Numerical Methods of Optimal Quantum Control

Michael H. Goerz

DEVCOM Army Research Lab

QuCS Lecture Series, August 24, 2023

What is Quantum Control?

Steer a quantum system in some desired way

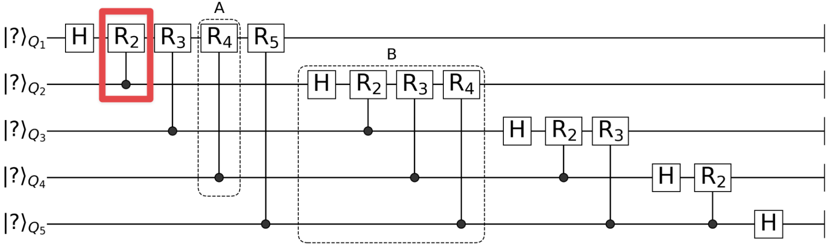
UNCLASSIFIED

Quantum Gates

docs.yaoquantum.org

Quantum Fourier Transformation

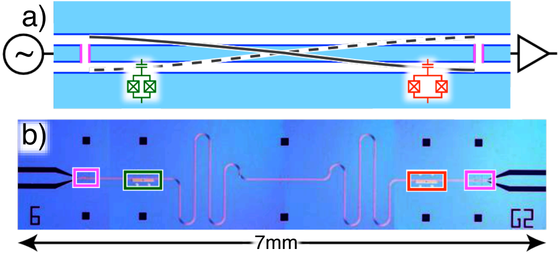
The Quantum Fourier Transformation (QFT) circuit is to repeat two kinds of blocks repeatly:



Quantum Fourier Transformation circuit of size 5

The basic building block control phase shift gate is defined as

Two-Transmon Gate

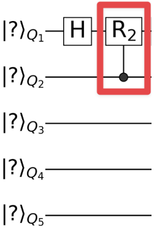


Majer *et al.* Nature 449, 443 (2007)

$$\hat{H} = \hat{H}_0 + \epsilon(t)\hat{H}_1$$

$\uparrow$

microwave field in transmission line

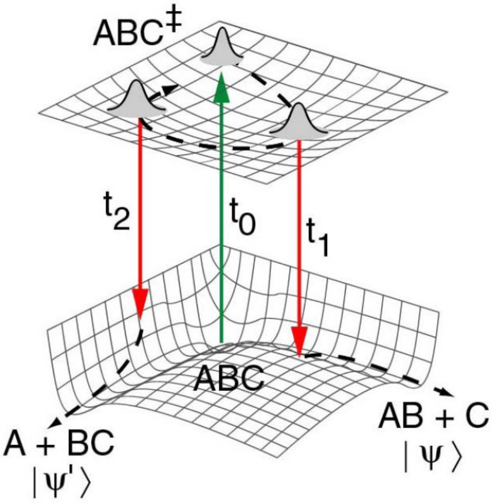


$$\left.\begin{aligned} |00\rangle &\to \mathrm{CR}_2\,|00\rangle \\ |01\rangle &\to \mathrm{CR}_2\,|01\rangle \\ |10\rangle &\to \mathrm{CR}_2\,|10\rangle \\ |11\rangle &\to \mathrm{CR}_2\,|11\rangle \end{aligned}\right\}$$

with the same $\epsilon(t)$;

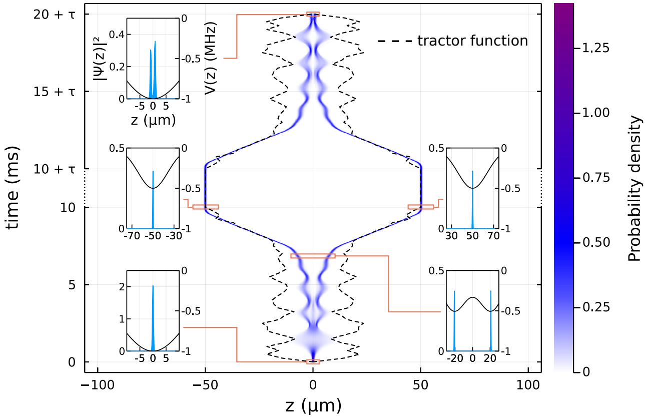
acting on logical subspace

Controlling photo-chemical reactions



- Kosloff, Rice, et. al. *Wavepacket dancing: Achieving chemical selectivity by shaping light pulses*. Chem. Phys. 139, 201 (1989).
- Tannor, Jin. *Design of femtosecond pulse sequences to control photochemical products*, in *Mode Selective Chemistry* (Springer, 1991)
- Shi, Rabitz. *Optimal control of bond selectivity in unimolecular reactions*. Comput. Phys. Commun. 63, 71 (1991)
- Judson, Rabitz. *Teaching lasers to control molecules*. Phys. Rev. Lett. 68, 1500 (1992).

Tractor atom interferometry



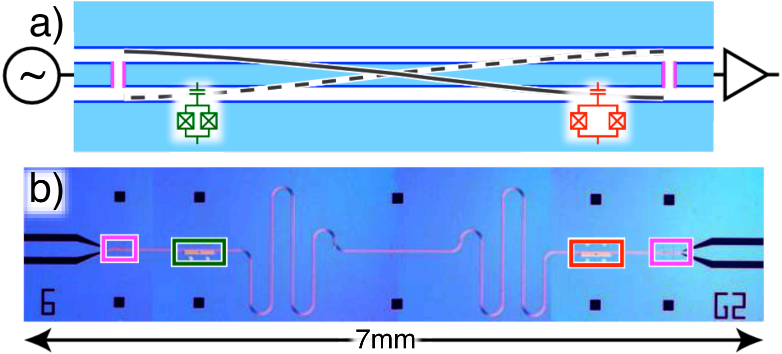
Raithel *et al.* Quantum Sci. Technol. 8, 014001 (2022)

Find non-adiabatic tractor potential closing interferometric path

Outline

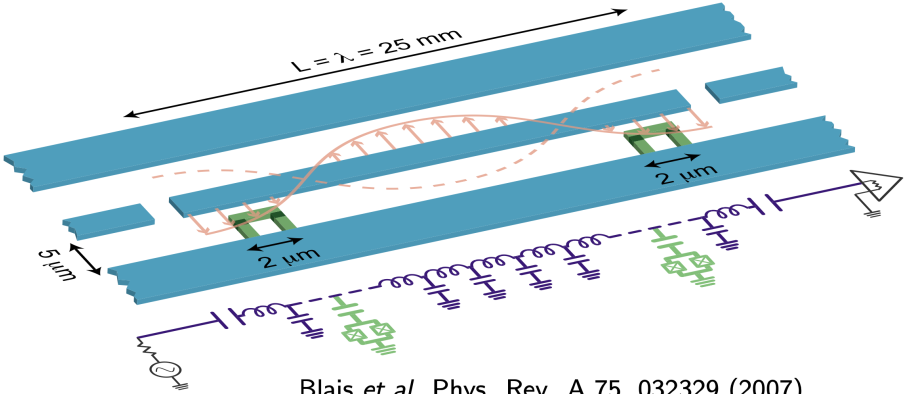
- Formulating the control problem
  - Quantum gates with coupled transmon qubits
  - Unconstrained control problems
- Gradient-ascent (GRAPE)
  - Simulating time dynamics
  - Evaluating gradients
  - Semi-automatic differentiation: evaluate arbitrary functionals
  - Example: Maximizing gate entanglement
- Krotov's method
- QuantumControl.jl: efficiently implementing quantum control
- Parametrized and constrained control problems

