Applying Reinforcement Learning to Hamiltonian Engineering

APPLYING REINFORCEMENT LEARNING TO HAMILTONIAN ENGINEERING

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Bachelor of Arts

in

Physics

by

Will Kaufman

DARTMOUTH COLLEGE

Hanover, New Hampshire

May 20, 2021

Examining Committee:

Chandrasekhar Ramanathan, Chair

Lorenza Viola

Bo Zhu

Abstract

Hamiltonian engineering encompasses a variety of important problems in quantum physics and quantum control. For solid state spin systems in particular, the ability to engineer an effective Hamiltonian to decouple dipolar interactions is desirable for improving spectroscopy and increasing spin coherence times. Average Hamiltonian theory (AHT) has been used to design pulse sequences for Hamiltonian engineering, but the computational complexity of calculating higher-order terms inherently limits its performance. This work explores reinforcement learning as an alternative approach to designing pulse sequences for Hamiltonian engineering. The AlphaZero algorithm was trained to construct pulse sequences of varying lengths to decouple all interactions in spin systems. High-fidelity pulse sequences were identified after introducing additional constraints to the algorithm's tree search, and were trained to be robust to multiple sources of error by including those errors in training. The RL-based pulse sequences have lower fidelity than the CORY48 pulse sequence (a state-of-the-art AHT-based pulse sequence) both in computational simulations and experiment, indicating that the current RL implementation does not outperform existing approaches using AHT.

Contents

Abstract . . . ii

1 Background **1**

1.1 Spin-1/2 Systems and Nuclear Magnetic Resonance . . . 2

1.2 Hamiltonian Engineering . . . 9

1.2.1 Average Hamiltonian Theory . . . 9

1.2.2 WAHUHA-4 Pulse Sequence . . . 13

1.2.3 CORY-48 Pulse Sequence . . . 15

2 Reinforcement Learning **17**

2.0.1 Algorithm Examples . . . 19

3 Methods **23**

3.1 Simulated Spin Systems . . . 23

3.2 Reinforcement Learning Implementation . . . 25

3.2.1 Action Space and Representation . . . 25

3.2.2 State Space and Representation . . . 27

3.2.3 Reward Function . . . 28

3.2.4 AlphaZero Algorithm . . . 28

3.2.5 Neural Network Architecture . . . 33

3.3 AHT-Constrained Search . . . 34

4 Results **36**

4.1 Robustness Evaluation 42

4.2 Experimental Validation 49

5 Conclusion and Future Work **51**

Appendix **54**

5.1 Important Quantum Mechanics Concepts 54

5.2 Neural Network Architecture 56

5.3 Pulse Sequence Definitions 56

5.3.1 CORY48 56

5.3.2 AlphaZero Pulse Sequences: No Errors 56

5.3.3 AlphaZero Pulse Sequences: Rotation Errors 57

5.3.4 AlphaZero Pulse Sequences: All Errors 58

Chapter 1

Background

Some physicists believe that quantum physics is on the cusp of a "second quantum revolution" (NIST (2018)). The ability to probe quantum systems with finer precision and control them to a greater extent, as opposed to passively observing emergent phenomena that arise, has the potential to radically transform many different areas, from sensing to simulation to quantum computation.

Hamiltonian engineering is an important problem in the effort to more finely control quantum systems. Every system has a Hamiltonian that characterizes its dynamics, and as the name suggests, Hamiltonian engineering seeks to "engineer" a different Hamiltonian so that the system's dynamics are altered. This has been applied to NMR spectroscopy by engineering Hamiltonians to remove interactions that cause noisier measurements. Engineering Hamiltonians can also be used for sensing (where certain interactions are sensitive to external fields while being robust to decoherence) and for quantum simulation (where novel Hamiltonians must be implemented in existing systems).

Two specific examples where Hamiltonian engineering can be readily applied include NMR spectroscopy and spin bath engineering. For example, in NMR spectroscopy of organic solids, the collective magnetization of spin-1/2 protons in a sample can be measured, and the resulting signal informs the particular chemical environments in which the spins exist. However, for solid-state samples, magnetic dipolar interactions between spins lead to signal decay, inhibiting our ability to learn about the chemical environments. However, various methods have been developed to effectively *decouple* the magnetic dipolar interactions between spins, improving the signal drastically.

Magnetic dipole interactions between a central spin and many surrounding bath spins (such as an NV center surrounded by many P1 centers) also lead to unwanted effects, in this case a decay of the central spin coherence time. Long coherence times for the system of interest are desirable, and finding ways to decouple interactions between the central spin and bath spins would help that endeavor.

1.1 Spin-1/2 Systems and Nuclear Magnetic Resonance

A spin-1/2 particle (such as an electron or proton) has two possible values for its intrinsic angular momentum or "spin," so the corresponding Hilbert Space for a single spin-1/2 particle has dimension two. I_x, I_y and I_z denote the operators for nuclear spin angular momentum about x, y, and z, respectively. When placed in a magnetic field, the magnetic dipole moment (which is proportional to spin) will interact with the field and the spin will begin to precess. For a collection of N spin-1/2 particles, the corresponding Hilbert Space has dimension 2^N. For an ensemble of spins, the term $I_z^{(j)}$ is shorthand for a tensor product of operators, all identity except for the jth operator

$$I_z^{(j)} = \mathbb{1} \otimes \mathbb{1} \otimes \cdots \otimes I_z \otimes \cdots \otimes \mathbb{1} \tag{1.1}$$

Similarly, the term $I_z^{(j)} I_z^{(k)}$ is shorthand for a product of identity operators except for the jth and kth operators.

Nuclear magnetic resonance (NMR) encompasses the study of nuclear spins and the various interactions they have with external fields and with each other. The key interactions considered in this work are given below in equations 1.2 through 1.7, but a complete description of the interactions present in NMR can be found in Haeberlen (1976).[1] Figure 1.1 graphically depicts a spin system in a lattice and the dipolar interactions between spins.

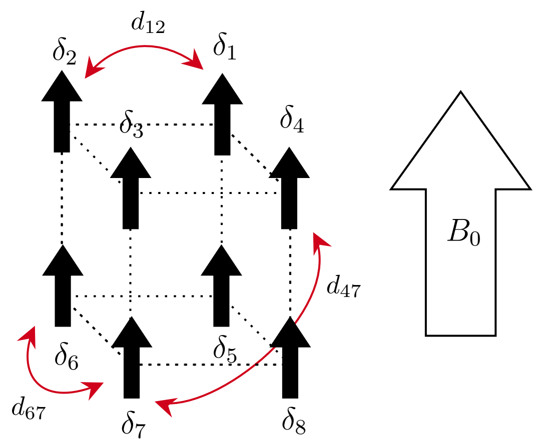


Figure 1.1: A system of spins in a cubic lattice. Each spin may have a different local magnetic field due to its surrounding electronic structure, which causes the chemical shifts δ_i away from the Larmor frequency ω_0. Dipolar interactions between pairs of spins are shown in red arrows, and only shown for a few pairs of spins. The main B_0 field is present everywhere in the lattice.

The system Hamiltonian can be decomposed into several parts, as shown in equations 1.2 to 1.5.

$$H_{\text{sys}} = H_Z + H_{\text{CS}} + H_D \tag{1.2}$$

$$H_Z = \omega_0 \sum_j I_z^{(j)} \qquad \text{(Zeeman)} \tag{1.3}$$

$$H_{\text{CS}} = \sum_j \delta_i I_z^{(j)} \qquad \text{(chemical shift)} \tag{1.4}$$

$$H_D = \sum_{j,k} d_{jk} \left(3 I_z^{(j)} I_z^{(k)} - \mathbf{I}^{(\mathbf{j})} \cdot \mathbf{I}^{(\mathbf{k})}\right) \qquad \text{(dipolar)} \tag{1.5}$$

The chemical shift and dipolar terms are collectively called the internal Hamiltonian

$$H_{\text{int}} = H_{\text{CS}} + H_D \tag{1.6}$$

[1]The quadrupolar interaction and J coupling are not considered here, but it's good to remember they're still there in general.

because those interactions are internal to the sample.

The Zeeman interaction H_Z captures the coupling of spins to the main external magnetic field along the z-axis. The oscillation frequency associated with the Zeeman interaction is called the Larmor frequency $\omega_0 = \gamma_i B_0$, and only differs between spins if the gyromagnetic ratio differs (e.g. between H^1 and C^{13}). We assume going forward that all spins are of the same species.[2] The chemical shift interaction H_{CS} captures the local magnetic field differences for each spin due to the electronic structure surrounding the spin. The dipolar interaction between spins is represented by H_D.

In addition, the system can also be externally manipulated by an additional control term, so that the total Hamiltonian $H(t) = H_{\text{sys}} + H_{\text{ctrl}}(t)$. The control term, commonly written in NMR as $H_{\text{rf}}(t)$, is the interaction between the spins and a time-varying transverse magnetic field along the x-axis.[3] In the lab frame, the rf term is given by equation 1.7.

$$H_{\text{rf}}(t) = u(t)\omega_1 \sum_j I_x^{(j)} \tag{1.7}$$

where $u \in [-1, 1]$ parameterizes the strength of the B_1 field.

In an NMR spectrometer, a sample is placed into an external magnetic field B_0 (defining the principal axis or z-axis). The spins in the sample reach thermal equilibrium and are characterized by the density operator

$$\rho_{\text{th}} = \frac{\exp\{-\beta H_{\text{int}}\}}{Z} \tag{1.8}$$

where Z is the partition function $Z = \text{Tr}\{\exp\{-\beta H_{\text{int}}\}\}$. The dominant term in the inter-

[2]The external magnetic field B_0 varies slightly from spin to spin due to experimental imperfections, but those field inhomogeneities are ignored.

[3]The reason it is labeled as H_{rf} is because the field's oscillation frequency is matched to the Larmor frequency ω_0 which is in the radiofrequency range.

nal Hamiltonian is H_Z, so the first-order approximation for ρ_{th} is given by

$$\rho_{\text{th}} \approx \mathbb{1} - \delta \sum_j I_z^{(j)} \tag{1.9}$$

The identity term does not contribute to the system's dynamics ($U\mathbb{1}U^\dagger = \mathbb{1}$) so the collective spin term $\sum_j I_z^{(j)}$ is the lowest-order term that captures the dynamics and measurement outcomes in the spin system. The collective spin determines the net magnetization of the sample, so the thermal equilibrium state corresponds to initial net magnetization along z.

The Zeeman term dominates the system Hamiltonian (and the dynamics it generates are not very interesting), so it is oftentimes useful to work in the interaction frame of the Zeeman term, also known as the "rotating frame." The rotating frame transformation operator is given by

$$U_Z(t) = \exp\left(-i\omega_0 t \sum_j I_z^{(j)}\right) \tag{1.10}$$

which corresponds to rotation about the z-axis with angular velocity ω_0.[4] The interaction frame Hamiltonian is then given by

$$\begin{aligned} H^{(R)}(t) &= U_Z(t)^\dagger \left[H_{\text{CS}} + H_D + H_{\text{rf}}(t)\right] U_Z(t) \\ &= H_{\text{CS}} + H_D + H_{\text{rf}}^{(R)}(t) \end{aligned} \tag{1.11}$$

Both H_{CS} and H_D commute with H_Z and therefore commute with $U_Z(t)$. The rf field $H_{\text{rf}}(t)$ does not commute with H_Z ($[I_x, I_z] \neq 0$), so we need to consider how the rf field interaction transforms in the rotating frame.

If the transverse magnetic field oscillates at the Larmor frequency of the spins with phase offset ϕ

$$u(t) = \cos(\omega_0 t + \phi) \tag{1.12}$$

[4]Refer to the appendix for additional information on interaction frames.

then this field can be thought of as two counter-rotating magnetic fields whose net effect is the transverse field $\mathbf{B_1}(t)$

$$\mathbf{B_1}(t) = \frac{1}{2}\left[B_1(\cos(\omega_0 t + \phi)\hat{\mathbf{x}} + \sin(\omega_0 t + \phi)\hat{\mathbf{y}}) + B_1(\cos(\omega_0 t + \phi)\hat{\mathbf{x}} - \sin(\omega_0 t + \phi)\hat{\mathbf{y}})\right] \quad (1.13)$$

In a frame rotating about the z-axis with frequency ω_0, it looks like there is a fixed magnetic field from one of the rotating fields and a counter-rotating field with frequency $2\omega_0$. If we ignore the high-frequency field (i.e. the rotating wave approximation), the spins in the rotating frame see a fixed magnetic field at an azimuthal angle ϕ in the xy-plane, and begin to precess about the new field. In the rotating frame, the rf Hamiltonian effectively becomes

$$H_{\text{rf}}^{(R)}(t) = u_1(t)\omega_1 \sum_j I_x^{(j)} + u_2(t)\omega_1 \sum_j I_y^{(j)} \quad (1.14)$$

This rf field in the rotating frame can therefore rotate the spins about the x or the y axis by controlling u_1 and u_2 respectively. By applying this transverse B_1 field for a specific duration t with phase ϕ, the system in the rotating frame is transformed according to the unitary operator

$$U = \exp\{-i\omega_1 t(\cos(\phi)I_x + \sin(\phi)I_y)\} \quad (1.15)$$

In particular, if $t = \frac{\pi}{2\omega_1}$, the spins can be rotated from the initial equilibrium state to a state with net magnetization in the xy-plane. This is called a $\pi/2$-pulse (because it rotates the spins $\pi/2$ radians or $90°$). Increasing the strength of the B_1 field or the duration of the pulse increases the rotation angle, and adjusting the phase of the pulse adjusts the rotation axis. From this point forward, we will work in the rotating frame unless otherwise specified, which amounts to neglecting H_Z in the system Hamiltonian and using equation 1.14 instead of 1.7 for H_{rf}.

In NMR experiments, the general procedure is the following:

1. Place the sample in the main magnetic field and wait for the sample to reach thermal

equilibrium ρ_{th} (equation 1.8). The net magnetization of the sample is along the z axis.

2. Apply a $\pi/2$-pulse to rotate the spins and the net magnetization vector into the xy-plane.

3. Apply a pulse sequence repeatedly so the system evolves under an engineered Hamiltonian. The density operator then evolves according to the propagator U determined by the pulse sequence, so at multiples of the cycle time T of the pulse sequence the density operator is given by

$$\rho(NT) = U^N \rho_{\text{th}} {U^\dagger}^N \tag{1.16}$$

4. Measure the oscillating induced emf due to the precessing spins. This measurement corresponds to the complex signal given by

$$\begin{aligned} \langle I_+(t)\rangle &= \langle I_x(t)\rangle + i\langle I_y(t)\rangle \\ &= \operatorname{Tr}(I_x\rho(t)) + i\operatorname{Tr}(I_y\rho(t)) \end{aligned} \tag{1.17}$$

The Fourier transform of the signal gives a spectrum of the sample, including chemical shift frequencies δ_i of different spin species. There are *many* other variations on NMR experiments, but that is the basic methodology. See figure 1.2 for a diagram showing a typical NMR experiment, including the "pulse train" that is applied to the system.

