Research Output

Jad Soucar

November 29, 2024

Bellow is a list of papers I've written in the fields of mathematics and computer science. The papers themselves are lengthy so I'm also including short summaries of each. My hope is that my previous research output along with my extensive industry experience is sufficient in proving my proficiency in machine learning and the basics of computer science necessary to succeed within the masters program. I'm also linking my Github where you can find code for machine learning algorithms, numerical solvers/simulators, along with consumer-facing software I've developed.

1 Autonmous Trading Using Deep Q Learning

Was the first author on a paper in which we explore the application of Deep Reinforcement Learning (DRL) to the domain of autonomous equity trading, with a particular focus on the use of Deep Q Networks (DQNs) to develop trading agents capable considering risk during their decision making process. To do this we incorporate a comprehensive set market indicators into the state space and several risk metrics into the reward function to guide trading decisions towards not only profitability but also risk-adjusted returns. The risk metrics we tested include the Sharpe Ratio, Sortino Ratio, and Treynor Ratio. We tested Vanilla DQN, Fixed Target Distribution DQN and Double DQN with each risk metrics to find the optimal trading agent. We found that most trading agents trained earn a percentage increase of around 6%-13%. Specifically we found that incorporating the Sharpe ratio into the reward function produces the best return, and that the Double DQN algorithm is optimal across all risk metrics.

The Paper is Attached Bellow.

2 UCLA Semel Institute of Human Behavior

Contributed to two research papers. The first of which discusses the effect of machine translation on the accuracy of sentiment analysis, with an emphases on transcripts from the G8 and G20 Summits. My work for the first paper involved building a Russian transcription and translation pipeline on UCLA's Hoffman super cluster. The second paper evaluates the use of Long Short Term Memory Neural networks to classify the same transcripts from the G8 and G20

summits. My work for the second paper involved implementing a training the LSTM model that was evaluated. Both papers have recently been submitted to the the Society for Computation in Linguistics's 2024 Conference.

Both Papers are Attached Bellow.

3 Model Predictive Control Algorithms for Modeling Wayfinding

For the past two centuries, research on human cognition and decision-making has centered on rational choice theory, which assumes individuals are utility maximizers capable of evaluating all possible options to make optimal decisions. However, human behavior data contradicts this, as real-world decision-making is constrained by limited time and cognitive resources. To address this gap, the proposal introduces three axiomatic principles of energy-efficient decision-making, supported by an optimal-control model and a Deep Q learning framework. These models aim to provide more realistic and computationally efficient ways to model decision-making across fields like market behavior, crowd dynamics, and even large language models (LLMs). The proposed frameworks could enhance LLMs by enabling cost-effective planning of multiple tokens ahead, improving over the current auto-regressive systems. Despite broader AI applications, the theory is grounded in classical wayfinding, such as foraging, to formalize its principles.

The Paper is Attached Bellow.

4 Novel Quantum Algorithms for Simulating Noise

First we explore methods to decompose a noise operator into a sum of cliffords via mixed integer linear programming and using these decompositions to classically simulate noisy quantum circuits within the stabilizer formalism first proposed by Aaronson and Gottesman (2004). We find that the a Clifford decomposition is not guaranteed to have low rank decompositions and that at best the runtime of simulating noise using Clifford decompositions would be $O(2^k)$, where k is the number of Kraus operators applied. Second, we explore the process of dilating our space to convert our noise operators into unitaries in a larger space. Specifically we propose two unitary dilation based algorithms for simulating noise; Sz.-Nagy and Stinespring's dilation algorithms. The two algorithms yield respective run-times of $O(\frac{1}{\delta\Delta^2} sn^3\eta^{-2}1.17^t)$ and $O(n^2\prod_{i=1}^{k}|\xi_i| + n^3\sum_{i=1}^{k}|\xi_i|^3)$. Third we propose a theoretical framework for generalizing the T-Gadget approach developed by Bravyi and Gosset (2016). Through large scale numerical simulations we show that the generalized framework can be used to produce noise simulation algorithms with efficient runtime and space complexity.

The Paper is Attached Bellow.

5 Quantum Decision Making Algorithms

In this paper, we have explored the complexity of decision making in the single-celled organism Stentor roeselii. Through an analysis of the data provided by Dexter, we confirmed that the protist's behavioral responses, though simple in nature, follow statistical trends that imply a form of primitive learning or adaptation. While previous attempts to model this behavior using classical machine learning frameworks, such as decision trees and neural networks, have failed to capture the probabilistic aspects of the organism's actions, we proposed a novel quantum behavioral model that aligns with these complexities. The quantum framework we introduced successfully simulates the organism's decision-making process by incorporating both noise and probabilistic features observed in Stentor roeselii's behavior. By utilizing quantum information theory, particularly quantum circuits with amplitude dampening and memory effects, our model captures the stochastic nature of the protist's decision-making hierarchy. This allows for the representation of inter- and intra-sequence learning that was absent in earlier models or hidden behind a black box.

The Paper is Attached Bellow.

6 A Model for Trust Driven Advertising

Was the first author on a paper where we propose that cognitive processes underlie economic relations. In the paper we develop a conceptual, mathematical, and computational framework for modeling market exchange as a series of dynamically interacting cognitive processes. We've submitted the paper to the 10th International Conference on Computational Social Science and the 2024 Cognitive Science Society Conference.

The Paper is Attached Bellow.

7 Recasting a Labor Adjusted Aiyagari Income Model into an MFG System

Wrote a paper examining the Aiyagari Income Model MFG system proposed by Achdou, Han, Lasry, Lions, and Moll. My paper was focused on how total labor supply was represented, and how one could adjust the model to account for idiosyncratic spikes or shortages in the total labor supply. Throughout this paper we present a brief primer of the Aiyagari model of income that's been modified to incorporate idiosyncratic labor supply, along with necessary back ground/derivations of the HJB and KFP equations, before finally introducing a derivation of the Mean Field Game representation of our labor adjusted income model.

The Paper is Attached Bellow.

Sentiment Analysis of Russian Political Discourse: Does Translation Matter?

Anonymous ACL submission

1 Introduction

Current trends in the automated analysis of online media texts endeavor to identify 'misinformation', or the spread of misleading information. Although emotional and subjective language can be exploited for disinformation purposes, automated analyses often struggle to classify texts on the level of discourse pragmatics. This difficulty is compounded in cross-cultural communication, where speaker intent can be misinterpreted due to the transformation of meaning that occurs in translation (Lotman, 1990). Few authors question how pragmatic systems may be encoded across languages (Comstock, 2015), and whether this will affect the interpretation of their model outputs. By comparing the emotional and subjective language employed by journalists while questioning the Russian president, this paper problematizes the assumption that a sentiment analysis performed on a translation and its source text will be equivalent.

2 Related work

The successful classification of discourse-level phenomena combines multiple linguistic features or domains (Becker et al., 2020). Co-occurring markers of polarity and subjectivity may isolate contexts that promote misinformation (Carrasco-Farré, 2022); however, analyses performed on Russian texts in translation typically fail to ascertain whether the pragmatics of the translated text more closely reflect that of the source or target language (Araujo et al., 2016). As a result, even with the numerous authors that have applied sentiment analysis techniques to misinformation in Russian discourse (Pocyte, 2019; Yaqub et al., 2020), the effect of translation on analysis outputs remains an important topic of study.

Summit	Term	Russian	English
G8	2000-2003	757	874
	2004-2007	2129	2611
	2008-2011	1412	1709
	2012-2015	611	737
G20	2000-2003	–	–
	2004-2007	–	–
	2008-2011	1598	1887
	2012-2015	2241	2474
Total		12338	14667

Table 1: The number of words collected in each summit are presented for the Russian transcripts and English translations. The translations are nearly 19% longer.

3 Methods

We investigated (i) if the total lemma count and frequency of emotional and subjective words differ by the nature of the political event (the more exclusive G8 summit versus the G20 summit); (ii) if the total lemma count and frequency differ by presidential term (2000-2019); and (iii) how the observed frequencies of lemmas produced during press conferences correspond to the expected lemma frequencies calculated from Russian-language or English-language online media sources (the National Russian Corpus and Google Ngrams).

The corpus comprises all publicly available transcripts of press conferences held by the Russian president at G8 and G20 summits from 2000-2015 (Comstock, 2023). The written transcripts were accessed at the Kremlin online press archives (`http://kremlin.ru`, `http://en.kremlin.ru/`). Questions were originally posed in Russian. A composite list of positive, negative, and subjective words was compiled from the Harvard IV-4, Loughran, McDonald, and Lexicoder sentiment dictionaries. Translation accuracy and the applicability of the composite list were confirmed by a professional Russian translator.

4 Results

We observe that the language context in which the sentiment analysis is performed does not remain consistent, even across similar political venues and short time differences.

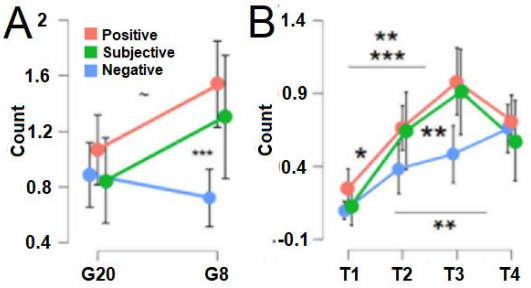


Figure 1: Sentiment analysis by summit. (A) The total lemma count differed by sentiment type and summit. (B) The total lemma count differed by presidential term.

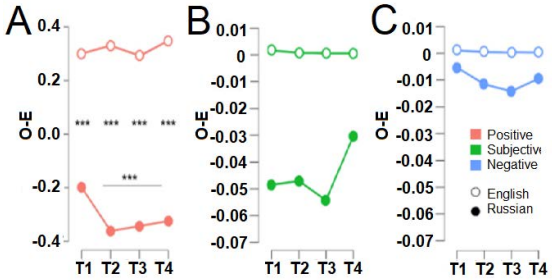


Figure 2: Sentiment analysis by sentiment type. The difference between observed and expected frequencies by (A) positive, (B) subjective, and (C) negative sentiment.

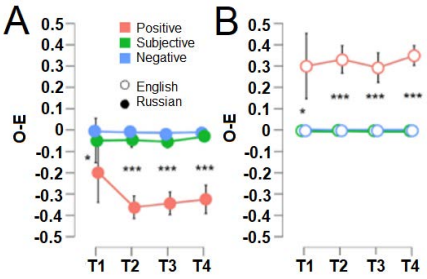


Figure 3: Sentiment analysis by language. The observed frequency of lemmas in each language relative the expected frequency of the same lemmas.

5 Discussion

The translation and original text do not produce an equivalent effect. The translation largely reproduces the expected distribution of emotional content, with a slight increase in positive items, whereas the original Russian text employs significantly less positive emotion. Subjective words trend with positive items in terms of the total count, which may reflect general pragmatic norms to upgrade positive assessments and minimize negative ones. Overall, the translator used a smaller range of words than the Russian text, accommodating general language norms, whereas the Russian text remains more specific and illustrates a wider range of lemmas. This is generally considered to be the advantage of utilizing a human translator: the text reads more naturally because it conforms to target language norms. However, we see that this practice also changes the emotional tone of the text.

6 Conclusion and limitations

The sentiment analyses illustrate that classification outputs, like reader perceptions of a translated text, may differ notably. We anticipate greater significance will appear with a more robust exclusion of outliers. Analysis of the effect of extreme outliers by language type is a future direction of research.

References

Matheus Araujo, Julio Reis, Adriana Pereira, and Fabricio Benevenuto. 2016. An evaluation of machine translation for multilingual sentence-level sentiment analysis. pages 1140–1145.

Maria Becker, Michael Bender, and Marcus Müller. 2020. Classifying heuristic textual practices in academic discourse: A deep learning approach to pragmatics. *International Journal of Corpus Linguistics*, 25(4):426–460.

Carlos Carrasco-Farré. 2022. The fingerprints of misinformation: How deceptive content differs from reliable sources in terms of cognitive effort and appeal to emotions. *Humanities and Social Sciences Communications*, 9(1):1–18.

Lindy Comstock. 2023. Journalistic practice in the international press corps: Adversarial questioning of the russian president. *Journal of Language Aggression and Conflict*, 11(2):145–175.

Lindy B Comstock. 2015. Facilitating active engagement in intercultural teleconferences: A pragmalinguistic study of russian and irish participation frameworks. *Intercultural pragmatics*, 12(4):481–514.

Yuri Lotman. 1990. *Universe of the mind: A semiotic theory of culture*. Indiana University Press.

Agniete Pocyte. 2019. From russia with fear: The presence of emotion in russian disinformation tweets.

Ussama Yaqub, Mujtaba Ali Malik, and Salma Zaman. 2020. Sentiment analysis of russian ira troll messages on twitter during us presidential elections of 2016.

An evaluation of sentiment models relative human coding in pragmatically-defined speech

Anonymous ACL submission

1 Introduction

Computational methods aim to closely replicate human sentiment encodings, yet the simple NLP models traditionally employed by linguists show variable success in task performance (Kansara and Sawant, 2020). This extended abstract illustrates the ways in which more sophisticated yet accessible sentence- or discourse-level techniques, such as Long-Short Term Memory (LSTM) modeling, can better match human sentiment analyses as compared to word-level techniques, such as Bag-of-Words models, by mimicking aspects of language pragmatics (Comstock, 2015; Thomas, 2014). Our aim is to encourage linguists to consider the implications of model selection for their analysis task.

2 Methods

We utilize a corpus of questions posed to Russian Presidents at the G8 and G20 press conferences from 2000 to 2019. Transcripts are sourced from the Kremlin press archive (`http://kremlin.ru`, `http://en.kremlin.ru/`). All texts are in Russian. The corpus comprised 256 questions (12338 words).

2.1 Human coding

Questions occur within a larger text referred to as a "questioning turn–QT" (Clayman et al., 2006). Human coding consisted of labeling (i) individual sentences–ST within a QT as “positive,” “negative,” and “neutral,” and then aggregating sentence-level codes to (ii) classify the entire QT according to one of the three categories. A second analysis considered additional contextual information to subdivide each category: (i) positive politically-related questions vs. non-political human interest questions, and (ii) questions hostile towards the policy described vs. toward the lack of solidarity exhibited among summit members. Details of the coding scheme have been published (Comstock, 2023).

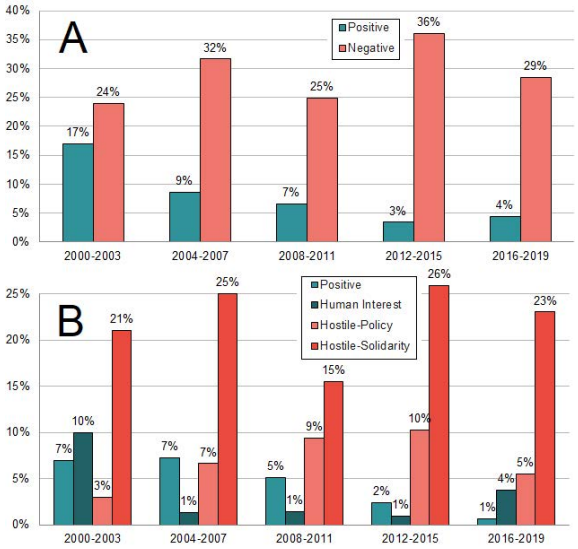


Figure 1: Human coding. (A) Simple sentiment coding. (B) Contextual sentiment coding. Axes show the percentage of correct labels for STs by presidential term.

Human coding illustrated a trend towards less positive sentiment over time (see Figure 1). Contextually-determined subcategories were well-represented, particularly in the positive category.

2.2 Matching human coding: Bag-of-Words

The Bag-of-Words (BoW) analysis used a modified Lexicoder Sentiment Dictionary to determine key sentiment word cues (Young and Soroka, 2012) from the English transcripts; afterward, words that were summit-specific (i.e., "resolution", etc.) or carried a different sentiment in Russian were removed by a professional Russian translator. Positive and negative words were divided by the total number of words in a given presidential term to determine sentiment frequency by term.

2.3 Matching human coding: Neural Network

The model and training data were adapted from an open-source Kaggle competition for sentiment

analysis of Russian news. The best-performing convolutional neural network bidirectional long-short-term memory (CNN-BiLSTM) model was used (https://www.kaggle.com/code/thehemen/cnn-bilstm-russian-news-classifier). Training data comprised 8263 excerpts of varying length from pre-classified Russian news articles. The model predicted sentiments of STs and QTs in our data, assigning a “positive,” “negative,” or “neutral” label.

3 Results

When compared to the human coding, both computational models were ineffective in matching the actual percentages of human-coded sentiment. The BoW analysis not only showed much lower percentages of captured sentiment but also did not accurately reflect the trends shown in the human coding (see Figure 2). While the LSTM analysis of STs captures "more sentiment," this may be misleading: it is ineffective in capturing the true scope of sentiment and the trends over time as compared to the QT analysis.

Figure 2: Sentiment analyses. (A) BoW analysis. (B) LSTM analysis of STs. (C) LSTM analysis of QTs. Axes show the percentage of correct labels for words, STs, or QTs by presidential term.

4 Discussion

Although more sophisticated models such as LSTM increase complexity and involve a greater learning curve to utilize, the results are markedly better at reflecting trends across time. As neither model is accurate on an item-by-item basis, LSTM models are strongly preferable to capture sentiment analysis trends. Consideration of extra contextual data appears to have boosted the QT model performance: when individual items are reviewed with their assigned labels, we see that the BoW analysis captures a large portion of "positive" and "hostile-policy" human-coded data, but not the "human interest" or "hostile-solidarity" data. The LSTM performs markedly better in accounting for these more pragmatically defined subcategories.

5 Limitations and further direction

In our extended abstract, we will provide text excerpts to illustrate how the various models focus on different pragmatic elements of the sentence. We will also have space to provide statistical analyses. LSTM performance is highly dependent on training data; using a dataset more closely related to political questioning than the given Russian media dataset might improve or change our findings. Additionally, there is a large gap in complexity between BoW and neural network analyses. Considering a wider variety of models will provide more detailed insight into which computational methods are effective for different use cases.

References

Steven E Clayman, Marc N Elliott, John Heritage, and Laurie L McDonald. 2006. Historical trends in questioning presidents, 1953-2000. *Presidential Studies Quarterly*, 36(4):561–583.

Lindy Comstock. 2023. Journalistic practice in the international press corps: Adversarial questioning of the russian president. *Journal of Language Aggression and Conflict*, 11(2):145–175.

Lindy B Comstock. 2015. Facilitating active engagement in intercultural teleconferences: A pragmalinguistic study of russian and irish participation frameworks. *Intercultural pragmatics*, 12(4):481–514.

Dhvani Kansara and Vinaya Sawant. 2020. Comparison of traditional machine learning and deep learning approaches for sentiment analysis. In *Advanced Computing Technologies and Applications: Proceedings of 2nd International Conference on Advanced Computing Technologies and Applications—ICACTA 2020*, pages 365–377. Springer.

Jenny A Thomas. 2014. *Meaning in interaction: An introduction to pragmatics*. Routledge.

Lori Young and Stuart Soroka. 2012. Affective news: The automated coding of sentiment in political texts. *Political Communication*, 29(2):205–231.

Autonomous Equity Trading using Deep Reinforcement Learning

Jad Soucar (jadsoucar@g.ucla.edu)
Department of Mathematics, 520 Portola Plaza, University of California Los Angeles
Los Angeles, CA 90095 USA
Ninaad Surana (ninaadsurana@g.ucla.edu)
Department of Mathematics, 520 Portola Plaza, University of California Los Angeles
Los Angeles, CA 90095 USA

March 17, 2024

Abstract

In this paper, we explore the application of Deep Reinforcement Learning (DRL) to the domain of autonomous equity trading, with a particular focus on the use of Deep Q Networks (DQNs) to develop trading agents capable of navigating the complex and dynamic landscape of the financial markets. We introduce three variants of the DQN model; Vanilla DQN (V-DQN), Target DQN (T-DQN), and Double DQN (D-DQN). Our models incorporate a comprehensive set market indicators into the state space and several risk metrics into the reward function to guide the trading decisions towards not only profitability but also risk-adjusted returns. The risk metrics include the Sharpe Ratio, Sortino Ratio, and Treynor Ratio. We test each Q learning scheme outlined above with each risk metrics to determine the best trading agent. We find that most trading agents earn a percentage increase of around 6%-13%. After training, we find that incorporating the Sharpe ratio into the reward function produces the best return, and that the Double DQN algorithm is optimal across all risk metrics.

Keywords: Reinforcement Learning, Q Learning, Finance, Agent Based Modeling, Equity Trading

1 Introduction

In recent years, advancements in the fields of Deep and Reinforcement Learning have transformed algorithmic trading. With the surge in computational power and increasing amounts of financial data, traders and researchers have been exploring novel methods to create autonomous trading agents that can operate successfully in financial markets. One such strategy is Deep Q Learning, which has dominated various domains such as game playing, robotics, and now, financial trading.

With this in mind, we form the questions, how do we create such an agent, and guide it to make valid decisions? How do we ensure the stability of the agent? How do we improve upon previous attempts? We observed that in existing literature, many researchers had attempted to create such a Deep Q Learning system using ordinary Profit-Loss based Reward systems and state spaces that contain only the asset's price. In order to improve performance, we decided to incorporate a wider range of performance indicators and risk metrics into our agent's decision making process. We hope to create an agent that can mimic a human investor with strong insight into market trends and behaviour. The primary concerns surrounding autonomous trading are rooted in the belief that technical analysis does not account for underlying factors that drive growth. For example if an autonomous agent sees unexpected behaviour in the markets, or is unable to access long horizons of meaningful data, it may be unable to act as an informed human would by analyzing the key indicators and risk metrics. However, trading algorithms introduce their own advantages, such as a lack of emotion or bias, and the ability to place buy and sell orders instantly without need for time-intensive deliberation. Moreover, algorithms have

the ability to identify trends and patters that may be difficult for a human trader to observe. Overall, we seek to combine all the advantages of both human and autonomous trading, and build an agent that can be continuously improved. In our implementation, we aim to contribute to the growing field of Reinforcement Learning (RL), and expand its applications into financial markets.

In this paper, we focus on two problems native to the use of RL in finance. The first issue is the noise contained in the underlying data, which can make training a Deep RL network difficult. To mitigate this problem we implement Fixed Target Distribution Deep Q Learning and Double Deep Q Learning and compare their efficacy to a traditional Deep Q Learning framework. Next we acknowledge that many RL trading agents only use their profit or loss to inform investment decision. However, in reality traders are often concerned with hedging risk. In an effort to replicate this type of trading behavior we use risk metrics as a way of assessing an agent's investment decisions. To do this, we implement different combinations of metrics such as the Sharpe Ratio, Sortino Ratio, and Treynor Ratio. We then present results of the agent's performance using different training schemes. We see that the agent always delivers promising rates of return, between 6% and 13%. Specifics on the implementation of our trading environment, algorithms used, and experimental results will be detailed in further sections.

1.1 Paper Organization

We begin with a brief introduction of necessary mathematical prerequisites of Q Learning. Next, we discuss the necessary financial prerequisites, before proposing several deep Q learning frameworks to solve the problem of optimal equity trading. We end with an evaluation of our results and consider possible future work on this problem.

2 Mathematical Background

In this section we aim to provide the relevant mathematical and programmatic background for implementing a Deep Q Learning algorithm. We discuss what Q learning is and how the principles from classical Q learning can be used to construct deep Q learning algorithms in Python. We will be borrowing heavily from Brunton and Kutz's book titled "Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control".

2.1 General Reinforcement Learning

Before we begin our discussion of Q learning, we will begin with an exposition of the two primary categories of RL: model-based and model-free. Model-based RL systems create a model of the environment, which includes the dynamics of how actions lead to subsequent states and rewards. Essentially, it predicts the future states and rewards for actions taken from a given state. Take for example the gambler problem, which provides explicit probability measure $P(s',s,a)$ that provides explicit probabilities as to whether the actor will win or lose a gamble. In other words, the gambler queries the environment for its transition dynamics. The second approach is model-free, which is where a system learns a policy or value function directly from interactions with the environment without constructing an explicit model of the environment's dynamics. The system learns what to do by trial and error, adjusting its actions based on the rewards received.

The second distinction that we'd like to note is two subcategories of model-free RL; off and on-policy learning. On-Policy learning algorithms learn the value of the policy that is currently being used to make decisions. Put simply, the policy used to generate behavior (exploration) is the same policy that is being evaluated and improved. Essentially, it learns on the job, using the data generated by its current strategy. By contrast off-policy learning algorithms can learn about one policy (the target policy) while using a different policy (the behavior policy) to generate behavior. This means it can learn from data that was generated from a previous policy or even from data generated by other agents. This separation allows for greater flexibility and efficiency in some contexts.

2.2 Q Learning

With the general background of RL out of the way, we note that Q learning is a model-free and off-policy training scheme. That means that Q learning does not rely on an environment that can produce exact transition probabilities and must instead use trial and error of optimal or sub-optimal actions in order to learn the optimal policy. In order to formalize this type of training scheme we first define the quality function $Q(s,a)$, which tells you the joint value/quality of taking an action a given a current state s.

To formally define $Q(s,a)$ we first introduce some notation. We let $R(s',s,a)$ be the reward of transitioning from state s to s' through action a, γ be the discount factor of future reward, and $V(s')$ be the expected value over all possible future rewards given a current state s'. With that we define $Q(s,a)$ as

$$\begin{aligned} Q(s,a) &= \mathbb{E}\left(R(s',s,a) + \gamma V(s')\right) \\ &= \sum_{s'} P(s'|s,a)(R(s',s,a) + \gamma V(s')) \end{aligned} \tag{1}$$

In other words, Q is the expected sum of the instantaneous reward of the state-action pair (s,a) along with the discounted future rewards of being at a new state s' brought on by (s,a). Using the quality function $Q(s,a)$ we can define an optimal policy π and value function $V(s)$, that considers which action a is optimal and what the expected reward from taking that action is.

$$V(s) = \max_a Q(s,a) \tag{2}$$

$$\pi(s) = \arg\max_a Q(s,a) \tag{3}$$

We can even rewrite the equations above in terms of Bellman's equation, which tells us that the value of being in state s is the expected current and future return of applying the optimal action a.

$$V(s) = \max_\pi \mathbb{E}\left(R(s',s,a) + \gamma V(s')\right) \tag{4}$$

We've defined what a q value is and how to construct the quality function $Q(s,a)$, but now we define a recursive equation to update $Q(s,a)$ as the agent learns through trial and error.

$$Q^{\text{new}}(s_k,a_k) = Q^{\text{old}}(s_k,a_k) + \alpha\left(\underbrace{r_k + \gamma\max_a Q(s_{k+1},a) - Q^{\text{old}}(s_k,a_k)}_{\text{TD Error}}\right) \tag{5}$$

Let's dissect this update equation. As we engage in trial and error learning, we update our $Q(s,a)$ by slowly nudging our q values up or down by a factor of the difference between the actual current and future reward $r_k + \gamma\max_a Q(s_{k+1},a)$ and our predicted reward $Q^{\text{old}}(s_k,a_k))$. This difference is sometimes referred to as the "Temporal Difference (TD) error". We also note that α in this case is the learning rate.

We draw the readers attention to two interesting features of the update equation. First to the term $\gamma\max_a Q(s_{k+1},a)$. The reason why we maximize Q over a is to guarantee that the quality $Q(s_k,a_k)$ is a function of the optimal actions that can be taken given a new state s' brought on by (s_k,a_k). r_k on the hand, which is the reward derived from the pair (s_{k-1},a_{k-1}) is not required to be the optimal reward, and will most likely be sub-optimal during the training process. This is precisely the reason why Q Learning is an off-policy learning scheme.

Second we note that when calculating the TD error, the agent calculates the future reward by looking one step into the future. This idea of looking forward in time by one time step to calculate error is why the error is called the TD error. Naturally, the degree to which the agent looks into the future can be modified.

2.3 Deep Q Learning Variations

Now we discuss the process of recasting Q Learning into training schemes that incorporate deep learning methods like neural networks. This is especially useful in settings where the state space is too large to reasonably store all q values in a Q table. Here, a deep learning approach seeks to parameterize $Q(s,a)$ as dependant on some weights θ such that

$$Q(s,a) \approx Q(s,a,\theta) \tag{6}$$

In order to optimize for the parameters θ within a Q Learning framework we use a loss function that is eerily similar to the TD error defined in equation (5). This framework is known as a Vanilla Deep Q Network (V-DQN).

$$\mathcal{L} = \mathbb{E}[(r_k + \gamma \max_a Q(s_{k+1}, a, \theta) - Q(s_k, a_k, \theta))^2] \tag{7}$$

With the loss function properly defined, we can move on to the neural network architecture, and how it allows us to approximate the Q function. In practice, we find that the architecture of the network varies based on the use case. For example a team of DeepMind engineers in 2013 coupled the loss function described above with several convolutions layers and fully connected layers. The inputs were several consecutive frames for the game, which represented the agent's state. And the output was one of several possible action that the agent could take. Regardless of the architecture used the value contained in each output node approximates the value $Q(s,a)$ for the associated (s,a) which corresponds to the network's $(\text{Input}, \text{output})$ pair.

The deep Q networks described above can be difficult to train due large variance in the estimated Q function. To mitigate this problem we can introduce two separate schemes to deal with this problem.

2.3.1 Stabilization Schemes

First we introduce the fixed target distribution Q learning framework, known as T-DQN. This framework involves introducing a second neural network to approximate the temporal difference target, instead of using the same main network to predict the quality of the next state. Under this framework TD Error and TD Target are defined as follows

$$\underbrace{\overbrace{r_k + \gamma \max_a Q(s_{k+1}, a; \theta^-)}^{\text{TD Target}} - Q^{\text{old}}(s_k, a_k; \theta)}_{\text{TD Error}} \tag{8}$$

where $Q(s', a'; \theta^-)$ is the Q value predicted by the target network for the next state-action pair, and $Q(s,a;\theta)$ is the Q value predicted by the main Q network for the current state-action pair.

$$TD_{\text{target}}^{\text{T-DQN}} = r + \gamma \max_{a'} Q(s', a'; \theta^-), \tag{9}$$

However, unlike the main network, instead of updating the target network at every training step, which can lead to high variance in the target Q values and general instability in training, the target network's weights are only updated after a fixed number of steps or episodes. This creates a stable target distribution for the Q values against which the policy network is updated. However, given the training scheme of the target network it is functionally just a delayed copy of the main network. Because the target and actual Q values are being estimated by similar networks, we run the risk of overestimating the target when our data has high variance. To mitigate the problem of overestimation we can propose another framework called the Double Deep Q Network (D-DQN).

Like fixed target distribution Q learning, a double deep Q learning framework relies on a target network and a main network. However in this case, the main Q network is responsible for choosing the best action, but the target network is used to evaluate the quality of that action. Its a fine difference that can be formulated as follows

$$TD_{\text{target}}^{\text{DDQN}} = r + \gamma Q(s', \underset{a'}{argmax} Q(s', a'; \theta); \theta^-), \tag{10}$$

3 Financial Background

In this section we transition to some necessary financial background for creating our trading agent. Specifically we introduce several widely used market features and risk metrics. For the sake of this section and the rest of the paper, assume that

$$S_t = \text{Price at time } t \tag{11}$$

3.1 Market Features

We introduce 4 commonly used market features: Moving Average Convergence Divergence (MACD) Aguirre et al. (2020), Relative Strength Index (RSI) Adrian (2011), Commodity Channel Index (CCI) Maitah et al. (2016), Average Directional Index (ADX) Gurrib (2018).

3.1.1 MACD

In order to define MACD, we must first define the Exponential Moving Average (*EMA*)

$$\text{EMA}_t = \left(S_t \times \frac{2}{N+1}\right) + \text{EMA}_{t-1} \times \left(1 - \frac{2}{N+1}\right) \tag{12}$$

where N is the number of periods that the EMA is calculated over. For example $N = 12$ means that *EMA* is calculated over 12 days.

The MACD is a trend-following momentum indicator that shows the relationship between two moving averages of a stock's price, whereas the signal line is the EMA of that same difference. The MACD can help identify the overall trend. When the MACD Line is above the Signal Line, it's considered bullish. When the MACD Line is below the Signal Line, it's considered bearish.

$$\text{MACD Line} = \text{EMA}_{12}(S_t) - \text{EMA}_{26}(S_t) \tag{13}$$

$$\text{Signal Line} = \text{EMA}_9(\text{MACD Line}) \tag{14}$$

3.1.2 Relative Strength Index (RSI)

The RSI is a momentum oscillator that can be defined as follows. The RSI is a momentum oscillator that measures the speed and change of price movements. RSI oscillates between zero and 100. Traditionally, RSI is considered overbought when above 70 and oversold when below 30. Before defining RSI, we must first define a few terms. G_t represent the gain at time t, where $G_t = \max(S_t - S_{t-1}, 0)$. L_t represent the loss at time t, where $L_t = \max(S_{t-1} - S_t, 0)$.

The average gain and average loss over a specified period N (traditionally 14) are calculated as follows:

$$\text{Average Gain} = \frac{1}{N}\sum_{i=1}^{N} G_{t-i} \tag{15}$$

$$\text{Average Loss} = \frac{1}{N}\sum_{i=1}^{N} L_{t-i} \tag{16}$$

The relative strength (RS) is the ratio of average gain to average loss:

$$RS = \frac{\text{Average Gain}}{\text{Average Loss}} \tag{17}$$

The Relative Strength Index (RSI) is then calculated using the RS value:

$$RSI = 100 - \left(\frac{100}{1 + RS}\right) \tag{18}$$

3.1.3 Commodity Channel Index (CCI)

The CCI compares a securitie's current price change with its average price change over a given period. It's used to identify cyclical trends in commodities but can be applied to other types of assets. Though it can vary by market and the time frame you're looking at, CCI readings above +100 typically indicate overbought conditions, suggesting a price reversal might be near. Readings below -100 indicate oversold conditions, possibly heralding a bullish reversal. In order to formalize CCI we first define Typical Price (TP), and Simple Moving Average of TP (SMATP).

$$\text{TP} = \frac{S_{t,\text{high}} + S_{t,\text{low}} + S_t}{3} \tag{19}$$

$$\text{SMATP} = \frac{\sum \text{TP}_t}{N} \tag{20}$$

$$\text{Mean Deviation} = \frac{1}{N}\sum_{t=0}^{N} |TP - SMATP| \tag{21}$$

$$\text{CCI} = \frac{\text{TP} - \text{SMATP}}{0.015 \times \text{Mean Deviation}} \tag{22}$$

3.1.4 Average Directional Index (ADX)

The Average Directional Index (ADX) is a technical analysis indicator used to quantify the strength of a trend. The ADX itself doesn't indicate trend direction, but it measures the strength of the current trend, whether up or down. Values above 25 usually indicate a strong trend, while values below 20 indicate a weak trend or trading range. Values in between suggest a developing trend. However, in order to define ADX, we must once again define several new terms.