Quantum gates with coupled transmon qubits

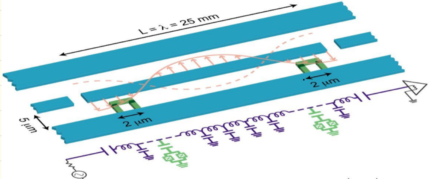


Majer *et al.* Nature 449, 443 (2007)

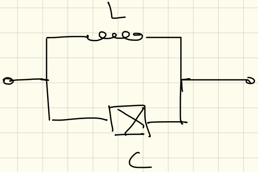
Quantum gates with coupled transmon qubits



Blais *et al.* Phys. Rev. A 75, 032329 (2007)



Blais *et al.* Phys. Rev. A 75, 032329 (2007)



$$
\begin{aligned}
\hat{H} = {} & \omega_1 \hat{n}_1 - \frac{\alpha_1}{2}\left(\hat{n}_1 - \hat{n}_1^2\right) \\
& + \omega_2 \hat{n}_2 - \frac{\alpha_2}{2}\left(\hat{n}_2 - \hat{n}_2^2\right) \\
& + g\left(\hat{b}_1^+ \hat{b}_2 + \hat{b}_1 \hat{b}_2^+\right) \\
& + \varepsilon(t)\left[\hat{b}_1 + \hat{b}_1^+ + \lambda \hat{b}_2 + \lambda \hat{b}_2^+\right]
\end{aligned}
$$

$$\hat{n}_1 = \hat{b}_1^+ \hat{b}$$

$\omega_1 = 4.4$ GHz

$\omega_2 = 4.6$ GHz

$\alpha_1 = 210$ MHz

$\alpha_2 = 215$ MHz

Rotating wave approximation:

$$\varepsilon(t) = \Omega(t) \cdot \cos(\omega_d t) \qquad \omega_d = 4.5\,\text{GHz}$$

$$\cos(\omega_d t) = \frac{1}{2}\left(e^{i\omega_d t} + e^{-i\omega_d t}\right)$$

$$\tilde{\omega}_{1,2} = \omega_{1,2} - \omega_d$$

$$\begin{aligned}\hat{H} = {} & \tilde{\omega}_1 \hat{n}_1 - \frac{\alpha_1}{2}\left(\hat{n}_1 - \hat{n}_1^2\right) \\ & + \tilde{\omega}_2 \hat{n}_2 - \frac{\alpha_2}{2}\left(\hat{n}_2 - \hat{n}_2^2\right) \\ & + J\left(\hat{b}_1^+ \hat{b}_2 + \hat{b}_1 \hat{b}_2^+\right) \\ & + \frac{\Omega_{re}(t)}{2}\left[\hat{b}_1 + b_1^+ + \lambda \hat{b}_2 + \lambda b_2^+\right] \\ & + i\frac{\Omega_{im}(t)}{2}\left[b_1^+ - \hat{b}_1 + \lambda b_2^+ - \lambda b_2\right]\end{aligned}$$

2 controls!

Logical Subspace

$$\hat{b}_1 = \sum_{n=1}^{\infty} \sqrt{n}\, |n-1\rangle\langle n|$$

$|0\rangle, |1\rangle$ — logical subspace

embedded in larger Hilbert space,

$|0\rangle, |1\rangle, |2\rangle, |3\rangle, \dots$ (truncate)

Optimization Functional

$$J(\{\epsilon_k(t)\}) = J_T(\{|\Psi_k(T)\rangle\}) + \cancel{\int_0^T g_a(\{\epsilon_k(t)\})\,dt} + \cancel{\int_0^T g_b(\{\Psi_k(t)\})\,dt}$$

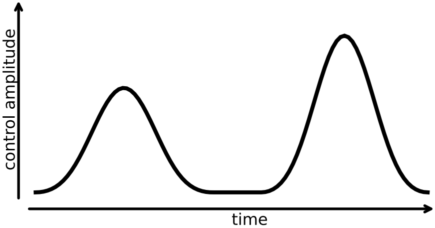
CNOT Gate $\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{pmatrix}$

$|00\rangle \to |00\rangle$, $|01\rangle \to |01\rangle$,

$|10\rangle \to |11\rangle$, $|11\rangle \to |10\rangle$

$$\boxed{J_T = 1 - \left| \frac{1}{4} \sum_k \langle \Psi_k(T) | \Psi_k^{tgt} \rangle \right|^2}$$

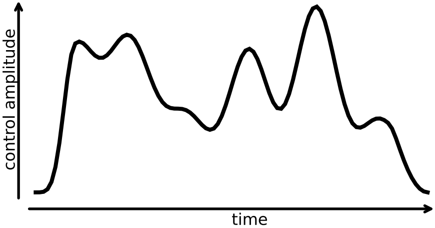
Time Discretization



$$i\hbar\frac{\partial}{\partial t}\left|\Psi(t)\right\rangle = \hat{\mathsf{H}}(\{\epsilon_l(t)\})\left|\Psi(t)\right\rangle$$

$$i\hbar\frac{\partial}{\partial t}\hat{\rho}(t) = \mathcal{L}(\{\epsilon_l(t)\})[\hat{\rho}(t)]$$

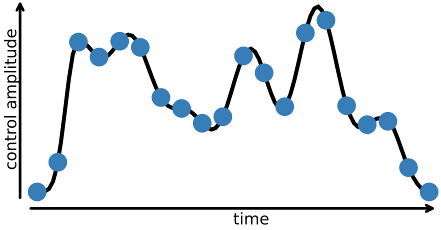
Time Discretization



$$i\hbar\frac{\partial}{\partial t}\left|\Psi(t)\right\rangle = \hat{\mathsf{H}}(\{\epsilon_l(t)\})\left|\Psi(t)\right\rangle$$

$$i\hbar\frac{\partial}{\partial t}\hat{\rho}(t) = \mathcal{L}(\{\epsilon_l(t)\})[\hat{\rho}(t)]$$

