



QuantumControl.jl: Pulse-Level Control of Quantum Gates via Semi-Automatic Differentiation

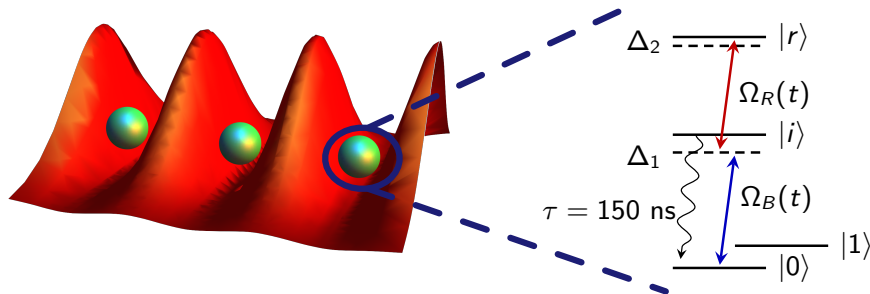
Michael H. Goerz

DEVCOM Army Research Lab

Yao Community Call • September 1, 2022

Optimal Control Tasks — Quantum Gates

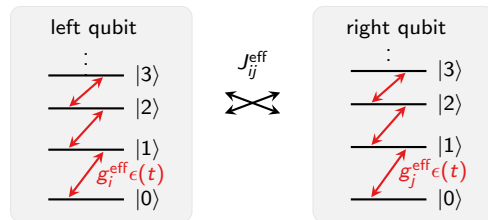
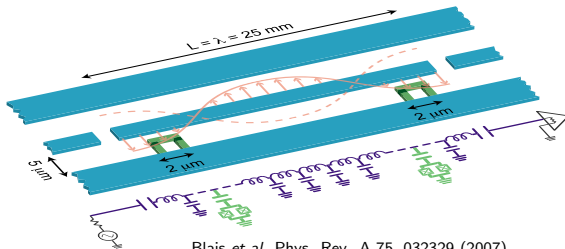
Rydberg atoms



Goerz *et al.* Phys. Rev. A 90, 032329 (2014)

Optimal Control Tasks — Quantum Gates

Two transmon qubits with shared transmission line



Goerz *et al.* EPJ Quantum Tech. 2, 21 (2015)

Goerz *et al.* npj Quantum Information 3, 37 (2017)

Quantum Control Problem

- Bunch of states: $\{|\Psi_k(t)\rangle\}$
 - e.g. two-qubit gate: $|00\rangle, |01\rangle, |10\rangle, |11\rangle$

Quantum Control Problem

- Bunch of states: $\{|\Psi_k(t)\rangle\}$
 - e.g. two-qubit gate: $|00\rangle, |01\rangle, |10\rangle, |11\rangle$
- Hamiltonian(s) with control fields: $\hat{H}_k(\{\epsilon_l(t)\}) \rightarrow$ time propagation
 - assume piecewise-constant: ϵ_{ln} for n 'th time interval of l 'th control

Quantum Control Problem

“Pulse-level” control

- Bunch of states: $\{|\Psi_k(t)\rangle\}$
 - e.g. two-qubit gate: $|00\rangle, |01\rangle, |10\rangle, |11\rangle$
- Hamiltonian(s) with control fields: $\hat{H}_k(\{\epsilon_l(t)\}) \rightarrow$ time propagation
 - assume piecewise-constant: ϵ_{ln} for n 'th time interval of l 'th control

Quantum Control Problem

“Pulse-level” control

- Bunch of states: $\{|\Psi_k(t)\rangle\}$
 - e.g. two-qubit gate: $|00\rangle, |01\rangle, |10\rangle, |11\rangle$
- Hamiltonian(s) with control fields: $\hat{H}_k(\{\epsilon_l(t)\}) \rightarrow$ time propagation
 - assume piecewise-constant: ϵ_{ln} for n 'th time interval of l 'th control

Functional

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

Quantum Control Problem

“Pulse-level” control

- Bunch of states: $\{|\Psi_k(t)\rangle\}$
— e.g. two-qubit gate: $|00\rangle, |01\rangle, |10\rangle, |11\rangle$
- Hamiltonian(s) with control fields: $\hat{H}_k(\{\epsilon_l(t)\}) \rightarrow$ time propagation
— assume piecewise-constant: ϵ_{ln} for n 'th time interval of l 'th control

Functional

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

Gradient-based “open loop” optimization

$$(\nabla J)_{ln} \equiv \frac{\partial J}{\partial \epsilon_{ln}} \Rightarrow \text{L-BFGS-B}$$

Efficient Quantum Control

- Get your data structures right
 - grid representation (FFT), sparsity

Efficient Quantum Control

- Get your data structures right
 - grid representation (FFT), sparsity
- Get your propagation right
 - polynomial expansions, in-place BLAS

