally, two broad types of variational quantum circuit ansatzes exist[8]. They are physically motivated ansatz (PMA), which is inspired by classical wave function methods that systematically approach the exact electronic wave function, and hardware heuristic ansatz (HHA), which considers specific hardware structures and employs entangling blocks. Both types are currently integrated into the `q2chem.qalgo` module.

Unitary coupled-cluster is one of the most commonly used PMAs for quantum computing. Generally, the UCC wave function is defined as

$$|\Psi(\theta)^{\mathrm{UCC}}\rangle = \exp(\hat{T}(\theta) - \hat{T}^\dagger(\theta))|\Psi^{\mathrm{HF}}\rangle. \tag{11}$$

In Q^{2}Chemistry, the spin-adapt CCD0 cluster operators[69] are used to construct the UCCSD and UCCGSD wave functions. A fermion-to-qubit mapping is performed, and first-order Trotter-Suzuki decomposition[70,71] is implemented to convert Eq. (11) into the product of Pauli strings:

$$|\Psi(\theta)^{\mathrm{UCC}}\rangle = \prod_i \prod_j \exp(\mathrm{i}\theta_i c_{ij}\hat{P}_{ij})|\Psi^{\mathrm{HF}}\rangle, \tag{12}$$

where θ_i is the variational parameter corresponding to the i-th anti-Hermitian fermion excitation operator $\hat{T}_i - \hat{T}_i^\dagger$, which is transformed into qubit form $\sum_j \mathrm{i}c_{ij}\hat{P}_{ij}$. Each exponential term is mapped to the quantum circuit following Algorithm 2. An example of the mapped circuit is presented in Fig. 3a.

The hardware efficient ansatz (HEA) circuit[14] is implemented as an HHA ansatz. If the type of entanglement gates, the ordering of entanglement qubits and the number of layers are set, Q^{2}Chemistry automatically generates the parametric hardware efficient circuit for simulation. Fig. 3b shows an example of a two-layer HEA circuit with all neighboring qubits entangled by the three-parameter controlled-U gate.

Algorithm 2: Map $\exp(\mathrm{i}\theta\hat{P})$ to circuit. HY is the Hadamard-Y gate defined as $\mathrm{HY} = \sqrt{2}/2\times(\mathrm{Z}+\mathrm{Y})$

Data: $\hat{P}$, θ
Result: C: the quantum circuit
$N_q \leftarrow$ number of qubits, C $\leftarrow$ empty circuit;
for i=0; $i\leqslant N_q-1$; i+=1 **do**
  $p_i = P[i]$;
  if $p_i == \hat{\sigma}_x$ **then**
    C+ = H_i
  else if $\mathbf{p_i} == \hat{\sigma}_\mathrm{y}$ **then**
    C+ = HY_i
  end
end
for i=N_q-2; $i\geqslant 0$; i-=1 **do**
  C += $\mathrm{CNOT}_{(i+1,i)}$
end
C += $\mathrm{RZ}(-2\theta)_{N_q-1}$
for i=0; $i\leqslant N_q-2$; i+=1 **do**
  C += $\mathrm{CNOT}_{i+1,i}$
end
for i=0; $i\leqslant N_q-1$; i+=1 **do**
  $p_i = P[i]$;
  if $p_i == \hat{\sigma}_x$ **then**
    C+ = H_i
  else if $p_i == \hat{\sigma}_y$ **then**
    C+ = HY_i
  end

DOI: 10.52396/JUSTC-2022-0118
JUSTC, **2022**, 52(12): 2

end

Several techniques are implemented in Q²Chemistry to reduce the computational overhead, including qubit tapering for Hamiltonian[72], qubit excitation based (QEB) operator[73], Pauli entangler[74,75], symmetry-based operator selection[33] and the ADAPT-VQE algorithm[76]. Q²Chemistry supports combination of above methods, for example, using Pauli entanglers that entangle at most 4 qubits together with ADAPT-VQE leads to an iterative-qubit-coupled-cluster (iQCC)[75] like algorithm.

