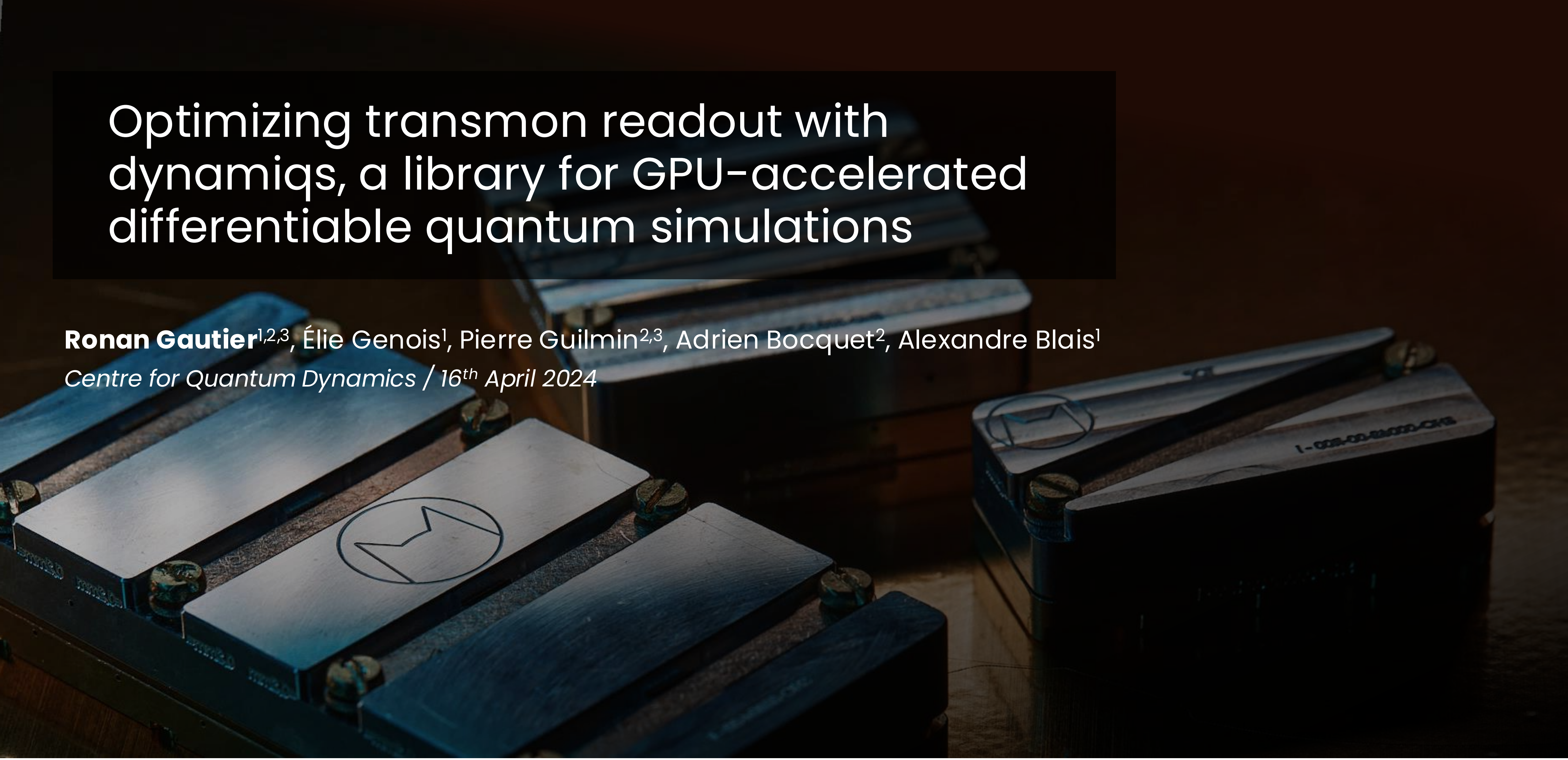
Optimizing transmon readout with dynamiqs, a library for GPU-accelerated differentiable quantum simulations

Ronan Gautier[1,2,3], Élie Genois[1], Pierre Guilmin[2,3], Adrien Bocquet[2], Alexandre Blais[1]

