Contents

Chapter 1

Variational Quantum Algorithms

Bavithra Govintharajah
Supervisor: Prof. Dr. Fabian Hassler

Simulation of large quantum systems or solving large-scale linear algebraic problems is classically inefficient due to high computational costs. Quantum computers promise to unlock these uncharted territories, although fault-tolerant quantum computers will likely not be available soon. Quantum devices available at present have serious bottlenecks such as the limited number of qubits and noise that severely limit circuit depths. Variational Quantum Algorithms (VQAs) have emerged as promising candidates to address the constraints of near-term quantum computing. They employ a quantum-classical hybrid approach to train a parametrized quantum circuits. Various VQAs have now been proposed for several applications in various disciplines spanning physics, machine learning, quantum chemistry, and combinatorial optimization and they appear to the best hope for obtaining quantum advantage with near term devices. Nevertheless, open questions remain including the trainability, accuracy, and efficiency of VQAs. In this chapter, an introduction to the concept of VQAs is presented. The key structure of these algorithms, their limitations, and their advantages are discussed in detail. Furthermore, an example of applying a VQA to simulate small molecules is presented.

1.1 Introduction

When early quantum devices became available, it was clear that progress in both the algorithmic and the hardware side was going to require the realization of proof-of-principle implementations of quantum algorithms. The modest resource requirements and flexibility of variational quantum information processing compared to other algorithms such as the Shor's algorithm and the Quantum Phase Estimation (QPE) algorithm, positioned variational approach as one of the first milestones in the road-map of quantum computing experiments. The introduction of the first Variational Quantum Algorithms (VQA) in 2014 coincided with the transition of experimental quantum computing from primarily an academic endeavor to an industrial enterprise. The inevitable question was, and still is: will these early noisy quantum devices deliver an advantage over classical computing and efficient solutions to complex problems? Hope is that, this is the case, and it is believed that a viable path towards a definite affirmative answer lies in the further development and improvement of quantum variational approaches.

One important aspect that marks experimental advancements in quantum computing is that the number of available physical qubits must exceed a specific threshold to explore classically intractable problems. At the time of writing this document, the largest circuit model quantum processors reported comprise of just 10-100 qubits with error rates of $10^{-3} - 10^{-2}$. This is called the Noisy Intermediate Scale Quantum (NISQ) era, a term first coined by John Preskill in 2018 [1]. NISQ devices are characterized by limited coherence times, lower gate fidelities, and fewer qubits. Therefore, NISQ devices will not have adequate resources to implement quantum error correction, and as a consequence, only shallow circuits are executable. However, these machines will be large enough to implement quantum dynamics that cannot be efficiently simulated with existing classical computers because classical computers are notoriously bad at simulating dynamics of highly entangled many-particle quantum systems [1].

The very first VQAs; the Variational Quantum Eigensolver (VQE) [3] and the Quantum Approximate Optimization Algorithm (QAOA) [4] paved the way for a new paradigm in quantum algorithm design, by considering the limitations of NISQ devices. Before these proposals, quantum algorithmic development was aimed at fault-tolerant quantum computers with error corrections capabilities. In contrast, VQAs seek to remove the stringent requirements for coherent evolution by formulating the computation as an approximate optimization problem. Here, a functional[2] in the space of heuristic functions defined by a variational quantum circuit is optimized. From this perspective variational quantum information processing and VQAs concern a strategy for finding extremums of *quantum functionals*, which are defined as mappings from the state space of a quantum system onto a scalar [5]. The flexibility in the definition of these heuristics allows the expansion of operability of these approaches within the NISQ regime, circumventing the need for quantum error correction.

The purpose of this chapter is to give a short introduction to the art of variational quantum algorithms. This chapter is brief but self-contained, and only a solid knowledge of quantum mechanics and the basics of quantum information is assumed. The reader is encouraged to go through Chapter 4 of Ref. [2] beforehand to better understand the implementation of VQAs on actual quantum devices. The chapter begins with a conceptual introduction to VQAs highlighting the motivation behind the use of variational methods on NISQ devices under Section 1.2. The subsequent Sections 1.3,1.4, 1.5 and 1.6 delve deeper into concepts introduced in 1.2. Finally a hands-on example is given under Section 1.7 where a detailed analysis is performed on the application of the variational quantum eigensolver to solve the electronic structure problem of H_2 molecule using Qiskit [6]. Basic familiarity with Hartree-Fock theory and post Hartree-Fock methods might prove useful in extending the hands-on example but it is not a requirement to understand the example. At the end of this chapter, the reader will hopefully feel comfortable with the use of VQAs to solve common optimization problems and have a clear overview of the theory to adjust various parts of the algorithm to suit one's needs.

[1]Even just storing arbitrary quantum states in classical computers is difficult because the amount of classical resources required to represent a given quantum state is related to the degree of entanglement of the state. This problem just gets worse when studying the dynamics. Check Exercise 4.46 in Ref. [2].

[2]A functional is simply a mapping that takes a function as input and returns a scalar as output.

1.2 Variational quantum algorithms

VQAs arise from three main concepts in variational quantum information processing as given below [5]:

1. *Approximate nature of the calculations:* Instead of guaranteeing an exact solution to the problem, VQAs offer approximate solutions. This helps us gain more flexibility in the number of resources required, in exchange for performance guarantees on the algorithm.

2. *The use of variational quantum circuits:* Variational quantum circuits (also called variational ansatz or variational forms or parametrized quantum circuits) are quantum circuits in which some of the unitary operations have tunable parameters obtained by parameterizing the corresponding quantum gates. These parameters are systematically updated when the algorithm is executed. To give an oversimplified analogy, the variational quantum circuit is like a stencil of the problem where the sizes of the holes can be varied. Computational problems which encode solutions as extreme values of a functional can be naturally described using the variational quantum circuits as heuristics. The flexibility in choosing variational circuits allows one to pick circuits that can operate within the capabilities of the NISQ devices. This operational robustness thus removes the requirement for error correction.

3. *Employing hybrid quantum-classical computing strategy:* The variational quantum circuits require an optimization routine to tune the parameters. This task is allocated to a classical computer. The optimization routine is carried out in a quantum-classical loop via classical communication. This feature motivates the idea of hybrid quantum-classical computing. The key idea of this approach is to 'divide and conquer' where the main computational task is divided into sub-tasks and allocated between the classical and quantum computers. By outsourcing pre-processing and post-processing involved to a classical computer, it is possible to create algorithms that exhibit advantage using fewer quantum resources than an algorithm that implements the entire task only on a quantum device.

These elements are combined to create a general strategy that can be applied to find approximate solutions to optimization problems. The main goal of a VQA is to find a function that extremizes the value of specific quantities of interest depending on the problem. For example, in the Variational Quantum Eigensolver (VQE) [3], the goal is to find a quantum circuit that prepares the ground state for a given Hamiltonian, usually the electronic structure problem. Another prominent example of a VQA is the Quantum Approximate Optimization Algorithm (QAOA) [4] where the goal is to find a quantum circuit that prepares the state which maximizes a problem specific cost function with encoded constraints.

1.2.1 Outline of the algorithm

In practice, VQAs are carried out following an iterative procedure on a Quantum Processing Unit (QPU) with continuous feedback to and from a classical central processing unit (CPU)[3]. A VQA consists of several modular components split between the QPU and CPU and can be readily connected, extended, and enhanced individually with developments in algorithms and quantum hardware. A full protocol consists of:

[3]This is similar to the Graphical Processing Unit (GPU) and the CPU working together in a normal computer. In this case, the QPU plays the role of a GPU.

1. *An **initial state** preparation:* we start by preparing the initial input state $|\varphi\rangle$;

2. *Application of a **parametrized unitary**:* a parametrized unitary $\hat{U}(\vec{\theta})$ is applied to the input state on the QPU preparing the output state $\hat{U}(\vec{\theta})\,|\varphi\rangle$. This parametrized unitary is defined by the choice of variational ansatz. It should correspond to a quantum circuit that cannot be efficiently computed using classical resources [4];

3. *A **cost function** and a method to estimate/evaluate it:* the value of the function is estimated using a quantum circuit. Ultimately, the estimation procedure reduces to performing measurements on the output state. To estimate the function to a given accuracy, Step 1 has to be repeated several times to prepare trial states to collect sufficient measurement statistics;

4. *A feedback loop with **classical optimization routine**:* based on the function estimate from Step 3. A classical optimization technique proposes a new set of values for the optimal variational parameters $\vec{\theta}'$.

Steps 1-4 are repeated until some problem-specific convergence criteria are satisfied. The following sections will deal with each of the components in-depth and a summarized diagram of the full protocol is shown in Fig. 1.1.

1.2.2 The variational paradigm and general quantum systems

The variational principle in quantum mechanics states that, for a given quantum state $|\Psi\rangle$ and an observable $\hat{O}$, the expectation value

$$\langle\hat{O}\rangle_{|\Psi\rangle} = \frac{\langle\Psi|\hat{O}|\Psi\rangle}{\langle\Psi|\Psi\rangle} \geq \lambda_0, \tag{1.1}$$

where λ_0 is the lowest eigenvalue of operator $\hat{O}$. The equality holds only when $\langle\hat{O}\rangle_{|\Psi\rangle} = \lambda_0$ and $|\Psi\rangle$ is said to be in the ground state. For a detailed discussion on the variational principle see Appendix 1.A.

In many instances, the eigenstates corresponding to the lowest few eigenvalues, and their properties are of primary interest. In physical systems studied in physics, this is usually the case since low energy states play a dominant role in determining the properties of a system at moderate temperatures, and in optimization problems, they often encode the optimal solution. The variational principle leads to a simple method to find the ground state of a quantum system with $\hat{O} = \hat{H}$: prepare several parametrized 'trial states' $|\Psi(\vec{\theta})\rangle$, measure the expectation value of the Hamiltonian, choose the state where the expectation value is minimized. This brings us to the topic below.

How are variational methods implemented in a quantum computer?

Let us consider a quantum system $\mathcal{S}$ composed of N qubits which will act as our quantum computer. Take a Hamiltonian $\hat{H}$ of a different system $\mathcal{Q}$ which need not have any relations to $\mathcal{S}$ other than acting on a Hilbert of space of less than N qubits. The goal is to find and study the system $\mathcal{Q}$ using $\mathcal{S}$. The Hamiltonian $\hat{H}$ naturally arises in the description of all physical systems. For example, this Hamiltonian could be derived from a physical system such as a collection of interacting spins or an interacting electronic system.