In addition to ground state methods, the `q2chem.qalgo` module offers a couple of post-VQE algorithms for calculating excited states, including the variational quantum deflation (VQD)[18], quantum subspace expansion (QSE)[19] and equation-of-motion (EOM)[45,77,78] theory. VQD consecutively constructs an effective Hamiltonian, the lowest eigenvalue of which corresponds to the 1st, 2nd, 3rd, … excited state energy:

$$\hat{H}^{i}_{\text{eff}} = \hat{H}^{i-1}_{\text{eff}} + \alpha_i |\Psi^{i-1}\rangle\langle\Psi^{i-1}|, \tag{13}$$

where $\hat{H}^{0}_{\text{eff}}$ is the original second-quantized electronic Hamiltonian. QSE and EOM use a set of fermion excitation operators or a state-transfer operator to construct and solve a generalized eigenvalue problem $\boldsymbol{MC} = \boldsymbol{SCE}$ by additional quantum measurements on the top of the ground state circuit. These methods are implemented in Q²Chemistry for calculating electron excitations, ionization potentials and electron affinity energies for both molecules and periodic systems.

5.2 Parallel evaluation of an expectation value

As shown in Eq. (2), the Hamiltonian is expressed as the summation of a polynomial number of mutually uncorrelated Pauli strings. The expectation values of each Pauli string can thus be calculated independently. Fig. 4 gives an example of circuits used for evaluating the expectation value $\sum_i c_i \langle\Psi|\hat{P}_i|\Psi\rangle$ for a given Hamiltonian containing a number of Pauli strings under some fermion-to-qubit transformation. For each of the circuits, the ansatz parts are the same while the measurement parts are constructed according to the specific form of the Pauli string. During the calculation of the expectation value, Q²Chemistry automatically distributes these circuits to different quantum devices or simulator processes. On each device, a subset of circuits are executed and then measured. The measurement outcomes are then postprocessed to calculate expectation values locally. Finally, the results are reduced across all the devices to obtain the total energy.

It should be noted that on real quantum devices, since the quantum states are nonreplicable, all the circuits should be executed. However, on a classical simulator, the memory data of quantum states can be reused. Therefore, using a quantum circuit simulator, the ansatz part only needs to be executed once, and the simulated quantum state can then be copied to each process for later measurements. This strategy is used in the simulations of Section 6.

6 Applications

We present several simulations to show the power of Q²Chemistry, including a scalability test for the MPS-based quantum circuit simulator and the numerical simulation results for ground- and excited-state calculations.

6.1 Scalability benchmark

The parallel measurement for the expectation value introduced in Section 5.2 is extended to a two-level parallelism for the MPS simulator:

(I) The first level parallelization over Hamiltonian. Subsets of Pauli strings from the Hamiltonian in Eq. (1) are distributed to each process. The expectation values of Pauli strings are calculated independently, and a reduce-sum is performed across all processes to obtain the final energy.

(II) The second low-level parallelization using multithread parallelism on CPU or GPU, to accelerate the calculations of linear algebra routines such as matrix multiplication and singular value decomposition.

The two-level strategy enables good parallel scalability if the adapted dynamical distribution algorithm is used to achieve load balance. To reduce memory usage, in the first level, the quantum circuit is stored and evolved only on the 0th process. Although for small molecules such as H_2 (4 qubits), the circuit evolution may contribute over 80% to the total execution time, the number of Pauli strings in the Hamiltonian will quickly go beyond 10^5 due to the $O(N^4)$ scaling, and the time cost of circuit simulation will become negligible if larger systems with more than 12 qubits are involved. The scaling benchmark of Q²Chemistry simulating a Cr_2 molecule (STO-3G basis set) using the MPS backend is given in Fig. 5. Q²Chemistry achieves good parallel performance up to 768 CPU cores for this 72-qubit system. Note that the main purpose of this test is for parallelization scaling instead of obtaining reliable results. Therefore, only 51 of the 3131 variational parameters are considered, leading to a quantum circuit with 119884 gates. The upper bound for the bond dimension is also set to a relatively low value of 64.