Centre for Quantum Dynamics / 16th April 2024

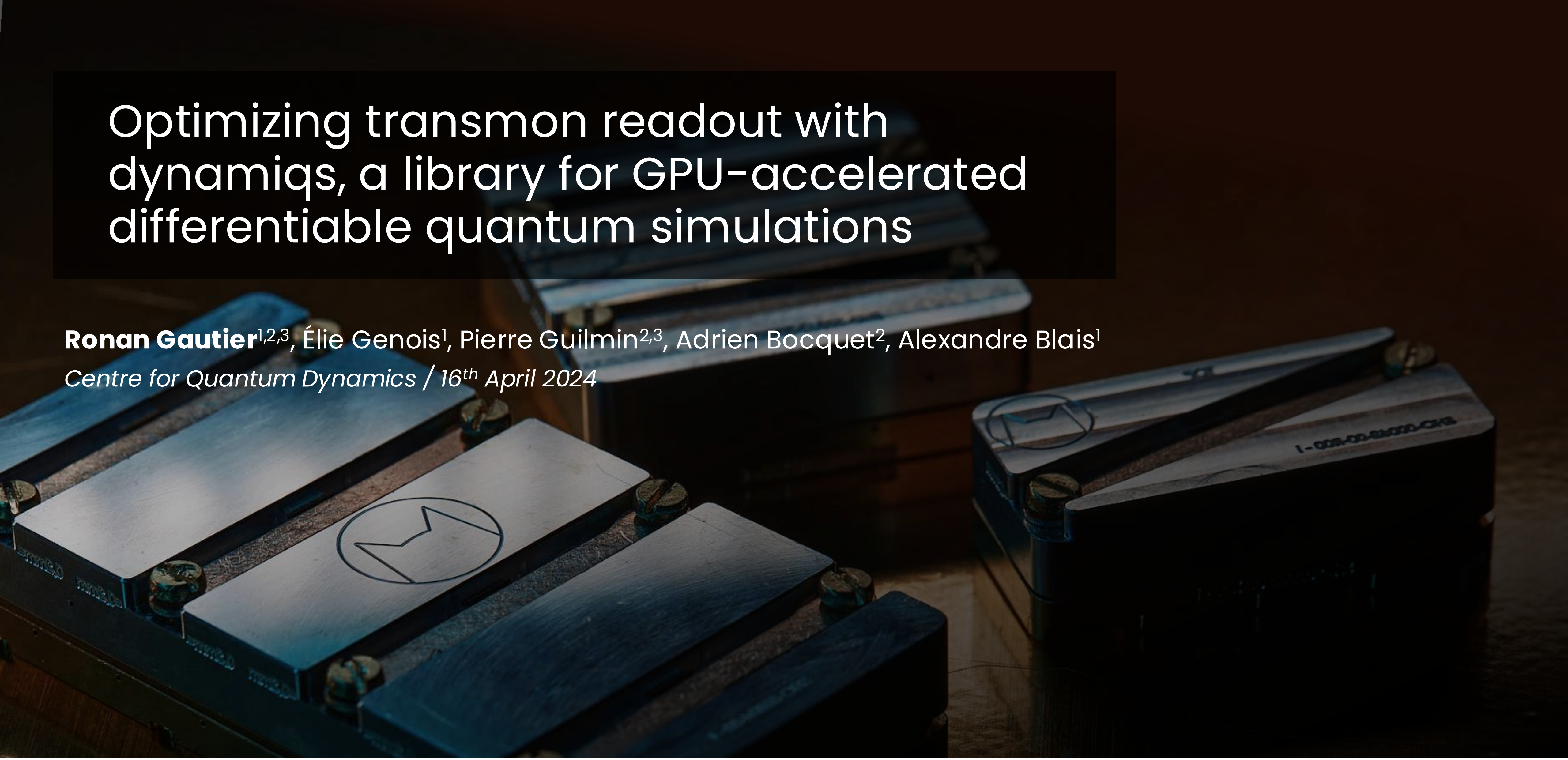


1 Institut Quantique
2 Alice & Bob
3 Inria

Optimizing transmon readout with dynamiqs, a library for GPU-accelerated differentiable quantum simulations

Ronan Gautier[1,2,3], Élie Genois[1], Pierre Guilmin[2,3], Adrien Bocquet[2], Alexandre Blais[1]

Centre for Quantum Dynamics / 16th April 2024

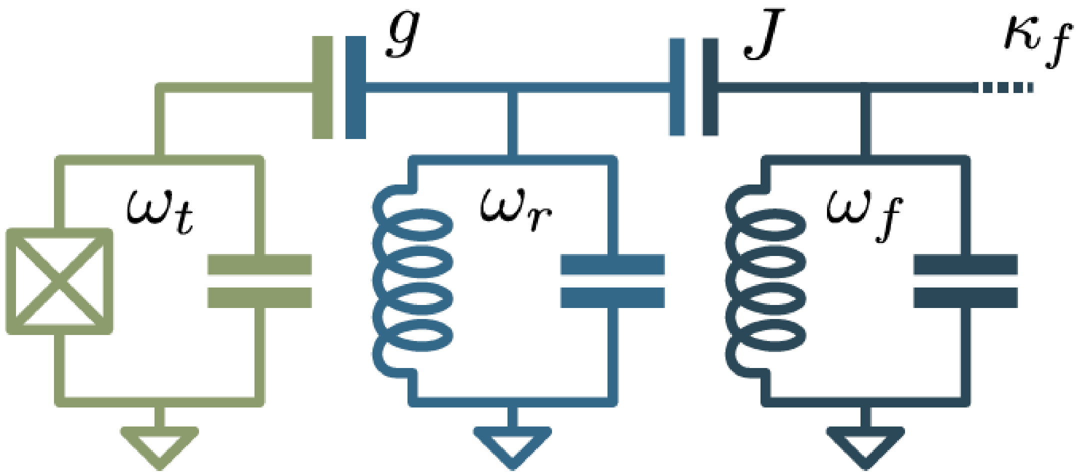


1 INSTITUT QUANTIQUE 2 ALICE & BOB 3 Inria

Dispersive readout

We want to optimize the dispersive readout of a transmon.

$$H = 4E_C n_t^2 - E_J \cos(\phi_t) + \omega_r a^\dagger a + \omega_f f^\dagger f + ig n_t (a^\dagger - a) + J(f^\dagger a + a^\dagger f)$$



- Full cosine model, including Purcell filter
- MW drive on Purcell filter and/or transmon

Difficult numerical problem

- ~400 parameters (1ns bins x 100ns x 2 drives)
- Hilbert space size ~ 8000 (5 x 40 x 40)
- GHz dynamics
- Open quantum system

Quantum optimal control

Gradient-free methods

- CRAB (Doria, PRL 2010)
- Nelder-Mead (Egger, PRL 2014)
- Model-free RL (Sivak, PRX 2022)

  ↳ Do not scale to many parameters

Gradient-based methods

- GRAPE (Boutin, PRA 2016)
- Krotov (Koch, JP:CM 2016)