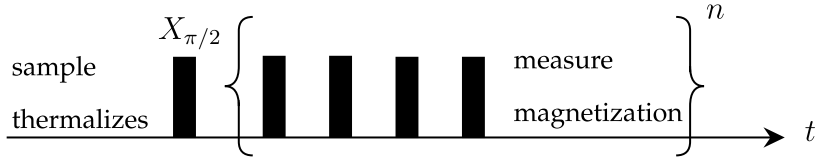


Figure 1.2: A typical NMR experiment, including initializing the state, a $\pi/2$-pulse to measure magnetization in the xy-plane, and many repetitions of a pulse sequence and magnetization measurements. After each pulse sequence (between the brackets), the dynamics appear to have evolved due to an engineered Hamiltonian.

If the NMR sample is a liquid, then the sample's molecules are constantly tumbling past each other. As a result, dipolar interactions between spins from different molecules are not constant over the course of the experiment, and in aggregate the dipolar interactions are averaged to zero. This is called "motional averaging" of the dipolar interactions

$$H_{\text{int}} = H_{\text{CS}} + H_D \longrightarrow H_{\text{CS}}$$

The chemical shifts for each spin do stay constant (because the local magnetic field due to molecular structure stays the same), so the resulting spectrum clearly resolves chemical shift peaks. Recall that H_Z is not present in the rotating frame.

In contrast, motional averaging does not occur in solid-state NMR. Because the spins are in fixed positions relative to other spins in the sample, dipolar interactions affect the net magnetization of the sample and lead to a broadening of the peaks in the spectrum. This line-broadening inhibits accurate measurements of chemical shifts in solid samples. See figure 1.3 for a comparison of spectra between liquid- and solid-state NMR.

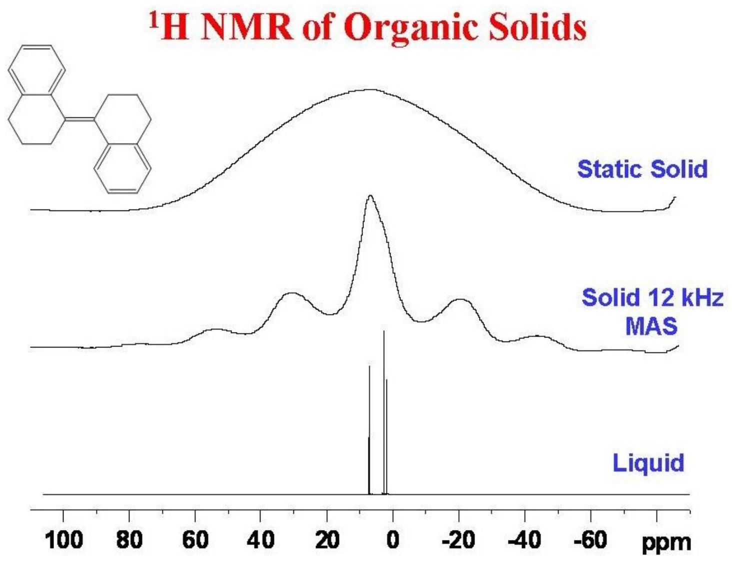


Figure 1.3: NMR spectra for a solid sample, solid sample with magic-angle spinning, and liquid sample. The goal of NMR spectroscopy is to decouple dipolar interactions (top) so that the peaks can be clearly resolved (bottom). Magic angle spinning (MAS) is one technique for narrowing linewidths in solid samples. From Facey (2008).

1.2 Hamiltonian Engineering

Hamiltonian engineering is an important problem within the broader field of quantum control. The goal of Hamiltonian engineering is to choose parameters for the control Hamiltonian so that, when measured stroboscopically[5], the system appears to evolve under a target effective Hamiltonian H_{target} instead of the system Hamiltonian H_{sys}. In most cases the target Hamiltonian is time-independent.

If we are able to engineer an effective Hamiltonian, then we have also created a unitary transformation $U(T) = \exp\{-iH_{\text{target}}T\}$. Conversely, if we have implemented a unitary transformation $U(T)$, then there exists an effective time-independent Hamiltonian that characterizes the dynamics over time T (note that this Hamiltonian is not unique).

1.2.1 Average Hamiltonian Theory

Average Hamiltonian theory (AHT) provides a framework with which the Hamiltonian engineering problem can be approached. The following presentation of AHT is inspired by Brinkmann (2016); Gerstein and Dybowski (1985); Haeberlen (1976).

For a particular Hamiltonian $H(t)$, the propagator $U(t)$ is defined by equation 5.3

$$i\frac{dU(t)}{dt} = H(t)U(t), U(0) = \mathbb{1} \tag{1.18}$$

As the name suggests, average Hamiltonian theory lets us express the propagator $U(t)$ at each time t in terms of an *average* time-independent Hamiltonian $\overline{H}$

$$U(t) = \exp\left\{-i\overline{H}(t)t\right\} \tag{1.19}$$

The above equation seems to suggest that $\overline{H}(t)$ is not time-independent–it is explicitly dependent on time! It is true that $\overline{H}(t)$ itself is not time-independent, but *given* a time T,

[5]Stroboscopically means measured at regular intervals $0, T, 2T, \ldots$.

we can find $\overline{H}(T)$ so that the propagator $U(T)$ can be expressed *as though* the Hamiltonian were $\overline{H}(T)$ for the time interval $t \in [0, T]$.

How then do we determine the average Hamiltonian? One approach is to use the Magnus Expansion (Blanes et al. (2009); Blanes et al. (2010)). The Magnus Expansion begins with the ansatz that an exponential solution for the propagator exists

$$U(t) = \exp\{\Omega(t)\}, \Omega(0) = 0 \tag{1.20}$$

and proceeds by finding a series expansion for $\Omega(t)$

$$\Omega(t) = \sum_k \Omega_k(t) \tag{1.21}$$

The first few terms are presented below

$$\Omega_1(t) = -i\int_0^t dt_1 H(t_1) \tag{1.22}$$

$$\Omega_2(t) = -\frac{1}{2}\int_0^t dt_1 \int_0^{t_1} dt_2 [H(t_1), H(t_2)] \tag{1.23}$$

$$\vdots$$

Derivations of the Magnus Expansion can be found in Gerstein and Dybowski (1985); Blanes et al. (2009); Blanes et al. (2010).

From $\Omega(t)$, an expression for the average Hamiltonian can be derived by making the comparison

$$\Omega(t) = -i\overline{H}(t)t \implies \overline{H}(t) = \frac{i}{t}\Omega(t) \tag{1.24}$$

This then gives us a series expansion for the average Hamiltonian $\overline{H}(t) = \sum_k \overline{H}^{(k)}(t)$, with the first few terms given below.

$$\overline{H}^{(0)} = \frac{1}{t}\int_0^t dt_1 H(t_1) \tag{1.25}$$

$$\overline{H}^{(1)} = \frac{1}{2it}\int_0^t dt_1 \int_0^{t_1} dt_2 \left[H(t_1), H(t_2)\right] \tag{1.26}$$

$$\begin{aligned}\overline{H}^{(2)} = -\frac{1}{6t}\int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_2} dt_3 \{&[H(t_1),[H(t_2),H(t_3)]] \\ &+ \left[\left[H(t_1),H(t_2)\right], H(t_3)\right]\}\end{aligned} \tag{1.27}$$

Although there appears to be a simple pattern to the terms, this is not the case. Higher-order terms can be determined through a recursive relationship or through particularly complicated explicit forms.

The Magnus Expansion converges when

$$\int_0^t dt_1 ||H(t)||_2 \ll 1 \tag{1.28}$$

which effectively requires that we consider timescales much smaller than the natural timescale of the Hamiltonian.[6] This is also written more succinctly as $||H||t \ll 1$.

Because convergence requires the norm of the Hamiltonian to be small, there are cases when working in the interaction frame of a strong interaction is desirable. In NMR, going from the rotating frame (of the Zeeman interaction) to the interaction frame of the rf field removes the strong H_{rf} term from the nuclear spin Hamiltonian. This interaction frame is commonly called the "toggling frame."

$$H(t) = H_{\text{int}} + H_{\text{rf}}(t) \implies \widetilde{H}(t) = \widetilde{H}_{\text{int}}(t) \tag{1.29}$$

In the toggling frame, the Magnus Expansion can again be used to determine the average Hamiltonian $\overline{\widetilde{H}}(t)$, but using $\widetilde{H}_{\text{int}}(t)$ in place of $H(t)$ in equation 1.25. The upshot is that

[6] A much more detailed overview regarding convergence can be found in Blanes et al. (2009).

by going into the toggling frame, the smaller-norm average Hamiltonian converges for longer time values.

At this point, the identification of an "average Hamiltonian" is of little benefit: $\overline{H}(t)$ might be different at different times, and it may be the average Hamiltonian in the *toggling* frame instead of the lab frame. To address these concerns, we can require that $H_{\text{rf}}(t)$ be "cyclic" and "periodic." For a fixed time $T > 0$,

$$U_{\text{rf}}(T) = \mathbb{1} \qquad \text{(cyclic)} \qquad (1.30)$$

$$H_{\text{rf}}(t + NT) = H_{\text{rf}}(t), N \in \mathbb{Z} \qquad \text{(periodic)} \qquad (1.31)$$

T is called the "cycle time" and should be short enough so that the Magnus Expansion converges. The cyclic and periodic properties ensure that the average Hamiltonian is *the same* and that the toggling frame coincides with the rotating frame at multiples of the cycle time NT.

The AHT framework permits continuous control via the rf field, but we will focus on discrete control, where a finite set of rf field pulses can be applied at multiples of time τ. The set of pulses we will consider are $\pi/2$-pulses about the x- or y-axis, which we will label as $\{X, Y, \overline{X}, \overline{Y}\}$ where the bar denotes a $-\pi/2$ rotation. An X pulse corresponds to the unitary operator

$$U_X = \exp\left\{-i\left[\pi/2\left(\sum_j I_x^{(j)}\right) + H_{\text{int}} t_p\right]\right\} \qquad (1.32)$$

where t_p is the pulse duration. Pulses along the y-axis swap the I_x operator with I_y, and reverse rotations negate the $\pi/2$ term. In the delta-pulse limit (i.e. $t_p = 0$), an X pulse corresponds to the unitary operator $U_X = \exp\{-i\pi/2 \sum_j I_x^{(j)}\}$ and a Y pulse similarly corresponds to $U_Y = \exp\{-i\pi/2 \sum_j I_y^{(j)}\}$.

1.2.2 WAHUHA-4 Pulse Sequence

One of the first pulse sequences developed for Hamiltonian engineering in NMR was the WAHUHA-4 pulse sequence (WHH-4) from Waugh et al. (1968). The pulse sequence is defined as

$$\tau, X, \tau, \overline{Y}, \tau, \tau, Y, \tau, \overline{X}, \tau \tag{1.33}$$

where the sequence should be read from left to right (an X pulse is applied first). The pulse sequence is also shown in figure 1.4.

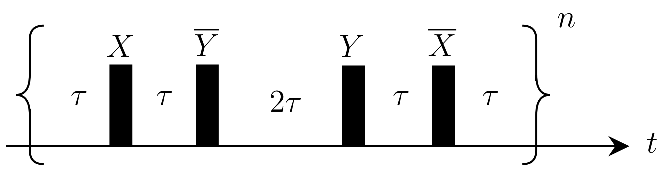


Figure 1.4: The WHH-4 sequence repeated n times in a pulse train. If the magnetization of the sample is measured after each WHH-4 sequence, the dynamics will appear to evolve according to the average Hamiltonian given below.

The propagator for a single cycle of the WHH-4 sequence is

$$U_{\text{WHH-4}} = \exp\{-iH_{\text{int}}\tau\}U_{\overline{X}}\exp\{-iH_{\text{int}}\tau\}U_Y\exp\{-iH_{\text{int}}2\tau\}U_{\overline{Y}}\exp\{-iH_{\text{int}}\tau\}U_X\exp\{-iH_{\text{int}}\tau\} \tag{1.34}$$

Now, using AHT, we express $U_{\text{WHH-4}} = \exp\{-i\overline{H}_{\text{WHH-4}}6\tau\}$ and find an approximation for the average Hamiltonian $\overline{H}_{\text{WHH-4}}$. The toggling frame propagator as well as the toggled $I_z^{(j)}$ and $I_z^{(j)}I_z^{(k)}$ terms are determined by the four pulses in the sequence. For $t \in [0, \tau]$, the toggling frame corresponds to the lab frame. But after the X pulse, the toggling frame propagator is now U_X, so the toggled spin terms are $\widetilde{I}_z^{(j)} = U_X^\dagger I_z^{(j)} U_X = I_y^{(j)}$ [7] The toggling frame and corresponding terms for each time interval are given in table 1.1.

[7]This can be seen by noting that $\exp\{-i\theta I_x\} = \exp\{-i\theta/2\sigma_x\} = \cos(\theta/2)\mathbb{1} - i\sin(\theta/2)\sigma_x$, then using the commutation relations for Pauli matrices to simplify. As a rule of thumb, U_X rotates the lab frame's y-axis into the z-axis so that $\widetilde{I}_z = I_y$, and U_Y rotates z into y.

Table 1.1: The WHH-4 toggling frame propagator and toggled terms at different time intervals.

Time interval	Toggling frame propagator U_{rf}	$\widetilde{I}_z$	$\widetilde{I}_z^{(j)}\widetilde{I}_z^{(k)}$
$[0,\tau]$	$\mathbb{1}$	I_z	$I_z^{(j)}I_z^{(k)}$
$[\tau,2\tau]$	U_X	I_y	$I_y^{(j)}I_y^{(k)}$
$[2\tau,4\tau]$	$U_{\overline{Y}}U_X$	I_x	$I_x^{(j)}I_x^{(k)}$
$[4\tau,5\tau]$	U_X	I_y	$I_y^{(j)}I_y^{(k)}$
$[5\tau,6\tau]$	$\mathbb{1}$	I_z	$I_z^{(j)}I_z^{(k)}$

Note that in the dipolar interaction H_D, the isotropic term $\mathbf{I}^{(j)}\cdot\mathbf{I}^{(k)}$ is invariant under rotations and therefore not transformed by the toggling frame.

We see that WHH-4 is cyclic because $U_{\text{rf}}(6\tau) = \mathbb{1}$, and the applying the WHH-4 sequence repeatedly would satisfy the periodic requirement. To determine the lowest-order term in the Magnus Expansion $\overline{H}^{(0)}$, we use the toggled terms from table 1.1 in equation 1.25.

$$
\begin{aligned}
\overline{H}^{(0)} &= \frac{1}{6\tau}\int_0^{6\tau} dt_1 \widetilde{H}_{\text{int}}(t_1) \\
&= \frac{1}{6\tau}\left[\sum_j \delta_j\left(2\tau I_z^{(j)} + 2\tau I_y^{(j)} + 2\tau I_x^{(j)}\right) + \right. \\
&\qquad \left.\sum_{jk} d_{jk}\left(3(2\tau I_z^{(j)}I_z^{(k)} + 2\tau I_y^{(j)}I_y^{(k)} + 2\tau I_x^{(j)}I_x^{(k)}) - 6\tau \mathbf{I}^{(\mathbf{j})}\cdot\mathbf{I}^{(\mathbf{k})}\right)\right] \\
&= \left[1/3\sum_j \delta_j\left(I_z^{(j)} + I_y^{(j)} + I_x^{(j)}\right) + \sum_{jk} d_{jk}\left(I_z^{(j)}I_z^{(k)} + I_y^{(j)}I_y^{(k)} + I_x^{(j)}I_x^{(k)} - \mathbf{I}^{(j)}\cdot\mathbf{I}^{(k)}\right)\right] \\
&= 1/3\sum_j \delta_j\left(I_z^{(j)} + I_y^{(j)} + I_x^{(j)}\right)
\end{aligned}
$$

The WHH-4 sequence, to lowest order, refocuses or "cancels out" the dipolar interaction between spins while retaining the chemical shift term. Instead of the principal axis being along z, however, the spin operator acts along the diagonal axis $(1,1,1)$.

