Direct Collocation for Quantum Optimal Control

Aaron Trowbridge[1], Aditya Bhardwaj[2], and Kevin He[2]

[1]Robotics Exploration Lab, Carnegie Mellon University
[2]Schuster Lab, Stanford University

Abstract

We present an adaptation of the *direct collocation* trajectory optimization method for solving problems in quantum optimal control (QOC). This approach addresses several limitations of standard methods, including the ability to solve minimum time problems, a crucial objective for realizing high-performance quantum computers. We demonstrate that this approach leads to improved performance on simulated systems as well as on nascent hardware devices, compared to other existing methods. To the best of our knowledge, this is the first time that direct collocation, which is commonplace in the field of robotic control, has been applied to QOC.

Contents

1 Introduction **1**
1.1 Types of Problems . . . 1

2 Background **2**
2.1 Gradient-Based Methods . . . 2
2.2 Gradient-Free Methods . . . 2

3 Direct Collocation **3**
3.1 Notation . . . 3
3.2 State Transfer Problems . . . 3
3.3 Free Time Problems . . . 3
3.4 Minimum Time Problems . . . 4
3.5 Summary . . . 4

4 Quantum Control Problems **4**
4.1 Isomorphic Formulation . . . 5
4.2 Padé Integrator Dynamics . . . 5
4.3 Base Problem Definition . . . 6

5 Results **6**
5.1 Solution Examples . . . 6
5.2 Comparison to Other Methods . . . 6
5.3 Hardware Results . . . 6

1 Introduction

Controlling quantum systems is in principle the problem of optimizing over the space of quantum state trajectories given the ability to control, over an interval of time, certain terms in the time-dependent Hamiltonian describing the system. We will consider n-dimensional quantum systems with time-dependent Hamiltonians of the form

$$H(\mathbf{a}(t), t) = H_0 + \sum_i a^i(t) H_i, \tag{1}$$

where $t \in [0, T]$, $\mathbf{a}(t) \in \mathbb{R}^m$ is the control trajectory, referred to as the *pulse*, H_0 is the system's *drift* term, and H_i are the *drive* terms.

The field of optimal control, which has its origins in space systems and robotics, has produced, especially in recent years, a large body of sophisticated methods for solving control problems fundamentally identical to the problems posed in QOC. However, many of these methods have not seemed to have been adopted by those (primarily physicists) working on control problems for quantum systems. This work aims to bridge the gap between robotic control and quantum control, and in so doing to provide a new perspective on the problem of quantum control.

1.1 Types of Problems

There are typically three flavors of QOC problems, corresponding to three types of states:

1. *Pure quantum states* $\psi(t)$: Minimize the infidelity between the final state $\psi(T)$ and the goal state ψ_{goal}:

$$\ell(\psi(T)) = 1 - |\langle \psi(T) | \psi_{\text{goal}} \rangle|^2 \tag{2}$$

   Where $\psi(0) = \psi_{\text{init}}$ and $\psi(t)$ satisfies the Schröedinger equation $\dot{\psi} = -iH(\mathbf{a}(t), t)\psi$.

2. *Unitary operators* $U(t)$: Minimize the infidelity or trace distance between the final unitary $U(T)$ and the desired final unitary U_{goal}: respectively,

$$\ell(U(T)) = 1 - \left(\text{tr} \sqrt{U(T)^\dagger U_{\text{goal}}} \right)^2 \quad \text{or} \quad \ell(U(T)) = \frac{1}{2} \| U(T) - U_{\text{goal}} \|_{\text{tr}}, \tag{3}$$

   Here $U(0) = I$ and $U(t)$ also satisfies the Schröedinger equation $\dot{U} = -iH(\mathbf{a}(t), t)U$.

3. *Mixed quantum states* or *density matrices* $\rho(t)$: Minimize the infidelity or trace distance between the final state $\rho(T)$ and the goal state ρ_{goal}: respectively,

$$\ell(\rho(T)) = 1 - \left(\text{tr} \sqrt{\rho(T) \rho_{\text{goal}}} \right)^2 \quad \text{or} \quad \ell(\rho(T)) = \frac{1}{2} \| \rho(T) - \rho_{\text{goal}} \|_{\text{tr}}. \tag{4}$$

   Here $\rho(0) = \rho_{\text{init}}$ and $\rho(t)$ satisfies the von Neumann equation $\dot{\rho} = -i[H(\mathbf{a}(t), t), \rho]$.