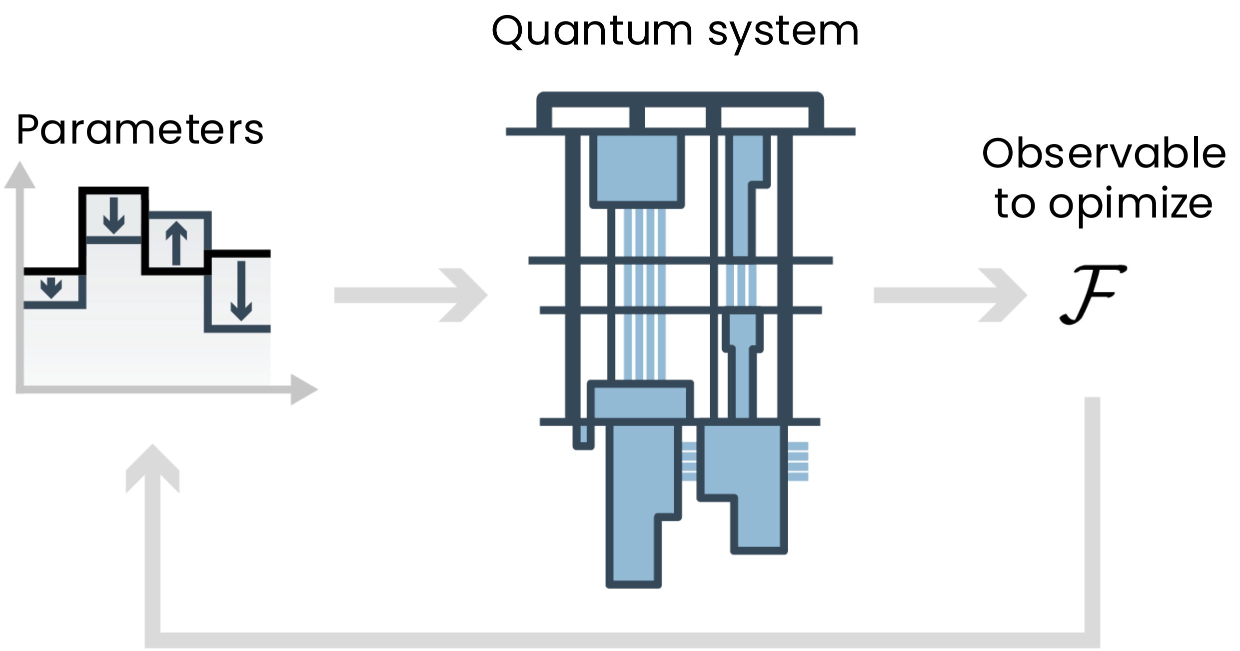
  ↳ Only state-to-state or gate optim.

- Automatic differentiation (Leung, PRA 2017)

  ↳ $O(N_T \times N^2)$ memory ➡ 4 Terabytes !

