Methods and Constants

Development Process

When working on our process, maintaining a safe programming environment became essential as we switched between qBraid and local environments. Our git environment was very standard, one main branch with branches off, and commits required pull requests to move upstream.

File layout

We chose to solve each problem in an independent notebook to make parallel work simpler. Additionally near the end, we refactored to hold key constants and methods in the `globalConstants.py` file.

Any methods used had to come associated with doc-strings as well.

Task 2.1 - Lab Frame Simulation

Overview

This project simulates a superconducting circuit with two modes (a "memory" mode and a "buffer" mode) subject to parametric drives. This is solely for the lab-frame.

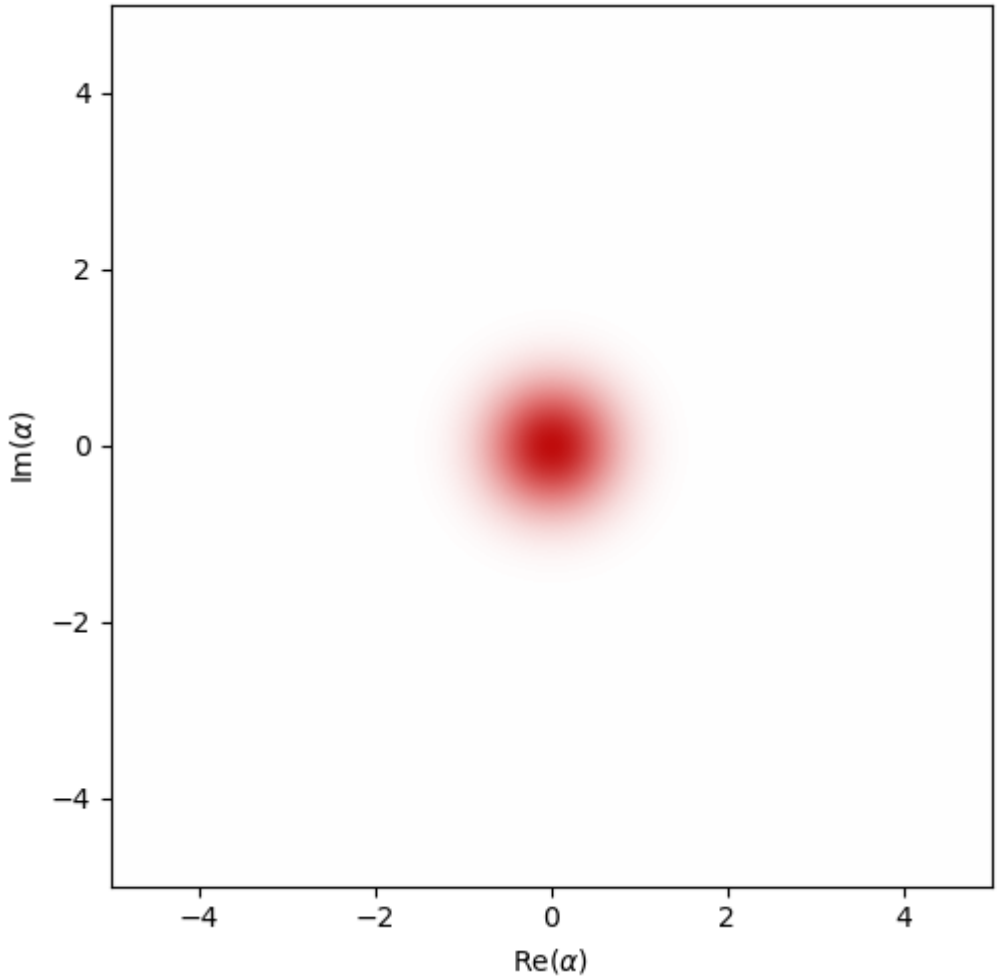
Math

- **Modes and Frequencies:**
- Two modes with bare resonant frequencies ω_{a_0} and ω_{b_0} are defined. The AC Stark shifts δ_a and δ_b are computed iteratively to update these frequencies.
- **Driving Terms:** A flux pump with amplitude ϵ_p and a buffer drive with amplitude ϵ_d modulate the system. The flux pump produces an Autler–Townes Splitting term with sinusoidal modulation, while the buffer drive acts directly on the buffer mode. -
- **Hamiltonians:** :
    1. **Effective Lab–Frame Hamiltonian H_{lab}:** Incorporates the bare Hamiltonian, ATS term (with sine and cosine functions of the phase operator), and the buffer drive.