Propagation Methods

$$i\frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle$$

Propagation Methods

$$i\frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle$$

$$i\frac{\partial}{\partial t} \hat{\rho}(t) = \mathcal{L}\hat{\rho}(t)$$

Propagation Methods

$$i\frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle$$

$$i\frac{\partial}{\partial t} \hat{\rho}(t) = \mathcal{L}\hat{\rho}(t)$$

- ODE Solver

Propagation Methods

$$i\frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle$$

$$i\frac{\partial}{\partial t} \hat{\rho}(t) = \mathcal{L}\hat{\rho}(t)$$

- ODE Solver
- Evaluate analytical solution

$$|\Psi(t + dt)\rangle = e^{-i\hat{H}dt} |\Psi(t)\rangle$$

Propagation Methods

$$i\frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle$$

$$i\frac{\partial}{\partial t} \hat{\rho}(t) = \mathcal{L}\hat{\rho}(t)$$

- ODE Solver
- Evaluate analytical solution

$$|\Psi(t + dt)\rangle = e^{-i\hat{H}dt} |\Psi(t)\rangle$$

- Split-Step

Propagation Methods

$$i\frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle$$

$$i\frac{\partial}{\partial t} \hat{\rho}(t) = \mathcal{L}\hat{\rho}(t)$$

- ODE Solver
- Evaluate analytical solution

$$|\Psi(t + dt)\rangle = e^{-i\hat{H}dt} |\Psi(t)\rangle = \sum_{n=0}^{\infty} c_n P_n(-i\hat{H}dt) |\Psi(t)\rangle$$

- Split-Step

Propagation Methods

$$i\frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle$$

$$i\frac{\partial}{\partial t} \hat{\rho}(t) = \mathcal{L}\hat{\rho}(t)$$

- ODE Solver
- Evaluate analytical solution

$$|\Psi(t + dt)\rangle = e^{-i\hat{H}dt} |\Psi(t)\rangle = \sum_{n=0}^{\infty} c_n P_n(-i\hat{H}dt) |\Psi(t)\rangle$$

- Split-Step

Propagation Methods

$$i\frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle$$

$$i\frac{\partial}{\partial t} \hat{\rho}(t) = \mathcal{L}\hat{\rho}(t)$$

- ODE Solver
- Evaluate analytical solution

$$|\Psi(t + dt)\rangle = e^{-i\hat{H}dt} |\Psi(t)\rangle = \sum_{n=0}^{\infty} c_n P_n(-i\hat{H}dt) |\Psi(t)\rangle$$

- Split-Step
- Taylor first order + adjust for unitarity $\Rightarrow$ Crank-Nicolson

Propagation Methods

$$i\frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle$$

$$i\frac{\partial}{\partial t} \hat{\rho}(t) = \mathcal{L}\hat{\rho}(t)$$

- ODE Solver
- Evaluate analytical solution

$$|\Psi(t + dt)\rangle = e^{-i\hat{H}dt} |\Psi(t)\rangle = \sum_{n=0}^{\infty} c_n P_n(-i\hat{H}dt) |\Psi(t)\rangle$$

- Split-Step
- Taylor first order + adjust for unitarity $\Rightarrow$ Crank-Nicolson
- High order Taylor + Krylov subspace $\Rightarrow$ Arnoldi-Lanczos

Propagation Methods

$$i\frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H} |\Psi(t)\rangle$$

$$i\frac{\partial}{\partial t} \hat{\rho}(t) = \mathcal{L}\hat{\rho}(t)$$

- ODE Solver
- Evaluate analytical solution

$$|\Psi(t + dt)\rangle = e^{-i\hat{H}dt} |\Psi(t)\rangle = \sum_{n=0}^{\infty} c_n P_n(-i\hat{H}dt) |\Psi(t)\rangle$$

- Split-Step
- Taylor first order + adjust for unitarity $\Rightarrow$ Crank-Nicolson
- High order Taylor + Krylov subspace $\Rightarrow$ Arnoldi-Lanczos
- Faster converging polynomials $\Rightarrow$ Chebychev

Chebyshev Propagation

Chebyshev Polynomials

$$P_n(x) = \cos(n\theta) ; \quad \theta = \arccos(x) ; \quad P_n(x) \quad \text{is defined for} \quad x \in [-1, 1]$$

Chebyshev Propagation

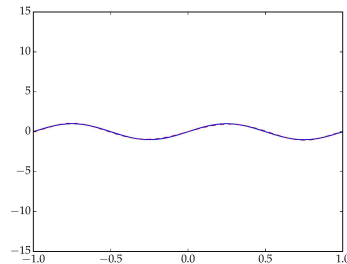
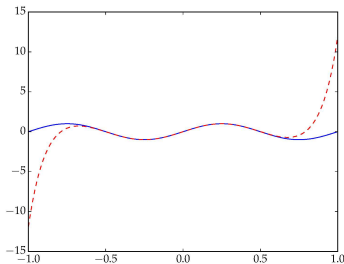
Chebyshev Polynomials

$$P_n(x) = \cos(n\theta) ; \quad \theta = \arccos(x) ; \quad P_n(x) \text{ is defined for } x \in [-1, 1]$$

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

Chebyshev Propagation

Approximate $f(x) = \sin(2\pi x)$ on $x \in [-1.0, 1.0]$ by 9th degree polynomial



Courtesy of Daniel Reich

Chebyshev Propagation

Chebyshev Polynomials

$$P_n(x) = \cos(n\theta) ; \quad \theta = \arccos(x) ; \quad P_n(x) \text{ is defined for } x \in [-1, 1]$$

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

Chebyshev Propagation

Chebyshev Polynomials

$$P_n(x) = \cos(n\theta); \quad \theta = \arccos(x); \quad P_n(x) \text{ is defined for } x \in [-1, 1]$$