H_t represents the high price at time t, L_t represents the low price at time t, C_t represents the close price at time t. DM^+ and DM^- denote the positive and negative directional movements, respectively. TR denotes the true range. $+DI$ and $-DI$ represent the positive and negative directional indicators, respectively. We can define each of these terms bellow.

$$DM^+ = \max(H_t - H_{t-1}, 0) - \min(L_t - L_{t-1}, 0) \tag{23}$$

$$DM^- = \max(L_{t-1} - L_t, 0) - \min(H_{t-1} - H_t, 0) \tag{24}$$

$$TR = \max(H_t - L_t, |H_t - C_{t-1}|, |L_t - C_{t-1}|) \tag{25}$$

$$+DI = 100 \times \frac{DM^+}{TR} \tag{26}$$

$$-DI = 100 \times \frac{DM^-}{TR} \tag{27}$$

$$ADX = 100 \times \frac{\text{EMA of } |(+DI) - (-DI)|}{TR} \tag{28}$$

3.2 Risk Metrics

We now introduce 4 risk metrics, that allow a market agent to evaluate the risk of an investment: Sharpe Ratio, Sortino Ratio, Treynor Ratio, Beta.

3.2.1 Beta (β)

Beta (β) measures a security's sensitivity to market movements. It represents the tendency of a security's returns to respond to swings in the market and is a key component in the Capital Asset Pricing Model (CAPM).

$$\beta = \frac{\text{Cov}(R_s, R_m)}{\text{Var}(R_m)} \tag{29}$$

3.2.2 Sharpe Ratio

The Sharpe Ratio evaluates the performance of an investment compared to a risk-free asset, after adjusting for its risk. It is used to understand how much excess return is being received for the extra volatility that one bears for holding a riskier asset.

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p} \tag{30}$$

3.2.3 Sortino Ratio

The Sortino Ratio improves upon the Sharpe Ratio by focusing only on the downside deviation instead of the total standard deviation. This makes it a better measure of the risk-adjusted return when the investor is concerned about downside risk.

$$\text{Sortino Ratio} = \frac{R_p - R_f}{\sigma_d} \tag{31}$$

3.2.4 Treynor Ratio

The Treynor Ratio is similar to the Sharpe Ratio but uses beta (β) instead of standard deviation to measure volatility. It assesses the returns earned in excess of the risk-free rate per unit of market risk and is particularly useful for diversified portfolios.

$$\text{Treynor Ratio} = \frac{R_p - R_f}{\beta_p} \tag{32}$$

4 Model Development

In this section we develop our model by first defining the environment,state,action, and reward space. Then we choose a specific deep Q network architecture, and define the control flow of our model.

4.1 Environment Setup

The stock market is a multi-agent complex system where various agents interact with each other through the purchase or sale of various financial assets. For the sake of our model we restrict our RL agent to trading stock in publicly traded companies. The limit order book (LOB) plays a crucial role in how stocks are priced.

The LOB is a real-time database that lists all open buy orders (bids) and sell orders (asks) for a stock at various price levels. Bids are listed in descending order with the highest price at the top, while asks are listed in ascending order with the lowest price at the top. The LOB facilitates price discovery as the matching of buy and sell orders based on price and time priority determines the current price of the stock. This processes of matching prices based on the difference between the bid and ask price (spread), can be directly observed through the dynamic pricing of the asset. In other words, despite the motivation's of the agents remaining hidden, the state of the environment can be readily observed at the current price S_t. The agent can attempt to recognize patterns in the price S_t, and construct market features similar to those defined in section 3.1.

Above we describe the observable features of the environment, however, for our model we also assume that the RL agent has a set of personal rules that they must follow. The first rule is that the agent can only place a trade of one share per transaction. The second rule we impose, for the sake of simplicity, is that our agent can only place one trade per day. Third we restrict the agent to buying whole shares, and exclude any option of

purchasing or selling fractional shares. Fourth, we only allow the RL agent to purchase a single stock. While there are models that allow RL agents to construct portfolios by analyzing several stocks we restrict our RL agent to a single stock for simplicity. Fifth, we allow the agent to borrow money infinitely, such that it can also place a buy order at any point. However, the agent can only place a sell order if it has shares in its inventory. We consider this to be analogous to a trader depositing more money in their trading account. Although we note the presence of transaction fees. However, due to the relatively small volume of trades made by the RL agent, we consider these fees to be negligible, and as a result omit them.

4.1.1 State Space

We initially define our state space to model the task of trading our chosen stock. As such our state s at time t is:

$$\hat{s_t} = p_t \tag{33}$$

where p_t is the current price of a single share of stock. However, as stated above, a market participant will often times calculate additional performance indicators. So we allow the agent to incorporate the market indicators listed in section 3.1 into its state. This is analogous to a trader who observes and allows those indicators to influence their trading decisions. By including market indicators into agent's state, we are formally allowing the deep Q network to use those metrics when calculating $Q(s,a)\ \forall s \in S, \forall a \in A$. With the inclusion of market indicators are state resembles the following.

$$\tilde{s_t} = [p_t,\ MACD,\ RSI,\ CCI,\ ADX] \tag{34}$$

Additionally, we make two key observations about the deficiencies of the state as defined in equation 34. First the trader is more concerned about the difference in price Δ_t (As defined in 4.1.3) then the magnitude of the price itself, since the Δ is more closely related to the agent's profit. Second the trader not only has a memory but ought to use its memory of previous market movements when making a trading decision. Using these two observation's we define an improved state that gives the RL agent access to a tuple of the last N state delta's.

$$\begin{aligned} s_t &= [\Delta_{t-N+1}, \Delta_{t-N+2}, ..., \Delta_N] \\ &\in \mathbb{R}^{|\tilde{s_t}| \times N} \end{aligned} \tag{35}$$

where Δ is defined as follows

$$\Delta_t = p_t - p'_t \tag{36}$$

and $p_{t'}$ and p_t come from states $s_{t'}$ and s_t respectively. Specifically, t' is the time of purchase of the cheapest share in the agent's inventory. The intuition behind this construction decision is that an agent will always prioritize locking in profits through selling the stock in its inventory that will generate the largest Δ.

4.1.2 Action Space

The action space of the trading agent is whether to buy, hold, or sell the stock in each time step at the given state. As stated in the environment setup, our agent can only place one trade per day, and the transaction can only be of one share. As such, we define our action space to be:

$$A = \{-1, 0, 1\} \tag{37}$$

where -1 and 1 represent sales and purchases respectively, and 0 means holding.

4.1.3 Reward Function

The reward we set depends on the action our agent decides. If the agent places a sell order, we reward it according to the returns it realises, if the agent places a buy order, we reward it according to the risk of buying that share (i.e a small risk will generate a larger reward), and if the agent chooses to hold for that time step, we give it zero reward. Specifically, the reward R given the states at time t and t' with action a_t is defined as

$$R(s_t, a_t, s_{t'}) = \begin{cases} RiskMetric(s_t) & \text{if } a_t > 0, \\ \Delta_t & \text{if } a_t < 0, \\ 0 & \text{if } a_t = 0. \end{cases} \tag{38}$$

where our Risk Metric is a function to be chosen from those defined in Section 3.2, and Δ_t is defined as in equation 36. We expect that incorporating risk into our reward function will guide the trading agent to make risk sensitive trades similar to a human trader.

4.2 Deep Q Network Setup

4.2.1 Architecture

We implement a Deep Q Network System to compute optimal Q functions and choose appropriate actions. Here we choose three frameworks outline in section 2.1 to implement and test; V-DQN, T-DQN, and D-DQN.

Regardless of our choice of framework, we define all neural networks to have identical architecture. We choose an Artificial Neural Network with six fully connected layers including: our input layer of size $|\tilde{s_t}|N$, a hidden layers of size 128, two hidden layers of size 256, a hidden layer of size 128, and our output layer of size $|A_t|$. Additionally, we define the consistent learning rate to be $\alpha = 0.001$. For clarity, we graphically illustrate the architectures of each DQN we've chosen to implement bellow.

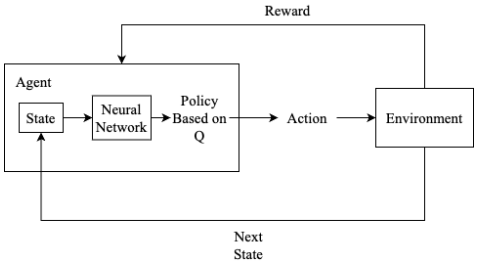


Figure 1: V-DQN Framework

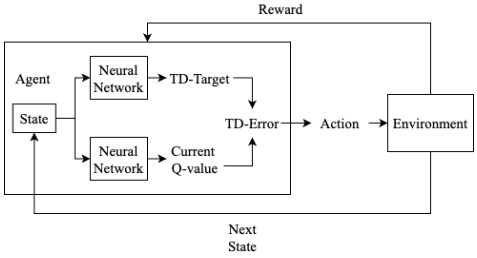


Figure 2: T-DQN Framework

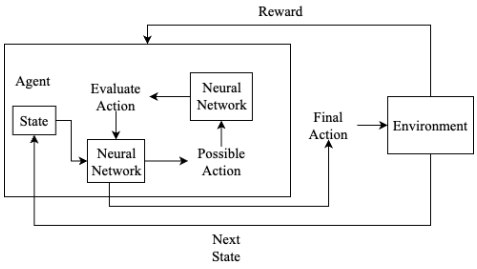


Figure 3: D-DQN Framework

4.2.2 Policy

In all choices above, we follow the ε-greedy policy to catalyze off-policy actions by introducing randomness. Formally ε-greedy is implemented by replacing the $\max_a Q(s_{k+1}, a)$ term in equation 5 with $Q(s_{k+1}, \tilde{a}_{k+1})$ where

$$\tilde{a}_{k+1} = \begin{cases} \text{random action from } A(s_t), & \text{with probability } \varepsilon \\ \arg\max_a Q(s_t, a), & \text{with probability } 1-\varepsilon \end{cases} \tag{39}$$

As training continues we slowly take $\varepsilon \to 0$ using a simulated annealing strategy. The reason why the ε-greedy strategy can be beneficial, is because it allows the agent to explore the space of possible actions freely at the beginning of the training process while also emphasizing exploitation of the agents accumulated knowledge toward the end of training. Please note that for the purpose of feeding the state into the deep Q network we must first normalize by applying a soft max function to each column of the matrix s_t, then we flatten the matrix.

4.2.3 Loss Function

We've defined the V-DQN loss function in equation 7 but in practice the loss is implemented using various approximations. First the expectation in the Q Learning loss function is approximated using experience replay by sampling a mini-batch of experiences from the replay buffer. Given a replay buffer D that contains experiences (s, a, r, s'), a mini-batch of N experiences $B = \{(s_i, a_i, r_i, s_i')\}_{i=1}^N$ is sampled uniformly at random from D. Using this strategy we can approximate the loss function in equation 7 (V-DQN Loss) as follows. The same strategy can be applied to the loss functions for T-DQN and D-DQN.

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\left(r_i + \gamma \max_{a'} Q(s_i', a'; \theta) - Q(s_i, a_i; \theta)\right)^2\right] \tag{40}$$

The next adjustment practitioners use is the integration of the the Huber loss. This is designed to make our loss function quadratic for small values of the error and linear for large values of the error. The parameter δ effectively determines the sensitivity of the loss function to outliers. Previously we defined TD Error in equation 5, which for the sake of integrating the Huber loss we'll equate to δ. This final improvement yields the following loss function.

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} L(\delta) \tag{41}$$

where

$$\delta = \left(r + \gamma \max_{a'} Q(s', a')\right) - Q(s, a) \tag{42}$$

$$L(\delta) = \begin{cases} \frac{1}{2}\delta^2 & \text{for } |\delta| \leq 1, \\ |\delta| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{43}$$

4.2.4 Optimizer

Lastly, we use the Adam optimizer in all frameworks. This is an adaptive optimization algorithm used in training our neural networks. It incorporates momentum to accelerate convergence in relevant directions and introduces bias correction to improve parameter update accuracy. By computing adaptive moments of gradients, Adam scales updates effectively and ensures numerical stability with a small constant in the denominator. These combined features make Adam a robust and efficient choice for optimizing our neural networks.

4.2.5 Hyper parameters

We use the following hyper parameters which we keep consistent across all implementations and frameworks: our discount rate $\gamma = 0.95$, the parameters for our epsilon-greedy policy $\varepsilon_0 = 1$, $\varepsilon_{\text{decay}} = 0.995$, $\varepsilon_{\text{min}} = 0.01$, the window size $N = 10$, and the number of training episodes per framework $E = 10$.

5 Experimental Design & Results

For the purpose of this paper we use daily Google stock data, including low,high,close,and adjusted close prices. Specifically we use Jan 1, 2018 - Jan 2019 data to train the model and Jan 1, 2019 - Jan 2020 data to validate the model we also allow the agent to retrieve the market indicators listed in section 3.1. to incorporate into its state space defined in section 4.1.1. All computations regarding the market indicators are defined in section 3.1.

We then test the following variations to find the best performing trading agent. Bellow are our results expressed in terms of percentage growth over the time. However, its important that we explain how we obtain the percent growth value given that we don't incorporate the idea of starting balance into our model. So in order to calculate percent growth we assume that the initial balance B_0 is the minimum balance across the episode. Intuitively if we assume $B_0 = |\min t B_t|$ we functionally enforce that the agent never goes bankrupt. Using this "implied initial balance" we calculate the percent change using $\frac{B_T - B_0}{B_0}$.

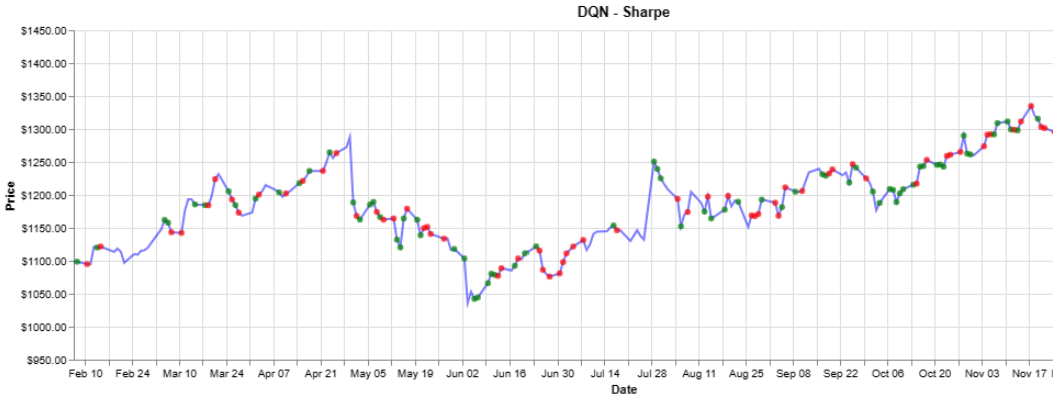
Metric	V-DQN	T-DQN	D-DQN
Sharpe	8.14%	13.05%	11.21%
Sortino	0.62%	-0.087%	2.61%
Treynor	6.68%	6.15%	10.26

Table 1: Performance Metrics of Different DQN Variants

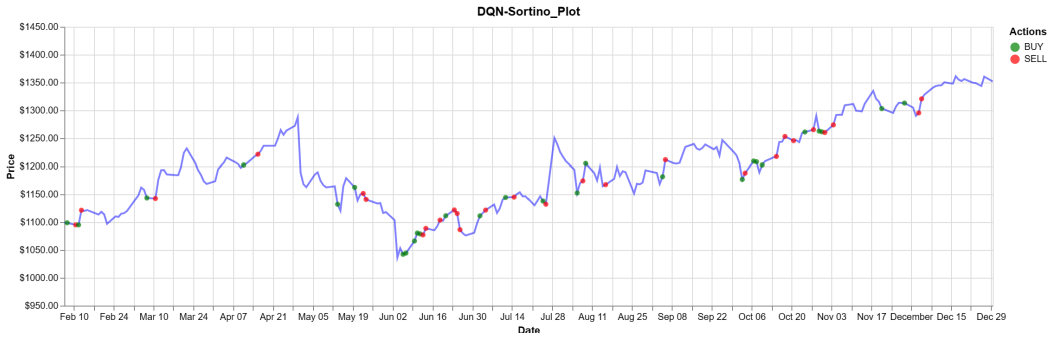
We note that the percentage growth in Google's stock from 2019-2020, was 27.75%, so each trading agent is still under performing the market. From table 1, we observe that the Sharpe ratio is better suited to guide the agent towards favorable trading decisions. We speculate that this is the case because the Sharpe ratio considers both systematic and unsystematic risk along with downside and upside risk through its use of the standard deviation of portfolio returns in the denominator. This makes it a comprehensive measure of risk-adjusted return, suitable for evaluating strategies that might be exposed to a wide range of market risks. On the other hand the Treynor ratio uses beta in the denominator and focuses on the systematic risk of the stock itself relative to the overall market. Sortino's ratio is also more limited in scope then the Sharpe ratio since it focuses on downside risk. We also invite the readers attention to the plots bellow, where one can observe an interesting feature in the D-DQN plots. We observe that the returns for the D-DQN algorithms outperform the other frameworks and that the buy and sell orders are more sparse then V-DQN. We hypothesize that this happens because Double DQN is able to effectively stabilize the TD target which Shields the agent's purchase behavior from idiosyncratic price shocks that may cause sporadic traeding behavior.

5.1 V - DQN Plots

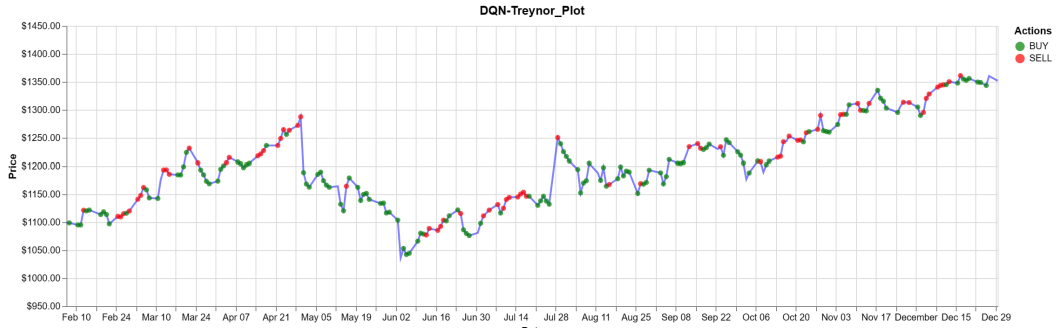
5.1.1 Sharpe



5.1.2 Sortino Plots

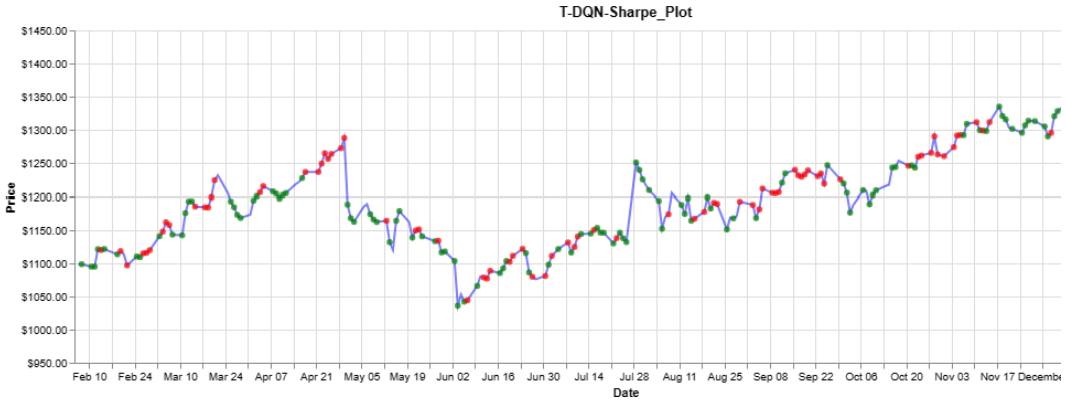


5.1.3 Treynor Plots



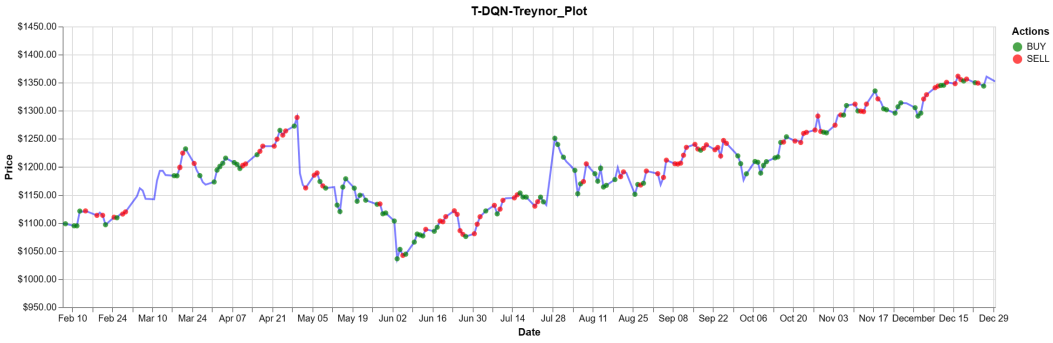
5.2 T - DQN Plots

5.2.1 Sharpe



5.2.2 Sortino

5.2.3 Treynor

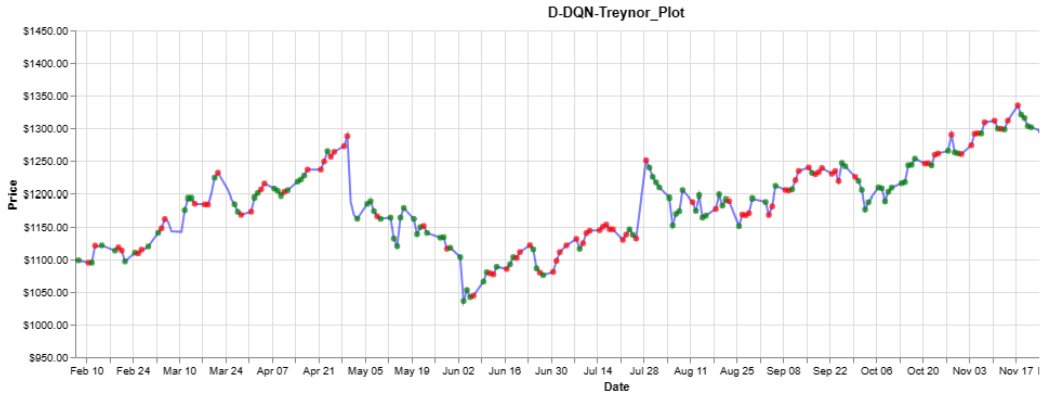


5.3 V - DQN

5.3.1 Sharpe

5.3.2 Sortino

5.3.3 Treynor



6 Future Improvements

While our agent yields positive returns, we identify numerous areas of possible further exploration to enable it to outperform that market. The first is the lack of training data. By presenting our agent with daily closing prices we limit the amount of actions it can take to one per day. This means that feedback is only as frequent as the agent's trade. Possible improvements to this could be training on a longer price history, or to provide more frequent pricing information. Next, we recognize that an upgraded agent could have the ability to place a trade more than one unit of stock per action. Specifically, we could allow the agent to trade up to k units of stock per day, where k would be another hyper-parameter to adjust. To accomplish this, the Deep Learning framework would need to be deeper in the number of layers and nodes, accounting for the higher dimensional action space. Further, we acknowledge that different neural network frameworks may present various advantages. For example, implementing a Long-Short Term Memory (LSTM) Recurrent Neural Network (RNN) could incorporate the concept of price memory without needing to manually maintain a window of states. Along the same lines, a Concurrent Neural Network (CNN) could be incorporated to understand our state space matrix and produce more informed evaluations of our action-state value functions. Additionally, incorporating more episodes in our training schemes could be greatly influential in yielding stronger results, but this would require more computing power and runtime. Lastly, limitations on when the agent can buy and sell units based on a notion of current balance vs. current holdings would make this project more seamlessly transferable to real world trading applications. In particular, introducing restrictions such as only being able to buy as many units as allowed by current balance, or having to sell units if balance is too low would more accurately simulate how humans trade.

References

1. Briola, Antonio, et al. Deep Reinforcement Learning for Active High Frequency Trading, 19 Aug. 2023, arxiv.org/abs/2101.07107.
2. Brunton, Steven Lee, and José Nathan Kutz. Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control. Cambridge University Press, 2021.
3. Dai, Zhenwen, et al. In-Context Exploration-Exploitation for Reinforcement Learning, 11 Mar. 2024, arxiv.org/abs/2403.06826.
4. De Asis, Kristopher, et al. Multi-Step Reinforcement Learning: A Unifying Algorithm, 11 June 2018, arxiv.org/abs/1703.01327.
5. Hill, R. Carter, et al. Principles of Econometrics. Wiley, 2018.
6. Luo, Yunfei, and Zhangqi Duan. Agent Performing Autonomous Stock Trading under Good and Bad Situations, 6 June 2023, arxiv.org/abs/2306.03985.
7. Mnih, Volodymyr, et al. Playing Atari with Deep Reinforcement Learning, 19 Dec. 2013, arxiv.org/abs/1312.5602.
8. Pricope, Tidor-Vlad. Deep Reinforcement Learning in Quantitative Algorithmic Trading: A Review, 31 May 2021, arxiv.org/abs/2106.00123.
9. Ravichandiran, Sudharsan. Deep Reinforcement Learning with Python: Master Classic RL, Deep RL, Distributional RL, Inverse RL, and More with Openai Gym and Tensorflow. Packt, 2020.
10. Singh, Prabhsimran. "PSKRUNNER14/Trading-BOT: Stock Trading Bot Using Deep Q Learning." GitHub, 31 Dec. 2020, github.com/pskrunner14/trading-bot/.
11. Sutton, Richard S., and Andrew Barto. Reinforcement Learning: An Introduction. The MIT Press, 2020.

Appendices

A Additional Risk Metrics

A.1 Value at Risk (VaR)

Value at Risk (VaR) quantifies the maximum expected loss over a specified time period within a given confidence interval, assuming normal market conditions. It is widely used to assess the risk of a portfolio.

$$\begin{aligned} \text{VaR}_\alpha(t) = -\inf\{x \\ : F(x) \\ > 1-\alpha\} \end{aligned} \tag{44}$$

A.2 Expected Shortfall (ES) or Conditional VaR (CVaR)

Expected Shortfall (ES), or Conditional VaR (CVaR), measures the expected loss on days when there is a VaR breach. It provides a more comprehensive view of tail risk than VaR by averaging losses in the tail beyond the VaR threshold.

$$\text{ES}_\alpha = \frac{1}{1-\alpha}\int_\alpha^1 \text{VaR}_u(L)du \tag{45}$$

B Additional Q Learning Schema

B.1 SARSA

SARSA is also a model-free RL scheme, but unlike classical Q Learning is on-policy. The advantage of SARSA is that it unlocks the full power of temporal difference learning by allowing the user to design an agent that can look n steps into the future when calculating the TD(n) error. In this setting we use all equations and definition listed above except for the update equation, which is now defined as follows.

$$Q^{\text{new}}(s_k,a_k) = Q^{\text{old}}(s_k,a) + \alpha(R_\Sigma^{(n)} - Q^{\text{old}}(s_k,a_k)) \tag{46}$$

$$R_\Sigma^{(n)} = (\sum_{j=1}^{n}\gamma^j r_{k+j}) + \gamma^{n+1}Q^{\text{old}}(s_{k+1},a) \tag{47}$$

Its very important to note that in order to calculate $r_k,...,r_{k+n}$ we must define which actions the agent will take at every future time step. Without that we can't know what the future rewards will be. So in order to solve this problem the SARSA model forces the agent to choose the optimal on-policy action at each projected time step.

There is a variation of TD learning that was developed by Sutton in 2015, where instead of looking at any n-step temporal difference, we take a exponential weighted average of all possible n-step temporal differences. This idea is known as TD-λ learning and can be formalized by defining the following update scheme.

$$R_\Sigma^\lambda = (1-\lambda)(\sum_{k=1}^{\infty}\lambda^{n-1}R_\Sigma^{(n)}) \tag{48}$$

$$Q^{\text{new}}(s_k,a_k) = Q^{\text{old}}(s_k,a) + \alpha(R_\Sigma^\lambda - Q^{\text{old}}(s_k,a_k)) \tag{49}$$

Recasting SARSA into a training scheme that incorporates neural networks would result in a loss function of the following form, where a user could implement n-step temporal difference learning or the TD(λ) framework described above.

$$\mathcal{L} = \mathbb{E}[(R_\Sigma^{(n)} - Q(s_k,a_k))^2] \tag{50}$$

B.2 $Q(\sigma)$ Learning

In 2018 Sutton and Barto first formulated the $Q(\sigma)$ method which was meant to unify the two fundamental approaches discussed above; Q Learning and SARSA. It introduces a parameter σ, which varies on a scale from 0 to 1, allowing the algorithm to interpolate between these two methods. This allows for $Q(\sigma)$ to adaptively balance between the exploration of new strategies (via Q Learning's off-policy nature) and the exploitation of current knowledge (via SARSA's on-policy nature), potentially leading to more efficient learning in complex environments.

$Q(\sigma)$ operates by dynamically adjusting the parameter σ for each time step of each episode, which determines the mix of on-policy (SARSA) and off-policy (Q Learning) learning. The updates to the Q values in $Q(\sigma)$ are based on a combination of the expected value (like in Q Learning) and the actual reward received plus the value of the next n state-action pair (like SARSA), with the balance between these two determined by σ. We can formalize $Q(\sigma)$'s update rule as

$$Q(S_t,A_t) = Q(S_t,A_t) + \alpha[R_{t+1} + \gamma(\sigma_{t+1}Q(S_{t+1},A_{t+1}) + (1-\sigma_{t+1})V_\pi(St+1)) - Q(S_t,A_t)] \tag{51}$$

The natural question to ask is how σ_t is chosen at every time step. Asis, Hernandez-Garcia, Holland, and Sutton proposed in 2018 that in most settings σ should be gradually decreased over the course of the episode in a manner similar to simulated annealing.

Furthermore, just like SARSA, we can expand this formulation to include multiple "look-forward" steps. The variation of $Q(\sigma)$ that incorporate TD(n) learning.

When recasting this scheme to a deep learning problem, we can apply identical reasoning as for SARSA to generating a loss function of $Q(\sigma)$ Learning.

$$\mathcal{L} = \mathbb{E}[R_{t+1} + \gamma(\sigma_{t+1}Q(S_{t+1},A_{t+1}) + (1-\sigma_{t+1})V_\pi(St+1)) - Q(S_t,A_t] \tag{52}$$

A Model Predictive Control & Deep Q Learning Approach to Wayfinding

Jad Soucar (jadsoucar@gmail.com)
USC Viterbi Daniel J. Epstein Department of Industrial
Systems Engineering 3715 McClintock Avenue, GER 240
Los Angeles, CA 90089-0193

November 29, 2024

Mentoring Organization: Red Hen Lab
Keywords: Reinforcement Learning, Agent Modeling, Control Theory, Wayfinding

1 Summary

For the last 200 years research into human cognition and decision-making has revolved around rational choice theory, which assumes that actors are utility maximizers capable of searching and finding rational and optimal decisions. However human behavioral data doesn't support the assumptions that we have infinite time nor infinite cognitive energy to search the full space of candidate choices to produce optimal decisions. Take for example a hiker, who has an infinite number of paths to choose from. It's irrational to assume that the hiker is capable of fully evaluating each of the infinite paths available to them to then choose the most rational/ optimal path. And yet that is the operating assumption for most modern cognitive models. To remedy this problem, I propose three axiomatic principles of energy-efficient decision making, along with an optimal-control based model and Deep Q learning model that capture those principles. I hope that a formalization of my theory of energy-efficient decision making along with the open-source python code I develop can be used by researchers in private and academic settings to develop more accurate computational models of human decision making, that can be used to model agent behavior in a number of fields ranging from market behavior to crowd dynamics.

Beyond classical applications to human and animal way finding, I believe that MPC and Deep Q learning based way finding models I will develop throughout this proposal have potential applications to the development of large language models. Currently LLM's operate using auto-regressive frameworks that are capable of searching the possibility space of the "next word" or token. The model then chooses the most likely next token based on the context embedded in the previous sequence of tokens. However the MPC and Deep Q frameworks Ill develop bellow have the potential capability to cost effectively search the space of the next N tokens to more efficiently plan the sequence of tokens being generated. While applications to LLMs and other generative AI algorithms may seem particularly interesting to the reader, I've chosen to formalize our theory of decision making by analyzing the classical way finding problem of foraging, since foraging is an especially intuitive example. Before ending with an explanation as to how the two models can be generalized to LLMs.

Proposal Structure

I will begin by proposing 3 principles of energy-efficient decision before introducing some relevant mathematical background for their computational implementation. I end by exploring two possible models of agent-based wayfinding that incorporate the 3 principles referenced above along with their applications.

2 Proposed Principles of Energy-Efficient Decision Making

Actors engaged in wayfinding will oftentimes miss the optimal trajectory towards their target, whether that be food, shelter or a mate, due to cognitive energy constraints. In order to construct mathematical models around this idea, I will first break down the concept of costly cognition, along with its natural consequences.

1. **Costly Cognition**: Recent literature suggests that cognition requires 20% of the body's total energy consumption. It's these energy constraints that limit an actor's ability to fully flush out a set of candidate decisions before adopting an action. Instead, a finite cognitive energy budget forces actors to develop rough sketches of various candidate decisions. Each of these sketches define a subspace of more detailed decisions that fall within the category of the decision sketch. In other words, cognitive energy constraints improve the efficiency of the decision-making process by constraining the space of possible decision, but can also guide actors towards energy-efficient choices by eliminating an optimal decision space during the initial screening process.

2. **The Subspace Selection Problem**: The costly cognition principal forces actors to develop decision subspaces in order to reduce cognitive energy load. The natural question is "How is that subspace selected?". I propose that the decision subspaces are evaluated using fuzzy and incomplete heuristics. The internal system then weighs the probability of success relative to the cognitive energy associated with exploring each of those subspaces. Only after this process is complete will an actor elect to conduct a more thorough search of a particular subspace to come to a final decision.