- Adjoint state method

  ↳ ✔ **Low memory + any closed or open system + any parametrized problem + fast**

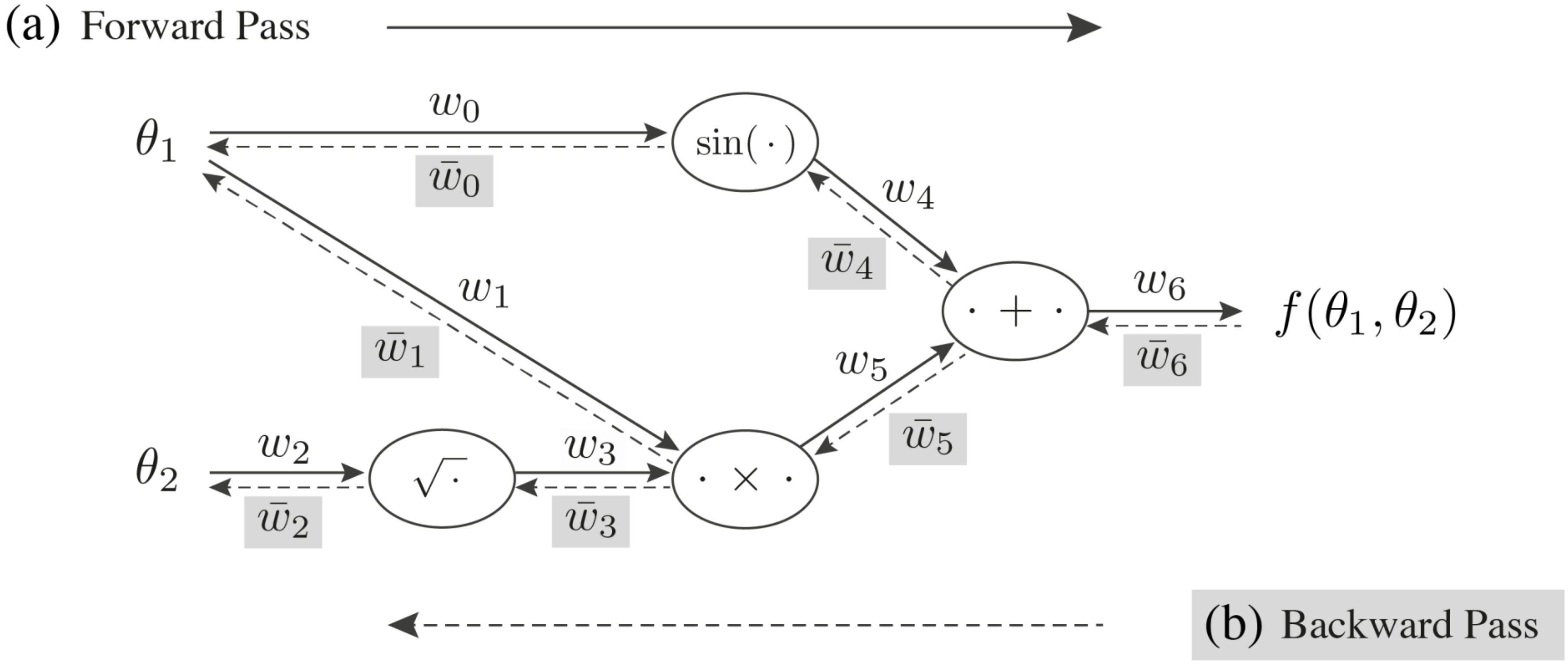


Primer on automatic differentiation

We want to differentiate the function

$$f(\theta_1, \theta_2) = \sin(\theta_1) + \theta_1\sqrt{\theta_2}$$

Graph of operations



Primer on automatic differentiation

We want to differentiate the function

$$f(\theta_1, \theta_2) = \sin(\theta_1) + \theta_1\sqrt{\theta_2}$$

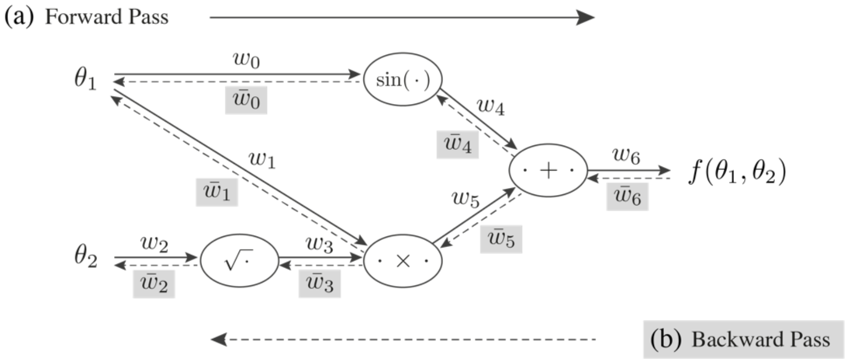


Table of operations

Forward pass		
$w_0 = \theta_1$		
$w_1 = \theta_1$		
$w_2 = \theta_2$		
$w_3 = \sqrt{w_2}$	$= \sqrt{\theta_2}$	
$w_4 = \sin w_0$	$= \sin\theta_1$	
$w_5 = w_1 w_3$	$= \theta_1\sqrt{\theta_2}$	
$w_6 = w_4 + w_5$	$= \sin(\theta_1) + \theta_1\sqrt{\theta_2}$	

Backward pass		
$\bar{w}_0 = \bar{w}_4 \frac{\partial w_4}{\partial w_0}$	$= \bar{w}_4 \cos(w_0)$	$= \cos(\theta_1)$
$\bar{w}_1 = \bar{w}_5 \frac{\partial w_5}{\partial w_1}$	$= \bar{w}_5 w_3$	$= \sqrt{\theta_2}$
$\bar{w}_2 = \bar{w}_3 \frac{\partial w_3}{\partial w_2}$	$= \bar{w}_3 / 2\sqrt{w_2}$	$= \theta_1 / 2\sqrt{\theta_2}$
$\bar{w}_3 = \bar{w}_5 \frac{\partial w_5}{\partial w_3}$	$= \bar{w}_5 w_1$	$= \theta_1$
$\bar{w}_4 = \bar{w}_6 \frac{\partial w_6}{\partial w_4}$	$= \bar{w}_6 \cdot 1$	$= 1$
$\bar{w}_5 = \bar{w}_6 \frac{\partial w_6}{\partial w_5}$	$= \bar{w}_6 \cdot 1$	$= 1$
$\bar{w}_6 = 1$		(seed)

Differentiating through a matrix multiplication requires storing the original matrices!