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

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

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

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

Chebyshev Propagation

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

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 $\vec{w} = e^{-i \hat{A} dt} \vec{v} \in \mathbb{C}^N$

```

1: procedure CHEBY( $\vec{v}, \hat{A}, dt$ )
2:    $\Delta =$  spectral radius of  $\hat{A}$ 
3:    $E_{\min} =$  minimum eigenvalue of  $\hat{A}$ 
4:    $[a_0 \dots a_n] = \text{EXPCHEBYCOEFFS}(\Delta, E_{\min}, dt)$ 
5:    $d = \frac{1}{2}\Delta$ ;  $\beta = d + E_{\min}$ 
6:    $\vec{v}_0 = \vec{v}$ 
7:    $\vec{w}^{(0)} = a_0 \vec{v}_0$ 
8:    $\vec{v}_1 = \pm \frac{i}{d} (\hat{A} \vec{v}_0 - \beta \vec{v}_0)$ 
9:    $\vec{w}^{(1)} = \vec{w}^{(0)} + a_1 \vec{v}_1$ 
10:  for  $i = 2 : n$  do
11:     $\vec{v}_i = \pm \frac{2i}{d} (\hat{A} \vec{v}_{i-1} - \beta \vec{v}_{i-1}) + \vec{v}_{i-2}$ 
12:     $\vec{w}^{(i)} = \vec{w}^{(i-1)} + a_i \vec{v}_i$ 
13:  end for
14:  return  $e^{\pm i \beta dt} \vec{w}^{(n)}$ 
15: 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 Chebyshev coefficients $[a_0 \dots a_n]$ allowing to approximate $f(\hat{A} dt)$ to pre-defined precision.

```

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

Efficient Quantum Control

- Get your data structures right
 - grid representation (FFT), sparsity
- Get your propagation right
 - polynomial expansions, in-place BLAS

Efficient Quantum Control

- Get your data structures right
 - grid representation (FFT), sparsity
- Get your propagation right
 - polynomial expansions, in-place BLAS
- Optimal Control Methods
 - parallelization

Automatic Differentiation (AD)

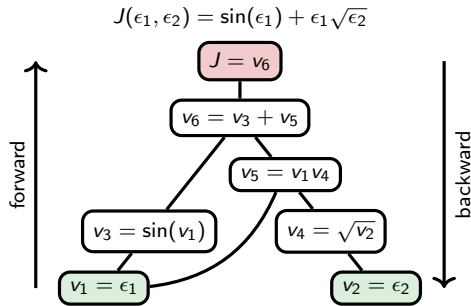
backward-mode “adjoint”

$$\bar{v}_j \equiv \frac{\partial J}{\partial v_j}$$

$$= \sum_i \bar{v}_i \frac{\partial v_i}{\partial v_j}$$

sum over all v_i

which depend on v_j



Automatic Differentiation (AD)

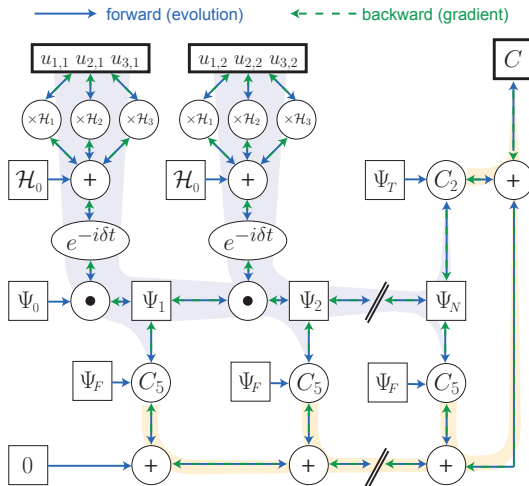


Fig. 2 in Leung *et al.* Phys. Rev. A 95, 042318 (2017)

AD Advantages

■ Arbitrary functionals

μ	Cost-function contribution	$C_\mu(\mathbf{u})$
1	Target-gate infidelity	$1 - \text{tr}(K_T^\dagger K_N)/D ^2$
2	Target-state infidelity	$1 - \langle \Psi_T \Psi_N \rangle ^2$
3	Control amplitudes	$ \mathbf{u} ^2$
4	Control variations	$\sum_{j,k} u_{k,j} - u_{k,j-1} ^2$
5	Occupation of forbidden state	$\sum_j \langle \Psi_F \Psi_j \rangle ^2$
6	Evolution time (target gate)	$1 - \frac{1}{N} \sum_j \text{tr}(K_T^\dagger K_j)/D ^2$
7	Evolution time (target state)	$1 - \frac{1}{N} \sum_j \langle \Psi_T \Psi_j \rangle ^2$

Table 1 in Leung *et al.* Phys. Rev. A 95, 042318 (2017)

AD Advantages

■ Arbitrary functionals

μ	Cost-function contribution	$C_\mu(\mathbf{u})$
1	Target-gate infidelity	$1 - \text{tr}(K_T^\dagger K_N)/D ^2$
2	Target-state infidelity	$1 - \langle \Psi_T \Psi_N \rangle ^2$
3	Control amplitudes	$ \mathbf{u} ^2$
4	Control variations	$\sum_{j,k} u_{k,j} - u_{k,j-1} ^2$
5	Occupation of forbidden state	$\sum_j \langle \Psi_F \Psi_j \rangle ^2$
6	Evolution time (target gate)	$1 - \frac{1}{N} \sum_j \text{tr}(K_T^\dagger K_j)/D ^2$
7	Evolution time (target state)	$1 - \frac{1}{N} \sum_j \langle \Psi_T \Psi_j \rangle ^2$