3. **Decision as Future Projections**: I've discussed how costly cognition forces actors to rely on rough decision sketches to choose a decision space to thoroughly explore. I propose that a human decision maker evaluates those sketches by constantly projecting themselves into possible futures based on those decision sketches. Each projection is made onto a temporal window chosen by the actor. For example, a decision made in the pursuit of a goal which the actor heavily values will encourage a projection further into the future than one of minimal importance to the actor. Regardless of the future projections size, an actor must consistently re-generate decision sketches at every time step, to adjust to new internal and external conditions. This process of constantly projecting oneself into the future to create a continuous stream of decisions produces the effect of a receding horizon. This conceptualization is in line with our intuition of decision making as a process that becomes increasingly fuzzy and uncertain the further forward in time an actor looks.

3 Mathematical Background

In this section I provide relevant mathematical foundation of deep reinforcement learning and model predictive control. In future sections I will use those two frameworks to construct two candidate models of cognitively constrained wayfinding that incorporates the 3 core principles of decision making outlined above

3.1 Deep Reinforcement Learning

3.1.1 General Reinforcement Learning

Before I begin my discussion of Q learning, I will start with an exposition of the two primary categories of RL: model-based and model-free. Model-based RL systems create a model of the environment, which includes the dynamics of how actions lead to subsequent states and rewards. Essentially, it predicts the future states and rewards for actions taken from a given state. Take for example a gambler who is given a probability function $P(s', s, a)$ that provides explicit probabilities as to whether the actor will win or lose a gamble. In other words, the gambler queries the environment for its transition dynamics. The second approach is model-free, which is where a system learns a policy or value function directly from interactions with the environment without constructing an explicit model of the environment's dynamics. The system learns what to do by trial and error, adjusting its actions based on the rewards received.

3.1.2 Q Learning

With the general background of RL out of the way, its important to note that Q learning is a model-free training scheme. That means that Q learning does not rely on an environment that can produce exact transition probabilities and must instead use trial and error of near-optimal actions in order to learn the optimal policy. In order to formalize this type of training scheme I first define the quality function $Q(s,a)$, which tells you the joint value/quality of taking an action a given a current state s.

To formally define $Q(s,a)$ I first introduce some notation. Let $R(s',s,a)$ be the reward of transitioning from state s to s' through action a, γ be the discount factor of future reward, and $V(s')$ be the expected value over all possible future rewards given a current state s'. With that I define $Q(s,a)$ as

$$Q(s,a) = \mathbb{E}\left(R(s',s,a) + \gamma V(s')\right) = \sum_{s'} P(s'|s,a)(R(s',s,a) + \gamma V(s')) \tag{1}$$

In other words, Q is the expected sum of the instantaneous reward of the state-action pair (s,a) along with the discounted future rewards of being at a new state s' brought on by (s,a). Using the quality function $Q(s,a)$ we can define an optimal policy π and value function $V(s)$, that considers which action a is optimal and what the expected reward from taking that action is.

$$V(s) = \max_a Q(s,a) \tag{2}$$

$$\pi(s) = \arg\max_a Q(s,a) \tag{3}$$

We've defined what a q value is and how to construct the quality function $Q(s,a)$, but now we define a recursive equation to update $Q(s,a)$ as the agent learns through trial and error.

$$Q^{\text{new}}(s_k,a_k) = Q^{\text{old}}(s_k,a_k) + \alpha\left(r_k + \gamma \max_a Q^{\text{old}}(s_{k+1},a) - Q^{\text{old}}(s_k,a_k)\right) \tag{4}$$

Let's dissect this update equation. As we engage in trial and error learning, we update our $Q(s,a)$ by slowly nudging our q values up or down by a factor of the difference between the actualized current reward r_k in addition to the best possible future reward $r_k + \gamma \max_a Q(s_{k+1},a)$ (TD-Target) and our predicted reward $Q^{\text{old}}(s_k,a_k)$). This difference is sometimes referred to as the "Temporal Difference (TD) error". We also note that α in this case is the learning rate.

$$\underbrace{\overbrace{r_k + \gamma \max_a Q(s_{k+1},a;\theta^-)}^{\text{TD Target}} - Q^{\text{old}}(s_k,a_k;\theta)}_{\text{TD Error}} \tag{5}$$

We also note that when calculating the TD error, the agent calculates the future reward by looking one step into the future. Naturally, the degree to which the agent looks into the future can be modified using the following updated scheme which looks n steps into the future. This process is known as TD-N learning.

$$Q^{\text{new}}(s_k,a_k) = Q^{\text{old}}(s_k,a) + \alpha(r_k + R_\Sigma^{(n)} - Q^{\text{old}}(s_k,a_k)) \tag{6}$$

$$R_\Sigma^{(n)} = (\sum_{j=1}^{n} \gamma^j r_{k+j}) \tag{7}$$

where r_{k+j} is the reward derived from future applications of the agent's chosen policy π.

3.1.3 Deep Q-Learning

Now we discuss the process of recasting Q Learning into training schemes that incorporate deep learning methods like neural networks. This is especially useful in settings where the state space is too large to reasonably store all q values in a Q table. Here, a deep learning approach seeks to parameterize $Q(s,a)$ as dependant on some weights θ such that

$$Q(s,a) \approx Q(s,a,\theta) \tag{8}$$

In order to optimize for the parameters θ within a Q Learning framework we use a loss function that is eerily similar to the TD error defined in equation (4).

$$\mathcal{L} = \mathbb{E}[(r_k + \gamma \max_a Q(s_{k+1},a,\theta) - Q(s_k,a_k,\theta))^2] \tag{9}$$

We can also choose to recast n-step temporal difference learning framework (TD-N) described above into the following neural network loss function.

$$\mathcal{L} = \mathbb{E}[(r_k + R_\Sigma^{(n)} - Q(s_k,a_k))^2] \tag{10}$$

With the loss function properly defined, we can move on to the neural network architecture, and how it allows us to approximate the Q function. In practice, we find that the architecture of the network varies based on the use case. For example a team of DeepMind engineers in 2013 coupled the loss function described above with several convolutions layers and fully connected layers. The inputs were several consecutive frames for the game, which represented the agent's state. And the output was one of several possible action that the agent could take. Regardless of the architecture used the value contained in each output node approximates the value $Q(s,a)$ for the associated (s,a) which corresponds to the network's $(\text{Input}, \text{output})$ pair.

3.1.4 Policy

In practice, we follow the ε-greedy policy to catalyze off-policy actions by introducing randomness. Formally ε-greedy is implemented by replacing the $\max_a Q(s_{k+1},a)$ term in equation 5 with $Q(s_{k+1},\tilde{a}_{k+1})$ where

$$\tilde{a}_{k+1} = \begin{cases} \text{random action from } A(s_t), & \text{with probability } \varepsilon \\ \arg\max_a Q(s_t,a), & \text{with probability } 1-\varepsilon \end{cases} \tag{11}$$

As training continues we slowly take $\varepsilon \to 0$ using a simulated annealing strategy. The reason the ε-greedy strategy can be beneficial is that it allows the agent to explore the space of possible actions freely at the beginning of the training process while also emphasizing exploitation of the agent's accumulated knowledge toward the end of training. Please note that for the purpose of feeding the state into the deep Q network, we must first normalize by applying a soft max function to each column of the matrix s_t and then flatten the matrix.

3.1.5 Loss Function

We've defined the DQN loss function in equation (9) and (10) but in practice the loss is implemented using various approximations. First the expectation in the Q Learning loss function is approximated using experience replay by sampling a mini-batch of experiences from the replay buffer. Given a replay buffer D that contains experiences (s,a,r,s'), a mini-batch of N experiences $B = \{(s_i,a_i,r_i,s_i')\}_{i=1}^N$ is sampled uniformly at random from D. Using this strategy we can approximate the loss function in equation (9) as follows.

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\left(r_i + \gamma \max_{a'} Q(s_i',a';\theta) - Q(s_i,a_i;\theta)\right)^2\right] \tag{12}$$

The next adjustment practitioners use is the integration of the the Huber loss. This is designed to make our loss function quadratic for small values of the error and linear for large values of the error. The parameter δ effectively determines the sensitivity of the loss function to outliers. Previously we defined TD Error in

equation (5), which for the sake of integrating the Huber loss we'll equate to δ. This final improvement yields the following loss function.

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} L(\delta) \tag{13}$$

where

$$\delta = \left(r + \gamma \max_{a'} Q(s', a')\right) - Q(s, a) \tag{14}$$

$$L(\delta) = \begin{cases} \frac{1}{2}\delta^2 & \text{for } |\delta| \leq 1, \\ |\delta| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{15}$$

3.1.6 Optimizer

Lastly, in practice the Adam optimizer is used to minimize the loss function. This is an adaptive optimization algorithm used in training our neural networks. It incorporates momentum to accelerate convergence in relevant directions and introduces bias correction to improve parameter update accuracy. By computing adaptive moments of gradients, Adam scales updates effectively and ensures numerical stability with a small constant in the denominator. These combined features make Adam a robust and efficient choice for optimizing our neural networks.

3.2 Model Predictive Control

The general formulation of MPC involves solving an optimization problem at each control step. The objective is to find the control inputs that minimize the objective function subject to the model dynamics and constraints.The general setup for an MPC problem is as follows.

1. **Choice Functional**:

$$J(U_t^{t+\Delta}, x_t) = \sum_{k=t}^{t+\Delta} l(x(k), u(k)) + F(x(T)) \tag{16}$$

   Where:

   - J: The choice function to minimize.
   - $u(t)$: The discrete function of control inputs
   - $U_t^{t+\Delta}$: A sequence of future control inputs $u(t), u(t+1), \ldots, u(t+\Delta)$. For the sake of simplicity the subscript of U denotes when the sequence begins, and the superscript denotes when it ends.
   - x_t: The current state of the system.
   - l: The stage cost function, evaluating the cost of each predicted state and control action.
   - F: The terminal cost function, evaluating the cost of the final state.
   - T: Terminal Time End of Episode.
   - Δ: Predictive Horizon

2. **Subject to**:

   - System dynamics: $x(k+1) = f(x(k), u(k))$
   - Initial condition: $x(0) = x_0$
   - Control and state constraints: $u(k) \in A, x(k) \in S$, where A is the action space and S is the state space.

3. **Iterative Algorithm:** The user then iteratively implements the following algorithm to trace the evolution of an agent's state from a set of initial conditions to a desired terminal state.

(a) Initialize the agent at state x_0

(b) Optimize a choice functional over the predictive horizon. In other words find $\tilde{u} = \arg\min_U J(U_0^\Delta), x_0)$ to determine the next "best course of action" U_0^Δ

(c) Apply the system dynamics to determine the next state $x_1 = f(x_0, \tilde{u}(0))$. *Note that despite finding a sequence of controls from* 0 *to* Δ*, the agent only applies the first control in the sequence.*

(d) Reinitialize the agent at x_1, and optimize the choice functional $J(U_1^{1+\Delta}, x_1)$ over the time horizon.

(e) Iteratively apply steps (a)-(c) until the terminal time N is reached.

4 Candidate Wayfinding Models

In this section we propose two wayfinding models. One model will be built using a model predictive control (MPC) framework, while the other builds upon principles from Reinforcement Learning (RL). The two models take slightly different approaches to the wayfinding problem. The core difference being that the RL system focuses on single priority agents with future projections of variable size, while the MPC model focuses on multi-priority agents with future projections of fixed size. Before we discuss the architecture of the two candidate models, we'll begin with some core assumptions.

4.1 Core Assumptions

We begin by outlining several core operating assumption that will be used for both the MPC and RL models.

First we assume that an agent is capable of traversing a $d = 2,3$ dimensional rectangle. We may choose to define a topology G within $\mathcal{D}$. G may be mountainous, flat, or another terrain the user chooses.

$$\mathcal{D} = [-\delta, \delta]x...x[-\delta, \delta] \in \mathbb{R}^d \tag{17}$$

where food sources F is a finite set of coordinate in $\mathcal{D}$ such that every $f \in F$ is picked from D according to a uniform distribution $f \sim U(\mathcal{D})$. We also assume that the space $\mathcal{D}$ is populated with finite environmental information. We define a set of environmental stimuli where i_n is an environmental stimulus like scent, auditory noise, temperature, or a binary flag b for whether food exists at a specific coordinate, and x is that information's coordinate position in $\mathbb{R}^d$.

$$\forall i \in I, i = \{i_1, ..., i_n, b, x \in \mathbb{R}^d\} \tag{18}$$

Next we assume that the agent is equipped with energy that can be spent on taking an action or on the cognitive cost of deciding which action to take. So we let the agent's energy at time t be E_t.

$$E_t \in [0, 1] \tag{19}$$

Next we assume that an agent is capable of collecting information from its environment. However we only allow the agent to collect information from within a radius r around its current location. This is a realistic restriction since no agent can reasonably be assumed to have complete and perfect knowledge of its environment. So the information that an agent can access at any given time is defined as follows, where $s_{t,x}$ is the agents current position in $\mathbb{R}^d$

$$I_{r,t} = \{i = \{i_1, ..., i_2, x \in \mathbb{R}^d\} \in I \text{ if } |x - s_{t,x}| \leq r\} \tag{20}$$

Using the assumptions outlined above we can define the agents state s_t as a tuple of the agent's energy, current information, and current position in space at a given time t. The state space S contains all possible s_t.

$$s_t = \{E_t, I_{r,t}, s_{t,x}\} \tag{21}$$

The action space A will be defined as follows, where ϕ is the magnitude of the agent's step and θ is the direction of the step. For the purposes of our simulation, both θ and ϕ are discretized.

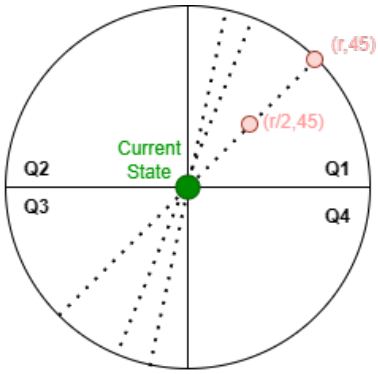


Figure 1: Action Space Visualization

$$\forall a \in A, a = \{\theta, \phi\} \tag{22}$$

If an agent takes an action $a = \{\theta, \phi\}$, we assume that the energy expended on taking the action is directly tied to the terrain G, since walking up a hill or mountain would naturally require more energy then walking downhill or on flat ground. We define this change in energy using the following dynamics:

$$c(s_{t,x}, a_t) = -\beta \nabla G(s_{t,x} + a_t) \tag{23}$$

Finally the reward function will be a piecewise function that represents how close an agent is to its goal. Note that the agent receives a significantly larger reward once it actually attains its goal, which we define as the agent reaching the source within a small margin ε. Conversely, if the energy is depleted below some threshold T_E, the agent receives a negative reward, to disincentives a zero energy state, which constitutes death.

$$R(s_t) = \begin{cases} 1 & \text{if } \exists f \in F \text{ s.t } |f - s_{t,x}| < \varepsilon, \\ -\beta_1 s_{t,E} & \text{if } s_{t,E} < T_E, \\ \beta_2 |f - s_{t,x}| & \text{otherwise.} \end{cases} \tag{24}$$

4.2 Deep Q Learning Wayfinding Model

We now transition to a possible application of the Deep-Q Learning Framework to the wayfinding problem. For our purposes we use Q-Learning since its model-free nature more accurately reflects an actor's uncertainty regarding the exact outcomes of its actions when engaging in the wayfinding process. Using the state space, action space, environment, and reward function described above in section 4.1 we construct the following deep Q learning architecture.

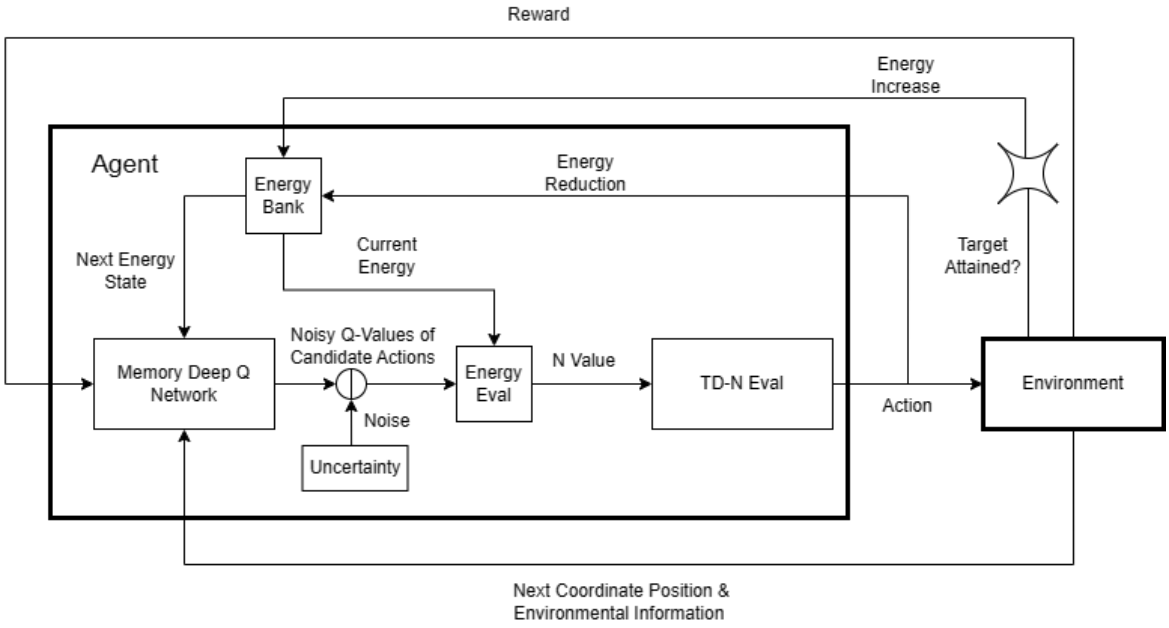


Figure 2: RL Framework

1. We begin be giving the agent prior knowledge of it environment through a fuzzy memory. This fuzzy memory functionally serves as the agent's intuition. To do this we train a Deep Q Network using the TD-N loss function defined in equation (10), along with the experience replay and Huber loss defined in section 3.1.5. We note that for training the memory deep Q network we fix N. We also use an epsilon greedy policy with an epsilon that converges to 0.33 towards the end of the training epoch. Intuitively the reason for such a high epsilon is to ensure that the agent's memory deep q network is populated with a wide breadth of experiences. The algorithm in trained using a the classical vanilla deep Q network architecture diagrammed bellow.

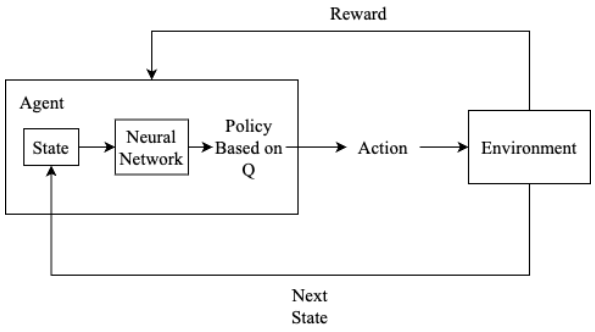


Figure 3: V-DQN Framework

We define the neural network unit in the diagram above to have the convolutional architecture outlined in figure 4. The reason why we choose a convolutional architecture is because the environmental information along with the agent's current coordinate position is inherently spatial and ought to be represented using a tensor in $\mathbb{R}^d$ instead of a flattened vector. We also exclude the energy E_t from the initial input, and instead only input $s_{t,x}$ and $I_{t,x}$. Because the energy cannot be spatially represented its concatenated to the flattened vector labeled bellow.

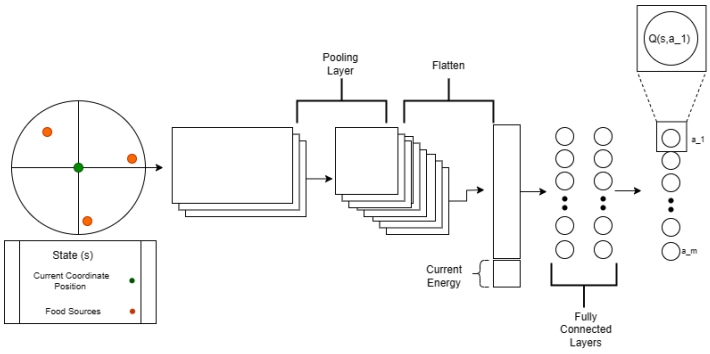


Figure 4: V-DQN Framework

Note that the memory deep network module is simply an approximation of a the Q function which takes an input of a state and outputs a estimated value of that state given each action in the action space. However because memory is often fuzzy and realistic agents don't have perfect recall we add slight Gaussian noise to the output. We define the Q function approximated by the Memory Deep Q Network as Q^{MB}

$$Q^{\text{MB}}(s) = \{(s,a,q(s,a_1)),(s,a,q(s,a_2)),...,(s,a_m,q(s,a_2))\} + \mathcal{N}(0,\sigma) \tag{25}$$

2. Next we initialize our agent with an initial coordinate position in $\mathbb{R}^d$, and a complete energy of $E_0 = 1$. Given that the agent is initialized with a random initial coordinate position, the agent will consequently gain access to all information within a radius r around its initial position. In other words the agent begins the simulation with a initial state of s_0 as defined above.

3. The agent then queries its memory bank for noisy q-values of all possible actions given its state. Those values are then passed to the energy evaluation unit. Within the energy evaluation unit the agent evaluates the average possible reward of each quadrant. This done by calculating quadrant-wise q-value averages. Then based on the magnitude of the possible reward the agent assigns an N value for each quadrant. Next the energy evaluation unit measures whether the cognitive cost of projecting forward N steps in each quadrant is less then the projected reward. If the cognitive cost is larger, the agent ignores that quadrant. Furthermore, if the amount of energy E_t is too low to afford exploring all promising quadrants, only the top K quadrants are chosen. Intuitively what that means is that the agent is only dedicating cognitive energy to create future projections of itself in decision subspaces that the agent can afford and whose potential reward outweigh the expected cognitive costs.

4. The agent then picks the actions in each quadrants selected which provide the largest q-value. Using those selected actions the TD-N evaluation unit finds the TD-N Target defined in equation (5) for each quadrant given the N value that was recommended by the energy evaluation unit. Note that the TD-N calculation is created using the policy recommended by the Memory Deep Q Network. The TD-N evaluation unit then chooses the course of action with the highest TD-N Target. The agent then applies that action to produce its next coordinate position. The energy cost of the action defined in equation (23) along with the cognitive energy used during the TD-N evaluation process (which is positively correlated to the amount of quadrants explore and the magnitude of the N values for each quadrant) is subtracted from the total energy E_t stored in the Energy bank.

5. Now that the agent has applied an action which produces its next coordinate position, it can interact with the environment to update its state with the new information $I_{t,r}$ it has access to and determine whether its reward. The way we've defined the reward function in equation (24) implies that the agent will receive a higher reward the closer it is to the food source. If a food source is attained, this triggers an energy increase for the agent. If the energy is depleted below some threshold T_E, the agent receives a negative reward. The last step involves storing the state, action pair that the agent chose along with the next state

and reward that state action pair generated in a memory bank that is used to retrain the Memory Deep Q Network.

6. Periodically, the Memory Deep Q Network Q^{MB} is retrained using the new experiences it has stored in its memory bank using the mini-batching technique outlined in section 3.1.5 along with the same architecture outlined in Step 1. Intuitively these periodic retraining session can be interpreted as an agent sleeping, since recent research indicates that sleep and memory consolidation are related processes that involve integrating new information into long-term memory.

7. Note that the only terminal state for the agent is when energy reaches 0 or if the simulation runs until the max iterations set by the user.

4.3 Model Predictive Control Wayfinding Model

We now transition to a possible application of the Model Predictive Control (MPC) to the wayfinding problem. Due to the flexibility of the MPC model framework, we allow our agent to have multiple priorities. This seems like a natural extension since most wayfinders are forced to balance multiple priorities. Take for example a mammal who must balance its priorities of finding food, shelter and a mate. For the purposes of this model we impose that the agent's set of priorities F is finite.

$$F = \{P_1, P_2, ..., P_m\} \tag{26}$$

In order to account for a multi-priority agent we make several small alterations to how we define the environment, environmental information, and reward function in the section 4.1. First we adjust the environment we create m goal sets F_i for each priority each of which is a finite set of coordinates in $\mathcal{D}$ such that every $f_j \in F_i$ is picked from D according to a uniform distribution $f_j \sim U(\mathcal{D})$. For example if priority P_k is finding shelter, then there would be an associated finite set of coordinates F_k that represents where in $\mathcal{D}$ shelter exists. This means that the elements in the set of environmental stimuli I would also include binary flag b_i for whether a goal associated with each priority exists at a specific coordinate.

$$\forall i \in I, i = \{i_1, ..., i_n, b_1, b_2, ..., b_m, x \in \mathbb{R}^d\} \tag{27}$$

This adjustment means we must also add a new reward function for each priority. So our new generalized definition of the reward function defined in equation (24), would be as follows.

$$R_i(s_t) = \begin{cases} 1 & \text{if } \exists f \in F_i \text{ s.t } |f - s_{t,x}| < \varepsilon, \\ -\beta_1 s_{t,E} & \text{if } s_{t,E} < T_E, \\ \beta_2 |f - s_{t,x}| & \text{otherwise.} \end{cases} \tag{28}$$

With those core adjustments defined, we can draw our attention to the central question for multi-priority agents; How does the agent choose which priority to focus on. To answer this question we propose an activation function a_n, which measures to what extent an agent should focus on its n^{th} priority.

$$a_n(s_t) = \frac{R_n(s_t)\mathbb{P}_n(s_t)}{D(P_n)\prod_0^{n-1} R_n(s_t)\mathbb{P}_n(s_t)\prod_{n+1}^{m} R_n(s_t)\mathbb{P}_n(s_t)} \tag{29}$$

Let break down how this activation function works. The activation function is directly related to the the reward of being in a current state with respect to the n^{th} priority, weighted by the probability of that reward being attained. The probability measure $\mathbb{P}_n$ incorporates both the agent's intuitive and experiential understanding of the environment signals i_n to produce a probability of success. In other words the higher the expected reward of pursuing the n^{th} priority, the more likely that the agent will choose to pursue that priority. However, the activation function is inversely related to $D(P_n)$, which represents the cognitive complexity of attaining the goals of the n^{th} priority. This is fairly intuitive since an agent will tend to steer away from complex priorities due to their large cognitive costs unless the expected reward is sufficiently large. Lastly the activation function

of the n^{th} priority is inversely related to the expected rewards of the other priorities. In this way, the activation function also models the opportunity cost of pursuing one priority over another.

Next we construct a choice functional that must be maximized at every given time step of the simulation, similar to that defined in equation (16). Where $U_t^{t+\Delta}$ is the sequence of actions $u(k) \in A$ from t to $t+\Delta$.

$$J(U_t^{t+\Delta}, s_{t,xx}) = \sum_{k=t}^{t+\Delta} \sum_{j=1}^{m} a_j(s_t) R_j(s_{k,x}, u(k)) \tag{30}$$

Through this choice function the decision to name the a_n the activation function becomes more clear, since a_n quite literally switches certain priorities on and off. Next we define the agent's dynamics. In order to adhere to the costly cognition principle, the agent's dynamics must be directly linked to its cognitive energy consumption. In other words if the agent chooses to explore a subspace that is relatively complex, its energy state E_t must be reduced, and if a goal like finding food or shelter is reached, its energy state must be reverted to the maximum energy state of 1. We formalize the system's dynamics below.

$$\tilde{d} = \sum_{j=0}^{m} a_j(s_t) D(P_j) \tag{31}$$

$$E_{t+1} = H(s_t, a_t) \tag{32}$$

$$H(s_t, a_t) = \begin{cases} E_t - \beta_1 \tilde{d} - c(s_{t,x}, a_t) & \text{if } \sum_{j=0}^{m} R_m(s_t) <= 1 \\ 1 & \text{otherwise.} \end{cases} \tag{33}$$

$\tilde{d}$ is the weighted complexity of the subspace chosen. $\tilde{c}(a)$ is the energy cost of taking an action and $\tilde{c}(s_{t,x}, a_t)$ is defined in equation (23). With the choice functional and system dynamics defined we implement the same iterative algorithm outlined in section 3.2. By design this scheme captures that core principles of energy-efficient decision making since at every time step the actor scans over a number of cognitively feasible decision subspaces, selects a subspace, evaluates candidate futures within its horizon, collapses down to a single decision, then iterates.

Algorithm 1 Model Predictive Control Algorithm

$s_{t,x} = x_0$, $t = 0$, $E_t = E_0$ $N =$ **Terminal Time**
while $t < N$ **do**
  $\tilde{u} = \arg\min_U J(U_t^{t+\Delta}, s_{t,x})$
  $\tilde{d} = \sum_{j=0}^{m} a_j(s_t) D(P_j)$
  $E_{t+1} = H(s_t, \tilde{u}(t))$
  $s_{t,x} = s_{t,x} + \tilde{u}(t)$
  $t = t + 1$
end while

We propose one small adjustment to the algorithm listed in section 3.2 and formalized above in order to imbue the agent with a sense of memory. Recall that MPC functions by making projections over a time horizon Δ, but only uses the first part of that projection to adjust its behavior before generating another projection. Instead we propose taking a weighted linear combination of the K past projections. For example if an agent is currently at time t, and is optimizing the function J to determine an action at time t, then it would weight the actions taken at time t in each of the last K future projection control sequence $U_{t-k}^{t+\Delta-k}(t)$. We could also reasonably adjust the weights γ of the linear combination to reflect the degree to which an agent prioritizes new future projections over older projections.

Algorithm 2 Model Predictive Control Algorithm With Memory

$s_{t,x} = x_0$, $t = 0$, $E_t = E_0$ $N =$ **Terminal Time**
while $t < N$ **do**
    $\tilde{u} = \sum_{k=0}^{K} \gamma^k \arg\min_U J(U_{t-k}^{t-k+\Delta}, s_{t-k,x})$
    $\tilde{d} = \sum_{j=0}^{m} a_j(s_t) D(P_j)$
    $E_{t+1} = H(s_t, \tilde{u}(t))$
    $s_{t,x} = s_{t,x} + \tilde{u}(t)$
    $t = t + 1$
end while

5 A Potential Application to Large Language Models (LLMs)

As reference in section 1, the applications of energy efficient agent-based decision making models like the MPC and RL models developed above have potential applications that span far beyond basic food/shelter foraging. In fact we propose that the models we developed in this proposal have direct applications to the development of more efficient LLM models.

For the purposes of applying our 2 models to improving LLM models, we must first re frame the LLM model as an agent in and of itself. This agent has an evolving state which is an expanding sequence of words/tokens t_i. Whereas the action space, of the agent is the next word in the sequence that the agent can predict.

$$s_n = \{t_1, t_2, t_3, ..., t_n\} \tag{34}$$

$$A = \{a - z, 0 - 9, \tag{35}$$

At every time step the the agent evaluates which token in the action space A maximizes a likelihood of that token coming next given a current sequence of tokens s_t, then chooses that token and adjusts the state. Under this framing, it becomes fairly clear why an RL or MPC model can be useful in this setting. In the case of the MPC and RL models, the LLM agent would be capable of predicting N tokens in advance before making a prediction. Furthermore the two models also allow the LLM agent to actively explore multiple sub spaces of potential token sequences in an energy-efficient manner before committing to a prediction. This approach may allow LLMs to avoid providing inaccurate information by blindly choosing the next most probable token, and replacing that approach with one that can generate text in the same way as a measured human speaker (i.e Thinking before you speak!).

References

1. Briola, Antonio, et al. Deep Reinforcement Learning for Active High Frequency Trading, 19 Aug. 2023, arxiv.org/abs/2101.07107.
2. Brunton, Steven Lee, and José Nathan Kutz. Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control. Cambridge University Press, 2021.
3. Dai, Zhenwen, et al. In-Context Exploration-Exploitation for Reinforcement Learning, 11 Mar. 2024, arxiv.org/abs/2403.06826.
4. De Asis, Kristopher, et al. Multi-Step Reinforcement Learning: A Unifying Algorithm, 11 June 2018, arxiv.org/abs/1703.01327.
5. Hill, R. Carter, et al. Principles of Econometrics. Wiley, 2018.
6. Luo, Yunfei, and Zhangqi Duan. Agent Performing Autonomous Stock Trading under Good and Bad Situations, 6 June 2023, arxiv.org/abs/2306.03985.
7. Mnih, Volodymyr, et al. Playing Atari with Deep Reinforcement Learning, 19 Dec. 2013, arxiv.org/abs/1312.5602.
8. Pricope, Tidor-Vlad. Deep Reinforcement Learning in Quantitative Algorithmic Trading: A Review, 31 May 2021, arxiv.org/abs/2106.00123.
9. Ravichandiran, Sudharsan. Deep Reinforcement Learning with Python: Master Classic RL, Deep RL, Distributional RL, Inverse RL, and More with Openai Gym and Tensorflow. Packt, 2020.
10. Singh, Prabhsimran. "PSKRUNNER14/Trading-BOT: Stock Trading Bot Using Deep Q Learning." GitHub, 31 Dec. 2020, github.com/pskrunner14/trading-bot/.
11. Sutton, Richard S., and Andrew Barto. Reinforcement Learning: An Introduction. The MIT Press, 2020.
12. Engl, E. and Attwell, D. Non-signalling energy use in the brain. *J Physiol*, 593: 3417-3429, 2015.
13. McCubbins, Colin H. and McCubbins, Mathew D. and Turner, Mark B., Building a New Rationality from the New Cognitive Neuroscience, July 9, 2018, in Riccardo Viale and Konstantinos Katsikopoulos (eds.) *Handbook on Bounded Rationality*, Routledge Publishing House (Forthcoming), Duke Law School Public Law & Legal Theory Series No. 2018-52.
14. Sarishma, D., Sangwan, S., Tomar, R., Srivastava, R. A Review on Cognitive Computational Neuroscience: Overview, Models, and Applications. In: Tomar, R., Hina, M.D., Zitouni, R., Ramdane-Cherif, A. (eds) *Innovative Trends in Computational Intelligence*. EAI/Springer Innovations in Communication and Computing. Springer, Cham, 2022.
15. Slovic, Paul, Finucane, Melissa, Peters, Ellen, MacGregor, Donald G. Rational actors or rational fools: implications of the affect heuristic for behavioral economics, *The Journal of Socio-Economics*, Volume 31, Issue 4, 2002, Pages 329-342, ISSN 1053-5357.
16. National Research Council (US) Committee on Opportunities in Neuroscience for Future Army Applications. Opportunities in Neuroscience for Future Army Applications. Washington (DC): National Academies Press (US); 2009. 4, Optimizing Decision Making.
17. Saks, Michael J., and Kidd, Robert F. "Human information processing and adjudication: Trial by heuristics." *Law & Soc'y Rev*. 15 (1980): 123.
18. Sarkar, Soumodip, Osiyevskyy, Oleksiy. Organizational change and rigidity during crisis: A review of the paradox, *European Management Journal*, Volume 36, Issue 1, 2018, Pages 47-58.

