Characterizing Qubit Dynamics using Physics Informed Machine Learning Techniques

Abhishek Mhatre

PY 580 Final Project

December 15, 2025

Abstract

This project implements a machine learning model for single-qubit dynamics driven by two quadrature control channels. Using data generated from simulated qubit evolution under a time-dependent Hamiltonian, an LSTM-based sequence model is trained to predict Bloch-vector trajectories $(\langle\sigma_x\rangle, \langle\sigma_y\rangle, \langle\sigma_z\rangle)$ given an initial qubit state and a pulse sequence $(I(t), Q(t))$. The learned model is then embedded inside a gradient-based optimal control loop to solve an inverse problem: given a target Bloch vector, optimize a control pulse that steers the system to the target. We demonstrate that our machine learning method solves the initial problem with $\sim 99.97\%$ fidelity but struggles to accurately describe the qubit dynamics for the inverse problem, resulting in a higher infidelity of $\sim 3.58\%$ and significantly high distance separation between predicted and targeted state. The results for the pulse optimization problem are analyzed to determine possible causes of the model collapse and potential fixes.

1 Introduction

High-fidelity quantum operations are essential for quantum information science. These operations are possible only if one knows the accurate description of the qubit dynamics as this allows to design time-dependent controls that implement quantum gates [1–3]. Therefore, precise characterization of qubit interaction becomes important in all fields of quantum sciences: quantum computing, sensing, and communication. Traditional open-loop control relies on analytical models that often fail to capture experimental realities such as pulse distortions or crosstalk. A recently published paper [4] demonstrates that supervised machine learning models can efficiently learn and predict the response of a single qubit to arbitrary control pulses, offering a powerful alternative to traditional numerical simulations or analytical models. This learned model is then used to tackle the inverse problem: predicting optimized pulse sequence to reach a particular target state (quantum optimum control). Here, I attempt to reproduce the results from [4] using simulated data and simplified Hamiltonian interaction. This is achieved in a two step process. First (forward problem), we trained physics-informed Recurrent Neural Network (RNN), specifically Long Short-Term Memory (LSTM) to learn the quantum dynamics on a training dataset and predict the final state of the qubit from the given input: Initial state and Pulse Sequence. Then, I lock the weights of this trained model and optimize a microwave pulse sequence to realize quantum states of interest (inverse problem).

The project report is organized as follows: In Sec.(2), we explain the setup of the forward problem, the system Hamiltonian, state representation, etc., in essence, the physics of our simulated quantum world. The dataset construction, description of our machine learning model, and its

performance is illustrated in Sec.(3). The setup for pulse optimization control (inverse problem) is further elucidated in Sec.(4), followed by learned model's performance and its analysis in Sec.(5). I conclude this report with a brief discussion of my results and future work relevant to the field.

2 Simulted Qubit Dynamics

Driven qubit Hamiltonian

A Hamiltonian characterizing interaction between any of the six initial bloch states and microwave pulse, very basically, can expressed as

$$H(t) = H_0 + p(t)H_1$$

where H_0 is the static part acting on an initial Bloch state and $p(t)$ is the control pulse sequence, driving the qubit to a specific target state. I have considered a slightly more complicated system for this work, a single qubit driven by two control pulses, denoted as the in-phase and quadrature components of a microwave drive, $I(t)$ and $Q(t)$, respectively. The motivation behind this was to stick to the convention used in signal processing. In this simplified model, the time-dependent Hamiltonian is

$$H(t) \;=\; \frac{1}{2}\sigma_z \;+\; \frac{1}{2}I(t)\,\sigma_x \;+\; \frac{1}{2}Q(t)\,\sigma_y, \tag{1}$$

where $\hat{\sigma}_{x,y,z}$ are the Pauli operators. The work that motivates this project simulates an extremely complicated full-blown Hamiltonian for a superconducting transmon [4]. In our Hamiltonian, first term represents a static drift (free precession along the z-axis in the lab frame), while the other terms represent pulse-induced rotations in the Bloch sphere. Throughout this work, we assume unitary evolution (no decoherence) and, as a result, the system dynamics can be completely described by a time-dependent Schrödinger equation:

$$\hat{H}(t)\,|\psi(t)\rangle \;=\; i\hbar\frac{d}{dt}\,|\psi(t)\rangle \tag{2}$$