1.2.3 CORY-48 Pulse Sequence

AHT can be taken even further than the WHH-4 sequence by considering higher-order terms in the Magnus Expansion. The CORY48 sequence is a 72τ, 48-pulse sequence that was designed to decouple all interactions so that $H_{\text{target}} = 0$ (Cory et al. (1990)). Furthermore, the CORY48 sequence decouples the dipolar interactions to *second order*, and is robust to errors including pulse rotation errors and resonance offset errors. The full sequence is presented in equation 1.35.

$$\begin{aligned}
&X, \tau, Y, 2\tau, \overline{X}, \tau, Y, 2\tau, X, \tau, Y, 2\tau, X, \tau, Y, 2\tau, X, \tau, \overline{Y}, 2\tau, X, \tau, Y, 2\tau \\
&\overline{Y}, \tau, \overline{X}, 2\tau, Y, \tau, \overline{X}, 2\tau, \overline{Y}, \tau, \overline{X}, 2\tau, \overline{Y}, \tau, \overline{X}, 2\tau, \overline{Y}, \tau, X, 2\tau, \overline{Y}, \tau, \overline{X}, 2\tau \\
&\overline{X}, \tau, Y, 2\tau, \overline{X}, \tau, \overline{Y}, 2\tau, \overline{X}, \tau, Y, 2\tau, X, \tau, \overline{Y}, 2\tau, \overline{X}, \tau, \overline{Y}, 2\tau, X, \tau, \overline{Y}, 2\tau \\
&Y, \tau, \overline{X}, 2\tau, Y, \tau, X, 2\tau, Y, \tau, \overline{X}, 2\tau, \overline{Y}, \tau, X, 2\tau, Y, \tau, X, 2\tau, \overline{Y}, \tau, X, 2\tau
\end{aligned} \tag{1.35}$$

The full AHT analysis is not presented here, but one can imagine evaluating the three lowest-order terms from equations 1.25 to 1.27 using each of the 48 time intervals. The lowest-order term is not difficult to calculate, but the higher-order terms quickly become intractable as the integrals include more and more nested commutators.

The free evolution of a spin system under H_{int} can be compared to the engineered Hamiltonians from the WHH-4 and CORY48 sequences by running three sets of experiments. The state is initialized to ρ_{th}, rotated by a $\pi/2$-pulse so the net magnetization is along x, and propagated using one of the pulse sequences (i.e. no pulse sequence for free evolution, the WHH-4 sequence, and the CORY48 sequence). Measuring the net magnetization along x gives the correlation function $C_{XX}(t)$ (initialize state along x and measure signal along x). Similarly, C_{YY} can be measured by initializing the state along y and measuring along y. If the engineered Hamiltonian is close to zero, the correlation functions decay slowly.

By comparing the average correlation ($C_{\text{avg}} = (C_{XX}C_{YY}C_{ZZ})^{1/3}$) for free evolution

(FID), the WHH-4 sequence, and the CORY48 sequence in adamantane, it is clear that the longer CORY48 sequence outperforms the shorter WHH-4 sequence in decoupling interactions and extending the coherence time (see figure 1.5).

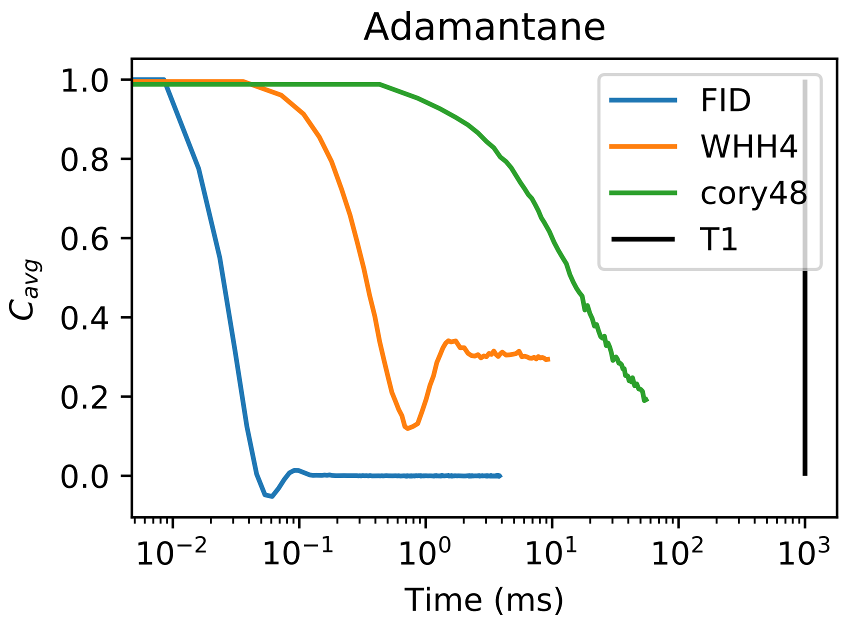


Figure 1.5: The average correlation for a free induction decay (FID), the WHH-4 sequence, and the CORY48 sequence in an adamantane sample. Both the WHH-4 and the CORY48 extend the coherence times of the sample, but CORY48 achieves nearly two orders of magnitude greater improvement. The T_1 time for adamantane is shown at $t = 1$s.

However, AHT may not be a suitable framework with which to develop pulse sequences that improve upon CORY48. The computational resources required to decouple interactions to higher orders would be significant.

Are there other methods for Hamiltonian engineering that can achieve higher-fidelity control in spin systems? This thesis seeks to answer that question by exploring reinforcement learning.

Chapter 2

Reinforcement Learning

> Reinforcement learning is learning what to do–how to map situations to actions–so as to maximize a numerical reward signal. (Sutton and Barto (2018))

Reinforcement learning (RL) has been applied to a variety of different problems, including interacting with physical environments such as robotic manipulation using visual input (Lillicrap et al. (2015)) or playing chess (Silver et al. (2018)). A significant attraction of RL is its generality: any problem that can be formulated as a Markov decision process (MDP) can be approached with RL, and in most cases no prior knowledge of the problem is assumed. Hamiltonian engineering fits into the RL paradigm by considering discrete time steps at which the control Hamiltonian can be set. The environment is the quantum system we are trying to control, the actions are control Hamiltonian parameters at a given time step, and the reward is how "close" the dynamics are to the desired dynamics under the target Hamiltonian. Each of these are developed in more detail in section 3.2.

In the language of RL, an *agent* is responsible for performing *actions* on the *environment* to maximize a *reward signal* (or reward). The agent can see the *state* of the environment which informs what actions to take. See figure 2.1 for a diagram of the agent-environment interactions. In turn, the agent's actions affect the environment's state. The agent must then balance exploration of the environment for learning about novel, possibly better policies, with exploitation of its prior knowledge to maximize rewards.[1]

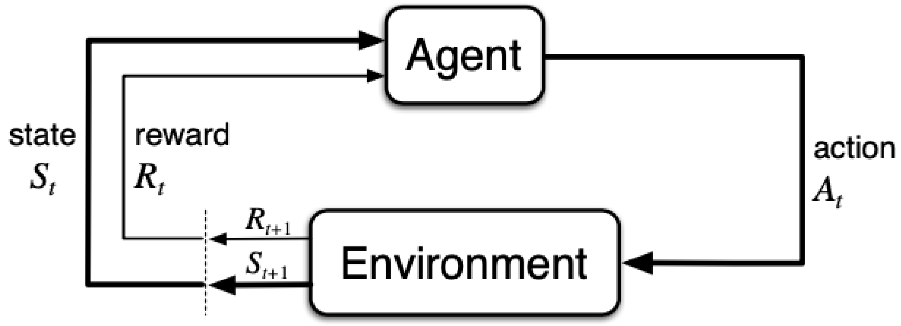


Figure 2.1: The general reinforcement learning paradigm. From Sutton and Barto (2018).

The agent interacts with the environment in a series of *timesteps*. Each timestep begins with a state s_i, the agent chooses an action a_i based on the state, and optionally receives a reward r_i for the particular state-action combination. An *episode* is a collection of timesteps that begin in an initial state and end in a terminal state. For Hamiltonian engineering, an episode is the complete construction of a pulse sequence, where each timestep selects the next pulse depending on the sequence of previous pulses.

The agent decides what actions to perform at each timestep using a policy function π, which maps each state $s \in S$ to either a probability distribution over the action space $\mathcal{P}(A)$ (a "stochastic policy") or simply a particular action $a \in A$ (a "deterministic policy"). Policy functions can be implemented as a dictionary of state-action pairs when the state space is small, or as neural networks which are optimized using gradient-based methods.

The *return* at timestep t is defined as

$$G_t := \sum_{k=t}^{T} \gamma^{k-t} R_k, \quad \gamma \in [0, 1) \tag{2.1}$$

or the total discounted rewards until the end of the episode at timestep T. If the episode has a finite number of timesteps the discount factor is often ignored. The *state-value function* (or value function) is the expected return from a particular state while following a

[1]For a more detailed description of the RL framework, see Sutton and Barto (2018).

policy π

$$v_\pi(s) := \mathbb{E}[G_t | S_t = s] \tag{2.2}$$

If there is only a reward at the end of the episode (as is the case with pulse sequence design), then the value function gives the expected final reward at the end of the episode.

2.0.1 Algorithm Examples

There are many RL algorithms that have been developed, tailored to different characteristics of the RL problem. The RL algorithms can be differentiated according to the following characteristics.

Model An agent in a *model-free* RL algorithm learns a policy directly from observations of the environment. An agent in a *model-based* RL algorithm instead uses observations to model the environment, then uses that model to learn a policy. For example, an agent that plays chess could use a model of the game to model possible future moves by their opponent.

Policy An *on-policy* algorithm learns the policy that is used to interact with the environment. An *off-policy* algorithm has separate policies for learning and environment interaction.

Action space As mentioned before, an action space can be *discrete* (finitely many actions) or *continuous* (infinitely many actions).

Learning method There are general classes of RL algorithms that learn from the agent-environment interactions in different ways. Examples include temporal-difference (TD) learning and policy gradient methods.

More information on RL can be found in Sutton and Barto (2018).

The RL algorithms listed below were each tested as potential options for Hamiltonian engineering. Ultimately, the AlphaZero algorithm was used as the primary RL algorithm in this work as it was the best-performing of those that were tested. A more complete description of the algorithm can be found in section 3.2.

Q-learning and DQN

Q-learning is a model-free, off-policy, temporal-difference (TD) control algorithm applicable to discrete action spaces (Watkins (1989)). As the name suggests, the action-value function $Q(s, a)$ is learned from experiences with the environment according to the following update rule

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \tag{2.3}$$

To encourage exploration of the state and action spaces, the agent typically follows an ϵ-greedy policy (with probability $1 - \epsilon$ perform the action that maximizes Q, otherwise perform a random action), which makes this algorithm *off-policy*.

If the state and action spaces are small, then it is feasible for all the Q-values to be stored in memory. In general, however, the action-value function is approximated using a neural network, and the parameters are updated to minimize the loss given by 2.3. This approach is called "deep Q-network" or DQN (Mnih et al. (2013)). DQN is suitable for large state spaces (as demonstrated through Atari games in Mnih et al. (2013)), but still requires the action space to be discrete.

PPO

Proximal-policy optimization (PPO) is a model-free, on-policy policy gradient algorithm that improves data efficiency and robustness compared to existing algorithms (such as DQN or other policy gradient methods) (Schulman et al. (2017)). In contrast with other policy gradient methods, the PPO objective function is clipped according to a hyperparameter ϵ.

$$L(\theta) = \mathbb{E}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{2.4}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}$ is the probability ratio of actions under the new and old policies, and $\hat{A}_t$ is an estimator of the *advantage function* at time t, or the excess return above what was expected by the value function

$$\hat{A}_t := G_t - v_\pi(S_t) \tag{2.5}$$

By clipping the objective, the gradient updates are non-zero only for a region close to the previous policy, thus preventing policy changes that move too far away from the old policy. PPO has been applied successfully to both continuous and discrete action spaces.

Evolutionary Reinforcement Learning

Evolutionary reinforcement learning (ERL) is a model-free, off-policy hybrid algorithm that uses both policy gradient and evolutionary algorithm methods (Khadka and Tumer (2018)). ERL addresses problems found in many gradient-based RL algorithms, including temporal credit assignment (associating actions to rewards that may be separated by many time steps), exploration of the state and action spaces, and policy convergence sensitivity to hyperparameters. To do so, ERL maintains a population of actors with individual policies that interact with the environment. The population's experiences are recorded in a shared replay buffer, and a separate actor/critic pair learn from the replay buffer using policy gradient (such as DDPG or DQN). Periodically, the policy gradient-based actor is copied into the population and evolutionary algorithms are used to iterate a new "generation" of actors (by preferentially selecting high-performing actors). By using both gradient-based and gradient-free methods, ERL attempts to achieve both efficiency and policy diversity, making it a candidate for a variety of RL problems.

Pulse sequence design for Hamiltonian engineering has also been approached using the ERL algorithm in Peng et al. (2021), and achieved similar results to this RL implementation.

AlphaZero

AlphaZero is a model-based, off-policy, actor-critic algorithm that was originally developed to play chess, shogi, and Go (Silver et al. (2018)). There are two major components to the algorithm: self-play and neural network training. Because AlphaZero is a model-based algorithm, the agent is able to consider the state space without necessarily interacting with the environment. The agent plays many episodes against itself, using a neural network to estimate a policy $\pi(s)$ and a value function $v_\pi(s)$. During self-play, the agent rolls out many possible state-action trajectories, chooses an action that maximizes the expected reward, and collects data on the state, which actions it considered, and the final reward at the end of the episode. The collected data in turn is used to train the policy and value neural networks $\{\pi, v_\pi\}$.

Chapter 3

Methods

This section describes the simulation methods and RL algorithm used in this work. The implementation of both can be found at `https://github.com/wjkaufman/rl_pulse`.

3.1 Simulated Spin Systems

Isolated spin systems composed of a small number of spins can be easily simulated on a classical computer. Because the Hilbert space grows exponentially as the spins increase (2^N basis states for N spins and 2^{2N} matrix elements for operators), it quickly becomes impractical to simulate large numbers of spins. To balance the need to capture dipole interactions and to decrease simulation time, $N = 3$ spin-1/2 systems were used in the reinforcement learning algorithm, and $N = 4$ spin-1/2 systems were used to evaluate pulse sequences. The QuTiP Python package was used for all simulations (Johansson et al. (2013)).