Table 1 in Leung *et al.* Phys. Rev. A 95, 042318 (2017)

■ Arbitrary equations of motion

e.g., quantum trajectories — Abdelhafez *et al.* Phys. Rev. A 99, 052327 (2019)

AD Advantages

- Arbitrary functionals

μ	Cost-function contribution	$C_\mu(\mathbf{u})$
1	Target-gate infidelity	$1 - \text{tr}(K_T^\dagger K_N)/D ^2$
2	Target-state infidelity	$1 - \langle \Psi_T \Psi_N \rangle ^2$
3	Control amplitudes	$ \mathbf{u} ^2$
4	Control variations	$\sum_{j,k} u_{k,j} - u_{k,j-1} ^2$
5	Occupation of forbidden state	$\sum_j \langle \Psi_F \Psi_j \rangle ^2$
6	Evolution time (target gate)	$1 - \frac{1}{N} \sum_j \text{tr}(K_T^\dagger K_j)/D ^2$
7	Evolution time (target state)	$1 - \frac{1}{N} \sum_j \langle \Psi_T \Psi_j \rangle ^2$

Table 1 in Leung *et al.* Phys. Rev. A 95, 042318 (2017)

- Arbitrary equations of motion

e.g., quantum trajectories — Abdelhafez *et al.* Phys. Rev. A 99, 052327 (2019)

- GPU support

Entanglement Measures

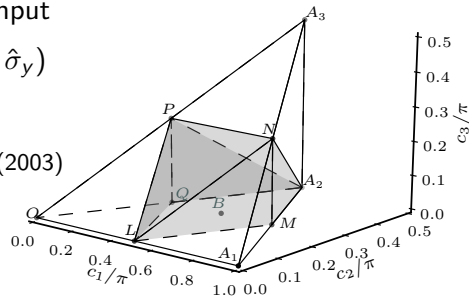
Quantum Gate Concurrence

Max concurrence that can be generated for a separable input

1 $c_1, c_2, c_3 \propto \text{eigvals}(\hat{U}\tilde{U})$; $\tilde{U} = (\hat{\sigma}_y \otimes \hat{\sigma}_y) \hat{U} (\hat{\sigma}_y \otimes \hat{\sigma}_y)$

2 $C(\hat{U}) = \max |\sin(c_{1,2,3} \pm c_{3,1,2})|$

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



Entanglement Measures

Quantum Gate Concurrence

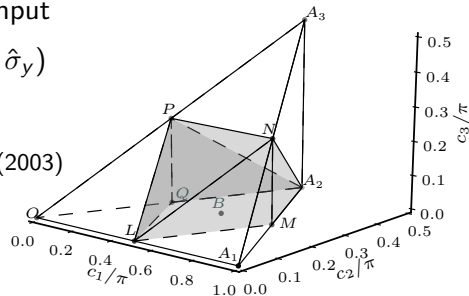
Max concurrence that can be generated for a separable input

1 $c_1, c_2, c_3 \propto \text{eigvals}(\hat{U}\tilde{U})$; $\tilde{U} = (\hat{\sigma}_y \otimes \hat{\sigma}_y) \hat{U} (\hat{\sigma}_y \otimes \hat{\sigma}_y)$

2 $C(\hat{U}) = \max |\sin(c_{1,2,3} \pm c_{3,1,2})|$

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

Not analytic!



Entanglement Measures

Quantum Gate Concurrence

Max concurrence that can be generated for a separable input

$$1 \quad c_1, c_2, c_3 \propto \text{eigvals}(\hat{U}\tilde{U}) ; \quad \tilde{U} = (\hat{\sigma}_y \otimes \hat{\sigma}_y) \hat{U} (\hat{\sigma}_y \otimes \hat{\sigma}_y)$$

$$2 \quad C(\hat{U}) = \max |\sin(c_{1,2,3} \pm c_{3,1,2})|$$

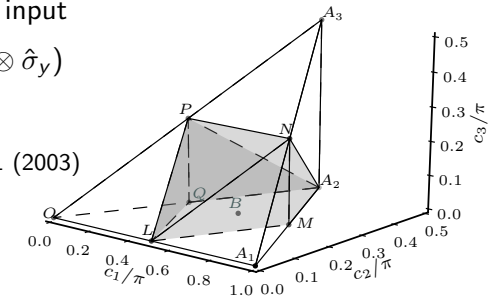
Childs *et al.* Phys. Rev. A 68, 052311 (2003)

Perfect Entanglers Functional

Find a two-qubit gate with maximum entangling power

$$F_{PE} = \left(\frac{1}{\det U_B} \right) \left(\frac{1}{4} (\text{tr}^2[U_B^T U_B] - \text{tr}[U_B^T U_B U_B^T U_B]) \right) \left(\frac{1}{16} \text{Re}^2[\text{tr}[U_B^T U_B]] \right) + \\ + \left(\frac{2}{\det U_B} \right) \left(\frac{1}{4} (\text{tr}^2[U_B^T U_B] - \text{tr}[U_B^T U_B U_B^T U_B]) \right) \left(\frac{1}{16} \text{Im}^2[\text{tr}[U_B^T U_B]] \right) \\ \left(\frac{1}{16} \text{Re}[\text{tr}^2[U_B^T U_B]] \right)$$