Research in Industrial Projects for Students

Sponsor

NASA Ames Research Center

Final Report

Classical Simulation of Non-Clifford Noise Channels in Stabilizer Formalism

Student Members

Jad Soucar (Project Manager), *University of California, Los Angeles*, jadsoucar@g.ucla.edu

Brandin Farris, *Oregon State Univeristy*

Daley McMahon, *University of Pennsylvania*

Peter Ye, *Yale University*

Academic Mentor

Zhuoyang (John) Ye, yezhuoyang98@g.ucla.edu

Sponsoring Mentors

Namit Anand, Namit.Anand@us.kbr.com

Jeffrey Marshall, jmarshall@usra.edu

Date: August 15, 2024

This project was jointly supported by NASA Ames Research Center and NSF Grant DMS 1925919.

Abstract

We introduce the basics of quantum computing and simulation of quantum systems on classical computers. We then discuss noise in quantum systems and how it is classically modelled, along with the difficulties of simulating quantum noise on classical computers. Our primary question is how to efficiently simulate quantum noise by leveraging existing techniques based upon the Gottesman-Knill theorem, which provides efficient simulation of circuits containing only Clifford gates. We follow this by describing our progress thus far in exploring three primary approaches. First, exploring methods to decompose a noise operator into a sum of cliffords via mixed integer linear programming and using these decompositions to classically simulate noisy quantum circuits within the stabilizer formalism first proposed by Aaronson and Gottesman (2004). We find that the a Clifford decomposition is not guaranteed to have low rank decompositions and that at best the runtime of simulating noise using Clifford decompositions would be $O(2^k)$, where k is the number of Kraus operators applied. Second, we explore the process of dilating our space to convert our noise operators into unitaries in a larger space. Specifically we propose two unitary dilation based algorithms for simulating noise; Sz.-Nagy and Stinespring's dilation algorithms. The two algorithms yield respective run-times of $O(\frac{1}{\delta\Delta^2} s n^3 \eta^{-2} 1.17^t)$ and $O(n^2 \prod_{i=1}^{k} |\xi_i| + n^3 \sum_{i=1}^{k} |\xi_i|^3)$. Third we propose a theoretical framework for generalizing the T-Gadget approach developed by Bravyi and Gosset (2016). Through numerical simulations we show that the generalized framework can be used to produce noise simulation algorithms with efficient runtime and space complexity.

Contents

Chapter 1

Introduction

In the early 1980s, Feynman and Manin among others proposed the concept of a so-called quantum computer that would simulate quantum systems more effectively than classical computers, following the development of quantum computing theory [1; 13, ch. 4]. Since then, quantum computing has been developed, studied, and applied to problems beyond simulating quantum physics. Today, new quantum algorithms like Shor's algorithm theoretically achieve exponential speed up in prime factorization and have the potential to be used in fields such as cryptography, where quantum algorithms can potentially break widely used RSA encryption schemes [13, ch. 4.1]; and in optimization, where quantum annealing can solve optimal trading trajectory problems [**?**].

Unfortunately, real-world quantum computers face significant challenges in producing consistent results due to errors and decoherence brought on by quantum noise; when cutting-edge applications may require circuits comprised of billions of gates, the error in implementing a single gate in order to perform reliable computations must be orders of magnitudes smaller than what is currently accomplishable on state-of-the-art quantum processors. In order to solve this problems, researchers have been developing and testing quantum error correction codes to increase the fault-tolerance of quantum devices, at the expense of increased time or memory requirements [7]. Efficient simulation of noisy circuits on classical computers would therefore allow those without access to quantum hardware to debug or understand the boundaries of potential quantum speeds ups of quantum algorithms [**?**].

However, the classical simulation of quantum systems remains a difficult problem due to exponential computational complexity requirements. Namely, since an n-qubit quantum system is represented by $(\mathbb{C}^2)^{\otimes n}$, the amount of memory required to naïvely store a state vector scales as $\mathcal{O}(2^n)$; moreover, to evolve the system under a unitary transformation $U \in \mathcal{H}((\mathbb{C}^2)^{\otimes n})$ requires $\mathcal{O}(2^{3n})$ time. Functionally, classical simulations have been limited to systems with fewer than approximately 50 qubits [16], where for an example, the Google Quantum AI team experimentally performed a computational task on 53 qubits with quantum hardware, which was later then estimated to take approximately 10,000 years[1] to simulate using classical simulation [15]. One solution to this computational cost was introduced by Gottesman and Knill, where he introduced the stabilizer formalism, a method to simulate a subset of quantum circuits (namely circuits composed of Clifford gates) in polynomial time. This was later improved to universality via Bravyi's method which involves including T gates to the set of Cliffords [5]. Bravyi shows that any unitary can be simulated by decomposing it into C, H, P, and T gates and T gates can be simulated within the stabilizer formalism by adding a mild exponent to the computational cost.

[1] This figure is said to be exaggerated, by [15] and others.

However, some noise operators are not unitary and as a result, simulating this quantum noise can be computationally expensive. It is at this hurdle that our research is situated: In collaboration with NASA Ames Research Center (ARC) we aim to exploit the algebraic structure of quantum noise to develop a computationally cheaper representation of noise to be deployed in classical quantum simulations.

In this report, we will first introduce the necessary background to simulating quantum circuits within the stabilizer formalism, and then we will talk about the different methods used for simulating non-unitary noise channels. The first two methods we propose are dilation methods, which allow us to lift our noise operators to a higher Hilbert space such that the lifted operator is unitary, and then we employ Bravyi's method to apply the unitaries within the stabilizer formalism . The second method we propose is an extension of Bravyi's work which generalizes his T gadget to diagonal and off diagonal operators, which allow us to implement many noise channels, as many are composed of diagonal and off-diagonal operators [6]. We end with methods that held promise but were unfinished, namely decomposing our noise operators into a sum of Cliffords, and applying each Clifford to a copy of our circuit.

Chapter 2

Background

2.1 Gottesman–Knill theorem

Theorem (Gottesman–Knill theorem)**.** *A quantum computation performed with Clifford operations and measurements of observables in the computational basis may be simulated in polynomial time on a classical computer.*

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

The Gottesman-Knill theorem tells us we can simulate circuits composed of Clifford gates in polynomial time. The Clifford gate set is a finite group defined by the generating set of C, H, and P gates (mod $U(1)$). The Clifford group is the normalizer of the Pauli group, meaning that conjugating a Pauli operator by a Clifford unitary results in another Pauli operator. Specifically for $U(n)$ being the n-qubit unitaries and $P(n)$ being the n-qubit Pauli group,

$$C(n) \equiv \{c \in U(n) | \forall p \in P(n), cpc^\dagger \in P(n)\}$$

The process that allows us to simulate these Clifford circuits in polynomial time is based on the stabilizer formalism. Note that the Gottesman-Knill theorem and the stabilizer formalism supports cheap implementation of quantum entanglement, a property that is hard for other frameworks to simulate [21].

2.1.1 Stabilizer formalism

A stabilizer state is any state $|s\rangle$ such that $|s\rangle = C|0\rangle^{\otimes n}$ for some n-qubit Clifford C. All stabilizer states are in *unique* correspondance to a stabilizing group, $\text{Stab}(|s\rangle) = \{P \in P(n) \mid P|s\rangle = |s\rangle$. The stabilizing group is a finite group of order n where $|s\rangle$ is an n-qubit stabilizer state. Notice that when we apply a Clifford operator C to a stabilizer state $|s\rangle$ with a stabilizer group $\text{Stab}(|s\rangle) = \langle g_1, g_2, \ldots g_n \rangle$:

$$C|s\rangle = Cg_i|s\rangle = Cg_iC^\dagger C|s\rangle$$

Then for the generators g_i of $\text{Stab}(|s\rangle)$, $Cg_iC^\dagger$ are the generators of the stabilizing group of $C|s\rangle$. Because these generating sets are in one to one correspondance with a stabilizer

state, we can track the generators of the stabilizing group across the circuit instead of the state vector itself. This initially seems computationally expensive as the process involves matrix multiplication, but there is a computationally efficient way to update our stabilizer group, and that is via the tableau method.

2.1.2 Stabilizer Tableau

The stabilizer tableau is a compact representation of the generating set of the stabilizer group that can be efficiently updated using bitwise operations. The matrix can be represented as a binary matrix:

$$\left[\begin{array}{c|cccc|cccc} s_1 & x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & z_{1,1} & z_{1,2} & z_{1,3} & z_{1,4} \\ s_2 & x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & z_{2,1} & z_{2,2} & z_{2,3} & z_{2,4} \\ s_3 & x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & z_{3,1} & z_{3,2} & z_{3,3} & z_{3,4} \\ s_4 & x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} & z_{4,1} & z_{4,2} & z_{4,3} & z_{4,4} \end{array}\right]$$

where $\mathbf{s}$ represents the sign(1 for minus and 0 for positive), and $\mathbf{x}$ and $\mathbf{z}$ are binary matrices representing the Pauli X and Z components, respectively. Each row of the tableau corresponds to a generator of the stabilizer group. For example, the first stabilizer is equal to

$$S_i = x_{1,1}z_{1,1} \otimes x_{1,2}z_{1,2} \otimes x_{1,3}z_{1,3} \otimes x_{1,4}z_{1,4},$$

$$x_{i,j}z_{i,j} : 00 \to I, 01 \to Z, 10 \to X, 11 \to Y$$

The *ith* column of the x matrix $\mathbf{x}_i$ and z matrix $\mathbf{z}_i$ together correspond to the *ith* qubit. Notice, however, that this does not imply that the stabilizers of the *ith* column are necessarily the stabilizers of the *ith* qubit in the system, as the stabilizer formalism supports entanglement, so the qubits may be inseparable. As an example, consider the following tableau:

$$\left[\begin{array}{c|cc|cc} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{array}\right]$$

The generating set for this stabilizing group is $\text{Stab}(|\phi^+\rangle) = \{ZZ, XX\}$. Notice the columns of the first and second qubits do not correspond to the stabilizing group of the individual qubits, as they are an entangled state $|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Recall that the set $\{C, H, P\}$ are a generating set for all Cliffords, thus when creating and applying Clifford circuits, we apply C, H, and P gates to our tableau. There are rules for how to update the tableau when applying these gates which are outlined in [2]. Their improved tableau also allows for random measurements in $O(n^2)$ time. The Stabilizer Tableau allows an efficient way to store and update the stabilizing group for any n-qubit stabilizer state when applying Cliffords to the system. Recall, however, that the Clifford gate set is not universal in that we cannot apply all unitaries via a composition of C, H, and P gates. As it turns out, including one more gate, canonically the T gate, makes this generating set universal.

2.2 Noise Channels

In this discrete setting, noise channels acting on our system are completely positive, trace preserving maps mapping density matrices to density matrices, $\phi : \rho \mapsto \rho'$. This mapping is called operator sum representation.

Definition. *Operator-sum representation. A map $\Phi : \rho \mapsto \rho'$ has a Kraus operator-sum representation (OSR) [i.e., $\Phi(\rho) = \sum_\alpha K_\alpha \rho K_\alpha^\dagger$ with $\sum_\alpha K_\alpha^\dagger K_\alpha = I$] if and only if it is linear and completely positive trace preserving (CPTP).*

It is important to note here that the Kraus operators K_i may not be Clifford. In fact, many of them are not even unitary. The noise channels we're interested in simulating are noise channels where some or all of the operators are non-Clifford, as such channels are more difficult to simulate. Consider two noise channels, in terms of their constituent Kraus operators of some operator-sum representations: the phase dampening map and the amplitude dampening map. The phase damping map is defined as $\Phi(\rho') = p\rho + (1-p)Z\rho Z$. Then the Kraus operators are $K_0 = \sqrt{p}I$ and $K_1 = \sqrt{1-p}Z$. This map can be understood as:

$$\rho \mapsto \rho' = \begin{cases} \rho & \text{with probability } p \\ Z\rho Z & \text{with probability } 1-p \end{cases}$$

The amplitude damping map is defined as $\Phi(\rho) = K_0 \rho K_0^\dagger + K_1 \rho K_1^\dagger$ where the Kraus operators are $K_0 = |0\rangle\langle 0| + \sqrt{1-\gamma}|1\rangle\langle 1|$ and $K_1 = \sqrt{\gamma}|0\rangle\langle 1|$. This map can be understood as:

$$\begin{aligned} |0\rangle &\to |0\rangle \quad \text{with probability } 1 \\ |1\rangle &\to |0\rangle \quad \text{with probability } p \end{aligned}$$

or if the density matrix is written as $\begin{pmatrix} \rho_{00} & \rho_{01} \\ \rho_{10}^* & 1-\rho_{00} \end{pmatrix}$ then ρ' is given as

$$\rho \mapsto \rho' = \begin{cases} \begin{pmatrix} \rho_{00} & \sqrt{1-\gamma}\rho_{01} \\ \sqrt{1-\gamma}\rho_{01}^* & (1-\gamma)(1-\rho_{00}) \end{pmatrix} & \text{with probability } 1-\gamma \\ \gamma(1-\rho_{00})|0\rangle\langle 0| & \text{with probability } \gamma \end{cases}$$

2.2.1 Quantum trajectories method

Obviously, working with density matrices is much more expensive than working with pure states, so we wish to avoid implementing OSR. Instead of working directly with operator sum representation, we can apply a method that allows us to work only with pure states, and still apply a full channel. Given a set of Kraus channels $\xi_i = \{K_{i,j}\}$ and a pure state $|\psi\rangle$, one can sample the output of applying the channels to a ket by sampling and applying a single Kraus to the ket per channel by the given probability distribution:

$$p_i = \langle \psi K_{i,j}^\dagger | K_{i,j} \psi \rangle$$

More details of the algorithm can be found here[**?**]. This means that the probability of choosing a given Kraus is dependent on the current state of the system. We also note that the Kraus operators need not be Clifford or unitary, meaning they do not fit within the stabilizer formalism and cannot be immediately decomposed into CHPT gates to allow a gadgetized implementation. Thus we introduce various methods for applying these non-unitary operators K to our stabilizer state $|s\rangle$.

Theorem 2.1 (Solovay-Kitaev algorithm [11])**.** *Any unitary matrix can be approximated through the Solovay-Kitaev algorithm within error ϵ, which has been optimized to use $O(\log(1/\epsilon)^{1.45})$ total gates.*

The Solovay-Kitaev algorithm provides a computational efficient way to decomposing unitary matrices into compositions of C,H,P and T gates. However, notice that the Clifford decomposition of the T gate has rank 2. Thus, in order to simulate t number of T-gates, we need 2^t tableus. [5] introduces the idea of a T-gadget, which can substitute the spot of each T-gate.

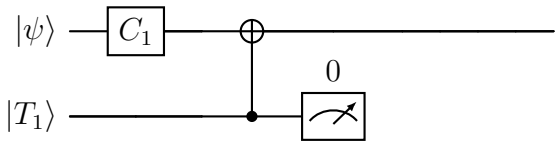


Figure 2.1: T-gadget. We perform a forced measurement on the ancillary qubit to the $|0\rangle$ state.

Rather than relying upon a Clifford decomposition of the T-gate, one can initialize an ancillary qubit in the $|T\rangle$ state and apply a series of Clifford operators and measurements on the extra qubit and current state to achieve the same result as applying a T-gate. Specifically, this $|T\rangle = \begin{pmatrix} 1 \\ e^{i\pi/4} \end{pmatrix}$. Although using T-gadgets removes the cost of applying the T-gate, there is a new cost in terms of storing these ancillary qubits held in the $|T\rangle$-state. Specifically, we must store $|T\rangle^{\otimes t}$. Notice that $|T\rangle$ actually is not a stabilizer state. Thus, the cost comes from the stabilizer decomposition of $|T\rangle^{\otimes t}$.

Notice that $|T\rangle = |0\rangle + e^{i\pi/4}|1\rangle$. Thus, an upper bound on $|T\rangle^{\otimes t} = (|0\rangle + e^{i\pi/4}|1\rangle)^{\otimes t}$ which has rank at most 2^t. However, because $|T\rangle$ is tensored with itself many times, there is reason to believe that one can compress the decomposition of the large tensor product into one with fewer terms. Bravyi and Gosset goes through some details showing that one can find a decomposition of rank $\frac{1.17^t}{\delta}$ which approximates $|H\rangle^{\otimes t}$ within an error bound of δ.

Notice that $|T\rangle = e^{i\pi/8} HP^\dagger |H\rangle$ for $|H\rangle = \begin{pmatrix} \cos(\pi/8) \\ \sin(\pi/8) \end{pmatrix}$. Thus, if one can represent $|H\rangle^{\otimes t}$ with a low rank decomposition, then they can also obtain an efficient decomposition of $|T\rangle^{\otimes t}$ by applying Clifford gates and a phase shift.

We work with $|H\rangle$ instead of $|T\rangle$ because $|H\rangle$ has nicer properties. Namely, that $|H\rangle = \frac{|0\rangle+|+\rangle}{2v}$ where $v := \cos(\pi/8)$. Thus, if we define $|\tilde{0}\rangle := |0\rangle$ and $|\tilde{1}\rangle := |+\rangle$, then we get the expression:

$$|H\rangle^{\otimes t} = (\frac{|0\rangle + |+\rangle}{2v})^{\otimes t} = \frac{1}{(2v)^t} \sum_{x \in \mathbb{F}_2^t} |\tilde{x_1}...\tilde{x_t}\rangle \tag{2.1}$$

Definition 2.1 ($Z(\mathcal{L})$)**.** *The normalization function Z maps subspaces $\mathcal{L}$ of $\mathbb{F}_2^t$ as follows:*

$$Z(\mathcal{L}) = \sum_{x \in \mathcal{L}} 2^{-|x|/2} \tag{2.2}$$

Where $|\cdot|$ is defined as the hamming weight of $\cdot$.

The approximations of $\mathcal{H}^{\otimes t}$ are based on choice of $\mathcal{L}$. For any given k-dimensional subspace $\mathcal{L}$, there is the corresponding approximation which has rank 2^k:

$$|\mathcal{L}\rangle = \frac{1}{\sqrt{2^k Z(\mathcal{L})}} \sum_{x \in \mathcal{L}} |\tilde{x_1}...\tilde{x_t}\rangle \tag{2.3}$$

The above decomposition has rank 2^k because there are a total of 2^k bitstrings in a k-dimensional subspace $\mathcal{L}$, and each bitstring corresponds to a unique stabilizer state.

Definition 2.2 (Approximation Error:)**.** *The error function δ is defined such that any given approximation $|\mathcal{L}\rangle$, the error is given by:*

$$\delta|\mathcal{L}\rangle = 1 - ||\langle H^{\otimes t}|\mathcal{L}\rangle||^2$$

Bravyi and Gosset provides several algebraic steps and shows that this error is precisely equal to [5]:

$$\delta(|\mathcal{L}\rangle) = 1 - \frac{2^k v^{2t}}{Z(\mathcal{L})} \tag{2.4}$$

Thus, to minimize the error, one must minimize $Z(\mathcal{L})$. Bravyi and Gosset's paper shows that if k is chosen to the positive number satisfying $4 \geq 2^k v^{2t}\delta \geq 2$, then one can find a subspace with error less than δ after sampling $O(\frac{1}{\delta})$ k-dimensional subspaces [5]. Moreover, the rank of this subspace will be 2^k which is approximatly $\frac{2*v^{-2t}}{\delta}$ which is in $O(\frac{1.17^t}{\delta})$.

Chapter 3

Clifford and stabilizer decompositions

In Bravyi et al.'s 2019 paper [4], the so-called sum-over-Clifford method is presented to simulate arbitrary unitary circuits by finding δ-approximate decompositions into stabilizer states, and is stated to be comparable in efficiency to gadgetization methods, but at the at the benefit of being more robust.

Proposition 3.1 (Sum-over-Cliffords method, [4, sec. 2.3.2])**.** *Given a unitary circuit U on an n-qubit system, we claim we can approximate within arbitrary error δ the output $U|0^n\rangle$ by a superposition ψ of $k \approx \mathcal{O}(\delta^{-2})$ stabilizer states:*

$$\||U|0^n\rangle - |\psi\rangle\| \leq \delta, \qquad |\psi\rangle := \sum_{\alpha=1}^{k} b_\alpha C_\alpha |0^n\rangle,$$

where C_α are Clifford operators, using some form of random sampling.

The choice of approximating with stabilizer decompositions is not an arbitrary one: the rank of a pure state's stabilizer decomposition is actually a measure of magic for pure states. In other words, stabilizer rank quantifies the difficulty[1] of simulating a circuit with our pure state as an outcome [10].

Definition 3.1 (Stabilizer rank, [6])**.** *The stabilizer rank $\chi(\psi)$ of a pure state ψ on an n-qubit system is the smallest integer for which ψ can be written as a superposition of $\chi(\psi)$ stabilizer states.*

However, as there are finitely many stabilizer states, almost every[2] state vector will attain the maximal stabilizer rank 2^n of the size of a basis for the system, which can be known via Sard's theorem. Consequently, this implies arbitrary unitary circuits are difficult to simulate. It is for this reason we introduce the concept of approximate stabilizer decompositions, which analogously quantifies the approximation of a arbitrary unitary circuit with a circuit that is easier to simulate.

Definition 3.2 (Approximate stabilizer rank, [4, def. 2])**.** *Given $\delta > 0$, the δ-approximate stabilizer rank $\chi_\delta(\psi)$ of a pure state ψ on an n-qubit system is the smallest integer for which ψ can be approximated within δ as a superposition of $\chi_\delta(\psi)$ stabilizer states.*

[1] With respect to the stabilizer formalism, meaning in terms of T-cost.

[2] I.e., with probability 1.

For the purpose of obtaining decompositions of unitary operators themselves, we introduce analogous definitions for Clifford decompositions.

Definition 3.3 (Exact and approximate Clifford rank)**.** *For an operator on a quantum system, we define the exact and approximate Clifford ranks for decompositions into linear combinations of Clifford operators similarly.*

3.1 Restatement of decomposition problem

Problem. *Suppose we are in $\mathbb{C}^n$ and have a large but finite subset of vectors $\mathcal{X} = \{x_1, \ldots, x_\alpha\}$ that spans the space. Given an arbitrary vector in $\mathbb{C}^n$, how can we find a decomposition by elements of $\mathcal{X}$ within error of at most δ that minimizes the rank, or number of terms, of the decomposition?*

In our case, our overfull spanning sets are the stabilizer states and Clifford operators for a state space and its space of operators. We were tasked with investigating Clifford decompositions of operators by NASA Ames for use in modified stabilizer simulations; such has been mentioned as an area of development for other quantum simulators [8, 14].

- Known extensively in signal processing literature as the sparsification problem [20].
- Known problem in quantum information theory in constructing certain types of codes, such as Reed-Muller codes [3].

Finding a δ-approximate decomposition into k elements of $\mathcal{X}$ is equivalent to being within δ distance of the k-dimensional subspace spanned by those elements.

Definition 3.4 (Grassmannian, $\mathrm{Gr}_k(\mathbb{C}^n)$)**.** *We define the Grassmannian $\mathrm{Gr}_k(\mathbb{C}^n)$ to be the collection of all k-dimensional subspaces of $\mathbb{C}^n$. On account of the correspondence between the k-dimensional subspaces and k-rank projection operators, we are endowed with a inner product and metric given by:*

$$\langle W, V\rangle_{\mathrm{Gr}_k(\mathbb{C}^n)} := \langle \mathrm{Proj}_W, \mathrm{Proj}_V\rangle = \mathrm{Tr}(\mathrm{Proj}_W^\dagger \mathrm{Proj}_V),$$
$$\mathrm{dist}_{\mathrm{chord}}(W, V) = \tfrac{1}{\sqrt{2}}\| \mathrm{Proj}_V - \mathrm{Proj}_W \| = \tfrac{1}{\sqrt{2}}\sqrt{\mathrm{Tr}[(\mathrm{Proj}_V - \mathrm{Proj}_W)^\dagger(\mathrm{Proj}_V - \mathrm{Proj}_W)]}$$

Grassmannians have more structure than simply an inner product, and are actually dimension $k(n-k)$ compact smooth manifolds constructed from a quotient of the unitary group $U(\mathbb{C}^n)$. This, however, is not relevant to us, as we will primarily care about only geometric considerations related to the various forms of metrics and inner products that can be induced on $\mathrm{Gr}_k(\mathbb{C}^n)$.

With a metric and inner product established on the Grassmannians relating the geometry of linear subspaces, this problem of finding such k-rank δ-approximations turns into a covering problem $\mathrm{Gr}_k(\mathbb{C}^n)$.

3.2 Computational geometry viewpoint

Covering lemmas are basic tools in computational geometry as intermediary technical result concerning pairing down a cover for a subset of a metric space to form a subcover with desirable properties.

Theorem 3.1 (3r-covering lemma)**.** *Given a cover for a subset of a sufficiently nice metric space, there exists a subcover consisting of pairwise disjoint sets that can be dilated by* 3 *to once again cover the entire subset.*

Lifting a cover of δ-balls on $\mathrm{Gr}_k(\mathbb{C}^n)$ to $\mathbb{C}^n$ via the isometry provides a covering by regions that guarantee a δ-approximate k-decomposition, hence we wish to reduce our search space of subspaces check for decompositions.

Optimal configurations for packings minimize the angles between the centers of these balls, subject to restrictions on the maximal diameter according to a pre-determined error bound.

Theorem 3.2 (Conway-Hardin-Sloan simplex bound, [18, cor. 4])**.** *Given a finite set $S \subset \mathrm{Gr}_k(\mathbb{C}^n)$, the largest inner product α between any two distinct subspaces in S satisfies*

$$\alpha \geq k\frac{k|S| - n}{n|S| - n},$$

where equality occurs if and only if $|S| = n^2$ and and S form a simplex in a hyperplane of $\mathbb{R}^{n^2-1}$.

Recalling our problem, since we are starting with a given overfull basis to pare down to a packing, finding any near-optimal configuration depends on the geometry of the spanning set.

3.2.1 Geometry of stabilizer states

The configuration of stabilizer states of an n-qubit system are fairly well understood, as per Garcìa, Markov, and Cross's 2017 paper, *On the Geometry of Stabilizer States* [9].

- Stabilizer states are distributed uniformly on the unit circle of the state space, and locally are identical in with respect to relations with nearest neighbors.
- The maximal inner product attainable by any two n-qubit stabilizer states is $1/\sqrt{2}$, or equivalently, the minimal distance is $\sqrt{2-\sqrt{2}} \approx 0.7$.
- The probability of randomly sampling nearby stabilizer states approaches zero as the number of qubits increases, which would make it difficult to naïvely construct high resolution packings.

3.2.2 Geometry of Clifford operators

It is known that the collection of Clifford operators for a quantum system form a 3-design [22].

Definition 3.5 (Unitary t-design, [22, def. 1])**.** *A finite collection of unitaries $\mathcal{S}$ of dimension d is said to be a t-design for some integer t if for all linear operators X on $\mathbb{C}^{d\otimes t}$, the following holds:*

$$\sum_{C\in\mathcal{S}} C^{\otimes t} X (C^\dagger)^{\otimes t} = \int_{U(d)} U^{\otimes t} X (U^\dagger)^{\otimes t}\, d\mu_{\mathrm{Haar}}\,.$$

A rough interpretation of this property is that the Cliffords operators are distributed very evenly, enough for 3-twirling to be equivalent to Haar-random unitary twirls.

Oszmaniec, Sawicki, and Horodecki [14] connects unitary t-designs to ε-nets, meaning subsets of unitaries that approximate every unitary operation up error ε: only universal gatesets form ε-nets for arbitrary $\varepsilon > 0$, therefore there is a limit to which we can approximate with Cliffords.

3.3 Discussion on decomposition methods

Naïvely, searching through subspaces of very high-dimensional spaces is hard: in the case of k dimensional-subspaces spanned by Clifford operators, our search space is on the order of $\binom{\mathcal{O}(2^n)}{k}$.

Greedy algorithms for searching through a collection of subspace coverings containing a vector generally exist, but likely would require looking through the literature to deal with the non-orthogonality of potential decompositions; a good place to start would be [20, sec. III.D] and move on to reviewing more current methods in signal processing for the sparse representation problem.

However, since stabilizer states and Clifford operators are distributed relatively uniformly in space, these points may already form a near-optimal packing configuration. Therefore if looking for a computational advantage by reducing our search space of subspaces, we may already be in a worse case scenario. It follows that issues regarding the lower bounds on minimal error present in stabilizer or Clifford decompositions may present issues in implementing these methods computationally.

3.4 Application: Clifford decompositions of small Kraus operators with mixed-integer linear programming

One approach to simulating a Kraus operator K_i within the stabilizer formalism is to decompose the operator into a weighted sum of Cliffords,

$$K_i \approx \sum_{i}^{\chi_\delta} c_i R_i$$

Where χ_δ is the rank required to have δ L_2 error in the decomposition. Given a circuit $|\psi\rangle \to U_1 \circ K_1 \circ U_2 \circ K_2$ and the decompositions of the Kraus operators into Cliffords, we can simulate the circuit by creating $\chi_{\delta,i}$ copies of our stabilizer tableau for Kraus operator K_i, and applying one Clifford R_i from the decomp to one tableau S_i. At the end of our circuit, we can convert each tableau into it's vector representation, scale by the stored scalar c_i, and combine it with all other tableaus to retrieve the final ket of the circuit.

To find such decompositions, we can formulate this as an optimization problem by introducing the following variables:$c_{\text{real}} \in \mathbb{R}^m$, the real part of the coefficients; $c_{\text{imag}} \in \mathbb{R}^m$, the imaginary part of the coefficients;$y \in \{0,1\}^m$, binary variables indicating whether a Clifford operator is used;$e_{\text{real}} \in \mathbb{R}^4$, the real part of the error termsl; $e_{\text{imag}} \in \mathbb{R}^4$, the imaginary part of the error terms. λ is the weight placed on minimizing the rank. The larger lambda is the more weight will be placed on minimizing rank instead of error and

vice versa. The objective is to minimize the sum of the errors and the number of vectors used:

$$\min \sum_{i=1}^{4} e_{\text{real},i} + \sum_{i=1}^{4} e_{\text{imag},i} + \lambda \sum_{j=1}^{m} y_j$$

Subject to the following constraints for the real and imaginary parts:

$$\sum_{j=1}^{\chi_\delta} (R_{\text{real},ij} c_{\text{real},j} - R_{\text{imag},ij} c_{\text{imag},j}) - x_{\text{real},i} \leq e_{\text{real},i} \quad \forall i$$

$$x_{\text{real},i} - \sum_{j=1}^{\chi_\delta} (R_{\text{real},ij} c_{\text{real},j} - R_{\text{imag},ij} c_{\text{imag},j}) \leq e_{\text{real},i} \quad \forall i$$

$$\sum_{j=1}^{\chi_\delta} (R_{\text{real},ij} c_{\text{imag},j} + R_{\text{imag},ij} c_{\text{real},j}) - x_{\text{imag},i} \leq e_{\text{imag},i} \quad \forall i$$

$$x_{\text{imag},i} - \sum_{j=1}^{\chi_\delta} (R_{\text{real},ij} c_{\text{imag},j} + R_{\text{imag},ij} c_{\text{real},j}) \leq e_{\text{imag},i} \quad \forall i$$

Linking constraints to ensure coefficients are zero if not used:

$$c_{\text{real},j} \leq M y_j \quad \forall j$$

$$c_{\text{real},j} \geq -M y_j \quad \forall j$$

$$c_{\text{imag},j} \leq M y_j \quad \forall j$$

$$c_{\text{imag},j} \geq -M y_j \quad \forall j$$

Note that our search space is the finite set of n-qubit Cliffords, which becomes intractible at $n = 3$.

$$|C_n| = \prod_{j=1}^{n} 2(4^j - 1)4^j = 2^{n^2+2n} \prod_{j=1}^{n} (4^j - 1)$$

This tells us that at higher qubit systems, finding a suitable Clifford decomposition using mixed integer linear programming becomes infeasible. Also, the memory scales as $O(\prod_i^k \xi_\delta(K_i))$ because we must create $\xi_\delta(K_i)$ copies of however many tableaus we currently have each time we wish to apply another Kraus. As an example, consider a set of two Kraus operators, each with a rank two decomposition. Then every time we apply either Kraus to our system, our system grows by a factor of two, so the number of tableaus we are storing grows to 2^k where k is the number of Kraus operators being applied. The scaling of this method is unfavorable, which led us to consider other potential solutions.

Chapter 4

Dilation Methods

A *noise channel* $\xi_i = \{K_1, ..., K_{|\xi_i|}\}$ is a collection of distinct Kraus operators that form a completely positive and trace preserving (CPTP) map [13]. Where the *channel* size $|\xi_i|$ is number of Kraus operators contained in the noise channel. Kraus operators are traditionally non-clifford and therefor can not be naively applied within stabilizer formalism.

However through the application of stabilizer tableau's [2] and T-gadgets [6], we can implement circuits of unitary operators using stabilizer formalism in a reasonable runtime. In an effort to exploit these tools Suri & Marshall proposed the use of unitary dilation's to simulate noise using tools originally designed for simulating unitary circuits [19]. Bellow we discuss how unitaries can be simulating using stabilizer formalism, before exploring the potential applications of applying noise to a quantum circuit through Stinespring and Sz.-Nagy dilation's Dilations. We note that for the remainder of this section we assume that any noise channel ξ represents local noise and thus operates on one qubit.

4.1 Simulating Unitaries

A unitary operators can be arbitrarily approximated using the CNOT, Hadamard, Phase and T gates. In other words {CNOT, H, P, T} are considered a universal gate set.

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \text{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \text{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \text{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$$

While we know that its theoretically possible to form a Clifford + T approximation of any unitary, it is useful to have an algorithm capable of finding these decompositions. The problem of decomposing arbitrary unitaries into, {CNOT,H,P,T} gates is a well studied problem with recent results indicating that the upper bound on the number of {CNOT,H,P,T} gates required to form an accurate decomposition using the Solovay-Kiteav algorihtm is $O(log(\frac{1}{\epsilon})^{1.44042...+\delta})$ [11].