Remark: In this paper, for simplicity, we will focus on the first type of problem, and just consider the case of optimizing a single pure quantum state $\psi(t)$ — all methods discussed apply to the other two types of problems as well.

2 Background

In practice, to do any type of trajectory optimization it is necessary to discretize the time interval $[0, T]$ into N time steps of size Δt, the states and controls at each time step are denoted ψ_k and $\mathbf{a}_k$, respectively. Typically the following optimization is then solved to find the optimal pulse:

$$\underset{\mathbf{a}_{1:T-1}}{\text{minimize}} \quad J(\mathbf{a}_{1:T-1}) = \ell(\psi_N(\mathbf{a}_{1:T-1})), \tag{5}$$

$$\text{subject to} \quad c(\mathbf{a}_k) \leq 0, \quad \forall k \tag{6}$$

where

$$\psi_{k+1} = \exp\left(-iH(\mathbf{a}_k)\Delta t\right)\, \psi_k \tag{7}$$

and $\psi_1 = \psi_{\text{init}}$. Here, during the time interval Δt the controls $\mathbf{a}_k$ are held constant, which is referred to as a *zero-order hold*, and allows the dynamics to exactly take the form above.

Current approaches for solving this problem fall into two categories: *gradient-based* methods and *basis function* methods. Both of these are *indirect* methods, as they optimize only over the controls $\mathbf{a}_{1:T-1}$, as opposed to considering both the states and controls as decision variables in what is known as direct methods. The advantages of direct methods, over indirect methods, are the main point of this paper. Before getting to the details of direct methods we will first introduce these two current mainstream approaches to solving QOC problems.

2.1 Gradient-Based Methods

This method involves initializing the controls $\mathbf{a}_{1:T-1}$ to some initial guess, and then iteratively updating them using a gradient descent algorithm. At each iteration is necessary to *rollout* the system with the given controls (this is also referred to as a *shooting method*) to get ψ_N and evaluate the cost function $J(\mathbf{a})$ to be able to take a gradient of it and update the controls:

$$\mathbf{a} \leftarrow \mathbf{a} - \beta \nabla J(\mathbf{a}) \tag{8}$$

There are efficient ways to compute this gradient, but the approach is still limited in other ways. These include the fact that the solution is dependent on the initial guess, gradient-based methods are in general prone to falling into local minima, and it is not possible to enforce constraints on the states.

The most popular gradient-based method is known as GRAPE (GRadient Ascent Pulse Engineering), which is available through the popular QuTiP python library. Q-CTRL also implements its own gradient-based optimization tool, which is virtually identical, is the industry standard, and is what we compare our results to in this paper.

2.2 Gradient-Free Methods

For this class of approaches, the key idea is to reduce the number of decision variables enough so that gradient-free optimization algorithms—e.g. the Nelder-Mead downhill simplex method—can be utilized. In the case of the popular CRAB algorithm (also implemented in QuTiP and compared against later on) this is accomplished by parameterizing the pulse by a set of basis functions. The CRAB algorithm, specifically, utilizes a Fourier basis parameterization for each pulse component:

$$a(t) = 1 + \frac{\sum_{n=1}^{N_c} b_n \sin(\omega_n t) + c_n \cos(\omega_n t)}{\lambda(t)}. \tag{9}$$

Where N_c is the number of terms to *chop* the series off at and $\lambda(t)$ is chosen to enforce the boundary conditions. Standard gradient-free methods can then be used to solve the new optimization problem:

$$\underset{b_{1:N_c}, c_{1:N_c}, \omega_{1:N_c}}{\text{minimize}} \quad J(b_{1:N_c}, c_{1:N_c}, \omega_{1:N_c}). \tag{10}$$

3 Direct Collocation

In this section we describe *direct collocation*, a direct approach to solving trajectory optimization problems. At a high level direct collocation allows us to overcome the limitations of indirect methods by including the states as decision variables, along side the controls, at each time-step; respectively denoted x_k and u_k. In this formulation the dynamics—which are implicit in indirect methods, coming in during the rollout phase—are enforced as constraints between *knot points* $z_k = (x_k, u_k)^\top$.

With the dynamics enforced as constraints, numerical non-linear solvers are free to violate these constraints during intermediate points of the optimization routine, en route to satisfying them at convergence. This capability, along with the ability to enforce constraints on the state variables, lend theoretical and empirical evidence towards the superiority of direct methods over indirect.