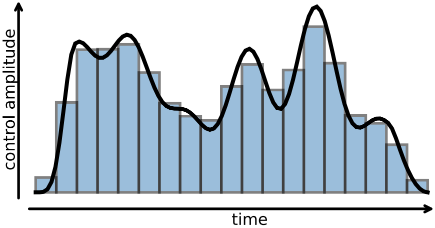
Time Discretization



$$i\hbar\frac{\partial}{\partial t}\left|\Psi(t)\right\rangle = \hat{\mathsf{H}}(\{\epsilon_l(t)\})\left|\Psi(t)\right\rangle$$

$$i\hbar\frac{\partial}{\partial t}\hat{\rho}(t) = \mathcal{L}(\{\epsilon_l(t)\})[\hat{\rho}(t)]$$

Time Discretization



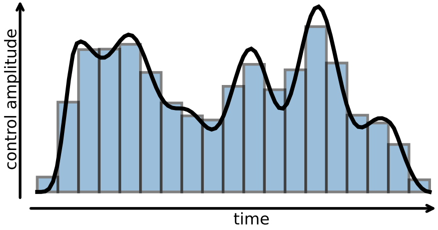
$$i\hbar\frac{\partial}{\partial t}\left|\Psi(t)\right\rangle = \hat{\mathrm{H}}(\{\epsilon_l(t)\})\left|\Psi(t)\right\rangle$$

$$i\hbar\frac{\partial}{\partial t}\hat{\rho}(t) = \mathcal{L}(\{\epsilon_l(t)\})[\hat{\rho}(t)]$$

Piecewise constant: $\hat{\mathrm{H}}_n = \hat{\mathrm{H}}(\{\epsilon_{nl}\})$ with $\epsilon_{nl} = \epsilon_l(t = t_n)$ for n'th time slice

$$J(\{\epsilon_l(t)\}) = J_T(\{|\Psi_k(T)\rangle\}) + \int_0^T \dots dt$$

Time Discretization



$$i\hbar \frac{\partial}{\partial t} \left|\Psi(t)\right\rangle = \hat{\mathrm{H}}(\{\epsilon_l(t)\}) \left|\Psi(t)\right\rangle$$

$$i\hbar \frac{\partial}{\partial t} \hat{\rho}(t) = \mathcal{L}(\{\epsilon_l(t)\})[\hat{\rho}(t)]$$

Piecewise constant: $\hat{\mathrm{H}}_n = \hat{\mathrm{H}}(\{\epsilon_{nl}\})$ with $\epsilon_{nl} = \epsilon_l(t = t_n)$ for n'th time slice

$$J(\{\epsilon_{nl}\}) = J_T(\{|\Psi_k(T)\rangle\}) + \int_0^T \dots dt$$

Gradient $\nabla J \equiv \frac{\partial J}{\partial \epsilon_{nl}}$ $\Rightarrow$ LBFGS

Gradient Ascent Pulse Engineering (GRAPE)

How to calculate ∇J for PWC controls

Khaneja, Reiss, Kehlet, Schulte-Herbrüggen, Glaser. *Optimal control of coupled spin dynamics: Design of NMR pulse sequences by gradient ascent algorithms*. J. Magnet. Res. 172, 296 (2005)

GRAPE Scheme

$$J_T = 1 - \left| \frac{1}{4} \sum_k \underbrace{\langle \Psi_k(T) \vert \Psi_k^{(tgt)} \rangle}_{\equiv \tau_k} \right|^2$$

$$= 1 - \frac{1}{16} \sum_{kk'} \tau_{k'}^* \tau_k$$

$$\frac{\partial J_T}{\partial \epsilon_{nl}} = -\frac{1}{16} \sum_{kk'} \left(\frac{\partial \tau_{k'}^*}{\partial \epsilon_{nl}} \tau_k + \tau_{k'}^* \frac{\partial \tau_k}{\partial \epsilon_{nl}} \right)$$

$$= -\frac{2}{16} \operatorname{Re} \left[\sum_{kk'} \tau_{k'}^* \frac{\partial \tau_k}{\partial \epsilon_{nl}} \right]$$

Aside: Wirtinger derivatives — derivatives w.r.t. complex numbers

$$J_T(\{\tau_k\}) = J_T(\{\mathrm{Re}[\tau_k], \mathrm{Im}[\tau_k]\}); \qquad J_T \in \mathbb{R}, \quad \tau_k \in \mathbb{C}$$

$$\frac{\partial J_T(\{\tau_k\})}{\partial \epsilon_{nl}} = \sum_k \left(\frac{\partial J_T}{\partial \mathrm{Re}[\tau_k]} \frac{\partial \mathrm{Re}[\tau_k]}{\partial \epsilon_{nl}} + \frac{\partial J_T}{\partial \mathrm{Im}[\tau_k]} \frac{\partial \mathrm{Im}[\tau_k]}{\partial \epsilon_{nl}} \right); \qquad \epsilon_{nl} \in \mathbb{R}$$

Define
$$\frac{\partial J_T(\{\tau_k\})}{\partial \tau_k} \equiv \frac{1}{2} \left(\frac{\partial J_T}{\partial \mathrm{Re}[\tau_k]} - i \frac{\partial J_T}{\partial \mathrm{Im}[\tau_k]} \right)$$
$$\frac{\partial J_T(\{\tau_k\})}{\partial \tau_k^*} \equiv \frac{1}{2} \left(\frac{\partial J_T}{\partial \mathrm{Re}[\tau_k]} + i \frac{\partial J_T}{\partial \mathrm{Im}[\tau_k]} \right) = \left(\frac{\partial J_T}{\partial \tau_k} \right)^*$$

$$\frac{\partial J_T(\{\tau_k\})}{\partial \epsilon_{nl}} = \sum_k \left(\frac{\partial J_T}{\partial \tau_k} \frac{\partial \tau_k}{\partial \epsilon_{nl}} + \frac{\partial J_T}{\partial \tau_k^*} \frac{\partial \tau_k^*}{\partial \epsilon_{nl}} \right) = 2\mathrm{Re} \left[\sum_k \frac{\partial J_T}{\partial \tau_k} \frac{\partial \tau_k}{\partial \epsilon_{nl}} \right]$$

Gradient of State Overlap

$$\frac{\partial}{\partial \epsilon_{nl}} \tau_k = \frac{\partial}{\partial \epsilon_{nl}} \langle \Psi_k(T) \vert \Psi_k^{tgt} \rangle$$