As discussed in section 2 the {CNOT,H,P} are contained in the Clifford group, and can therefore be efficiently simulated using stabilizer formalism. Whereas the T gate can be efficiently simulated by adding one ancillary $|T\rangle$ magic gate for every T gate in the unitary decomposition. At this point its useful to have some measure of magic as it will determine how many $|T\rangle$ ancilia must be added to simulate a unitary.

Definition 4.1 (T-Count). *ϵ T-Count $t_\epsilon(\cdot)$ is the number of T gates needed to approximate an arbitrary unitary in terms of $C = \{CNOT, H, P, T\}$ gates such that $||K_j - \prod_{i=0}^{l} C_i|| \leq \epsilon$*

Using the improved Solovay-Kitaev algorithm [11] along with T-count, we can define a general algorithm for simulating any unitary within stabilizer formalism.

Algorithm 1 Unitary Simulation UAPPLY(U,S)

Input 1: A decomposed Unitary $U = \prod_{i=0}^{\chi} C_i$
Input 1: A set of stabilizer tableau's S
for $j = 1$ to χ **do**
    if C_j contains a T gate **then**
        Use a T-Gadget
    else
        Apply C_j using the associated tableau rule.
    end if
end for

Algorithm 2 Unitary Simulation USIM(U)

Input 1: A Unitary U
Decompose U into $\prod_{j=1}^{\chi} C_j = \hat{U}$
Compute $t = t_\epsilon(U)$
Create a circuit S with t $|T\rangle$ gates and a state $|\psi\rangle$
UAPPLY($\hat{U}$,S)

For example for the unitary $U \in \mathbb{C}^{2\times 2}$, where $U = HPT$, USIM(U) would return the following circuit

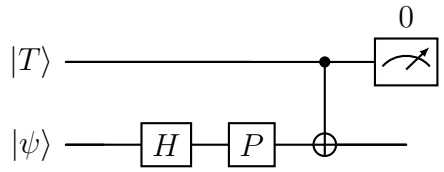


Figure 4.1: Unitary Simulation of $U = HPT$

4.2 Sz.-Nagy Dilation Approach

The first dilation approach will involve dilating each Kraus operator $K_i \in \xi$ separately. To do this we note that all Kraus operators are contractions lemma 4.1, and as a result they can all be dilated using the Sz.-Nagy unitary dilation.

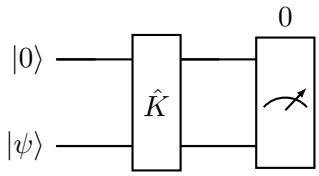
Lemma 4.1 (Kraus Contraction). *For all $K_j \in \xi_i$ K_j is a contraction operator*

Using the fact that a operator sum representation of a noise channel is trace preserving we know that $\sum_{i=0}^{k} KK^\dagger = I$. By extension we know that $\langle vK^\dagger, Kv\rangle = ||v||^2$ which implies that $||Kv||^2 \leq ||v||^2$. Therefore by definition a Kraus operator K is a contraction which proves lemma 4.1.

Theorem 4.1 (Sz.-Nagy Dilation [19]). *For every linear contraction operator A on a complex finite-dimensional Hilbert space $\mathcal{H}$, there exists a unitary dilation operator $U : \mathcal{H} \otimes \mathcal{H} \to \mathcal{H} \otimes \mathcal{H}$ in the following form:*

$$U_A = \begin{pmatrix} A & \sqrt{I - AA^\dagger} \\ \sqrt{I - A^\dagger A} & -A^\dagger \end{pmatrix}.$$

Using lemma 4.1 and theorem 4.1 we can design a quantum circuit that is equivalent to $K|\psi\rangle$, Where $\hat{K}$ is the Sz.-Nagy dilation of K.



The circuit detailed above is equivalent to first computing $\hat{K}(|0\rangle \otimes |\psi\rangle) = |0\rangle \otimes K|\psi\rangle + |1\rangle \otimes \sqrt{1 - K^\dagger K}|\psi\rangle$, and second measuring onto the 0 computational basis to retrieve component of interest, $K|\psi\rangle$. We can now define an algorithm that generalizes this process to stabilizer formalism

Algorithm 3 NAGYSINGLE(K)

1: **Input:** A Kraus operator K
2: Dilate $K \to \hat{K}$
3: Decompose $\hat{K}_{\xi_i,j}$ into $\prod_{j=1}^{\chi} C_j$
4: Compute $t = t_\epsilon(\hat{K})$
5: Create a set of tableaux that represent a circuit S with t $|T\rangle$ gates, one $|0\rangle$ ancilla, and a state $|\psi\rangle$
6: UAPPLY($\hat{K}$,S)
7: Apply a forced measurement onto the 0 computational basis to retrieve $K_1|\psi\rangle$

For example, a possible call of Algorithm 3, NAGYSINGLE(K), could generate and execute a circuit like that drawn bellow.

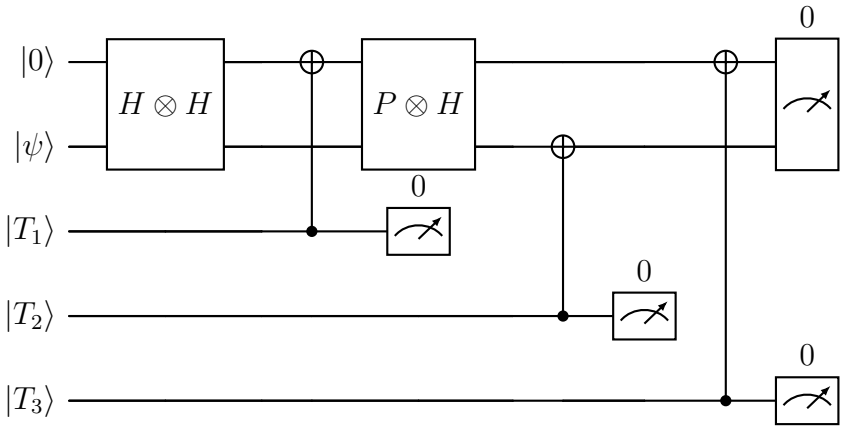


Figure 4.2: Circuit of Sz.-Nagy dilated Kraus $\hat{K} = H \otimes H \cdot T \otimes I \cdot P \otimes H \cdot I \otimes T \cdot T \otimes I$

Equipped with this intuition we can now construct an algorithm that allows us to apply multiple noise channels, instead of an individual Kraus operator. Note that the algorithm defined bellow makes use of the quantum trajectories method since it allows us to approximate the application of k noise channels without requiring the application of all k noise channels in their entirety. In fact, because the Sz.-Nagy dilation is a unitary dilation for one Kraus operator at a time, it is only natural that we make use of the quantum trajectories method which also applies one Kraus operator at a time. The quantum trajectories approach converges to the application of the all k noise channels in their entirety. [**?**].

Algorithm 4 Sz.-Nagy's Algorithm

1: **Input 1:** A list of noise channels $\Xi = \{\xi_1, \xi_2, \ldots, \xi_k\}$
2: **Input 2**: Error for the T-Gadget compression (δ)
3: **Input 3**: Error for the the {CNOT,H,P,T} decomposition (ϵ)
4: **Input 4**: Error for the approximate inner product at line 19 (η)
5: **Input 5**: Error for the the final state (Δ)
6: $t \leftarrow 0$
7: **for** $i = 1$ to k **do**
8: **for** $j = 1$ to $|\xi_i|$ **do**
9: Dilate $K_{\xi_i,j} \rightarrow \hat{K}_{\xi_i,j}$
10: Compute $t_\epsilon(\hat{K}_{\xi_i,j})$
11: **end for**
12: $t \leftarrow t + \max_{\hat{K}_{\xi_i,j} \in \xi_i}\{t_\epsilon(\hat{K}_{\xi_i,j})\}$
13: **end for**
14:
15: **for** $q = 1$ to $\lceil \frac{1}{\Delta^2} \rceil$ **do**
16: Create a circuit S with t $|T\rangle$ gates, one $|0\rangle$ ancilla, and a state $|\psi\rangle$
17: **for** $i = 1$ to k **do**
18: Pick $r \in [0, 1]$
19: **for** $j = 1$ to $|\xi_i|$ **do**
20: Copy $S \rightarrow S_1$
21: UAPPLY($\hat{K}_{\xi_i,j}$,S_1)
22: Retrieve $K_{\xi_i,j}|\psi\rangle$ and compute $p_{i,j} = \|K_{\xi_i,j}|\psi\rangle\|$
23: **if** $p_{i,j} \leq r$ **then**
24: Set $S \leftarrow S_1$
25: **break**
26: **else**
27: $r \leftarrow r - p_{i,j}$
28: Delete S_1
29: **end if**
30: **end for**
31: **end for**
32: Apply a forced measurement onto the 0 computational basis to retrieve a singular quantum trajectory $|Q_q\rangle = K_k \cdots K_2 K_1 |\psi\rangle$
33: **end for**
34: Take the average of all $\lceil \frac{1}{\Delta^2} \rceil$ $|Q_q\rangle$ to retrieve $\xi_k \circ \xi_{k-1} \circ ... \circ \xi_1(|\psi\rangle)$

Sz.-Nagy's Algorithm can be described as follows. We begin with k noise channels $\Xi = \{\xi_1, \xi_2, \ldots, \xi_k\}$ and an initial state $|\psi\rangle$. Next we find an upper bound for the total number of T-gates a single quantum trajectory may require, namely we must compute $t = \prod_{i=0}^{k} T_\epsilon(\xi_i)$, where $T_\epsilon(\xi_i) = \max_{\hat{K}_{i,j} \in \xi_i} \{t_\epsilon(\hat{K}_{i,j})\}$. Lastly we create a circuit S with t T-gadgets and one ancila, which can be represented using a set of $\frac{1.17^t}{\delta}$ tableaus $\{s_i\}_{i=0}^{\frac{1.17^t}{\delta}}$ [5]. At this point we have completed instantiating our circuit, so we can begin computing a quantum trajectory first uniformly generating a number $r \in [0, 1]$. Next we take $\xi_1 \in \Xi$ and pick the first dilated Kraus operator $\hat{K}_{1,1}$. We then make a copy S_1 of the set of tableaus S and apply the Clifford + T decomposition of $\hat{K}_{1,1}$. S_1 functionally serves as a playground to test out a potential $\hat{K}$ compute $p_{1,1} = ||\hat{K}_{1,1} |\psi\rangle||$ up to some error η using fast norm estimation [4]. If $p_{1,1}$ is $\leq r$ keep S_1 and dispose of S. If not dispose of S_1 and keep S and $r = r - p_{1,1}$. It is not guaranteed that the first Kruas operator will be choosen so we repeat the process of computing $p_{i,j}$'s until a Kraus is chosen. Only after a Kraus operator has been chosen for the first channel can we move onto the next channel to repeat the process of selecting a second Kraus operator. We continue until we have chosen a Kraus operator from each noise channel. At the end of this process we retrieve a single quantum trajectory $|Q_q\rangle = K_k \cdots K_2 K_1 |\psi\rangle$ through a forced measurement onto the 0 computational basis. However in order to achieve a final state $|\psi'\rangle$ that approximates a traditional application of the k noise channels within some error Δ, we must compute $\lceil \frac{1}{\Delta^2} \rceil$ quantum trajectories [**?**].

Theorem 4.2 (Sz.-Nagy Runtime Complexity)**.**

$$O(\frac{1}{\delta \Delta^2} s n^3 \eta^{-2} 1.17^t)$$

Where $s = \prod_{i=0}^{k} |\xi_i|$, $t = \prod_{i=0}^{k} T_\epsilon(\xi_i)$, n is the dimension of the input $|\psi\rangle$, η is the acceptable error of each inner product calculation, δ is the acceptable error of the T-gadget compression, and Δ is the acceptable error of the final solution. Where error is defined as $1 - \langle\psi|\psi'\rangle^2$

The runtime complexity is a result of computing at most s inner products per trajectory using fast norm estimation each of which grows at a rate of $O(\frac{s n^3 \eta^{-2} 1.17^t}{\delta})$. The cost of computing the Sz.-Nagy Dilation is $O(sn^3)$ which is dominated by the leading term contained in the runtime complexity. [4]

Theorem 4.3 (Sz.-Nagy Space Complexity)**.** *The space complexity of the simulating all $\lceil \frac{1}{\delta^2} \rceil$ trajectories grows as*

$$O(2 \frac{1.17^t (t+2)^2}{8\delta})$$

The space complexity is a result of storing $\frac{1.17^t}{\delta}$ tableau's each of which can be stored as a binary matrix with an associated cost of $\frac{(t+2)}{8}$ bytes. Additionally at any given point we store two copies of the set of tableau's which doubles the space required.

4.3 Stinespring's Dilation Approach

The second dilation approach involves embedding the entire noise channel ξ into a unitary operator using Stinespring's Dilation.

Theorem 4.4 (Stinespring's Dilation)**.** *Given a quantum error channel* $\xi = \{K_1, K_2, ...K_m\}$ *we can lift the channel to a unitary of dimension* $2^m x 2^m$ *by the following construction, where we fill in the remaining entries using a Gram-Schmidt process such that the* K_{ξ_1} *is unitary.*

$$K_{\xi_1} = \begin{bmatrix} K_1 & \cdots & \cdots \\ K_2 & \cdots & \vdots \\ \vdots & \ddots & \vdots \\ K_m & \cdots & \cdots \end{bmatrix}$$

Using theorem 4.4 we can design a process that is equivalent to $\xi_k \circ \xi_{k-1} \circ ... \circ \xi_1(|\psi\rangle \langle\psi|)$ but operates within the stabilizer formalism. Take for example, the application of one noise channels ξ_1 which has size $|\xi_1| = 2$ and has been dilated to K_{ξ_1}. We find that $K_{\xi_1}(|0\rangle \otimes |\psi\rangle) = \begin{pmatrix} K_{\xi_1,1} |\psi\rangle \\ K_{\xi_1,2} |\psi\rangle \end{pmatrix} = |\psi'\rangle$. We can then compute $\text{tr}(\langle\psi'|\psi'\rangle)$ which is equivalent to $\sum_i^2 K_i |\psi\rangle \langle\psi| K_i^\dagger \rho'$. We can now generalize this one noise channel example to k noise channels using the following algorithm.

Algorithm 5 Stinespring's Algorithm

Input 1: A list of noise channels $\Xi = \{\xi_1, \xi_2, \ldots, \xi_k\}$
$t \leftarrow 0$
for $i = 1$ to k **do**
    Dilate $K_{\xi_i} \rightarrow \hat{K}_{\xi_i}$
    Compute $t_\epsilon(\hat{K}_{\xi_i})$
    $t \leftarrow t + \max_{\hat{K}_{\xi_i} \in \xi_i} \{t_\epsilon(\hat{K}_{\xi_i})\}$
end for

Create a set of tableaux that represent a circuit S with t $|T\rangle$ gates, $log_2(\sum_{i=1}^2 |\xi_i|)$ $|0\rangle$ ancilla, and a state $|\psi\rangle$

for $i = 1$ to k **do**
    UAPPLY($\mathbb{I}^{\otimes i-1} \otimes K_{\xi_i}$,S)
end for
Extract the final state $|\psi'\rangle$ and compute $\text{tr}(\langle\psi'|\psi'\rangle) = \rho'$

For example the associated circuit for applying two noise channels ξ_1 and ξ_2 where $t_\epsilon(K_{\xi_1}) = 1$ and $t_\epsilon(K_{\xi_2}) = 2$, would be the following.

The first step of Stinespring's Algorithms, like Sz.-Nagy's algorithm, is to compute the number of T gates that will be used throughout the simulation. The first major deviation is that unlike Nagy's algorithm where only one ancillary qubit is required, for Stinespring's algorithm 1 ancillary qubit is added for every noise channel. The circuit S will contain $log_2(\sum_{i=1}^2 |\xi_i|)$ ancillary $|0\rangle$ qubits and t T-gadgets which can again be stored using $\frac{1.17^t}{\delta}$ stabilizer tableau's. Afterwords we simply apply each dilated noise channel, before finally taking the outer product of the final state $|\psi'\rangle$. We note that taking the outer product of the final state $|\psi'\rangle$ is a post-processing step that can only be computing outside of stabilizer formalism. However we can reduce the computation burden of this step by only summing the outer product of all $2^n \times 2^n$ blocks of the output vector.

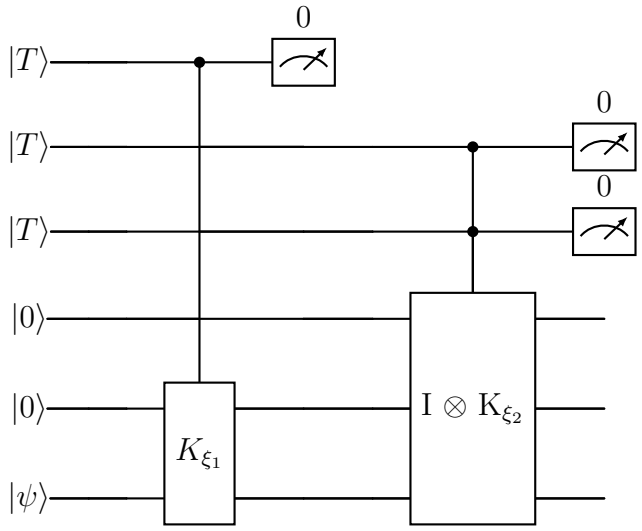


Figure 4.3: The application of two arbitrary noise channel onto a circuit using Stinespring's dilation.

Theorem 4.5 (Stinespring's Algorithm Runtime Complexity)**.** $O(n^2 \prod_{i=1}^{k} |\xi_i| + n^3 \sum_{i=1}^{k} |\xi_i|^3)$, *where n is the qubit size of the initial state $|\psi\rangle$*

The runtime complexity is a result of the $prod_{i=1}^{k}|\xi_i|$ outer products one must take each of which contributes a cost of n^2. The final outer product computation is the most expensive portion of the algorithm, however we do note that computing the stein springs dilation's requires a Gram-Schmit which contributes a cost of $n^3 \sum_{i=1}^{k} |\xi_i|^3$

Theorem 4.6 (Stinespring's Algorithm Space Complexity)**.** $O(\frac{(1+log_2(s)+t)^2 1.17^t}{8\delta})$, *where $s = \prod_{i=1}^{k} |\xi_i|$, n is the qubit size of the initial state $|\psi\rangle$, $t = \sum_{i=1}^{k} t_\epsilon(K_{x_i})$, and δ is the acceptable error of the T-gadget compression.*

The space complexity is similar to that of Sz.-Nagy's algorithm, with the only change being that each tableau has dimension $(1 + log_2(s) + t)$ instead of $(2 + t)$.

4.4 Examples

Next we investigate how many noise channels can be reasonably simulated using both dilation approaches. We take amplitude dampening channel as a case study since its one of the few frequently used non-Clifford 1-qubit noise channels.

Definition 4.2 (Amplitude Dampening Channel)**.** $\xi_{AD} = \left\{ K_1 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix}, K_2 = \begin{bmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{bmatrix} \right\}$

From the space complexity of both Nagy and Stinespring's algorithms we know that the T-count is the driving factor in memory consumption. As a result our first step is minimizing the t-count in order to best estimate how many ξ_{AD} channels a user could apply. To do this we first note that K_2 is a projector so an equivalent process to applying K_2 is first applying an X gate then applying a forced projective measurement of 0. In other words $K|\psi\rangle = P_{|0\rangle} X |\psi\rangle$ is a T gate free operation. Therefore we can focus our efforts on quantifying the T-Count of K_1. The Sz.-Nagy dilated form of K_1 can be recast as the following circuit.

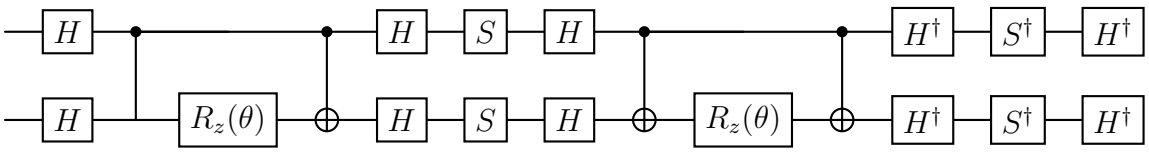


Figure 4.5: Stinespring Dilated Amplitude Dampening Circuit Equivalence

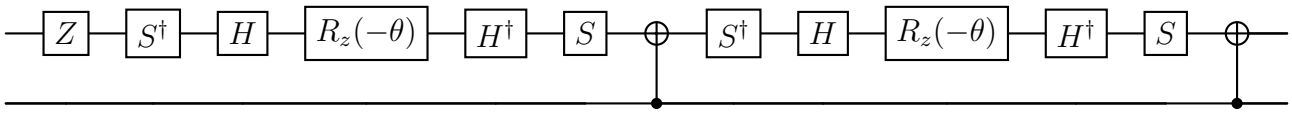


Figure 4.4: Sz.-Nagy Dilated Amplitude Dampening Circuit Equivalence

The Stinespring dilation of ξ_{AD} can also be recast as the following circuit through a few manipulations. Both circuit derivations are detailed in Appendix A.

We can observe that most of our T-Cost comes from the 2 R_z gates in both the Sz.-Nagy and Stinespring dilated unitaries. Luckily we know that the approximate t-cost is a function of how precise we want our p to be. For example a $p = .01$ would require 3 T gates per R_z gate, while a $p = .1$ would require 2 T gates per R_z gate [12]. Using these T count estimates for both the Sz.-Nagy and Stinespring algorithms we find that a user could reasonably apply **10-15** noise channels on a 16GB of memory.

Chapter 5

K-gadgets and Compression

The dilation methods are versatile in that they can simulate all combinations of Kraus operators. However, some quantum systems may experience the same type of noise repeatedly. In systems like these, rather than converting Kraus operators into unitary matrices and decomposing them into sequences of C,H,P and T gates, perhaps one can apply a K-gadget similar to what was done by Bravyi and Gosset [5]. The previous Chapter noted how the T-cost of even simple amplitude dampening channels could reach up to four per Kraus operator. The main benefit of using a K-gadget method is that instead of requiring four T-gadgets, we would only need a single K-gadget per Kraus operator.

Lemma 5.1. *Operating under the same gadget construction as the T-gadget, one can compress diagonal matrices* $K_1 = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$ *and off-diagonal matrices* $K_2 = \begin{pmatrix} 0 & c \\ d & 0 \end{pmatrix}$ *with corresponding magic states* $m_1 = \begin{pmatrix} a \\ b \end{pmatrix}$ *and* $m_2 = \begin{pmatrix} c \\ d \end{pmatrix}$. *The corresponding gadgets are noted in the appendix, and are easy to verify.*

Moreover, both of these magic states can be written as a linear combination of two stabilizer states as the stabilizer states form a basis over $\mathbb{C}^2$ and $\mathbb{C}^2$ is dimension 2 over $\mathbb{C}$. Thus, any Kraus operator which is either diagonal or off-diagonal has a corresponding magic state $|M\rangle = \frac{|\tilde{0}\rangle + |\tilde{1}\rangle}{c}$ where $|\tilde{0}\rangle = |f\rangle$ and $|\tilde{1}\rangle = \alpha\,|g\rangle$, for any linearly independent pair of stabilizers $|f\rangle$ and $|g\rangle$ and some normalization constant c. For example, the Kraus channel $\begin{pmatrix} 1 & 0 \\ 0 & \sqrt{3}2 \end{pmatrix}$, corresponding to the amplitude dampening channel with parameter $p = 0.5$, has the corresponding magic state $|M\rangle = \begin{pmatrix} 1 \\ \frac{\sqrt{3}}{2} \end{pmatrix}$. This magic state can be written as a linear combination of $|0\rangle$ and $|+\rangle$ where $|M\rangle = (1 - \frac{\sqrt{3}}{2})\,|0\rangle + (\frac{\sqrt{6}}{2})\,|+\rangle = \frac{|+\rangle + \frac{2-\sqrt{3}}{\sqrt{6}}|0\rangle}{\frac{2\sqrt{6}}{6}}$, where $\alpha = \frac{(2-\sqrt{3})}{\sqrt{6}} \approx 0.1$. Notice that this parameterization of $|M\rangle$ in terms of α and two stabilizer states is really just a generalization of the $|H\rangle$-state, with the $|H\rangle$-state being the result for $\alpha = 1$ and two specific stabilizer states. Like Bravyi and Gosset's method, we aim to find an efficient decomposition of:

$$|\mathcal{L}\rangle^{\otimes t} = \frac{1}{\sqrt{K(\mathcal{L})}} \sum_{x \in \mathcal{L}} |\tilde{x}_1 ... \tilde{x}_t\rangle = \frac{1}{\sqrt{K(\mathcal{L})}} \sum_{x \in \mathcal{L}} \alpha^{|x|}\,|x_1 x_2 ... x_t\rangle$$

Where $K(\mathcal{L}) = \sum_{x,y\in\mathcal{L}} \nu^{|x+y|}\alpha^{|x|+|y|}$, with $\nu = \langle f|g\rangle$. We similarly define an approximation corresponding to a subspace $\mathcal{L}$ with:

$$|\mathcal{L}\rangle = \frac{1}{\sqrt{K(\mathcal{L})}} \sum_{x\in\mathcal{L}} |\tilde{x}_1 ... \tilde{x}_t\rangle = \frac{1}{\sqrt{K(\mathcal{L})}} \sum_{x\in\mathcal{L}} \alpha^{|x|} |x_1 x_2 ... x_t\rangle$$

We use the same error function, and we can see that:

$$\langle M^{\otimes t}|\mathcal{L}\rangle = \frac{1}{\sqrt{K(\mathbb{F}_2^1)^t K(\mathcal{L})}} \sum_{x\in\mathbb{F}_2^t, y\in\mathcal{L}} \alpha^{|x|+|y|}\nu^{|x+y|} \tag{5.1}$$

Rather than summing over all $x \in \mathbb{F}_2^t$, we can sum over all possible values of $|x+y|$, ranging from 0 to t. For any given y, the number of terms where $|x+y| = i$ is equal to $\sum_{j=0}^{i} \binom{|y|}{j}\binom{t-|y|}{j-i}$ where j is the number of one-bits that are flipped. Thus, $\binom{|y|}{j}\binom{t-|y|}{j-i}$ counts the number of x terms which are j 1-bit-flips and $i-j$ 0-bit-flips away from y. Thus, using some combinatorics, we can simplify Equation 5.1 to the following expression:

$$\langle M^{\otimes t}|\mathcal{L}\rangle = \frac{1}{\sqrt{K(\mathbb{F}_2^1)^t K(\mathcal{L})}} \sum_{y\in\mathcal{L}} \sum_{i=0}^{t} \alpha^{2|y|}\nu^i \sum_{j=0}^{i} \frac{1}{\alpha^{2j-i}} \binom{|y|}{j}\binom{t-|y|}{j-i} \tag{5.2}$$

The reason why we care about this error function is because it is significantly easier to calculate, as rather than iterating over all bitstrings in $\mathbb{F}_2^t$, one only needs to iterate over the terms in $\mathcal{L}$, which is a subspace significantly smaller than $\mathbb{F}_2^t$. Using this inner product formula, we ran several simulations to test how much we could compress $|M\rangle^{\otimes t}$, as well has what the error would be relative to each compression rate.

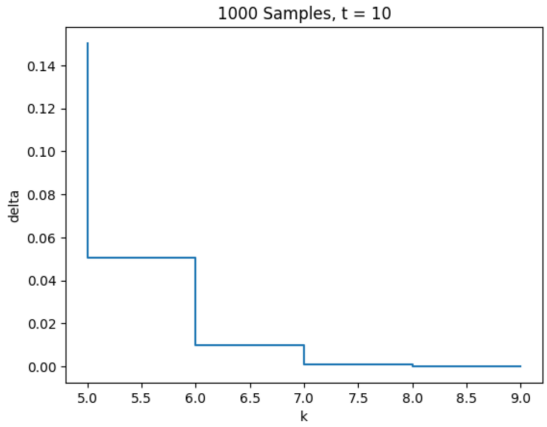


Figure 5.1: Caption: Minimum error across 1000 samples of subspaces across different dimensions. $|M\rangle$ has $\alpha = 0.5$.

In figure 5.1 we sampled 1000 subspaces of $\mathbb{F}_2^t$ with various for each dimension ranging from 5 through 9 and we plotted the minimum estimation error across the 1000 samples for each dimension. Notice using a 7-dimensional subspace, one can expect to achieve error below 0.01 across 1000 samples using a dimension seven subspace. For comparison, the method used in Bravyi and Gosset's paper only suggests a dimension nine decomposition.

The figure indicates that such magic states can be indeed be compressed, and that the true compression rate is lower than what is indicated in [5] for small t values.

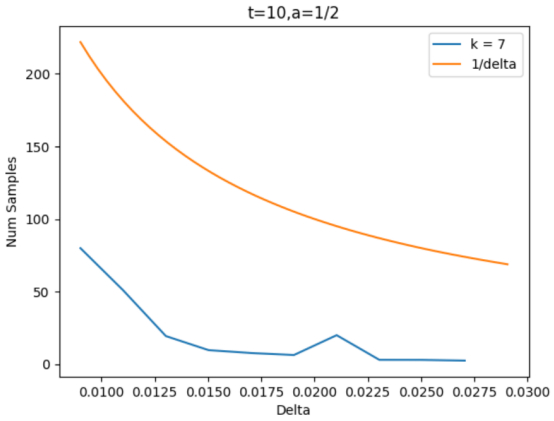


Figure 5.2: Number of samples required to achieve subspace with specific error values

Moreover, the number of samples actually required to find a seven dimensional subspace with corresponding approximation error less than 0.01 is actually quite small as well. Figure 5.2 shows the number of samples required to achieve a dimension 7 subspace with error less than a specific delta when approximating $|M\rangle^{\otimes t}$. This is shown in comparison to the expected number of samples needed to approximate $|H\rangle^{\otimes t}$ using the T-state compression method. Figure 5.2 show that the expected number of samples to achieve any given error is actually fewer in the K-gadget case when $\alpha = 0.5$ than the T-gadget case where α is effectively equal to 1.

Figures 5.1 and 5.2 indicate that magic states with $\alpha = 0.5$ can also be compressed more efficiently than the bounds found in [5], and that sampling a sufficiently accurate approximation would require fewer samples than expected in the $\alpha = 1$ case.

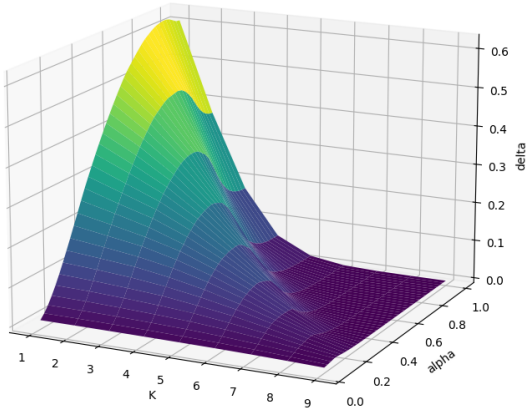


Figure 5.3: Minimum error across 250 samples for various α values and subspace dimensions.

Figure 5.3 shows the minimum error in approximating $|M\rangle^{\otimes 10}$ across 250 samples, while varying the dimensions sampled and the α values that make up $|M\rangle$. Indeed, the error monotonically decreases as the dimension of the approximation increases. However, the error does not monotonically increase as α increases.

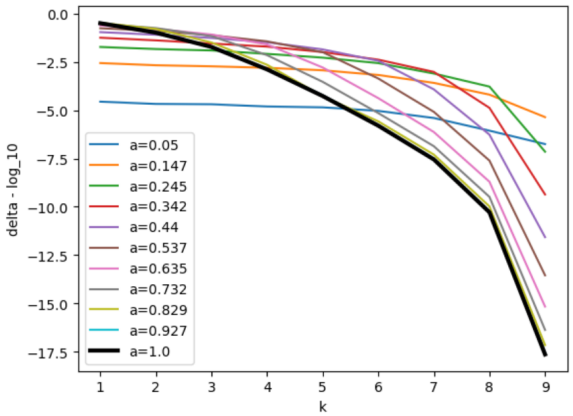


Figure 5.4: Approximations error across different k and α-values.

Figure 5.4 is the result of figure 5.3 when viewed from a particular angle. The figure shows thats that the gradient of the error with respect to the dimension differs for different values of α. In fact, the magnitude of the gradient increases as α increases, indicating that larger values for α benefit more from having higher rank than smaller values of α. The plot also indicates that magic states written with higher values of α can achieve overall lower levels of error than magic states with lower values of α.

The research on K-gadgets is still ongoing and there are many open questions that we have yet to answer. Our goal with K-gadgets is to find a similar bound to the bound provided in the T-gadget compression method, which would inform what subspaces to look over for lower rank compositions, as well as an the order of the expecterd number of samples required to find a subspace with sufficiently low error. We are also interested in how changing the two stabilizer states which compose $|M\rangle$ would affect the compression rate or the sampling process.

Chapter 6

Conclusion and Discussion

We end with a brief comparison and discussion of the four methods proposed throughout this report. Note that the Clifford decomposition approach is intentionally omitted since the $O(2^k)$ space complexity allowed us to rule out this approach early on in our research. the complexity analysis of the K-gadget approach is a rough estimate based on the numerical simulations listed in section 5.

Name	**Runtime Complexity**	**Space Complexity**	**# of Noise Channels**
Sz.-Nagy	$O(\frac{1}{\delta\Delta^2} s n^3 \eta^{-2} 1.17^t)$	$O(2^{\frac{1.17^t (t+2)^2}{8\delta}})$	10-15
Stinespring	$O(n^2 \prod_{i=1}^{k} \lvert\xi_i\rvert + n^3 \sum_{i=1}^{k} \lvert\xi_i\rvert^3)$	$O(\frac{(1+log_2(s)+t)^2 1.17^t}{8\delta})$	10-15
K-Gadget	$O(\lvert\Xi\rvert \cdot \lvert\xi\rvert)$	$O(\frac{(k+1)^2(1+0.2\alpha)^k}{8\delta})$	57-11,312

Table 6.1: Comparison of Algorithms. Number of noise channels is computed given a 16GB limit in memory. For Sz.-Nagy and Stinespring the range is a result of varying precision of the noise and for the K-gadget approach the range is a result of varying α from 1 to 0

Simulating Quantum circuits is a computationally difficult problem, and the stabilizer formalism provides one of many possible routes to classical simulation. The stabilizer formalism is quite restrictive, only allowing a small set of unitary operators to be simulated efficiently. Furthermore, simulating non-unitary noise channels within the stabilizer formalism is a difficult problem as it does not natively support them. We have provided two main methods for implementing them.