[4]One should remember that certain sets of quantum gates such as the Clifford group gates, compose quantum circuits that can be efficiently simulated (i.e in polynomial time) with classical resources according to the Gottesman-Knill theorem

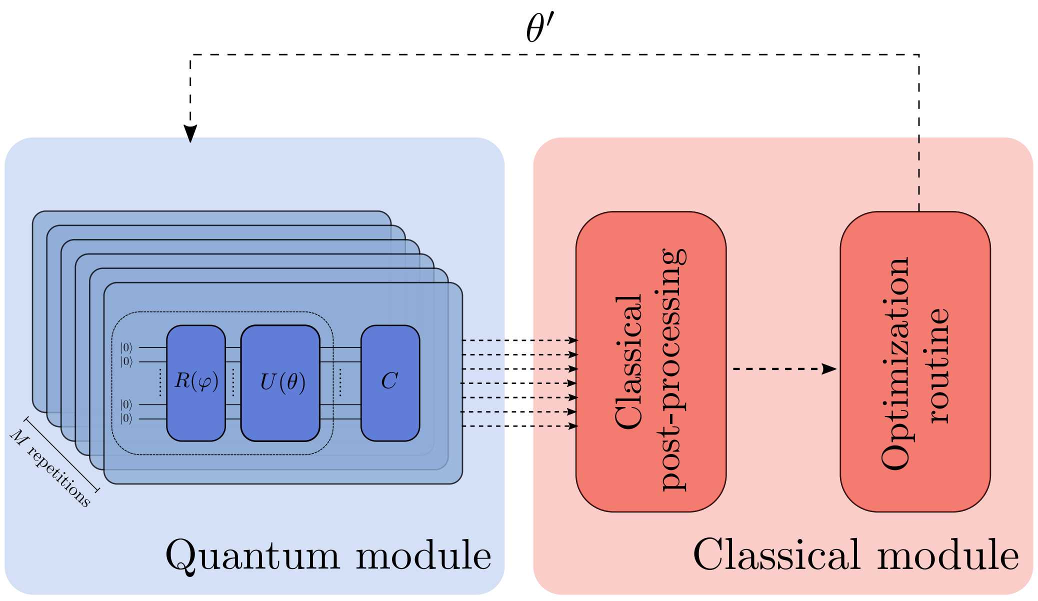


Fig. 1.1: **Control flow of a variational quantum algorithm using hybrid quantum-classical model [5]:** the algorithm is implemented on a quantum device by preparing the initial state $|\varphi\rangle = R(\varphi)\,|0\ldots0\rangle$ and executing the parametrized unitary $U(\vec{\theta})$. the circuit C implements the measurement scheme to estimate the cost function. To estimate the cost function, the state preparation and necessary measurements are repeated several times to accumulate sufficient statistics. The results are then post-processed on a classical computer which might include statistical modeling. Also, the circuit C could be replaced by a circuit that computes gradients of the cost function, which are required by some optimization routines. An iteration of the optimization algorithm generates a new candidate for the optimal variational parameters $\vec{\theta}'$. The cycle between the quantum and classical modules is repeated until convergence. Here, discontinuous arrows represent transfer of *classical* information.

Hamiltonians are not the only operators that can be measured on quantum devices; in general, any expectation value of an operator can be obtained by extending this method. Our attention is restricted to the class of operators whose expectation value can be measured efficiently on $\mathcal{S}$ and mapped to $\mathcal{Q}$ [7]. A sufficient condition for this property is that the operator $\hat{O}$ has a decomposition into a polynomial sum of simple operators[5] as

$$\hat{O} = \sum_{\alpha} c_\alpha \hat{O}_\alpha \tag{1.2}$$

where $\hat{O}$ acts on $\mathcal{Q}$, α runs over several terms polynomial in the size of the system, h_α is a constant coefficient, and each $\hat{O}_\alpha$ has a simple measurement prescription on $\mathcal{S}$. This will enable straightforward measurement of the expectation values of $\hat{O}$ on $\mathcal{Q}$ by weighted summation of projective measurements of $\mathcal{S}$. A simple and relevant example is the decomposition of a Hermitian operator into a weighted sum of Pauli strings (See Appendix 1.B).

After deciding the operator and the corresponding measurement scheme through polynomial expansion as given by Eq. (1.2), the system $\mathcal{S}$ needs to be prepared into the

[5]Almost all the Hamiltonians describing physical systems such as the Ising Model, Heisenberg Model, and electronic structure problem can be written as a polynomial number of terms, concerning the system size [8].

parametrized trial states. How they are prepared is the key element of VQAs and is discussed in detail under 1.3. To give a more general description, influence of the environment on the trial state preparation must be taken into account. This is better described by an ensemble given by a parametrized density matrix $\rho(\vec{\theta})$. In an ideal case where the state preparation is error free and a pure state is maintained, $\rho(\vec{\theta}) = |\Psi\vec{\theta}\rangle\langle\Psi\vec{\theta}|$. In density matrix formalism, the expectation value of an operator is given by

$$\langle\hat{O}\rangle_\rho = \mathrm{Tr}\left[\rho\hat{O}\right] \tag{1.3}$$

and the variational principle in Eq. (1.1) still holds. It can be rewritten as

$$\langle\hat{O}\rangle_\rho(\vec{\theta}) = \left\langle\hat{O}\right\rangle(\vec{\theta}) = \mathrm{Tr}\left[\rho(\vec{\theta})\hat{O}\right] \geq \lambda_0. \tag{1.4}$$

The fact that this principle still holds for mixed states is the main reason for the robustness of VQAs to errors and environmental influence. By finding the set of parameters $\vec{\theta}$ that minimizes $\langle\hat{O}\rangle$, one is in effect, finding a set of *experimental parameters* that are most likely to produce the ground state on average, potentially affecting a blind purification of the state of the system being produced [7].

1.3 Variational ansatz construction

Variational quantum circuit or the ansatz construction is a crucial aspect of VQA that dictates the success of the algorithm. It is a very active field of research and in most cases finding an optimal ansatz for a problem is difficult than finding a solution to the computational problem itself. The parameters are continuously varied to prepare a lot of different trial states. One can make this parametrization classical - i.e, given a list of parameters $\vec{\theta} \in \mathbb{R}^{N_{params}}$, we can prepare a different trial state $|\Psi\vec{\theta}\rangle$ for each $\vec{\theta}$. One might ask whether the classical parametrization makes evaluation of $\langle\hat{O}\rangle$ classically tractable, but this is avoided by using gate angles as parameters to control the quantum circuit. With this key idea in mind, now we are ready to define a variational ansatz formally according to [9].

Definition 1.3.1. *A* ***variational ansatz*** *on n_p parameters corresponds to a pair $(U, |\vec{0}\rangle)$, where U is smooth map from the parameter space $\vec{\theta} \in \mathbb{R}^{n_p}$ to the unitary operator $U(\vec{\theta})$ on $\mathbb{C}^{2^N}$, and $|\vec{0}\rangle \in \mathbb{C}^{2^N}$ is the initial state, which is acted on to generate the variational state $|\psi(\vec{\theta})\rangle = U(\vec{\theta})|\vec{0}\rangle$, with variational energy $E(\vec{\theta}) = \langle\psi(\vec{\theta})|H|\psi(\vec{\theta})\rangle$.*

Assuming that the vectors $\vec{\theta}$ are continuously defined, a variational ansatz can be imagined to sweep out a (possibly open) manifold in the Hilbert space of the states. In simpler terms, it explores some smooth space that lies within the set of allowed Hilbert space. Except for possible boundaries, the dimension of this manifold is precisely N_{params}. This points out a limit of the variational paradigm; to explore the entire 2^N-dimensional Hilbert space and guarantee that the ground state is found, $O(2^N)$ parameters [9] and a similar scaling of time are required [6]. So an exponential advantage is not expected. Instead, the 'art' of a VQA design is in choosing a manifold that contains points *sufficiently close* to the true ground state and the 'art' of application design is in choosing a problem such that 'sufficiently close' is achievable.

With the limitations in mind, the variational circuit must be chosen carefully. Because the choice of ansatz greatly affects the performance of a VQA. From the perspective of the

[6]This doesn't come off as a surprise, because performing such a task is known to be a QMA-hard problem [10].

problem, the ansatz influences both the closeness to the final solution and the convergence speed. Moreover, given the exponentially small fraction of the Hilbert space that will be explored, randomly generated ansatz cannot be expected to be 'good enough' due to the difficulties faced when optimizing such an arbitrary ansatz. Other than the computational concerns, NISQ hardware has to be taken into account as well. Due to the short coherence times of NISQ devices, deeper circuits are more susceptible to errors, and some gates can be costly to compose with hardware native gates. Accordingly, most of the ansätze developed to date are categorized either as more problem-inspired or more hardware efficient, depending on their structure and application. In the following subsections, a short explanation of arbitrary unitary evolution $U(\theta)$ and common constructions of both the ansatz types are presented.

1.3.1 Arbitrary unitary evolution

Understanding how arbitrary unitary evolutions $U(\theta)$ are executed in a quantum circuit is crucial for efficient ansatz construction. An arbitrary unitary evolution can be generated by an Hermitian operator $\hat{g}$ which defines an evolution in terms of the parameter θ,

$$U(\theta) = e^{-i\hat{g}\theta}. \tag{1.5}$$

From an abstract viewpoint, these evolutions can always be described as the time evolution of the corresponding quantum state, so that the generator $\hat{g}$ is often just the Hamiltonian $\hat{H}$[7]. Since any $\hat{H}$ can be written as a sum of Pauli strings according to Eq. (1.2) it is useful to analyze the case when $\hat{g}$ is a Pauli operator $\hat{P} \in \mathbb{P}^{\mathbb{N}}$. This makes

$$U_{\hat{P}}(\theta) = e^{-i\hat{P}\theta}. \tag{1.6}$$

Taylor expanding Eq. (1.6) and using the properties of the Pauli operators (see Appendix 1.B) yields,

$$\begin{aligned}
U_{\hat{P}}(\theta) &= \sum_k \frac{(-i\theta\hat{P})^k}{k!} \\
&= \sum_k \frac{(-i\theta\hat{P})^{2k}}{2k!} + \frac{(-i\theta\hat{P})^{2k+1}}{(2k+1)!} \\
&= \sum_k (-i)^{2k}\frac{\theta^{2k}}{2k!}P_i^{2k} + (-i)^{2k+1}\frac{\theta^{2k+1}}{(2k+1)!}P_i^{2k+1} \\
&= \sum_k (-1)^k\frac{\theta^{2k}}{2k!}I + (-i)\frac{\theta^{2k+1}}{(2k+1)!}P_i.
\end{aligned} \tag{1.7}$$

The terms are just the Taylor expansions of sin and cos and thus, $U(t)$ becomes a single-qubit rotation given by

$$U_{\hat{P}}(\theta) = \cos\theta I - i\sin\theta\hat{P}. \tag{1.8}$$

It is important to note that although $\hat{P}$ when acting as a unitary operator by itself does not generate any entanglement between the qubits, $U_{\hat{P}}(\theta)$ is an entangling gate! Not surprisingly, every possible operation on a quantum computer may be approximated by a series of $U_{\hat{P}}(\theta)$ for different operators P_i.

[7]Note, however, that the Hamiltonian generating evolution does not necessarily need to be the operator that describes the energy of the system of interest.

Suzuki-Trotter expansion

The Suzuki-Trotter expansion [11] is a general method to approximate a general, difficult to implement unitaries in the form of eq. (1.3.1) as a function of the θ parameter. This is done by decomposing the generator $\hat{g}$ into a sum of non-commuting operators $\{\hat{o}_k\}$ such that

$$\hat{g} = \sum_k a_k \hat{o}_k, \tag{1.9}$$

for some coefficients a_k. The operators $\hat{o}_k$ are chosen such that the evolution $e^{-i\hat{o}_k\theta}$ is easy to implement. An example of this is the expansion of the Hamiltonian $\hat{H}$ by Pauli strings as shown previously. The full evolution over θ can now be decomposed into $m \in \mathbb{Z}^+$ smaller, equally sized chunks

$$e^{-i\hat{g}\theta} = \lim_{m\to\infty} \left(\prod_k e^{-i\frac{a_k \hat{o}_k}{m}\theta} \right)^m . \tag{1.10}$$

When Pauli strings are used as in Eq. (1.6), this becomes a multi-qubit rotation which themselves can be decomposed into primitive one and two-qubit gates.