6.2 Numerical results

Fig. 6a shows the potential energy curve of the H_2 molecule. The calculation is carried out by extending the simulators in

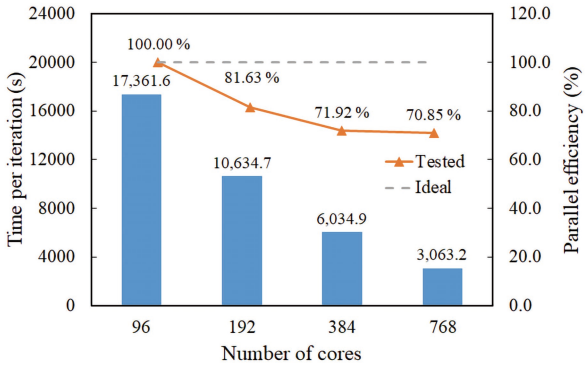
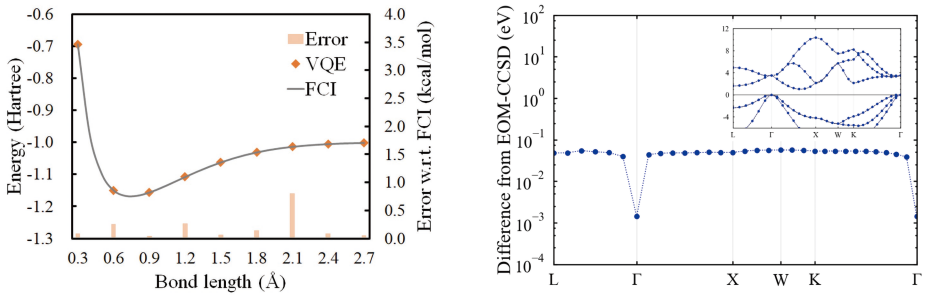


Fig. 5. Simulating the Cr_2 molecule using the MPS backend. The STO-3G basis set and the symmetry-reduced UCCSD ansatz are used, where the qubit Hamiltonian contains 305041 Pauli strings. The time cost refers to one VQE iteration (including evolution of the quantum circuit and calculation of energy) being tested. The distributed parallelization is implemented using OpenMPI and Python's `mpi4py` package.

DOI: 10.52396/JUSTC-2022-0118
JUSTC, **2022**, 52(12): 2



(a) (b)

Fig. 6. (a) The VQE optimized potential energy curve of H_2 calculated by the MPS backend using the ccJ-pVDZ basis set. The VQE results are calculated by interfacing with the external Julia-implemented MPS simulator. (b) Energy difference between EOM-ADAPT-C and classical EOM-CCSD band structures for Si. Inset gives the EOM-ADAPT-C band structure calculated using Q²Chemistry. The FCI and EOM-CCSD energies are obtained using the PySCF code.

the `q2chem.qcirc` module to an external MPS circuit simulator written in Julia[59,65]. Benefiting from the high-level modular structure introduced in Section 2, this external simulator and the parallelization techniques introduced in Section 5.2 can be easily implemented within the framework of Q²Chemistry by adding a few lines of code. The ground-state energies are variationally optimized using the ccJ-pVDZ basis set[79] and the symmetry-reduced UCCSD[33] circuit (leading to 40 qubits and 53 variational parameters). The BOBYQA optimizer is used to perform gradient-free optimizations. For each geometry, 2000 optimization steps are performed within 24 h using 560 CPU cores.

Fig. 6b calculates the quasi-particle band structures for silicon. For such periodic systems, the Hamiltonian and UCC-based wave function ansatz need to include the constraint of crystal momentum conservation[44], which is automatically handled by the `q2chem.qchem` module. The simulation uses the GTH-SVZ basis set with a GTH-PADE pseudopotential. A UCCGSD operator pool with complementary operators [45] is employed together with the ADAPT algorithm for the ground state ADAPT-C wave function. With a $1\times1\times1$ k-point grid, 16 qubits are simulated using the state vector backend. The EOM-ADAPT-C method[45] is used for band structure calculation, which achieves a mean absolute difference of 0.047 eV from EOM-CCSD and the deviation is as small as $\sim 10^{-3}$ eV at the Γ point (Fig. 6b).