State representation on the Bloch sphere

We begin with the six pure cardinal states on the Bloch sphere $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle, |+i\rangle, |-i\rangle\}$, all are initial states. Each training sample chooses one of these cardinal states. Any Bloch vector $\mathbf{r} \in \mathbb{R}^3$ is simply the expectation value of pauli matrices of the respective co-ordinate

$$\mathbf{r} \;=\; \big(\langle\sigma_x\rangle,\, \langle\sigma_y\rangle,\, \langle\sigma_z\rangle\big) \tag{3}$$

Physical states satisfy $\|\mathbf{r}\| \leq 1$; pure states lie on the surface ($\|\mathbf{r}\| = 1$), while mixed states lie inside the Bloch sphere($\|\mathbf{r}\| < 1$). The setup in the previous subsection is used to create simulated datasets using QuTiP's solver [5]. Given an initial state and a discrete-time pulse sequence, trajectory of the Bloch vector evloves over time as described in Eq.(4).

$$\mathbf{r}_{0:T-1} \;=\; \{\mathbf{r}(0), \mathbf{r}(1), \ldots, \mathbf{r}(T-1)\}, \tag{4}$$

3 Learning Qubit Dynamics with an LSTM Network

Data generation with QuTiP

Supervised training data is generated by randomly sampling initial states and control pulses, simulating the resulting qubit evolution, and recording the Bloch trajectory [5]. The simulation pipeline is as follows:

1. **Initial state sampling:** Begin with six cardinal pure states as initial states. Each is chosen uniformly at random.
2. **Pulse sampling:** Sample discrete control sequences (I_t, Q_t) for $t = 0, \ldots, T-1$ from a uniform distribution and then apply a Gaussian smoothing filter to mimic bandwidth limitations.
3. **Forward simulation:** The random qubit evolves according to Eq. (1) and the values of $\langle\sigma_x\rangle, \langle\sigma_y\rangle, \langle\sigma_z\rangle$ is recorded at each time step.

The dataset is split into three parts: training, validation, and test sets. To emulate realistic estimation of Pauli expectations, small Gaussian noise is added to the trajectory. The model is trained on the training set, hyperparameters are selected based on validation loss, and model's performance on the test set is reported.

Sequence-to-sequence model

We train a sequence model that maps an initial condition and a pulse sequence to a Bloch-vector trajectory:

$$\Big(s,\ \{(I_t, Q_t)\}_{t=0}^{T-1}\Big)\ \mapsto\ \{\hat{\mathbf{r}}(t)\}_{t=0}^{T-1}, \tag{5}$$

where $s \in \{0, \ldots, 5\}$ is an index labeling the initial cardinal state.

The model is an LSTM-based recurrent neural network. The initial state index is one-hot encoded and passed through a linear layer to produce an embedding, which seeds the initial hidden state of the LSTM. The LSTM reads the pulse sequence and produces a hidden feature of dimension 128 at each time step, which is decoded into a 3D Bloch-vector prediction using a linear layer followed by a tanh activation to bound outputs in $[-1, 1]$. These bounds ensure that the states predicted are physically possible.

Training objective and optimization

The primary supervised loss is quantified as mean-squared error over time and batch:

$$\mathcal{L}_{\text{MSE}} \ = \ \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=0}^{T-1} \big\|\hat{\mathbf{r}}_b(t) - \mathbf{r}_b(t)\big\|_2^2. \tag{6}$$

We train with the Adam optimizer and use a learning-rate scheduler that reduces the step size on plateaus of validation loss. Performance is monitored by training/validation loss curves and by comparing the true expectation values to the predicted ones.

