FYP Report

Lin Zhonglin

February 27, 2024

Contents

1 Overview

1. Background:
2. Key part of my work: (depends on how far I go)
   (a) The optimised pulse for a particular state preparation
   (b) how it differs from other pulse (Chiyuan's alternate pulse way)
   (c) Go from model Hamiltonian to physical Hamiltonain, discuss the approximations' effect, compare fideilty

TODO:

1. Go through the Hamiltonian derivation → for comparison of fidelity later
2. does Qutip considers the cavity Hilbert space truncation discussed below?

Issues to be dealt with:

1. local minima / maxima in optimization algorithm
2. Reinhold: Optimizing a full unitary operation is sensible for qubits, or systems of a few qubits, but in the cavity manipulations to be performed in the following sections, this is not usually a sensible operation, since doing operations involving states at the border of the truncation (i.e. the maximum photon number state considered in the simulation), would typically require a larger truncation to simulate accurately (driving $|n_{ph}\rangle$) will almost always in part bring us partially to $|n_{ph}+1\rangle$.

2 Cheatsheet

3 Background, theory and motivation

content: introduce the idea of bosonic-modes-encoded qubits, give a simple example; motivate Bosonic-modes-encoded qubits -> that one physical device now can encode multiple qubits -> this also makes error correction more efficient -> easier to control multiple qubits

introduce cavity-transmon system motivate for cavity-DQD system

theory of cavity-DQD: original Hamiltonians -> rotation, approximaition, dispersive regime, etc... -> the Hamiltonian I used for pulse optimization

briefly bridge to and cover the overview of content of this report: 1. basic theory on optimization, QuTip, GRAPE, CRAB, etc... 2. pulse optimization for state preparation 3. compare fidelity from approximate Hamiltonian with fidelity from original Hamiltonian 4. possible future work: gate design, model noise

4 Hamiltonians

- How to verify that the Hamiltonian does describe thephysical system?
  (Reinhold thesis:) Initially derived in an NMR context, GRAPE, and related methods, have found use in a wide variety of quantum systems and applications (Dolde et al., 2014; Anderson et al., 2015). Since GRAPE crucially depends on the model of the system, its successful application is powerful evidence that the Hamiltonian used accurately captures the system dynamics over a broad range of driving conditions.

4.1 Transmon-cavity system

5 Reinhold thesis -> GRAPE

We assume we have a quantum system, whose Hamiltonian can be written in the following form:

$$H(\vec{\epsilon}(t)) = H_0 + \sum_{k=1}^{m} \epsilon_k(t) H_k \tag{1}$$

Goal: Find optimal control field $\vec{\epsilon}(t)$ that optimizes some function $f(\vec{\epsilon}(t)) = f(H(\vec{\epsilon}(t)))$.

1. To optimize a large number of parameters, we need:
   (a) an efficient means of calculating the gradients of cost function w.r.t parameters
   (b) sub-optial local minima are sufficiently unlikely or close to global minima
2. Given methods to calculate cost function and gradients, we have two main classes of algorithm for performing optimization:
   - line-search methods: one alternates between picking a direction in parameter space, radiating from our current point, and subsequently performing a 1-d minimization protocol to find the minimum along this line.
     → basic gradient descent: choose direction as gradient at the point.
     → Newton method: choose direction using both gradient and Hessian matrix at the point. → quasi-Newton, L-BFGS
   - trust-region methods
3. calcualte gradient of $f(\vec{\epsilon}(t))$ with respect to $\vec{\epsilon}(t)$
   (a) For an analytical function $f(\vec{\epsilon})$, gradient is given by $\nabla f(\vec{\epsilon}) = \sum_{i=1}^{m} \frac{\partial f(\vec{\epsilon})}{\partial \epsilon_i} \hat{\epsilon}_i$
   (b) For discrete $\vec{\epsilon}$, gradient of $f(\vec{\epsilon})$ is given by the finite difference method.
   (c) approximate the gradient: skipping here.
   (d) When calculating / approximating gradient is too expensive, cosider optimization algorithm other than gradient descent.
4. take a step (step size as a parameter) in the direction.
   step size too small: optimization takes long
   step size too large: may never reach true optima
5. calcualte gradient again
6. repeat until convergence
7. issue of local minima / maxima:
   - Optimize for infidelity (1 - fidelity) instead of fidelity. Because infidelity is lower bounded by 0. Terminate optimization when infidelity reaches 0 or close to 0.