7 Conclusions

In this study, we demonstrate that the Q²Chemistry package is suitable for simulating and developing quantum algorithms for quantum chemistry applications. Q²Chemistry provides versatile functionalities for simulating ground- and excited-state properties. The simulator backend, including the parallelized MPS algorithm, achieves high performance for large-scale simulations using a moderate amount of computational resources. Directions for future development include more classical simulation methods, including high-dimensional tensor network based methods, more integrated quantum algorithms, and high-performance quantum circuit optimization algorithms. With the flexibility to link to different quantum devices, Q²Chemistry can be used as a useful platform in pursuing a practical quantum advantage.

Supporting information

The supporting information for this article can be found online at https://doi.org/10.52396/JUSTC-2022-0118. The supporting information includes one figure. The way to count CNOT gates is described in the supporting information.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (21825302), the Fundamental Research Funds for the Central Universities (WK2060000018), the National Supercomputing Center in Jinan, and the USTC Supercomputing Center.

Conflict of interest

The authors declare that they have no conflict of interest.

Preprint statement

Research presented in this article was posted on a preprint server prior to publication in *JUSTC*. The corresponding preprint article can be found here: arXiv: 2208.10978. https://arxiv.org/abs/2208.10978.

Biographies

Yi Fan received his B.S. degree in Chemistry from the University of Science and Technology of China (USTC) in 2017 and is currently a Ph.D. candidate at USTC. His research interests include quantum chemistry methods and quantum computing algorithms for solid materials.

Zhenyu Li is a Professor of chemistry at the University of Science and Technology of China (USTC). He received his Ph.D. degree in Physical Chemistry from USTC in 2004. Since then, he worked as a postdoctoral researcher at the University of Maryland, College Park, and the University of California, Irvine. In 2007, he joined USTC as a faculty member. His research interests focus on using or developing electronic structure and molecular simulation methods to study chemical systems.

References

[1] Preskill J. Quantum Computing in the NISQ era and beyond. *Quantum*, **2018**, *2*: 79.
[2] McArdle S, Endo S, Aspuru-Guzik A, et al. Quantum computational chemistry. *Rev. Mod. Phys.*, **2020**, *92*: 015003.
[3] Yung M H, Casanova J, Mezzacapo A, et al. From transistor to trapped-ion computers for quantum chemistry. *Sci. Rep.*, **2014**, *4* (1): 3589.

