AMSURE Report

Robust Optimal Control for Quantum Systems

Qianyu Zhu (Julie)

Advisor: Fortino Garcia and Georg Stadler

October 24, 2022

Project description

In classical computers, information is stored in a unit known as a bit which can take a value of either 0 or 1. A computer operates on strings of input bits through the application of logic gates to create a desired output. In sharp contrast, quantum computers are based on *qubit*, which can take the value of both 0 and 1, with some probability. Most modern quantum computers typically have only the most basic logic gates natively supported and more complex operations on strings of qubits must be designed from scratch. Even worse, current quantum computers have a lot of sources of noise making designing accurate gates difficult.

In this project we'll implement a desired logic gate for a quantum computer by solving an optimal control problem. We want to find the best way to control the quantum system which maximizes the accuracy of the desired transformation given that the state of a system evolves according to Schrödinger's equation. Since the coefficients in the Schrödinger's equation may have some uncertainty, it is desirable to find the controls that are also robust to noise. There are several ways to design noise-resilient gates, such as risk-neutral optimization to minimize the expectation of the objective function subject to uncertain parameters. A result of a risk-neutral optimization is plotted in Fig.1, where a traditional "noise-free" optimization rapidly loses accuracy when noise is present while the "Risk-Neutral" performs (on average) better

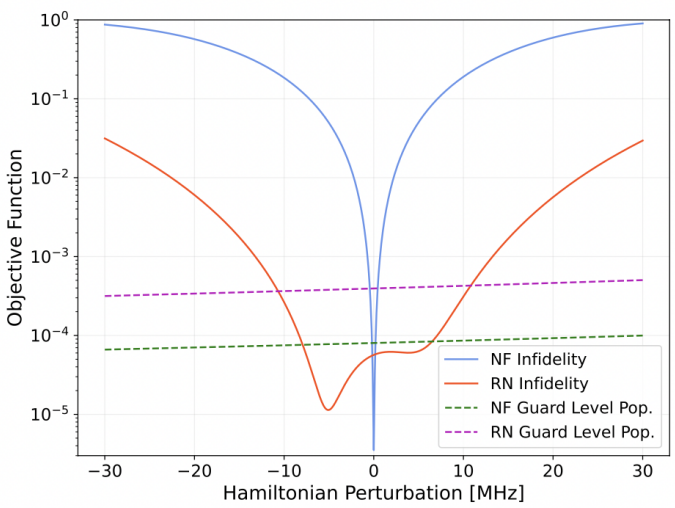


Figure 1: Infidelity objective and guard level objective as functions of the noise in the Schrödinger equation model. Here 'NF' and 'RN' correspond to the "Noise-Free" and "Risk-Neutral" cases.

for a wider range of noise.

For this project we'll investigate choosing different objective functions to balance worst-case and average performance for various noise levels and build logic gates for quantum computers that are robust to noise. To perform all of the optimization numerically we will use the open-source Julia library Juqbox.jl (https://github.com/LLNL/Juqbox.jl).

Weekly progress

1 Week 1 (Background of quantum system and logic gates)

This week I started with understanding basic concepts and postulates in quantum mechanics and quantum computing, including qubit, quantum logic gate, and quantum system [3].

1.1 Some terminologies

The following is brief introduction to the main concepts we would frequently use in the future:

1. Qubit

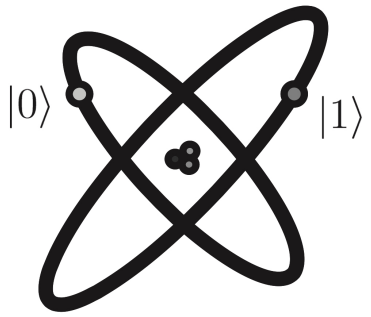


Figure 2: A qubit has some probability to stay at energy state 0 or 1.

Qubit is the most simple example of a quantum mechanical system. Like traditional bit in classical computer, a qubit may take two states: the ground state $|0\rangle$ or the excited state $|1\rangle$ (we are using the Dirac notation). One is allowed to including more states, but they are too rare and beyond our consideration at the moment.

While a bit is clearly 0 or 1, a qubit enjoys certain randomness of being a linear combination of $|0\rangle$ and $|1\rangle$ before it is observed, which is called *superposition* (imagine an biased coin being tossed, its states before and after falling onto the ground!). Such qubit $|\psi\rangle$ is connected with a *state vector*

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

with the normalization condition

$$|\alpha|^2 + |\beta|^2 = 1, \tag{2}$$

α, β also called the *probabilityamplitude* for different states, and they are complex numbers. One can see that $|\alpha|^2$ and $|\beta|^2$ actually represent the respective probability of a qubit falling into $|0\rangle$ or $|1\rangle$ when observed at a specific time.

Multiple qubits:
Suppose we have two qubits, with state vector

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle \tag{3}$$

It is clear that the measurement outcomes of two qubits are strongly correlated. For

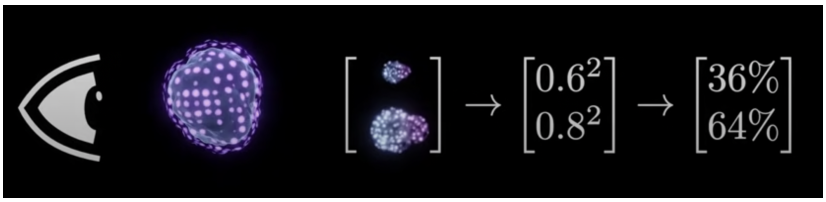


Figure 3: When we obeserve a qubit, the qubit falls into either energy state 0 or state 1, with corresponding probability.

example, a measurement of the second qubit of the *Bell state* or *EPR pair*:

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

would always gives the same result as the measurement of the first qubit.

2. Quantum Logic Gate

   Analogous to the electrical circuit containing logic gates, a quantum computer is built from *quantum circuit* consisting of elementary *quantum gates* to carry around and manipulate the quantum information. A simple example is the *NOT gate*:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

   Quantum gates on a single qubit can be described by 2*2 matrices, and in fact any unitary matrix can be used as a valid quantum logic gate! More examples (*Z gate* and *Hadamard gate*):

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

   However, there is no similar quantum gates parallel to AND, OR, NAND, or XOR gates in classic computer, since they are irreversible, and we suffer from loss of information (not unitary!). On the other hand, unitary quantum gates are always invertible, since the inverse of a unitary matrix is also a unitary matrix. This is one reason quantum computing can have extraordinary power compared with traditional one.

3. Evolution of Quantum System

The evolution of a quantum state (wave function) $|\psi\rangle$ is described by the *Schrödinger's equation*,

$$i\hbar\frac{\partial\,|\psi(x,t)\rangle}{\partial t} = H\,|\psi(x,t)\rangle\,, \qquad H = \left[-\tfrac{\hbar^2}{2m}\tfrac{\partial^2}{\partial^2 x} + V(x,t)\right] \tag{4}$$

H represents the Hamiltonian operator, which looks different in different physical systems, and $\hbar$ is Planck's constant (usually taken as 1 for simplicity).

If H is independent of time, the solution to (4) is:

$$|\psi(t_2)\rangle = exp\left[\frac{-iH(t_2-t_1)}{\hbar}\right]|\psi(t_1)\rangle = U(t_1,t_2)\,|\psi(t_1)\rangle \tag{5}$$

In our problem setting, the quantum system is not closed because experimentalists hope to use electromagnetic field or laser to control the system. However, it still evolves according to the Schrödinger's equation with a time-varying Hamiltonian, to some good approximation.

If we know the Hamiltonian of a system and the Planck's constant, we would understand the dynamics completely, but that turns out to be a challenging work, requiring substantial experiments and observations.

1.2 Introduction to research topic

With these terms well-defined, we now introduce our quantum optimal control problem of implementing unitary gates in a quantum computer. The state of the quantum system is described by the state vector $\psi \in \mathbb{C}^N$. The evolution of $\psi(t)$ in the time interval $t \in [0,T]$ is governed by Schrodinger's equation:

$$\begin{cases} \dot{\psi} = -iH(t)\psi \\ \psi(0) = \psi_0 \end{cases}, \qquad H(t) = H_s + H_c(t), \qquad t \in [0,T] \tag{6}$$

The quantum system would evolve under the system Hamiltonian H_s without perturbation, while the control Hamiltonian H_c models the action of external control fields on the system. Given a valid quantum system, we can determine H_s, ψ_0, and the rough structure of H_c, with some controllable parameters.

The goal of the quantum optimal control problem is, given a desired logic gate transformation $V_{tg} \in \mathbb{C}^{N\times N}$, we will optimize the control Hamiltonian H_c (by minimizing an object

function, which will be introduced next week), so that the state vector $|\psi_t\rangle$ evolves from initial state ψ_0 to target state $\psi_T = V_{tg}\,|\psi_0\rangle$, under the PDE constraint (6).

In practice, H_0 and H_c would subject to a certain amount of noise. Our designed H_c should ensure a consistent and stable behavior under specific noise influence.