1.3.2 Problem-inspired ansatz construction

Historically, problem-inspired ansatz was the first to be proposed and implemented on actual hardware. Unitary Coupled Cluster (UCC) ansatz [12], Variational Hamiltonian Ansatz (VHA) [7, 13], factorized UCC and adaptive approaches are a few such problems inspired variational ansatz that stems from quantum chemistry and many-body physics. Within so-called problem-inspired approaches, unitary evolution is in the form of Eq. (1.3.1), with generators $\hat{g}$ derived from the system properties. Knowledge about the physics behind the problem Hamiltonian being Trotterized can reduce substantially the number of gates needed to implementing the Suzuki-Trotter method. Let us now look at a couple of problem-inspired ansatz examples.

Example 1: The Unitary Coupled-Cluster Ansatz

The UCC ansatz is widely used in quantum chemistry problems where the goal is to simulate a fermionic molecular Hamiltonian and obtain the ground state energy. The idea of implementing UCC on quantum computers arose from the observation that the UCC ansatz, which adds quantum correlations to the Hartree-Fock approximation, is inefficient to represent on a classical device [14]. It was first realized on a photonic processor by Peruzzo and his group in 2014 [3].

Typically, we expect the ground state of an interacting system to be a linear combination of low-energy excitations of the corresponding non-interacting problem. In the electronic structure problem this corresponds to excitations of a few particles from the Hartree-Fock state $|\Psi_{HF}\rangle$ of the Hamiltonian $\hat{H}$. The UCC ansatz proposes a candidate for a ground state based on excitations generated by the *cluster operator* $\hat{T}$ from $|\Psi_{HF}\rangle$ to low-lying non-interacting higher energy states as $e^{\hat{T}(\theta)-\hat{T}^\dagger(\theta)}|\Psi_{HF}\rangle$ [12]. The cluster operator $\hat{T}$ which captures the excitation dynamics are given by[8],

$$\hat{T} = \hat{T}^{(1)} + \hat{T}^{(2)} + \hat{T}^{(3)} + \ldots = \sum_k \hat{T}^{(k)} \qquad \text{(Cluster operator)}, \qquad (1.11a)$$

$$\hat{T}^{(1)} = \sum_{\substack{i \,\in\, \text{empty} \\ k \,\in\, \text{filled}}} \theta^i_k \; \hat{c}^\dagger_i \hat{c}_k \qquad \text{(Single excitations)}, \qquad (1.11b)$$

$$\hat{T}^{(1)} = \sum_{\substack{i,j \,\in\, \text{empty} \\ k,l \,\in\, \text{filled}}} \theta^{i,j}_{k,l} \; \hat{c}^\dagger_i \hat{c}^\dagger_j \hat{c}_k \hat{c}_l \qquad \text{(Double excitations)}. \qquad (1.11c)$$

The operator $\hat{c}_i$ refers to the fermionic annihilation operator of the i-th Hartree-Fock orbital and the sets *empty* and *filled* refer to the corresponding occupied and unoccupied Hartree-Fock orbitals. Here, θ^i_k and $\theta^{i,j}_{k,l}$ are the free parameters of the problem. Due to the decreasing importance of higher $\hat{T}^{(k)}$, the series is usually truncated after the second or third term. The ansatz is termed UCCSD or UCCSDT, respectively referring to the inclusion of single, double, and triple excitations from the Hartree-Fock ground state. To implement this ansatz on a quantum computer, second quantization methods are used to map the fermionic operators to the qubit operators[9] resulting in an ansatz of the form in Eq. (1.10). It is later on converted to a parametrized ansatz via the Trotterization.

Example 2: The Quantum Alternating Operator Ansatz

The Quantum Alternating Operator Ansatz was originally introduced as a part of the QAOA algorithm [4], and later generalized as a standalone variational ansatz. One can imagine the QAOA as a Trotterized version of quantum annealing where the order p of the Trotterization determines the precision of the solution. Indeed, the adiabatic evolution can be obtained in the limit of $p \to \infty$. The goal of QAOA is to map an input state $|\psi_0\rangle$, to the ground state of the problem Hamiltonian $\hat{H}_C$ by applying a unitary $e^{-i\gamma_j \hat{H}_C}$ and a mixer unitary $e^{-i\beta_j \hat{H}_M}$ sequentially. That is,

$$U(\vec{\gamma}, \vec{\beta}) = \prod_{j=1}^{p} e^{-i\beta_j \hat{H}_M} \; e^{-i\gamma_j \hat{H}_C}, \qquad (1.12)$$

where $\vec{\theta} = (\vec{\gamma}, \vec{\beta}))$. A key strength of QAOA is the fact that the feasible subspace for certain problems is smaller than the complete Hilbert space, and this restriction results in a better performing algorithm.

1.3.3 Hardware-efficient ansatz construction

Problem-inspired ansatz has been shown computationally, to ensure faster convergence to a satisfying solution. However, the deeper circuits associated with Trotterization can be challenging to realize experimentally. The hardware-efficient ansatz is a generic name used for variational ansatz that is aimed at NISQ with a focus on circuit depth reduction. It was introduced in 2017 by Kandala and Mezzacapo as an experiment on a 7 qubit superconducting device [15]. The key idea is to generate random parametrized quantum circuits only using the better-performing native gates of the underlying hardware with a particular qubit topology. This avoids the circuit depth overhead arising from translating

[8]Given in *physicist* notation. Computational quantum chemistry softwares use the *chemist* notation in which the ordering of the operators change: physicist: $ijkl \to$ chemist: $iklj$

[9]There are many variants of the UCC ansatz, with some of them reducing the depth of the quantum circuit by considering more efficient methods for compiling the fermionic operators using group theory.

an arbitrary unitary into the native gates. By doing so, one expects to reduce errors during the computation while being able to generate meaningful parametrized trial states. Let us now see, how one can construct such an ansatz with basic single-qubit gates and two-qubit entangling gates according to the method presented in [15].

The circuit is constructed from *blocks* of single-qubit gates and entangling gates applied to multiple qubits simultaneously. A single sequence of application of a block is called a *layer*. The ansatz circuit generally has multiple such layers and it enables exploration of a larger portion of the Hilbert space. The unitary of a hardware-efficient ansatz with N_l is given by

$$U(\vec{\theta}) = \prod_{k=1}^{N_l} U_k(\theta_k) W_k, \tag{1.13}$$

where $\vec{\theta} = (\theta_1, \dots, \theta_{N_l})$ are the variational parameters, unitary $U_k(\theta_k) = e^{-i\theta_k \hat{O}_k}$ is the unitary generated by the Hermitian operator $\hat{O}_k$, and W_k represents the non-parametrized quantum gates. Typically $\hat{O}_k$ are Pauli strings acting locally on each qubit. In those cases, U_k becomes a product of combinations of single-qubit rotations, each one defined as in Eq. (1.8). W_k is the entangling unitary constructed from the native gate set. For example for superconducting qubit architectures, CNOT or CZ gates can be used and in trapped ion platforms, XX gates can be used [16, 17]. The choice gate set, qubit topology, and the ordering influence the subspace of the Hilbert space covered by the ansatz and how fast it converges to a solution for a specific problem.

In an ideal case, one would expect to perform entangling gates between all pairs of qubits. However, the reality is different due to the limited qubit connectivity which varies among different hardware platforms, thus limiting the ability to perform two-qubit entangling gates. One main advantage of this ansatz is its versatility and problem-agnostic nature. It is especially useful to study Hamiltonians that are similar to the interactions in the hardware. On the other hand, the problem-agnostic nature also leads to problems in optimization when parameters are initialized randomly.

1.4 Cost function

After constructing a variational ansatz, the focus should be on finding a way to extract useful classical information from it. Therefore, the next step in a VQA is to encode the problem into an objective function, sometimes called the cost function. From Section 1.2.2, cost function is just the expectation value of the operator being measured. To reformulate it appropriately, the standard definition of a VQA cost function, denoted $f_{\hat{O}}(\vec{\theta})$ is the expectation value of an observable $\hat{O} \in \mathrm{Herm}(\mathbb{C}^{2^N})$, that is,

$$f_{\hat{O}}(\vec{\theta}) = \langle \psi(\vec{\theta}) | \hat{O} | \psi(\vec{\theta}) \rangle \tag{1.14}$$

where normalization of the wavefunction is assumed. Let us look at a couple of examples of cost functions.

Example 1: Approximating the ground state energy of a physical system

Information about the dynamics of a system is encoded in its Hamiltonian $\hat{H}$, and it naturally arises in the description of all physical systems. In physics, one is often interested in computing the ground state energy E_0 of a Hamiltonian H given by,

$$E_0 = \min_{|\psi\rangle} \frac{\langle \psi | \hat{H} | \psi \rangle}{\langle \psi | \psi \rangle}. \tag{1.15}$$

In this case it is apparent that the corresponding cost function is given by

$$f_{\hat{H}}(\vec{\theta}) = \langle\psi(\vec{\theta})|\hat{H}|\psi(\vec{\theta})\rangle\,. \tag{1.16}$$

The UCCSD ansatz from Section 1.3.2 and the hardware efficient ansatz from Section 1.3.3 are used commonly to solve problems of this kind. This will also be the example problem analysed under Section 1.7.

Example 2: Approximating optimization problems

For problems unrelated to physical systems, encoding into an equivalent qubit Hamiltonian is still possible, thereby opening a path to solving them on a quantum computer. To solve a combinatorial optimization problem using a quantum algorithm, the problem is first encoded onto a quantum system. One of the most widely used models in physics, that is also used to represent optimization problems, is the Ising model. A quantum version of it can be obtained easily by replacing the spin variables with Pauli Z-operators

$$\hat{H}_C \equiv \hat{H}(\sigma_1^z, \hat{\ldots}, \sigma_n^z) = -\sum_{1\leq i<j\leq n} J_{ij}\sigma_i^z\sigma_j^z - \sum_{i=1}^{n} h_i\sigma_i^z \tag{1.17}$$

where J_{ij} and h_i are real numbers, and σ_i^z refers to the Pauli Z-operator acting on the i^{th} qubit. Several NP-complete optimization problems and even many NP-hard problems can naturally be expressed as a problem of finding the ground state, or minimum energy configuration, of a quantum Ising Hamiltonian by choosing appropriate values for J_{ij} and h_i [18]. The QAOA ansatz from Section 1.3.2 is usually the preferred ansatz for solving these combinatorial optimization problems and one can read such an example of QAOA applied to a tail-gate assignment problem in Ref. [19].

To sum up the idea, a cost function defines a hyper-surface called the cost landscape that maps the trainable parameters to real numbers. The task of the optimizer is to traverse the cost landscape and find the global minima. The choice of optimizer depends on how complicated the landscape is, as discussed in detail under Section 1.6. Therefore, the choice of a good objective function is also crucial to ensure convergence to the solution in addition to the variational ansatz choice. A good objective function of a problem should meet the following criteria.

1. *Faithfulness:* The optimum of the cost function must correspond to the solution of the problem

2. *Efficient estimation:* It should be possible to estimate the cost function efficiently by performing a finite number of measurements with classical post-processing.

3. *Operational meaningfulness:* It's useful when the value of the cost function reflects the quality of a solution (e.g. lower cost means better solution).

4. *Trainability:* It must be possible to efficiently optimize the parameters to meet convergence criteria

Even though the properties given above sound trivial, they are difficult to satisfy when running an actual experiment on NISQ devices with constraints on circuit depth and ancilla requirements.