The two dilation methods allow us to lift our Kraus operator or noise channel to a higher Hilbert space such that the lifted operator is unitary, and apply that unitary matrix to our stabilizer formalism using Clifford + T decomposition and applying Bravyi and Gosset's work. The Stinespring's approach allows us to simulate the whole channel with the high upfront computational cost of using the Gram Schmidt process to create our unitary matrices. It remains an open question whether there are more computationally friendly approaches to this process. The post-processing cost of converting the output to OSR is also a hefty computation, requiring $\prod_{i=1}^{k} \lvert\xi_i\rvert$ outer products. Comparatively, the Sz.- Nagy approach incurs more cost during the runtime of the circuit itself, requiring the use of quantum trajectories. Namely, the requirement of applying the Kraus operator K to the current state to find it's respective probability is the largest contributer to the algorithm's runtime

It should be noted that the K-gadgets method allows us to simulate diagonal and off-diagonal matrices with reasonable runtime. This means that we are not restricted to operators that satisfy the noise channel requirement $\sum KK^\dagger = I$, but this method can be applied to any diagonal and off-diagonal matrix. As is explained in Appendix A, this can also work

for arbitrary 2×2 operators, but with an exponential cost of 2^k where is the number of operators applied. It remains an open question what other methods can be employed to more efficiently simulate these arbitrary operators.

Appendix A

Circuit Equivalences

A.1 Circuit Derivations For Section 3.4

First and foremost we'd like to thank Namit Anand for proposing the application of Matchgates to solve this problem for Sz.-Nagy dilation and for solving this problem for the Stinespring dilation case.

A.1.1 Amplitude Dampening Circuit for Sz.Nagy Dilation

The goal will be to get approximate minimal T-Counts of the amplitude dampening channel using Sz. Nagy Dilation and Matchgates. Where the amplitude dampening channel is defined as follows

$$\xi_{AD} = \left\{ K_1 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix}, K_2 = \begin{bmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{bmatrix} \right\}$$

We will focus our attention on K_1, whose Sz.-Nagy dilation is the following, when $p = \sin^2(\frac{\theta}{2})$

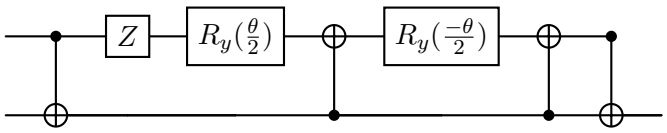
$$\hat{K}_\theta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\frac{\theta}{2}) & 0 & \sin(\frac{\theta}{2}) \\ 0 & 0 & -1 & 0 \\ 0 & \sin(\frac{\theta}{2}) & 0 & -\cos(\frac{\theta}{2}) \end{bmatrix}$$

We can show that CNOT $\hat{K}_\theta$ CNOT recovers a Givens rotation

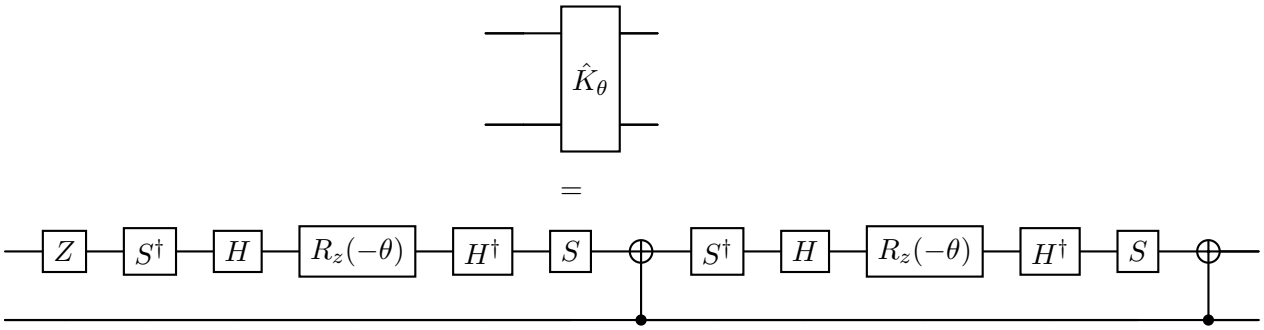
$$\text{CNOT}\,\hat{K}_\theta\,\text{CNOT} = G_{AB}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2}) & 0 \\ 0 & \sin(\frac{\theta}{2}) & -\cos(\frac{\theta}{2}) & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

In other words these two circuits are identical

Ramelow established some useful circuit identities for Givens rotations which we use to rewrite G_{AB} as the following circuit [17]



Using the identity that $R_y(\theta)$ can be rewritten as $R_y(\theta) = S^\dagger H Rz(-\theta) H^\dagger S$, we can again rewrite the following circuit equality



A.1.2 Amplitude Dampening Circuit for Stinespring's Dilation

The Stinespring dilation for ξ_{AD} gives us the following unitary matrix if we parameterize $p = \sin^2(\theta/2)$

$$G(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta/2) & -\sin(\theta/2) & 0 \\ 0 & \sin(\theta/2) & \cos(\theta/2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This is known as a Givens rotation. It is easy to check that:

$$G(\theta) = \exp\left[-i\theta/2(Y \otimes X - X \otimes Y)\right]$$

To find an optimal Clifford+T decomposition of this gate, we use the following observations. First, recall that the iSWAP gate is a Clifford unitary defined as:

$$\text{iSWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can parameterize this gate as the following equality and can recover the above gate for $\theta = \pi$.

$$\text{iSWAP}(\theta) = \exp\left[i\theta/2(X \otimes X + Y \otimes Y)\right]$$

The first claim is that:

$$G(\theta) = (T^{-1} \otimes T) \cdot \text{iSWAP}(\theta) \cdot (T \otimes T^{-1})$$

This tells us how to generate the Givens rotation using T-gates and iSWAP gates. Now, using the fact that $[X \otimes X, Y \otimes Y] = 0$, we can simplify the iSWAP gate as:

$$\text{iSWAP}(\theta) = \exp\left[i\theta/2(X \otimes X)\right] \cdot \exp\left[i\theta/2(Y \otimes Y)\right]$$

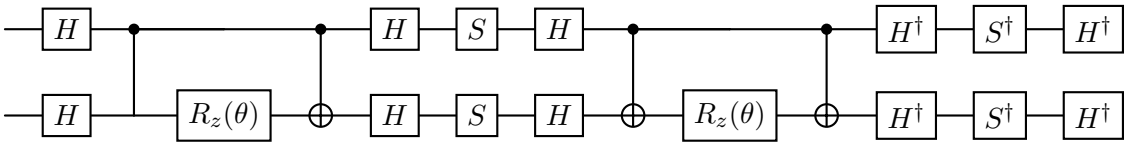
Moreover, using the following two identities:

$$HZH^\dagger = X \quad \text{and} \quad (HSH)Z(HSH)^\dagger = -Y$$

$$\exp\left[i\theta/2(Z \otimes Z)\right] = \text{CNOT} \cdot (I \otimes R_z(\theta)) \cdot \text{CNOT}$$

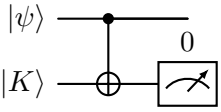
we can write the iSWAP gate as the following circuit

$$\text{iSWAP}(\theta) = H^{\otimes 2} \cdot \text{CNOT} \cdot (I \otimes R_z(\theta)) \cdot \text{CNOT} \cdot H^{\dagger \otimes 2} \cdot (HSH)^{\otimes 2} \cdot \text{CNOT} \cdot (I \otimes R_z(\theta)) \cdot \text{CNOT} \cdot (HSH)^{\dagger \otimes 2}$$

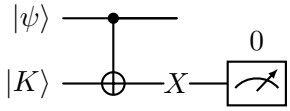


A.2 K-gadgets

Utilizing the fact that we can force a measurement in the Z basis to be $|0\rangle$, we create a gadget that can apply a Kraus operator to a state $|\psi\rangle$. Let $K = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$. Then the gadgetized K is $|K\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$. Then to apply the diagonal Kraus operator, we apply the following circuit:



Which produces $K|\psi\rangle$. A similar circuit works for the diagonal matrix $K = \begin{bmatrix} 0 & a \\ b & 0 \end{bmatrix}$. Let $|K\rangle = \begin{bmatrix} b \\ a \end{bmatrix}$. Then



gives you $K|\psi\rangle$.

Note that you can also simulate any Kraus operator $K = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ by decomposing it into the sum of the diagonal components and the off-diagonal components. However, it has scaling similar to Clifford decomp of rank 2, that is, it scales as 2^k where k is the number of applied Kraus operators.

Bibliography

[1] *40 years of quantum computing*, Nature Reviews Physics, 4 (2022), p. 1.

[2] S. Aaronson and D. Gottesman, *Improved simulation of stabilizer circuits*, Physical Review A, 70 (2004).

[3] A. Ashikhmin and A. R. Calderbank, *Grassmannian Packings From Operator Reed–Muller Codes*, IEEE Transactions on Information Theory, 56 (2010), pp. 5689–5714.

[4] S. Bravyi, D. Browne, P. Calpin, E. Campbell, D. Gosset, and M. Howard, *Simulation of quantum circuits by low-rank stabilizer decompositions*, Quantum, 3 (2019).

[5] S. Bravyi and D. Gosset, *Improved Classical Simulation of Quantum Circuits Dominated by Clifford Gates*, Physical Review Letters, 116 (2016).

[6] S. Bravyi, G. Smith, and J. A. Smolin, *Trading Classical and Quantum Computational Resources*, Physical Review X, 6 (2016).

[7] G. et al., *Suppressing quantum errors by scaling a surface code logical qubit*, Nature, 614 (2023).

[8] H. J. Garcia and I. L. Markov, *Simulation of Quantum Circuits via Stabilizer Frames*, IEEE Transactions on Computers, 64 (2015), pp. 2323–2336.

[9] H. J. García, I. L. Markov, and A. W. Cross, *On the Geometry of Stabilizer States*, Nov. 2017. arXiv:1711.07848 [quant-ph].

[10] M. Howard and E. Campbell, *Application of a Resource Theory for Magic States to Fault-Tolerant Quantum Computing*, Physical Review Letters, 118 (2017), p. 090501.

[11] G. Kuperberg, *Breaking the cubic barrier in the Solovay-Kitaev algorithm*, 2023. Version Number: 1.

[12] G. J. Mooney, C. D. Hill, and L. C. L. Hollenberg, *Cost-optimal single-qubit gate synthesis in the clifford hierarchy*, Quantum, 5 (2021), p. 396.

[13] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, 10th anniversary ed., June 2012.

[14] M. Oszmaniec, A. Sawicki, and M. Horodecki, *Epsilon-nets, unitary designs and random quantum circuits*, 2020. Version Number: 3.

[15] F. Pan and P. Zhang, *Simulating the Sycamore quantum supremacy circuits*, Mar. 2021. arXiv:2103.03074 [physics, physics:quant-ph].

[16] E. Pednault, J. A. Gunnels, G. Nannicini, L. Haoresh, T. Magerlein, E. Solomonik, E. W. Draeger, E. T. Holland, and R. Wisnieff, *Breaking the 49-Qubit Barrier in the Simulation of Quantum Circuits*, Tech. Rep. LLNL-JRNL-747743, Lawrence Livermore National Laboratory (LLNL), Livermore, CA (United States), Feb. 2018.

[17] S. Ramelow, A. Fedrizzi, A. M. Steinberg, and A. G. White, *Matchgate quantum computing and non-local process analysis*, New J. Phys., 12 (2010), p. 083027.

[18] A. Roy, *Bounds for codes and designs in complex subspaces*, Journal of Algebraic Combinatorics, 31 (2010), pp. 1–32.

[19] N. Suri, J. Barreto, S. Hadfield, N. Wiebe, F. Wudarski, and J. Marshall, *Two-Unitary Decomposition Algorithm and Open Quantum System Simulation*, Quantum, 7 (2023).

[20] J. Tropp, *Greed is Good: Algorithmic Results for Sparse Approximation*, IEEE Transactions on Information Theory, 50 (2004), pp. 2231–2242.

[21] G. Vidal, *Efficient Classical Simulation of Slightly Entangled Quantum Computations*, Physical Review Letters, 91 (2003).

[22] Z. Webb, *The Clifford group forms a unitary 3-design*, 2015. Publisher: arXiv Version Number: 3.

Quantum Informatic Model of Protist Decision Making

Jad Soucar[1] **and Francis F. Steen**[2]

[1]USC Viterbi Daniel J. Epstein Department of Industrial Systems Engineering 3715 McClintock Avenue, GER 240 , Los Angeles, CA 90089-0193 , Email: soucar@usc.edu

[2]Department of Communication, 345 Portola Plaza, Los Angeles, CA 90095, USA , Email: steen@comm.ucla.edu

I. Introduction

The process through which multi-celled organisms make decisions is immeasurably complex and the set of possible actions the organism can take is infinite. This level of complexity makes modeling behavior incredibly difficult, which leaves researchers capable of only developing behavioral models that serve as heuristics for what an organism may do. So with this in mind we aim to answer the question; Is there an organism that still exhibits signs of complex decision making, but whose decision making process can be accurately modeled? To answer this question we move away from multi-celled organism and examine a single celled cilliate called Stentor Roeselli.

In 1906, Biologist Herbert Spencer Jennings reported that Stentor Roeselli exhibited signs of complex behavior, potentially indicating a form of decision making [6]. Jennings' work seemed to indicate that the protist was capable of choosing from a set of possible actions and that the choice was dependent on past experiences. Later in 2019 Dexter et al. [2] verified Jennings' experiment by carefully documenting the behavior of Stentor Roeselli within a dataset that is now publicly available and concluded that the protist was exhibiting some sort of structured decision making.

Using the Jenning's and Dexter's findings, researchers have sought to develop learning models capable of replicating the behavior of the protist. For example in 1969 Wood verified that the sort of primitive learning like that exhibited by Stentor Roeslii was possible [18]. Additionally researchers have verified that the behavior of single celled organisms, like Stentor Roeslli partially satisfy tenants of complex decision making models such as associative learning frameworks, although such frameworks have been shown to be incomplete descriptions of the protist's behavior [5], [14]. Other approaches in literature have focused on behavioral models that grade each decision Stentor Roseli can make based on its respective cost and utility. For example Staddon grades potential decisions of the Stentor Roeselii via the the energy required along with the opportunity cost associated with carrying out the decision [15].

While some have focused on qualitative behavioral theories, other have applied quantitative machine learning models like decision trees, random forrests and artificial feed forward neural networks to the problem. However each of these models were ineffective in fully capturing the behavior measured in Dexter's experiments, with the best modeling producing an accuracy of roughly 59% [17]. As a result Trinh et al. conclude that Stentor Roseli's decision making process "cannot be fully explained by habituation, sensitization or operand behaviour" [17]. They go onto claim that while, machine learning models and other established learning models are able to capture an impressive amount of behavioral features of more complex life forms [12], they have been largely unsuccessful in modeling single celled decision making.

So in an effort to develop a faithful model and introduce a new tool to the body of literature on this problem, we will motivate then propose a novel quantum informatics framework for modeling the protist's behavior that aims to computationally replicate Dexter's statistical findings. To do this we will first analyze Stentor's Roseli's behavior, next we will motive the application of quantum information theory, before finally proposing a quantum behavioral model along with implementation techniques from the field of classical quantum simulation.

II. Behavior of Stentor Roeseli

In this section we will describe the behavior of Stentor Roeselii along with the tools it has at its disposal to carry out those decision. First we note that the Stentor Roeselii contains is relatively large (up to 1200 micrometers), trumpet shaped, and covered with rows of cilia that help it move and feed by sweeping food into its cystostome [13].

Now that we've briefly covered the taxonomy of the Stentor Roeselli we move onto the behavioral observations made by Jennings and later verified or amended by Dexter [6], [2]. Jennings reported a hierarchy of behaviors that the protist exhibited when exposed to an irritant or toxin. We list the possible actions in the same order Jennings reported they would occur; resting (R), bending away (B), ciliary alteration (A), contraction (C), and detachment from the surface (D). They conclude that their is strong statistical evidence that the Stentor Roeselli follows a behavioral hierarchy. However they note that the hierarchy was rarely observed in full and that they instead observed many partial instances of the hierarchy with cases of the protist skipping steps in the hierarchy. The one exception to the skipping phenomenon is that detachment (D) is always the last action exhibited. We note that (A/B) are low energy actions, while (C) and (D) require more energy to

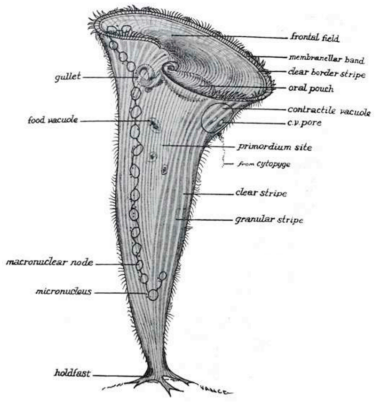


Fig. 1. Drawing of S. coeruleus showing principal features [16], largely shared with S. roeseli

enact. This suggests that the hierarchy observed is in line with optimal foraging theory, which posits that organisms evolve to minimize energy expenditure while maximizing the inflow of energy [11].

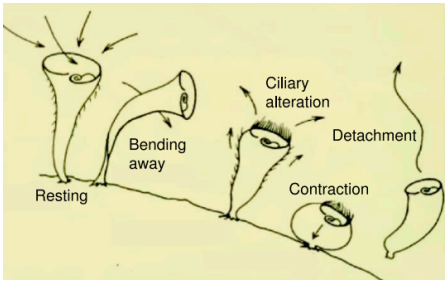


Fig. 2. Drawing of the observed behaviors of the Stentor Roeselii. [6]

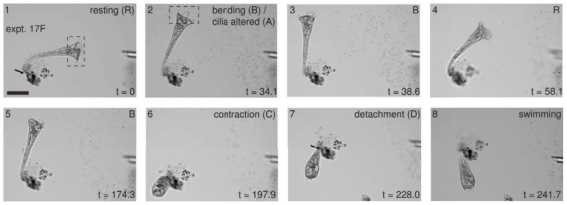


Fig. 3. An observed sequence of behaviors [2]

A. Statistical Analysis

As previously stated Dexter's experimental results are publicly available, so we will spend some mining their dataset for insights that can help inform the structure of a model of Sentor Roeselli's behavior. Dexter's dataset is formatted as a set of sequences formed by A,B,C,D and are separated by pluses (p) which are instances where the researchers manually injected a toxin into the environment. For example a sequence could look like "ABpACCpCCCD" or "CpCCD". Given the structure of the data we define inter-sequence and intra-sequence Where inter-sequence trends deal with the protist's decision making up until the toxin is eliminated then reintroduced using a pulse. Intra-sequence trends deal with how decision making changes across multiple interact

First we cover the inter-sequence trends. Its simple to observe in Dexter's data that (A/B) never follows (C) except for two samples which Dexter attributes to accidental pulses. In other words once a series of contraction begins the protist will only return to a rest state after a detachment or a successful contraction that eliminates the toxin. Dexter observed that that when the protist contracts (C) there is a 50% chance that the contraction will be followed by a detachment (D), and that the probability of remaining attached after k consecutive contraction is $\frac{1}{2^k}$. This indicates that the decision to detach (D) post contraction was tantamount to a coin flip. While it might be tempting to claim that the spontaneity of detachment is evidence against learning we find that the 50% probability of detaching post contraction can be attributed the 50% probability that (C) is successful in eliminating the toxin. Specifically, through an analysis of Dexter's dataset we note that the (A/B) had a success rate of approximately 10%, (C) was successful around 50% of the time and (D) had a 100% success rate. These results indicate that first the cheaper (A/B) actions were less effective in eliminating the toxin from the protist's environment and second that the contraction is successful at the same rate as a coin flip.

Next we move onto intra-sequence trends. Dexter's experiments indicate that that (A) and (B) often occurred together and as result can be treated as a single action (A or/and B). Using this notational adjustment we note that the first action taken by a protist with no "experience" of interacting with toxins is (A/B).Additionaly dexter notes that all 44 instances of (D) were directly preceded by (C) and that in 30/44 instances (D) was preceded by (A or/and B). This means that in roughly 30% of cases the (A/B) action was skipped entirely. We note that this corresponds with the 30% conditional probability that (A/B) is reapplied given that (A/B) failed to remove the toxin in the protist's most recent interaction with the toxin. This seems to indicate that protist "learned" that the (A or/and B) response was not sufficient to eliminate the toxin and as a result will opt for the more energy intensive option of contracting (C) immediately after the second encounter with the toxin [7].

III. Quantum Influence

Based on the statistical analysis described above both Dexter and Jennings conclude that the Stentor Roselii possesses the capacity for hierarchical decision making. In 2009 Bray argues in his book titled "Wetware: A Computer in Every Living Cell" that this sort of hierarchical decision could be explained by biochemical pathways that act like logic circuits or neural networks [1].

However the failure of previous machine learning based models [17] to capture the behavior of Stentor Roselii implies that the protist's behavior cannot be fully explained through simple digital circuits. In fact the statistical analysis described above indicates that the protist's behavior is laden with certain amount of noise and uncertainty. Take for example the decision

to detach (D) after a contraction (C). This choice follows can be modeled as a memory less random variable, in other words the protists flips a coin repeatedly until a head appears and only then does it detach.

Bray even concedes that the key difference between electronic and biological circuits is that cellular circuitry is noisy and its outcomes can be difficult to predict. However its these very features that support the claim that biological systems are more inline with quantum logic then digital logic. Since the outputs of quantum circuits suffer from noise and are at their core probabilistic. In fact the claim that quantum mechanical phenomena play a nontrivial role in biology has been verified within several biological mechanisms such as light harvesting in photosynthesis, vision, olfactory sensing, and magnetroception [4]. So we claim that the behavior of stentor roselii is best described not by a neural networks or other deterministic and digital imitations of biological wetware, but instead through quantum wetware. Before we begin designing a quantum simulator for S.Roselii we will briefly introduce quantum computing.

IV. Quantum Computing Background

In quantum computing, the fundamental unit of information is the qubit, which is a two-level quantum system. A qubit can exist in a superposition of the basis states $|0\rangle$ and $|1\rangle$, and its general state is described by a linear combination:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where α and β are complex numbers known as probability amplitudes. The squared magnitudes of these amplitudes represent the probabilities of measuring the qubit in the respective states:

$$P(0) = |\alpha|^2, \quad P(1) = |\beta|^2$$

Since these probabilities must sum to 1, the norm of the state vector must also be 1. This leads to the normalization condition:

$$|\alpha|^2 + |\beta|^2 = 1$$

Before a moving on from 1-qubit systems we note that an important state we will make frequent use of. The $|+\rangle$ state represents a superposition of $|0\rangle$ and $|1\rangle$ where the probability of measuring either basis is equal.

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\end{pmatrix}$$

For a system of n qubits, the state space is the tensor product of n single-qubit state spaces. This allows us to represent multi-qubit states in the form of tensor products of individual qubits. Speficially the state of an n-qubit system is an element of the 2^n-dimensional Hilbert space, $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$. If we have a set of single-qubit states $|\psi_1\rangle, |\psi_2\rangle, \ldots, |\psi_n\rangle$, the overall state of the n-qubit system is given by the tensor product:

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$$

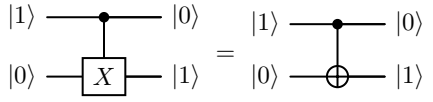
Quantum evolution is governed by unitary operations, which are linear transformations that preserve the norm of quantum states. A matrix U is unitary if it satisfies:

$$U^\dagger U = UU^\dagger = I$$

where $U^\dagger$ is the conjugate transpose of U and I is the identity matrix. Unitaries ensure that the evolution of the qubit preserves the normalization condition, which maintains valid probability distributions for measurement outcomes. In fact the group unitary matrices form a complete gate set, which measn that there always exists a unitary that can map one arbitrary qubit to another [8]. Specifically we can construct a unitary $U_{|\hat{\psi}\rangle} : |\psi\rangle \to \left|\hat{\psi}\right\rangle$ as follows.

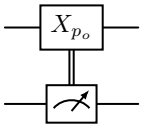
$$U_{|\hat{\psi}\rangle}(|\psi\rangle) = |\psi\rangle\langle\hat{\psi}| + |\psi^\perp\rangle\langle\hat{\psi}^\perp|$$

These unitary transformations are fundamental in quantum algorithms, as they allow controlled and reversible evolution of quantum states, without violating the probabilistic interpretation of quantum mechanics. Quantum circuits provide a graphical method of describing quantum algorithms. Each qubit is represented by a horizontal line, and operations on qubits are represented by gates applied along this line. Each quantum gate corresponds to a unitary operation, and the sequence of gates in a circuit defines the overall unitary transformation applied to the system. For example the controlled X gate, often called the CNOT gate, introduces entanglement between two qubits. If the control qubit is $|1\rangle$, the target qubit is flipped. The circuit below shows the application of a CNOT gate to two qubits:



This concept of controlled unitaries can be expanded from the CNOT gate to arbitrary unitaries

In a quantum circuit we can also classically control qubits. For example if $|0\rangle$ is measured we can classically apply the X gate with a set probability p_o by classically sampling from a uniform distribution. We will represent classical control using the double wire.



Next we describe the measurement process. The measurement process is the process of extracting classical information from a quantum circuit. We do this by projecting the state down to an element in the computational basis with a probability corresponding to the square of that element's amplitude. This process is known as the Born rule [8]. For example the standard computational basis of a two qubit state would be the outcomes $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$, where the square root of the corresponding probabilities are exactly the first, second,

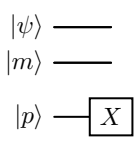
third and fourth entries respectively in the 2-qubit state vector $|\psi\rangle \in \mathbb{R}^4$. A measurment is typically described graphically using the following icon.

V. Proposed Quantum Behavioral Model

In this section we describe a iterative quantum circuit that captures the Stentor Roselii's behavior. First we introduce a 2 qubit state vector $|\psi\rangle$ where the measurement outcome $|11\rangle$ is the the (A/B) action, $|10\rangle$ is contraction (C), $|01\rangle$ is detachment (D), $|00\rangle$ is the resting state $|00\rangle$. We could represent $|\psi\rangle$ as the tensor product of two qubit $|\psi_1\rangle \otimes |\psi_2\rangle$ or as 4×1 vector where r, c, ab, d correspond with the square root of the probability that the protist will choose to rest (R), contract (C), deploy the (A/B) action, or detach (D) repsectivly.

$$|\psi\rangle = \begin{pmatrix} r \\ d \\ c \\ ab \end{pmatrix} = |\psi_1\rangle \otimes |\psi_2\rangle$$

Second we introduce a toxin qubit $|p\rangle$ which is $|1\rangle$ when a toxin is present and $|0\rangle$ when there is no toxin detected. We note that $|p\rangle$ is functionally a digital bits in that it will only ever take on a zero or one value. We also introduce a memory qubit $|m\rangle$ which allows the protist to store its previous action. Specifically if $|m\rangle = |0\rangle$ the previous action was (A/B), if $|m\rangle = |1\rangle$ the previous action was (C) or (D) and if there was no previous action then $|m\rangle = |+\rangle$ since neither (A/B) or (C/D) has taken place. So the full state of the protist, which contains the internal state qubits and environmental data qubits can be written as $|\psi\rangle \otimes |m\rangle \otimes |p\rangle$. For example assuming the $|p\rangle$ is initially set to $|0\rangle$ (no toxin) we can describe a protist who has just detected a toxin using the following circuit.



where X is the Pauli X gate, which flips the qubit its applied to and in this case "turns on" the toxin bit [8].

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1}$$

Next we describe the mechanism through which the protist adjusts the probability of an action based on the state of $|p\rangle$. In quantum circuits the process of increasing or decreasing the probabilities of certain outcomes can be replicated through amplitude dampening, a rotation based mechanism which decreases the probability of a certain outcome using quantum gates. We will use amplitude dampening to adjust the inter-sequence and intra-sequence statistics of the protist. For example given that the $|\psi\rangle = \big(0, 1, 0, 0\big)$, which would indicate that the protist will definitively choose (A/B) as its next action, one could decrease the probability of choosing (A/B) after a failed application using a Givens Rotation $G_{1,2}(\theta)$. θ would represent the inter-sequence learning rate, or how quickly the protist changes its preference between (A/B) and (C).

$$G_{AB,C}(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & -\sin\theta & \cos\theta \end{pmatrix} \tag{2}$$

We also define a controlled G gate which outlines the various adjustments a protist can make. First if the previous action was an unsuccessful (A/B) then the protist decreases the probability of choosing (A/B) again and increase the probability of choosing (C). On the other hand if the last action was an unsuccessful (C) then the protist maps $|\psi\rangle$ to $|t\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}$ for two reasons. First after the protist chooses a contraction the probability of returning to (A/B) before the end of the sequence drops to 0 and second it represents the 50/50 chance, which Dexter observed, that (C) is reapplied or the protist detaches (D).

$$G(|m\rangle, |p\rangle) = \begin{cases} G_{AB,C}(\theta), & \text{if } |m\rangle = |0\rangle \text{ and } |p\rangle = |1\rangle \\ U_{|t\rangle}(|\psi\rangle), & \text{if } |m\rangle = |1\rangle \text{ and } |p\rangle = |1\rangle \\ U_{|00\rangle}(|\psi\rangle), & \text{if } |p\rangle = |0\rangle \end{cases}$$

Using these tools we can begin to piece together a a quantum circuit that reflects the behavior of the Stentor Roselii. We begin by setting the initial state of the system to $|00 + 0\rangle$, which represents a resting state with no toxins detected and no recorded last action. In order to simulate the presence of a toxin we apply $X \otimes X \otimes I \otimes X$ which flips the toxin bit, guarantees a first response of (A/B), and yields the following state $|11 + 1\rangle$. We set the probability of (A/B) to 1 since this is the default starting action for a protist as observed in Dexter's experimental data. Next we measure the the first two qubits of the circuit $|\psi\rangle$ and find that the $|\psi\rangle$ has collapsed to $|11\rangle$ and the protist has chosen (A/B). Based on that response we record the last action in $|m\rangle$ using the following gate.

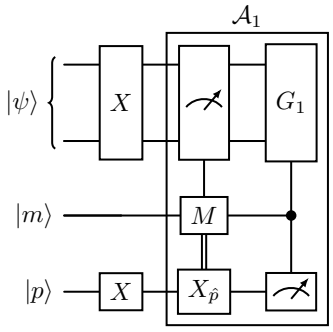
$$M = \begin{cases} U_{|+\rangle}(|m\rangle), & \text{if the measurement outcome is } |00\rangle \\ U_{|0\rangle}(|m\rangle), & \text{if the measurement outcome is } |11\rangle \\ U_{|1\rangle}(|m\rangle), & \text{if the measurement outcome is } |10\rangle \text{ or } |01\rangle \end{cases}$$

In this case the measurement outcome is $|11\rangle$, which corresponds to the (A/B) action and will classically trigger a $X_{\hat{p}}$ gate applied to $|p\rangle$ at a probability $\hat{p}$. Where $\hat{p}$ in this case is 0.10 as observed in Dexter's dataset, but in general can be determined by the following function. Notice that the probability of switching off the toxin bit is 0 when the protist is at rest.

$$\hat{p} = \begin{cases} 0, & \text{if the measurement outcome is } |00\rangle \\ 0.10, & \text{if the measurement outcome is } |11\rangle \\ 0.50, & \text{if the measurement outcome is } |10\rangle \text{ or } |01\rangle \end{cases}$$

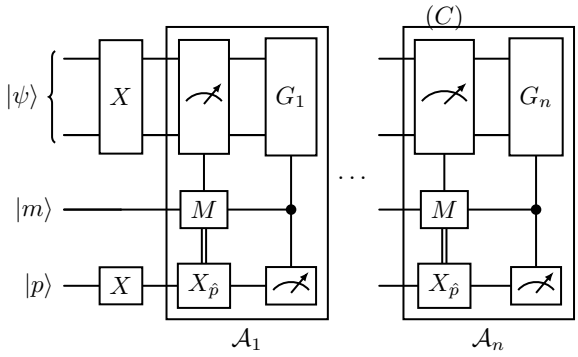
Next we measure the toxin qubit $|p\rangle$ to determine whether or not the action was successful. If the action was successful

and $|p\rangle = |0\rangle$ then we apply $G_1 \leftarrow U_{|00\rangle}(|\psi\rangle)$ which returns the protist to a resting state. This process can be represented using the following quantum circuit.



Now assume that we measure $|p\rangle = |1\rangle$, then we conclude that the previous action was unsuccessful in eliminating the toxin. So the protist measures $|m\rangle$ and determines that the last action was (A/B) which prompts it to decrease the probability of choosing (A/B) using $G_1 \leftarrow G_{AB,C}(\theta)$ with some inter-sequence learning rate θ. The state is then measured again to determine the next action. This process continues until the toxin is removed, which would require the measurement of $|p\rangle$ to return $|0\rangle$, or the measurement of $|\psi\rangle$ to return $|10\rangle$ which corresponds with the (C) action. Assuming we measure $|\psi\rangle$ to be $|10\rangle$ (C) on the n^{th} iteration we classically trigger an $X_{\hat{p}}$ gate applied to $|p\rangle$ at a probability of $\hat{p} = 0.50$. If the toxin remains then we apply $G_n \leftarrow U_{|t\rangle}(|\psi\rangle)$ which sets the probability of choosing (C) to the statistically predicted rate of $\frac{1}{2}$. We repeat this process and continue to to apply $U_{|t\rangle}(|\psi\rangle)$ which decreases the probability of (C) to $\frac{1}{\sqrt{2}^k}$ after k failed (C) applications until detachment (D) is triggered which flips $|p\rangle$. In other words the toxin is eliminated and the sequence terminates.

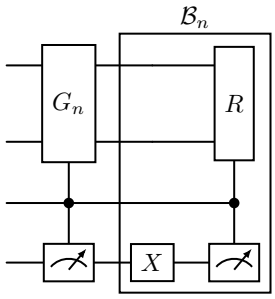
We can describe this process using the following circuit where a measurement with the superscript (C) represents the first measurement where (C) is chosen.