An elementary illustration for single qubit could be:

$$H(t) = \hat{\omega}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} + f(t)\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$
$$\hat{\omega} = \omega + \delta, \qquad \delta \approx Unif(-\epsilon, \epsilon),$$

(This turns out to be a bad example at the moment—-why Schrodinger equation, a PDE, behaves like matrix ODEs? and why Hamiltonian, which contains second derivative in space, ends up being a 2*2 matrix? please refer to week3 if confusion happens)

Let $T = 100ns$, we may hope initial state $(1,0)^T$ to reach $\frac{1}{\sqrt{2}}(1,1)^T$, initial state $(0,1)^T$ to reach $\frac{1}{\sqrt{2}}(1,-1)^T$ at time T. In other words, our Hamiltonian will work like a Hadamard gate after a certain time period T,

$$U(T) = exp\int_0^T -iH(s)\mathrm{d}s, \qquad \psi(t) = U(T)\psi_0.$$

1.3 Matrix-form Quantum System

For the problem to be well-posted, the gate transformation should be satisfied for at least a basis in the state vector space. Let the columns of U_0 be such a basis, each satisfies (6), leading to a similar equation in matrix form,

$$\begin{cases} \frac{\mathrm{d}U(t)}{\mathrm{d}t} = -iH(t)U(t), \\ U(0) = U_0, \qquad t \in [0,T]. \end{cases}$$

2 Week 2 (Explanation of the optimal control problem)

This week I studied the workflow for solving general quantum optimal control problem, especially the single-quantum case named *Rabi oscillation*. The following are several additional explanations about (6):

2.1 Hamiltonian model

Several Hamiltonian models exist for describing the quantum physics of different superconducting circuits, and in our experiment we work with a composite quantum system consisting of Q qubits, whose Hamiltonian satisfies:

$$H_s = \sum_{q=1}^{Q} \left(\omega_q a_q^\dagger a_q - \frac{\xi_q}{2} a_q^\dagger a_q^\dagger a_q a_q - \sum_{p>q} \xi_{pq} a_p^\dagger a_p a_q^\dagger a_q \right) \tag{7}$$

where ω_q represents the ground state transition frequency and ξ_q, ξ_{pq} represents the self-Kerr coefficient of subsystem q and the cross-Kerr coefficient between subsystems p and q, respectively (known for a specific system). a_q denotes the lowering operator or annihilation operator for subsystem q, which takes the form

$$a_q = \begin{pmatrix} 0 & \sqrt{1} & 0 & 0 & \cdots \\ 0 & 0 & \sqrt{2} & 0 & \cdots \\ 0 & 0 & 0 & \sqrt{3} & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{pmatrix} \in \mathbb{R}^{n_q \times n_q}$$

H_c is usually designed to be continuous. However, the space of all continuous functions is huge and we do not want to build H_c from scratch. In fact, we have a rough model for H_c and keep tuning its parameters throughout experiment,

$$H_c = f(t)A,$$

where $f(t) = f(t, \alpha)$ is a real-valued, continuous function depends on time and parameters α, and A is some time-independent, skew-Hermitian matrix ($A^\dagger = A^{-1}$). In our experiment, $A = \sum_{q=1}^{Q} (a_q + a_q^\dagger)$.

2.2 Rotating Wave Approximation

The system constants in (7) are approximately

$$\omega_q \approx 5GHz = 5 \cdot 10^9 Hz,$$
$$\xi_q \approx 5 \cdot 10^{-3} GHz = 5 \cdot 10^6 Hz,$$

When numerically solving the Schrodinger's equation, larger frequency requires computer to use smaller step-size in order to ensure accuracy, which usually means higher computational cost. Here ω_q dominates the step-size, and we somehow wish to eliminate this term.

We multiple both sides by an unitary rotating frame transformation matrix $R(t)$, careful computation yields

$$\frac{\mathrm{d}\tilde{\psi}}{\mathrm{d}t} = -iH\tilde{(t)}\tilde{\psi},$$
$$H\tilde{(t)} = R(t)H(t)R^{\dagger}(t) + i\dot{R}(t)R^{\dagger}(t),$$
$$\psi\tilde{(t)} = R(t)\psi(t).$$

In single-qubit system,

$$R(t) = e^{i\omega_{r,q}a^{\dagger}at},$$

tensor product is used for multi-qubit system.

This leads to Hamiltonian in rotating frame:

$$\tilde{H}_s = \sum_{q=1}^{Q}\left((\omega_q - \omega_{r,q})a_q^{\dagger}a_q - \frac{\xi_q}{2}a_q^{\dagger}a_q^{\dagger}a_q a_q - \sum_{p>q}\xi_{pq}a_p^{\dagger}a_p a_q^{\dagger}a_q\right), \tag{8}$$

$$\tilde{H}_c(t,\alpha) = \sum_{q=1}^{Q} f_q(t,\alpha)\left(e^{-i\omega_{r,q}t}a_q + e^{i\omega_{r,q}t}a_q^{\dagger}\right). \tag{9}$$

To eliminate ω_q in (8), we choose $\omega_{r,q} = \omega_q$. And we hope to integrate $e^{\pm i\omega_{r,q}t}$ in (9) into $f_q(t,\alpha)$, which is real-valued. We set f to be:

$$f_q(t) := 2\,\mathrm{Re}(d_q(t)e^{i\omega_{r,q}t}) = d_q(t)e^{i\omega_{r,q}t} + \bar{d_q}(t)e^{-i\omega_{r,q}t} \tag{10}$$

Substituting (10) into (9), we get

$$\tilde{H}_c(t,\alpha) = \sum_{q=1}^{Q}\left(d_q(t)a_q + \bar{d_q}(t)a_q^{\dagger} + d_q(t)e^{2i\omega_{r,q}t}a_q^{\dagger} + \bar{d_q}(t)e^{-2i\omega_{r,q}t}a_q\right) \tag{11}$$

$$= \sum_{q=1}^{Q}\left(p_q(t)(a_q + a_q^{\dagger}) + iq_q(t)(a_q - a_q^{\dagger})\right) \tag{12}$$

the terms oscillating with frequency $2\omega_{r,q}$ are ignored in practical approximation, since tiny oscillation will be averaged out in integration.

By the steps above, we manage to remove the high frequency ω_q from H_s, and the highly oscillatory factor $e^{i\omega_{r,q}t}$ from H_c (at least not exist explicitly). From now on, we would deal with problems in rotating frame by default, with target gate $\tilde{V}_{tg} = R(t)V_{tg}$. " $\sim$ " symbol will be ignored for simplicity and clarity among all following content.

2.3 Gate Fidelity and Loss Function

Fidelity is a non-unique distance measure between quantum states. The gate fidelity decides how similar two gates are, in other words, to what extent they overlap with each other.

The overlap between the target gate matrix and the solution operator matrix at the final time can be measured by

$$O_T := \langle U(T), V_{tg} \rangle_F,$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenious matrix scalar product.

Furthermore, we can quantify the difference between $U(T)$ and V_{tg} by a loss function (the target gate infidelity)

$$L_1(U(T,\alpha)) = 1 - \frac{1}{N^2}|\langle U(T,\alpha), V_{tg}\rangle_F|^2 \tag{13}$$

$$= 1 - \frac{1}{N^2}|Tr(U(T,\alpha)^\dagger V_{tg}|^2, \tag{14}$$

α underlines that the solution operator matrix and loss function subject to some controllable parameters. N equals to the dimension of our state vector.

3 Week 3 (Toy-model: one-qubit system with swap gate)

3.1 Notes for Numerical Methods

For later purpose of numerically solving the Schrodinger's equation, I read through the first four chapter of "A First Course in the Numerical Analysis of Differential Equations", and comb some basic schemes in Appendix A. (maybe unrelated to this project, just useful notes:))

3.2 From physical system to matrix ODEs

The evolution of a quantum state (wave function) $|\psi\rangle$ is described by the *Schrödinger's equation*,

$$i\hbar\frac{\partial\left|\psi(x,t)\right\rangle}{\partial t}=H\left|\psi(x,t)\right\rangle,\qquad H=\left[-\tfrac{\hbar^2}{2m}\tfrac{\partial^2}{\partial^2 x}+V(x,t)\right] \tag{15}$$

H represents the Hamiltonian operator, $V(x,t)$ is a real-valued potential energy function, and $\hbar$ is Planck's constant.

To solve the equation numerically, we employ a recipe which transforms a PDE into a system of ODEs. Let us start with a quantum harmonic oscillator, which has potential function $V(x,t)=V(x)=\frac{1}{2}m\omega^2x^2$.

$$H\left|\psi\right\rangle=-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial^2 x}\psi+\frac{1}{2}m\omega^2x^2\psi.$$

We define the scaled momentum p and position q operators by

$$p\psi=-ix_0\partial_x\psi,\qquad q\psi=\frac{x}{x_0}\psi,\qquad x_0=\sqrt{\frac{\hbar}{m\omega}},$$

and the original PDE becomes

$$H\psi=\frac{\hbar\omega}{2}(p^2+q^2).$$

Some computation gives

$$H\psi=\hbar\omega(a^\dagger a+\frac{1}{2}),$$
$$a^\dagger=\frac{1}{\sqrt{2}}(q-ip),\qquad a=\frac{1}{\sqrt{2}}(q+ip),$$

a is called the lowering or annihilation operator, $a^\dagger$ is called the raising or creation operator.