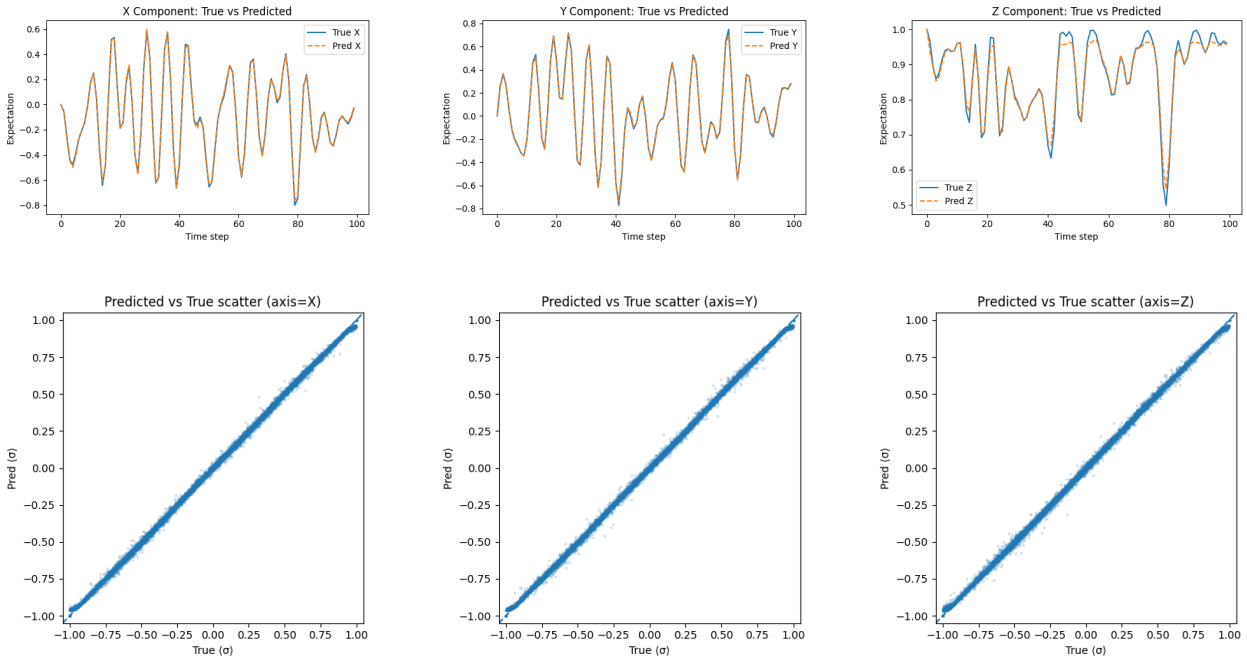

Figure 1: Top row: For random test examples, true and predicted components $\langle \sigma_{x,y,z} \rangle$ over time to confirm that the network captures the phase and the amplitude. Bottom row: Scatter plots of predicted vs. true $\langle \sigma_i \rangle$ confirm that model is accurate over whole region of interest. The fidelity of these results is satisfactorily high at $\sim 99.97\%$.

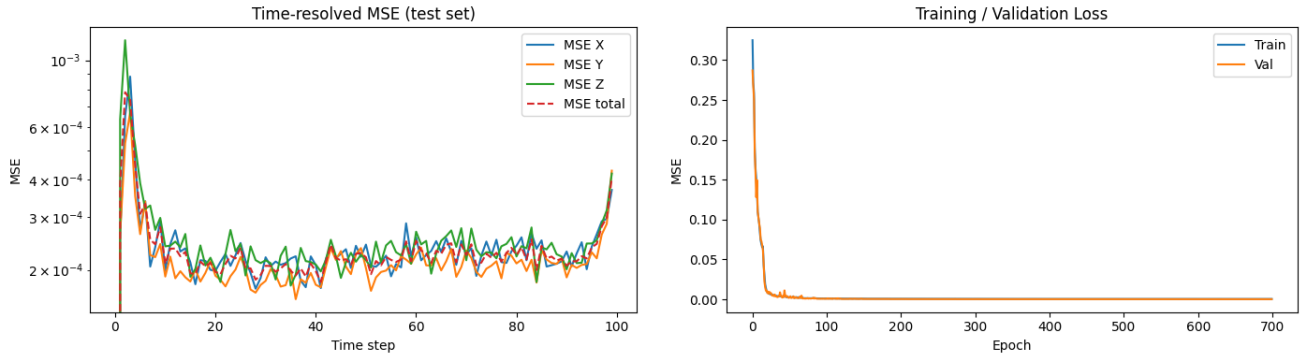

Figure 2: Right: The MSE for all the predicted expectation values over all time steps. Left: Training and validation loss over 700 epochs during a supervised training of the LSTM network.