Adjoint state method

- Parametrized master equation

$$\frac{d\rho}{dt} = \mathcal{L}\rho = -i[H,\rho] + \sum \mathcal{D}[L_k]\rho$$

$\hookrightarrow H = H(\theta)$ $\quad\hookrightarrow L_k = L_k(\theta)$

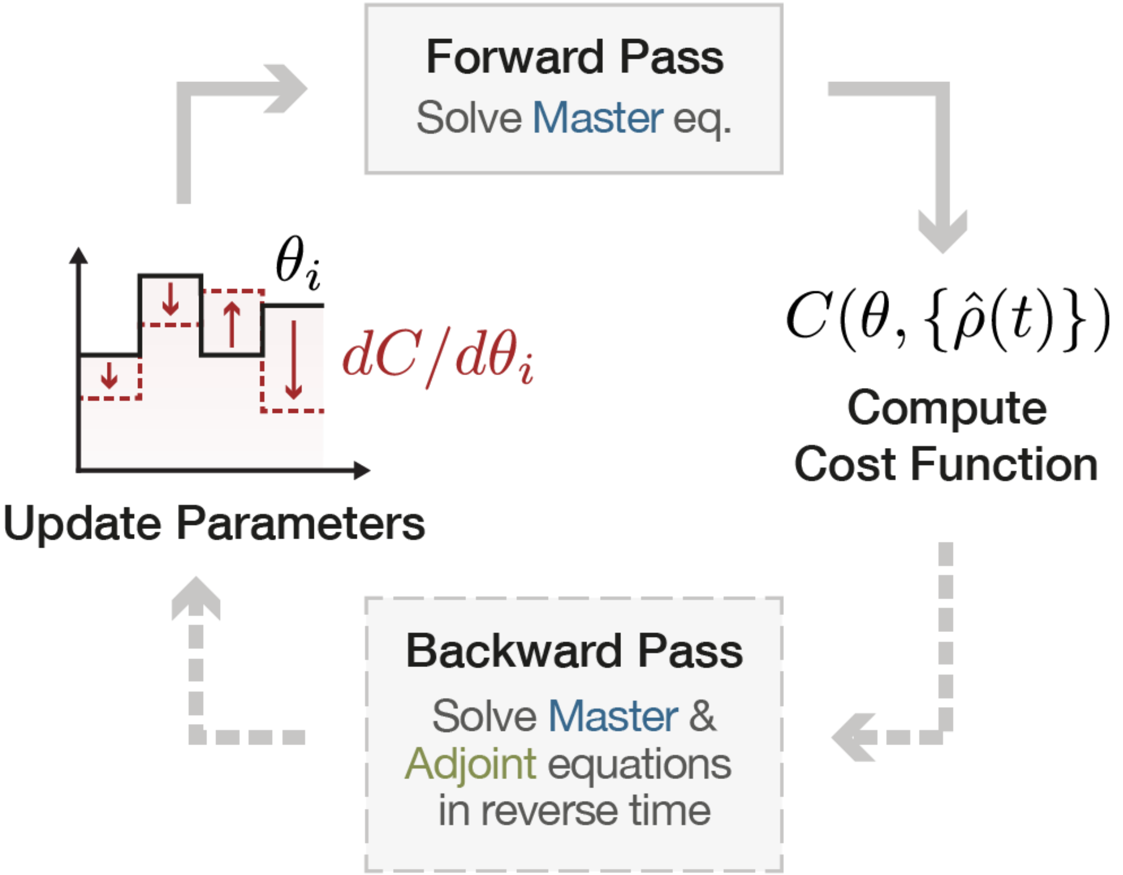
- Cost function $C = C(\theta, \rho(t_0), \ldots, \rho(t_n))$

- Adjoint state $\phi(t) = dC/d\rho(t)$

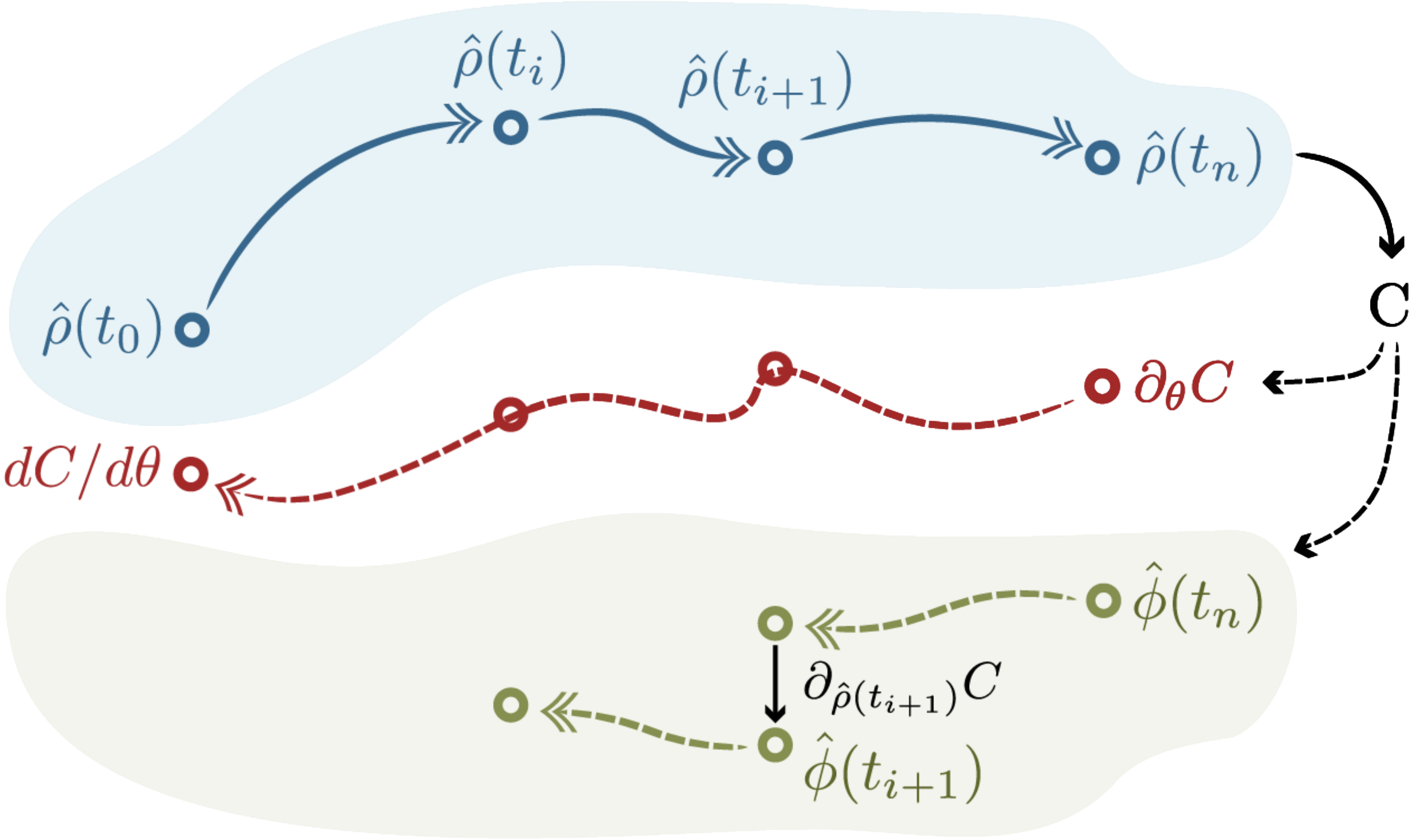
$$\frac{d\phi}{dt} = -\mathcal{L}^\dagger\phi = -i[H,\phi] - \sum \mathcal{D}^\dagger[L_k]\phi$$