U_B : projection into logical subspace, in Bell basis



Watts *et al.* Phys. Rev. A 91, 062306 (2015)

Goerz *et al.* Phys. Rev. A 91, 062307 (2015)

Entanglement Measures

Quantum Gate Concurrence

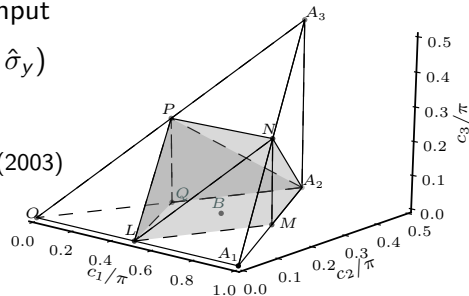
Max concurrence that can be generated for a separable input

$$1 \quad c_1, c_2, c_3 \propto \text{eigvals}(\hat{U}\tilde{U}) ; \quad \tilde{U} = (\hat{\sigma}_y \otimes \hat{\sigma}_y) \hat{U} (\hat{\sigma}_y \otimes \hat{\sigma}_y)$$

$$2 \quad C(\hat{U}) = \max |\sin(c_{1,2,3} \pm c_{3,1,2})|$$

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

To a computer, everything is analytic!



Entanglement Measures

Quantum Gate Concurrence

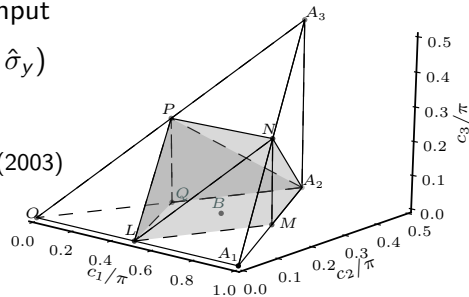
Max concurrence that can be generated for a separable input

$$1 \quad c_1, c_2, c_3 \propto \text{eigvals}(\hat{U}\tilde{U}) ; \quad \tilde{U} = (\hat{\sigma}_y \otimes \hat{\sigma}_y) \hat{U} (\hat{\sigma}_y \otimes \hat{\sigma}_y)$$

$$2 \quad C(\hat{U}) = \max |\sin(c_{1,2,3} \pm c_{3,1,2})|$$

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

To a computer, everything is analytic!



Quantum Fisher Information

$$F(\hat{\rho}) = \sum_{i \neq j} \frac{2(p_i - p_j)^2}{p_i + p_j} \left| \langle \phi_i | \hat{S}_z | \phi_j \rangle \right|^2$$

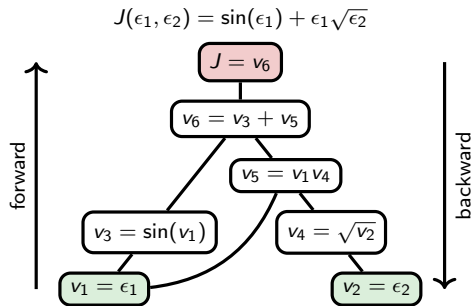
where p_i , $|\phi_i\rangle$ are eigenvalues / eigenstates of $\hat{\rho}$

— Ma *et al.* Phys. Rep. 509, 89 (2011)

AD Compromises

1 Numerical scaling

- AD memory overhead
- computational overhead (at least on CPU)



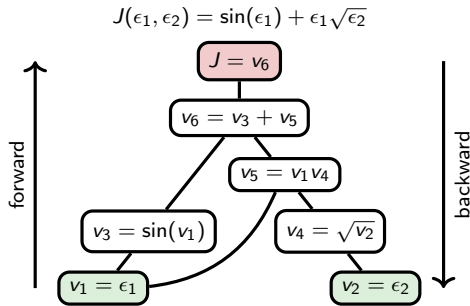
AD Compromises

1 Numerical scaling

- AD memory overhead
- computational overhead (at least on CPU)

2 Framework limitations

- Complex numbers?
- In-place operations?
- Double-precision?



AD Compromises

1 Numerical scaling

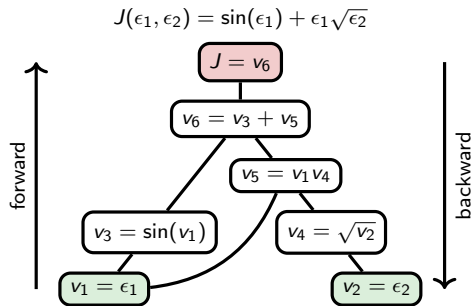
- AD memory overhead
- computational overhead (at least on CPU)

2 Framework limitations

- Complex numbers?
- In-place operations?
- Double-precision?

3 Code reuse

- Re-implement propagation methods?
- Re-use existing GRAPE implementation?



AD Compromises

1 Numerical scaling

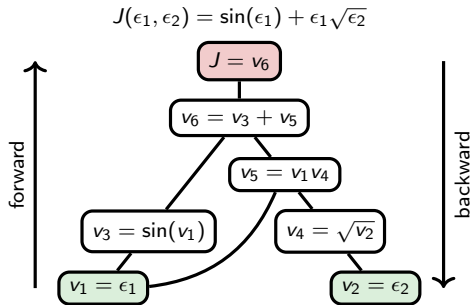
- AD memory overhead
- computational overhead (at least on CPU)

2 Framework limitations

- Complex numbers?
- In-place operations?
- Double-precision?