Forward-model Evaluation Results

4 Inverse Problem: Pulse Optimization via the Learned Model

Problem statement

After training, we freeze the network weights and use the learned model in a gradient-based control loop to find the pulses bestrealizing the quantum operations of interest via backpropagation. Given an initial state index s and a target Bloch vector $\mathbf{r}_\star$, we seek a pulse sequence $\{(I_t, Q_t)\}_{t=0}^{T-1}$ such that the final predicted state matches the target:

$$\{(I_t, Q_t)\}^\star \;=\; \arg\min_{\{(I_t,Q_t)\}} \; \mathcal{E}\big(\hat{\mathbf{r}}(T-1),\, \mathbf{r}_\star\big). \tag{7}$$

This constitutes an inverse problem because the control inputs are unknown and must be computed to achieve a desired output state.

Pulse parameterization and bandwidth limitation

To keep the optimized pulses physically realistic, we optimize an unconstrained pulse variable $\tilde{u}_t \in \mathbb{R}^2$ and map it to bounded amplitude via

$$u_t \;=\; u_{\max} \tanh(\tilde{u}_t), \tag{8}$$

which enforces $|I_t|, |Q_t| \le u_{\max}$. We then apply a differentiable smoothing operation (a 1D convolution) to mimic limited bandwidth, ensuring that the optimization method cannot cheat by using unphysical high-frequency components or rapidly changing values not physically realizable.

Fidelity for pure and mixed states

Since the goal of this work is to support both pure and mixed target states, we use a fidelity definition valid for general qubit density matrices. For states represented by Bloch vectors $\mathbf{r}$ and $\mathbf{s}$ with $\|\mathbf{r}\|, \|\mathbf{s}\| \le 1$, the Uhlmann fidelity [6] is

$$F(\mathbf{r}, \mathbf{s}) = \frac{1}{2}\left(1 + \mathbf{r}\cdot\mathbf{s} + \sqrt{(1-\|\mathbf{r}\|^2)(1-\|\mathbf{s}\|^2)}\right), \qquad 0 \le F \le 1. \tag{9}$$

This expression reduces to $(1 + \mathbf{r}\cdot\mathbf{s})/2$ for pure states, where $\|\mathbf{r}\| = \|\mathbf{s}\| = 1$.

Energy loss function

The energy loss function to be optimized is as follows:

$$\mathcal{E} \;=\; \alpha\left[-\log(F(\hat{\mathbf{r}}(T-1), \mathbf{r}_\star) + \epsilon)\right] \;+\; \beta\,\|\hat{\mathbf{r}}(T-1) - \mathbf{r}_\star\|_2 \;+\; \gamma \sum_{t=1}^{T-1} \|u_t - u_{t-1}\|_2^2, \tag{10}$$

The first term in Eq.(10) serves the purpose of maximizing the fidelity of the predicted final state (from the optimized pulse) with the target state. This is followed by two L2 penalties. L2 norm (second term) "pulls" the latest Bloch vector to the target state early in the optimization, providing a reliable gradient direction. To force the model to produce physically realistic optimized pulses, a squared L2 norm (third term) is introduced. It ensures that the optimization cannot exploit unphysical pulse sequences to minimize the energy loss.

Variables and notation

- $\hat{\mathbf{r}}(T-1)$: The predicted final Bloch vector produced by the frozen ML model when driven by the candidate pulse sequence $u_{0:T-1}$.
- $\mathbf{r}_\star$: The target Bloch vector that the control aims to reach at the final time step.
- $u_t = (I_t, Q_t)$: the 2D control input at discrete time step t, consisting of the in-phase (I_t) and quadrature (Q_t) components.
- α, β, γ: non-negative weighting coefficients that set the relative importance (or penalty) of the fidelity term, the final-state distance term, and the pulse smoothness regularizer in the loss function.
- $\epsilon = 10^{-8}$: a small numerical constant used to avoid singularities such as $\log(0)$ in the loss function.

5 Results for the Inverse Problem

The learned model , despite achieving fidelity of 99.97% on the validation dataset during training, fails to predict the optimized pulse sequence, significantly. Despite a fidelity of 96.42%, the distance error is huge ~ 1.95 (Max distance error $= 2$) and the predicted optimized pulse does not produce the desired state. A similar performance was seen for various initial and target states with significant infidelity and huge distance error.