In any real physical system, system-environment interactions are present and the spin system must be treated as open, but for simulation purposes the system is assumed to be isolated. Consequently, the dynamics are given by unitary time-evolution operators. In principle, simulations of open quantum systems could be used with reinforcement learning algorithms, and may in fact lead to better control by accounting for non-unitary system dynamics.

For each simulated N-spin system, the chemical shift strengths $\{\delta_j\}_j$ and the dipolar coupling strengths $\{d_{jk}\}_{j,k}$ are drawn from normal distributions

$$\delta_j \sim \mathcal{N}(\mu = 0, \sigma = 1) \tag{3.1}$$

$$d_{jk} \sim \mathcal{N}(\mu = 0, \sigma = d) \tag{3.2}$$

so that the energy scale of the system is defined by the standard deviation of the chemical shifts. The parameter d controls the regime of the spin system, where $d \gg 1$ is a "strongly coupled" spin system and $d \ll 1$ is "weakly coupled." We focus on strongly coupled spin systems, and set $d = 100$ unless otherwise specified. With the chemical shift and dipolar coupling strengths, the internal Hamiltonian H_{int} can be calculated (see equation 1.2).

Once H_{int} has been calculated, the free evolution propagator $U(\tau) := U_\tau$ can be calculated via matrix exponentiation (see equation 5.4) and the pulse operators $\{U_X, U_{\overline{X}}, U_Y, U_{\overline{Y}}\}$ can be calculated using equation 1.32.

Because the (random) choices of chemical shift and dipolar coupling strengths affect the dynamics of the spin system, an ensemble of spin systems was used both in the reinforcement learning algorithm and in evaluation of pulse sequences. Each spin system used different random chemical shift and dipolar coupling strengths drawn from the same distributions (i.e. the dipolar strength parameter d was the same for all systems). If experimental imperfections were also included, then each system also used different random errors. It was found that using ensembles of 50 spin systems reduced the variance in fidelity enough while keeping runtimes relatively short, so the reinforcement learning algorithm used 50 spin systems and evaluation used 100 spin systems.

To assess the performance of a pulse sequence for a Hamiltonian engineering problem, the pulse sequence propagator is compared to the target propagator given by

$$U_{\text{target}} = \exp\left\{-iH_{\text{target}}T\right\} \tag{3.3}$$

For this work, $H_{\text{target}} = 0$, so $U_{\text{target}} = \mathbb{1}$. Under ideal conditions, the two propagators should be "close" in some way. One measure of "closeness" is the following fidelity function for two unitary operators

$$\text{fidelity}(U, U_{\text{target}}) = \left| \frac{\text{Tr}\{U^\dagger U_{\text{target}}\}}{\text{Tr}\{\mathbb{1}\}} \right| \tag{3.4}$$

The fidelity approaches 1 as the unitaries get "closer" and the average Hamiltonian approaches the target Hamiltonian, and approaches 0 as the unitaries get further apart.

3.2 Reinforcement Learning Implementation

As explained previously, the reinforcement learning methodology is sufficiently general to be applied to a wide range of problems in a variety of different ways. The specific application of RL to the Hamiltonian engineering problem is described below.

3.2.1 Action Space and Representation

In an NMR spectrometer, the actions we can perform, or the "knobs" we can turn, are the rf field amplitudes in equation 1.7. The spectrometer's electronics limit the strength and the granularity of the rf field, but this still allows for a staggeringly large *continuous* action space. Instead of allowing for continuous control of the rf field[1], we only consider a finite set of pulses, each with an interval of free evolution after the pulse. The set of pulses we use is the set of $\pi/2$ rotations about the x- or y-axis along with a "null pulse"

[1]Which is precisely the action space used in gradient ascent pulse engineering (GRAPE) algorithms from Khaneja et al. (2005), although it does not use RL.

or delay. The action set is represented as $\{D, X, \overline{X}, Y, \overline{Y}\}$.

In the physical system, the actions correspond to unitary operators that determine the overall propagator. If the free evolution interval has length τ, then the D action corresponds to propagator

$$D \longrightarrow U_\tau = \exp\{-iH_{\text{int}}\tau\}$$

And the other pulse actions have corresponding propagators

$$\begin{aligned} X &\longrightarrow U_\tau U_X \\ \overline{X} &\longrightarrow U_\tau U_{\overline{X}} \\ Y &\longrightarrow U_\tau U_Y \\ \overline{Y} &\longrightarrow U_\tau U_{\overline{Y}} \end{aligned}$$

Where U_X is defined in equation 1.32. In the simulations, $\tau = 10^{-4}$ and the pulse width $t_p = 10^{-5}$ unless otherwise specified. These are in the short τ, short pulse width regimes ($d\tau \ll 1, dt_p \ll 1$). See figure 3.1 for the relationship between rf field amplitudes, unitary operators, and the actions.

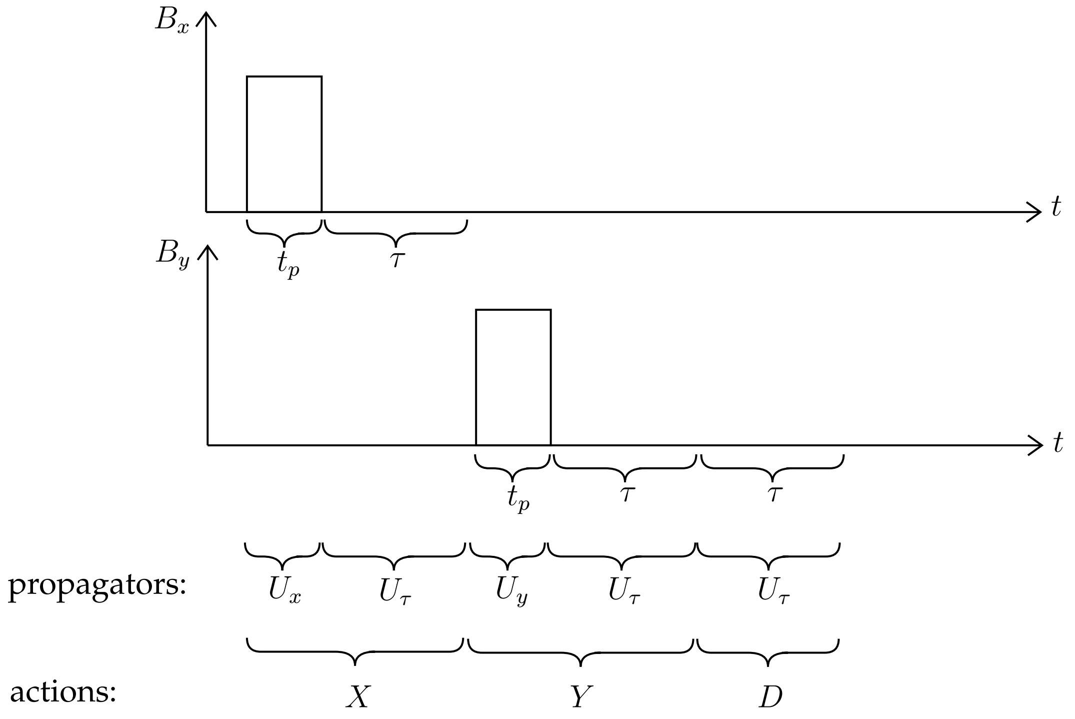


Figure 3.1: The rf field amplitudes, the unitary operators, and corresponding actions.

In principle, the actions could also be defined as pulses without a τ delay of free evolution and a delay action. However, in practice this leads to worse performance: for episodes with a fixed number of timesteps, the actions taken influence how much time elapses in the pulse sequence. As a result, the "optimal" performance is found to minimize the elapsed time by applying alternating pulses (e.g. $X, \overline{X}, X, \overline{X}, \dots$). The unitary operator fidelity generally improves as the time interval decreases, so this is an unhelpful solution when pursuing high fidelity at longer time intervals.

3.2.2 State Space and Representation

The "state" can refer to the physics notion of state–such as the density operator ρ of a quantum system–or the RL notion of state–a particular context in which the agent must choose an action. They are closely related but not the same. The "physical" state space is comprised of every reachable state $\rho(t)$ of the spin system given an initial thermalized state, the free evolution dynamics from H_{int}, and the control Hamiltonian given by the rf field H_{rf}. In an RL context, however, the agent does not observe the complete state of the system at each point in time. Instead, the agent only knows the sequence of actions that have been performed on the system and any measurements that have been made on the system. Therefore, a more representative state space for the RL problem is all possible sequences of actions performed on the system.[2] This is more "realistic" in the sense that the agent only knows what the experimenter would know, without getting any additional information "for free."

Furthermore, Hamiltonian engineering (and unitary gate engineering more generally) should be independent of the state of the system. That is, the engineered Hamiltonian (or gate) should have high fidelity for a complete basis of states, not for any one state in particular. This further justifies the state representation as a sequence of actions.

3.2.3 Reward Function

As mentioned above, the fidelity function takes on values between 0 and 1, where values close to 1 are desirable. However, a tenfold improvement in performance doesn't correspond to a tenfold increase in fidelity (for instance a fidelity of 0.9 compared to 0.99). A suitable reward function is then calculated by

$$r = -\log\big(1 - \text{fidelity}(U_{\text{actual}}, U_{\text{target}}) + \epsilon\big) \tag{3.5}$$

where $\epsilon > 0$ ensures the reward isn't infinite.

3.2.4 AlphaZero Algorithm

As mentioned in the introduction, the AlphaZero algorithm has two main components: self-play and neural network training. The original publication and its supple-

[2]Making measurements on the system could also be considered an action (measurement affects the physical state), and the resulting measurement outcome could be included in the state (Porotti et al. (2019)). This possibility, however, is not explored here.

mentary materials have a complete description of the algorithm (Silver et al. (2018)), but pseudocode is presented below as well.

During self-play, the agent iteratively constructs a pulse sequence by performing Monte Carlo Tree Search (MCTS) from the current state. From state s_i (which is represented by the sequence of actions performed so far, $s_i = (a_0, \ldots, a_i)$), the agent stochastically samples actions based on the prior probability from the policy π, the number of times the agent has already "visited" that action, and the average estimated reward (or value) of that branch of the tree. The agent is more likely to select actions that have a higher prior probability, a lower visit count, and a higher average reward. The exact formula for selection is given in algorithm 1, and is a variant of the predictor upper confidence tree (PUCT) algorithm from Rosin (2011).

The agent performs many rollouts of selecting actions, estimating the rewards, and updating visit counts and reward estimates along the particular sequence of actions. After a certain number of rollouts, the agent finally picks a particular action to perform by sampling from the empirical distribution of visit counts. See figure 3.2 for a visual depiction of the MCTS process.

Algorithm 1: Monte Carlo Tree Search (MCTS) for selecting the next action.

```
Data: Current state $s = (a_0, \ldots, a_{i-1})$, policy and value networks $(\pi, v)$, number of
      rollouts $n_{\text{roll}}$, Dirichlet noise parameter $\alpha$, noise weight $w \in [0, 1]$, PUCT
      parameters $c_{\text{base}}, c_{\text{init}}$
Result: Probabilities $\{p_k\}_k$ of selecting action $a_k$ from state $s$, and randomly
        sampled action $a$
Initialize root node associated with current state $s$ ;
Create child nodes of the root node corresponding to each action in the action
  space ;
Initialize set of statistics
  $\{N(s, a_k) = 0, W(s, a_k) = 0, Q(s, a_k) = 0, P(s, a_k) = \pi(s, a_k)\}_k$ for each child node
  $a_k$, where $N$ is the visit count, $W$ is the total value, $Q$ is the mean value, and $P$ is
  the prior probability from the policy network $\pi$ ;
Sample multivariate Dirichlet noise from Dir$(\alpha)$ and add it to the priors
  $\{P(s, a_k)\}_k$ ;
for $n_{roll}$ tree rollouts do
    Set $s_0 \leftarrow s$, $i \leftarrow 0$ ;
    while $s_i$ is not a leaf node do
        Select a child node of $s_i$ associated with action $a_k$ by maximizing
          $Q(s_i, a_k) + U(s_i, a_k)$ where
              $U(s, a) = C(s)P(s, a)\sqrt{N(s)}/(1 + N(s, a))$
          And $C(s) = \log((1 + N(s) + c_{\text{base}})/c_{\text{base}}) + c_{\text{init}}$ ;
        Let $s_{i+1} \leftarrow s_i + (a_k)$ and $i \leftarrow i + 1$ ;
    if the leaf node $s_L$ represents a pulse sequence with the desired length then
        Calculate the fidelity of the pulse sequence in spin system ensemble ;
        Calculate the reward $r$ from fidelity ;
    else
        Evaluate the leaf node $s_L$ using $(\pi, v)$ to create child nodes of $s_L$ with
          initialized statistics
          $\{N(s_L, a_k) = 0, W(s_L, a_k) = 0, Q(s_L, a_k) = 0, P(s_L, a_k) = \pi(s_L, a_k)\}_k$, and
          an estimated reward $r = v(s_L)$ ;
    For each of the nodes $s_i$ visited from $s_0$ to $s_L$, update the visit counts
      $N(s_i, a_i) = N(s_i, a_i) + 1$ and the values
      $W(s_i, a_i) = W(s_i, a_i) + r, Q(s_i, a_i) = \frac{W(s_i, a_i)}{N(s_i, a_i)}$ ;
Calculate the estimated probability $p_k \leftarrow N(s, a_k)/N(s)$ for each action $a_k$ ;
Randomly sample an action $a_k$ from the distribution $\{p_k\}$ ;
```

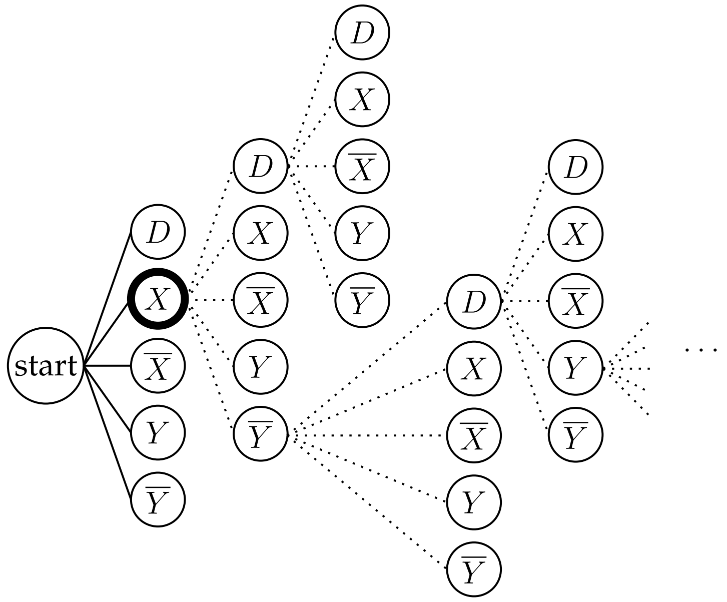


Figure 3.2: An example Monte Carlo Tree Search (MCTS) starting with root state $s = (X)$. The first rollout explores the action branch D, creating each of the child nodes for D. The next rollout instead explores $\overline{Y}$ starting from (X). The final two rollouts depicted choose $(\overline{Y}, D)$ and $(\overline{Y}, D, Y)$ from (X). The rollouts continue for a set number of rollouts, and the next action selected from (X) is sampled proportionally according to visit counts of each of the children nodes.