1.5 Measurement scheme

Let us now address the question of how to evaluate the VQA cost function defined in Eq. (1.14). Using Eq. (1.2) and Theorem 1.B.1, most interesting observables can be written as a linear combination of a polynomial number of Pauli strings with real coefficients, i.e.,

$$\hat{O} = \sum_{i=1}^{\text{poly}(N)} c_i \sigma_i^{\alpha} \tag{1.18}$$

where $\alpha = x, y, z$ identify the Pauli spin operator. For example, most physically motivated Hamiltonians are k-local. That is, each of the Pauli strings in the sum acts non-trivially on at most k qubits. Examples of k-local systems include Ising and Heisenberg spin systems, van der Waals gases, strong and weak interactions, and lattice gauge theories. In fact, any system that is consistent with special and general relativity evolves according to local interactions [20]. In a k-local system there are $\binom{n}{k}$ possible subsets of k qubits. Therefore the number of Pauli strings that appear in the decomposition of a k-local Hamiltonian is bounded by a polynomial. When Eq. (1.18) is encoded into a cost function as in Eq. (1.14) we find that

$$\begin{aligned} f_{\hat{O}}(\vec{\theta}) &= \langle\psi(\vec{\theta})|\hat{O}|\psi(\vec{\theta})\rangle \\ &= \sum_{i=1}^{\text{poly}(N)} \langle\psi(\vec{\theta})|c_i\sigma_i^{\alpha}|\psi(\vec{\theta})\rangle \\ &= \sum_{i=1}^{\text{poly}(N)} c_i \langle\psi(\vec{\theta})|\sigma_i^{\alpha}|\psi(\vec{\theta})\rangle . \end{aligned} \tag{1.19}$$

From this, we see that the evaluation of the cost function boils down to evaluating the expectation value of a weighted sum of Pauli strings assuming that the values of c_i are known beforehand. The expectation value of a tensor product of an arbitrary number of Pauli operators can be estimated by measuring each of the qubits locally [2]. This is an operation that requires a coherence time of $O(1)$ (i.e. incurring a constant cost in time) under the assumption that parallel qubit rotations and readouts are possible [3].

1.5.1 Precision and Chebyshev's inequality

Suppose we wish to estimate $E = \langle\psi(\vec{\theta})|P|\psi(\vec{\theta})\rangle$, for some Pauli string $P \in \mathbb{P}^N$, up to a precision $\varepsilon > 0$. In order to achieve that, let

$$P = \sum_{k=1}^{2^N} \lambda_k |\phi_k\rangle\langle\phi_k| \tag{1.20}$$

denote the spectral decomposition of P. Here E can be taken as the expected value of a random variable X with possible outcomes $\{\lambda_i\}_{i=1}^{2^N}$ with

$$p(X = \lambda_k) = |\langle\phi_k|\psi(\vec{0})\rangle|^2. \tag{1.21}$$

That is, X is the random variable representing the outcome of measuring operator $\hat{O}$ on the state $|\psi(\vec{\theta})\rangle$. *Chebyshev's inequality* is a well known result in probability theory which states that if we take M copies of X, denoted by $X_1, \ldots, X_M$, then

$$p\left(\left|\frac{\sum_{i=1}^{M} X_i}{M} - E\right| \geq \varepsilon\right) \leq \frac{\sigma^2}{M\varepsilon^2}, \tag{1.22}$$

where σ denotes the variance of X. Since the variance of X can be assumed to be bounded above by a constant, from Eq. (1.22) it can be deduced that

$$M \sim \frac{1}{\varepsilon^2} \tag{1.23}$$

measurements must be performed, to obtain, with high probability, an estimate of E to within additive precision ε. The limitations in current NISQ hardware has severe implications on how well the above strategy will allow one to estimate the energies E, as the measurement outcomes will become noisy. This inherent statistical nature of the outcomes will influence the optimization strategy. This is discussed in Section 1.6.

1.5.2 Measurement scheme optimization

The strategy presented here is a very naive way of estimating the cost function by sampling every Pauli string and using the outcome statistics to estimate the corresponding expectation value. As mentioned before, the key advantage of this approach is that the coherence time to make a single measurement after preparing the state is $O(1)$. Conversely, the disadvantage of this approach is the scaling of the total number of operations, as a function of the desired precision ε as $O(\varepsilon^{-2})$ [3]. Moreover, this scaling also reflects the number of state preparation repetitions required. This has led to an active area of research in measurement optimization strategies. One example of such a technique would be to use the commutation properties of Pauli strings and make simultaneous measurements. The problem is then to divide the Pauli strings into a small number of commuting subsets. It turns out that this problem is equivalent to the clique cover problem, which is known to be NP-complete [21]. Researchers have proposed new methods using approximation algorithms for the clique cover problem as a grouping, with various degrees of success. By speeding up the measurement scheme with new techniques, it is possible to realize faster algorithms with efficient use of quantum resources.

1.6 Optimization routine

When choosing the variational ansatz, the hope is that it explores a region of Hilbert space in which the minimum $\min_{\vec{\theta}} f_{\hat{O}}(\vec{\theta})$ is near the actual ground state energy of $\hat{O}$. Thus, after making the ansatz choice, the challenge that remains is to find the $\vec{\theta}$ that minimizes the cost function $f_{\hat{O}}$. To this end, a classical optimization routine is employed. Numerical optimization is a vast field, and therefore identifying and adapting existing optimization techniques that are suited to quantum variational approaches requires considerable effort because the success of a VQA depends on the reliability and efficiency of the classical optimization method used.

The following should be kept in mind when choosing an optimizer.

1. Due to short coherence times in the NISQ era, complicated analytical gradient circuits cannot be implemented

2. The optimizer should be resilient to noisy data and precision on objective function evaluation that is limited by the number of shots in the measurement.

3. Objective function evaluations take a long time, which means shots frugal optimizers are favored.

Which classical optimization routine works best with VQAs is an open question. Broadly speaking, the optimization algorithms used in VQAs can be split into two groups:

1. *Gradient-based approaches:* As the name suggests, gradient-based methods optimize the cost function $f_{\hat{O}}(\vec{\theta})$ via its gradient $\nabla_{\vec{\theta}} f_{\hat{O}}(\vec{\theta})$. The calculated gradient will indicate the direction in which the cost function shows the greatest change. By going in the opposite direction to that, one can find the (local) minima.
   Advantages:
   - Convergence properties are well established in research.
   - When the cost function landscape is simple and smooth, it works incredibly well.

   Disadvantages:
   - Time taken for convergence is large when the gradient evaluation is expensive.
   - Unreliable under noisy gradient evaluations.
   - Suffers in the presence of exploding or vanishing gradients.

   Weighing the pros and cons, gradient-based optimizers are a good choice when there is previous knowledge of the cost landscape and it is known to be smooth to ensure efficient gradient evaluations.

2. *Gradient-free approaches:* Gradient-free methods rely on evaluating the function at many points in the cost landscape and inferring a minimum from these evaluations.
   Advantages:
   - Works well for complicated energy landscapes that aren't smooth as they don't require the ability to evaluate gradients efficiently.
   - Works decently even in the presence of vanishing and exploding gradients.

   Disadvantages:
   - Requires several cost function evaluations in general.
   - Doesn't converge as fast as gradient-based approaches in smooth cost landscapes.

   Gradient-free approaches are suitable when evaluating the gradient is very expensive and when there is no prior knowledge about the cost landscape.

Effect of hardware noise

As a final remark, it is important to mention that VQAs are somewhat resilient to noise. This is due to the ability to counteract errors in the evaluation by varying the parameter. That is, if an error slightly moves the target minimum, it is still possible to find it by changing the parameters accordingly as explained under Section 1.2.2. The cost function is statistical by design. Unlike a typical optimization routine in classical computing, the inherently stochastic environment due to the limited number of measurements, hardware noise, and the presence of barren plateaus (i.e. exponentially vanishing gradients) makes the choice of a suitable classical optimizer difficult. This has lead to active research in the development of quantum-aware optimizers which are believed to deal best with the quantum noise in expectation value evaluations.

Architecture	Molecule	qubits	Ansatz	Optimization routine	Computed state	ref
Photonic chip	HeH^+	2	hardware-specific parametrized ansatz	Nelder- Mead	ground state	[3]
Transmon qubits	H_2	2	UCC	grid-scan, local optimization	ground state	
Transmon qubits	H_2	2	hardware-efficient ansatz	SPSA	ground state	[15]
Transmon qubits	LiH	4	hardware-efficient ansatz	SPSA	ground state	[15]
Transmon qubits	BeH_2	6	hardware-efficient ansatz	SPSA	ground state	[15]
Ca^+ ion trap	H_2	2	UCC	grid-scan, local optimization	ground state	[22]
Ca^+ ion trap processor	LiH	3	approximate UCC	grid-scan, local optimization	ground state	[22]
Transmon qubits	H_2	2	hardware-specific parametrized ansatz	PSO	ground & excited states	[23]
Si photonic chip	two chlorophyll units in 18-mer ring of LHII complex	2	parametrized Hamiltonian ansatz with truncation scheme	PSO	ground & excited states	[24]
Transmon qubits via cloud	deuteron	2-3	UCC	grid-scan	ground state	[25]

Table 1.1: **Representative experimental demonstrations of the VQE algorithm using various quantum computer architectures** [26]. Abbreviations: SPSA, simultaneous perturbation stochastic approximation; PSO, particle swarm optimization.

1.7 Qiskit hands-on example: Molecular quantum simulation using the VQE

The VQE is a popular VQA candidate that has already entered the experimental regime as can be seen in Table 1.1. It is an application of the time-independent variational principle, where the ansatz preparation as well as cost function estimation are both implemented on a quantum processor. As it's a VQA, the modular structure discussed in section 1.2.1 makes up the basis of VQE. The specific details of the VQE can be broken down into iterative steps:

1. Preparation of the trial states by applying the chosen parametrized unitaries.
2. Determination of the expectation value of every term in the qubit representation of the Hamiltonian via an efficient partial tomography. A large number of executions of quantum circuits is required to evaluate the expectation values. This step consists of three nested iterations:
   (a) *Outer iteration*: to systematically update parameters $\vec{\theta}$ of quantum state $|\Psi(\vec{\theta})\rangle$.
   (b) *Middle iteration*: to evaluate the expectations value by calculating the weighted sum of Pauli strings.
   (c) *Inner iterations*: to evaluate the expectation value of a Pauli strings through sampling.
3. Parameter update using a classical optimization routine
4. Repetition of steps 2 and 3 until convergence criteria are satisfied.

This section is based on Ref. [27] and Ref. [15]. Let us now apply the theory from previous sections to solve the problem under study in Example 1 from 1.4. The goal is to analyze the energy landscape and estimate the ground state energy of H_2 molecule using VQE. Fig. 1.2 summarizes the important steps involved in solving this problem.

Before we begin: The calculations and simulations given in this section are performed using Qiskit[10]. Since H_2 is a small molecule, the problem is classically tractable and good classical benchmarking results are already available. Hence the reader will be able to replicate the results on their own with the limited computational resources of a personal computer.