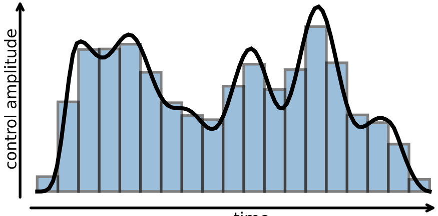
$$\vert \Psi_k(T) \rangle = \hat{U}_N \dots \hat{U}_2 \hat{U}_1 \vert \Psi_k(0) \rangle$$

$$\frac{\partial \tau_k}{\partial \epsilon_{nl}} = \frac{\partial}{\partial \epsilon_{nl}} \langle \Psi_k(0) \vert \hat{U}_1^{\dagger} \hat{U}_2^{\dagger} \dots \hat{U}_n^{\dagger} \dots \hat{U}_N^{\dagger} \vert \Psi_k^{tgt} \rangle$$

$$= \underbrace{\langle \Psi_k(0) \vert \hat{U}_1^{\dagger} \dots \hat{U}_{n-1}^{\dagger}}_{\langle \Psi_k(t_n) \vert} \frac{\partial \hat{U}_n^{\dagger}}{\partial \epsilon_{nl}} \underbrace{\hat{U}_{n+1}^{\dagger} \dots \hat{U}_N^{\dagger} \vert \Psi_k^{tgt} \rangle}_{\vert \chi_k(t_{n+1}) \rangle}$$

forward-prop　　　　　bw-prop

Piecewise-constant time propagation



$$i\hbar\frac{\partial}{\partial t}\left|\Psi(t)\right\rangle = \hat{\mathrm{H}}(\{\epsilon_l(t)\})\left|\Psi(t)\right\rangle$$

$$i\hbar\frac{\partial}{\partial t}\hat{\rho}(t) = \mathcal{L}(\{\epsilon_l(t)\})[\hat{\rho}(t)]$$

PWC propagator: $\hat{\mathrm{U}}_n = \exp[-\frac{i}{\hbar}\hat{\mathrm{H}}_n dt]$ for n'th time slice

$\Rightarrow$ evaluate $\hat{\mathrm{U}}_n \left|\Psi\right\rangle$ (or $\mathcal{U}_n[\hat{\rho}]$) as a polynomial expansion

- Hermitian Hamiltonian $\rightarrow$ Chebychev polynomials
- Non-Hermitian Hamiltonian or Liouvillian $\rightarrow$ Newton polynomials

Chebychev Propagation

Chebychev Polynomials

$$P_0(x) = 1; \quad P_1(x) = x; \quad P_n(x) = 2xP_n(x) - P_{n-1}(x)$$

$P_n(x)$ are defined for $x \in [-1, 1]$

$$|\Psi(t+dt)\rangle = e^{-\mathrm{i}\hat{\mathrm{H}}\,dt}\,|\Psi(t)\rangle = \sum_n a_n \underbrace{P_n(-\mathrm{i}\hat{\mathrm{H}}_{\text{norm}})\,|\Psi(t)\rangle}_{\equiv|\Phi_n\rangle},$$

$$\hat{\mathrm{H}}_{\text{norm}} = 2\frac{\hat{\mathrm{H}} - E_{\text{min}}\,\mathbf{1}}{\Delta} - \mathbf{1}\,, \qquad a_n = (2-\delta_{n0})e^{-\frac{\mathrm{i}}{\hbar}\left(\frac{\Delta}{2}+E_{\text{min}}\right)dt} J_k(\alpha)\,,$$

$$|\Phi_0(x)\rangle = |\Psi(t)\rangle\,; \quad |\Phi_1(x)\rangle = -\mathrm{i}\hat{\mathrm{H}}_{\text{norm}}\,|\Phi_0\rangle\,; \quad |\Phi_n(x)\rangle = -2\mathrm{i}\hat{\mathrm{H}}_{\text{norm}}\,|\Phi_{n-1}\rangle + |\Phi_{n-2}\rangle$$

Chebychev Propagation – Pseudocode

Algorithm 2 CHEBYCHEV-PROPAGATOR Evaluate $\vec{w} = f(\pm\hat{A}\,dt)\vec{v}$, with $f(\pm\hat{A}\,dt) = \mathrm{e}^{\pm\mathrm{i}\hat{A}\,dt}$.

Input: input vector $\vec{v} \in \mathbb{C}^N$; operator $\hat{A} \in \mathbb{C}^{N\times N}$; time step dt;
Output: Approximation of propagated vector $\vec{w} = e^{-i\hat{A}\,dt}\vec{v} \in \mathbb{C}^N$

procedure CHEBY($\vec{v}$, $\hat{A}$, dt)
    Δ = spectral radius of $\hat{A}$
    $E_{\min}$ = minimum eigenvalue of $\hat{A}$
    $[a_0 \dots a_n] =$ EXPCHEBYCOEFFS(Δ, $E_{\min}$, dt)
    $d = \frac{1}{2}\Delta$; $\beta = d + E_{\min}$
    $\vec{v}_0 = \vec{v}$
    $\vec{w}^{(0)} = a_0\vec{v}_0$
    $\vec{v}_1 = \pm\frac{\mathrm{i}}{d}\left(\hat{A}\vec{v}_0 - \beta\vec{v}_0\right)$
    $\vec{w}^{(1)} = \vec{w}^{(0)} + a_1\vec{v}_1$
    for $i = 2 : n$ **do**
        $\vec{v}_i = \pm\frac{2\mathrm{i}}{d}\left(\hat{A}\vec{v}_{i-1} - \beta\vec{v}_{i-1}\right) + \vec{v}_{i-2}$
        $\vec{w}^{(i)} = \vec{w}^{(i-1)} + a_i\vec{v}_i$
    end for
    return $e^{\pm\mathrm{i}\,\beta\,dt}\vec{w}^{(n)}$
end procedure

Algorithm 3 CHEBYCHEVCOEFFICIENTS for $f(\pm\hat{A}\,dt) = e^{\pm i\hat{A}\,dt}$.

Input: spectral radius Δ of $\hat{A}$; minimum eigenvalue $E_{\min}$ of $\hat{A}$; time step dt
Output: Array of Chebychev coefficients $[a_0 \dots a_n]$ allowing to approximate $f(\hat{A}\,dt)$ to pre-defined precision.

procedure EXPCHEBYCOEFFS(Δ, $E_{\min}$, dt)
    $\alpha = \frac{1}{2}\Delta\,dt$
    $a_0 = J_0(\alpha)$ ▷ 0'th order Bessel-function of first kind
    for $i = 1 : n_{\max} \approx 4\lfloor\alpha\rfloor$ **do**
        $a_i = 2J_i(\alpha)$ ▷ i'th order Bessel-function of first kind
        if $|a_i| <$ limit **then** exit loop with $n = i$
    end for
    return $[a_0, \dots a_n]$
end procedure