DOI: 10.52396/JUSTC-2022-0118
JUSTC, **2022**, 52(12): 2

[4] Tilly J, Chen H, Cao S, et al. The Variational Quantum Eigensolver: a review of methods and best practices. **2021**. https://arxiv.org/abs/2111.05176. Accessed August 1, 2022.
[5] Cerezo M, Arrasmith A, Babbush R, et al. Variational quantum algorithms. *Nat. Rev. Phys.,* **2021**, *3* (9): 625–644.
[6] Magann A B, Arenz C, Grace M D, et al. From pulses to circuits and back again: A quantum optimal control perspective on variational quantum algorithms. *PRX Quantum,* **2021**, *2*: 010101.
[7] Fedorov D A, Peng B, Govind N, et al. VQE method: a short survey and recent developments. *Mater. Theory,* **2022**, *6*: 2.
[8] Cao Y, Romero J, Olson J P, et al. Quantum chemistry in the age of quantum computing. *Chem. Rev.,* **2019**, *119*: 10856–10915.
[9] Aspuru-Guzik A, Dutoi A D, Love P J, et al. Simulated quantum computation of molecular energies. *Science,* **2005**, *309*: 1704–1707.
[10] Peruzzo A, McClean J, Shadbolt P, et al. A variational eigenvalue solver on a photonic quantum processor. *Nat. Commun.,* **2014**, *5*: 4213.
[11] Hempel C, Maier C, Romero J, et al. Quantum chemistry calculations on a trapped-ion quantum simulator. *Phys. Rev. X,* **2018**, *8*: 031022.
[12] Nam Y, Chen J S, Pisenti N C, et al. Ground-state energy estimation of the water molecule on a trapped-ion quantum computer. *npj Quantum Inf.,* **2020**, *6*: 33.
[13] O'Malley P J J, Babbush R, Kivlichan I D, et al. Scalable quantum simulation of molecular energies. *Phys. Rev. X,* **2016**, *6*: 031007.
[14] Kandala A, Mezzacapo A, Temme K, et al. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature,* **2017**, *549* (7671): 242–246.
[15] Colless J I, Ramasesh V V, Dahlen D, et al. Computation of molecular spectra on a quantum processor with an error-resilient algorithm. *Phys. Rev. X,* **2018**, *8*: 011021.
[16] McClean J R, Romero J, Babbush R, et al. The theory of variational hybrid quantum-classical algorithms. *New J. Phys.,* **2016**, *18*: 023023.
[17] Lanyon B P, Whitfield J D, Gillett G G, et al. Towards quantum chemistry on a quantum computer. *Nat. Chem.,* **2010**, *2*: 106–111.
[18] Higgott O, Wang D, Brierley S. Variational quantum computation of excited states. *Quantum,* **2019**, *3*: 156.
[19] McClean J R, Kimchi-Schwartz M E, Carter J, et al. Hybrid quantum-classical hierarchy for mitigation of decoherence and determination of excited states. *Phys. Rev. A,* **2017**, *95*: 042308.
[20] Liu J, Fan Y, Li Z, et al. Quantum algorithms for electronic structures: basis sets and boundary conditions. *Chem. Soc. Rev.,* **2022**, *51*: 3263–3279.
[21] McClean J R, Boixo S, Smelyanskiy V N, et al. Barren plateaus in quantum neural network training landscapes. *Nat. Commun.,* **2018**, *9* (1): 4812.
[22] Napp J. Quantifying the barren plateau phenomenon for a model of unstructured variational ansätze. **2022**. https://arxiv.org/abs/2203.06174. Accessed August 1, 2022.
[23] Anschuetz E R, Kiani B T. Beyond barren plateaus: Quantum variational algorithms are swamped with traps. **2022**. https://arxiv.org/abs/2205.05786. Accessed August 1, 2022.
[24] Arute F, Arya K, Babbush R, et al. Hartree-Fock on a superconducting qubit quantum computer. *Science,* **2020**, *369* (6507): 1084–1089.
[25] Huggins W J, O' Gorman B A, Rubin N C, et al. Unbiasing fermionic quantum Monte Carlo with a quantum computer. *Nature,* **2022**, *603* (7901): 416–420.
[26] Bartlett R J, Kucharski S A, Noga J. Alternative coupled-cluster ansätze Ⅱ. The unitary coupled-cluster method.. *Chem. Phys. Lett.,* **1989**, *155*: 133–140.
[27] Taube A G, Bartlett R J. New perspectives on unitary coupled-cluster theory. *Int. J. Quantum Chem.,* **2006**, *106*: 3393–3401.