- Explicit expression of gradient

$$\frac{dC}{d\theta} = \frac{\partial C}{\partial \theta} - \int_{t_0}^{t_n} \partial_\theta \mathrm{Tr}\left[\phi^\dagger(t)\mathcal{L}(t,\theta)\rho(t)\right] dt$$



Reverse-time backpropagation

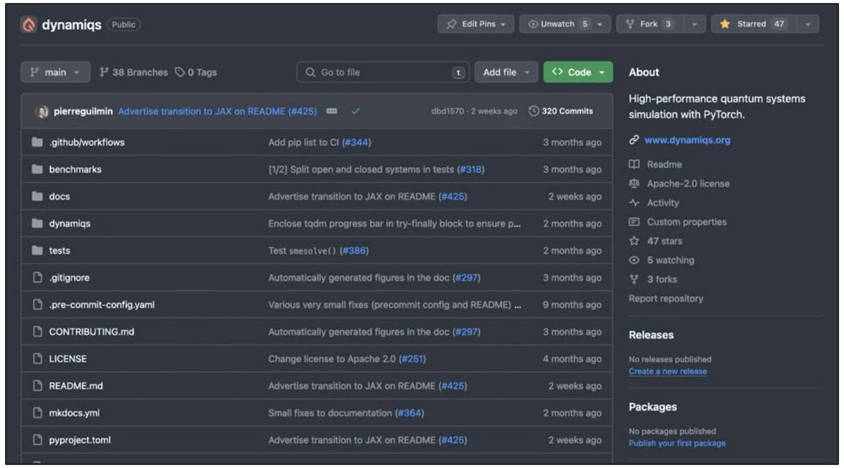


Memory: $\mathcal{O}(N^2)$ ➡ 488 MB

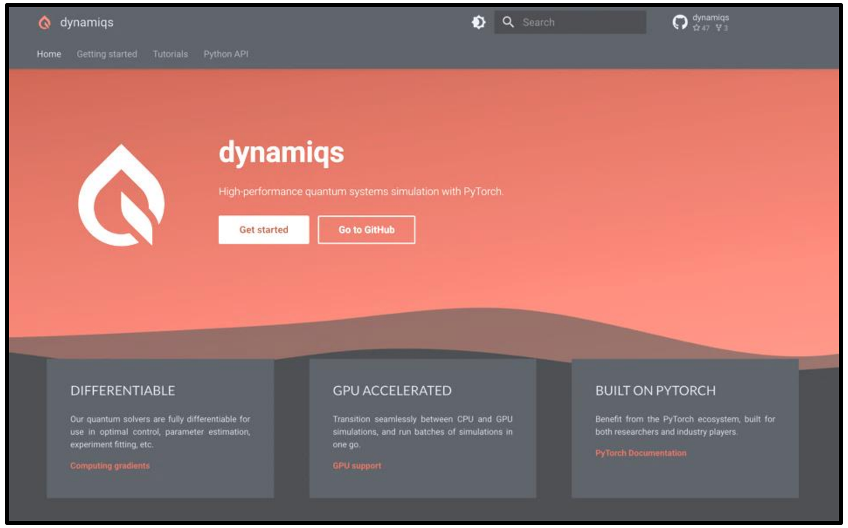
Optimal control with dynamiqs

- Open-source
- Closed, open and stochastic quantum systems
- End-to-end differentiable
- Works on GPUs → (very) fast simulations
- Batching
- QuTiP-like API
- …

github.com/dynamiqs



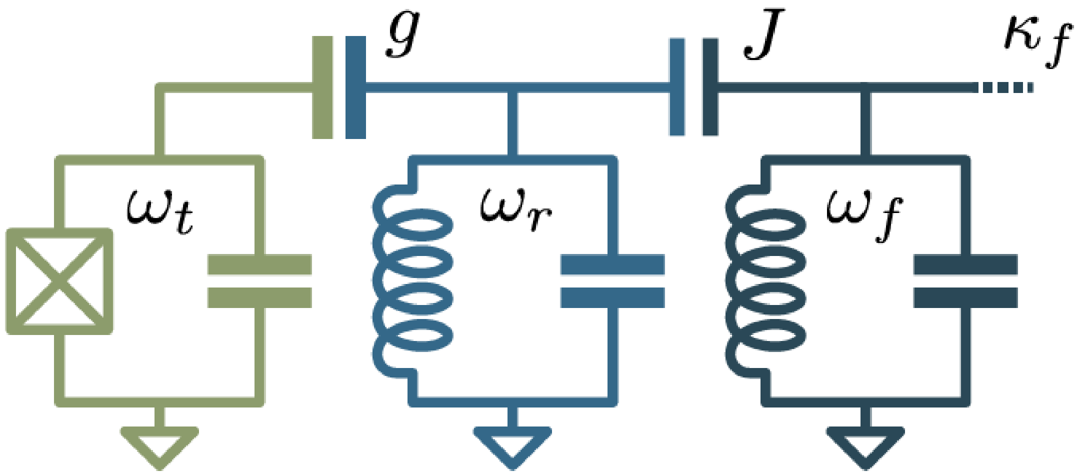
www.dynamiqs.org



Dispersive readout

We want to optimize the dispersive readout of a transmon.

$$H = 4E_C n_t^2 - E_J \cos(\phi_t) + \omega_r a^\dagger a + \omega_f f^\dagger f + ig n_t (a^\dagger - a) + J(f^\dagger a + a^\dagger f)$$