3 Code reuse

- Re-implement propagation methods?
- Re-use existing GRAPE implementation?



We don't have to compromise!

Semi-Automatic Differentiation

arXiv:2205.15044

Quantum Optimal Control via Semi-Automatic Differentiation

Michael H. Goerz, Sebastián C. Carrasco, and Vladimir S. Malinovsky

DEVCOM Army Research Laboratory, 2800 Powder Mill Road, Adelphi, MD 20783, USA



We develop a framework of “semi-automatic differentiation” that combines existing gradient-based methods of quantum optimal control with automatic differentiation. The approach allows to optimize practically any computable functional and is implemented in two open source Julia packages, `GRAPE.jl` and `Krotov.jl`, part of the `QuantumControl.jl` framework. Our method is based on formally rewriting the optimization functional in terms of propagated states, overlaps with target states, or quantum gates. An analytical application of the chain rule then allows to separate the time propagation and the evaluation of the functional when calculating the gradient. The former can be evaluated with great efficiency via a modified GRAPE scheme. The latter is evaluated with automatic differenti-

Funding

DEVCOM Army Research Laboratory, Cooperative Agreement Numbers W911NF-16-2-0147, W911NF-21-2-0037; DIRA-TRC No. DTR19-CI-019

ant-ph] 27 May 2022

arXiv:2205.15044

Semi-Automatic Differentiation

arXiv:2205.15044

Semi-Automatic Differentiation

$$J(\{\epsilon_{ne}\}) = J_T(\{\chi_u(\tau)\}) + \dots$$

$$\nabla J = \frac{\partial J_T}{\partial \epsilon_{ne}}$$

$$\frac{\partial J_T}{\partial \epsilon_{ne}} = 2 \operatorname{Re} \left[\sum_u \underbrace{\frac{\partial J_T}{\partial \langle \chi_u(\tau) |}}_{\langle \chi_u(\tau) |} \cdot \frac{\partial |\chi_u(\tau)\rangle}{\epsilon_{ne}} \right] ; \quad |\chi_u(\tau)\rangle = \frac{\partial J_T}{\partial \langle \chi_u(\tau) |}$$

$$= 2 \operatorname{Re} \left[\sum_u \frac{\partial}{\partial \epsilon_{ne}} \langle \chi_u(\tau) | \chi_u(\tau) \rangle \right]$$

arXiv:2205.15044

Semi-Automatic Differentiation

$$\frac{\partial}{\partial \epsilon_{ue}} \langle \chi_u(\tau) | \hat{U}_n \dots \hat{U}_1 | \chi_u(0) \rangle \quad ; \quad \hat{U}_n = e^{-i \hat{H}_n dt}$$

$$= \underbrace{\langle \chi_u(\tau) | \hat{U}_n \dots \hat{U}_{n+1}}_{bv} \frac{\partial \hat{U}_n}{\partial \epsilon_{ue}} \underbrace{\hat{U}_{n-1} \dots \hat{U}_1 | \chi_u(0) \rangle}_{fw}$$

↑

$$| \chi_u(\tau) \rangle = \frac{\partial \mathcal{Z}}{\partial \langle \chi_u(\tau) |}$$

$$\mathcal{Z}(\hat{A})$$

$$\hat{A}_{iu} = \langle q_i | \chi_u \rangle$$

$$\frac{\partial \mathcal{Z}}{\partial \langle \chi_u(\tau) |} = \sum_i \frac{\partial \mathcal{Z}}{\partial u_{iu}} | q_i \rangle$$

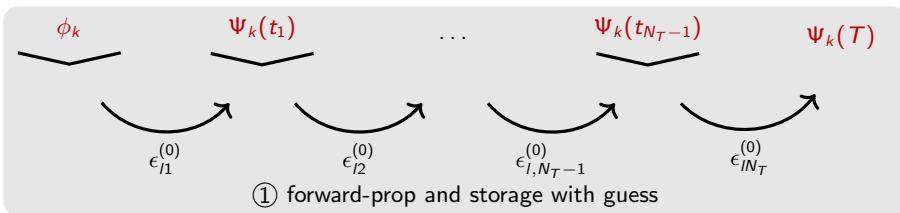
Gradient of Time Evolution Operator

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

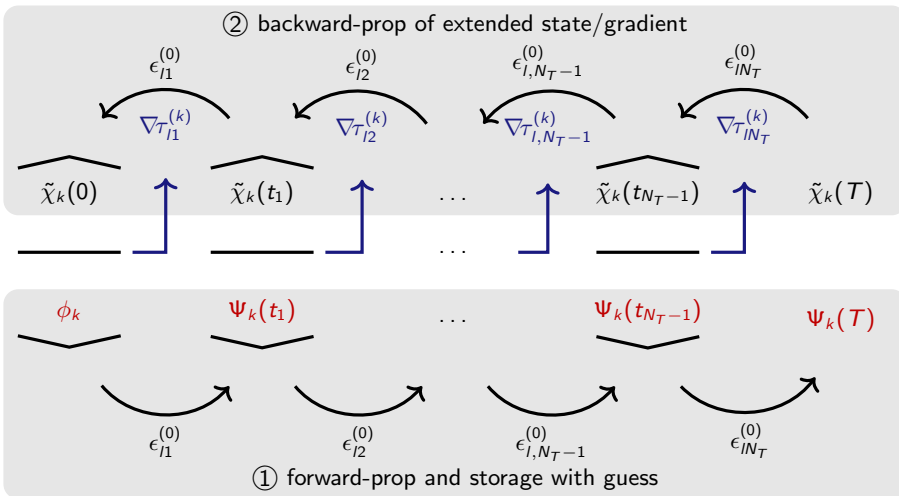
$$\hat{U}_n = \exp[-i\hat{H}_n dt_n]; \quad \hat{H}_n^{(l)} = \frac{\partial \hat{H}_n}{\partial \epsilon_l(t)}$$