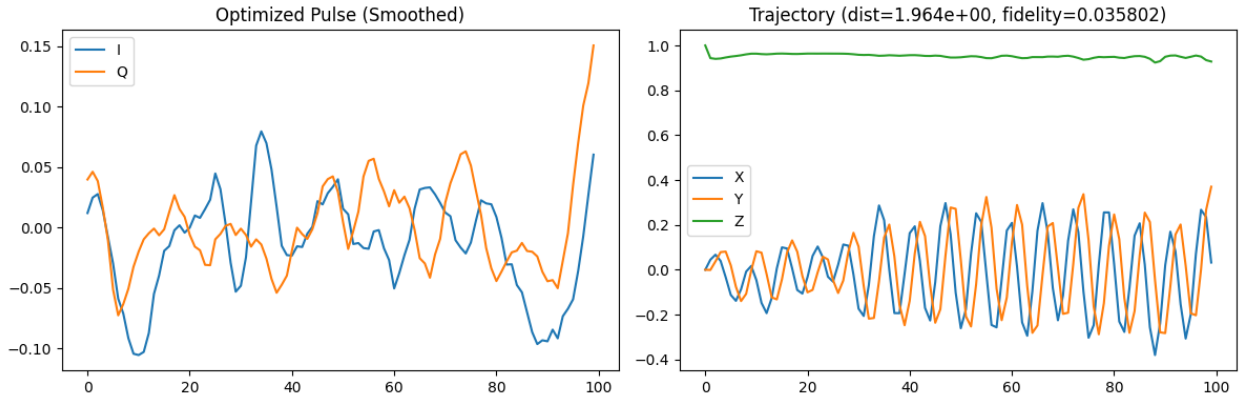

Figure 3: Here is an example of pulse I(t) and Q(t) over time steps predicted by the model. The target state is $|1\rangle = (0, 0, -1)$ from an initial state of $|0\rangle = (0, 0, 1)$. The optimized pulse predicted by the model leads us to the following final state $(0.032252, 0.370191, 0.928396)$. Typo: The title of plot on the right is reporting infidelity and not fidelity

Failure Analysis & Potential Causes

It is clear from the trajectories in Fig. (3) that X and Y values are oscillating around zero and Z almost stays constant near ~ 1. This has led me to conclude that my initial bloch vector is rotating about Z the entire time and never tries to push itself in the direction of the target state. Here are some of the potential causes of the model collapse, despite low MSE on the training data.

- Training dataset does not cover the entire Bloch space. In Fig. (3), we can see that the pulse amplitude ranges between -0.1 to 0.15 for both I and Q. This indicates the possibility that the model may not have learned to cover the whole region of state space, thereby, failing to explore far from the initial state. Sending high amplitude and diverse structured pulses (for e.g. with sign-flips, constant segments, sine waves) may provide comprehensive training to reduce this error. Another effective solution could be iteratively expanding the dataset using pulses encountered during control.

- Optimizer is getting stuck near Z ~ 1. In the pulse optimization function, scheduler learning rate is computed after every 10 epochs and a drop to a learning rate $\sim 10^{-7}$ was observed very early in the optimization, thereby, freezing the pulse sequence updates.

6 Conclusion

This project implemented a two-stage workflow inspired by [4]: (i) learning forward qubit dynamics from pulse-response trajectories, and (ii) solving an inverse pulse-design problem by backpropagating through the gradient differentiable learned model. The forward model successfully captured Bloch-vector trajectories generated from simulations and respected physical validity constraints.

In the inverse-control stage, pulse optimization performance is supersensitive to the choice of pulse parameterization, amplitude constraints, and the relationship between the training pulse distribution and the pulses produced by optimization. Achieving total inversions such as $|0\rangle \rightarrow |1\rangle$ in the presence of the drift term (H_0) in Eq. (1) are difficult [7]. As H_0 sets the "intrinsic limit" on how fast we can reach the required target [7]. A convenient solution to counter this effect is to drop the drift term entirely, but not ideal for real-life experiments. A model with larger control amplitudes, composite pulses, or a rotating-frame formulation that reduces effective drift might produce high-fidelity results. These considerations highlight the central challenge addressed by Machine-Learning-based Quantum Optimum Control: ensuring that learned models remain accurate in the regions of state space that are most relevant for control.

Overall, the results support the main conceptual claim of [4]: supervised learning can produce a fast differentiable model of qubit dynamics that enables gradient-based control. Future work includes extending the simulator to open-system dynamics (collapse operators), incorporating realistic hardware distortions, and iteratively refining the dataset using pulses discovered during optimization to overcome the limitations of this model.