5.1 Figure of merit, cost function

state transfer case single-state state transfer:

$$
\begin{aligned}
f(\vec{\epsilon}(t)) &= \mathcal{F}(\vec{\epsilon}(t)) \quad \text{state fidelity} \\
&\equiv \left|\langle\psi_{targ}|\, U(T,\vec{\epsilon}(t))\, |\psi_{init}\rangle\right|^2 \\
&= \left|\langle\psi_{targ}|\, \mathcal{T}\exp\left\{-\frac{i}{\hbar}\int_0^T dt H(\vec{\epsilon}(t))\right\} |\psi_{init}\rangle\right|^2 \\
&\quad \text{make the problem numerical, make } \vec{\epsilon}(t) \text{ a piece-wise constant function with } N = T/\delta t, \\
&\quad \delta t \text{ is a parameter, usually set to time resolution of AWG.} \\
&= \left|\langle\psi_{targ}|\, U_N U_{N-1} ... U_1\, |\psi_{init}\rangle\right|^2, \quad \text{where, } U_k = \exp\left\{\frac{i\delta t}{\hbar} H(\vec{\epsilon}(k\delta t))\right\}
\end{aligned}
$$

multi-state state transfer:

$$
\begin{aligned}
f(\vec{\epsilon}(t)) &= \mathcal{F}(\vec{\epsilon}(t)) \quad \text{Fidelity} \\
&\equiv \begin{cases} \left|\sum_k \left\langle \psi_{targ}^{(k)} \right| U(T,\vec{\epsilon}(t)) \left|\psi_{init}^{(k)}\right\rangle\right|^2 & \text{coherent} \\ \sum_k \left|\left\langle \psi_{targ}^{(k)} \right| U(T,\vec{\epsilon}(t)) \left|\psi_{init}^{(k)}\right\rangle\right|^2 & \text{inoherent} \end{cases} \\
&\quad \text{for coherent case with number of state = dimension of Hilbert space,} \\
&= \left|\mathrm{Tr}\left[U(T,\vec{\epsilon}(t)) U_{targ}^{\dagger}\right]\right|^2 \quad \text{unitary fidelity}
\end{aligned}
$$

open system case:

5.2 Pulse constraints

1. constraints come from AWG(Arbitrary Wave Generator)'s amplitude and bandwidth constraints → consider adding a set of constraint terms to the cost function

$$f(\vec{\epsilon}(t)) = \mathcal{F}(\vec{\epsilon}(t)) + \sum_i \lambda_i g_i(\vec{\epsilon})$$

2. pulse amplitude: the output power of our AWG is limited, and thus to be feasible, we need $\epsilon(t) \epsilon_{max}$ for all t. → hard cutoff / soft cutoff

   (a) Employ an optimization algorithm which naturally allows for such constraints → see K-BFGS-B in scipy.optimize

   (b) Parametrise optimization problen

$$
\begin{aligned}
&\overset{\text{maximize}}{\vec{\epsilon}(t)} \quad \mathcal{F}(\vec{\epsilon}(t)) \rightarrow \overset{\text{maximize}}{\vec{x}} \quad \mathcal{F}(\vec{\epsilon}(\vec{x})) \\
&\text{where, } \epsilon_k = \epsilon_{max} \tanh(x_k)
\end{aligned}
$$

3. pulse bandwidth:

   - soft cutoff
   - hard cutoff
   - linear frequency-dependent penalty

5.3 limit the intermediate photon number

Since computer memory is finite, we are forced to choose a photon number truncation nph such that the operator a becomes a $n_{ph} \times n_{ph}$ matrix. When we do this, we are in effect replacing our infinite-dimensional oscillator with a finite-dimensional qudit. This replacement is only valid if all of the system dynamics relevant for the desired state transfers occurs within the $\{|0\rangle, \cdots, |n_{ph}-1\rangle\}$ subspace. For generic applied drives this is not the case. In order to enforce this property, we modify the optimization problem to find a solution which operates identically under several different values of n_{ph}. Writing the fidelity as computed with a truncation nph as $F_{n_{ph}}$, we have:

$$\overset{\text{maximize}}{\vec{\epsilon}} \left(\sum_k F_{n_{ph}+k}(\epsilon(t)) \right) - \left(\sum_i \lambda_i g_i(\epsilon(t)) \right) \tag{2}$$

- To enforce that the behavior is identical in the different truncations, we add the penalty term:

$$g_{\text{discrepancy}}\left(\boldsymbol{\epsilon}(\boldsymbol{t})\right) = \sum_{k_1 \neq k_2} \left(\mathcal{F}_{n_{\text{ph}}+k_1}(\boldsymbol{\epsilon}(t)) - \mathcal{F}_{n_{\text{ph}}+k_2}(\boldsymbol{\epsilon}(t)) \right)^2$$

  I think it's making the cost function symmetrical w.r.t. n_{ph}.

- A more recently developed, and more direct, method is to add a penalty term for any occupation of the final photon state ($|n_{ph}-1\rangle$) in the truncated Hilbert space at any time:

$$g_{\text{trajectory}} = \sum_{k=1}^{N} \left| \left\langle n_{\text{ph}} - 1 \mid \psi_{\text{fwd}}^{(k)} \right\rangle \right|^2$$

  Not sure what this is doing, putting here for possible future use.

Not sure whether this is implemented in QuTip.

5.4 Troubleshooting optimization convergence

Possible issue:

-

Check list:

1. Check that the time given $T = N\delta t$ is appropriate.
   - Specified in units that are consistent with the units specifying the Hamiltonian. For instance, if the Hamiltonian is specified in GHz, then the time step should be in units of ns.
   - N value appropriate
2. check constraints
   - Completely remove all constraints and penalties, and make sure that the algorithm works in this context before re-introducing them
3. check algorithm
   - check termination conditions
     Gradient based search algorithms usually have termination conditions specified in terms of the norm of the gradient. It is often necessary to lower the gradient norm threshold for termination to ensure that it does not give up -> gtol in scipy.minimize
4. check ionitial guess
   - Avoid special initial guesses, this depends on the algorithm. For gradient-based algo, make sure initial gradient is not vanishing.
5. miscellaneous checks
   - Truncation of cavity-state Hilbert space large enough?

5.4.1 Simple diagnostic pulses

- For instance, in the transmon-cavity system, a simple diagnostic pulse is one which produces a single photon state:

$$|0\rangle \otimes |g\rangle \longrightarrow |1\rangle \otimes |g\rangle$$

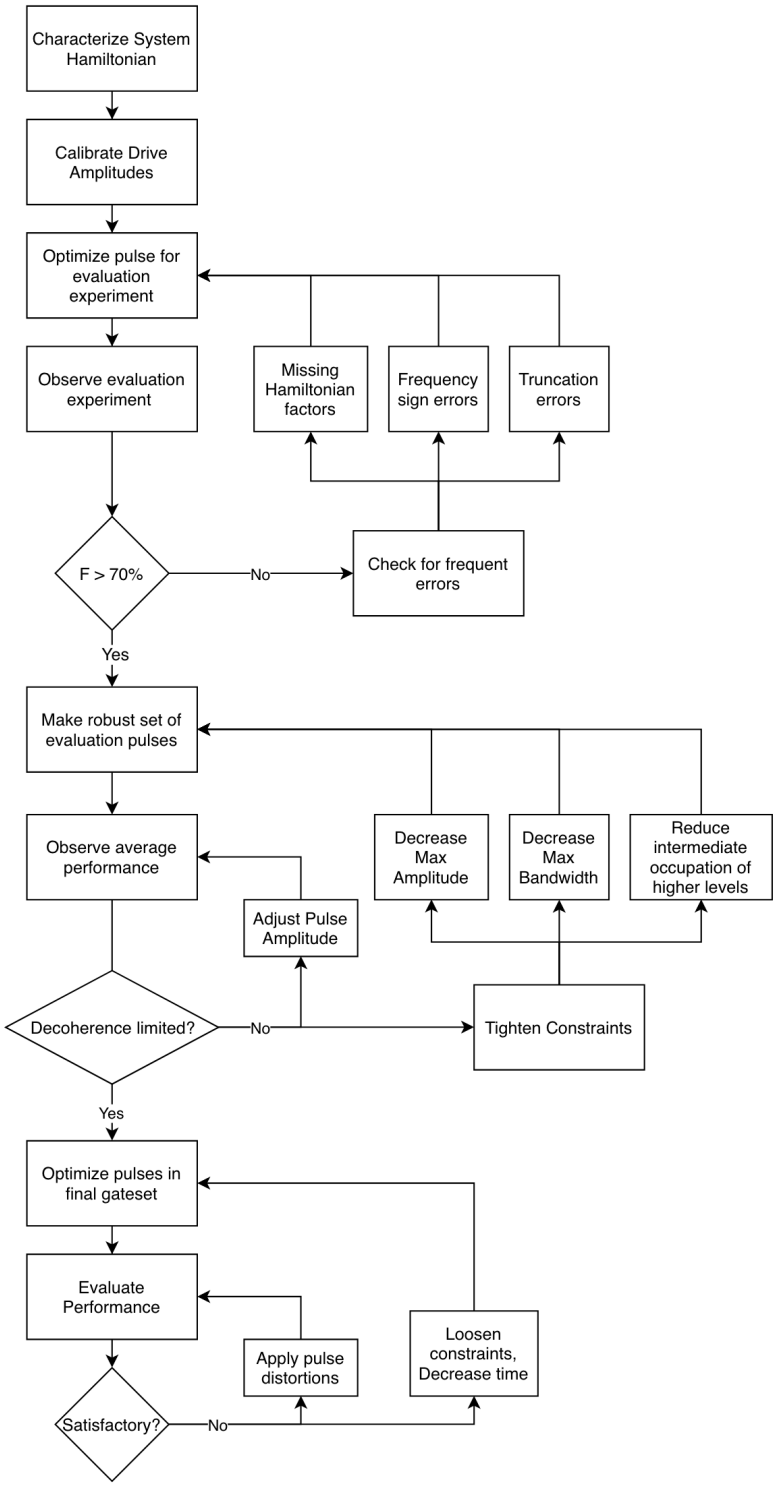

Figure 1: Flowchart for optimal control system

5.5 Robust control

6 QuTip

6.1 QuTip quantum optimal control

- Quantum control
  - prepare some specific state
  - effect some state-to-state transfer
  - effect some transformation (or gate) on a quantum system

6.1.1 GRAPE in QuTip

GRAPE algorithm in QuTip:

1. overview:

   - The combined Hamiltonian is approximated as: $H(t) \approx H(t_k) = H_0 + \sum_{j=1}^{M} u_{jk} H_j$
     where, k is a timeslot index, j is the control index, and M is the number of controls. Hence t_k is the evolution time at the start of the timeslot, and u_{jk} is the amplitude of control j throughout timeslot k. Number of time steps $N = T/\delta t$.