1.7.1 Classical pre-processing

As discussed in the Introduction, the focus will be on the tools developed to solve the electronic structure problem. Due to the hybrid approach, choices made during the classical pre-processing steps would also influence the overall performance and quantum resource requirements of the algorithm as discussed in 1.4. So let us begin by formulating our problem and translating it into the language of second quantization. This step is classically tractable after some approximations and for the case of the electronic structure problem, there are several computational chemistry packages readily available. In this example, Python-based Simulations of Chemistry Framework (PySCF) [29] is used to obtain the molecular drivers required as input. The classical pre-processing stage might seem at first glance very simple with PySCF and Qiskit in terms of the number of lines of code required, but it is rich in physics. The aim of this section is to produce a self-contained

[10]At the time of writing, version 0.24.0

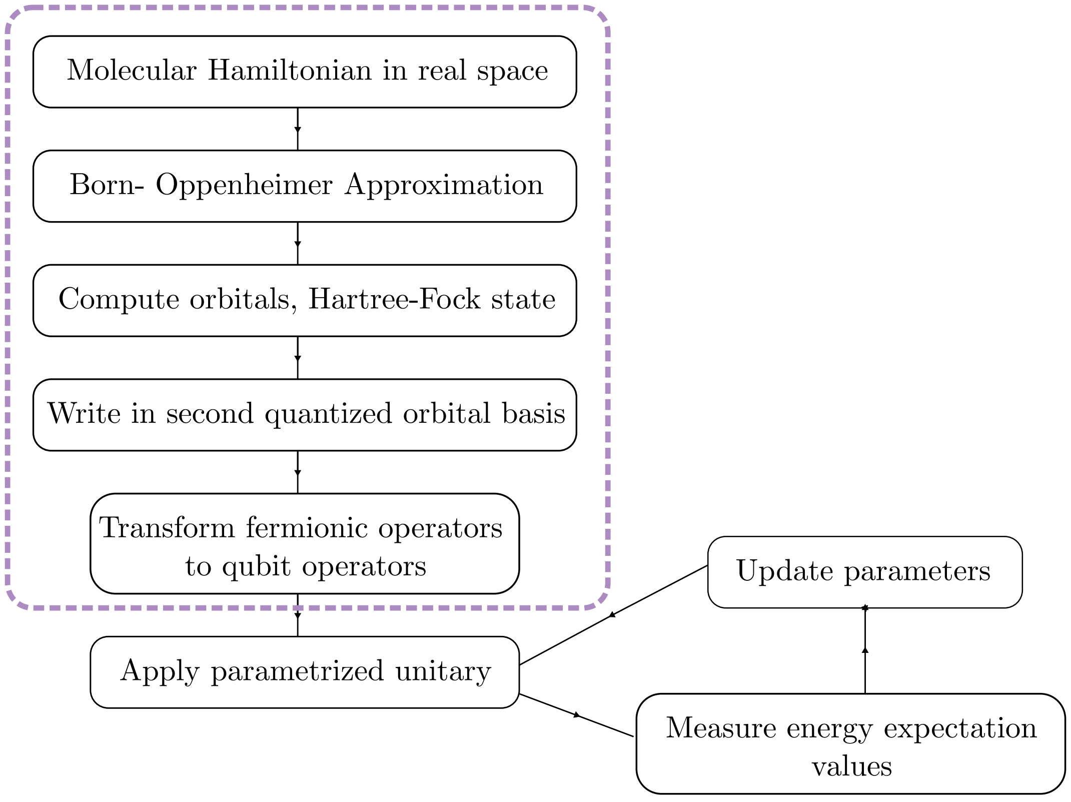

Molecular Hamiltonian in real space
Born- Oppenheimer Approximation
Compute orbitals, Hartree-Fock state
Write in second quantized orbital basis
Transform fermionic operators to qubit operators
Apply parametrized unitary
Update parameters
Measure energy expectation values

Fig. 1.2: **A flow chart adapted from [28] describing the steps required to compute molecular energies using VQE.** Here, steps inside the dashed box represents classical preparation.

short summary of the essential knowledge required for quantum computation for molecular quantum mechanics, and the interested reader is referred to Ref. [30] and [31] for further information.

What is the electronic structure problem?

The fundamental goal in electronic-structure problems is obvious: solve for the ground-state energy of many-body interacting fermionic Hamiltonians. The Hamiltonian of a molecule consisting of N_α nuclei and N_e electrons is

$$\begin{aligned} H = &-\sum_i \frac{\hbar^2}{2m_e}\nabla_i^2 - \sum_\alpha \frac{\hbar^2}{2M_\alpha}\nabla_\alpha^2 - \sum_{i,\alpha}\frac{e^2}{4\pi\epsilon_0}\frac{Z_\alpha}{|\vec{r}_i - \vec{R}_\alpha|} \\ &+ \frac{1}{2}\sum_{i\neq j}\frac{e^2}{4\pi\epsilon_0}\frac{1}{|\vec{r}_i - \vec{r}_j|} + \frac{1}{2}\sum_{\alpha\neq\beta}\frac{e^2}{4\pi\epsilon_0}\frac{Z_\alpha Z_\beta}{|\vec{R}_\alpha - \vec{R}_\beta|}, \end{aligned} \tag{1.24}$$

where M_α, $\vec{R}_\alpha$ and Z_α represent the mass, position and the atomic number of the α^{th} nucleus and $\vec{r}_i$ is the position of the i^{th} electron. For conciseness, switching to atomic units and substituting $M'_\alpha = \frac{M_\alpha}{m_e}$ yields

$$H = -\sum_i \frac{\nabla_i^2}{2} - \sum_\alpha \frac{\nabla_\alpha^2}{2M'_\alpha} - \sum_{i,\alpha}\frac{Z_\alpha}{|\vec{r}_i - \vec{R}_\alpha|} + \frac{1}{2}\sum_{i\neq j}\frac{1}{|\vec{r}_i - \vec{r}_j|} + \frac{1}{2}\sum_{\alpha\neq\beta}\frac{Z_\alpha Z_\beta}{|\vec{R}_\alpha - \vec{R}_\beta|}. \tag{1.25}$$

Since our primary interest lies in solving this for molecules, where a nucleon is over 1000 times heavier than the electron, the Born-Oppenheimer approximation is applied. As a result, the problem is simplified for a given nuclear configuration where one needs to solve only an electronic Hamiltonian given by

$$H_{\text{elec}} = -\sum_i \frac{\nabla_i^2}{2} - \sum_{i,\alpha} \frac{Z_\alpha}{|\vec{r}_i - \vec{R}_\alpha|} + \frac{1}{2}\sum_{i\neq j} \frac{1}{|\vec{r}_i - \vec{r}_j|}. \tag{1.26}$$

The aim is to find energy eigenstates of H_{elec} and the corresponding energy eigenvalues to an accuracy of at least 1.6×10^{-3} hartree[11], known as the *chemical accuracy.* Knowing the energies up to chemical accuracy enables reaction rate calculations within an order of magnitude. This is one of the promises that one-day quantum computers hope to deliver.

All about quantization, basis sets and mapping

The state under consideration is an electronic wavefunction. This means the anti-symmetric nature of the wavefunction as a consequence of Pauli's exclusion principle should be taken into account. The anti-symmetrization can be accounted for in either *first* or *second* quantization methods. The names are largely due to historical reasons, but it's important to remember that the choice affects how the physical system is simulated on a quantum computer. First quantization methods retain the anti-symmetric nature of the wavefunctions explicitly whereas second quantization maintains it through correct exchange statistics and the properties of the field operators. The key difference in the representations becomes more prominent in the context of fermion to qubit mappings. Whether the simulation is done in first quantization or second quantization is distinct from whether a *grid-based* method or *basis set* method is used. The details of grid-based methods and basis set methods are beyond the scope of this chapter. Therefore a summary of the different methods are given in Table 1.3. In the case of quantum simulation, the number of basis functions in the basis set and the quantization method used determines the number of gate operations required. For NISQ devices this leads to a compromise between the accuracy obtainable and the number of basis functions due to the limited qubit numbers. The number of qubits required to store the wavefunction in each quantization method is summarized in Table 1.2.

Quantization	Number of qubits
First quantization, basis sets method	$N[\log_2(M)]$
Second quantization, basis sets method	M

Table 1.2: **The number of qubits required to store the wavefunction using different quantization methods.** N is the number of simulated electrons in the problem, M is the number of spin orbitals used in the basis set. The number of qubits can often be further reduced [27].

[11]1 hartree $= \frac{e^2}{4\pi\epsilon_0 a_0} = 27.211$ eV

Grid-based methods	Basis set methods
• Wavefunction is evaluated on every point in a spatial grid – ***First quantization:*** Wavefunction in position representation is explicitly anti-symmetrized. The N-electron wavefunction is given by $\|\Psi\rangle = \int_{\vec{x}_1,\ldots,\vec{x}_N} \psi(\vec{x}_1,\ldots,\vec{x}_N)\mathcal{A}(\|\vec{x}_1,\ldots,\vec{x}_N\rangle)d\vec{x}_1\ldots d\vec{x}_N,$ where $\psi(\vec{x}_1,\ldots,\vec{x}_N) = \langle\vec{x}_1,\ldots,\vec{x}_N\|\Psi\rangle$, $\mathcal{A}$ denotes anti-symmetrization and $\vec{x}_i = (\vec{r}_i,\sigma_i) = (x_i,y_i,z_i,\sigma_i)$ gives the position and spin of the i^{th} electron. – ***Second quantization:*** Real space grid is described by a set of δ functions $\delta(r-r_i)$ positioned at r_i which serve as basis. Creation operator becomes $\hat{c}^{\dagger}_{x_i,y_i,z_i,\sigma}$ which creates an electron with spin σ at grid point (x_i,y_i,z_i) in 3 dimensional space. • Memory required to store wavefunction scales exponentially with the size of the system. • Useful when simulating systems where the Born-Oppenheimer approximation is not applicable • Grid-based methods in second quantization has not yet been used in any quantum or classical simulations [27]	• Hamiltonian is projected onto M basis functions $\{\phi_p(\vec{x}_i)\}$, $\vec{x}_i = (\vec{r}_i,\sigma_i)$ which approximate electron spin orbitals. • The N electron wavefunction is written as a Slater determinant given by $\psi(\vec{x}_i,\ldots,\vec{x}_{N-1}) = \frac{1}{\sqrt{N!}}\begin{vmatrix} \phi_0(\vec{x}_0) & \ldots & \phi_{M-1}(\vec{x}_0) \\ . & . & . \\ . & . & . \\ \phi_0(\vec{x}_{N-1}) & \ldots & \phi_{M-1}(\vec{x}_{N-1}), \end{vmatrix}$ and in occupation number representation in Fock space as $\psi(\vec{x}_i,\ldots,\vec{x}_{N-1}) = \|f\rangle = \|f_{M-1},\ldots f_p,\ldots f_0\rangle$ • Resource requirement is reduced by exploiting the knowledge on symmetry and the general spatial form of the orbital functions. • The number of basis functions M used determines the number of qubits and gate operations required to solve a problem • Known as the *Galerkin discretization*, which ensures that the energy converges to the true value from above as $M \to \infty$ • Most common basis sets are the Slater-type orbitals (STOs), Gaussian-type orbitals (GTOs) and plane wave basis sets.

Table 1.3: **Overview of the grid-based methods and basis set methods used in computational many body problems.**

Once the quantization step is over, we proceed to the mapping step. Here the operators that act on *indistinguishable* fermions must be mapped to operators that act on *distinguishable* qubits. To this end, an encoding method, which is just a map from the fermionic Fock space to the qubit Hilbert space is applied. This will lead to every fermionic state being represented by a qubit state.

There are multiple second quantization encoding methods, which are described next.