Implementation

1. **Parameter and Operator Definitions:** - Define units, system parameters (frequencies, coupling strengths, and loss rates). - Compute the AC Stark shifts iteratively. - Construct the Fock space and corresponding annihilation operators for each mode.

2. **Hamiltonian Construction:**
   - **Effective Lab–Frame Hamiltonian** (H_{lab})
     - Defined by the bare mode energies, the ATS driving term modulated by the flux pump, and the buffer drive.
3. **Simulation:** - Define collapse operators to model dissipation in both modes. - Initialize the system in the vacuum state. - Use `dq.mesolve` to simulate the time evolution under both Hamiltonians. - Visualize the state dynamics using Wigner function plots.



Wigner probability distribution of the lab-frame state

Task 2.2 - Rotating Frame Simulation

Overview

This section focuses exclusively on the implementation and simulation of the **rotated–displaced Hamiltonian**. In this approach, the Hamiltonian is transformed into a frame that both rotates and displaces the original operators. This transformation simplifies the description of the driven system by absorbing part of the drive into the new frame, and it introduces additional correction terms.

Math

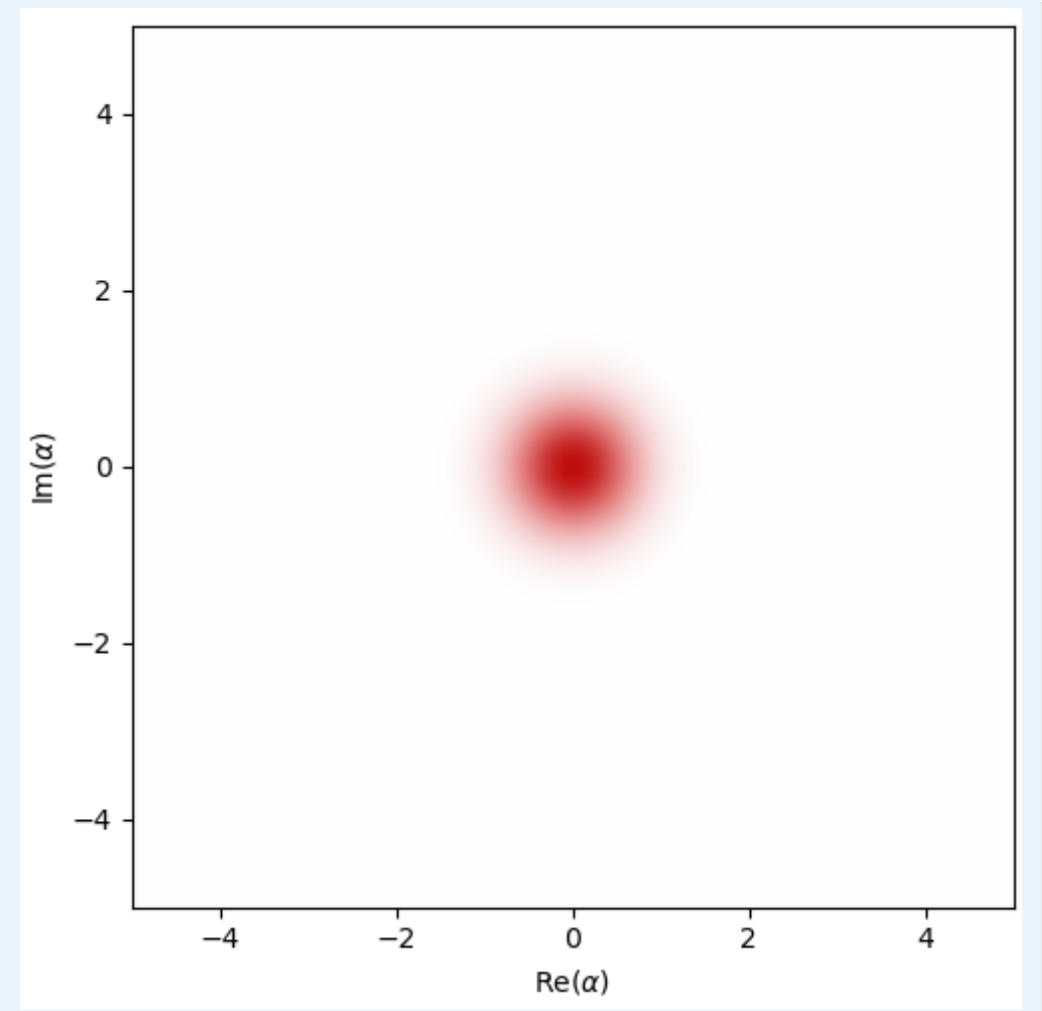
- **Displacement Fields:** The displacement for each mode is defined as a combination of two frequency components: $\alpha(t) = \alpha_1 e^{-i\omega_p t} + \alpha_2 e^{i\omega_p t}, \quad \beta(t) = \beta_1 e^{-i\omega_p t} + \beta_2 e^{i\omega_p t}$ where $\alpha_{1,2}$ and $\beta_{1,2}$ are determined by the system parameters and the flux pump amplitude.
- **Rotated–Displaced Operators:** The operators in the new frame are defined as: $a_f = a\, e^{-i\omega_{\mathrm{rot},a} t} + \alpha(t)\, I, \quad b_f = b\, e^{-i\omega_{\mathrm{rot},b} t} + \beta(t)\, I$, where I is the identity operator and $\omega_{\mathrm{rot},a}$ and $\omega_{\mathrm{rot},b}$ are the chosen rotation frequencies.
- **Hamiltonian Contributions:** The total rotated–displaced Hamiltonian H_{rd} consists of several parts:
  - **Linear Term** (h_0): Represents the bare mode energies in the new frame.
  - **Rotating Frame Correction** (h_{rot}): Compensates for the energy offset due to the frame rotation.
  - **Displacement Derivative Correction** (h_{dis}): Arises from the time derivative of the displacement fields.
  - **ATS Term** (h_{ats}): Includes a Taylor expansion of the sinusoidal modulation (due to the flux pump) that produces the Autler–Townes Splitting.
  - **Buffer Drive** $(h_{\mathrm{buffer\ drive}})$: The drive acting on the buffer mode appears non-rotating in this frame.