   - The time evolution operator, or propagator, within the timeslot can then be calculated as: $U_k := e^{-iH(t_k)\Delta t_k}$
     where, Δt_k is the duration of the timeslot.
     The evolution up to (and including) any timeslot k (including the full evolution k=M) can the be calculated as $U(t_k) := U_k U_{k-1} \cdots U_1 U_0$
   - figure of merit: See Reinhold thesis section
   - optimization algorithm:
     (a) There are now $N \times M$ variables to minimize the figure of merit. The problem becomes a finite multi-variable optimiz
     (b) Optimization algorithm: (see Reinhold thesis section for details)
     Gradient and quasi-Newton method (considers Hessian) both implemented.
     (The default method in the QuTiP Qtrl GRAPE implementation is the L-BFGS-B method in Scipy)
     (c) calculating / approximating the gradient: see Reinhold thesis section

6.1.2 CRAB in QuTip

CRAB-Chopped RAdom Basis algorithm in QuTip:

1. rough overview:

2. Since the pulse complexity is usually very low, it is sufficient to transform the optimal control problem to a few parameter search by introducing a physically motivated function basis that builds up the pulse. Compared to the number of time slices needed to accurately simulate quantum dynamics (often equals basis dimension for Gradient based algorithms), this number is lower by orders of magnitude, allowing CRAB to efficiently optimize smooth pulses with realistic experimental constraints.

3. choosing functional basis:

   - Consider a priori knowledge of the system
     → such as symmetries, magnitudes of scales,...
     → integrate experimental constraints such as maximum frequencies allowed, maximum amplitude, smooth ramping up and down of the pulse ...
   - Consider expected solution (e.g. sign, smoothness, bang-bang behavior, singularities, maximum excursion or rate of change,....).

4. where CRAB differs from GRAPE:
    - Optimized pulse from CRAB is a smooth function.
    - CRAB optimizes pulse function basis coefficient instead of amplitude of poulse at time slices.
    - CRAB considers time slices only when calculating fidelity for a set of function basis coefficients.
5. "dressed" CRAB: escaping local optima

6.2 QuTip optimal control implementation

- dressed CRAB not implemented yet

Qtrl code:

- Qtrl uses the Scipy optimize functions to perform the multi-variable optimisation, typically the L-BFGS-B method for GRAPE and Nelder-Mead for CRAB.
- The Qtrl code is organised in a hierarchical object model

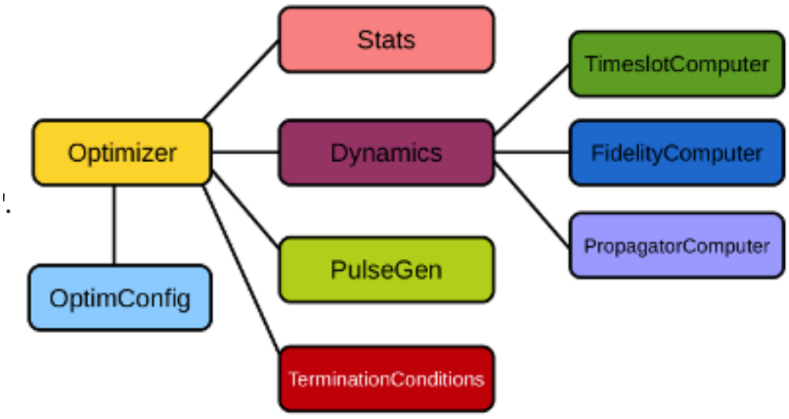

Figure 2: Qtrl code structure

Code template for a customized pulse optimization:

Miscellanesou:

- CRAB: how to set function basis used for optimization?
- how and what exactly is initial guess pulse set using p_type?

7 Diagnostic pulse optimization

8 Other pulse optimization software

- krotov
- sequencing

9 optimization in Scipy

scipy.minimize

10 Physical realizability

go back the Hamiltonian massagiing: Perturbation, frame change, throwing away small cross terms, exact compare fidelity

11 Conclusion and future work