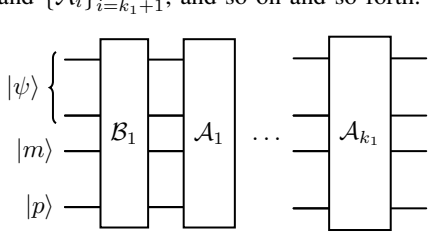
While the sequence itself has terminated the protist will use the sequence to inform future decisions. Recall the statistical analysis which showed that in 70% of cases where (D) was not preceded by (A or/and B) the protist had already tried and failed to apply the (A or/and B) response to an earlier interaction with the toxin. This indicated that upon a failed application of (A/B), the probability that the protist chooses (A/B) as a first response in the next interaction with a toxin decreases to 30% while the probability of (C) increases to 70%. Therefor once the protist terminates the sequence and begins an interaction with another toxin we allow the protist to access it's memory and adjust its bias towards one decision or another. For example if the sequence terminates with the action some action in $\{AB, C, D\}$ with the corresponding state $|m\rangle$ we adjust $|\psi\rangle$ using the following function R where $|c\rangle = \begin{pmatrix}\sqrt{0.3} & \sqrt{.7}\end{pmatrix}$ and $|ab\rangle = \begin{pmatrix}\sqrt{0.7} & \sqrt{.3}\end{pmatrix}$.

$$R(|m\rangle) = \begin{cases} U_{|c\rangle}|\psi\rangle, & \text{if } |m\rangle = |1\rangle \text{ and } |p\rangle = |0\rangle \\ U_{|ab\rangle}|\psi\rangle, & \text{if } |m\rangle = |0\rangle \text{ and } |p\rangle = |0\rangle \\ X \otimes X, & \text{if } |m\rangle = |+\rangle \text{ and } |p\rangle = |0\rangle \\ I \otimes I, & \text{if } |p\rangle = |1\rangle \end{cases}$$

For example assume that $|p\rangle$ is measured to be $|0\rangle$ on the n^{th} iteration. In that case $G_n \leftarrow U_{|00\rangle}(|\psi\rangle)$ which sends the protist to a resting state. Once the toxin is detected again, which would entail manutally applying an X gate to $|p\rangle$ the protist adjusts it's decision bias by applying R to $|\psi\rangle$ before proceeding with the rest of the sequence.



Notice that a full sequence, from the detection of a toxin to termination, can now be described compactly as the following circuit where $\{k_i\}$ is the set of iterations at which a sequence terminates. We note that an application of $\mathcal{B}$ is a user-level decision, and that this is very similar to Dexter's experimental setup where the researchers deliver toxins into an environment manually through the introduction of microbeads [2]. We also note that once the first sequence is over we can apply $\mathcal{B}_{k+1}$ and $\{\mathcal{A}_i\}_{i=k_1+1}^{k_2}$, and so on and so forth.



A. Depth Analysis

A natural question to ask at this point is what the expected length of each sequence given an initial state. First we let the random variable N be the number of iterations until termination. Second we note that the length of each sequence depends on the evolution of the probabilities of the (A/B) and (C) actions. So we let the state at the beginning of the sequence be $|\psi_0\rangle \in \mathbb{R}^2$, where ψ_1, ψ_2 are the probabilities of the (A/B) and (C) action respectively. We solve this problem

for an arbitrary $|\psi_0\rangle$ and find $\mathbb{E}[N||\psi_0\rangle]$. We note that this expectation can be broken up into an expectation of N_{AB} which is the number of iterations before (C) is first chosen or an (A/B) action results in removal of the toxins, and N_C which is the number of iterations before (D) is first chosen or (C) results in removal of the toxins.

$$\mathbb{E}[N||\psi_0\rangle] = \mathbb{E}[N_{AB}||\psi_0\rangle] + \mathbb{E}[N_C||\psi_0\rangle]$$

We observe that $\mathbb{E}[N_C||\psi_0\rangle] = 2$ since its geometrically distributed with probability $1/2$ and we proceed to the calculation of $\mathbb{E}[N_{AB}||\psi_0\rangle]$. For this computation we introduce the $AB_n \in \{0, 1\}$ random variable where n is the number of iterations that (A/B) action was consecutively selected, and a value of 0 represents an instance where (A/B) was not selected. We also note the following probability measure for (A/B), where ρ is the learning rate induced by $G_{A,B}(\theta)$.

$$\mathbb{P}(AB_n = 1) = \max(\psi_1 - \rho n, 0)$$
$$\mathbb{P}(AB \text{ Succeeds}) = .10$$

Using these two measures we conclude that the probability that the number of consecutive (A/B) actions is k is $\mathbb{P}(k) = \prod_{i=1}^{k} \frac{9}{10} * \max(\psi_1 - \rho n, 0)$.

$$\mathbb{E}[N_{AB}||\psi_0\rangle] = \sum_{i=0}^{\infty} i\mathbb{P}(i) = \sum_{i=0}^{\lfloor \frac{a}{\rho} \rfloor} i(\psi_1 - \rho n)$$

.

Take $\rho = 0.03$, which means that inter-sequence learning is slow, and let $\psi_1 = 1$. We find that the $\mathbb{E}[N_{AB}||\psi_0\rangle] = 2.6$, which is inline with the empirical expected depth of a sequence given that the protist has no experience with the toxin and as a result defaults to a first choice of (A/B). This would imply that $\theta = .4949$ We conclude this section by reiterating that the expected depth calculus can also serve as a powerful tool to calibrate the inter-sequence learning rate θ.

B. The Measurement Problem

We note that measuring the $|\psi\rangle$ and $|p\rangle$ qubits is commonly used in the quantum circuit described above. However unlike traditional measurements which collapse the qubits to a certain state, the measurement we apply is only meant to retrieve the probabilities associated with an action or the presence of a toxin. In fact the algorithm described above only operates as intended if the measurement does not de-cohere the state. This feature of the algorithm initially seems to contradicts the standard "Copenhagen" interpretation of quantum mechanics which does not allow for measurement without collapsing the quantum state and loosing all probabilistic information stored within the state. Despite the seeming contradiction the process of non-collapsing measurements is not unusual for biological systems and has been observed in the FMO complex, a light-harvesting protein found in green sulfur bacteria where it plays a critical role in directing excitation energy from antenna complexes to reaction centers.

Studies by Graham Fleming and others have shown that this energy transfer process involves quantum superposition, allowing the excitation to explore multiple pathways simultaneously [3]. During a fast excitation event, superposition states are formed, enabling the system to reversibly sample relaxation rates from all component exciton states. This process efficiently directs energy transfer toward the lowest energy sink, essentially allowing the system to select the optimal pathway, akin to performing a quantum computation that senses many states in parallel and identifies the correct solution, as indicated by the high efficiency of energy transfer. This behavior can be linked to the non-collapsing nature of measurement, where the wavefunction does not collapse during measurement but instead continues to evolve coherently, similar to the superposition states in the FMO complex. The fact that the FMO system maintains superposition to optimize energy transfer seems to indicate that biological systems can preserve the coherence of a quantum state, allowing measurements to reveal information about particle positions without disrupting the overall wavefunction. This possible explanation suggests that biological systems like Stentor Roseli can reversibly sample the quantum state and extract the necessary information to make a decision without collapsing the state.

Bellow we offer two explanation for how coherent measurement can unfold in biological and computational systems. Both explanations indicate that our model for explaining the behavior of Stentor Roselii is feasible despite the incorperation of coherent measurment.

1) Bohmian Feasibility: Bohmian mechanics, also known as the pilot-wave theory, offers a deterministic interpretation of quantum mechanics where particles have well-defined trajectories, guided by a pilot wave [10]. In this interpretation, the state of a quantum system is described by both a wavefunction $\psi(\mathbf{r}, t)$, which evolves according to the Schrödinger equation,

$$i\hbar\frac{\partial\psi(\mathbf{r},t)}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\psi(\mathbf{r},t) + V(\mathbf{r})\psi(\mathbf{r},t), \tag{3}$$

and a set of particle positions $\mathbf{r}(t)$, which evolve deterministically according to the guiding equation [10]:

$$\frac{d\mathbf{r}(t)}{dt} = \frac{\hbar}{m}\mathrm{Im}\left(\frac{\nabla\psi(\mathbf{r},t)}{\psi(\mathbf{r},t)}\right). \tag{4}$$

Unlike the Copenhagen interpretation, Bohmian mechanics does not require the wavefunction to collapse during measurements. Instead, measurement is viewed as an interaction that reveals the position of the particles while the wavefunction continues its unitary evolution. This non-collapsing feature is particularly promising for quantum computing, where maintaining coherent superposition is crucial. In a quantum computer based on Bohmian mechanics, measurements could be performed by interacting with the system's particle configuration, without disturbing the global wavefunction. This could preserve the coherence of qubit states and enable subsequent quantum operations, thereby maintaining the integrity of quantum information processing throughout the computation.

2) Standard Feasibility: While Bohemian mechanics offers a different mathematical formalism and ontological explanation for the process of extracting probabilistic information from states without decoherence, Quantum tomography offers

another solution that is inline with the standard Copenhagen interpretation [9].

Quantum tomography is a technique used to reconstruct the state of a quantum system by extracting the probabilities of various outcomes through a series of measurements. Unlike traditional measurements that can collapse or decohere the quantum state, quantum tomography is carefully designed to infer the underlying probabilities without disturbing the state irreversibly. This method reconstructs the density matrix, ρ, which encodes the probabilities of all possible measurement outcomes, including classical probabilities and quantum coherences, which represent superpositions and entanglements. Instead of directly measuring and collapsing the state, tomography relies on a series of weak, indirect, or varied measurements over many identical copies of the state. By statistically sampling from multiple preparations, the technique gathers information without fully projecting the state, thus maintaining coherence. Measurements in quantum tomography can include projective measurements applied to large ensembles and weak measurements that minimally disturb the state [9].

VI. Conclusion

In this paper, we have explored the complexity of decision making in the single-celled organism Stentor roeselii. Through an analysis of the data provided by Dexter, we confirmed that the protist's behavioral responses, though simple in nature, follow statistical trends that imply a form of primitive learning or adaptation. While previous attempts to model this behavior using classical machine learning frameworks, such as decision trees and neural networks, have failed to capture the probabilistic aspects of the organism's actions, we proposed a novel quantum behavioral model that aligns with these complexities.

The quantum framework we introduced successfully simulates the organism's decision-making process by incorporating both noise and probabilistic features observed in Stentor roeselii's behavior. By utilizing quantum information theory, particularly quantum circuits with amplitude dampening, memory effects, and coherent measurment our model captures the stochastic nature of the protist's decision-making hierarchy. This allows for the representation of inter- and intra-sequence learning that was absent in earlier models or hidden behind a black box.

Our findings highlight the potential of quantum simulation techniques in biology, particularly in understanding behaviors that are inherently noisy and probabilistic, such as those of Stentor roeselii. The proposed model not only offers an accurate computational tool for replicating Dexter's experimental results but also opens new avenues for modeling decision-making processes in biological systems where classical approaches may fail. Future work can extend this model to incorporate more complex biological systems, offering deeper insights into the intersection of quantum mechanics and biological behavior.

VII. Open Questions

What happens when we increase the memory of the protist? Will the protist begin to learn more complex inter-sequence patterns? For example it may learn that the "AC" pattern produces the highest efficacy. Furthermore if the protist has a larger memory how will the intra-sequence behavior change? For example if the protist uses the "CCC" pattern before eliminating the toxin will that result in a ≤ 0.70 probability that (C) is chosen again due to some implied inefficiency. This also begs the question; how much memory can we reasonably endow the protist with? For the sake of the analysis contained in this paper the protist has a memory of one time step, but one can reasonably argue that a larger memory capacity is possible.

VIII. Acknowledgements

I would like to thank Dr.Francis Steen for his invaluable mentorship not only throughout this project, but during my 4 years at the University of California, Los Angeles. I would also like to thank Google's Summer of Code program for sponsoring this work.

References

[1] Dennis Bray. *Wetware: a computer in every living cell*. Yale University Press, 2009.

[2] Joseph P Dexter, Sudhakaran Prabakaran, and Jeremy Gunawardena. A complex hierarchy of avoidance behaviors in a single-cell eukaryote. *Current biology*, 29(24):4323–4329, 2019.

[3] Gregory S. Engel, Tessa R. Calhoun, Elizabeth L. Read, Tae-Kyu Ahn, Tomáš Mančal, Yuan-Chung Cheng, Robert E. Blankenship, and Graham R. Fleming. Evidence for wavelike energy transfer through quantum coherence in photosynthetic systems. *Nature*, 446(7137):782–786, April 2007.

[4] Graham R. Fleming, Gregory D. Scholes, and Yuan-Chung Cheng. Quantum effects in biology. *Procedia Chemistry*, 3(1):38–57, 2011. 22nd Solvay Conference on Chemistry.

[5] Todd M Hennessey, William B Rucker, and Colin G McDiarmid. Classical conditioning in paramecia. *Animal learning & behavior*, 7(4):417–423, 1979.

[6] H. S. Jennings. *11. Introduction and Behavior of Cœlenterata*, pages 188–232. Columbia University Press, New York Chichester, West Sussex, 1931.

[7] Wallace F. Marshall. Cellular cognition: Sequential logic in a giant protist. *Current Biology*, 29(24):R1303–R1305, December 2019.

[8] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 10th anniversary edition, June 2012.

[9] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 10th anniversary edition, June 2012.

[10] Xavier Oriols and Jordi Mompart. Overview of bohmian mechanics, 2012.

[11] Graham Pyke. *Optimal Foraging Theory: An Introduction*. 01 2019.

[12] Francesco Savelli and James J Knierim. Ai mimics brain codes for navigation, 2018.

[13] S. Serrano, L. Arregui, B. Perez-Uz, P. Calvo, and A. Guinea. Guidelines for the identification of ciliates in wastewater treatment plants. *Water Intelligence Online*, 7(0):9781780401935–9781780401935, December 2015.

[14] Tomohiro Shirakawa, Yukio-Pegio Gunji, and Yoshihiro Miyake. An associative learning experiment using the plasmodium of physarum polycephalum. *Nano communication networks*, 2(2-3):99–105, 2011.

[15] John Eric Rayner Staddon. *Adaptive behavior and learning*. Cambridge University Press, 2016.

[16] Vance Tartar. *The biology of stentor: International series of monographs on pure and applied biology: Zoology*. Elsevier, 2013.

[17] Mi Kieu Trinh, Matthew T. Wayland, and Sudhakaran Prabakaran. Behavioural analysis of single-cell aneural ciliate, stentor roeseli, using machine learning approaches. *Journal of The Royal Society Interface*, 16(161):20190410, December 2019.

[18] David C Wood. Parametric studies of the response decrement produced by mechanical stimuli in the protozoan, stentor coeruleus. *Journal of neurobiology*, 1(3):345–360, 1969.

A Model for Trust Driven Advertising

Jad Soucar[1], Francis F. Steene[1]
[1] *Department of Mathematics, 520 Portola Plaza*
Los Angeles, CA 90095 USA
[2]*Department of Communication, 345 Portola Plaza*
Los Angeles, CA 90095 USA
jadsoucar@g.ucla.edu, steen@comm.ucla.edu

Keywords: Cognitive dynamics; cognitive microeconomics; dynamical systems; computational modeling; dynamic systems modeling; mathematical modeling; value residual; communicative potential; expressive cost; receptive cost; trust.

Abstract: Cognitive processes underlie economic relations. In this paper, we develop a conceptual, mathematical, and computational framework for modeling market exchange as a series of dynamically interacting cognitive processes. Specifically, we show how advertisers can build trust and gain confidence in their pricing power to the point that they erode trust and undermine the efficacy of their advertising. Customers conversely orient towards advertisers seeking information or turn away from them as unreliable communicators. These behaviors and the patterns they generate occur inside a state space of unallocated perceived value. They constitute a small subset of the full range of possible strategic and adaptive responses that define cognitive microeconomics.

1 Introduction

The principle that market exchange creates value is one of the foundations of economics. Evidence of trade in ochre crayons goes back around 300,000 years, to the beginning of our species (Brooks et al., 2018). The basis for a market exchange is that two parties assign asymmetrical value to a good (Smith, 1776). Yet the value assigned to the good by the seller relative to the cost is not necessarily a simple inverse of the value assigned to the good by the buyer relative to the cost; the asymmetry is often much more favorable to both parties.

As a first approximation, we can use the labor theory of value (Ricardo, 1821) to define the state space of win-win solutions. Consider a skilled flint knapper that makes five serviceable hand axes in a day, while it takes his hunter neighbor a day of work to hunt an animal, skin it, and prepare the pelt; each has a competitive advantage (Ricardo, 1821). Say the hunter would have to spend two days to make a single axe and the knapper two days to acquire a single pelt, creating a large unallocated value residual within the price equilibrium. For instance, if they agree to exchange a pelt for five axes, the hunter gets a value residual of five days' labor and the knapper a value residual of one day's labor. How does the market allocate this surplus value? All points within this space are in principle acceptable, as they would result in a net benefit to both parties.

Consider a product with twenty dimensions of value, from packaging to color, shelf-life to hipness. Winterfeldt and Fischer describe that for each of these dimensions, we can attach a cost of materials and labor; the sum of these represents the production cost to the manufacturer (Von Winterfeldt and Fischer, 1975). Similarly, for the customer to reproduce each of these qualities would require some cost, in many cases far exceeding production costs. The difference between these two arrays represents at once a value residual and a field of win-win solutions for trade. In an industrial society, most of the surplus value is created by machinery fueled by external energy; human labor represents only a fraction of the cost. This leaves plenty of surplus value to be divided among producers and consumers.

We propose that this value residual is fundamentally unallocated by the market. No laws of economics determine how the value residual, or surplus value (Marx, 2020), is divided between the two parties in a market exchange, or between producers and customers. The value residual could be magnanimously given away or taxed to finance a state; in the following, we aim to show how it can give rise to a complex dynamic of value and trust.

In summary, we propose that market exchange

relations are characterized by the creation of multidimensional state spaces of prospective value in which actors navigate based on imperfect information. Value spaces are viewpointed and need to partially overlap to enable cooperative markets. Surplus value is an expected feature of such transactions and its allocation is not determined by market forces alone. Instead, the value differential constitutes an open space where the participants navigate in an adaptive decision process we may term "wayfinding" (McCubbins and Turner, 2020).

2 Communicative Potential

The residual value represented by the difference between actual production costs by the manufacturer and the hypothetical production costs by the customer creates a communicative potential, expressed through the medium of advertising. On the one hand, prospective buyers need information about the products and services available in the market to decide whether to make a purchase. On the other hand, sellers have an incentive to provide relevant information about their product or service to persuade customers to purchase it.

In this communicative act, which in the case of advertising is typically a one-way street from advertisers to prospective customers, we propose that the two parties are competing to capture a share of the residual value. In efficient markets, the market price of a good will tend to move towards the production cost, in effect allocating the lion's share of the residual value to the customer. In the present model, advertising is a tool the producer uses to contest and attempt to recapture some of this value.

We propose the communicative potential as a holistic concept that incorporates the expressive potential of the advertiser as well as the receptive potential of the customer. We will explain how this communicative potential is dependent on the customer's cost of acquiring information and the degree to which they trust that information. Through the development and implementation of our dynamical model, we seek to understand how these potentials are related and which dynamics describe their intertwined trajectories.

2.1 Expressive and Receptive Potentials

Our starting point is the observation that communication is costly. On the part of the advertiser, this cost is expressive: how should the ad be formulated linguistically and visually, in which channels should it be transmitted, and at what times and frequencies? For the advertiser, the rate of return per ad matters: if a campaign does not result in increased sales, it may be a losing proposition to continue to increase the advertising budget. On the part of the customer, the cost is receptive: what is the information I need to make a decision, where can I find that information, how difficult will it be to collect it? The communicative dynamics arise in how these latent potentials engage with each other.

In order to prepare yourself for acquiring information, you need to free up your mind from competing concerns; you also need to activate the appropriate interpretive frames. This cognitive activation has a small but potentially significant metabolic cost; your brain is consuming sugars and oxygen and will at some point need rest to recover. By activating a certain cognitive frame, you selectively enhance your ability to gather certain types of information, but at the same time reduce your ability to acquire unrelated types of information. In this way the creation of a receptive potential imposes not only a cognitive and metabolic cost, but also an opportunity cost.

Moreover, your decision to prepare a targeted receptive potential carries a certain amount of uncertainty and risk. You must necessarily allocate resources to listen before you know what you are going to learn. Before you actually hear what the advertiser is going to say, you cannot know with certainty that the message is going to be informative along any of the dimensions you may be interested in. Let's consider a product that has twenty relevant dimensions of quality and functionality, from color and shape to expected lifetime and warranties; as a prospective customer, you may be searching for information regarding only five of these, and you cannot know beforehand that information relevant to your search will be provided. This uncertainty and risk adds to the cost of attention by creating a functional instability in the receptive potential.

The unavoidable cost associated with the creation of a receptive potential means that it's rational to be discriminative in gathering information (Daugherty et al., 2018), for instance by allocating a finite portion of available resources to the task, monitoring the unfolding act of communication, and extending or terminating the interaction according to a running assessment of the achieved and prospective costs and benefits.

3 Dynamics of Trust

3.1 Generation of Trust

The decision to allocate resources to the task of collecting information delivered through an advertisement is modulated by trust. Consequently, we must ask, how is trust generated? A simple way to model this is to say that trust is generated when a given allocation of attentional resources results in acquiring information that produces a coherent action. That is to say, we use information to guide our behavior, and trust is produced when the information provided allows us to guide our behavior so that our intentions are successfully realized. In a marketing interaction, let's say a prospective customer (a "prospect") has some initial interest in a product and is willing to allocate a modest attentional budget to an advertiser. The advertiser provides some information about the product and the prospect extracts some interpretation from this communicative act that results in a a set of expectations and intentions regarding the product. If these exceed some threshold of available resources and perceived net benefit, the prospect decides to purchase. If the product fulfills the prospect's expectations, as formed by the advertisement, trust is generated (Ogilvy, 1985). If you watch the ad for a smartphone and buy it, and you discover all kinds of features you like that you weren't even expecting, we have a situation of an "upside surprise"; in this case, goodwill is generated, a positive credit. This type of trust generation is described by Fung and Lee as an iterative process in which a customer must consistently evaluate whether the information acquired resulted in a successful interaction (Fung and Lee, 1999).

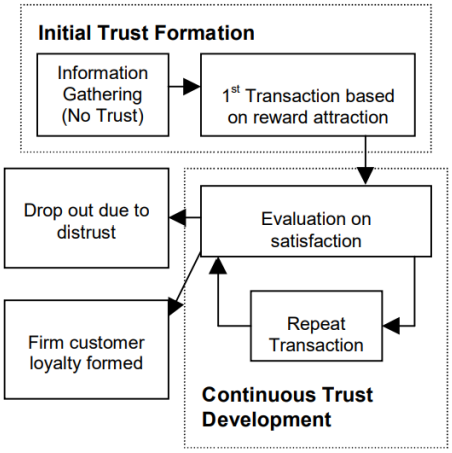


Figure 1: Fung & Lee Trust Development Cycle

3.2 Abuse of Trust

Just as trust can be produced and leveraged in social processes in other domains (Bednar and Page, 2021), advertisers can both produce and leverage trust. Having built a trusting relationship with their customers, advertisers can also behave in ways that leverage and potentially deplete goodwill and undermine trust. They may be motivated to do this in order to solve local problems; for instance, they may be trying to break into a crowded market already dominated by other players. Or they may be trying to introduce new products that haven't been fully tested or to increase profit margins.

Consider a company that has been engaging in producing trust by creating advertisements that honestly communicate the qualities and functionalities of a product and have successfully built trust. They might be tempted to leverage this trust by creating advertisements that exaggerate the benefits of the product. The prospect will attend to this ad, trustingly take it to be reliable, purchase the product, and be disappointed by its qualities. This will erode trust. While the advertiser may not be able to detect the loss of trust directly, it acts as a latent variable that can be inferred from observable behavior, such as the degree of responsiveness to ads (Hopkins, 1923).

4 Dynamical Model

As described above the primary factors that drive the advertiser-customer dynamics are the customers' trust in the advertisement and the cognitive cost trade-off associated with acquiring information from an advertisement. In modeling this dynamic we propose the following iterative system.

$$\tau = \text{Purchase Cycle} \tag{1}$$
$$N = \text{Terminal Purchase Cycle} \tag{2}$$
$$E(\tau) = \text{Expected Value} \tag{3}$$
$$A(\tau) = \text{Advertisement Count} \tag{4}$$
$$T(\tau) = \text{Trust} \tag{5}$$
$$P(\tau) = \text{Advertised Value} \tag{6}$$
$$\beta = \text{Experiential Value} \tag{7}$$
$$\alpha = \text{Expected Value per Unit of Trust} \tag{8}$$
$$\gamma = \text{Price Response Speed} \tag{9}$$
$$n = \text{Memory} \tag{10}$$

$$\sigma(z) = \frac{1}{(1+e^{-x})} - 1/2 \tag{11}$$

$$\mu(\tau) = \sigma(\frac{1}{n}\sum_{i=\tau-n}^{\tau-1}(\beta - P(\tau))) \tag{12}$$

$$A_e(\tau) = \frac{\gamma(A_c(\tau) - A_c(\tau-1))}{2\max(A_c(\tau), A_c(\tau-1))} \tag{13}$$

E, T, P are all initialized with initial values E_o, T_o, P_o, and all variables defined above are logical values.

Algorithm 1 Dynamical Model

for τ = *(1,2,3,...,N)* **do**
  $A_c = 0$
  while $E(\tau) < P(\tau)$ **do**
    $E(\tau) = E(\tau) + \alpha T(\tau)$
    $A_c(\tau) = A_c(\tau) + 1$
  end
  $T(\tau) = T(\tau-1) + \mu(\tau)$
  $P(\tau) = P(\tau-1) - A_e(\tau)$
  $E(\tau+1) = \beta T(\tau)$
end

4.1 Model Explanation

First, we acknowledge that other factors such as monopolization may impact buying behavior. For example if a company has created an ecosystem around their product than customers will have less perceived flexibility to change their buying behavior. Other factors like social class, gender, and life style play a major role in a customers purchase behavior. Even a company's ability to gradually improve the quality of their product can have an impact on a customer's purchase behavior.

However for the purpose of this simulation we begin with a few assumptions. First the advertiser creates a product with invariant experiential value β. In other words the quality of the product does not change. Next the customer has a tendency to purchase the product but has the flexibility to choose not to purchase the product. Take for example the BIC ballpoint pen, a product that customers tend to purchase but have the option to choose other pens on the market. Its also a product whose design and quality has not changed in over 50 years, As a result the advertiser is capable of only two distinct actions. First, increasing a customer's expected value of the product through repeated advertisements or adjusting the price of the product in order to catalyze a purchase.

At each purchase cycle τ_n, the advertiser enters an advertisement cycle where the customer is inundated with advertisements, counted within the variable A_c. After each advertisement, the customer's expected value $E(\tau)$ of the product is nudged closer to the advertiser's proposed value $P(\tau)$ at a rate of α, and in direct relation to the customer's trust in the advertisement $T(\tau)$ (Mitchell and Olson, 1977). If trust is low, then each advertisement will naturally be less effective in nudging the customer's expected value. In this way, the advertiser captures more of the surplus value. Once the gap between the customer's expected value $E(\tau)$ and the advertiser's proposed value $P(\tau)$ is closed, the customer makes a purchase and A_c is reset to 0 until the next advertisement cycle.

After the purchase, the customer evaluates the product by determining the differential between their experiential value and advertised value $\beta - P(\tau)$. If the differential is positive, the customer is satisfied, and trust $T(\tau)$ is nudged up, and vice versa. However, trust can be sticky (Weilbacher, 2003), implying that previous experiences with the product are considered by the customers. To implement this, the model draws on an arithmetic average of the customers' n previous evaluations of the product, which is fed into a sigmoid function $\sigma(z)$. The sigmoid function, in turn, produces the value used to nudge trust. This is in line with recent empirical results drawing from around 300,000 respondents across 71 countries which suggests that product reliability "has become a primary driver of consumer trust in recent years" (Khamitov et al., 2024). The use of the sigmoid function, on the other hand, allows us to bound the delta of trust after every purchase cycle τ which again reflects empirical results which suggest that trust is earned and broken with every customer experience and only in rare situations results in catastrophic loss or dramatic gain in trust. (Khamitov et al., 2024)

Next, the advertiser adjusts their proposed product value $P(\tau)$, based on the number of advertisements they've been running to trigger a sale. To adjust their proposed value, the advertiser calculates their advertisement expenditure by measuring the difference between the number of advertisements in the τ_n^{th} cycle and τ_{n-1}^{th} cycle. If the number of advertisements has increased, the advertiser will decrease their proposed price at a rate γ. This is done in response to a perceived loss of pricing power as their previous advertising cycle indicated a loss of efficacy. The latent variable driving this effect is a decline in trust in the ads on the part of the customer. After the advertiser's proposed price is updated, the customer resets their expected value to their experiential value weighted by their current degree of trust in the advertiser. This final step mimics a reasonable customer who sets their

expected value to the value they've just experienced post-purchase but who is still impacted by their residual trust in the advertisement. This type of dual factor approach to customer-side pricing is discussed by Sung and Chung who indicate that the price a customer is willing to pay is impacted both by the quality of the product and trust in the product, and that customers are willing to pay a premium for products produced by known/trusted brands (Sung et al., 2023).

4.2 Simulations

Below, we discuss three simulated examples along with their associated dynamics and interpretations. In figures 2-7, the x-axis denotes the purchase cycles τ_n. In figures 2, 4, and 6, the red horizontal line represents the experiential value β, the green curve represents the advertiser's proposed value, and the blue curve represents the customer's trust in the advertisement. In figures 3, 5, and 7, the blue curve represents the customer's trust in the advertisement, and the green curve represents the number of ads run by the advertiser during a given purchase cycle τ.

These simulations highlight the various interacting communicative and cognitive processes that create a repertoire of market behaviors. These processes include the advertiser's systematic production of trust, the advertiser's decision to leverage existing trust by raising prices, and to respond to decreasing ad efficacy by lowering prices. On the customer side, they include a willingness to allocate attentional resources to ads, to raise their expected value as a function of trust, to learn about the experienced value of a product, and to gradually lose trust when disappointed. We emphasize that we explore only a subset of all possible strategies. Market participants act with imperfect information about each other and the qualities of the product and attempt to manage cost and risk, including the financial and cognitive costs and risks of communication. Let's explore three scenarios: Communication Failure, Opportunistic Advertising, and Restrained Advertising.

4.3 Communication Failure

We begin with a scenario where the advertiser loses the trust of the customer, resulting in runaway ad spending and a communication breakdown. To model this, we substitute $A_e(\tau)$ in Algorithm 1 with $\gamma(\frac{dT(\tau)}{d\tau})$. Conceptually, this implies that trust is the driving force behind an advertiser's change in proposed value. When trust has positive momentum, the advertiser is incentivized to abuse that trust and increase the proposed value to increase revenue. Conversely, when trust is in decline, the advertiser seeks to rebuild trust by decreasing their proposed value to more closely align with the customer's expected value. In this scenario, when γ is sufficiently small, trust collapses, and the advertiser is too slow in adjusting their proposed value to regain trust quickly. This result is observed in Figure 2, showing a large lag between trust and the proposed price.

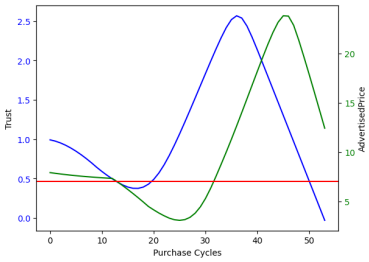


Figure 2: Communication Failure - Green: Advertised Price, Blue: Trust, Red: Initial Advertised Price $A(0)$

This delay results in trust reaching a negative value, where no amount of advertisements can help the advertiser regain trust, as each advertisement will decrease the customer's expected value via the update equation $E(\tau) = E(\tau) + \alpha T(\tau)$. In other words, once trust drops below 0, the customer begins to actively resist attempts by the advertiser to regain their trust. This can be seen in figure 3, where trust precipitates rapidly after the 40th purchase cycle and the advertiser fruitlessly increases the number of ads they run. This situation represents a communication failure between the advertiser and the customer, a term that Ries and Trout use to characterize the state of advertising in the 1970s and 80s (Trout and Ries, 1986).

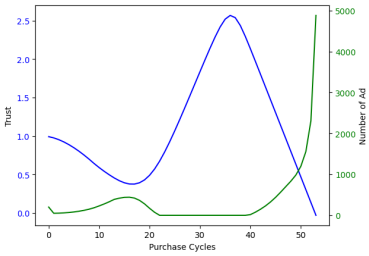


Figure 3: Communication Failure - Green: Number of Ads, Blue: Trust

4.4 Opportunistic Advertising

In this simulation, we revert to our main algorithm and choose a γ that produces stable oscillations. The results are visible in Figure 4, where both the trust and the advertiser's proposed value oscillate fairly regu-

larly. These stable oscillations model the push-and-pull relationship between the advertiser and customer, where pricing power is modulated by trust. Note the slight lag between trust and the advertiser's proposed value: this captures the causal arrow of increasing trust resulting in more efficient advertising and the generation of pricing power. Conversely, when the advertiser notices a drop in trust reflected in decreased sales and low efficacy of their ads, this induces the advertiser to lower their prices to maintain sales and eventually to rebuild trust.

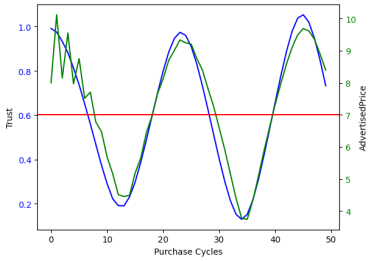


Figure 4: Opportunistic Advertising - Green: Advertised Price, Blue: Trust, Red: Initial Advertised Price $A(0)$

In Figure 5, we find the corresponding advertisement count at each purchasing cycle plotted with the customers' trust. Here, an interesting dynamic emerges between $A_c(\tau)$ and $T(\tau)$. The peaks of trust correspond to the troughs of advertisement and vice versa.