It is possible for us to find a family of eigenfunctions ϕ_n for H, which forms an orthonormal basis on a Hilbert space H (remember H is compact self-adjoint operator, by theorem from functional analysis). In particular,

$$a\phi_0=0,\qquad \phi_n=\frac{1}{\sqrt{n!}}(a^\dagger)^n\phi_0.$$

In quantum physics, $\phi_n=|n\rangle$ are called the n-th eigenstate. Any function $\psi\in\mathbb{H}$ can be expressed as

$$\psi(x,t)=\sum_{k=0}^{\inf}\psi_j(t)\phi_j(x),\qquad \psi_j(t)=\langle\phi_j,\psi(x,t)\rangle.$$

15 can be written as an inf-dim system of ODEs,

$$\frac{\mathrm{d}\left|\psi\right\rangle}{\mathrm{d}t} = -\frac{i}{\hbar}\tilde{H}\left|\psi\right\rangle, \qquad \tilde{H}_{ij} = \langle\phi_i, H\phi_j\rangle, \qquad \psi = \begin{bmatrix}\psi_0\\ \psi_1\\ \psi_2\\ \cdots\end{bmatrix},$$

From the formula of eigen-function ψ_n, we can derive the matrix form of a:

$$a = \begin{bmatrix}0 & \sqrt{1} & 0 & 0 & \cdots\\ 0 & 0 & \sqrt{2} & 0 & \cdots\\ 0 & 0 & 0 & \sqrt{3} & \cdots\\ 0 & 0 & 0 & 0 & \cdots\end{bmatrix}, \qquad a^\dagger = a^T.$$

3.3 Rabi Oscillation

In our research, our system consists of super-conducting qubits, which has different Hamiltonian, but all the eigenstates (eigen-functions of Hamiltonian) remain the same.

Let us consider the quantum system of single super-conducting qubit in a rotating frame of reference (refer to 2.2 for how we do this!). A truncated model expansion of Schrodinger's equation is used to model the

$$\begin{cases}\frac{\mathrm{d}\psi}{\mathrm{d}t} = -iH_c(t)\psi,\\ U(0) = \begin{bmatrix}1\\ \\ 0\end{bmatrix}, \qquad t \in [0, T].\end{cases}$$

where H_c satisfies

$$H_c(t) = f(t)a + \bar{f}(t)a^\dagger, \qquad a = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}, \qquad f(t) = \Omega \in \mathrm{C},$$

(why $H(t) = H_c(t)$ here? We do rotating frame transformation, and $a^\dagger a^\dagger aa = diag(N^2 - N) = 0 when N = 2$)

Analytic method(just for your reference)

Such system is called *Rabi Oscillator*, and enjoys an analytic solution. It is possible

(usually not) for us to compute the orthogonal decomposition of H_c:

$$H_c = X\Lambda X^{-1}, \qquad X = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -\Omega/|\Omega| \\ \bar{\Omega}/|\Omega| & 1 \end{bmatrix}, \qquad \Lambda = \begin{bmatrix} |\Omega| & 0 \\ 0 & -|\Omega| \end{bmatrix}.$$

Variable substitution $\tilde{\psi} = X^\dagger \psi$ yields

$$\dot{\tilde{\psi}} = -i\Lambda\tilde{\psi}, \qquad \tilde{\psi}(t) = \begin{bmatrix} Ae^{-i|\Omega|t} \\ Be^{|\Omega|t} \end{bmatrix}.$$

Transforming back and applying initial condition, we can solve A, B explicitly, and the analytic solution is

$$\psi(t) = \begin{bmatrix} \cos(|\Omega|t) \\ -i\frac{\bar{\Omega}}{|\Omega|}\sin(|\Omega|t) \end{bmatrix},$$

From probability perspective,

$$|\alpha|^2 = \cos^2(|\Omega|t) = \frac{1}{2}(\cos(2|\Omega|t)),$$
$$|\beta|^2 = \sin^2(|\Omega|t) = \frac{1}{2}(1 - \cos(2|\Omega|t).$$

The system oscillates between two states with period $T = \frac{\pi}{|2\Omega|}$, which only depends on the module of Ω.

Numerical method

For single-qubit system ($q = 1$), by (7), the Hamiltonian is

$$H_s = \omega a^\dagger a - \frac{\xi}{2}a^\dagger a^\dagger aa, \tag{16}$$
$$H_c = f(t)(a + a^\dagger). \tag{17}$$

since all the cross-term only exists for multiple subsystems.

With H_s defined as above, we permute the term in Schrodinger's equation

$$\dot{\psi} + i\omega a^\dagger a\psi = -i(\frac{\xi}{2}a^\dagger a^\dagger aa + f(t)(a + a^\dagger))\psi,$$

multiple both sides by

$$R(t) = e^{i\omega a^\dagger at}, \qquad R^{-1} = R^\dagger,$$

left hand side becomes

$$R(t)\dot{\psi} + i\omega R(t)a^\dagger a\psi = -iR(t)(\frac{\xi}{2}a^\dagger a^\dagger aa + f(t)(a + a^\dagger))\psi, \tag{18}$$

$$= \frac{\mathrm{d}}{\mathrm{d}t}\left[R(t)\psi\right], \tag{19}$$

set $\tilde{\psi} = R(t)\psi$, then $\psi = R^\dagger\tilde{\psi}$, (19) becomes

$$\begin{aligned}
\psi &= -iR(t)(\frac{\xi}{2}a^\dagger a^\dagger aa + f(t)(a + a^\dagger))R^\dagger(t)\tilde{\psi} \\
&= -i(\frac{\xi}{2}a^\dagger a^\dagger aa + f(t)(a + a^\dagger))\tilde{\psi} \\
&= -if(t)(a + a^\dagger)\tilde{\psi} \\
&= -iH_c\tilde{\psi}
\end{aligned}$$

To numerically compute the optimal function f, we would first assume f can be approximated by finite linear combination of a family $\{B_k(t)\}$:

$$f(t) = \sum_{k=1}^{N} \alpha_k B_k(t).$$

The model for control Hamiltonian can split into symmetric and anti-symmetric terms

$$\begin{aligned}
H_{sym} &= p(t)(a + a^\dagger), \\
H_{asym} &= q(t)(a - a^\dagger), \\
H_c &= H_{sym} + iH_{asym}.
\end{aligned}$$

The connection between $f(t), p(t), q(t)$ is as following:

$$f(t) = 2p(t)\cos\omega t - 2q(t)\sin\omega t.$$

4 Week 4 (Numerical methods and technical details in optimization)

4.1 Quadratic B-splines

We parameterize the control function $f(t)$ using basis functions called B-splines. Continue from equation (11),

$$d_k(t, \alpha) = \sum_{b=1}^{N_b} \hat{S}_b(t)\alpha_b, \qquad k \in [1, Q].$$

where the complex coefficients $\alpha_b^k = \alpha_b^{k(r)} + i\alpha_b^{k(i)}$ are control parameters to be tuned through optimization ($D = 2QN_b$ real valued parameters).

The basis functions $\hat{S}_b(t)$ are piece-wise quadratic B-spline wavelets centered on a uniform grid in time, (picture of B-spline family).

Since each spline has finite local support, at any fixed time t, only at most three splines have contribution to the control function, leading to efficient evaluation.

4.2 Symplectic Runge-Kutta method

The ODE is solved forward in time to compute the loss under certain parameters α. We transform the Schrodinger's equation into real-valued formulation in computer,

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} S(t) & -K(t) \\ K(t) & S(t) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} =: \begin{bmatrix} f^u(u,v,t) \\ f^v(u,v,t) \end{bmatrix}, \qquad \begin{bmatrix} u^0 \\ v^0 \end{bmatrix} = \begin{bmatrix} g^u \\ g^v \end{bmatrix},$$
$$u = \mathrm{Re}(\psi), \quad v = -\mathrm{Im}(\psi), \quad S(t) = \mathrm{Re}(H), \quad K(t) = \mathrm{Re}(H).$$

We will use the Stormer-Verlet Scheme to solve this real-valued ODEs *forward in time*. It is a two-stage partitioned Runge-Kutta (PRK) scheme which is symplectic, time-reversible, and second order accurate.

$$\begin{aligned} u^0 &= g^u, \\ u^{n+1} &= u^n + h\sum_{i=1}^{s} b_i^u k^{n,i}, \\ k^{n,i} &= f(U^{n,i}, V^{n,i}, tn + c_i^u h), \\ U^{n,i} &= u^n + h\sum_{i=1}^{s} a_{ij}^u k^{n,j}, \end{aligned}$$

We are using slightly different coefficients for real and imaginary parts:

$$a_{11}^u = a_{12}^u = 0, \quad a_{21}^u = a_{22}^u = \frac{1}{2}, \qquad a_{11}^v = a_{21}^v = \frac{1}{2}, \quad a_{12}^v = a_{22}^v = 0,$$
$$b_1^u = b_2^u = \frac{1}{2}, \quad c_1^u = 0, \quad c_2^u = 1, \qquad b_1^v = b_2^v = \frac{1}{2}, \quad c_1^v = c_2^v = \frac{1}{2}.$$

4.3 Propagation of gradient(mistakes happen here, have to check and put more details later)

I finished the computation part with the help of
`https://cs.stanford.edu/~ambrad/adjoint_tutorial.pdf`.

$$\begin{aligned} &\min_{\alpha} L(\alpha) := 1 - \frac{1}{N^2}\left|Tr(U_T^{\dagger}(\alpha)V_{tg})\right|^2, \\ s.t. \quad &\begin{cases} \frac{d}{dt}U(t) = -iH(t;\alpha)U(t), \\ U(0) = U_0 = I. \end{cases} \end{aligned} \tag{20}$$

The most important question in optimization may be how we update these parameters α according to the loss at current step. In Juqbox, we update parameters basing on gradient search: the gradient is computed by adjoint state method, and optimization by quasi-Newton method.

To find $\nabla_\alpha L$ at current value of α under the constraint of the ODEs, we employ Lagrange multiplier (or adjoint state vector). Consider the following Lagrangian functional

$$\mathbb{L}(\alpha, \psi, \lambda) = L(\alpha) - \int_0^T \lambda^T(t)\big(\dot{\psi} + iH(t;\alpha)\psi\big)\mathrm{d}t, \tag{21}$$

Integration by parts and take total derivative w.r. to α,

$$\nabla_\alpha \mathbb{L} = \nabla_\alpha L - \lambda^T \psi_\alpha\big|_0^T + \int_0^T \big(\dot{\lambda}\big)^T \psi_\alpha dt - \int_0^T i\lambda^T(H_\alpha\psi + H\psi_\alpha)dt \tag{22}$$

$$= \nabla_\alpha L - \lambda^T \psi_\alpha\big|_0^T + \int_0^T \big[(\dot{\lambda}^T - i\lambda^T H)\psi_\alpha - i\lambda^T H_\alpha \psi\big]dt \tag{23}$$

$$= 0. \tag{24}$$

The equations above provide us with a recipe for solving optimization problem through several gradient research.