- Full cosine model, including Purcell filter
- MW drive on Purcell filter and/or transmon
- Optimisation with dynamiqs

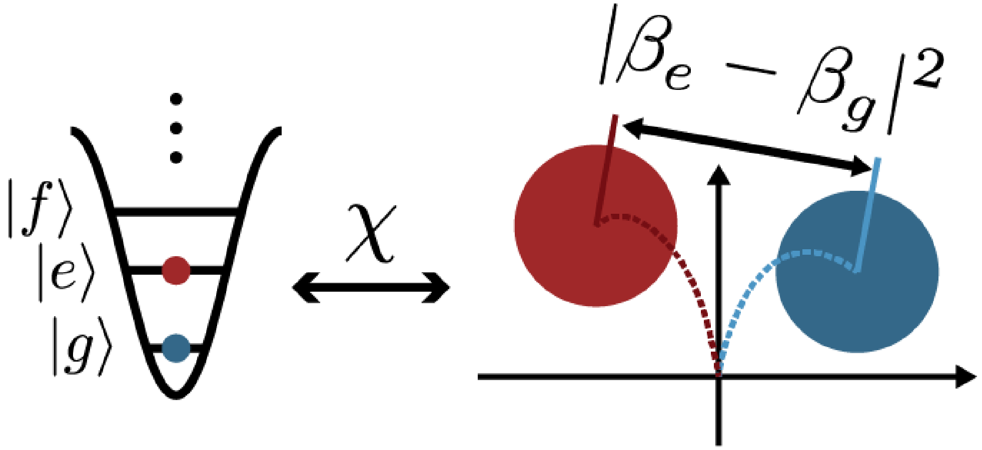
System parameters

$$E_J/2\pi = 16\,\text{GHz} \qquad \omega_t/2\pi = 6\,\text{GHz} \qquad \kappa_p/2\pi = 30\,\text{MHz} \qquad g/2\pi = 150\,\text{MHz}$$
$$E_c/2\pi = 315\,\text{MHz} \qquad \omega_r/2\pi = 7.2\,\text{GHz} \qquad \kappa_q/2\pi = 8\,\text{KHz} \qquad J/2\pi = 30\,\text{MHz}$$
$$E_J/E_c \approx 51 \qquad \omega_p/2\pi = 7.21\,\text{GHz} \qquad \bar{n}_{\text{crit}} = 16$$

Optimizing transmon readout

Signal-to-noise ratio (Bultink et al., 2017)

$$\mathrm{SNR} = \sqrt{2\eta\kappa_f \int_0^{\tau_m} dt\, |\beta_e - \beta_g|^2}$$

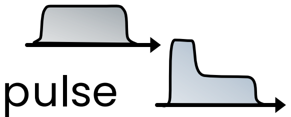


Optimizing transmon readout

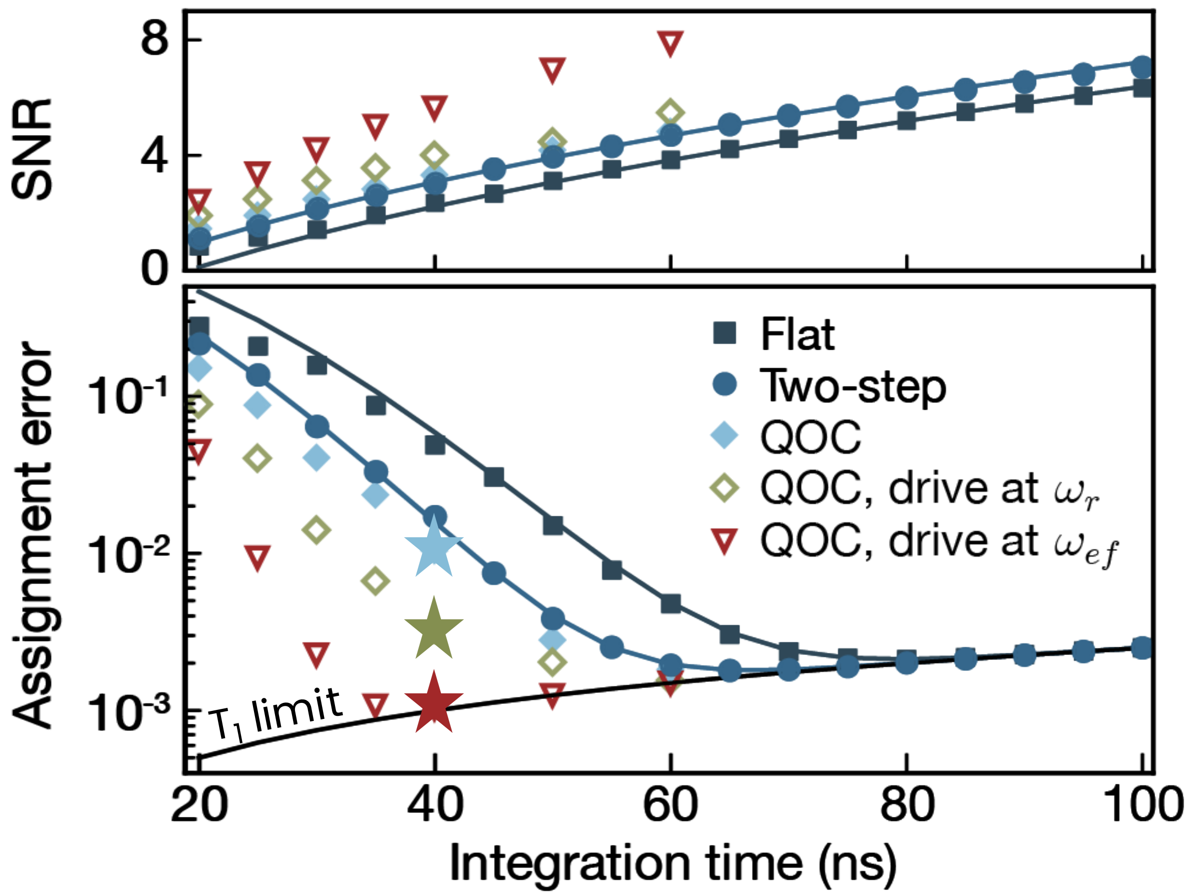
Signal-to-noise ratio (Bultink et al., 2017)

$$\mathrm{SNR} = \sqrt{2\eta\kappa_f \int_0^{\tau_m} dt\,|\beta_e - \beta_g|^2}$$