3.1 Notation

In this section we will adopt the following (mostly) conventional control theory notation to focus on the structure of the problems — in the following section we will apply these methods to quantum systems.

- x: represents the states, where $x_k \in \mathbb{R}^n$
- u: represents the controls, where $u_k \in \mathbb{R}^m$
- z: represents a knot point, where $z_k = (x_k, u_k)^\top \in \mathbb{R}^{n+m}$

3.2 State Transfer Problems

A core problem in trajectory optimization is the state transfer from problem, where we want to begin in an initial state, i.e. $x_1 = x_{\text{goal}}$, and find a control sequence $u_{1:N-1}$ such that x_N minimizes a loss ℓ relative to the goal state x_{goal}, . We will enforce the dynamics constraints implicitly, i.e. as $f(x_{k+1}, x_k, u_k) = 0$, for reasons of generality here, but as elucidated later, to increase computational efficiency. We can write this problem like so:

$$\begin{aligned} \underset{x,u}{\text{minimize}} \quad & \ell(x_N) && (11)\\ \text{subject to} \quad & f(x_{k+1}, x_k, u_k) = 0 && (12)\\ & x_1 = x_{\text{init}} && (13) \end{aligned}$$

3.3 Free Time Problems

If we do not know the optimal time interval in which to solve our problem, this framework allows us to easily modify our the previous problem so that the solver can find the optimal time on its own. This is accomplished by including making each time step Δt_k a decision variable and including it in the dynamics constraint:

$$\begin{aligned} \underset{x,u,\Delta t}{\text{minimize}} \quad & \ell(x_N) && (14)\\ \text{subject to} \quad & f(x_{k+1}, x_k, u_k, \Delta t_k) = 0 && (15)\\ & \Delta t^{\min} < \Delta t_k < \Delta t^{\max} && (16)\\ & x_1 = x_{\text{init}} && (17) \end{aligned}$$

Here the time steps, Δt_k are constrained to be within some reasonable bounds, $0 < \Delta t^{\min} < \Delta t^{\max}$, in order to prevent the solver from taking advantage of negative values and discretization error.

3.4 Minimum Time Problems

Given a high fidelity, *nominal* solution, $(x_{1:N}^{\text{nominal}}, u_{1:N-1}^{\text{nominal}}, \Delta t_{1-N-1})$, to the free time problem, we can warm start a new optimization problem where we seek to minimize the total time. This problem can be set up as follows:

$$\begin{aligned}
\underset{x,u,\Delta t}{\text{minimize}} \quad & \sum_{k=1}^{N-1} \Delta t_k && (18)\\
\text{subject to} \quad & f(x_{k+1}, x_k, u_k, \Delta t_k) = 0 && (19)\\
& \Delta t^{\min} < \Delta t_k < \Delta t^{\max} && (20)\\
& x_1 = x_{\text{init}} && (21)\\
& x_N = x_N^{\text{nominal}} && (22)
\end{aligned}$$

Here we constrain the final state x_N to equal the final state of the nominal trajectory—which can not be done with indirect methods—and forces the solver to find solutions that minimize the time objective without sacrificing the $\ell(x_T)$ objective.

3.5 Summary

Direct collocation allows us to solve QOC type problems in a way that overcomes the shortcomings of available indirect methods. In this framework it is possible to solve a wide variety of trajectory optimization problems – e.g., for an arbitrary non-linear objective $J(z)$, where $z = z_{1:N}$ is the collection of knot points at each time step, non-linear (or linear) constraints, abstractly written as $c(z) \in \mathcal{Z}$, where $\mathcal{Z}$ is the set of admissable values, we can write the following generalized direct collocation optimization problem:

$$\begin{aligned}
\underset{z}{\text{minimize}} \quad & J(z) && (23)\\
\text{subject to} \quad & f(z_k, z_{k+1}) = 0 && (24)\\
& c(z) \in \mathcal{Z} && (25)
\end{aligned}$$

Here $f(z_{k+1}, z_k) = 0$ is the dynamics constraints generalized to knot points – e.g, we might have $z_k = (x_k, u_k, \Delta t_k)^\top$ and $f(z_k, z_{k+1}) = f(x_{k+1}, x_k, u_k, \Delta t_k)$ along with $J(z) = J(x, u, \Delta t) = \ell(x_N)$. We write the problem in the form above, as it is the convention we have adopted in the corresponding software package: Pico.jl.