Start from an initial guess for $\alpha(t)$, $\big($in the actual model we used, $\alpha(t) = \alpha\big)$:

1. solve the Schrodinger's equation forward in time by the Stormer-Verlet scheme with given initial condition, we get loss and terminal conditions for the state variables;

2. To eliminate the term ψ_α in (23), we want the coefficient $\dot{\lambda}^T - i\lambda^T H = 0$ at ant time. Apply extra condition $\lambda(T) = 0$, solve (23) backward in time, we get $\lambda(t)$;

3. obtain $\nabla_\alpha L$ from (24).

(there are plenty of ways to utilize the Lagrange functional, in some materials this is solved by method of variation — an extensive field to explore)

The rest steps of optimization is finished by interior-point optimization package IPOPT. It uses the L-BFGS algorithm, which only relies on the objective function and its gradient.

5 Week 5 (Noise model introduction)

This week I spent half of the time combing the structure of this project, making slides for presentation, and half of the time studying risk-sensitive optimization method.

5.1 Loss function under noise

In quantum computation, high precision and robustness against noise are two highly demanded qualities for quantum system control. The robust control aims at achieving a desired gate operation V_{tg} at some final time T with high accuracy, which must be also insensitive to the uncertainties (pulse distortion, dephasing noise, crosstalks, etc.) so as to maintain high performance under certain interference.

Usually there is a trade-off relationship between the two virtues, and we will choose proper loss function to achieve a balance. The design of robust controls needs to be based on proper measure of the robustness. In previous chapter we defined loss function (13) under zero-noise condition, and now we define

$$L(\alpha, \epsilon) = \frac{1}{N^2} |\langle U_T(\alpha, \epsilon), V_{tg} \rangle_F|^2,$$

to be the gate infidelity (or error) under the control α and the uncertainty ϵ. We use the letter F to represent generalized loss function. If the probability distribution $P(\epsilon)$ is known, we can define the average infidelity

$$F(\alpha) = \int_\epsilon L(\alpha, \epsilon) dP(\epsilon), \tag{25}$$

or we can consider the worst-case fidelity (which is closely related to the variance of infidelity!!!)

$$F(\alpha) = (\max_\epsilon) L(\alpha, \epsilon). \tag{26}$$

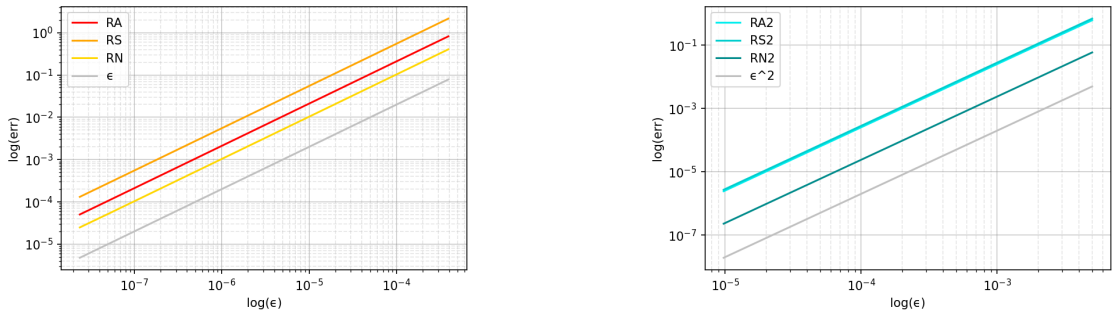


Figure 4: log-log plot between disturbance and gradient error.
left: first-order gradient checking, $k = 1$ implies linear convergence,
right: second-order gradient checking, $k = 2$ implies quadratic convergence.

5.2 Gradient checker

Back propagation has a lot of details and can be a little bit tricky to implement. Sometimes when you run it with gradient descent or some other optimization algorithm, the back propagation seemingly works (with subtle bugs), and yields fake optimal solutions. And you might just not know that there was this bug that was giving you worse performance. To eliminates this kind of issues, we used gradient checking to numerically check the derivatives computed by your code.

We are free to choose random parameters, and compute the loss $L(\alpha)$ and gradient ∇L using the code we implemented. Then, we add small perturbation ϵa_i to each single parameter and compute the loss $L(\alpha + \epsilon a_i)$ again.

Convergence of gradient is plotted in Fig. 4. By Taylor expansion,

$$\frac{L(\alpha + \epsilon a_i) - L(\alpha)}{\epsilon} \approx \nabla L + \vartheta(\epsilon),$$

we expect to observe linear convergence when cutting ϵ into half, see Fig. 4.

We can also plot higher-order convergence according to Taylor's expansion

$$f(x + \epsilon) = f(x) + \epsilon f'(x) + \frac{\epsilon^2}{2} f"(x) + \varepsilon(\epsilon^3),$$
$$\frac{f(x + \epsilon) - f(x - \epsilon)}{2\epsilon} = f(x) + \varepsilon(\epsilon^2),$$

The convergence rate for the LHS in the second line should be quadratic, see Fig. 4.

Fig.5 shows two extreme cases. When ϵ is extremely large ($\geq 10^{-2}$), the linear convergence

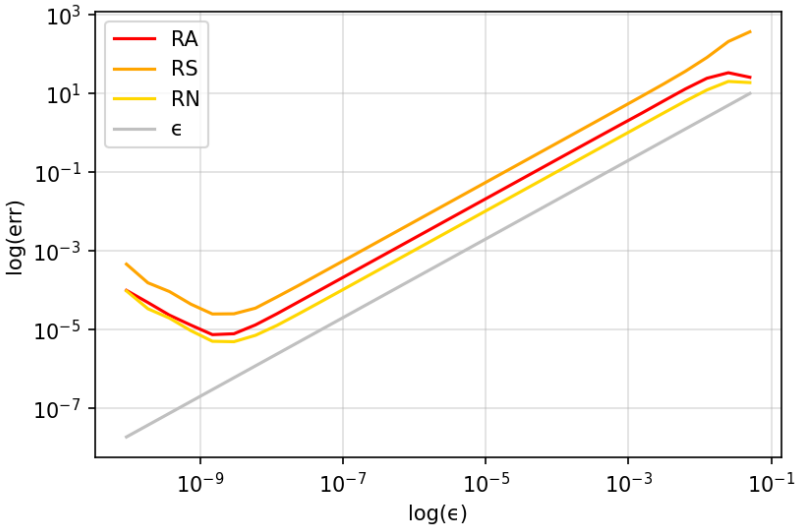


Figure 5: extreme cases w.r. to large and small disturbance

is ruined due to locality of Taylor expansion; on the other hand, when ϵ is very tiny ($\leq 10^{-8}$), the actual error is dominated by numerical error.

5.3 Possible noise forms

1. uniform noise

   $\epsilon \sim \text{Unif}(-\delta, \delta)$

2. Bimodal-Gaussian noise

3. high-dim, time dependent noise? use Monte-Carlo sampling!

5.4 Possible utility functions

1. exponential utility function

   $V_\mu = e^{\mu L}$

   We use the exponential function in our code.

2. HARA utility function

   $V_\mu = L^\mu$

6 Week 6 (Risk-aware optimization)

6.1 Risk-neutral method

What has already been done is the risk-neutral optimization for one-qubit system (three energy levels). We use the so-called risk-neutral utility function $F(\alpha) = \mathbb{E}[L(\alpha, \epsilon)]$, which is just (25).

Assuming the noise is uniformly distributed in $[-\delta, \delta]$, we could compute G

$$F(\alpha) = \mathbb{E}[L(\alpha, \epsilon)] = \int_{\epsilon} L(\alpha, \epsilon) dP(\epsilon) \approx \sum_{k=1}^{M} \omega_k L(\alpha, \epsilon_k), \tag{27}$$

where ω_k, ϵ_k are Gauss–Legendre quadrature weightsnodes.

(**adaptive sensitivity**

It is reasonable to believe, as iterating process goes, the algorithm's performance diversity tends to shrink (i.e. smaller variance in batch of data), and the risk-sensitivity diminishes.

The experiment result:)

6.2 Risk-Sensitive method

This part is motivated by Ge's paper on Risk-neutral optimization [2]. The average error is usually easier to evaluate and optimize, but have weak control over the error variance. The worst-case error is robust, but has worse performance in high-precision regime. To make better use of both measures, we consider the following risk-sensitive (RS) criterion

$$F_\mu(\alpha) = \int_{\epsilon} V_\mu[L(\alpha, \epsilon)] dP(\epsilon) \tag{28}$$

where $V_\mu(\cdot)$ is a pre-selected differentiable utility function parameterized by the sensitivity parameter μ, and it decides how much weight we want to distribute to different errors. (for examples of utility function, refer to 5.4.) For example, $V_\mu = e^{\mu L}$, when $\mu \to 0^+ / + \inf$, the loss approaches the average infidelity/worst-case infidelity.