1. *Jordan - Wigner encoding (JW)*

   - Occupation number of the spin orbital is stored in the $|0\rangle$ and $|1\rangle$ state of a qubit (unoccupied and occupied respectively).
   - The creation operator $\hat{c}_j^\dagger$ of the electronic orbital is associated with a qubit j via
   $$\hat{c}_j^\dagger \leftrightarrow \underbrace{\hat{Z}_1 \otimes \ldots \otimes \hat{Z}_{j-1}}_{\text{application of a phase}} \otimes \hat{Z}_j^\dagger \otimes \hat{I}, \tag{1.27}$$
   and has two operations: changing the occupation number locally and applying a phase according to parity.
   - Stores the parity non-locally and occupation number locally.
   - The large weight of the parity-counting phase can be costly for certain quantum simulation algorithms.

2. *Parity encoding (P)* [32]

   - It is the dual version of the JW mapping.
   - While JW stores the occupation number of each spin orbital in each qubit, the parity mapping stores the parity in each qubit.
   - Stores the parity locally and occupation number non-locally.
   - Parity operators are low weight while the occupation operators become high weight.

3. *Bravyi-Kitaev encoding (BK)* [33]

   - Combines the advantages of JW and P encding methods through non-trial transformation.
   - Applies only for systems with a spin orbital number equal to a power of 2
   - BK mapping allows for simulation of interacting fermion systems on a graph where each node corresponds to a local fermionic mode and each edge represents a pairwise interaction [26].

1.7.2 Simulating the Hydrogen molecule

Now that we have a clear understanding both the quantum and classical parts of the algorithm it's time to use that to solve for the ground state energy of H_2 molecule using use basis set method in second quantization.

Choose the basis set functions. An ideal choice of basis set function should be one that captures the physics of the problem, such that a good representation can be achieved using the smallest possible set. A good basis set [26]

- allows systematic and fast converging extrapolation to the basis set limit

- has a mathematical form which facilitates the evaluation of molecular integrals,
- should not only describe energies but also other useful properties.

Let us first consider the sets Slater-type orbitals with 3 Gaussians (STO-nG with n=3), split valence basis 6-31G and correlation consistent polarized valence double-ζ (cc-PVDZ) encoded using JW transformation and observe how the number of qubits vary.

```python
import matplotlib.pyplot as plt
import numpy as np
import pylab
import tikzplotlib
from qiskit import Aer
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.aqua.algorithms import NumPyEigensolver
from qiskit.aqua.algorithms import VQE
from qiskit.aqua.components.initial_states import Zero
from qiskit.aqua.components.optimizers import SPSA
from qiskit.aqua.operators import WeightedPauliOperator
from qiskit.chemistry import FermionicOperator
from qiskit.chemistry.components.initial_states import HartreeFock
from qiskit.chemistry.components.variational_forms import UCCSD
from qiskit.chemistry.drivers import PySCFDriver, UnitsType
from qiskit.circuit.library import EfficientSU2
from qiskit.ignis.mitigation.measurement import CompleteMeasFitter
from qiskit.providers.aer import QasmSimulator, AerProvider
from qiskit.providers.aer.noise import NoiseModel
from qiskit.test.mock import *

plt.rcParams["figure.dpi"] = 100
plt.style.use("seaborn")

basis_sets = ['sto3g', '631g', 'ccpvdz']
threshold = 0.00000001

def get_qubit_hamiltonian(d, basis_set, map_type='jordan_wigner'):
    """
    Obtain the qubit operators from the fermionic operators after second
    quantization.
    Args:
        d: inter-atomic distance
        basis_set: computational chemistry basis set
        map_type: encoding method

    Returns: qubit Hamiltonian as a weighted summed of Paulis, energy
    shift because of nuclear repulsion, number of particles and number of
    spin orbitals

    """
    driver = PySCFDriver(atom="H .0 .0 .0; H .0 .0 " + str(d) ,
                         unit=UnitsType.ANGSTROM,
                         charge=0,
                         spin=0,
                         basis=basis_set)
    molecule = driver.run()
    h1 = molecule.one_body_integrals
    h2 = molecule.two_body_integrals
    num_particles = molecule.num_alpha + molecule.num_beta
    num_spin_orbitals = molecule.num_orbitals * 2
    repulsion_energy = molecule.nuclear_repulsion_energy
    fermionic_op = FermionicOperator(h1,h2)
    qubit_op = fermionic_op.mapping(map_type=map_type, threshold=threshold)
    return qubit_op, repulsion_energy, num_particles, num_spin_orbitals


qubit_ops = {}
for set in basis_sets:
    qubit_ops["qubitOp_{0}".format(set)],_,_,_ =
    get_qubit_hamiltonian(0.735, set)

for label in qubit_ops:
    print("\n --- {} ---".format(label),qubit_ops[label])
```

```
--- qubitOp_sto3g --- Representation: paulis, qubits: 4, size: 15

 --- qubitOp_631g --- Representation: paulis, qubits: 8, size: 185

 --- qubitOp_ccpvdz --- Representation: paulis, qubits: 20, size: 2951
```

According to the output, `qubits` represent the number of qubits needed and `size` represent the number of Pauli strings in the Hamiltonian written as a weighted sum. For example, the 4 qubit Hamiltonian obtained with STO-nG basis set and JW encoding above is the sum of 15 weighted Pauli strings given by

$$\begin{aligned}H = h_0 I + h_1 Z_0 + h_2 Z_1 + h_3 Z_2 + h_4 Z_3 + h_5 Z_0 Z_1 + h_6 Z_0 Z_2 + h_7 Z_1 Z_2 + h_8 Z_0 Z_3 + h_9 Z_1 Z_3 \\ + h_{10} Z_2 Z_3 + h_{11} Y_0 Y_1 X_2 X_3 + h_{12} X_0 Y_1 Y_2 X_3 + h_{13} Y_0 X_1 X_2 Y_3 + h_{14} X_0 X_1 Y_2 Y_3.\end{aligned} \tag{1.28}$$

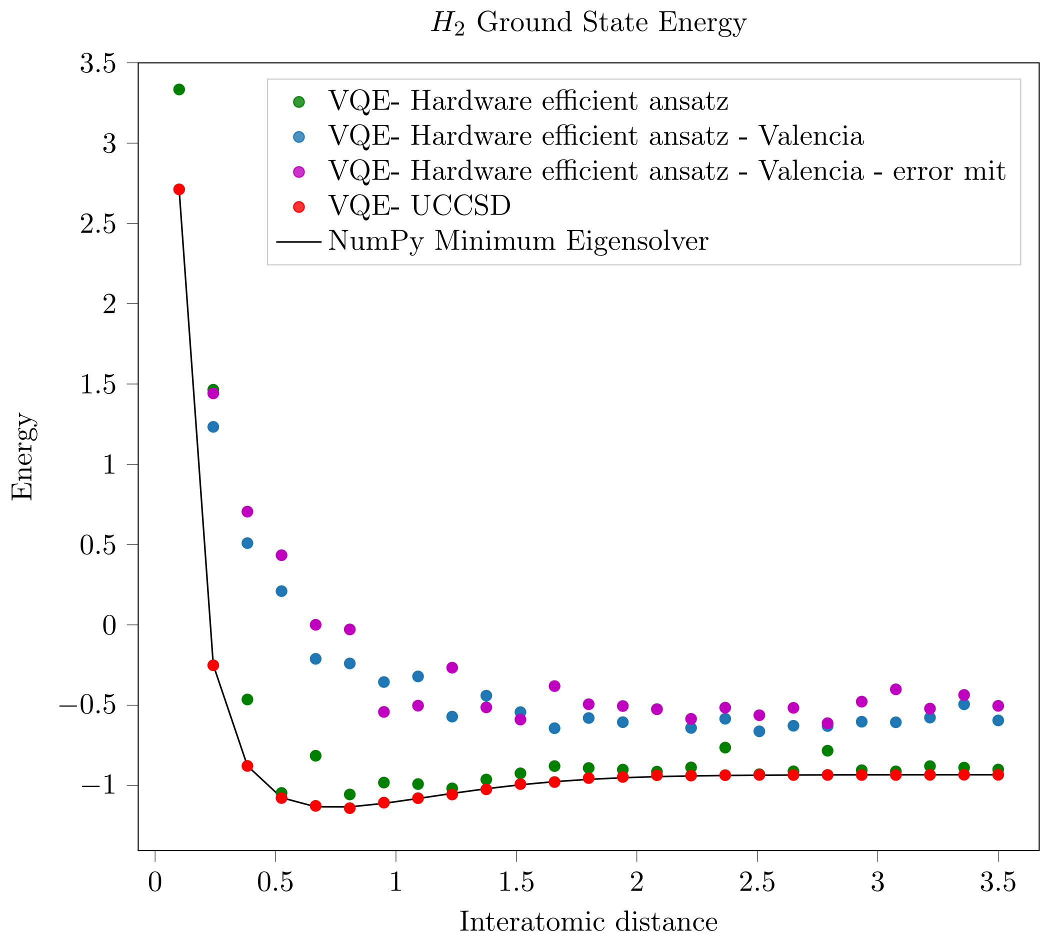
To be able to simulate the problem on a personal computer, it's better to choose the basis that will result in fewer qubits. Therefore let's fix the basis set to STO-3G for convenience, but do keep in mind that working in a suitably large basis set is crucial for obtaining accurate results.

```
bond_lengths = np.linspace(0.1, 3.5, 30)

devices = ["fake_valencia", "fake_valencia_with_mitigation"]

def construct_qinstance(device):
    """
    Create a quantum instance with required backend properties including
    noise models

    Args:
        device:  high performance simulator backend to be used
            qasm_simulator : ideal multi-shot execution
            fake_valencia : simulation of an actual device backend
            fake_valencia_with_mitigation :error mitigation applied

    Returns: QuantumInstance obj

    """
    simulator_backend = Aer.get_backend('qasm_simulator')
    device_backend = FakeValencia()
    qubit_connectivity = device_backend.configuration().coupling_map
    noise_model = NoiseModel.from_backend(device_backend)

    if device == "qasm_ideal":
        qinstance=QuantumInstance(backend=simulator_backend,
                                  seed_simulator=137)

    elif device == "fake_valencia":
        simulator = QasmSimulator(provider=AerProvider(),
                                  method='density_matrix')

        qinstance = QuantumInstance(backend=simulator,
                                    noise_model=noise_model,
                                    coupling_map=qubit_connectivity,
                                    seed_simulator=137,
                                    seed_transpiler=137)

    elif device == "fake_valencia_with_mitigation":
        simulator = QasmSimulator(provider=AerProvider(),
                                  method='density_matrix')

        qinstance = QuantumInstance(backend=simulator,
                                    seed_simulator=137,
                                    seed_transpiler=137,
                                    noise_model=noise_model,
                                    measurement_error_mitigation_cls =
    CompleteMeasFitter,
                                    cals_matrix_refresh_period=30)
    return qinstance


exact_energies = []

vqe_energies_effSU2 = []
vqe_energies_uccsd = []
vqe_energies_effSU2_fake = []
vqe_energies_effSU2_fake_mit = []


counts_effSU2 = []
values_effSU2 = []
counts_effSU2_fake = []
values_effSU2_fake = []
counts_effSU2_fake_mit = []
values_effSU2_fake_mit = []

counts_uccsd = []
values_uccsd = []

def store_intermediate_result_uccsd(eval_count, parameters, mean, std):
    """
    Callback function for VQE that enables storange of intermediate values
    Args:
        eval_count: number of evaluations for convergence
        parameters: variational paratmeters
        mean: mean value
        std: standard deviation

    Returns:
        counts, mean

    """
    counts_uccsd.append(eval_count)
    values_uccsd.append(mean)
    return counts_uccsd, values_uccsd

def store_intermediate_result_effSU2(eval_count, parameters, mean, std):
    counts_effSU2.append(eval_count)
    values_effSU2.append(mean)
    return counts_effSU2, values_effSU2