4 Quantum Control Problems

This section deals with formulating QOC problems as direct collocation trajectory optimization problems as described in the previous section. A simple form of these types of problems is as follows:

$$\begin{aligned}
\underset{\psi,\mathbf{a}}{\text{minimize}} \quad & J(\psi, \mathbf{a}) = \ell(\psi_N) = 1 - |\langle \psi_N | \psi_{\text{goal}} \rangle|^2 && (26)\\
\text{subject to} \quad & f(\psi_{k+1}, \psi_k, \mathbf{a}_k) = 0 && (27)\\
& \psi_1 = \psi_{\text{init}} && (28)
\end{aligned}$$

To implement this problem we must deal with a handful of quantum specific issues. First we will cover how to convert to real values the complex valued states $\psi \in \mathbb{C}^n$ and Hamiltonians $H \in \mathbb{C}^{n \times n}$. Next, we will discuss a novel, efficient way to enforce the dynamics, which naively requires a potentially very high dimensional matrix exponential. Finally we will discuss a few tricks for achieving certain types of solutions, such as *bang-bang* solutions, *smooth* solutions, and *guard state* solutions.

4.1 Isomorphic Formulation

To move between complex valued quantum states and real valued problem variables we will utilize the following isomorphic representations for complex vectors and matrices. We will use *tildes* to represent isomorphic representations from here on out. For the quantum states we then have

$$\psi \in \mathbb{C}^n \quad \longrightarrow \quad \tilde{\psi} = \begin{pmatrix} \operatorname{Re}\psi \\ \operatorname{Im}\psi \end{pmatrix} \in \mathbb{R}^{2n}, \tag{29}$$

and for matrices we have

$$H \in \mathbb{C}^{n\times n} \quad \longrightarrow \quad \tilde{H} = \begin{pmatrix} \operatorname{Re}H & -\operatorname{Im}H \\ \operatorname{Im}H & \operatorname{Re}H \end{pmatrix} \in \mathbb{R}^{2n\times 2n}. \tag{30}$$

The Schröedinger dynamics involve an evaluation of $\exp(-iH)$, so it will behoove us to define the following function G:

$$G(H) = \widetilde{-iH} = \begin{pmatrix} \operatorname{Im}H & \operatorname{Re}H \\ -\operatorname{Re}H & \operatorname{Im}H \end{pmatrix} \tag{31}$$

Which, as can be easily shown, aligns with the previous definition. Also, G is a linear function of H and $\mathbf{a} \in \mathbb{R}^m$, so we can then write

$$G(\mathbf{a}) = G(H(\mathbf{a})) = G_0 + \sum_j a^j G_j, \tag{32}$$

where $G_j := G(H_j)$ mnemonically represents the *generator* of the dynamics.

4.2 Padé Integrator Dynamics

Our dynamics can be precisely written in the form

$$f(\tilde{\psi}_{t+1}, \tilde{\psi}_t, \mathbf{a}_t) = \tilde{\psi}_{t+1} - \exp\left(G(\mathbf{a}_t)\cdot\Delta t\right)\,\tilde{\psi}_t, \tag{33}$$

but this does not account for how matrix exponentials are actually computed numerically. A common approach is to use the Padé approximant for the exponential,

$$\exp(A) = \left(I - \frac{1}{2}A + \frac{1}{9}A^2 + \dots\right)^{-1}\left(I + \frac{1}{2}A + \frac{1}{9}A^2 + \dots\right), \tag{34}$$

and truncate the series at some order in the numerator and denominator.

We take advantage of this structure by recognizing that in our framework the leading matrix inverse is computationally expensive and not necessary to compute directly — we can simply shift it over; i.e., write

$$f(\tilde{\psi}_{t+1}, \tilde{\psi}_t, \mathbf{a}_t) = \left(I - \frac{\Delta t}{2}G + \frac{\Delta t^2}{9}G^2 + \dots\right)\tilde{\psi}_{t+1} - \left(I + \frac{\Delta t}{2}G + \frac{\Delta t^2}{9}G^2 + \dots\right)\tilde{\psi}_t. \tag{35}$$

Here, $G = G(\mathbf{a}_t)$.