Let $\boldsymbol{\epsilon} = \{\epsilon_1, \epsilon_2, \cdots, \epsilon_M\}$ be a random/fixed mini-batch of size M, ω_k be the quadrature weights, then the empirical RS loss can be approximated by

$$F_\alpha(\alpha, \boldsymbol{\epsilon}) = \sum_{k=1}^{M} \omega_i V_\mu \big[L(\alpha, \epsilon_i)\big]. \tag{29}$$

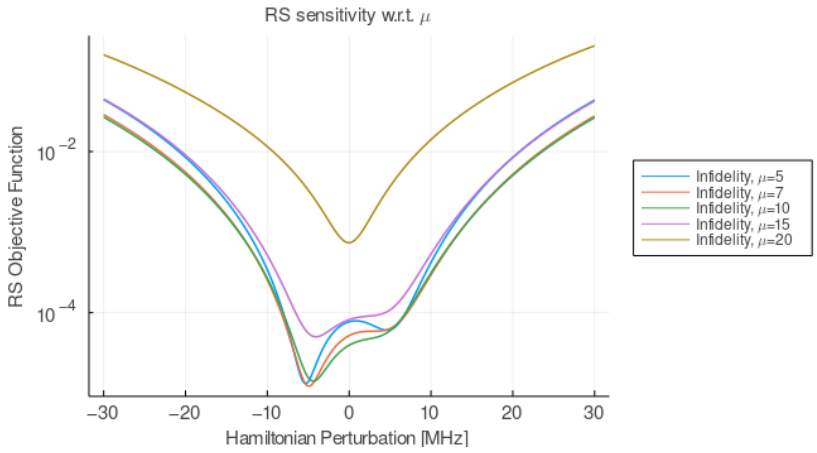


Figure 6: accuracy and robustness of RS method with different μ

The gradient is straight-forward by Chain Rule:

$$G_\alpha(\alpha, \boldsymbol{\epsilon}) = \sum_{i=1}^{M} \omega_i V'_\mu\big[L(\alpha, \epsilon_i)\big] \nabla_\alpha L(\alpha, \epsilon_i).$$

Normalizing the weights assigned to each sample, we get

$$G_\alpha(\alpha, \boldsymbol{\epsilon}) = \sum_{i=1}^{M} \tilde{\omega}_i \nabla_\alpha L(\alpha, \epsilon_i),$$

the importance of each uncertain noisy sample is re-evaluated by the function V_μ, and the adjusted weights are as below:

$$\tilde{\omega}_i = \frac{\omega_i V'_\alpha\big[L(\alpha, \epsilon_i)\big]}{\sum_{j=1}^{M} \omega_j V'_\alpha\big[L(\alpha, \epsilon_j)\big]}. \tag{30}$$

Fix the random seeds and loop over different value of μ, we get the plot in Fig. 6, $\mu = 10$ yields the best behavior for both non-noise accuracy and robustness. When $\mu \geq 20$, it seems it is easier for the algorithm to be trapped in a non-optimal local minimum and subjects to early stop.

Another potential issue with risk-sensitive method is, the gradient vanishes around global minimum, and it would take huge amount of time to finally converge to the optimum.

This week I will apply Risk-Neutral, Risk-Sensitive, and Risk-Averse methods to recover a $|0\rangle$ to $|2\rangle$ swap gate on a single qudit with 3 energy levels (and 1 guard state). Gradient checker is used along the process to double check the correctness of our code. At the same time, we hope to plan out what to experiment in the second half of the program.

6.3 Risk-Averse method

One of the most common loss function in actual quantum physics may be

$$F(\alpha) = \mathbb{E}[L(\alpha,\epsilon)] + \frac{\theta}{2}\cdot \mathrm{Var}[L(\alpha,\epsilon)] \tag{31}$$

$$\approx \sum_{k=1}^{M}\omega_k L(\alpha,\epsilon_k) + \frac{\theta}{2(M-1)}\sum_{k=1}^{M}\big(L(\alpha,\epsilon_k)-\overline{L}\big)^2 \tag{32}$$

where $\boldsymbol{\epsilon} = \{\epsilon_1,\epsilon_2,\cdots,\epsilon_M\}$ is the quadrature nodes, $\overline{L}$ is the sample mean, θ is hyper-parameter. In the second line, variance is approximated by the sample variance.

Similar to the RN case, the gradient is straight-forward by Chain Rule:

$$\begin{aligned}
G_\alpha(\alpha,\boldsymbol{\epsilon}) &= \sum_{k=1}^{M}\omega_k L(\alpha,\epsilon_k)\nabla_\alpha L(\alpha,\epsilon_k) + \frac{\theta}{M-1}\sum_{k=1}^{M}\big(L(\alpha,\epsilon_k)-\overline{L}\big)\nabla_\alpha L(\alpha,\epsilon_k)\\
&= \sum_{k=1}^{M}\left[\omega_k L(\alpha,\epsilon_k) + \frac{\theta}{M-1}\big(L(\alpha,\epsilon_k)-\overline{L}\big)\right]\nabla_\alpha L(\alpha,\epsilon_k)\\
&= \sum_{k=1}^{M}\tilde{\omega}_k\nabla_\alpha L(\alpha,\epsilon_k),
\end{aligned}$$

where the adjusted weight is $\tilde{\omega}_k = \omega_k L(\alpha,\epsilon_k) + \frac{\theta}{M-1}\big(L(\alpha,\epsilon_k)-\overline{L}\big)$.

A better approximation for variance is using the definition of variance

$$\begin{aligned}
\mathrm{Var}[L(\alpha)] &= \mathbb{E}_\epsilon\left[\Big(L(\alpha,\epsilon)-\mathbb{E}_\epsilon[L(\alpha,\epsilon)]\Big)^2\right]\\
&\approx \sum_{k=1}^{M}\omega_k\Big(L(\alpha,\epsilon_k)-\sum_{j=1}^{M}\omega_j L(\alpha,\epsilon_j)\Big)^2,
\end{aligned}$$

the loss function becomes

$$F(\alpha) \approx \sum_{k=1}^{M}\omega_k L(\alpha,\epsilon_k) + \frac{\theta}{2}\sum_{k=1}^{M}\omega_k\Big(L(\alpha,\epsilon_k)-\sum_{j=1}^{M}\omega_j L(\alpha,\epsilon_j)\Big)^2,$$

with the gradient

$$\begin{aligned}
G_\alpha(\alpha,\boldsymbol{\epsilon}) &= \sum_{k=1}^{M}\omega_k L(\alpha,\epsilon_k)\nabla_\alpha L(\alpha,\epsilon_k)\\
&\quad + \theta\sum_{k=1}^{M}\omega_k\left[L(\alpha,\epsilon_k)-\sum_{j=1}^{M}\omega_j L(\alpha,\epsilon_j)\right]\nabla_\alpha L(\alpha,\epsilon_k)\\
&= \sum_{k=1}^{M}\hat{\omega}_k\nabla_\alpha L(\alpha,\epsilon_k),
\end{aligned}$$

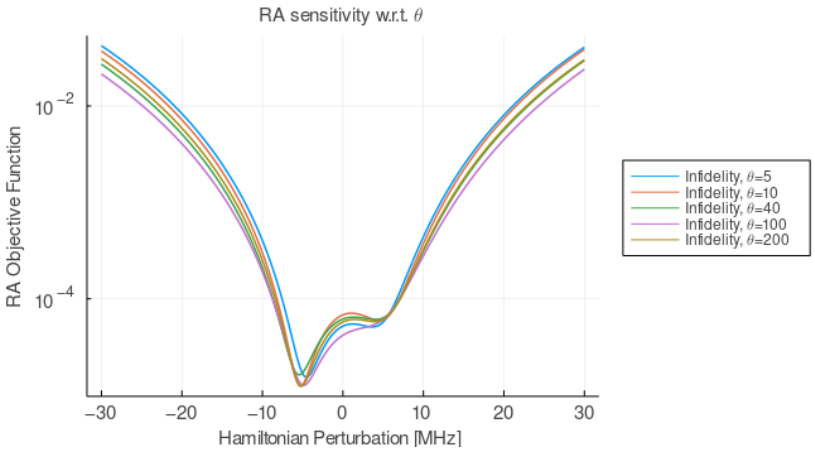


Figure 7: accuracy and robustness of RA method with different θ

where $\hat{\omega}_k = \omega_k + \theta\omega_k\left[L(\alpha,\epsilon_k) - \sum_{j=1}^{M}\omega_j L(\alpha,\epsilon_j)\right]$.

We tested how θ changes the accuracy and robustness of risk-averse method. It can be observed from Fig. 7 that when $\theta \leq 100$, the robustness of solution increases with θ, because we put more penalty on large variance. However, when $\theta \geq 100$, the influence is not obvious any more (at least not in this simple test problem).

7 Week 7 (One-dimension noise model, experiment and solution)

7.1 Compounded loss function

Our practical loss function consists of three parts:

1. Gate infidelity loss

   This loss measures how far we are from the desired state at time T. See definition in (13).

2. Leakage of population to the guard levels

   Theoretically, a qubit could have infinite energy states, and its state vector is of infinite dimensions. However, since our model only uses its finite-dimension truncation, we

would observe a "leakage of population" when $t > 0$ (non-zero probability of observing qubits in those truncated energy levels, and the norm of remaining state vector $\neq 1$).

To minimize such leakage loss, we divide the finite state vector into 1) essential levels, and 2) guard levels. In particular, we hope the population in higher guard levels to be smaller, such that it is less possible for the qubit to go to the rest truncated levels.

The leakage can be measured by the time integral of population in the guard energy state:

$$L_2(U(\alpha)) = \frac{1}{T}\int_0^T \langle U(t;\alpha)|WU(t;\alpha)\rangle_F \, dt,$$

where W is a diagonal N $\times$ N positive semi-definite weight matrix (question: is it diagonal?). The elements in W are zero for all essential states and are positive for the guard states (typically larger for higher energy levels in the model).