The agent will repeat the process of MCTS and action sampling until it has constructed a pulse sequence with the desired length. The agent records search statistics from each step of the pulse sequence construction: the state s_i, the empirical probability distribution of actions explored from MCTS, and the final reward r of the pulse sequence from simulation. These statistics are then used to train the policy and value neural networks

(π, v).

Algorithm 2: Pulse sequence construction via self-play.

```
Data: Pulse sequence length $T$
Result: Search statistics $S$
Initialize empty search statistics $S \leftarrow \{\}$ and initial state $s_0 \leftarrow ()$ ;
while Pulse sequence length is less than $T$ do
    Perform MCTS from state $s_i$ to get probabilities of selecting action $k$ ($\{p_k\}^{(i)}$)
      and randomly sample the next action $a_i$ ;
    Append $(s_i, \{p_k\}^{(i)})$ to the search statistics $S$ ;
    Apply action $a_i$, get updated state $s_{i+1} \leftarrow (a_0, \ldots, a_i)$ ;
Calculate the reward $r$ for the final pulse sequence ;
Include the reward $r$ in each element in the search statistics $S$, so
  $S = \{(s_i, \{p_k\}^{(i)}, r)\}$ ;
```

As the agent continually generates search statistics from self-play, the policy and value neural networks (π, v) in turn are trained on the collected data. The loss function to train the neural networks is

$$L(\theta) = (r_i - v_\theta(s_i))^2 - \pi_\theta(s_i) \cdot \log \mathbf{p_i} + c||\theta||^2 \tag{3.6}$$

The first term is the squared error of the reward estimates, the second is the cross-entropy loss of the policy estimates, and the third is L_2 regularization of the neural network parameters. The constant c controls the weight of L_2 regularization (in this work $c = 10^{-6}$).

The two processes of self-play and training are done simultaneously, with multiple agent processes constructing pulse sequences and generating data that is then used to train the policy and value networks. In turn, the updated network parameters θ are shared with the agents.

3.2.5 Neural Network Architecture

As in the original AlphaZero implementation, the policy and the value networks were combined into a single neural network that maps a state s to *both* a probability distribution over actions $\pi(s)$ and a value estimate $v(s)$. This is written as $f_\theta(s) = (\pi(s), v(s))$, where θ are the neural network parameters. This adds an additional degree of regularization, because any parameters that are used for both the policy and value estimates must minimize the loss for both the policy and the value estimates, preventing overfitting. The combined network includes a shared set of layers that transform the state to an intermediate representation, and two separate sets of layers (called "heads") that return the policy and value estimates. See figure 5.1 for the complete neural network architecture.

The state is represented by the sequence of actions performed on the system and an additional `START` "action." Because the action space is discrete ($\{D, X, \overline{X}, Y, \overline{Y}\}$), each action is represented by a one-hot vector[3]. For example, the `START` action corresponds to the vector $[1, 0, 0, 0, 0]$, a delay D corresponds to the vector $[0, 1, 0, 0, 0, 0]$, and $\overline{Y}$ corresponds to $[0, 0, 0, 0, 0, 1]$. The sequence of pulses $[\overline{X}, Y, D]$ is then represented by the 2D-array

$$\begin{gathered}[[1,0,0,0,0,0],\\ [0,0,0,1,0,0],\\ [0,0,0,0,1,0],\\ [0,1,0,0,0,0]]\end{gathered}$$

Because the state is represented by a sequence of actions, the size of the state depends on the length of the pulse sequence. The policy head outputs a 1×5 vector of probabilities associated with each of the possible actions. The value head outputs a single scalar for the estimated reward of the pulse sequence so far.

[3]A one-hot vector has a single entry equal to one, and all other entries are zero.

The state representation–as a sequence of actions–lends itself to using a recurrent neural network architecture. A gated recurrent unit (GRU) was used, though a long-short term memory (LSTM) layer performed similarly (Cho et al. (2014); Hochreiter and Schmidhuber (1997)). Both GRUs and LSTMs are capable of mapping variable-length input to a fixed-length output space, which is precisely what is needed when mapping the actions we have performed so far to the estimated policy and value of that state.

3.3 AHT-Constrained Search

Because actions are explored and selected via tree search, additional constraints can be added to prune branches of the tree that are unlikely to yield high rewards. This is akin to adding "rules of the game" that determine which actions can or cannot be applied. Thankfully, AHT gives us a straightforward constraint so that pulse sequences have the desired lowest-order average Hamiltonian $\overline{H}^{(0)}$.

When the target Hamiltonian $H_{\text{target}} = 0$, all interactions must be decoupled. If the toggling frame is defined so that $\widetilde{I}_z(t) = I_z$ and $\widetilde{I}_z(t) = -I_z$ for equal durations, then the lowest-order average $\overline{I}_z^{(0)} = 1/2(I_z - I_z) = 0$. In general, any term containing I_z will average to zero to lowest order when $\widetilde{I}_z$ is toggled for equal time along opposite axes (i.e. equal time on $+x$ and $-x$, $+y$ and $-y$, and $+z$ and $-z$-axes). Doing so would decouple the chemical shift term H_{CS} to lowest order.

We saw in section 1.2.2 that the dipolar interaction term is decoupled when $I_z^{(j)} I_z^{(k)}$ is toggled to $I_z^{(j)} I_z^{(k)}, I_y^{(j)} I_y^{(k)}$, and $I_x^{(j)} I_x^{(k)}$ for equal times. Note that if $\widetilde{I}_z = -I_z$, then $\widetilde{I}_z^{(j)} \widetilde{I}_z^{(k)} = (-I_z^{(j)})(-I_z^{(k)}) = I_z^{(j)} I_z^{(k)}$, so the overall term remains unchanged. Therefore the dipolar interaction is decoupled to lowest order when $I_z^{(j)} I_z^{(k)}$ is toggled for equal time along the x, y, and z-axes.

To decouple all interactions, both constraints above must be satisfied, which is achieved when I_z is toggled along each of the six axes for equal time ($\pm x, \pm y, \pm z$). By keeping track of the toggled operator $\widetilde{I}_z$ while constructing the pulse sequence, it is possible to prune branches of the tree that would violate this constraint. Choi et al. (2020) introduces a formal matrix representation for the $\widetilde{I}_z$ transformation and derives a set of constraints on pulse sequences for Hamiltonian engineering, some of which are mentioned above.

Chapter 4

Results

To analyze the performance of the AlphaZero algorithm for constructing pulse sequences, different sets of algorithm hyperparameters and system parameters were tested. AlphaZero was first run with no additional constraints to the tree search (i.e. the state space contains all possible sequences of pulses), and the simulated spin systems were idealized (i.e. delta-pulses and no experimental errors). Additional constraints were then added to the tree search while keeping idealized spin system simulations. Finally, experimental imperfections were introduced to the simulations.

As a preliminary consistency check, the AlphaZero algorithm was run with identical settings 10 times, and the resulting distribution of rewards during training was recorded. In figure 4.1, it is evident that most of the runs perform similarly, but a few significantly outperform the rest, achieving over an order of magnitude improvement in fidelity. Furthermore, most of the runs plateau after about 5000 training steps, indicating the policy network converged to a particular policy. The inconsistent performance across identical runs could be improved by further tuning the algorithm hyperparameters, such as the amount of noise added to the policy, increasing the exploration rate in the MCTS, or adjusting the relative rate of training to data collection.

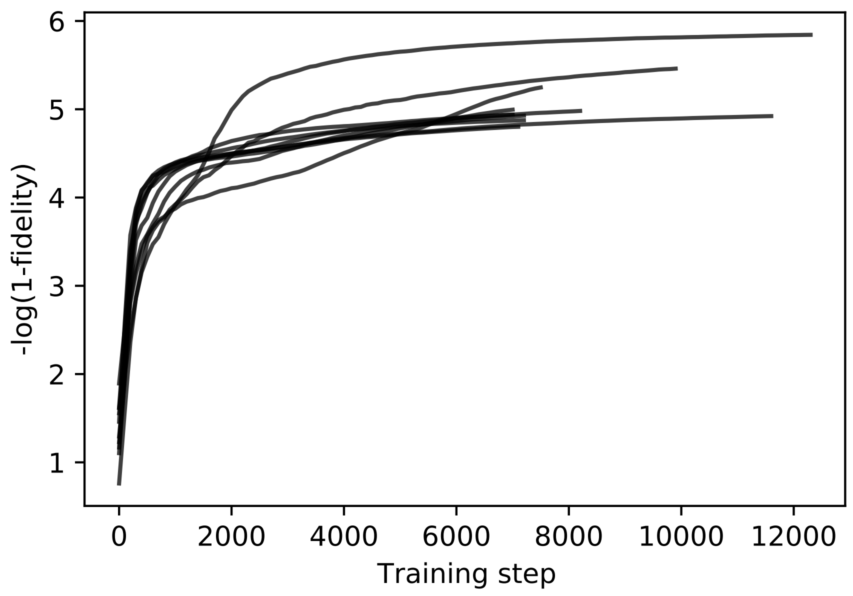
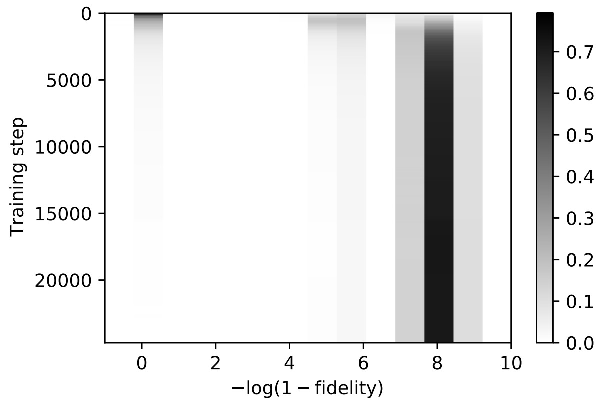
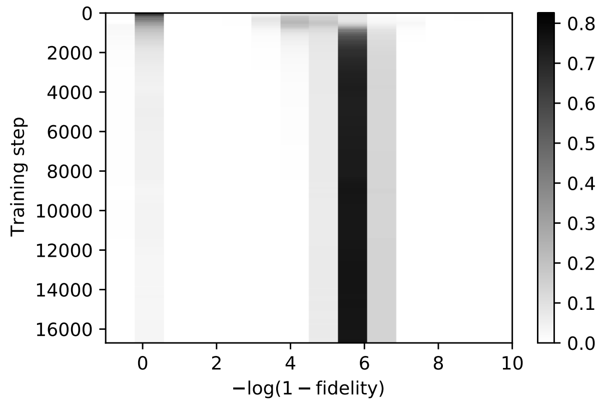


Figure 4.1: The mean reward ($-\log(1-\text{fidelity})$) seen during training for 10 identical runs of the AlphaZero algorithm searching for 24τ sequences. The simulations included finite pulse width with no experimental errors. Most runs converge to similar mean fidelities, but a few runs have orders of magnitude better performance.

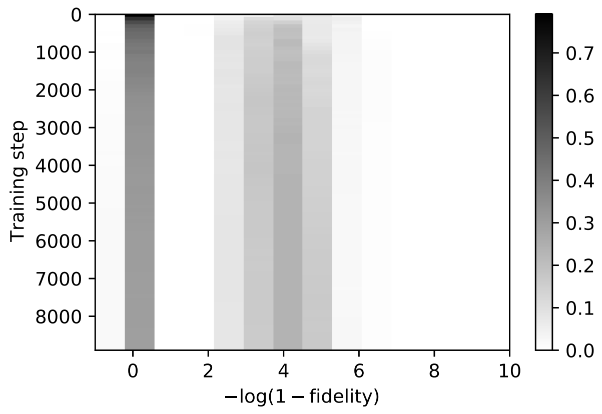
For the completely unconstrained search and idealized spin system simulations, AlphaZero converges to an effective policy only for the 12-pulse sequence (figure 4.2a). AlphaZero tries random pulse sequences with low fidelity early in training, but quickly "learns" to construct pulse sequences with fidelity $\approx 1-10^{-8}$.



(a) 12τ sequence



(b) 24τ sequence



(c) 48τ sequence

Figure 4.2: Distribution of rewards ($-\log(1-$ fidelity$)$) during AlphaZero training. No constraints were applied to the tree search, and the simulated spin systems were idealized.

However, for longer pulse sequences with 24 or 48 actions, unconstrained search is unable to converge to similarly effective policies as for shorter sequences. For the 24-pulse sequence (figure 4.2b), the policy after 2000 training steps constructs pulse sequences with fidelity $\approx 1-10^{-6}$, about two orders of magnitude worse than for the 12-pulse sequence. And for the 48-pulse sequence (figure 4.2b), the policy constructs pulse sequences with even lower fidelities. Even after 8000 training steps, the policy still constructs pulse sequences with fidelity ≈ 0 over 30% of the time.

The worsening performance as the pulse sequence length increases is understandable. For an n-pulse sequence with a possible actions, there are a^n possible pulse sequences, and $\frac{a^{n+1}-1}{a-1}$ possible states[1]. If $n = 12$, then there are over 244 million pulse sequences and over 300 million states. But for $n = 24$, then the number of pulse sequences jumps to 6×10^{16}, and for $n = 48$ there are 3.6×10^{33}. Because AlphaZero begins training *tabula rasa*, none of the states in the astronomically large state space are preferred by the agent, so it begins with a purely random search. Furthermore, rewards are only received once an *entire pulse sequence* has been constructed. The sparse reward signal (which is the sole driver of learning in RL algorithms) therefore makes it difficult for the agent to associate actions with rewards and learn an effective policy. This is known as the "credit assignment problem" and is an active area of research both in theoretical and applied RL (Arumugam et al. (2021)).

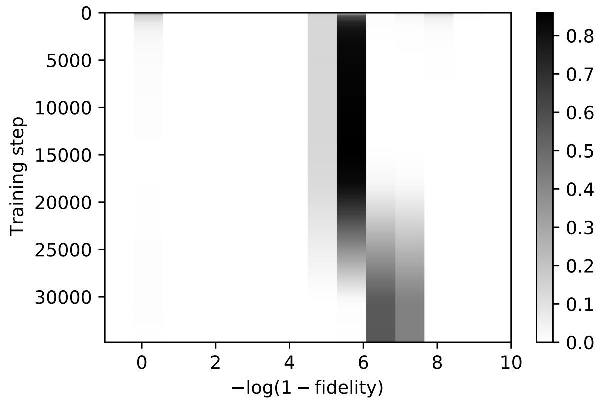
For example, suppose you needed to pass a test with 100 questions, and could re-take the test as many times as you want. It would take many more tries to pass if the only feedback you received was your overall score on the test, as opposed to your score on each question.