7 Acknowledgement

This work relies on the work of all the references listed below. Generative AI models have been used throughout this work, where necessary.

References

[1] Anthony P. Peirce, Mohammed A. Dahleh, and Herschel Rabitz. "Optimal control of quantum-mechanical systems: Existence, numerical approximation, and applications". In: *Physical Review A* 37.12 (1988), pp. 4950–4964. DOI: 10.1103/PhysRevA.37.4950. URL: https://link.aps.org/doi/10.1103/PhysRevA.37.4950.

[2] J Werschnik and E K U Gross. "Quantum optimal control theory". In: *Journal of Physics B: Atomic, Molecular and Optical Physics* 40.18 (Sept. 2007), R175. DOI: 10.1088/0953-4075/40/18/R01. URL: https://doi.org/10.1088/0953-4075/40/18/R01.

[3] Christiane P. Koch et al. "Quantum optimal control in quantum technologies. Strategic report on current status, visions and goals for research in Europe". In: *EPJ Quantum Technology* 9.1, 19 (Dec. 2022), p. 19. DOI: 10.1140/epjqt/s40507-022-00138-x. arXiv: 2205.12110 [quant-ph].

[4] Élie Genois et al. "Quantum optimal control of superconducting qubits based on machine-learning characterization". In: *Phys. Rev. Appl.* 24 (3 Sept. 2025), p. 034073. DOI: 10.1103/d9yg-d3qr. URL: https://link.aps.org/doi/10.1103/d9yg-d3qr.

[5] Neill Lambert et al. "QuTiP 5: The Quantum Toolbox in Python". In: *Physics Reports* 1153 (2026), pp. 1–62. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2025.10.001. URL: https://www.sciencedirect.com/science/article/pii/S0370157325002704.

[6] Armin Uhlmann. "The "transition probability" in the state space of a *-algebra". In: *Reports on Mathematical Physics* 9.2 (1976), pp. 273–279. DOI: 10.1016/0034-4877(76)90060-4.

[7] Christian Arenz et al. "The roles of drift and control field constraints upon quantum control speed limits". In: *New Journal of Physics* 19.10 (Oct. 2017), p. 103015. DOI: 10.1088/1367-2630/aa8242. URL: https://doi.org/10.1088/1367-2630/aa8242.

[8] Jens Koch et al. "Charge-insensitive qubit design derived from the Cooper pair box". In: *Phys. Rev. A* 76 (4 Oct. 2007), p. 042319. DOI: 10.1103/PhysRevA.76.042319. URL: https://link.aps.org/doi/10.1103/PhysRevA.76.042319.

[9] A.E.E. Dubois et al. "Untrained Physically Informed Neural Network for Image Reconstruction of Magnetic Field Sources". In: *Phys. Rev. Appl.* 18 (6 Dec. 2022), p. 064076. DOI: 10.1103/PhysRevApplied.18.064076. URL: https://link.aps.org/doi/10.1103/PhysRevApplied.18.064076.

[10] J.R. Johansson, P.D. Nation, and Franco Nori. "QuTiP 2: A Python framework for the dynamics of open quantum systems". In: *Computer Physics Communications* 184.4 (2013), pp. 1234–1240. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2012.11.019. URL: https://www.sciencedirect.com/science/article/pii/S0010465512003955.

[11] J.R. Johansson, P.D. Nation, and Franco Nori. "QuTiP: An open-source Python framework for the dynamics of open quantum systems". In: *Computer Physics Communications* 183.8 (2012), pp. 1760–1772. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2012.02.021. URL: https://www.sciencedirect.com/science/article/pii/S0010465512000835.

[12] Ekaba Bisong. "Google Colaboratory". In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress, 2019, pp. 59–64. ISBN: 978-1-4842-4470-8. DOI: 10.1007/978-1-4842-4470-8_7. URL: https://doi.org/10.1007/978-1-4842-4470-8_7.

[13] A.E.E. Dubois et al. "Untrained Physically Informed Neural Network for Image Reconstruction of Magnetic Field Sources". In: *Phys. Rev. Appl.* 18 (6 Dec. 2022), p. 064076. DOI: 10.1103/PhysRevApplied.18.064076. URL: https://link.aps.org/doi/10.1103/PhysRevApplied.18.064076.