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

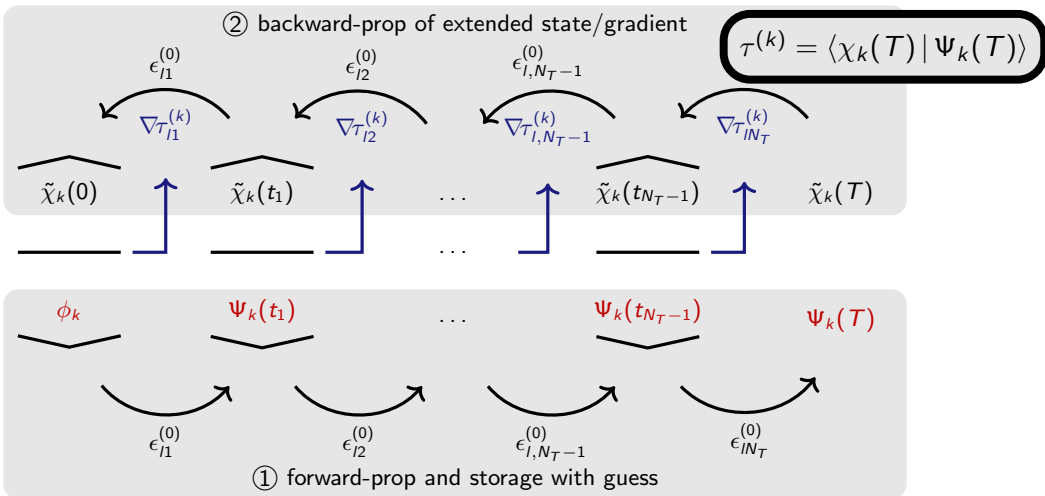
Generalized GRAPE scheme



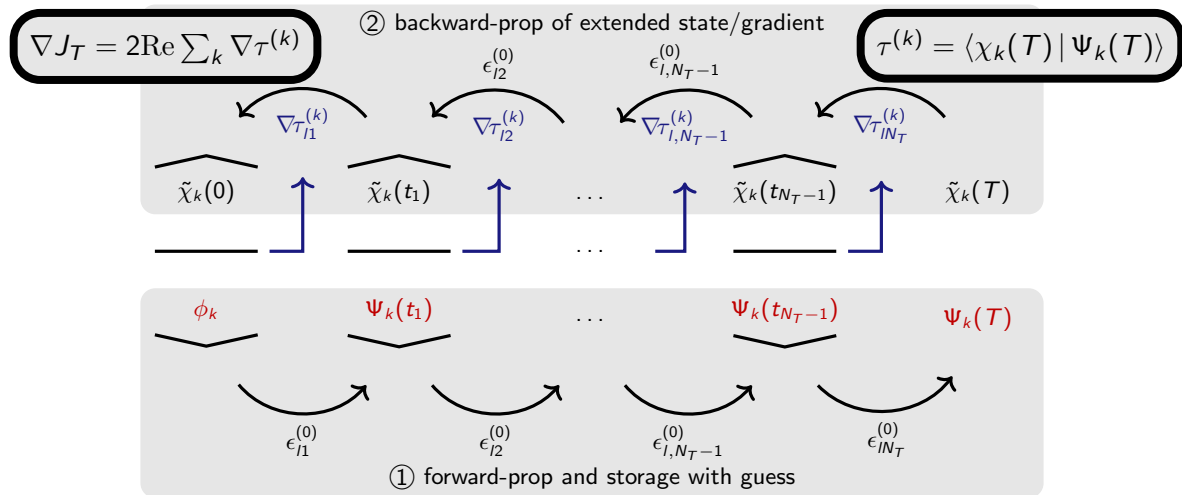
Generalized GRAPE scheme



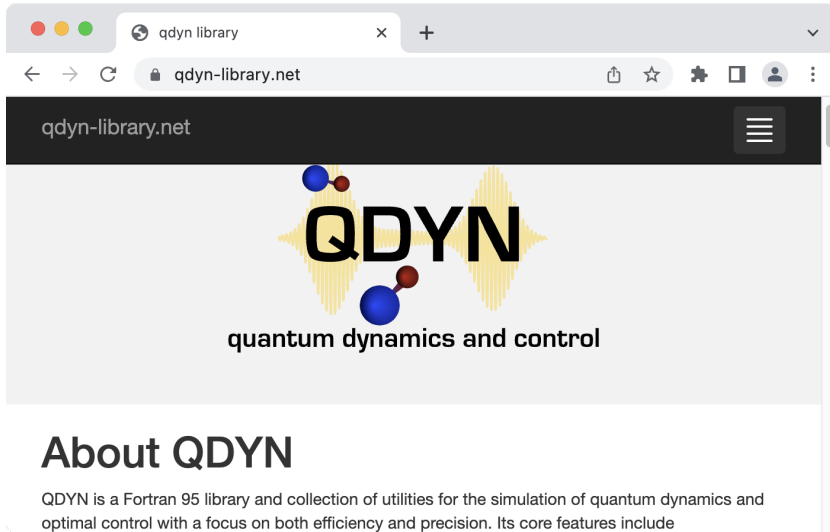
Generalized GRAPE scheme



Generalized GRAPE scheme



QDYN

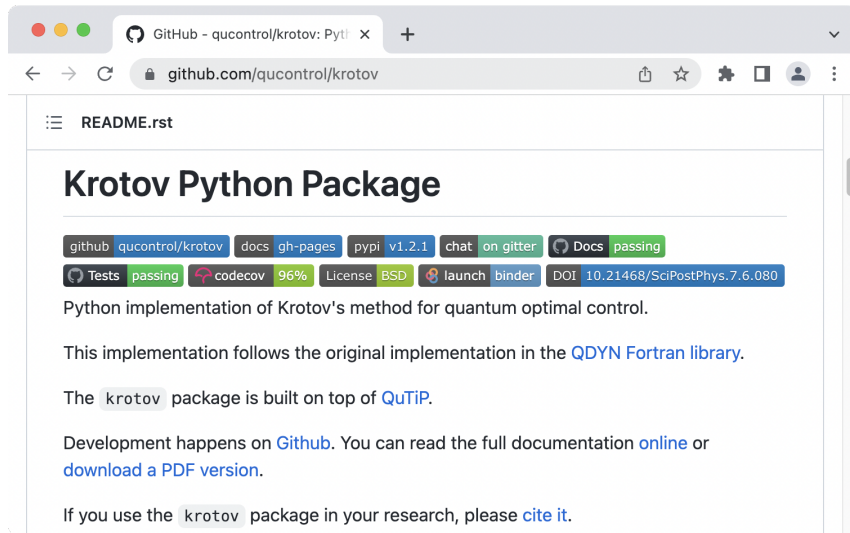


C. Koch group
FU Berlin



Fortran

Krotov Python Package



The screenshot shows a web browser displaying the GitHub repository for the Krotov Python Package. The browser's address bar shows the URL `github.com/qucontrol/krotov`. The repository page has a tab labeled "README.rst". The main heading is "Krotov Python Package". Below the heading is a row of badges: "github qucontrol/krotov", "docs gh-pages", "pypi v1.2.1", "chat on gitter", "Docs passing", "Tests passing", "codecov 96%", "License BSD", "launch binder", and "DOI 10.21468/SciPostPhys.7.6.080". The text describes the package as a "Python implementation of Krotov's method for quantum optimal control." It mentions that the implementation follows the original in the "QDYN Fortran library". It states that the "krotov" package is built on top of "QuTiP". It notes that development happens on "Github" and provides links for "full documentation online" and a "download a PDF version". It concludes by asking users to "cite it" if they use the package in their research.

github qucontrol/krotov docs gh-pages pypi v1.2.1 chat on gitter Docs passing

Tests passing codecov 96% License BSD launch binder DOI 10.21468/SciPostPhys.7.6.080