There is not an immediately obvious way to provide additional rewards to the agent during training[2], but there are ways to constrain AlphaZero's tree search to only consider subsets of the state space. As described in section 3.3, the lowest-order term in the average Hamiltonian can be set to the desired Hamiltonian by requiring that the toggling frame spend equal durations in different orientations. To decouple all interactions, the toggling frame must spend equal time along each axis (i.e. I_z must be toggled to $\pm I_x, \pm I_y, \pm I_z$ for equal times). By adding this constraint to the tree search, all pulse sequences at least have the correct average Hamiltonian to lowest order. This does not address the cyclic requirement for those pulse sequences (i.e. the toggling frame coincide with the lab frame at the end of the pulse sequence), the higher-order terms in the average Hamiltonian, nor experimental imperfections.

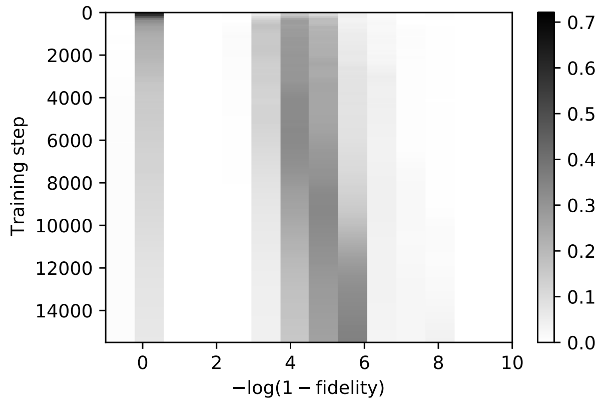
[1]This can be seen by counting how many subsequences of length k there are (a^k), and adding these for $k = 1, \ldots, n$.

[2]From an experimental perspective, we only care about the fidelity of the pulse sequence overall, not at intermediate times.

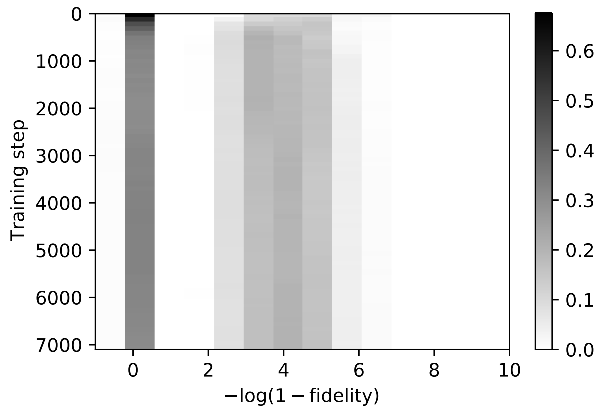
Figure 4.3 shows the distributions of pulse sequence fidelities for the $12\tau, 24\tau$, and 48τ sequences during training. Despite restricting the state space in a way that should guarantee higher fidelities, the fidelities are actually lower for the 12τ and 24τ sequences, and are comparable for the 48τ sequence. These discrepancies are likely due to the inherent randomness of the algorithm (as seen in figure 4.1), but more careful analysis is needed to understand the observed differences.



(a) 12τ sequence



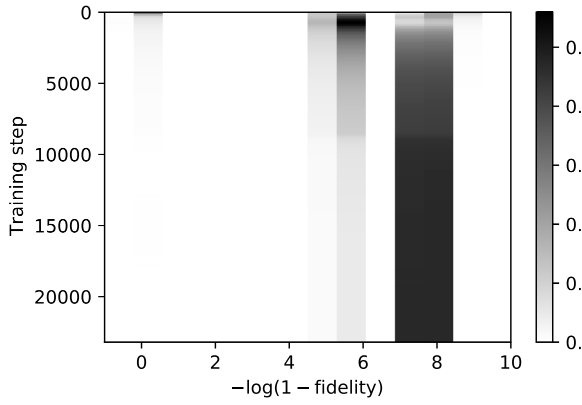
(b) 24τ sequence



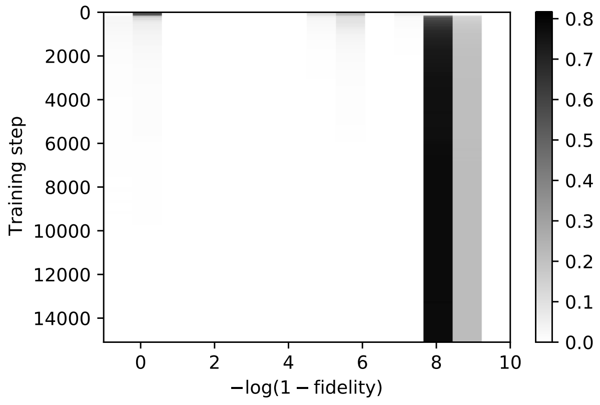
(c) 48τ sequence

Figure 4.3: Distribution of rewards ($-\log(1-\text{fidelity})$) during AlphaZero training. Lowest-order AHT constraints were applied to the tree search, and the simulated spin systems were idealized.

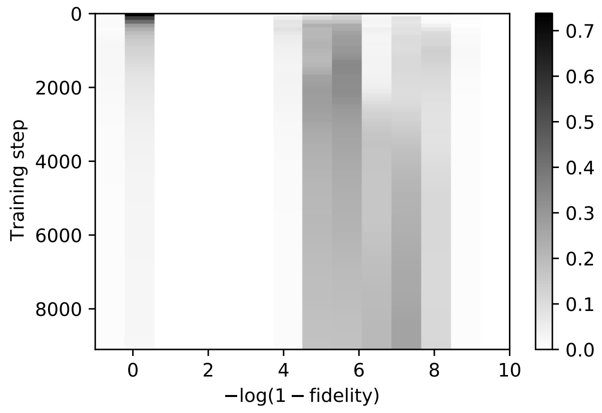
Because the lowest-order average Hamiltonian constraint didn't clearly improve performance, an even stronger constraint was added to the tree search to further restrict the state space: to decouple interactions to lowest order every 6τ, instead of decoupling interactions for the pulse sequence overall. There are 200 sequences of length 6τ that will decouple all interactions to lowest order, significantly fewer than $5^6 = 15625$ possible sequences without any constraints. The resulting fidelities (figure 4.4) are significantly higher for the 24τ and 48τ sequences during training, comparable to the shorter 12τ sequence.



(a) 12τ sequence



(b) 24τ sequence



(c) 48τ sequence

Figure 4.4: Distribution of rewards ($-\log(1 - \text{fidelity})$) during AlphaZero training. Lowest-order AHT constraints and refocusing all interactions every 6τ were applied to the tree search, and the simulated spin systems were idealized.

The 6τ refocusing constraint, while empirically beneficial for the AlphaZero tree search, raises some questions. By introducing this constraint, is the algorithm excluding high-fidelity pulse sequences that do not follow this constraint? The CORY48 pulse sequence in fact does not refocus *all* interactions to lowest order until the end of the pulse sequence, and its fidelity is much higher than any pulse sequence found using AlphaZero. Instead, it decouples dipolar interactions every 9τ, which is a more relaxed constraint.

During training, pulse sequences with fidelity above a given threshold are recorded, and the highest-fidelity pulse sequence is chosen as a “candidate” for further evaluation (it would be impractical to evaluate each of the thousands of pulse sequences constructed by AlphaZero). The AlphaZero algorithm was originally designed to play board games, where the trained policy function defined on the entire state space is more important than the specific sequence of moves played in training games. In contrast, for designing pulse sequences, the final policy is less important than identifying a single pulse sequence with high fidelity and robustness to errors. The policy is only a tool for exploring the state space for high-fidelity pulse sequences.

4.1 Robustness Evaluation

Although it is straightforward to simulate spin systems under ideal conditions without any experimental imperfections, this never happens in reality. Therefore, it is important to consider how the pulse sequences perform in the presence of errors and determine their robustness. The errors considered below include pulse rotation errors, phase transient errors, and resonant offset errors.

Pulse rotation errors, either under- or over-rotating the spins, are common in NMR and electron spin resonance (ESR) systems due to limited precision in both the rf-field strength and the pulse width t_p. If the pulse strength or duration is too high (or too low), then the toggling frame will be rotated greater (or less) than $\pi/2$ and the average Hamiltonian will not necessarily match the target Hamiltonian. See figure 4.5 for a visual depiction of rotation errors for a single spin.

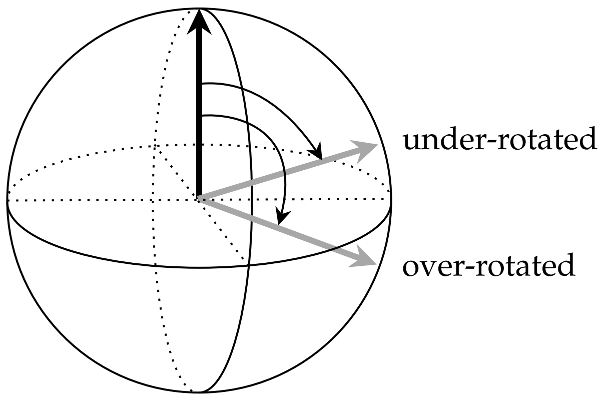


Figure 4.5: Pulse rotation errors for a single spin.

Phase transient errors are another pulse-related error that are present in experimental systems. When applying a pulse, the non-zero impedance in the circuit will induce small additional pulses in quadrature to the intended pulse when the rf field rises and falls. For example, applying an X pulse will also apply slight rotations about the y axis when the B_X field rises and falls. See figure 4.6 for an ideal pulse compared to a pulse with phase transients. To simulate phase transient errors, each of the $X, Y, \overline{X}$, and $\overline{Y}$ actions were modified to include slight rotations along $y, -x, -y$, and x immediately before and after the pulse. The rotation angle was parameterized as a fraction of $\pi/2$.

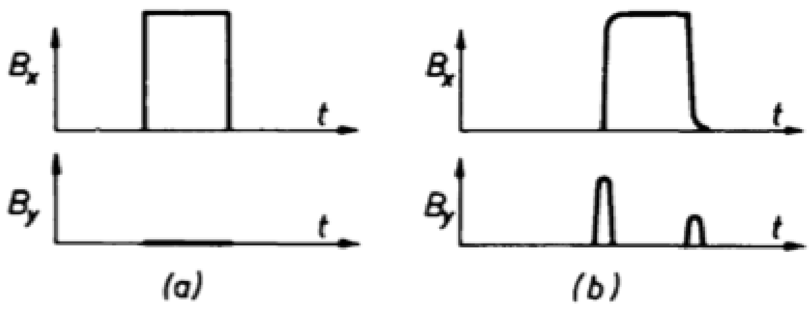


Figure 4.6: An ideal pulse (a) compared to a pulse with phase transients (b). Illustration from Haeberlen (1976).

Finally, resonance offset errors occur when the rf field is not matched to the Larmor frequency of the spins. The rotating frame therefore does not rotate *precisely* with the spins, but at an offset frequency.

The highest-fidelity pulse sequences from the $12\tau, 24\tau$, and 48τ sequence searches[3] with AHT and 6τ refocusing constraints were evaluated for robustness against the experimental imperfections introduced above. To compare each of the different-length sequences on the same footing, each pulse sequence was repeated over a total duration of 288τ. That is, the 12τ sequence was repeated 24 times, the 24τ sequence was repeated 12 times, the 48τ repeated 6 times, and CORY48 (a 72τ sequence) repeated 4 times.

For small rotation errors ($< 1\%$ of a $\pi/2$-pulse) the fidelity for all pulse sequences declines dramatically (see figure 4.7). This is not entirely unexpected however: the spin systems during training were idealized, so no errors of any kind were included. As a result, there was no reward signal for constructing *robust* pulse sequences, only pulse sequences that did well under ideal conditions.

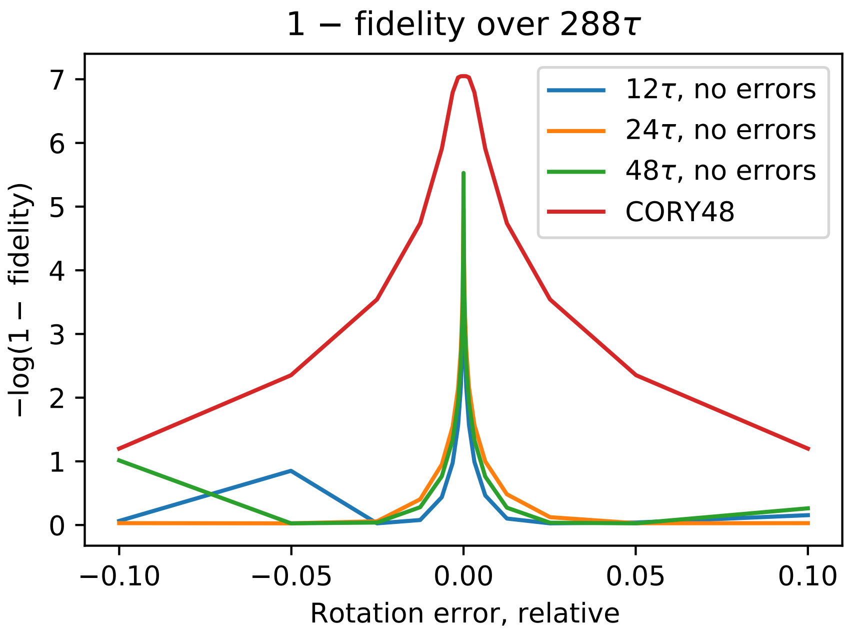


Figure 4.7: Robustness against rotation errors, relative to a $\pi/2$-pulse. Training simulations included no errors, and evaluation simulations included only rotation errors. The pulse sequences identified using AlphaZero have very poor robustness to rotation errors, while the CORY48 sequence (which was designed using AHT to be robust to such errors) has high fidelity for a much broader range of rotation errors.

[3]See the appendix for an explicit presentation of these pulse sequences.

Similarly, the pulse sequences are much less robust to phase transient error (figure 4.8), or resonance offset error (figure 4.9). Interestingly, the 12τ sequence seems to be highly robust to phase transient error, but the fidelity is comparably worse overall.

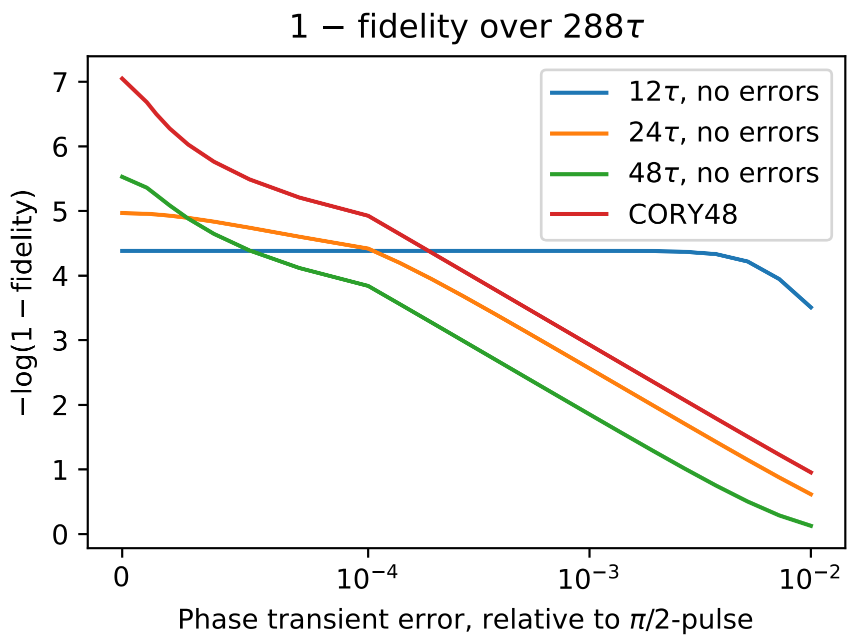


Figure 4.8: Robustness against phase transient errors, where the quadrature rotations are relative to a $\pi/2$-pulse. Training simulations included no errors, and evaluation simulations included only rotation errors.