Implementation

The rotated–displaced Hamiltonian is implemented as a time-dependent function `hamiltonian_rotating_displaced(t)`.
Key steps include:

1. **Computing Displacement Fields:** The fields $\alpha(t)$ and $\beta(t)$ and their time derivatives ($d\alpha(t)$ and $d\beta(t)$) are computed using the defined frequency ω_p.
2. **Constructing Rotated–Displaced Operators:** The function `op_rot_displaced` is used to generate the time-dependent operators a_f, $a_f^\dagger$, b_f, and $b_f^\dagger$.
3. **Building the Hamiltonian:** Each term is calculated:
   - h_0**:** The energy of the modes in the new frame.
   - h_{rot}**:** The correction due to the chosen rotating frame frequencies.
   - h_{dis}**:** The correction arising from the time derivatives of the displacement fields.
   - h_{ats}**:** Constructed using a Taylor series expansion of the sine and cosine functions to approximate the ATS term.
   - $h_{\mathrm{buffer\ drive}}$**:** The drive term on the buffer mode, adjusted for the rotated frame.
4. **Time-Callable Function:** The complete Hamiltonian is wrapped using `dq.timecallable`, making it ready for time-dependent simulation.

5. **Simulation:** Using `dq.mesolve`, the system's evolution under H_{rd} is computed. The simulation incorporates collapse operators to model dissipation in both modes.



Wigner probability distribution of the rotating-frame state

Task 2.3 - Comparison and Fidelity Analysis

Overview

This section details the comparison between the effective lab–frame Hamiltonian model and the full rotated–displaced Hamiltonian model by evaluating the fidelity between their resulting quantum states over time. The fidelity serves as a quantitative measure of how closely the effective model reproduces the dynamics of the full model.