3. Tikhonov regularization

   This loss penalizes parameters with large amplitude.

Our original loss will be the summation of the above losses, and the utility function is applied to the whole loss for simplicity.

7.2 Model description

Our test problem is a $|0\rangle \leftrightarrow |2\rangle$ swap gate on a three-energy-level qubit, with one guard energy level. We set the gate duration to T = 300 ns, the fundamental frequency to $\omega_1/2\pi = 4.10336$ GHz, the self-Kerr coefficient to $\xi_1/2\pi = 0.2198$ GHz.

The system and control Hamiltonian (in rotating frame) is of the form (8), (11) (with Q = 1), and H_s subjecting to an uniform-distributed noise $\epsilon \sim Unif(-10, 10)$ MHz. The uncertain $H_s(\epsilon)$ is

$$H_s'(\epsilon) = H_s + H(\epsilon), \qquad \frac{H(\epsilon)}{2\pi} = \begin{bmatrix} 0 & & & \\ & \epsilon/100 & & \\ & & \epsilon/10 & \\ & & & \epsilon \end{bmatrix}.$$

By default we only use one control function $f_1(t)$ for the single-qubit system, and it is split into real and imaginary parts after rotating frame transformation R:

$$R(f(t)) = d(t) = p(t) + iq(t), \tag{33}$$

as showed in (11).

Externally we apply two carrier wave frequency ($\Omega_1 = 0$ and $\Omega_2 = -\xi$) on $p_1(t)$, leading to

$$p(t) = \sum_i p_{1,i}(t) e^{it\omega_i}, i \in 1, 2. \tag{34}$$

The same is done on $q(t)$ and we have 4 functions to optimize in total. We use 12 B-splines to approximate each function, yielding 48 parameters. In addition, we constrain the controls to start and end at zero, leading to only 32 non-zero parameters, with maximum allowable amplitude $\alpha_{max}/2\pi = 12$ MHz. Maximum iteration for gradient descent is set to 150. Gauss-Legendre quadrature with 9 nodes is used to approximate integral in the loss function, balancing accuracy and computation time.

To illustrate the robustness of optimal control given by different methods, we use the perturbed Hamiltonian $H(\epsilon)$ to evaluate the loss, for evenly spaced ϵ between $[-30, 30]$ MHz. Gate infidelity and guard level leakage are plotted separately in Fig. 8.

To control experiment variables, we did several experiments for different methods (usually the convergence situation when would be influenced by random seeds), and include the best result from each methods in the comparison.

7.3 Behavior of different methods under noise

1. Noise-perturbation plot

   From Fig. 8 we can see that noise-aware methods tends to be less accurate under trivial noise, but enjoy more robustness against larger noise levels, while the accuracy of noise-free method is high under trivial noise, and decreases exponentially as noise increases.

   Although in this simple case, RN, RS, and RA methods have similar behavior, we could still tell RS and RA methods are more robust against large noise disturbance compared with RN method. It is useful to compare their behaviors in more complex problems to get comprehensive understanding.

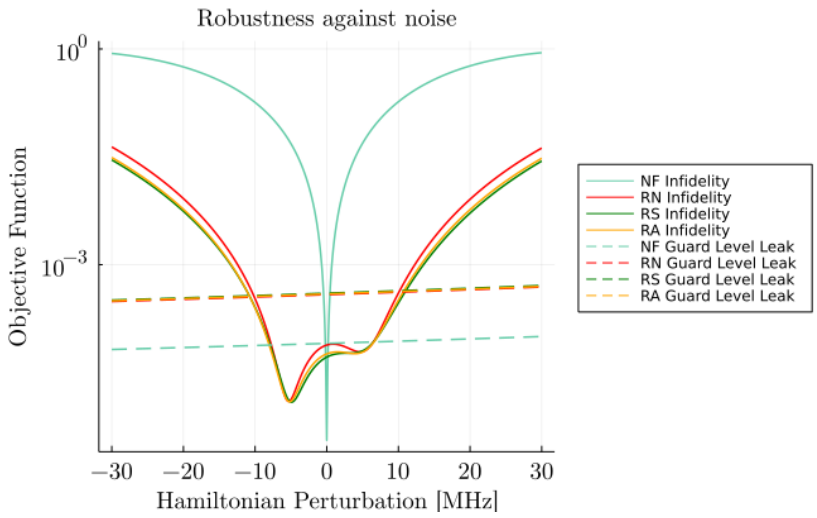


Figure 8: accuracy and robustness of noise-free, risk-neutral, risk-sensitive, and risk-averse methods under different noise level.

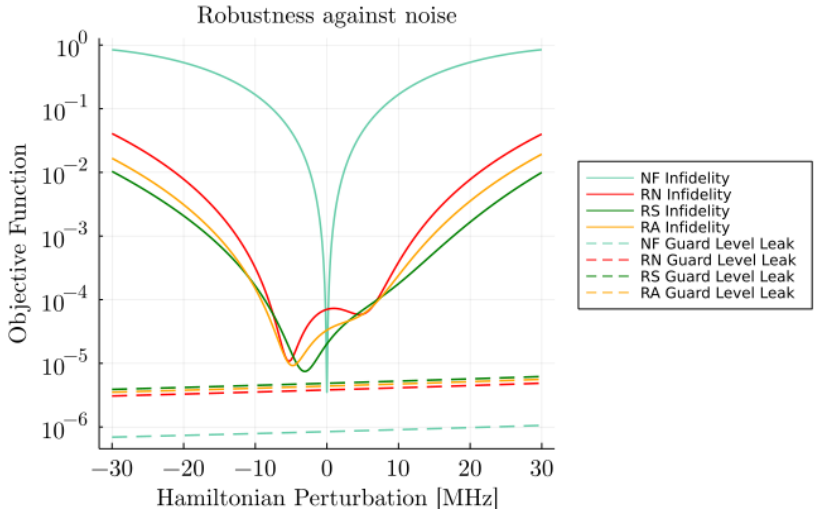
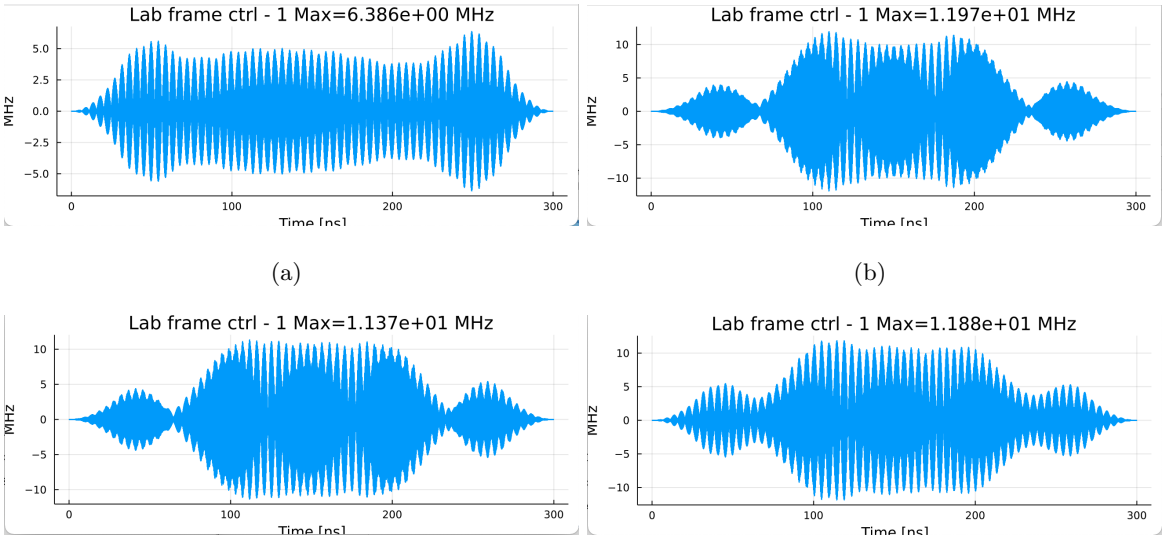


Figure 9: accuracy and robustness of noise-free, risk-neutral, risk-sensitive, and risk-averse methods under different noise level, when leakage loss is tuned down.



(a) (b)

(c) (d)

Figure 10: (a) "noise-free" (b) "risk-neutral" (c) "risk-sensitive" (d) "risk-averse" Control functions $f(t)$ in original frame, defined in 33

One conjecture for the reason of similar results is, as leakage loss is dominated the gate infidelity loss under small noise level, the re-weighting in RS and RA methods is based on infidelity loss (which are similar everywhere) instead of fidelity, suppressing the advantages of these two methods. One trial we did is to manually tune down the influence of leakage loss by $L_2* = 0.01$, the result is showed in Fig. 9.

2. Control function visualization

   Fig. 12 shows the control function in rotating frame. The control functions of noise-aware methods tend to have larger amplitudes and obvious periodicity compared with these of noise-free method. Whether these features imply worse/better stability remains to be study.