```

```python
def store_intermediate_result_effSU2_fake(eval_count, parameters, mean, std):
    counts_effSU2_fake.append(eval_count)
    values_effSU2_fake.append(mean)
    return counts_effSU2_fake, values_effSU2_fake

def store_intermediate_result_effSU2_fake_mit(eval_count, parameters, mean, std):
    counts_effSU2_fake_mit.append(eval_count)
    values_effSU2_fake_mit.append(mean)
    return counts_effSU2_fake_mit, values_effSU2_fake_mit


for dist in bond_lengths:
    qubitOp, shift, num_particles, num_spin_orbitals = get_qubit_hamiltonian(dist,basis_set='sto3g')
    result = NumPyEigensolver(qubitOp).run()
    exact_energy = np.real(result.eigenvalues) + shift
    exact_energies.append(exact_energy)
    initial_state = HartreeFock(num_spin_orbitals,
                                num_particles,
                                two_qubit_reduction=False,
                                qubit_mapping='jordan_wigner')
    optimizer = SPSA(maxiter=200)

    ansatz = [UCCSD(num_orbitals= num_spin_orbitals,
                    num_particles= num_particles,
                    initial_state= initial_state,
                    qubit_mapping='jordan_wigner',
                    two_qubit_reduction=False,),
              EfficientSU2(num_qubits=qubitOp.num_qubits,
                           entanglement='linear',
                           initial_state=Zero(qubitOp.num_qubits))]
    for psi in ansatz:
        if psi == ansatz[0]:
            print("----UCCSD----")
            vqe = VQE(qubitOp, psi, optimizer, callback=store_intermediate_result_uccsd)
            vqe_result = np.real(vqe.run(construct_qinstance('qasm_ideal'))['eigenvalue'] + shift)
            vqe_energies_uccsd.append(vqe_result)
            print("Interatomic Distance:", np.round(dist, 2), "VQE Result:", vqe_result)

        elif psi == ansatz[1]:
            print("----Hardware Efficient Ansatz----")
            vqe = VQE(qubitOp, psi, optimizer, callback=store_intermediate_result_effSU2)
            vqe_result = np.real(vqe.run(construct_qinstance('qasm_ideal'))['eigenvalue'] + shift)
            vqe_energies_effSU2.append(vqe_result)
            print("Interatomic Distance:", np.round(dist, 2), "VQE Result:", vqe_result)

            for device in devices:
                if device == "fake_valencia":
                    vqe = VQE(qubitOp, psi, optimizer, callback=store_intermediate_result_effSU2_fake)
                    vqe_result = np.real(vqe.run(construct_qinstance(device))['eigenvalue'] + shift)
                    vqe_energies_effSU2_fake.append(vqe_result)
                elif device == "fake_valencia_with_mitigation":
                    vqe = VQE(qubitOp, psi, optimizer,
    callback=store_intermediate_result_effSU2_fake_mit)
                    vqe_result =
    np.real(vqe.run(construct_qinstance(device))['eigenvalue'] + shift)
                    vqe_energies_effSU2_fake_mit.append(vqe_result)
                print('---{}---'.format(device),"Interatomic Distance:",
    np.round(dist, 2), "VQE Result:", vqe_result)
```

```
----UCCSD----
Interatomic Distance: 0.1 VQE Result: 2.7119857007258554
----Hardware Efficient Ansatz----
Interatomic Distance: 0.1 VQE Result: 3.3340488838888866
---fake_valencia--- Interatomic Distance: 0.1 VQE Result: 4.139296024741463
---fake_valencia_with_mitigation--- Interatomic Distance: 0.1 VQE Result: 3.90308358318452

----UCCSD----
Interatomic Distance: 0.24 VQE Result: -0.2514831954357013
----Hardware Efficient Ansatz----
Interatomic Distance: 0.24 VQE Result: 1.4645029490669041
---fake_valencia--- Interatomic Distance: 0.24 VQE Result: 1.233062149233312
---fake_valencia_with_mitigation--- Interatomic Distance: 0.24 VQE Result: 1.4417877166500799

----UCCSD----
Interatomic Distance: 0.38 VQE Result: -0.8781735458694451
----Hardware Efficient Ansatz----
Interatomic Distance: 0.38 VQE Result: -0.4645055062588499
---fake_valencia--- Interatomic Distance: 0.38 VQE Result: 0.5091323601539339
---fake_valencia_with_mitigation--- Interatomic Distance: 0.38 VQE Result: 0.7053884629541658

----UCCSD----
Interatomic Distance: 0.52 VQE Result: -1.077946447019994
----Hardware Efficient Ansatz----
Interatomic Distance: 0.52 VQE Result: -1.0465178360766256
---fake_valencia--- Interatomic Distance: 0.52 VQE Result: 0.2101641377953375
---fake_valencia_with_mitigation--- Interatomic Distance: 0.52 VQE Result: 0.43395689781122093

----UCCSD----
Interatomic Distance: 0.67 VQE Result: -1.126834155799024
----Hardware Efficient Ansatz----
Interatomic Distance: 0.67 VQE Result: -0.8147573466213908
---fake_valencia--- Interatomic Distance: 0.67 VQE Result: -0.21131100514472856
---fake_valencia_with_mitigation--- Interatomic Distance: 0.67 VQE Result: 0.0006752562516127991

----UCCSD----
Interatomic Distance: 0.81 VQE Result: -1.1410969468908667
----Hardware Efficient Ansatz----
Interatomic Distance: 0.81 VQE Result: -1.0559514724145356
---fake_valencia--- Interatomic Distance: 0.81 VQE Result: -0.24018451331369006
---fake_valencia_with_mitigation--- Interatomic Distance: 0.81 VQE Result: -0.028159296028826053

----UCCSD----
Interatomic Distance: 0.95 VQE Result: -1.1066470715561798
----Hardware Efficient Ansatz----
Interatomic Distance: 0.95 VQE Result: -0.981918023471895
---fake_valencia--- Interatomic Distance: 0.95 VQE Result: -0.35585277698967366
   .   .   .   .   .   .   .   .   .   .   .   .
   .   .   .   .   .   .   .   .   .   .   .   .
   .   .   .   .   .   .   .   .   .   .   .   .
   .   .   .   .   .   .   .   .   .   .   .   .
```

```
plt.plot(bond_lengths, exact_energies, label='NumPy Minimum Eigensolver')
plt.scatter(bond_lengths, vqe_energies_effSU2, label='VQE- Hardware
    efficient ansatz', color='g', marker='o')
plt.scatter(bond_lengths, vqe_energies_effSU2_fake, label='VQE- Hardware
    efficient ansatz - Valencia', marker='o')
plt.scatter(bond_lengths, vqe_energies_effSU2_fake_mit, label='VQE-
    Hardware efficient ansatz - Valencia - error mit', color='m',
    marker='o')
plt.scatter(bond_lengths, vqe_energies_uccsd, label='VQE- UCCSD',
    color='r', marker='o')

plt.xlabel('Interatomic distance')
plt.ylabel('Energy')
plt.title(r'$H_2$ Ground State Energy')
plt.legend(loc='upper right')
```



It's clear from the plot, how the noise from hardware can influence the calculation even for something small as the H_2 in STO-3G basis. Changing the basis set to a different one will lead to more qubits and deeper circuits. In principle it's possible to achieve reasonable good results with the hardware efficient ansatz. This simulation is just an exercise and hence doesn't give extremely good results, but very accurate results were achieved for H_2 with the hardware ansatz in Ref. [15]. As for the UCCSD ansatz, it almost coincides with the exact value calculation but entanglement requirements and deeper circuits will be a problem when extending it to larger molecules. As an exercise one can play around with making the optimization better using a different optimizer or initializing the optimizer with a suitable initial value. Another interesting task would be to look at how the optimizer converges. The function `store_intermediate_result` will store the intermediate values that the user can retrieve through the VQE callback. One can look at the execution counts to understand how the optimizer is performing. This usually falls into the topic of classical optimization and therefore omitted from this example.

1.8 Outlook and conclusions

Will NISQ devices running VQAs be able to outperform classical algorithms that find solutions to the same computational problems? Nobody knows *yet*, but there is hope. Even if the early generation NISQ devices are incapable of competing with classical methods that were honed over decades of research, experimental results encourage that VQAs can soon surpass classical methods, and so spur further scientific advancements. Given that variational algorithms are heuristic and provide only approximate solutions, one might wonder whether there will be value in these strategies once error-correction becomes possible, especially compared to quantum algorithms with a proven advantage on the same tasks. The first thing to acknowledge is that composing algorithms with proven advantage is challenging, and so far only a relatively small number of such algorithms have been discovered. Therefore, for many applications, variational algorithms may be the only quantum algorithms available, even in the error-correction era. A second consideration is that variational algorithms are intended to optimize the use of quantum resources, and therefore it is likely that their error-corrected implementations are more efficient than implementations of their counterparts with proven advantage. Consequently, quantum algorithms might still rival traditional algorithms regarding computational cost. Furthermore, since formal demonstrations of quantum advantage are based on asymptotic considerations, there is a chance for variational algorithms to be more efficient and adequately accurate for particular instances of a problem. Finally, the third consideration is that variational algorithms can be joined with traditional quantum algorithms to obtain even more powerful approaches. A concrete example of this is QPE and VQE. Procedures employed for simulating quantum systems with provable speed-up, such as QPE, require the preparation of states with sufficient overlap with the eigenstates of the Hamiltonian to measure eigenvalues with high probability. Using VQE, it will be possible to prepare such states, ultimately boosting the success probability of phase estimation. Finally, it must be pointed out that the success of variational methods ultimately depends on the quality of the quantum devices and the quantum operations they implement. Therefore, maximizing the utility of quantum devices, in particular, early NISQ devices, is another way to push quantum computing towards the practical frontier. These improvements can be introduced at the software level by finding more efficient protocols to implement quantum operations common to many algorithms. The work presented here is only a part of the continuously growing body of research in variational algorithms. The journey of this field has just started, and many exciting developments await us.

App. 1.A The variation principle

Consider an arbitrary physical system with Hamiltonian H. The time-independent Schrödinger equation then reads as,

$$H\,|\varphi_n\rangle = E_n\,|\varphi_n\rangle\,;\;\; n = 0, 1, 2, \ldots \tag{1.29}$$

Although the Hamiltonian H is known, this is not necessarily the case for its eigenvalues E_n and the corresponding eigenstates $|\varphi_n\rangle$. Hence, the variational methods are the most useful when an exact diagonalization of H is not feasible or not known prior. Such instances of H also come with a drawback as it is difficult, if not impossible, to evaluate the error in a calculation when the final exact solution is unknown. This method is most useful when an exact diagonalization of H is not feasible and not known prior.

Following Ref [30, 34, 35], the solution of Eq (1.29) with the approximate state $|0\rangle$, recast in the form of a variation principle for the energy written as an expectation value is given by,

$$E[\tilde{0}] = \frac{\langle\tilde{0}|H|\tilde{0}\rangle}{\langle\tilde{0}|\tilde{0}\rangle}. \tag{1.30}$$

Here, $|\tilde{0}\rangle$ denotes some approximate eigenstate. The square brackets indicate that the energy functional depends on the form of the wavefunction rather than a set of parameters[12].