Goerz, PhD Thesis, Appendix F

`https://michaelgoerz.net`

https://github.com/JuliaQuantumControl/QuantumPropagators.jl

Default

```julia
function cheby!(Ψ, H, dt, wrk; kwargs...)

    E_min = get(kwargs, :E_min, wrk.E_min)
    check_normalization = get(kwargs, :check_normalization, false)

    Δ = wrk.Δ
    β::Float64 = (Δ / 2) + E_min  # "normfactor"
    @assert abs(dt) ≈ abs(wrk.dt) "wrk was initialized for dt=$(wrk.dt), not dt=$dt"
    if dt > 0
        c = -2im / Δ
    else
```

src/cheby.jl

```julia
    for i = 3:wrk.n_coeffs
        # v2 = -2i * H_norm * v1 + v0 = c * (H * v1 - β * v1) + v0
        mul!(v2, H, v1)
        axpy!(-β, v1, v2)
        lmul!(c, v2)
        # v2 += v0
        axpy!(true, v0, v2)
        # Ψ += a[i] * v2
        axpy!(a[i], v2, Ψ)
        v0, v1, v2 = v1, v2, v0  # switch w/o copying
    end
    lmul!(exp(-im * β * dt), Ψ)
```

```
N  68% ¶ 161/236: 1⟩ α   src/cheby.jl                                  ⟨julia⟨master
/mul!(v2, H, v1)                                                          [1/1]
⟨0:nvim⟩                                               ⟨08/23 01:30 ⟨ophelia(jqc)⟩
```

Gradient of Time Evolution Operator

$$
\begin{pmatrix}
\frac{\partial \hat{\mathrm{U}}_n^\dagger}{\partial \epsilon_{n1}} \vert\chi_k(t_n)\rangle \\
\vdots \\
\frac{\partial \hat{\mathrm{U}}_n^\dagger}{\partial \epsilon_{nL}} \vert\chi_k(t_n)\rangle \\
\hat{\mathrm{U}}_n^\dagger \vert\chi_k(t_n)\rangle
\end{pmatrix}
= \exp\left[-\mathrm{i}
\begin{pmatrix}
\hat{\mathrm{H}}_n^\dagger & 0 & \dots & 0 & \hat{\mathrm{H}}_n^{(1)\dagger} \\
0 & \hat{\mathrm{H}}_n^\dagger & \dots & 0 & \hat{\mathrm{H}}_n^{(2)\dagger} \\
\vdots & & \ddots & & \vdots \\
0 & 0 & \dots & \hat{\mathrm{H}}_n^\dagger & \hat{\mathrm{H}}_n^{(L)\dagger} \\
0 & 0 & \dots & 0 & \hat{\mathrm{H}}_n^\dagger\,,
\end{pmatrix}
dt_n \right]
\begin{pmatrix}
0 \\ \vdots \\ 0 \\ \vert\chi_k(t_n)\rangle
\end{pmatrix}
$$

$$
\hat{\mathrm{U}}_n = \exp[-\mathrm{i}\hat{\mathrm{H}}_n dt_n]\,; \qquad \hat{\mathrm{H}}_n^{(l)} = \frac{\partial \hat{\mathrm{H}}_n}{\partial \epsilon_l(t)}
$$

— Goodwin, Kuprov, J. Chem. Phys. 143, 084113 (2015)

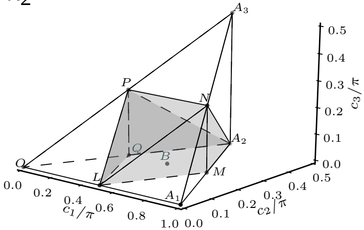
`https://github.com/JuliaQuantumControl/QuantumGradientGenerators.jl`

Optimizing for a Maximally Entangling Gate

Cartan decomposition

$$\hat{\mathsf{U}} = \hat{\mathsf{k}}_1 \exp\left[\frac{\mathrm{i}}{2}\left(c_1 \hat{\sigma}_x\hat{\sigma}_x + c_2 \hat{\sigma}_y\hat{\sigma}_y + c_3 \hat{\sigma}_z\hat{\sigma}_z\right)\right]\hat{\mathsf{k}}_2$$

$\hat{\mathsf{k}}_{1,2}$: Single qubit gates; $c_{1,2,3}$: Weyl chamber coordinates



Gate concurrence of two-qubit gate $\hat{\mathsf{U}}$

1. $c_1, c_2, c_3 \propto \text{eigvals}\left(\hat{\mathsf{U}}\tilde{\mathsf{U}}\right)$; $\tilde{\mathsf{U}} = (\hat{\sigma}_y \otimes \hat{\sigma}_y)\,\hat{\mathsf{U}}\,(\hat{\sigma}_y \otimes \hat{\sigma}_y)$
2. $C(\hat{\mathsf{U}}) = \max\left|\sin(c_{1,2,3} \pm c_{3,1,2})\right|$

Childs *et al.* Phys. Rev. A 68, 052311 (2003)

Not analytic!

Automatic differentiation (AD)

- Build computational graph for time propagation
- Elementary operations have known derivatives
- Let computer apply chain rule at each node in graph
- Backward pass to accumulate gradient

— Leung *et al.* Phys. Rev. A 95, 042318 (2017)

— Abdelhafez *et al.*, Phys. Rev. A 99, 052327 (2019)

— Schäfer, *et al.* Mach. Learn.: Sci. Technol. 1, 035009 (2020)

— Abdelhafez *et al.* Phys. Rev. A 101, 022321 (2020)

Automatic differentiation (AD)

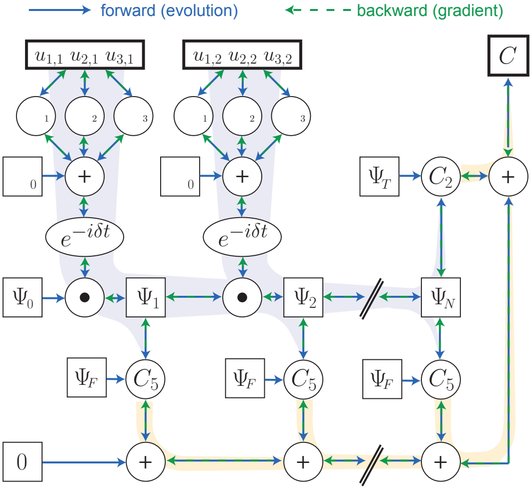


Fig. 2 in Leung *et al.* Phys. Rev. A 95, 042318 (2017)