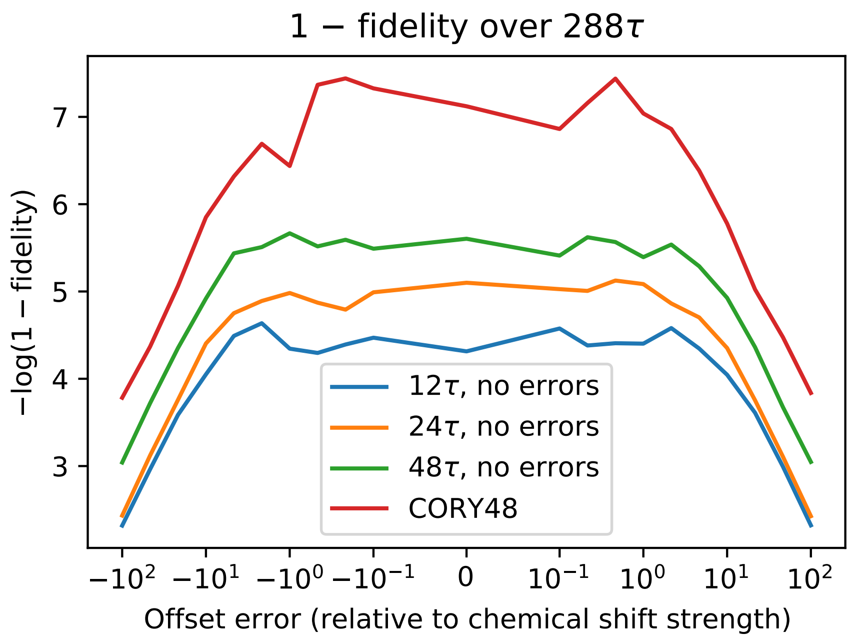


Figure 4.9: Robustness against resonant offset error, relative to the chemical shift strength. Training simulations included no errors, and evaluation simulations included only rotation errors.

To search for *robust* pulse sequences, errors were intentionally introduced into the spin systems during training. First, only rotation errors were included by sampling a random rotation error $\epsilon_r \sim \mathcal{N}(\mu = 0, \sigma = .01)$ for each spin system during training. Then instead of having perfect $\pi/2$-pulses, all pulses in a particular spin system rotate the spins by $\frac{\pi}{2}(1 + \epsilon_r)$. The resulting pulse sequences were significantly more robust to rotation errors (figure 4.10) compared to training without any errors (figure 4.7).

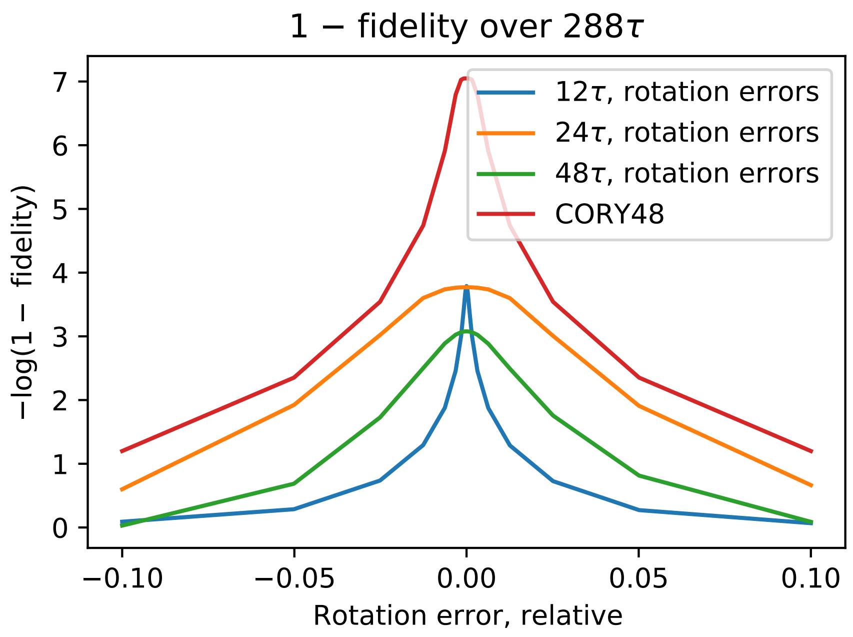


Figure 4.10: Robustness against rotation errors, relative to a $\pi/2$-pulse. Training simulations included only rotation errors, and evaluation simulations included only rotation errors. The pulse sequences identified from training with rotation errors have improved robustness to rotation errors, as compared with training in the absence of rotation errors.

Finally, the AlphaZero algorithm was run with multiple experimental imperfections: rotation error $\epsilon_r \sim \mathcal{N}(0, .01)$, phase transient error $\epsilon_{pt} \sim \mathcal{N}(0, 10^{-4})$, and resonance offset error $\epsilon_o \sim \mathcal{N}(0, 10^1)$. The policy function was rewarded for high-fidelity pulse sequences in the presence of those errors, and consequently should identify pulse subsequences robust to multiple errors. For a range of rotation, phase transient, and resonance offset error (figures 4.11 to 4.13), the pulse sequences from training *with* simulated errors are more robust than those from training with no errors.

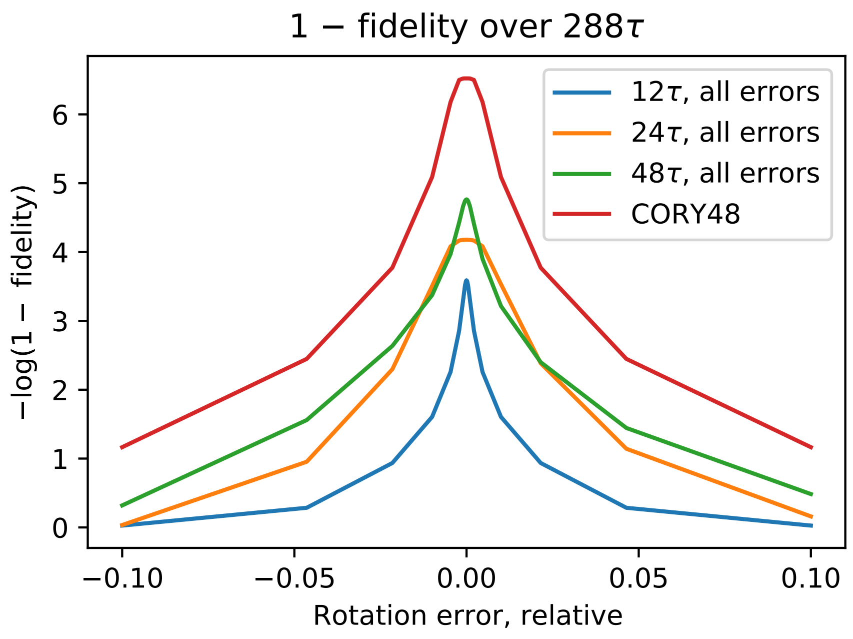


Figure 4.11: Robustness against pulse rotation errors. Training simulations included rotation, phase transient, and resonance offset errors ("all errors"), and evaluation simulations included all errors.

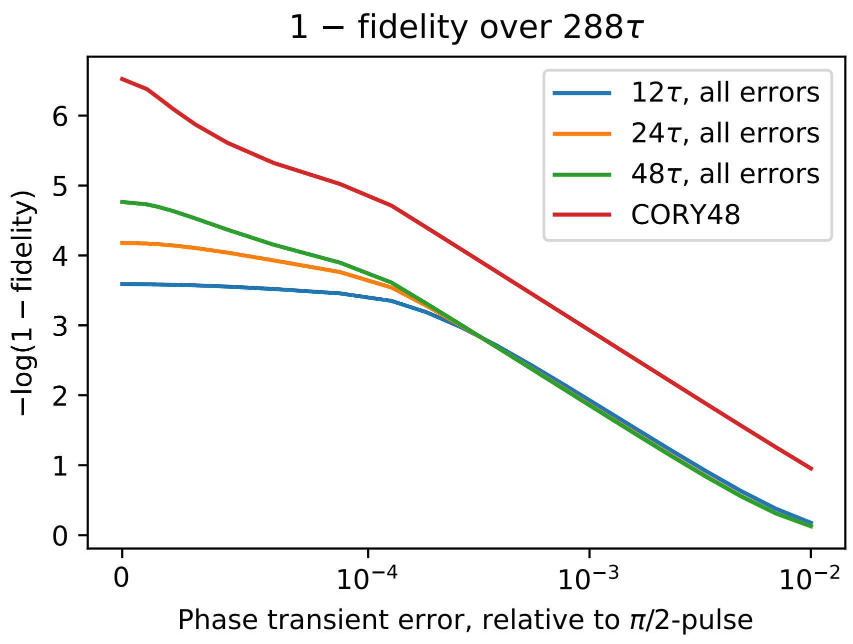


Figure 4.12: Robustness against phase transient errors, where the quadrature rotations are relative to a $\pi/2$-pulse. Training simulations included rotation, phase transient, and resonance offset errors ("all errors"), and evaluation simulations included all errors.

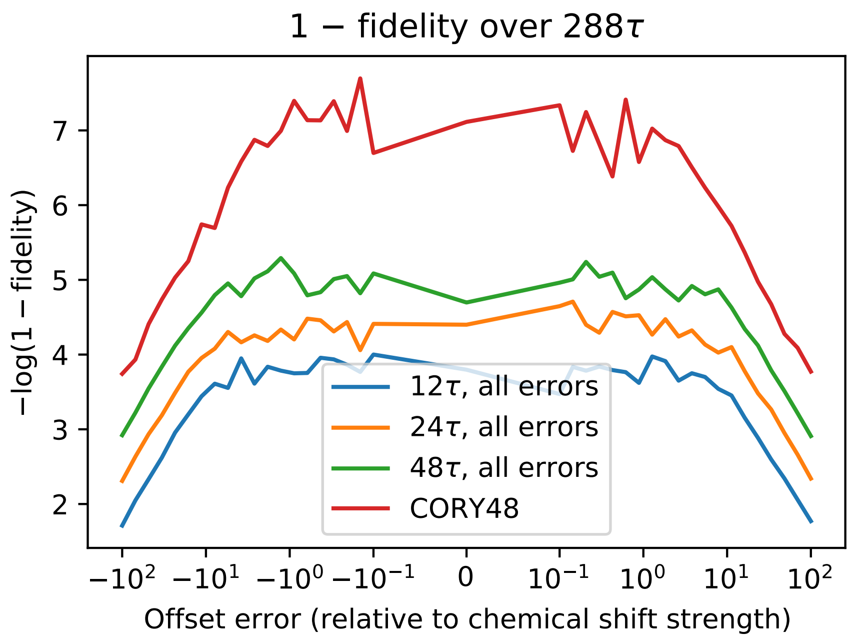


Figure 4.13: Robustness against resonance offset errors, relative to chemical shift strength. Training simulations included rotation, phase transient, and resonance offset errors (“all errors”), and evaluation simulations included all errors.

Despite the improved robustness by including errors in training, the resulting pulse sequences still have significantly lower rewards (and thus lower fidelity) than the CORY48 pulse sequence in simulated spin systems. This holds true across different possible errors and at every magnitude of error tested. In short, the reinforcement learning approach tried in this work is not as effective as the AHT-based analytical process used to design CORY48.

4.2 Experimental Validation

In addition to computational simulations, the best-performing 48τ pulse sequence was tested against the CORY48 pulse sequence in a solid-state NMR spectrometer. An adamantane sample was used.

While the fidelity of a unitary matrix is straightforward to calculate in simulation, measuring the fidelity in experiment is much more difficult, as it would require evaluation on a complete basis of states (i.e. quantum process tomography). Instead, the fidelity is qualitatively approximated by the average correlation C_{avg} (Peng et al. (2021))

$$C_{\text{avg}} = (C_{XX}C_{YY}C_{ZZ})^{1/3} \tag{4.1}$$

where the correlation function $C_{XX}(t)$ is the expected signal from initializing the state along X and measuring magnetization along X.

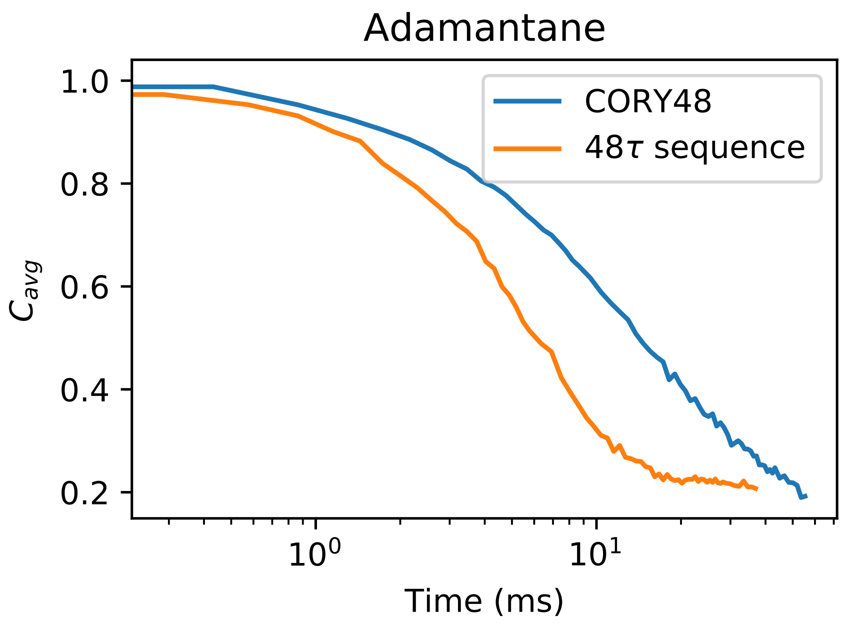


Figure 4.14: The average correlation in an adamantane sample for the 48τ-pulse sequence identified using AlphaZero with all errors in training, and for the CORY48 pulse sequence.

The experimental correlation functions (figure 4.14) further support the computational results: the RL-based 48τ pulse sequence does decouple interactions, but not as effectively as the CORY48 pulse sequence.

Chapter 5

Conclusion and Future Work

Hamiltonian engineering encompasses a variety of important problems in quantum physics and quantum control. For spin systems in particular, the ability to decouple dipolar interactions is desirable for improving spectroscopy and increasing spin coherence times. Existing approaches to Hamiltonian engineering via average Hamiltonian theory have been effective. The CORY48 pulse sequence was designed analytically using AHT to decouple all interactions to second order, and is robust to common errors in spin systems (Cory et al. (1990)). However, there is still room for improvement in increasing coherence times, and it is unclear whether analytical AHT-based methods can be taken further because the higher-order terms in the Magnus expansion become increasingly computationally expensive to calculate. Other approaches to Hamiltonian engineering may prove to be more promising.