[28] Steiger D S, Häner T, Troyer M. ProjectQ: an open source software framework for quantum computing. *Quantum,* **2018**, *2*: 49.
[29] ANIS M S, Mitchell A, Abraham H, et al. Qiskit. **2021**. https://github.com/Qiskit/qiskit. Accessed August 1, 2022.
[30] Suzuki Y, Kawase Y, Masumura Y, et al. Qulacs: a fast and versatile quantum circuit simulator for research purpose. *Quantum,* **2021**, *5*: 559.
[31] Luo X Z, Liu J G, Zhang P, et al. Yao.jl: Extensible, efficient framework for quantum algorithm design. *Quantum,* **2020**, *4*: 341.
[32] Bergholm V, Izaac J, Schuld M, et al. PennyLane: Automatic differentiation of hybrid quantum-classical computations. **2018**. https://arxiv.org/abs/1811.04968. Accessed August 1, 2022.
[33] Cao C, Hu J, Zhang W, et al. Progress toward larger molecular simulation on a quantum computer: Simulating a system with up to 28 qubits accelerated by point-group symmetry. *Phys. Rev. A,* **2022**, *105*: 062452.
[34] Bezanson J, Edelman A, Karpinski S, et al. Julia: A fresh approach to numerical computing. *SIAM Review,* **2017**, *59* (1): 65–98.
[35] Sun Q, Berkelbach T C, Blunt N S, et al. PySCF: the Python-based simulations of chemistry framework. *WIREs Comput. Mol. Sci.,* **2018**, *8*: e1340.
[36] Orús R. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Ann. Phys.,* **2014**, *349*: 117–158.
[37] Schollwöck U. The density-matrix renormalization group in the age of matrix product states. *Ann. Phys.,* **2011**, *326* (1): 96–192.
[38] Harris C R, Millman K J, van der Walt S J, et al. Array programming with NumPy. *Nature,* **2020**, *585* (7825): 357–362.
[39] Virtanen P, Gommers R, Oliphant T E, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods,* **2020**, *17*: 261–272.
[40] Paszke A, Gross S, Massa F, et al. PyTorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, editors. Advances in Neural Information Processing Systems 32. New York: Curran Associates, Inc., **2019**: 8024–8035.
[41] Jordan P, Wigner E. Über das paulische äquivalenzverbot. *Z. Physik,* **1928**, *47*: 631–651.
[42] Seeley J T, Richard M J, Love P J. The Bravyi-Kitaev transformation for quantum computation of electronic structure. *J. Chem. Phys.,* **2012**, *137*: 224109.
[43] Tranter A, Love P J, Mintert F, et al. A comparison of the Bravyi-Kitaev and Jordan-Wigner transformations for the quantum simulation of quantum chemistry. *J. Chem. Theory Comput.,* **2018**, *14*: 5617–5630.
[44] Liu J, Wan L Y, Li Z Y, et al. Simulating periodic systems on a quantum computer using molecular orbitals. *J. Chem. Theory Comput.,* **2020**, *16*: 6904–6914.
[45] Fan Y, Liu J, Li Z Y, et al. Equation-of-motion theory to calculate accurate band structures with a quantum computer. *J. Phys. Chem. Lett.,* **2021**, *12* (36): 8833–8840.
[46] Smith D G A, Gray J. opt_einsum - A Python package for optimizing contraction order for einsum-like expressions. *J. Open Source Softw.,* **2018**, *3* (26): 753.
[47] McClean J R, Sung K J, Kivlichan I D, et al. OpenFermion: The electronic structure package for quantum computers. **2017**. https://arxiv.org/abs/1710.07629. Accessed August 1, 2022.
[48] Liu J G, Zhang Y H, Wan Y, et al. Variational quantum eigensolver with fewer qubits. *Phys. Rev. Res.,* **2019**, *1*: 023025.
[49] Haghshenas R, Gray J, Potter A C, et al. Variational power of quantum circuit tensor networks. *Phys. Rev. X,* **2022**, *12*: 011047.
[50] Nguyen D, Mikushin D, Man-Hong Y. HiQ-ProjectQ: Towards user-friendly and high-performance quantum computing on GPUs. In: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, **2021**: 1056–1061.
[51] MindQuantum Developer. MindQuantum, version 0.6.0. **2021**. https://gitee.com/mindspore/mindquantum. Accessed August 1, 2022.