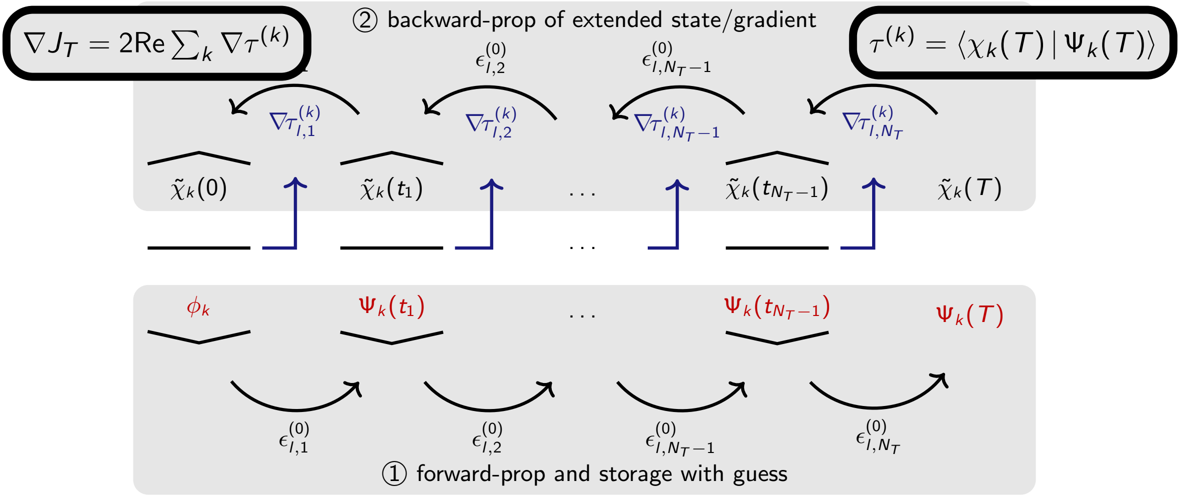
Semi - Automatic Differentiation

$$\nabla J_T = \frac{\partial J_T(\{|\Psi_k(T)\rangle\})}{\partial \epsilon_{nl}} \qquad |\Psi_k\rangle \mathrel{\hat{=}} z \qquad \langle\Psi_k| \mathrel{\hat{=}} z^*$$

$$= 2 \operatorname{Re} \sum_k \underbrace{\frac{\partial J_T}{\partial |\Psi_k(T)\rangle}}_{\equiv \langle \chi_k|} \cdot \frac{\partial |\Psi_k(T)\rangle}{\partial \epsilon_{nl}} \qquad ; \quad |\chi_k\rangle = \frac{\partial J_T}{\partial \langle \Psi_k(T)|}$$

$$= 2 \operatorname{Re} \sum_k \frac{\partial}{\partial \epsilon_{nl}} \langle \chi_k | \Psi_k(T)\rangle$$

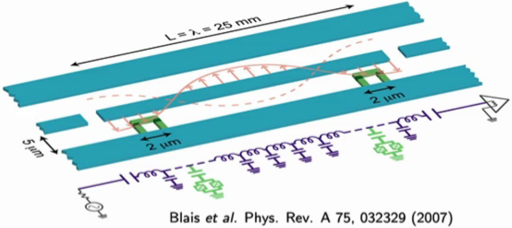
Generalized GRAPE scheme



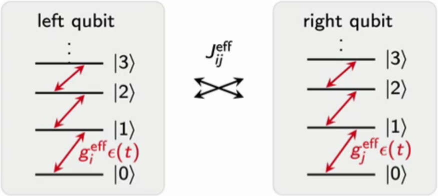
— Goerz *et al.* Quantum 6, 871 (2022)

Example: Optimization of Perfectly Entangling Quantum gate

Two Transmon qubits with a shared transmission line ¶



Blais *et al.* Phys. Rev. A 75, 032329 (2007)



Goerz *et al.* EPJ Quantum Tech. 2, 21 (2015)
Goerz *et al.* npj Quantum Information 3, 37 (2017)

Hamiltonian

https://github.com/JuliaQuantumControl/JuliaCon2023-Demo

GRAPE: discretize first, then calculate gradient

GRAPE: discretize first, then calculate gradient

Alternative: variational calculus $\frac{\partial J}{\partial \epsilon(t)}$ — *then* discretize

- Adjoint method: add TDSE as constraint with Lagrage multiplier $\langle \chi_k |$
  - Shi, Rabitz, J. Chem. Phys. 92, 364 (1990)
  - Zhu, Botina, Rabitz, J. Chem. Phys. 108, 1953 (1998)
- Krotov's method: constructive approach
  - Krotov, Feldman, Eng. Cybern. 21, 123 (1983)
  - Tannor, Kazakov, Orlov. In *Time-dependent quantum molecular dynamics* (1992)
  - Reich, Ndong, Koch. J. Chem. Phys. Physics 136, 104103 (2012)
  - Goerz *et al.* SciPost Phys. 7, 080 (2019) [Python implementation]

Krotov's Method

$$J = J_T(\{|\Psi_k(T)\rangle\}) + \int g_a(\{\epsilon_l(t)\})\,dt + \int g_b(\{|\Psi(t)\rangle\})\,dt$$

- given: guess $\epsilon_l^{(0)}(t)$
- necessary and sufficient conditions for new field $\epsilon_l^{(1)}(t)$

  so that $J(\{\epsilon_l^{(1)}(t)\}) \leq J(\{\epsilon_l^{(0)}(t)\})$

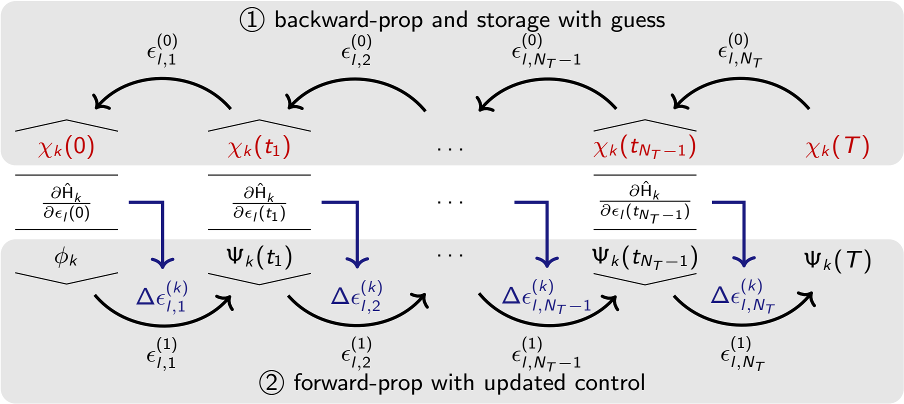
$$\frac{\partial g_a}{\partial \epsilon^{(1)}} = 2\,\mathrm{Im} \sum_k \langle \chi_k^{(0)}(t) | \frac{\partial H}{\partial \epsilon} | \Psi_k^{(1)}(t) \rangle$$

$$|\chi_k^{(0)}(t)\rangle = \frac{\partial J_T}{\partial \langle \Psi_k^{(0)}(t)|}$$

$$g_a = \frac{\lambda_a}{S(t)} \int (\Delta \epsilon_l(t))^2\,dt \quad ; \quad \Delta \epsilon_l(t) = \epsilon_l^{(1)}(t) - \epsilon_l^{(0)}(t)$$