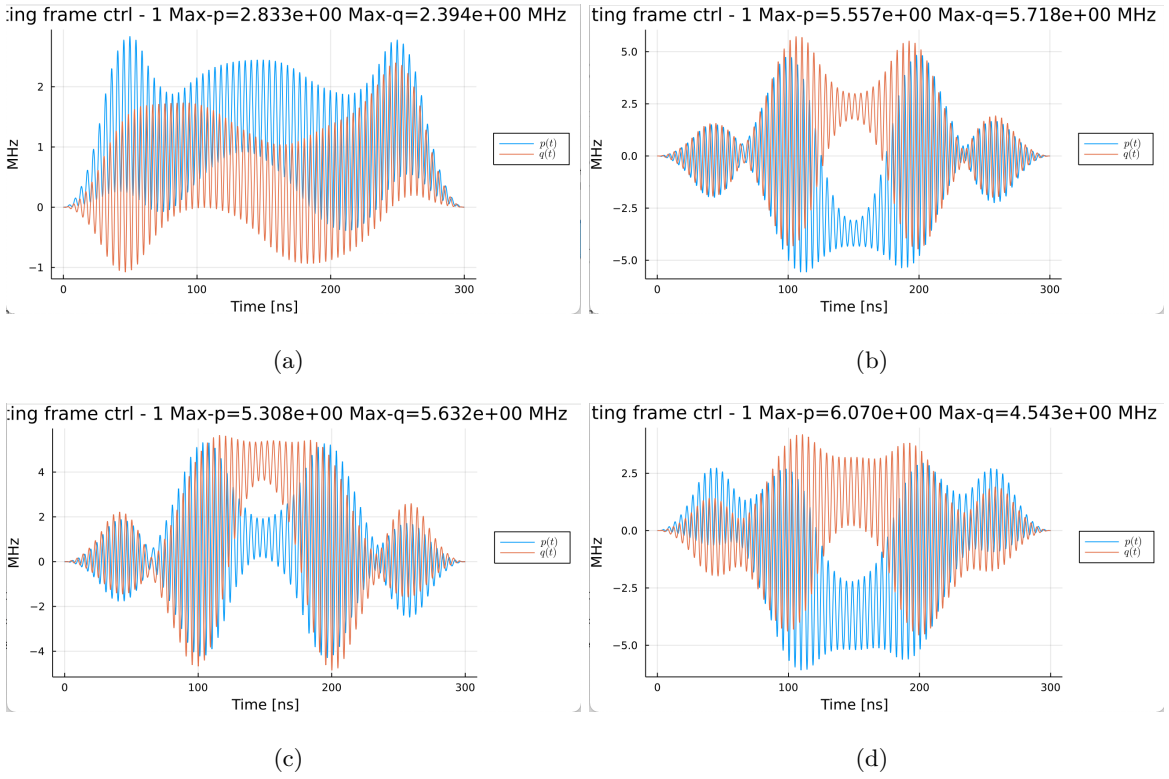


Figure 11: (a) "noise-free" (b) "risk-neutral" (c) "risk-sensitive" (d) "risk-averse" Control functions $p(t)$ and $q(t)$ in rotating frame (with carrier waves), defined in 33

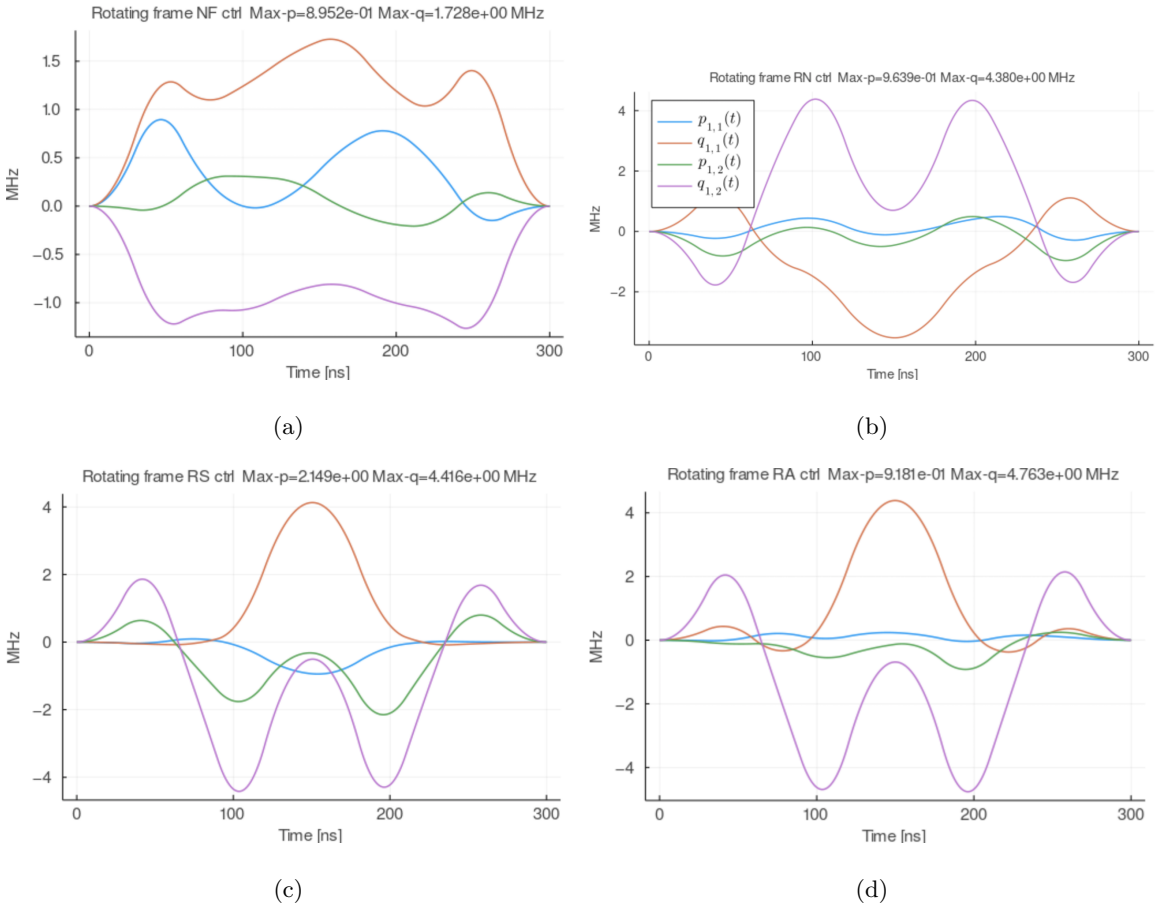


Figure 12: (a) "noise-free" (b) "risk-neutral" (c) "risk-sensitive" (d) "risk-averse"
Control functions $p_{1,i}, q_{1,i}$ in rotating frame (without carrier waves) for the cases, defined in 34

8 Week 8/9/10 (Multi-dimension noise model, experiment and solution)

These two weeks we studied more complex, higher-dimension noise models. Our starting point is the four-dimension uniform noise model showed below, applied on the rotating-frame system Hamiltonian defined in paragraph 7.2, where $\epsilon_i \sim (-10, 10)$ MHz. Higher energy levels subject to larger noise, which conforms to the reality.

$$H_s'(\epsilon) = H_s + H(\epsilon), \qquad \frac{H(\epsilon)}{2\pi} = \begin{bmatrix} \epsilon_1/10^3 & & & \\ & \epsilon_2/10^2 & & \\ & & \epsilon_3/10 & \\ & & & \epsilon_4 \end{bmatrix}.$$

There are two major changes we made to suit this model:

- Monte Carlo Sampling

  We still use discrete samples to approximate the integral in the noise-aware loss functions (27), (28), and (31). However, instead of multi-dimension Gauss quadrature which requires huge amount of data points in high dimension (exponentially grow with dimension), we randomly generate samples using Monte-Carlo method. In particular, 100 samples are used in this problem.

  A failed attempt is using 25 MC samples in a 7-dimension noise model. If we use quadrature, even a very low-order quadrature with 3 points in each dimension would yield $3^7 >> 25$ points in total. With only few samples, the MC quadrature error dominates the error completely, and we could not see any difference between different optimizations. So we cut down the noise dimension and employed more samples, which gives better experiment result.

- Standard of Comparison between Methods

  It would be harder to demonstrate comparisons as clear as in Fig. 13, since our Hamiltonian Perturbation is multi-dimension. Therefore, we plot the density of loss given by different controls under random noise. The density function is approximated by the histogram of loss from 10000 Monte-Carlo samples.

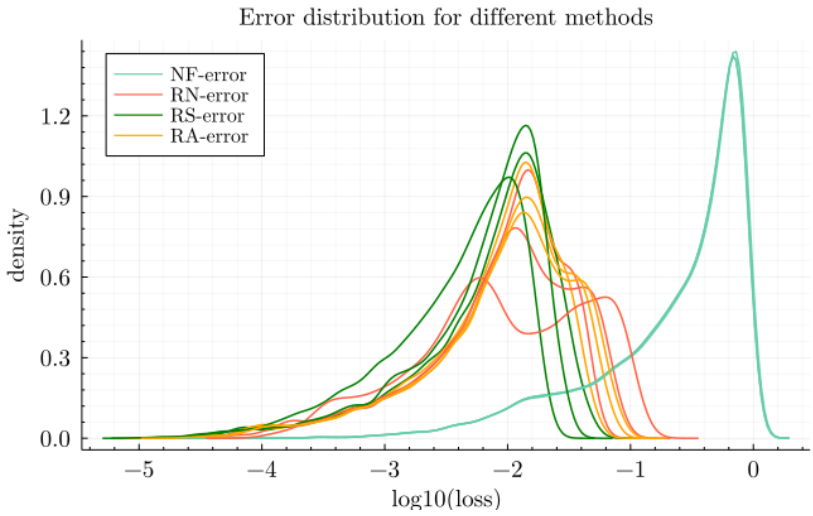


Figure 13: loss density of optimal control given by noise-free, risk-neutral, risk-sensitive, and risk-averse methods.