As the first step, a one-to-one relationship between the solutions to the Schrödinger equation and the stationary points of the energy functional $E[\tilde{0}]$ is established. Let $|0\rangle$ represent a solution to the Schrödinger equation (1.29) and $|\delta\rangle$, an allowed variation. The approximate state to the eigenstate is then given by,

$$|\tilde{0}\rangle = |0\rangle + |\delta\rangle\,. \tag{1.31}$$

Inserting (1.31) in (1.30) followed by an expansion in orders of $|\delta\rangle$ around $|0\rangle$ yields,

$$\begin{aligned} E[0+\delta] &= \frac{\langle 0|H|0\rangle + \langle 0|H|\delta\rangle + \langle\delta|H|0\rangle + \langle\delta|H|\delta\rangle}{\langle 0|0\rangle + \langle 0|\delta\rangle + \langle\delta|0\rangle + \langle\delta|\delta\rangle} \\ &= E_0 + \langle 0|H-E_0|\delta\rangle + \langle\delta|H-E_0|0\rangle + O(\delta^2) \\ &= E_0 + O(\delta^2) \end{aligned} \tag{1.32}$$

The first order variation in $E[\tilde{0}]$ therefore vanishes whenever $|\tilde{0}\rangle$ corresponds to one of the eigenstates $|0\rangle$. This shows that the eigenstates of the Schrödinger equation represent stationary points of the energy functional.

Conversely, to show that all stationary points of $E[\tilde{0}]$ represent eigenstates of the Schrödinger equation, let $|0\rangle$ be a stationary point of the energy functional. For a variation $|\delta\rangle$, expansion of the energy functional around the stationary point similar to (1.32) gives,

$$\langle 0|H-E[0]|\delta\rangle + \langle\delta|H-E[0]|0\rangle = 0. \tag{1.33}$$

For the variation $i\,|\delta\rangle$,

$$\langle 0|H-E[0]|\delta\rangle - \langle\delta|H-E[0]|0\rangle = 0. \tag{1.34}$$

Combining Eq. (1.33) and Eq. (1.34) one obtains,

$$\langle\delta|H-E[0]|0\rangle = 0. \tag{1.35}$$

Since this relation holds for any arbitrary $|\delta\rangle$, the eigenvalue equation is then,

$$H\,|0\rangle = E[0]\,|0\rangle\,. \tag{1.36}$$

This shows that the each stationary point $E[0]$ of the energy functional $E[\tilde{0}]$ also represents a solution to the Schrödinger equation with eigenvalue $E[0]$. To summarize in one line, the variational principle states that the solutions of the Schrödinger equation (1.29) are equivalent to a variational optimization of the energy functional (1.30).

Due to the flexible nature of variational methods, they can be adapted to very diverse problem scenarios. It also gives great scope to physical intuition in the choice of trial function. It's important to keep in mind that even though good approximations of eigenvalues are obtained rather easily, the approximate states may come with certain completely unpredictable erroneous features that cannot be checked.

[12]In most applications like in VQAs, the wavefunction is described in terms of a finite set of parameters. During such cases, the usual notation for functions is used.

App. 1.B Pauli basis

The contents of this section are from [9].

Definition 1.B.1. *The state of an N-qubit quantum register is represented by a norm-1 vector in the Hilbert space $\mathcal{H} = \mathbb{C}^{2^N}$, under the association $|\psi\rangle \in \mathcal{H} \equiv e^{i\phi}|\psi\rangle$ for $\phi \in \mathbb{R}$.*

Definition 1.B.2. *The **Pauli basis** on N qubits is defined as $\mathbb{P}^N := \{\mathbb{I}, X, Y, Z\}^{\otimes N}$, where $\mathbb{I}, X, Y, Z$ are the 2×2 matrices on $\mathbb{C}^2$:*

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Theorem 1.B.1. *The Pauli basis is a basis for the set of $2^N \times 2^N$ complex valued matrices. It is also a basis for the set of Hermitian matrices if one chooses real coefficients.*

Remark. *The Pauli basis is not a group under matrix multiplication, as the single-qubit Pauli matrices pick up a factor of i on multiplication. The closure of the Pauli basis is the Pauli group $\Pi^N = \{\pm i\} \times \mathbb{P}^N$; this is four times as large, and no longer has the basis properties of $\mathbb{P}^N$.*

Some useful properties of $\mathbb{P}^N$:

- $P^2 = \mathbb{I}$ for all $P \in \mathbb{P}^N$.
- For $P, Q \in \mathbb{P}^N$, either $[P, Q] = 0$, or $\{P, Q\} := 0$, and P commutes with precisely half of $\mathbb{P}^N$
- $P \in \mathbb{P}^N \neq \mathbb{I}$ has only two eigenvalues, ± 1 and the dimension of the corresponding eigenspaces is precisely 2^{N-1} (i.e. each P divides $\mathbb{C}^{2^N}$ in two)
- This division by two may be further continued, given $P, Q \neq \mathbb{I}$ such that $[P, Q] = 0$, P and Q divide the Hilbert space into 4 eigenspaces (labeled by combinations of their eigenvalues).
- To generalize, one can form a $[N, k]$ **stabilizer group** $\mathcal{S}$, generated by k commuting, Hermitian, non-generating elements of $\mathbb{P}^N$ (up to a complex phase); this diagonalizes $\mathbb{C}^{2^k}$ into 2^k unique eigensectors of dimension 2^{N-k}. When $N = k$, these sectors contain single eigenstates, which are called **stabilizer states**.
- Given such a stabilizer state $|\psi\rangle$ and Hermitian $P \in \mathbb{P}^N$, either $P|\psi\rangle = \pm|\psi\rangle$ or $\langle\psi|P|\psi\rangle = 0$.

Bibliography

[1] J. Preskill, Quantum Computing in the NISQ era and beyond, Quantum **2** (July), 79 (2018).

[2] M. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (2011).

[3] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nature Communications **5** (1), 4213 (2014).

[4] E. Farhi, J. Goldstone, and S. Gutmann, A Quantum Approximate Optimization Algorithm, pp. 1–16 (2014).

[5] J. Romero Fontalvo, Variational Quantum Information Processing, Doctoral dissertation, Harvard University, Graduate School of Arts and Sciences. (2019).

[6] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernández, J. Carballo-Franquis, A. Chen, and C. e. a. Chen, Zoufal, Qiskit: An Open-source Framework for Quantum Computing, Quantum Information Science Kit (2019).

[7] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New Journal of Physics **18** (2), 023023 (2016).

[8] S. Lloyd, Computational Capacity of the Universe, Physical Review Letters **88** (23), 237901 (2002).

[9] Y. Herasymenko and T. E. O'Brien, A diagrammatic approach to variational quantum ansatz construction, arXiv pp. 1–18 (2019).

[10] S. Aaronson, Lecture 16 - QMA, Quantum Complexity Theory– Course No. 6.845, MIT OpenCourseWare (2010).

[11] M. Suzuki, Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems, Communications in Mathematical Physics **51** (2), 183 (1976).

[12] A. G. Taube and R. J. Bartlett, New perspectives on unitary coupled-cluster theory, International Journal of Quantum Chemistry **106** (15), 3393 (2006).

[13] D. Wecker, M. B. Hastings, and M. Troyer, Progress towards practical quantum variational algorithms, Physical Review A - Atomic, Molecular, and Optical Physics **92** (4), 042303 (2015).

[14] M. H. Yung, J. Casanova, A. Mezzacapo, J. McClean, L. Lamata, A. Aspuru-Guzik, and E. Solano, From transistor to trapped-ion computers for quantum chemistry, Scientific Reports **4** (1), 1 (2014).

[15] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature **549** (7671), 242 (2017).

[16] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, A quantum engineer's guide to superconducting qubits, Applied Physics Reviews **6** (2), 021318 (2019).

[17] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J. S. Chen, N. C. Pisenti, M. Chmielewski, C. Collins, K. M. Hudek, J. Mizrahi, J. D. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A. M. Ducore, A. Blinov, S. M. Kreikemeier, V. Chaplin, M. Keesan, C. Monroe, and J. Kim, Benchmarking an 11-qubit quantum computer, Nature Communications **10** (1), 1 (2019).

[18] A. Lucas, Ising formulations of many NP problems, Frontiers in Physics **2**, 1 (2014).

[19] P. Vikstål, M. Grönkvist, M. Svensson, M. Andersson, G. Johansson, and G. Ferrini, Applying the quantum approximate optimization algorithm to the tail-assignment problem, Physical Review Applied **14** (3), 034009 (2020).

[20] S. Lloyd, Universal Quantum Simulators, Science **273** (5278), 1073 (1996).

[21] T. C. Yen, V. Verteletskyi, and A. F. Izmaylov, Measuring All Compatible Operators in One Series of Single-Qubit Measurements Using Unitary Transformations, Journal of Chemical Theory and Computation **16** (4), 2400 (2020).

[22] C. Hempel, C. Maier, J. Romero, J. McClean, T. Monz, H. Shen, P. Jurcevic, B. P. Lanyon, P. Love, R. Babbush, A. Aspuru-Guzik, R. Blatt, and C. F. Roos, Quantum chemistry calculations on a trapped-ion quantum simulator, Physical Review X **8**, 031022 (2018).

[23] J. I. Colless, V. V. Ramasesh, D. Dahlen, M. S. Blok, M. E. Kimchi-Schwartz, J. R. McClean, J. Carter, W. A. de Jong, and I. Siddiqi, Computation of molecular spectra on a quantum processor with an error-resilient algorithm, Physical Review X **8** (2018).

[24] R. Santagati, J. Wang, A. A. Gentile, S. Paesani, N. Wiebe, J. R. McClean, S. Morley-Short, P. J. Shadbolt, D. Bonneau, J. W. Silverstone, D. P. Tew, X. Zhou, J. L. O'Brien, and M. G. Thompson, Witnessing eigenstates for quantum simulation of hamiltonian spectra, Science Advances **4**, eaap9646 (2018).

[25] E. F. Dumitrescu, A. J. McCaskey, G. Hagen, G. R. Jansen, T. D. Morris, T. Papenbrock, R. C. Pooser, D. J. Dean, and P. Lougovski, Cloud quantum computing of an atomic nucleus, Physical Review Letters **120**, 210501 (2018).

[26] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, Quantum chemistry in the age of quantum computing, Chemical Reviews **119**, 10856 (2019).

[27] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, Quantum computational chemistry, Reviews of Modern Physics **92** (1) (2020).

[28] P. J. O'Malley, R. Babbush, I. D. Kivlichan, J. Romero, J. R. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, P. V. Coveney, P. J. Love, H. Neven, A. Aspuru-Guzik, and J. M. Martinis, Scalable quantum simulation of molecular energies, Physical Review X **6**, 1 (2016).

[29] Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma, S. Wouters, and G. K. Chan, Pyscf: the python-based simulations of chemistry framework,

[30] T. Helgaker, P. Jørgensen, and J. Olsen, *Molecular Electronic-Structure Theory* (Wiley, 2000).

[31] P. Atkins and R. Friedman, *Molecular Quantum Mechanics*, 5th edition (Oxford University Press, 2010).

[32] J. T. Seeley, M. J. Richard, and P. J. Love, The bravyi-kitaev transformation for quantum computation of electronic structure, The Journal of Chemical Physics **137** (22), 224109 (2012).

[33] Fermionic quantum computation, Annals of Physics **298** (1), 210 (2002).

[34] C. Cohen-Tannoudji, B. Diu, and F. Laloë, *Quantum Mechanics, Volume 2: Angular Momentum, Spin, and Approximation Methods* (Wiley, 2019).

[35] J. J. Sakurai and J. Napolitano, *Modern Quantum Mechanics* (Cambridge University Press, 2021).