Math

- **Fidelity Definition:** The fidelity $F(\rho, \sigma)$ between two quantum states ρ and σ is defined as:

$$F(\rho,\sigma) = \left(\mathrm{Tr}\sqrt{\sqrt{\rho}\,\sigma\,\sqrt{\rho}}\right)^2.$$

A fidelity of 1 indicates that the states are identical, while lower values indicate deviations between the models.

- **State Evolution:** Both the effective lab–frame Hamiltonian H_{lab} and the rotated–displaced Hamiltonian H_{rd} are simulated over the same time grid. At each time point, the fidelity between the state produced by the effective model and that produced by the full model is calculated.

Implementation

1. **Simulating Both Models:**
   - The effective lab–frame Hamiltonian is defined as `H_lab` and simulated using `dq.mesolve`.
   - The rotated–displaced Hamiltonian is defined as `H_rd` and similarly simulated with `dq.mesolve`.
   - Both simulations use the same initial state (typically the vacuum state) and identical time grids.
2. **Fidelity Calculation:**
   - For each time step, the fidelity is computed using `dq.fidelity(state_lab, state_rd)`, where `state_lab` is the state from the effective model and `state_rd` is the state from the full rotated–displaced model.
   - The fidelity values are stored in an array for subsequent analysis and visualization.
3. **Visualization:**
   - A plot of fidelity versus time is generated using matplotlib. This plot provides a visual representation of how well the effective model approximates the full dynamics.
   - In practice, a fidelity close to 1 over the simulation time indicates that the effective model captures the essential physics, while deviations reveal discrepancies likely due to higher-order corrections or transient effects not captured by the effective approximation.

Task 2.4 - Optimal Control for Cat State Preparation

Overview

This section describes the optimal control strategy used to steer the dynamics of the rotated–displaced Hamiltonian toward a target cat state. The goal is to find control scaling factors that modify the ATS (flux pump) and buffer drive amplitudes so that the final state of the system closely matches the desired cat state.

Math

- **Control Parameters:** Two control parameters are introduced: - k_p: Scaling factor for the ATS drive amplitude. - k_d: Scaling factor for the buffer drive amplitude.
    - **Modified Hamiltonian:** The original rotated–displaced Hamiltonian is modified by these factors. The effective flux pump becomes:

      $$\epsilon_{\text{eff}}(t) = k_p\,\epsilon_p\cos(\omega_p t),$$

      and the buffer drive is scaled as:

      $$\epsilon_{d,\text{eff}} = k_d\,\epsilon_d.$$

      These modifications directly affect the ATS term and the buffer drive term in the Hamiltonian.
    - **Objective Function:** The control objective is to maximize the fidelity between the final state at the end of the simulation and a pre-defined target cat state. The cost function to be minimized is:

      $$\text{cost} = 1 - F(\rho_{\text{final}}, \psi_{\text{target}}),$$

      where $F(\cdot,\cdot)$ is the fidelity measure, and ψ_{target} is the desired cat state. # Implementation

1. **Hamiltonian Modification:** The function `hamiltonian_rd_opt(t, control)` returns the full rotated–displaced Hamiltonian with the ATS and buffer drive terms scaled by k_p and k_d, respectively.
2. **Simulation Wrapper:** The function `simulate_rd(control, tgrid=tsave)` creates a time-callable Hamiltonian with the given control parameters and runs the simulation using `dq.mesolve`. The dynamics are obtained over the defined time grid.
3. **Objective Function:** The `objective(control)` function runs a simulation with the given control parameters and computes the infidelity $1 - F$ between the final state and the target cat state. This function is designed for use with an optimization routine.
4. **Optimization Routine:** Using the Nelder-Mead algorithm (a gradient-free method), the control parameters are optimized starting from an initial guess (typically $[1.0, 1.0]$). The optimization minimizes the cost function, thereby maximizing the fidelity with the target state.
5. **Final Simulation and Visualization:** With the optimal control parameters found, a full simulation is run. The fidelity between the state at each time and the target cat state is computed and plotted. In addition, the evolution of the state is visualized using a Wigner function animation for mode a.