Python implementation of Krotov's method for quantum optimal control.

This implementation follows the original implementation in the [QDYN Fortran library](#).

The `krotov` package is built on top of [QuTiP](#).

Development happens on [Github](#). You can read the full documentation [online](#) or [download a PDF version](#).

If you use the `krotov` package in your research, please [cite it](#).

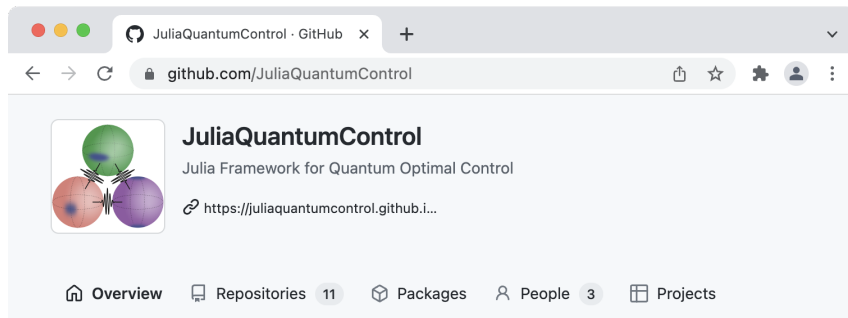


C. Koch group
FU Berlin



Python

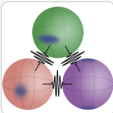
JuliaQuantumControl



The screenshot shows a web browser window displaying the GitHub repository for JuliaQuantumControl. The browser's address bar shows the URL `github.com/JuliaQuantumControl`. The repository page features a header with the repository name, a description "Julia Framework for Quantum Optimal Control", and a link to the repository's website. Below the header is a navigation bar with tabs for Overview, Repositories (11), Packages, People (3), and Projects. The Overview tab is selected, showing the README content.

JuliaQuantumControl · GitHub

github.com/JuliaQuantumControl

 **JuliaQuantumControl**
Julia Framework for Quantum Optimal Control
<https://juliaquantumcontrol.github.io/>

Overview Repositories 11 Packages People 3 Projects

README.md

A Julia Framework for Quantum Optimal Control.

The [JuliaQuantumControl](#) organization collects packages implementing a comprehensive collection of methods of open-loop quantum optimal control.



Julia

 Zygote

Demo

Benchmarks