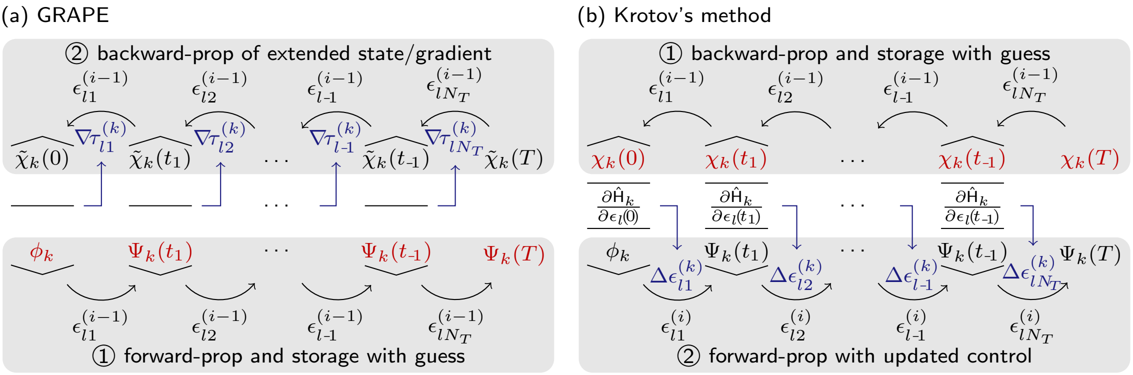
$$\Rightarrow \Delta \epsilon = \frac{S(t)}{\lambda_a} \sum_k \langle \chi_k^{(0)}(t) | \frac{\partial H}{\partial \epsilon} | \Psi_k^{(1)}(t) \rangle$$

Krotov Numerical Scheme



— Goerz *et al.* Quantum 6, 871 (2022)

GRAPE and Krotov Numerical Scheme Comparison

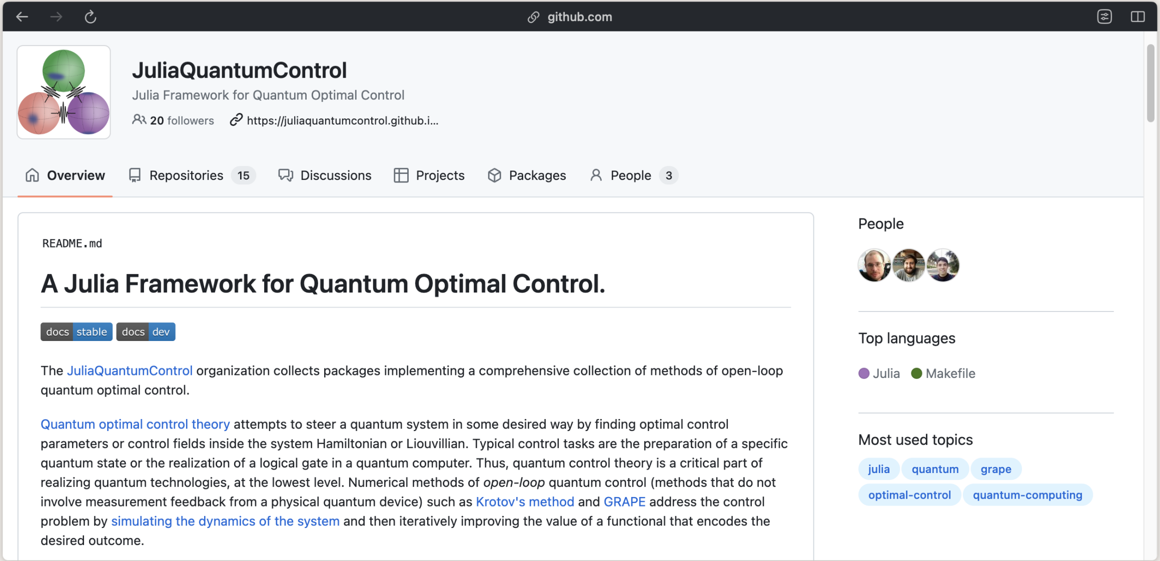


concurrent update

sequential update

— Goerz *et al.* Quantum 6, 871 (2022)

QuantumControl.jl



github.com

JuliaQuantumControl

Julia Framework for Quantum Optimal Control

20 followers https://juliaquantumcontrol.github.i...

Overview | Repositories 15 | Discussions | Projects | Packages | People 3

README.md

A Julia Framework for Quantum Optimal Control.

docs stable docs dev

The JuliaQuantumControl organization collects packages implementing a comprehensive collection of methods of open-loop quantum optimal control.

Quantum optimal control theory attempts to steer a quantum system in some desired way by finding optimal control parameters or control fields inside the system Hamiltonian or Liouvillian. Typical control tasks are the preparation of a specific quantum state or the realization of a logical gate in a quantum computer. Thus, quantum control theory is a critical part of realizing quantum technologies, at the lowest level. Numerical methods of *open-loop* quantum control (methods that do not involve measurement feedback from a physical quantum device) such as Krotov's method and GRAPE address the control problem by simulating the dynamics of the system and then iteratively improving the value of a functional that encodes the desired outcome.

People

Top languages

Julia Makefile

Most used topics

julia quantum grape optimal-control quantum-computing

Dynamical Generator

https://github.com/JuliaQuantumControl/JuliaCon2023-Slides

Glossary

Generator — Dynamical generator (Hamiltonian / Liouvillian) for the time evolution of a state, i.e., the right-hand-side of the equation of motion (up to a factor of i) such that $|\Psi(t+dt)\rangle = e^{-i\hat{H}dt}|\Psi(t)\rangle$ in the infinitesimal limit. We use the symbols G, $\hat{H}$, or L, depending on the context (general, Hamiltonian, Liouvillian). Examples for supported forms a Hamiltonian are the following, from the most general case to simplest and most common case of linear controls,

$$\hat{H} = \overbrace{\hat{H}_0}^{\text{drift term}} + \sum_l \overbrace{\hat{H}_l(\{\epsilon_{l'}(t)\}, t)}^{\text{control term}} \tag{G1}$$

$$\hat{H} = \hat{H}_0 + \sum_l \overbrace{a_l(\{\underbrace{\epsilon_{l'}(t)}_{\text{control function}}\}, t)}^{\text{control amplitude}} \hat{H}_l \tag{G2}$$

$$\hat{H} = \hat{H}_0 + \sum_l \overbrace{\epsilon_l(t)} \underbrace{\hat{H}_l}_{\text{control operator}} \tag{G3}$$