DOI: 10.52396/JUSTC-2022-0118
JUSTC, **2022**, 52(12): 2

[52] Cirq Developers. Cirq. **2022**. https://github.com/quantumlib/Cirq. Accessed August 1, 2022.
[53] Paddle Quantum Developers. Paddle Quantum. **2020**. https://github.com/PaddlePaddle/Quantum. Accessed August 1, 2022.
[54] Jones T, Brown A, Bush I, et al. QuEST and high performance simulation of quantum computers. *Sci. Rep.*, **2019**, *9* (1): 10736.
[55] Kottmann J S, Alperin-Lea S, Tamayo-Mendoza T, et al. TEQUILA: a platform for rapid development of quantum algorithms. *Quantum Sci. Technol.*, **2021**, *6* (2): 024009.
[56] Stair N H, Evangelista F A. QForte: An efficient state-vector emulator and quantum algorithms library for molecular electronic structure. *J. Chem. Theory Comput.*, **2022**, *18* (3): 1555–1568.
[57] McCaskey A J, Lyakh D I, Dumitrescu E F, et al. XACC: a system-level software infrastructure for heterogeneous quantum-classical computing. *Quantum Sci. Technol.*, **2020**, *5* (2): 024002.
[58] Guo C, Liu Y, Xiong M, et al. General-purpose quantum circuit simulator with projected entangled-pair states and the quantum supremacy frontier. *Phys. Rev. Lett.*, **2019**, *123*: 190501.
[59] Guo C, Zhao Y, Huang H L. Verifying random quantum circuits with arbitrary geometry using tensor network states algorithm. *Phys. Rev. Lett.*, **2021**, *126*: 070502.
[60] Liu X, Guo C, Liu Y, et al. Redefining the quantum supremacy baseline with a new generation sunway supercomputer. **2021**. https://arxiv.org/abs/2111.01066. Accessed August 1, 2022.
[61] McCaskey A, Dumitrescu E, Chen M, et al. Validating quantum-classical programming models with tensor network simulations. *PLoS ONE*, **2018**, *13* (12): e0206704.
[62] Pfeifer R N C, Haegeman J, Verstraete F. Faster identification of optimal contraction sequences for tensor networks. *Phys. Rev. E*, **2014**, *90*: 033315.
[63] Vidal G. Efficient classical simulation of slightly entangled quantum computations. *Phys. Rev. Lett.*, **2003**, *91*: 147902.
[64] Vidal G. Classical simulation of infinite-size quantum lattice systems in one spatial dimension. *Phys. Rev. Lett.*, **2007**, *98*: 070201.
[65] Guo C. QuantumSpins. **2020**. https://github.com/guochu/QuantumSpins. Accessed May 17, 2022.
[66] Gomez A N, Ren M, Urtasun R, et al. The reversible residual network: Backpropagation without storing activations. **2017**. https://arxiv.org/abs/1707.04585. Accessed October 21, 2022.
[67] Chen R T Q, Rubanova Y, Bettencourt J, et al. Neural ordinary differential equations. **2019**. https://arxiv.org/abs/1806.07366. Accessed October 21, 2022.
[68] Jones T, Gacon J. Efficient calculation of gradients in classical simulations of variational quantum algorithms. **2020**. https://arxiv.org/abs/2009.02823. Accessed August 1, 2022.
[69] Bulik I W, Henderson T M, Scuseria G E. Can single-reference coupled cluster theory describe static correlation? *J. Chem. Theory Comput.*, **2015**, *11* (7): 3171–3179.
[70] Grimsley H R, Claudino D, Economou S E, et al. Is the Trotterized UCCSD ansatz chemically well-defined? *J. Chem. Theory Comput.*, **2020**, *16*: 1–6.
[71] Babbush R, McClean J, Wecker D, et al. Chemical basis of Trotter-Suzuki errors in quantum chemistry simulation. *Phys. Rev. A*, **2015**, *91*: 022311.
[72] Bravyi S, Gambetta J M, Mezzacapo A, et al. Tapering off qubits to simulate fermionic Hamiltonians. **2017**. https://arxiv.org/abs/1701.08213. Accessed August 1, 2022.
[73] Yordanov Y S, Armaos V, Barnes C H W, et al. Qubit-excitation-based adaptive variational quantum eigensolver. *Commun. Phys.*, **2021**, *4* (1): 228.
[74] Ryabinkin I G, Yen T C, Genin S N, et al. Qubit coupled cluster method: A systematic approach to quantum chemistry on a quantum computer. *J. Chem. Theory Comput.*, **2018**, *14* (12): 6317–6326.
[75] Ryabinkin I G, Lang R A, Genin S N, et al. Iterative qubit coupled cluster approach with efficient screening of generators. *J. Chem. Theory Comput.*, **2020**, *16* (2): 1055–1063.
[76] Grimsley H R, Economou S E, Barnes E, et al. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nat. Commun.*, **2019**, *10*: 3007.
[77] Krylov A I. Equation-of-motion coupled-cluster methods for open-shell and electronically excited species: The hitchhiker's guide to Fock space. *Annu. Rev. Phys. Chem.*, **2008**, *59*: 433–462.
[78] Ollitrault P J, Kandala A, Chen C F, et al. Quantum equation of motion for computing molecular excitation energies on a noisy quantum processor. *Phys. Rev. Res.*, **2020**, *2*: 043140.
[79] Benedikt U, Auer A A, Jensen F. Optimization of augmentation functions for correlated calculations of spin-spin coupling constants and related properties. *J. Chem. Phys.*, **2008**, *129* (6): 064111.

DOI: 10.52396/JUSTC-2022-0118
JUSTC, **2022**, 52(12): 2