There is a visible difference between the deterministic optimization and the ones that take uncertainty into account. However, the differences between the noise-aware controls are less visible in Fig. 13. We think that while 100 samples is much better, there is still a rather large MC error in the optimization, which also makes a difference. In general, the MC error decays with sample size at the rate $\frac{1}{\sqrt{n}}$.

9 Week 11 (CVaR loss function)

This week we will implement the CVaR loss function (Conditional Value-at-Risk).

9.1 Introduction to VaR and CVaR

Remember the VaR (Value-at-Risk) is defined as the left-side β-quantile of the cdf H_X for some object function X:

$$\begin{aligned}\mathrm{VaR}_\beta[X] := H_X^{-1}(\beta) &= \inf_{t\in\mathbb{R}}\{t : Pr(X \le t) \ge \beta\} \\ &= \inf_{t\in\mathbb{R}}\{t : Pr(X \ge t) \le 1-\beta\}.\end{aligned}$$

Intuitively, this variable equals to the smallest possible value at which X greater than this value occur with probability at most β. Common choices of β include $0.9, 0.95$, and 0.995—-the bigger β is, the smaller risk we would like to take.

However, VaR is non-subadditive, non-convex, non-coherent, and contains no information as to the degree of exceedance of this lower bound. CVaR helps alleviate these issues.

CVaR is defined as

$$\mathrm{CVaR}_\beta[X] := \inf_{t\in\mathbb{R}}\{t + (1-\beta)^{-1}\,\mathbb{E}[(X-t)_+]\},$$

which is convex.

9.2 Smoothed CVaR function

In this section we want to analyze the smoothed version of the following object function

$$\min_{\alpha\in\mathbb{R}^n, t\in\mathbb{R}} J_\beta(\alpha, t) = t + \frac{1}{1-\beta}\,\mathbb{E}^\epsilon\left[(L_1(\alpha,\epsilon) - t)_+\right] + L_2(\alpha,\epsilon).$$

Now let us introduce the smooth function v_ϵ, and the smoothed CVaR problem

$$\min_{\alpha\in\mathbb{R}^n, t\in\mathbb{R}} J_\beta(\alpha, t) = t + \frac{1}{1-\beta}\,\mathbb{E}^\epsilon\left[v_\epsilon\left(L_1(\alpha,\epsilon) - t\right)\right] + L_2(\alpha,\epsilon).$$

We have two candidates for v_ϵ:

1. $v_{\epsilon,1} = \epsilon\log\left(1+\exp\left(\frac{x}{\epsilon}\right)\right), \qquad v'_{\epsilon,1} = \frac{e^{x/\epsilon}}{1+e^{x/\epsilon}}.$

2. $v_{\epsilon,2} = \begin{cases} 0, & x\le 0 \\ \frac{x^3}{\epsilon^2} - \frac{x^4}{2\epsilon^3}, & x\in(0,\epsilon) \\ x - \frac{\epsilon}{2}, & x\ge\epsilon \end{cases}, \qquad v'_{\epsilon,2} = \begin{cases} 0, & x\le 0 \\ \frac{3x^2}{\epsilon^2} - \frac{2x^3}{\epsilon^3}, & x\in(0,\epsilon) \\ 1, & x\ge\epsilon \end{cases}.$

when $\epsilon \le 0.001$, it almost equals to the $()_+$ function. We will use the later one with $\epsilon = 0.001$ throughout the experiment because there is serious blow-up issues with the first function.

9.3 Discretization of object function

Similar as before, we compute the expectation using the linear combination of finite samples (quadrature in low dimension, Monte-Carlo sampling in higher dimension). Assume ω_i is the

weight for the i^{th} sample, we have

$$\min_{\alpha\in\mathbb{R}^n, t\in\mathbb{R}} J_\beta(\alpha, t) \approx t + \frac{1}{1-\beta}\sum_{i=1}^{n} v_\epsilon\left(L(\alpha, \epsilon_k) - t\right)\omega_i$$

The discretized gradient for α and t are given by

$$\nabla_\alpha J = \frac{1}{1-\beta}\sum_{i=1}^{n} \omega_i \cdot v'_\epsilon\left(L(\alpha, \epsilon_k) - t\right) \cdot \nabla_\alpha L,$$

$$\partial_t J = 1 - \frac{1}{1-\beta}\sum_{i=1}^{n} \omega_i \cdot v'_\epsilon\left(L(\alpha, \epsilon_k) - t\right).$$

A Numerical methods

Our main question is to approximate

$$\boldsymbol{y}(t) = \boldsymbol{y}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(s, \boldsymbol{y}(s)s \approx \boldsymbol{y}_0 + (t - t_0)\boldsymbol{f}(t_0, \boldsymbol{y}_0), \tag{35}$$

which we could decompose into sub-questions in time interval $[t_n, t_{n+1}]$.

1. Euler's method and extension

   (a) Euler's method

   $$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n), \qquad n = 0, 1, \cdots, \tag{36}$$

   order 1, explicit.

   (b) Trapezoidal rule

   $$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{1}{2}h[\boldsymbol{f}(t_n, \boldsymbol{y}_n) + \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})], \tag{37}$$

   order 2, implicit.

   (c) Implicit midpoint rule

   $$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_n + \frac{1}{2}h, \frac{1}{2}(\boldsymbol{y}_n + \boldsymbol{y}_{n+1})), \tag{38}$$

   order 2, implicit.

   (d) Theta method

   $$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h[\theta\boldsymbol{f}(t_n, \boldsymbol{y}_n) + (1 - \theta)\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})], \tag{39}$$

   order 1 when $\theta = \frac{1}{2}$, 2 else wise, implicit.

2. Multi-step methods

   general multi-step method is in the form

   $$\sum_{m=0}^{s} a_m \boldsymbol{y}_{n+m} = h \sum_{m=0}^{s} b_m \boldsymbol{f}(t_{n+m}, \boldsymbol{y}_{n+m}), \qquad n = 0, 1, \cdots, \tag{40}$$

   when $b_s = 0$, explicit, otherwise implicit.

The method in (40) can be characterized using polynomials

$$\rho(w) := \sum_{m=0}^{s} a_m w^m \qquad and \qquad \sigma(w) := \sum_{m=0}^{s} b_m w^m. \tag{41}$$

(40) is of order $p \geq 1$ iff there exists $c \neq 0$ such that

$$\rho(w) - \sigma(w) lnw = c(w-1)^{p+1} + \mathcal{O}(|w-1|^{p+2}), \qquad w \to 1, \tag{42}$$

However, (42) is necessary but not sufficient for convergence. *The Dahlquist equivalence theorem* tells us 40 is convergent iff it satisfies (42) and ρ has all zeros reside the closed unit disc, and all zeros of unit modulus are simple (which is usually referred to as *root condition*.

Conventionally, we have a recipe to obtain orders $s+1$ and s for s-step implicit and explicit methods respectively. First choose arbitrary s-degree polynomial ρ with $\rho(1) = \sum a_m = 0$, and

$$\sigma(w) = \frac{\rho(w)}{lnw} + \mathcal{O}(|w-1|^p). \tag{43}$$

choose $\rho(w)$ to be sth-degree or $(s-1)$th-degree polynomial (s.t. $b_m = 0$), we got implicit and explicit method formulae.

(a) Adams method

This methods use $\rho(w) = w^{s-1}(w-1)$, and we have explicit *Adams-Bashforth* schemes & implicit *Adams-Moulton* schemes.

(b) Backward differentiation formulae (BDFs)

$\rho(w) = w^{s-2}(w^2-1)$ leads to explicit *Nystrom* methods and implicit *Milne* methods (convergent for $1 \leq s \leq 6$.

$$s = 2, \qquad \boldsymbol{y}_{n+2} - \frac{4}{3}\boldsymbol{y}_{n+1} + \frac{1}{3}Fy_n = \frac{2}{3}h\boldsymbol{f}(t_{n+2}, \boldsymbol{y}_{n+2}),$$
$$s = 3, \qquad \boldsymbol{y}_{n+3} - \frac{18}{11}\boldsymbol{y}_{n+2} + \frac{9}{11}Fy_{n+1} - \frac{2}{11}Fy_n = \frac{6}{11}h\boldsymbol{f}(t_{n+3}, \boldsymbol{y}_{n+3}),$$
$$1 \leq s \leq 6,$$
$$\beta = \Big(\sum_{m=1}^{s} \frac{1}{m}\Big)^{-1} \quad and \quad \rho(w) = \beta \sum_{m=1}^{s} \frac{1}{m} w^{s-m}(w-1)^m.$$

3. linear stability domain

   Consider the scalar linear equation

$$y' = \lambda y, \quad t \leq 0, \qquad y(0) = 1, \tag{44}$$

   The linear stability domain $\mathbb{D}$ of a specific numerical method is the set of $h\lambda \in \mathbb{C}$ such that $lim_{n\to \inf} y_n = 0$.

4. Stiff equations and solutions

   An ODE system is *stiff* if its numerical solution by some methods requires a significant depression of the step size to avoid instability.

   In short, given a linear system of ODEs

$$y' = Ay, \tag{45}$$

   if $r = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$ is large, usually the system is stiff.

References

[1] Juqbox. `https://github.com/LLNL/Juqbox.jl`. Accessed: 2022-06-20.

[2] Xiaozhen Ge and Re-Bing Wu. Risk-sensitive optimization for robust quantum controls. *Physical Review A*, 104(1), jul 2021.

[3] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[4] N. Anders Petersson and Fortino Garcia. Optimal control of closed quantum systems via b-splines with carrier waves, 2021.

[5] N. Anders Petersson, Fortino M. Garcia, Austin E. Copeland, Ylva L. Rydin, and Jonathan L. DuBois. Discrete adjoints for accurate numerical optimization with application to quantum control, 2020.