Reinforcement learning (RL) algorithms have expanded dramatically over the past decade into a broad array of problems. The AlphaZero algorithm was used to develop pulse sequences for decoupling all interactions ($\overline{H} = 0$) in a spin system. The RL algorithm by itself was only effective at identifying high-fidelity pulse sequences for short sequences. After introducing additional constraints to the algorithm's tree search, high-fidelity pulse sequences were found for all sequence lengths considered. And by introducing common errors in the spin system simulations during training, the algorithm found pulse sequences that were robust to those errors. However, the RL-based pulse sequences did not outperform the CORY48 pulse sequence in computational simulations or experiment.

Although the current RL implementation failed to outperform existing analytical approaches, there are several factors that may be considered for improvement. First, the hyperparameters used for the AlphaZero algorithm were largely unchanged from the original publication, with the exception of the exploration noise added during tree search. Hyperparameter tuning specifically for Hamiltonian engineering may improve the algorithm's performance and increase the fidelities of the resulting pulse sequences. In particular, changing the exploration rate and the training rate relative to data collection from tree search may be fruitful paths to explore.

In addition to optimizing the RL algorithm, using a different set of constraints in the tree search may be beneficial. The 6τ total decoupling constraint that was implemented is not followed in other pulse sequences, including CORY48, so it is possible that decoupling all interactions every 6τ is too strong a constraint on pulse sequences. Two obvious constraints to consider are decoupling dipolar interactions only every 9τ, and requiring the pulse sequence use equal numbers of $X, \overline{X}, Y, \overline{Y}$ pulses (both of which are done in the CORY48 sequence). Satisfying higher-order terms in the average Hamiltonian with additional constraints would also be beneficial.

There are other interesting avenues to consider to improve the RL algorithm. One of those avenues is curriculum learning, where the agent learns to solve progressively more difficult tasks (Narvekar et al. (2020)). For Hamiltonian engineering, this may be done by starting with short pulse sequences, and progressively constructing longer pulse sequences.

The spin system simulations for training in the RL algorithm and evaluation of pulse sequences were in the short-τ delay regime ($d\tau \ll 1$), which improves the ability of the pulse sequences to decouple interactions. However, this is not always realized in experiment, where $d\tau \approx 0.1$ or greater. It would be worth exploring training with longer τ delays and see how both computational and experimental results change.

Although the primary focus for this work is Hamiltonian engineering, there are many other quantum control problems that could be approached using RL algorithms. Such problems include state-to-state transfer, quantum gate implementation, or bath engineering.

Appendix

5.1 Important Quantum Mechanics Concepts

This section is meant to present a few fundamental ideas that are used throughout this work, but for a more thorough description of quantum mechanics see McIntyre et al. (2012); Sakurai and Napolitano (2017). In all equations that follow, $\hbar = 1$ is assumed.

Given a quantum system, a general state can be represented by a density operator $\rho(t)$. The density operator encapsulates all known information about the quantum system, specifically the expectation values associated with well-defined observable. The density operator is Hermitian ($\rho^\dagger = \rho$) and $\mathrm{Tr}(\rho) = 1$. For an observable $\hat{A}$, there is a corresponding Hermitian operator A, with the expectation is given by

$$\langle A \rangle = \mathrm{Tr}\{\rho A\} \tag{5.1}$$

The time evolution of the density operator is given by

$$\rho(t) = U(t)\rho(0)U(t)^\dagger \tag{5.2}$$

where the unitary operator U is the "propagator" defined by

$$i\frac{dU(t)}{dt} = H(t)U(t), U(0) = \mathbb{1} \tag{5.3}$$

When the Hamiltonian is time-independent or if the Hamiltonian commutes with itself at different points in time (i.e. $[H(t_1), H(t_2)] = 0$), equation 5.3 can be solved to obtain the propagator

$$U(t) = \exp\left[-i\int_0^t H(t')dt'\right] \tag{5.4}$$

However, if the Hamiltonian is time-dependent and doesn't commute with itself at different times, then the above equation is not valid. This difficulty with time-dependent Hamiltonians and methods with dealing with them are further discussed in 1.2.1.

In many cases it can be helpful to consider the *interaction frame* of a particular interaction. If the Hamiltonian is expressed as the sum of two terms

$$H(t) = H_A(t) + H_B(t) \tag{5.5}$$

then there is a unitary operator $U_A(t)$ given by

$$\frac{d}{dt}U_A(t) = -iH_A(t)U_A(t),\, U_A(0) = \mathbb{1} \tag{5.6}$$

that maps from the interaction frame to the lab frame. The dynamics in the interaction frame are described by the interaction frame Hamiltonian $\widetilde{H}(t) = U_A(t)^\dagger H_B(t)U_A(t)$

$$\frac{d}{dt}\widetilde{U}(t) = -i\widetilde{H}(t)\widetilde{U}(t),\, \widetilde{U}(0) = \mathbb{1} \tag{5.7}$$

The propagator in the lab frame $U(t)$ is related to $\widetilde{U}(t)$ by first time evolution in the interaction frame, then mapping from the interaction frame to the lab frame

$$U(t) = U_A(t)\widetilde{U}(t). \tag{5.8}$$

5.2 Neural Network Architecture

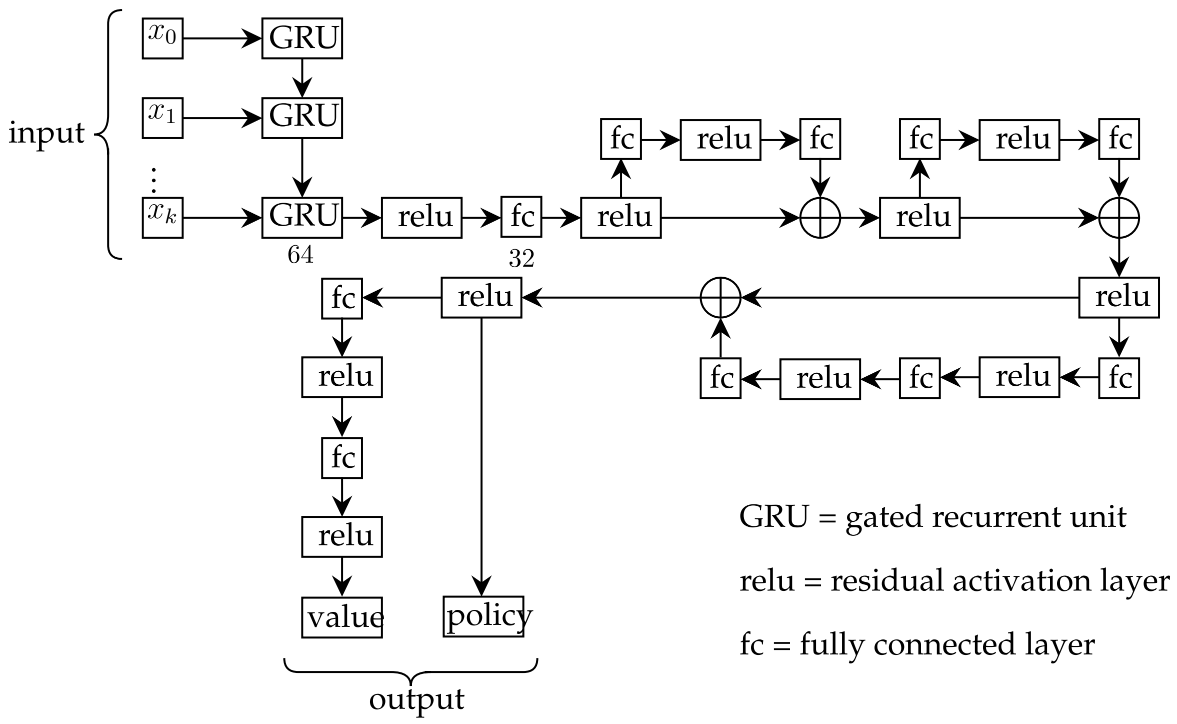


Figure 5.1: The policy and value network architecture.

5.3 Pulse Sequence Definitions

5.3.1 CORY48

$$X, \tau, Y, 2\tau, \overline{X}, \tau, Y, 2\tau, X, \tau, Y, 2\tau, X, \tau, Y, 2\tau, X, \tau, \overline{Y}, 2\tau, X, \tau, Y, 2\tau,$$

$$\overline{Y}, \tau, \overline{X}, 2\tau, Y, \tau, \overline{X}, 2\tau, \overline{Y}, \tau, \overline{X}, 2\tau, \overline{Y}, \tau, \overline{X}, 2\tau, \overline{Y}, \tau, X, 2\tau, \overline{Y}, \tau, \overline{X}, 2\tau,$$

$$\overline{X}, \tau, Y, 2\tau, \overline{X}, \tau, \overline{Y}, 2\tau, \overline{X}, \tau, Y, 2\tau, X, \tau, \overline{Y}, 2\tau, \overline{X}, \tau, \overline{Y}, 2\tau, X, \tau, \overline{Y}, 2\tau,$$

$$Y, \tau, \overline{X}, 2\tau, Y, \tau, X, 2\tau, Y, \tau, \overline{X}, 2\tau, \overline{Y}, \tau, X, 2\tau, Y, \tau, X, 2\tau, \overline{Y}, \tau, X, 2\tau$$

5.3.2 AlphaZero Pulse Sequences: No Errors

12τ

$$X, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau, X, \tau, \overline{Y}, \tau, X, \tau, X, \tau$$

24τ

$$\overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau, X, \tau, \overline{Y}, \tau, X, \tau,$$
$$\overline{Y}, \tau, \overline{Y}, \tau, X, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau, X, \tau, X, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau, \overline{Y}, \tau$$

48τ

$$\overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, Y, \tau, X, \tau, Y, \tau, X, \tau, X, \tau, Y, \tau,$$
$$Y, \tau, X, \tau, X, \tau, Y, \tau, X, \tau, X, 2\tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau,$$
$$\overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau, \overline{X}, \tau, \overline{Y}, 2\tau, \overline{Y}, \tau,$$
$$\overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau, Y, \tau, \overline{X}, \tau, Y, \tau, \overline{X}, \tau, Y, \tau$$

5.3.3 AlphaZero Pulse Sequences: Rotation Errors

12τ

$$\tau, X, \tau, X, \tau, Y, \tau, X, \tau, X, 2\tau, \overline{X}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau$$

24τ

$$\overline{Y}, \tau, X, \tau, Y, \tau, Y, \tau, X, \tau, Y, \tau, Y, \tau, \overline{X}, \tau, Y, \tau, Y, \tau, \overline{X}, \tau, Y, \tau,$$
$$Y, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau, \overline{Y}, \tau$$

48τ

$$\overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, Y, \tau, \overline{X}, \tau, \overline{X}, \tau, Y, \tau, \overline{X}, \tau,$$

$$\overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau, \overline{Y}, \tau, Y, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau,$$

$$\overline{Y}, \tau, X, \tau, Y, \tau, Y, \tau, X, \tau, Y, \tau, Y, \tau, \overline{X}, \tau, \overline{X}, \tau, Y, \tau, \overline{X}, \tau, \overline{X}, \tau,$$

$$\overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau$$

5.3.4 AlphaZero Pulse Sequences: All Errors

12τ

$$\overline{Y}, \tau, \overline{X}, \tau, Y, \tau, Y, \tau, \overline{X}, \tau, Y, \tau, Y, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau$$

24τ

$$\overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, Y, \tau, \overline{X}, \tau, Y, \tau, Y, \tau, \overline{X}, \tau, Y, \tau,$$

$$X, \tau, Y, \tau, X, \tau, X, \tau, Y, \tau, X, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau$$

48τ

$$X, \tau, X, \tau, \overline{Y}, \tau, X, \tau, X, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau,$$

$$\overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, Y, \tau, Y, \tau, Y, \tau, \overline{X}, \tau, Y, \tau, Y, \tau,$$

$$\overline{X}, \tau, \overline{X}, \tau, Y, \tau, Y, \tau, \overline{X}, \tau, Y, \tau, Y, \tau, Y, \tau, \overline{X}, \tau, Y, \tau, Y, \tau, \overline{X}, \tau,$$

$$X, \tau, \overline{Y}, \tau, \overline{Y}, \tau, X, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau, \overline{Y}, \tau, \overline{X}, \tau, \overline{X}, \tau$$

Bibliography

Arumugam, D., Henderson, P., and Bacon, P.-L. (2021). An information-theoretic perspective on credit assignment in reinforcement learning.

Blanes, S., Casas, F., Oteo, J., and Ros, J. (2009). The magnus expansion and some of its applications. *Physics Reports*, 470(5-6):151–238.

Blanes, S., Casas, F., Oteo, J. A., and Ros, J. (2010). A pedagogical approach to the Magnus expansion. *European Journal of Physics*, 31(4):907–918.

Brinkmann, A. (2016). Introduction to average hamiltonian theory. i. basics. *Concepts in Magnetic Resonance Part A*, 45A(6):e21414.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Choi, J., Zhou, H., Knowles, H. S., Landig, R., Choi, S., and Lukin, M. D. (2020). Robust dynamic hamiltonian engineering of many-body spin systems. *Physical Review X*, 10(3).

Cory, D., Miller, J., and Garroway, A. (1990). Time-suspension multiple-pulse sequences: applications to solid-state imaging. *Journal of Magnetic Resonance (1969)*, 90(1):205–213.

Facey, G. (2008). 1h nmr spectra of solids.

Gerstein, B. and Dybowski, C. (1985). *Transient Techniques in NMR of Solids: An Introduction to Theory and Practice*. Academic Press, 1 edition.

Haeberlen, U. (1976). Advances in magnetic resonance. In john S. Waugh, editor, *High Resolution Nmr in Solids Selective Averaging*, number 1, page ii. Academic Press.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Johansson, J., Nation, P., and Nori, F. (2013). Qutip 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234–1240.

Khadka, S. and Tumer, K. (2018). Evolution-guided policy gradient in reinforcement learning.

Khaneja, N., Reiss, T., Kehlet, C., Schulte-Herbruggen, T., and Glaser, S. J. (2005). Optimal control of coupled spin dynamics: design of nmr pulse sequences by gradient ascent algorithms. *J Magn Reson*, 172(2):296–305.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning.

McIntyre, D., Manogue, C., and Tate, J. (2012). *Quantum Mechanics*. Pearson Education.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey.

NIST (2018). The second quantum revolution. Website.

Peng, P., Huang, X., Yin, C., Joseph, L., Ramanathan, C., and Cappellaro, P. (2021). Deep reinforcement learning for quantum hamiltonian engineering.

Porotti, R., Tamascelli, D., Restelli, M., and Prati, E. (2019). Coherent transport of quantum states by deep reinforcement learning.

Rosin, C. D. (2011). Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230.

Sakurai, J. and Napolitano, J. (2017). *Modern Quantum Mechanics*. Cambridge University Press, 2 edition.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Watkins, C. J. C. H. (1989). Learning from delayed rewards.

Waugh, J. S., Huber, L. M., and Haeberlen, U. (1968). Approach to high-resolution nmr in solids. *Phys. Rev. Lett.*, 20:180–182.