Dynamical Generator

https://github.com/JuliaQuantumControl/JuliaCon2023-Slides

Glossary

Generator — Dynamical generator (Hamiltonian / Liouvillian) for the time evolution of a state, i.e., the right-hand-side of the equation of motion (up to a factor of i) such that $|\Psi(t+dt)\rangle = e^{-i\hat{H}dt}|\Psi(t)\rangle$ in the infinitesimal limit. We use the symbols G, $\hat{H}$, or L, depending on the context (general, Hamiltonian, Liouvillian). Examples for supported forms a Hamiltonian are the following, from the most general case to simplest and most common case of linear controls,

```
return hamiltonian(Ĥ₀, (Ĥ₁re, Ωre), (Ĥ₁im, Ωim))
```

$$\boxed{\hat{H} = \hat{H}_0 + \sum_l \overbrace{a_l(\{\underbrace{\epsilon_{l'}(t)}_{\text{control function}}\}, t)}^{\text{control amplitude}}\, \hat{H}_l \qquad \text{(G2)}}$$

$$\hat{H} = \hat{H}_0 + \sum_l \overbrace{\epsilon_l(t)}\, \underbrace{\hat{H}_l}_{\text{control operator}} \qquad \text{(G3)}$$

Generator Interface

juliaquantumcontrol.github.io

API / Subpackages / QuantumPropagators

```
@test check_generator(generator; state, tlist,
                      for_mutable_state=true, for_immutable_state=true,
                      for_expval=true, atol=1e-15)
```

verifies the given `generator`:

- `get_controls(generator)` must be defined and return a tuple
- all controls returned by `get_controls(generator)` must pass `check_control`
- `evaluate(generator, tlist, n)` must return a valid operator (`check_operator`), with forwarded keyword arguments (including `for_expval`)
- `evaluate!(op, generator, tlist, n)` must be defined
- `substitute(generator, replacements)` must be defined
- If `generator` is a `Generator` instance, all elements of `generator.amplitudes` must pass `check_amplitude`.

source

Propagator Interface

juliaquantumcontrol.github.io

Overview

The Propagator interface

As a lower-level interface than `propagate`, the `QuantumPropagators` package defines an interface for "propagator" objects. These are initialized via `init_prop` as, e.g.,

```
using QuantumPropagators: init_prop

propagator = init_prop(Ψ₀, H, tlist)
```

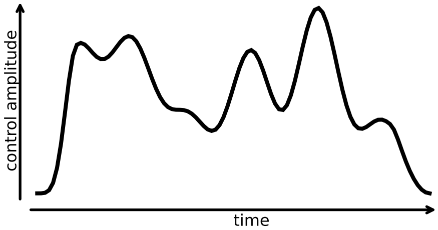
The `propagator` is a propagation-method-dependent object with the interface described by `AbstractPropagator`.

The `prop_step!` function can then be used to advance the `propagator`:

```
using QuantumPropagators: prop_step!

Ψ = prop_step!(propagator)  # single step
```

Parametrized Control Fields



piecewise-constant pulses

$\Rightarrow$ parametrized continuous controls

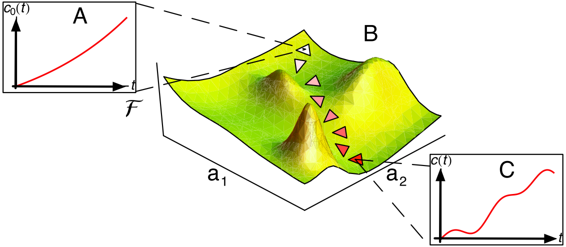
$$\epsilon(t) = \epsilon(\{u_n\}, t)$$

E.g. CRAB – Chopped Random (spectral) Basis

$$\epsilon(t) = \sum_{i=1}^{10} \left(a_n \cos(\omega_n t) + b_n \sin(\omega_n t) \right)$$

— Caneva *et al.* Phys. Rev. A 84, 022326 (2011)

Gradient-free optimization



Doria et al. PRL 106, 190501 (2011)

e.g. Nelder-Mead (simplex), genetic algorithms. . .

Gradients of parametrized pulses

$$
\begin{pmatrix} \frac{\partial \hat{\mathrm{U}}}{\partial u_1} \vert \Psi_k \rangle \\ \vdots \\ \frac{\partial \hat{\mathrm{U}}}{\partial u_N} \vert \Psi_k \rangle \\ \hat{\mathrm{U}} \vert \Psi_k \rangle \end{pmatrix} = \exp \left[-\mathrm{i}\mathcal{T} \int_0^T \begin{pmatrix} \hat{\mathrm{H}}(t) & 0 & \dots & 0 & \hat{\mathrm{H}}^{(1)}(t) \\ 0 & \hat{\mathrm{H}}(t) & \dots & 0 & \hat{\mathrm{H}}^{(2)}(t) \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & \hat{\mathrm{H}}(t) & \hat{\mathrm{H}}^{(N)}(t) \\ 0 & 0 & \dots & 0 & \hat{\mathrm{H}}(t) \end{pmatrix} dt \right] \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \vert \Psi_k \rangle \end{pmatrix}
$$

with $\hat{\mathrm{H}}^{(n)}(t) = \frac{\partial \hat{\mathrm{H}}(t)}{\partial u_n}$

— "GOAT": Machnes *et al.* Phys. Rev. Lett. 120, 150401 (2018)

Open Quantum Systems

Lindblad equation:

$$\frac{d}{dt}\hat{\rho}(t) = -i\left[\hat{\mathrm{H}}, \hat{\rho}(t)\right] + \mathcal{L}_D(\hat{\rho}(t))$$
$$= -i\left[\hat{\mathrm{H}}, \hat{\rho}(t)\right] + \sum_k \left(\hat{\mathrm{A}}_k\hat{\rho}\hat{\mathrm{A}}_k^\dagger - \frac{1}{2}\hat{\mathrm{A}}_k^\dagger\hat{\mathrm{A}}_k\hat{\rho} - \frac{1}{2}\hat{\rho}\hat{\mathrm{A}}_k^\dagger\hat{\mathrm{A}}_k\right)$$

Vectorization rule:

$$\mathrm{vec}\left(\hat{\mathrm{A}}\hat{\rho}\hat{\mathrm{B}}\right) = \left(\hat{B}^T \otimes \hat{A}\right)\vec{\rho}$$

Matrix representation of Lindbladian:

$$\hat{L} = -i(\mathbf{1}\otimes\hat{H}) + i(\hat{H}^T\otimes\mathbf{1}) + \sum_k\left[(\hat{A}_k^\dagger)^T\otimes\hat{A}_k - \frac{1}{2}\left(\mathbf{1}\otimes\hat{A}_k^\dagger\hat{A}_k\right) - \frac{1}{2}\left((\hat{A}_k^\dagger\hat{A}_k)^T\otimes\mathbf{1}\right)\right]$$

— Goerz *et. al.* arXiv:1312.0111v2 (2021), Appendix B