Given we can assume our dynamics model is not perfect and that numerical imprecision will arise regardless, it prompts us to lean toward efficiency. With these considerations we define the second and fourth order *Padé integrators* as

$$\mathbf{P}^{(2)}(\tilde{\psi}_{t+1}, \tilde{\psi}_t, \mathbf{a}_t) = \left(I - \frac{\Delta t}{2}G(\mathbf{a}_t)\right)\tilde{\psi}_{t+1} - \left(I + \frac{\Delta t}{2}G(\mathbf{a}_t)\right)\tilde{\psi}_t \tag{36}$$

$$\mathbf{P}^{(4)}(\tilde{\psi}_{t+1}, \tilde{\psi}_t, \mathbf{a}_t) = \left(I - \frac{\Delta t}{2}G(\mathbf{a}_t) + \frac{\Delta t^2}{9}G(\mathbf{a}_t)^2\right)\tilde{\psi}_{t+1} - \left(I + \frac{\Delta t}{2}G(\mathbf{a}_t) + \frac{\Delta t^2}{9}G(\mathbf{a}_t)^2\right)\tilde{\psi}_t \tag{37}$$

4.3 Base Problem Definition

Considering hardware limitations, where the pulse amplitude is limited to a finite range and where we must also have $\mathbf{a}_1 = \mathbf{a}_{N-1} = 0$, as well as numerical conditioning—for which we add a quadratic regulariztion term on the controls $\mathbf{a}_{1:N-1}$—we can now write the following constrained, isomorphic optimization problem, which will serve as the base problem we can build upon to achieve specific flavors of solutions:

$$\begin{aligned}
\underset{\tilde{\psi},\mathbf{a}}{\text{minimize}} \quad & J(\tilde{\psi},\mathbf{a}) = \ell(\tilde{\psi}_N) + \sum_{k=1}^{N-1} \mathbf{a}_k^\top R \mathbf{a}_k && (38)\\
\text{subject to} \quad & \mathbf{P}^{(n)}(\tilde{\psi}_{k+1},\tilde{\psi}_k,\mathbf{a}_k) = 0 && (39)\\
& \tilde{\psi}_1 = \tilde{\psi}_{\text{init}} && (40)\\
& \left|a_k^j\right| \leq a_{\max}^j && (41)\\
& \mathbf{a}_1 = \mathbf{a}_{N-1} = \mathbf{0} && (42)
\end{aligned}$$

Remark: In the following section, for brevity, we will omit the regularization term, as well as all of the above constraints except the dynamics from the problem definitions, but they should always be assumed to be present.

5 Results

5.1 Solution Examples

Base Problem Solutions

Bang-Bang Solutions

To achieve *bang-bang* solution—in which the control is maxed out in one direction and then another—we can *augment* the states of the problem and include the derivative of the controls, $\dot{\mathbf{a}}_k$, as the new control variable. Putting an L_1 cost on the derivatives will then make it so the solver wants to zero out as many of the $\dot{\mathbf{a}}_k$'s as possible, giving us the type of solution we want. The new dynamics for this augmented problem are then

$$\mathbf{f}(\tilde{\psi}_{k+1},\mathbf{a}_{k+1},\tilde{\psi}_k,\mathbf{a}_k,\dot{\mathbf{a}}_k) = \begin{pmatrix} \mathbf{P}^{(n)}(\tilde{\psi}_{k+1},\tilde{\psi}_k,\mathbf{a}_k) \\ \mathbf{a}_{k+1} - \mathbf{a}_k - \dot{\mathbf{a}}_k \cdot \Delta t \end{pmatrix}. \tag{43}$$

This allows us to write the following problem:

$$\begin{aligned}
\underset{\tilde{\psi},\mathbf{a},\dot{\mathbf{a}}}{\text{minimize}} \quad & J(\tilde{\psi},\mathbf{a},\dot{\mathbf{a}}) = \ell(\tilde{\psi}_N) + \sum_{k=1}^{N-1} \|\dot{\mathbf{a}}_k\|_1 && (44)\\
\text{subject to} \quad & \mathbf{f}(\tilde{\psi}_{k+1},\mathbf{a}_{k+1},\tilde{\psi}_k,\mathbf{a}_k,\dot{\mathbf{a}}_k) = 0 && (45)
\end{aligned}$$

Smooth Solutions

Guard State Solutions

Free Time Solutions

Minimum Time Solutions

5.2 Comparison to Other Methods

5.3 Hardware Results