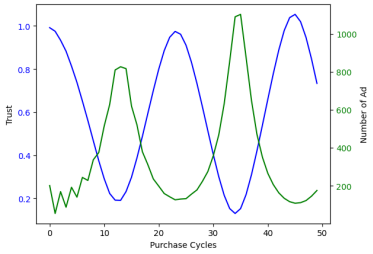


Figure 5: Opportunistic Advertising - Green: Number of Ads, Blue: Trust

This dynamic illustrates that when trust is high, the need for continuous advertisements is reduced since the customer already trusts the product due to repeated positive purchase experiences. Furthermore, when trust is high, the cognitive cost of accepting the advertiser's proposed price is decreased, as the customer has no reason to doubt or scrutinize the advertisement message. This means that each advertisement carries more weight, and, in aggregate, the total number of advertisements required to trigger a purchase is small. However, as customers experience that the product value is no longer as high as they were led to expect by the advertisements, their trust begins to decline and the efficacy of each ad rapidly begins to drop. The advertiser responds at first by increasing the frequency of ads in an attempt to maintain sales, peaking only when the payoff per ad drops too low and the game is no longer worth the candle: customers have lost almost all trust and are no longer responding much to the ads. At that point, the advertiser has lost his ability to maintain high prices and responds by moderating the value claims he makes in the ads. As customers begin to repeatedly find that the advertiser's claims are exceeded by experience, trust finally turns around. The advertiser notices that the efficacy of the ads is rising and responds by decreasing the ad frequency and raising prices, starting a new cycle.

In this scenario, we assume that the manufacturer has a fixed production capacity and is acting to maintain steady sales. A company with a growth capacity may behave differently, for instance by sustaining the frequency of advertising even in a high-trust and high-efficacy environment. This would change the shape of the ad frequency curve; however, it would not change the fundamental dynamics. A company eager to exploit the opportunity for growth would eventually encounter diminishing returns from their advertising campaigns due to falling trust and be forced to moderate their claims and lower their prices to maintain sales.

Do these swings make sense as attempts to secure a larger share of the available value residual? The oscillations pivot around the customers' expected value as determined through their experiences so that the end result is that the surplus value in play is evenly distributed around this axis. Any advantage of advertisers over customers is short-lived and negated in the next downturn; nevertheless, the short-term opportunities are real. Even if it is pointless to swing back and forth, the temptation to try to sell a product for more than it is worth is enduring.

Early advertising was dominated by medical preparations, promising miraculous cures (Gorlach, 2002). "Advertisements are now so numerous that they are very negligently perused," Samuel Johnson wrote in 1759, "and it is therefore become necessary to gain attention by magnificence of promises, and by eloquence sometimes sublime and sometimes pathetic" (Johnson, 1759). Such magnificent promises may work for a while, preying on the naive; how long the favorable wave of exploitation will last is not known in advance and one has the option of going out of business when the bluff is called.

This simple opportunistic advertising cycle is open to refinement, for instance by allowing trust to

be imported from the outside, such as with paid consumer testimonies or expert endorsements.

4.5 Restrained Advertising

By lowering γ, we can moderate the oscillations, signifying that advertisers are less responsive to the temptation to ramp up their claims when ad efficacy is high. Advertisers may learn that staying closer to the facts will progressively narrow the swings in ad efficacy. The outcome is visible in Figure 6, where the advertised price behaves like a dampened oscillator, converging towards the red line of expected value as $\tau \to \infty$.

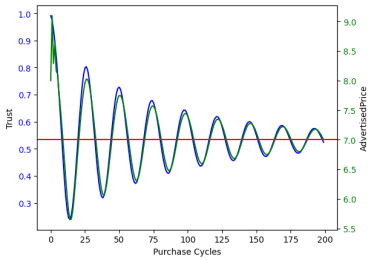


Figure 6: Restrained Advertising - Green: Advertised Price, Blue: Trust, Red: Initial Advertised Price $A(0)$

What the scenario shows is that an enterprise can establish a stable relation with its customers by providing a satisfactory product and advertising it at a value point that matches its experienced value. An advertiser adopting this strategy is showing a restraint that holds it back from exploiting the trust it painstakingly builds. This implies renouncing an opportunity for profit, possibly quite significant. A company wishing to stay in business over the long term, however, may well elect this strategy and prioritize long-term predictability and sustainability over short-term profit.

In Figure 7, the corresponding number of advertisements run during each purchase cycle is plotted, revealing that it similarly approaches a stable value. This means the company's advertising efforts and expenses will stabilize, reducing uncertainty and risk. By communicating effectively and honestly, the company can generate a reliable and sustainable business grounded in satisfied customers.

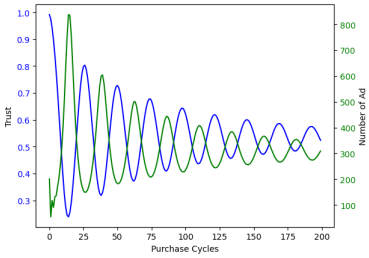


Figure 7: Restrained Advertising - Green: Number of Ads, Blue: Trust

An interesting entailment of the model is that raising the experienced value of the product may be the only stable way in which a company can secure a lasting larger share of the surplus value. This opens up an exploration of whether advertisements in principle are capable of altering not only the expected value of a good but also the experienced value.

The luxury market may be a good domain to study this issue. The experienced value of a product is not simply a fact of the market; it is a complex, multidimensional act of cognition. The experience of a product may be modulated by a sustained advertising campaign to establish an exclusive brand. By legally protecting a brand, the advertiser is able to control the associations to the product that the consumer is exposed to, thus constructing a cognitive frame within which the consumer will experience the product (Ogilvy, 1985). In this scenario, the purpose of advertising is to construct a cognitive platform that will raise the customers' assessment of the value of the product experience itself.

5 Conclusion

This paper has explored the dynamics of trust and communication in the context of advertising and market exchanges. We have developed a conceptual, mathematical, and computational model that captures some of the cognitive processes underlying these dynamics, focusing on the role of the cognitive and financial costs of production and reception of ads, the generation and opportunistic leveraging of trust, and the effects of strategic price adaptation.

Our model illustrates how trust is a pivotal element in the customer-advertiser relationship. Trust is built when advertisements accurately communicate product qualities, leading to fulfilled customer expectations. However, this trust is fragile and can be easily eroded if advertisers choose to exploit it by exaggerating product benefits, leading to customer disappointment. In addition, the model shows that the

cognitive cost of processing information from advertisements influences consumer behavior. Consumers are selective in their attention due to these costs and may choose to incrementally ignore advertisements as trust is eroded.

More generally, we propose that the behaviors of producers, advertisers, and customers take place in a space of asymmetrical perceptions of value that in itself does not determine economic outcomes. Instead, a rich panoply of cognitive processes that interact in complex ways allocate the prospective surplus value in ways that often do not reach a stable equilibrium. The emerging patterns may for instance oscillate stably for long periods, certain interventions will progressively dampen these oscillations and result in locally stable states, or trust may plunge below zero and result in communicative failure.

The information-processing approach to market exchanges defines a rich field of research in cognitive microeconomics and a space for computational models to create simulation frameworks for exploring this field. Future research may for instance examine the effects of importing trust into advertising by various means, compare the dynamics of broadcast advertising versus targeted digital advertising, and model how customers' experiences with a product, which in the present model we assume to be invariant, can in fact be nudged up through the cultivation of a brand image – a prospect that has potentially dramatic consequences for the long-term allocation of surplus value.

By situating microeconomics inside cognitive science, we assert that the processes that characterize market exchanges do not follow invariant laws, but instead create unstable possibilities with probabilistic outcomes. Within the context of the model developed above cognitive variables like $\{\beta, \alpha, \gamma, n\}$ form a space of parameters that define an infinite number of possible customer-advertiser dynamics. These parameters can be adjusted stochastically by relevant actors which presents them with a large possibility space of options at each decision point in the market exchange cycle. Only a subset of this space has been explored to date in actual market interactions and only a small subset of the parameter space has been explored in this paper, suggesting that we should expect the continued emergence of new behaviors. A much smaller proportion of behaviors has been modeled in terms of the cognitive processes involved, creating a rich territory for new discoveries.

REFERENCES

Bednar, J. and Page, S. E. (2021). The interplay between institutions and civic capacity: The paradox of democratic collapse.

Brooks, A. S., Yellen, J. E., Potts, R., Behrensmeyer, A. K., Deino, A. L., Leslie, D. E., and Clark, J. B. (2018). Long-distance stone transport and pigment use in the earliest middle stone age. *Science*, 360:90–94.

Daugherty, T., Hoffman, E., Kennedy, K., and Nolan, M. (2018). Measuring consumer neural activation to differentiate cognitive processing of advertising: Revisiting Krugman. *European Journal of Marketing*, 52(1/2):182–198.

Fung, R. and Lee, M. (1999). EC-Trust (Trust in Electronic Commerce): Exploring the Antecedent Factors. In *AMCIS 1999 Proceedings*, page 179.

Gorlach, M. (2002). A linguistic history of advertising, 1700-1890. *Amsterdam Studies in the History of Linguistic Science Series 4*, pages 83–104.

Hopkins, C. C. (1923). *Scientific Advertising*. New Line Publishing.

Johnson, S. (1759). The art of advertising exemplified. *The Idler*, 40.

Khamitov, M., Rajavi, K., Huang, D.-W., and Hong, Y. (2024). Consumer trust: Meta-analysis of 50 years of empirical research. *Journal of Consumer Research*, 51:7–18.

Marx, K. (2020). *Theories of Surplus Value: Volume 1*, volume 20. Pattern Books.

McCubbins, M. D. and Turner, M. (2020). Collective action in the wild. In Pennisi, A. and Falzone, A., editors, *The Extended Theory of Cognitive Creativity: Interdisciplinary Approaches to Performativity*, pages 89–102. Springer International Publishing.

Mitchell, A. A. and Olson, J. C. (1977). Cognitive effects of advertising repetition. *Advances in Consumer Research*, 4(1).

Ogilvy, D. (1985). *Ogilvy on Advertising*. Vintage Books, New York.

Ricardo, D. (1821). *On the principles of political economy*. J. Murray, London.

Smith, A. (1776). *An inquiry into the nature and causes of the wealth of nations: Volume One*. W. Strahan, London.

Sung, E., Chung, W. Y., and Lee, D. (2023). Factors that affect consumer trust in product quality: a focus on online reviews and shopping platforms. *Humanities and Social Sciences Communications*, 10(1):766.

Trout, J. and Ries, A. (1986). *Positioning: The battle for your mind*. McGraw-Hill New York, NY.

Von Winterfeldt, D. and Fischer, G. W. (1975). *Multi-Attribute Utility Theory: Models and Assessment Procedures*, pages 47–85. Springer Netherlands, Dordrecht.

Weilbacher, W. (2003). How advertising affects consumers. *Journal of Advertising Research*, 43(2):230–234.

Recasting the Aiyagari Model of Income into a Mean Field Game Problem

1 Introduction

In this paper, we seek to replicate the work done Achdou, Han, Lasry, Lions, and Moll in their article "Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach". The paper focuses on casting the Aiyagari Income Model as a Mean Field Game problem, that can be solved in terms of the associated Hamilton-Jacobi Bellman (HJB) and Focker-Plank (FP) equations. In this paper we review the Aiyagari model and the derivation of the MFG system.

For the purposes of this paper we add a small adjustment to the simulation in regards to the total labor supply. As currently formulated total labor supply remains constant throughout the simulation, however we elect to incorporate the idiosyncratic unemployment statuses of households in order to more accurately approximate how the system responds to shocks in labor supply. The motivation behind this adjustment comes from research done by Professor Anthony Klotz at Texas A&M who coined the term "The Great Resignation". The term describes the sudden jump in resignations amidst the COVID 19 pandemic. This phenomenon emphasizes the importance of considering labor supply changes when modeling household-firm interactions. Its important to note that the Aiyagari Model of Income does incorporate idiosyncratic state changes in employment, however its traditionally left out of the total labor supply computation. Throughout this paper we present a brief primer of the Aiyagari model of income that's been modified to incorporate idiosyncratic labor supply, along with necessary background of the HJB and KFP equations, before finally introducing a derivation of the Mean Field Game representation of our labor adjusted income model.

2 Aiyagari Model of Income: Introduction

The Aiyagari Model, is a macroeconomic framework that models the interaction between households and firms. Specifically, it models how households save and spend portions of their wealth in the face of idiosyncratic income shocks brought on by employers, and incomplete market information. The model assumes that households are heterogeneous in that they experience different income levels and wealth. The idiosyncratic income shocks are also a key feature of the model, since it describes unpredictable events in the lifetime of a household such as illness, job loss, or other personal economic changes like medical emergencies.

Under the Aiyagari framework, households consume in order to maximize the utility of their consumption. On the other hand households save not only to smooth consumption over their lifetime but also as a precaution against future income shocks. Our variation of the Aiyagari model will also feature a representative firm, that hires labor and purchases capital. The firm is also subject to the idiosyncratic productivity shocks brought on by sporadic changes in labor productivity. An equilibrium solution represents a distribution of wealth such that households are optimally choose their savings and spending to optimize for utility and smooth consumption.

Now we move on to the mathematical formulation of the Aiyagari model in terms of dynamic stochastic differential equations.

3 Aiyagari Model of Income: Formalization

3.1 Households

In order to define the necessary mathematical formulations of this problem we first define a few terms from the household's perspective, which will be used throughout this paper.

$$z_t = \text{Household labor output} \tag{1}$$
$$w_t = \text{Household wages} \tag{2}$$
$$r_t = \text{Risk-free interest rate} \tag{3}$$
$$a_t = \text{Household wealth} \tag{4}$$
$$c_t = \text{Household spending/consumption} \tag{5}$$
$$D = \text{Borrowing limit} \tag{6}$$

3.1.1 Utility Functional:

As stated above, households spend portions of their wealth to gain some sort of utility from their consumption, whether that utility be in the form of satisfaction, happiness, or value. Their goal is often to smoothen consumption and maximize utility. This goal is given in the form of the following utility functional in continuous time.

$$U = \max_{\{c_t\}} E_{t_0} \int_{t_0}^{\infty} e^{-\rho t} u(c_t) dt \tag{7}$$

ρ in this case is traditionally known as the rate of time preference. It reflects the household's preferences for present consumption over future consumption. In other words a $\rho \approx 0$ means that that a household prioritizes current spending over future spending since it increases the present utility. Whereas a higher ρ will discount the present value of the utility gained through consumption, and will place a higher weight on future consumption.

In discrete time analogue of the household utility function can be expressed as

$$U = \max_{c_t} \sum_{t}^{\infty} e^{-\rho t} u(c_t) \tag{8}$$

where ρ, again, is the degree to which a household prioritizes current consumption verses future consumption. If $\rho \approx 1$ then the household prioritizes future spending over current spending.

3.1.2 Utility Function:

The utility function $u(c_t)$ in equation (6) and (7) quantifies the satisfaction, happiness, or value an individual received from consumption at time t. the utility function is typically non-decreasing, since higher consumption will often lead to higher utility. However in order to capture the diminishing marginal utility of consumption, we impose that our utility function is strictly concave. In other words as consumption increase the marginal utility gained from consuming one additional unit decreases. Given those key features we can propose a few useful utility function. The first of which is called constant relative risk aversion (CRRA).

$$u(c) = \frac{c^{1-\gamma}}{1-\gamma} \tag{9}$$

Or we can use exponential utility.

$$u(c) = \frac{-1}{\theta} e^{-\theta c} \tag{10}$$

And lastly we can consider logarithmic utility

$$u(c) = \ln(c) \tag{11}$$

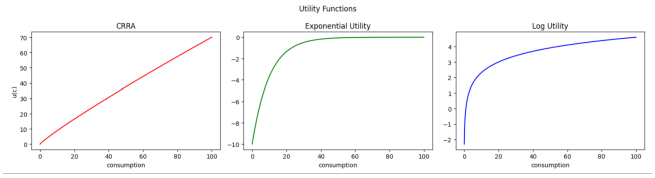


Figure 1: Utility Functions

3.1.3 Household Dynamics:

Households must also adhere to certain spending dynamics that relate their consumption wealth and income.

$$da_t = (z_t w_t + r_t a_t - c_t)dt \tag{12}$$

From the above equation we can observe that the change in wealth is function of a households wage income, income via risk-free investment, and the current consumption. The function

$$s(t,a,z) = z_t w_t + r_t a_t - c_t \tag{13}$$

will represent the amount saved by a household at time t, and will be important when solving the system using a mean field game framework.

We also suject household consumption to a borrowing constraint, which expressly prohibits a household from taking on an excessive amount of debt D.

$$a_t \geq -D \tag{14}$$

We also impose the following state constraint boundary condition, to ensure that constraint (14) is never violated.

$$c_t \leq z_t w_t + r_t(1 + a_t) - a_{t+1} \tag{15}$$

3.2 Firms:

On the other side of the interaction with households will be a representative Neoclassical firm. In order to define the necessary mathematical formulations of the firm-side problem we will again define a few terms from the firm's perspective. These terms in addition to those defined for Households will be used bellow.

$$A_t = \text{Total factor production} \tag{16}$$
$$\alpha = \text{Proportion of investment allocated to capital} \tag{17}$$
$$K_t = \text{Total capital supply} \tag{18}$$
$$L = \text{Total labor supply} \tag{19}$$
$$\delta = \text{Depreciation rate of capital} \tag{20}$$
$$Y_t = \text{Firm's output/productivity} \tag{21}$$
$$\tag{22}$$

A firms productivity is fully characterised by a firms investment in capital and labor. This is a popular production function that was first developed by Charles Cobb and Paul Douglas, and its the same production function used by the Aiyagari income model.

$$Y_t = A K_t^{\alpha} L_t^{1-\alpha} \tag{23}$$

3.2.1 Total Factor Production:

In the simplest case of the Aiyagari model A, sometimes refered to as the total factor productivity (TFP), is constant. However A_t can also be modeled as a function of time to reflect technological progress or other factors affecting productivity. A simple time-dependant model of the TFP can be represented as an exponential where A_0 is the initial level of productivity and γ^* is the rate of technological progress. In this way we model TFP in way that is analogous to Moore's law which describes the exponential growth of the number of transistors on a microchip. A phenomenon which has a direct effect on the productivity of digital systems.

$$A_t = A_0 e^{\gamma^* t} \tag{24}$$

3.2.2 Total Labor Supply:

Once again in the simplest case of the Aiyagari model the total labor supply L is considered constant. However L_t can also be modeled as a function of time to reflect growth in population or changes in labor supply. We will formalize this adjustment bellow in section 4.

3.2.3 Firm Side Problem & Derived Quantities:

Regardless of how K_t or L_t are expressed the firm's aim to maximize it production output remains the same. The problem can be given in terms of the productivity of the firm, appreciation/depreciation of assets, and wage spending.

$$\max_{K,L}\{AK_t^{\alpha}N^{1-\alpha} - (r_t + \delta)K_t - w_t L_t\} \tag{25}$$

Our aim will be to study the household behavior instead of the firm-side behavior, so we will not explicitly be using (25). However the firm's first order conditions with respect to capital supply K_t allows us to derive an explicit expression for the interest rate r_t.

$$r_t = \alpha A K_t^{\alpha-1} L_t^{1-\alpha} - \delta \tag{26}$$

And the firm's first order condition with respect to labor supply L allows to nail down an explicit expression for wages w_t

$$w_t = (1-\alpha)AK^{\alpha}L_t^{-\alpha} \tag{27}$$

3.2.4 Idiosyncratic Income Shocks:

In reality households are subject to idiosyncratic income shocks. In continuous time we can either represent these income shock as a diffusion process, where the change in income (z_t) can be expressed as

$$dz = \mu(z)dt + \sigma^2 dB_t \tag{28}$$

where sigma is the standard deviation of the magnitude of the shocks and $\mu(z)$ is drift of income. There are two ways of approaching the construction of $\mu(z)$. First we can assume that income slowly drifts up as a result of raises in wage or inflation adjustment. In this case

$$\mu(z) = mt + b \tag{29}$$

where m is the rate at which income drifts up and b is the average income at time 0. This construction may be useful if the reader seeks to implement the model for sufficiently large time horizon. However in the short term, we can treat z_t as a mean reverting process. Under this assumption, we can construct $\mu(z)$ as an Ornstein-Uhlenbeck Process.

$$\mu(z) = \theta(\bar{z} - z) \tag{30}$$

Where θ is the rate of mean reversion, and $\bar{z}$ is the average income to which z_t reverts.

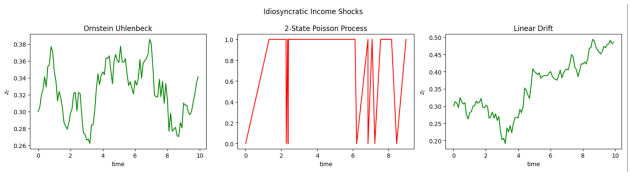


Figure 2: Idiosyncratic Income Shocks

4 Labor Supply Adjusted Model

Incorporating a change in employment or unemployment status into the total labor supply term in the Aiyagari Model, which traditionally uses a more simplified labor market representation, involves modifying the model to account for dynamic changes in labor force participation based on individual states which are also subject to stochastic changes. To implement this adjustment we adopt Achdou, Han, Lasry, Lions, and Mol's use of a 2 state Poisson process to model income idiosyncrasy.

A two-state Poisson jump process is a type of stochastic process that alternates between two distinct states, 0 and 1, with transitions occurring randomly at certain rates. We can denote the two states as $S_0 =$ unemployed and $S_1 =$ employed, the transitions between these states are governed by Poisson processes with different intensity rates.

Let λ_{01} be the rate at which transitions occur from S_0 to S_1 and λ_{10} be the rate for transitions from S_1 to S_0. The process can be described by the following transition probabilities:

$$\mathbb{P}(S_{t+\Delta t} = 1 | S_t = 0) = \lambda_{01}\Delta t \tag{31}$$

$$\mathbb{P}(S_{t+\Delta t} = 0 | S_t = 1) = \lambda_{10}\Delta t \tag{32}$$

However we propose that λ_{01} and λ_{10} are also subject to random fluctuations. We argue that the transition rates must not be fixed and instead ought to be sensitive to macroeconomic fluctuations. For instance, during a recession, job-separation rates might increase (due to layoffs and business closures), and job-finding rates might decrease (due to reduced hiring). Conversely, in a booming economy, job-finding rates might increase, and job-separation rates might decrease. Furthermore changes in government policy, such as increased unemployment benefits or job creation programs, can affect the incentives for individuals to find or leave jobs. Similarly, external shocks like technological changes or global events can rapidly alter the labor market landscape. Given these realistic conditions we assume that our transition rates are uniformly distributed.

The job-finding rate λ_{01} is drawn from a uniform distribution as follows:

$$\lambda_{01} \sim \text{Uniform}(a_{01}, b_{01}) \tag{33}$$

where a_{01} and b_{01} are the lower and upper bounds of the distribution, respectively. Similarly, the job-separation rate λ_{10} is drawn from a uniform distribution:

$$\lambda_{10} \sim \text{Uniform}(a_{10}, b_{10}) \tag{34}$$

where a_{10} and b_{10} are the lower and upper bounds of the distribution for λ_{10}, respectively.

We then use the above formulation of idiosyncratic employment changes to dynamically simulate the total labor supply at each time step in our simulation. We do this by letting L_t be the dynamically adjusting average described bellow.

$$L_t = \frac{z_0\lambda_{01} + z_1\lambda_{10}}{\lambda_{01} + \lambda_{10}} \tag{35}$$

This formula computes a weighted average of the productivities (z_j) for employed and unemployed households, where the weights are the transition probabilities between the states. This can be interpreted as an expectation of income or productivity level, taking into account the likelihood of being in either state. This allows L_t to reasonably estimate the total labor supply/productivity at any given time step.

Luckily, our labor supply adjustment does not greatly impact the deviating of our mean field game representation of the Aiyagari model, So we proceed by presenting a general derivation of the two core PDEs of a mean field game, before recasting the Labor Ajusted Aiyagari model.

5 General MFG System Derivation

5.1 General HJB Equation Derivation

In this section we derive the Hamilton-Jacobi-Bellman equation's generally.The Hamilton-Jacobi-Bellman (HJB) equation is a cornerstone in the theory of optimal control. It provides a necessary condition for optimality across a wide range of control problems, including both deterministic and stochastic systems. The derivation below outlines the fundamental steps involved in arriving at the HJB equation.

Assume a control system described by the differential equation

$$\dot{x}(t) = f(x(t), u(t), t) \tag{36}$$

where $x(t)$ represents the system state at time t, $u(t)$ denotes the control input, and f is a function defining the system dynamics. The objective is to find a control policy $u(t)$ that minimizes the cost functional

$$J = g(x(T)) + \int_{t_0}^{T} L(x(t), u(t), t)\, dt \tag{37}$$

where $g(x(T))$ is the terminal cost, and $L(x(t), u(t), t)$ is the running cost.

Step 1: Value Function. Define the value function V as the minimum cost-to-go (see principle of least action) from a state x at time t, under the given system dynamics and cost structure:

$$V(x(t), t) = \min_u \left\{ g(x(T)) + \int_t^T L(x(\tau), u(\tau), \tau)\, d\tau \right\} \tag{38}$$

Step 2: Principle of Optimality. The principle of optimality states that any optimal path's subpath is also optimal for its subproblem. This allows expressing the value function at $t + dt$ as:

$$\begin{aligned} V(x(t+dt), t+dt) = \min_u \{ & L(x(t), u(t), t)dt \\ & + V(x(t) + \dot{x}(t)dt, t+dt) \} \end{aligned} \tag{39}$$

Step 3: Hamilton-Jacobi-Bellman Equation. By expanding $V(x(t) + \dot{x}(t)dt, t+dt)$ using Taylor's series and neglecting higher-order terms, we obtain:

$$\begin{aligned} & V(x(t), t) + \frac{\partial V}{\partial t}dt + \nabla V \cdot \dot{x}(t)dt \\ & = \min_u \{ L(x(t), u(t), t)dt + V(x(t), t) \} \end{aligned} \tag{40}$$

After simplifying, the HJB equation emerges as:

$$\frac{\partial V}{\partial t} + \min_u \{ L(x, u, t) + \nabla V \cdot f(x, u, t) \} = 0 \tag{41}$$

This partial differential equation is central to solving optimal control problems by determining the value function $V(x,t)$ and thereby the optimal control policy.

5.2 General Focker Plank Equation

In this section we derive the general Kologmorov Focker-Plank equation. To do this We consider a continuous-time stochastic process described by the Stochastic Differential Equation (SDE)

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t$$

where X_t is the state variable at time t, $\mu(X_t, t)$ is the drift coefficient, $\sigma(X_t, t)$ is the diffusion coefficient, and dW_t represents the increment of a Wiener process (or Brownian motion).

Step 1: Chapman-Kolmogorov Equation. The Chapman-Kolmogorov equation for two times $s < t$ states the probability density function $p(x, t|y, s)$, of the process being in state x at time t, given it was in state y at time s, satisfies

$$p(x, t|z, 0) = \int p(x, t|y, s)p(y, s|z, 0)dy.$$

Step 2: Infinitesimal Generator. The infinitesimal generator L, acting on functions f of the process' state, is given by

$$Lf(x) = \lim_{\Delta t \to 0} \frac{\mathbb{E}[f(X_{t+\Delta t})|X_t = x] - f(x)}{\Delta t}.$$

For our SDE, this is

$$Lf(x) = \mu(x, t)\frac{\partial f}{\partial x} + \frac{1}{2}\sigma^2(x, t)\frac{\partial^2 f}{\partial x^2}.$$

Step 3: Derivation of the Fokker-Planck Equation. Considering a small time increment Δt and using the properties of X_t, the Fokker-Planck equation is derived as

$$\frac{\partial p}{\partial t} = -\frac{\partial}{\partial x}[\mu(x,t)p] + \frac{1}{2}\frac{\partial^2}{\partial x^2}[\sigma^2(x,t)p].$$

This equation describes how the probability density of the state variable X_t evolves over time due to both deterministic drifts and stochastic diffusions.

5.3 Aiyagari Model HJB Equation

In this section we will recast the Aiyagari model, where the idiosyncratic income changes are represented by a 2 state Poisson Process, into an HJB equation. To do this we will cook a bellman equation that serves the encodes similar features to the Aiyagari model above.

Consider the following income fluctuation problem in discrete time. Periods are of length Δ, individuals discount the future with discount factor

$$\beta(\Delta) = e^{-\rho\Delta} \tag{42}$$

and individuals with income y_j keep their income with probability

$$p_j(\Delta) = e^{-\lambda_j\Delta} \tag{43}$$

and switch to state y_{-j} with probability $1 - p_j(\Delta)$. The Bellman equation for this problem is:

$$\begin{aligned} v_j(a_t) = \max_c \big\{ & u(c)\Delta + \beta(\Delta)\left(p_j(\Delta)v_j(a_{t+\Delta})\right. \\ & \left. + (1-p_j(\Delta))v_{-j}(a_{t+\Delta})\right)\big\} \end{aligned} \tag{44}$$

subject to

$$a_{t+\Delta} = \Delta y_j + ra_t - c + a_t \tag{45}$$

$$a_{t+\Delta} \geq a \tag{46}$$

functionally the Bellman equation encodes a households desire to maximize current utility while balancing the probability of future income shocks on the capacity for consumption to produce utility.

For $j = 1, 2$. We will momentarily take $\Delta \to 0$ so we can use that for Δ small

$$\begin{aligned} \beta(\Delta) &= e^{-\rho\Delta} \\ &\approx 1 - \rho\Delta \end{aligned} \tag{47}$$

$$\begin{aligned} p_j(\Delta) &= e^{-\lambda_j\Delta} \\ &\approx 1 - \lambda_j\Delta. \end{aligned} \tag{48}$$

Substituting these into (4) 4we have

$$\begin{aligned} v_j(a_t) = \max_c \big\{ & u(c)\Delta + (1-\rho\Delta)\left((1-\lambda_j\Delta)v_j(a_{t+\Delta})\right. \\ & \left. + \lambda_j\Delta v_{-j}(a_{t+\Delta})\right)\big\} \end{aligned} \tag{49}$$

subject to (45) and (46). Subtracting $(1-\rho\Delta)v_j(a)$ from both sides and rearranging, we get

$$\begin{aligned} \Delta v_j(a_t) = \max_c \big\{ & u(c)\Delta + (1-\rho\Delta)\left(v_j(a_{t+\Delta}) - v_j(a_t)\right) \\ & + \lambda_j\Delta\left(v_{-j}(a_{t+\Delta}) - v_j(a_{t+\Delta})\right)\big\} \end{aligned} \tag{50}$$

Dividing by Δ, taking $\Delta \to 0$ and using that

$$\begin{aligned} & \lim_{\Delta\to 0} \frac{v_j(a_{t+\Delta}) - v_j(a_t)}{\Delta} \\ & = \lim_{\Delta\to 0} \frac{v_j(\Delta y_j + ra_t - c + a_t) - v_j(a_t)}{\Delta} \\ & = v_j'(a_t)(y_j + ra_t - c) \end{aligned} \tag{51}$$

we find (52); The HJB equation for the Aiyagari model.

$$\begin{aligned} \rho v_j(a) = \max_c \big\{ & u(c) + v_j'(a)(y_j + ra - c) \\ & + \lambda_j(v_{-j}(a) - v_j(a))\big\}, \end{aligned} \tag{52}$$

5.4 Aiyagari Model Focker Plank Equation

Recall that under our labor adjustment a household can either be employed or unemployed, and that the transition from one state to another follows a two-state Poisson process. For the sake of stability we assign an unemployed household a low productivity state and vice versa; $z_j \in \{z^L, z^H\}$. Also recall that wealth evolves as follows.

$$a_t = a_{t+\Delta} - \Delta s_j(a_t)).$$

We let $G_j(a,t)$ define the wealth distribution CDF of our population of households. Such that the bellow quantity represents the fraction of people with income z_j and wealth below a.

$$G_j(a,t) = \mathbb{P}(a_t \leq a, z_t = z_j)$$

We claim that the probability of an individual having wealth bellow a is the following. Intuitively this makes sense since the probability of a household having wealth bellow a at $t+\Delta$ is equal to the probability that a household has wealth bellow a at the previous time step plus the probability that the household's current wealth is greater then a at time t but has the capacity to cross bellow the threshold a during the next time step.

$$\mathbb{P}(a_{t+\Delta} \leq a) = \underbrace{\mathbb{P}(a_t \leq a)}_{\text{already below threshold } a} + \underbrace{\mathbb{P}(a \leq a_t \leq a - \Delta s_j(a))}_{\text{cross threshold } a} \tag{53}$$

which is equivalent to

$$\begin{aligned} \mathbb{P}(a_{t+\Delta} \leq a) = \mathbb{P}(a_t \\ \leq a - \Delta s_j(a)) \end{aligned} \tag{54}$$

Now in order to compute the joint probability of $\mathbb{P}(a_t \leq a, z_t = z_j)$ we incorporate the probabilities of income shocks as defined in (47) and (48).

$$\begin{aligned} \mathbb{P}(a_{t+\Delta} \leq a, y_{t+\Delta} = y_j) = (1-\Delta_j)\Pr(a_t \\ \leq a - \Delta s_j(a), y_t \\ = y_j) + \Delta_{-j}\Pr(a_t \\ \leq a - \Delta s_{-j}(a), y_t \\ = y_{-j}). \end{aligned} \tag{55}$$

We then use the definition of G_j, which yields the following.

$$\begin{aligned} G_j(a,t+\Delta) = (1-\Delta_j)G_j(a - \Delta s_j(a), t) \\ + \Delta_{-j}G_{-j}(a - \Delta s_{-j}(a), t) \end{aligned} \tag{56}$$

We then divide out by Δ, then take $\Delta - \to 0$. Using the fact that $g(a) = \partial_a G(a)$, we get the KFP equation.

$$0 = -\frac{d}{da}[s_j(a)g(a)] - \lambda_j g(a) + \lambda_{-j} g_{-j}(a),$$

References

Achdou, Y., Han, J., Lasry, J.-M., Lions, P.-L., & Moll, B. (2021). Income and wealth distribution in macroeconomics: A continuous-time approach. *The Review of Economic Studies*, *89*(1), 45-86. Retrieved from https://doi.org/10.1093/restud/rdab002 doi: 10.1093/restud/rdab002

Andersson, A. (n.d.). *Introduction to the hamilton-jacobi-bellman equation.* Online. Retrieved from https://www.math.chalmers.se/ donnerda/StochasticControl.

Dustmann, C., Schönberg, U., & Stuhler, J. (2016). Labor supply shocks, native wages, and the adjustment of local employment*. *The Quarterly Journal of Economics*, *132*(1), 435-483. Retrieved from https://doi.org/10.1093/qje/qjw032 doi: 10.1093/qje/qjw032

Klotz, A. (2022). The great resignation is still here, but whether it stays is up to leaders. In *Oecd forum.*

Risken, H. (1996). *The fokker-planck equation methods of solution and applications*. Berlin: Springer.