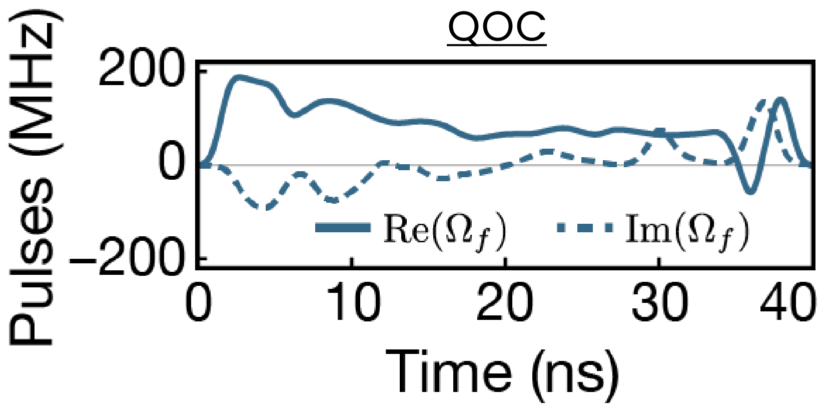
- 2 reference pulses
  - Flat pulse
  - Two-step pulse



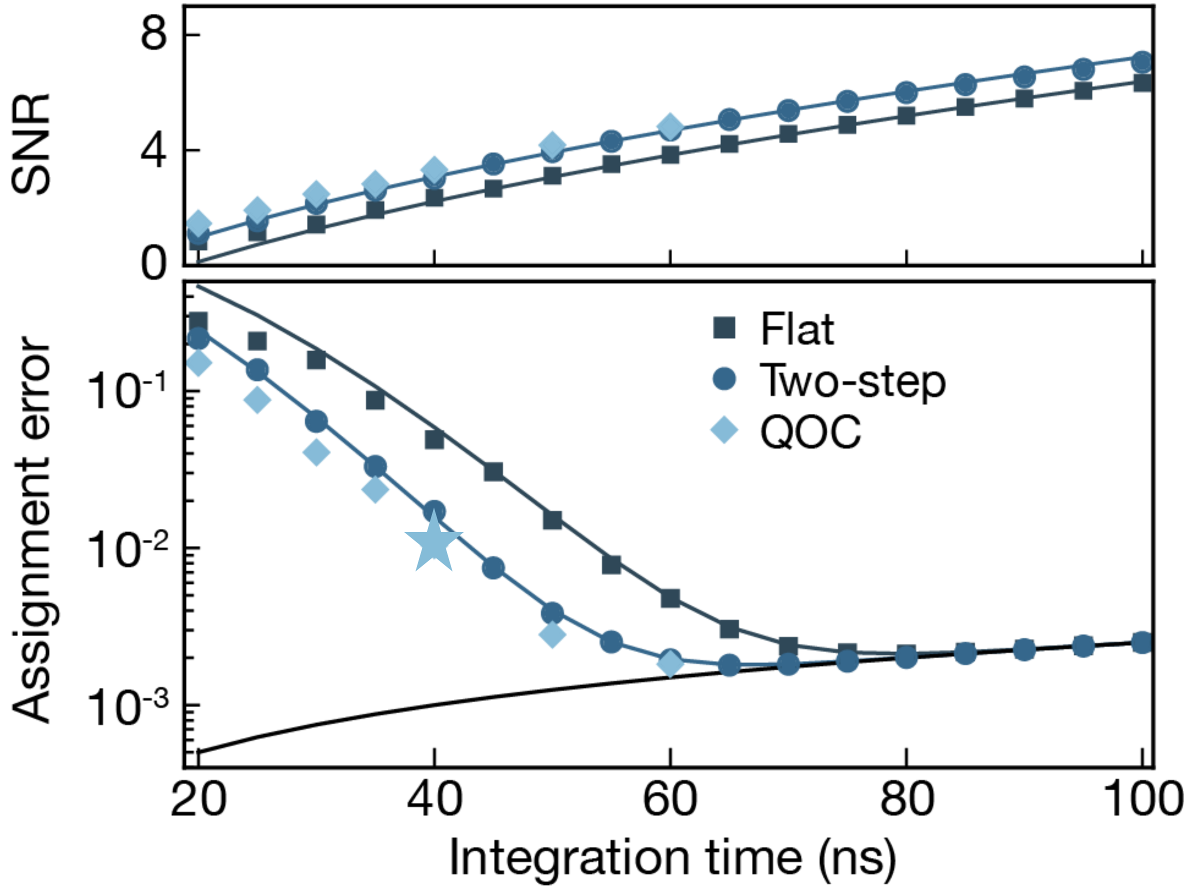
- 3 optimized pulses
  - Drive filter
  - Drive filter + transmon @ w_r
  - Drive filter + transmon @ w_{ef}
- Optimize pulse envelopes + carrier frequencies
- Fair comparison: limit $n < n_{crit}$



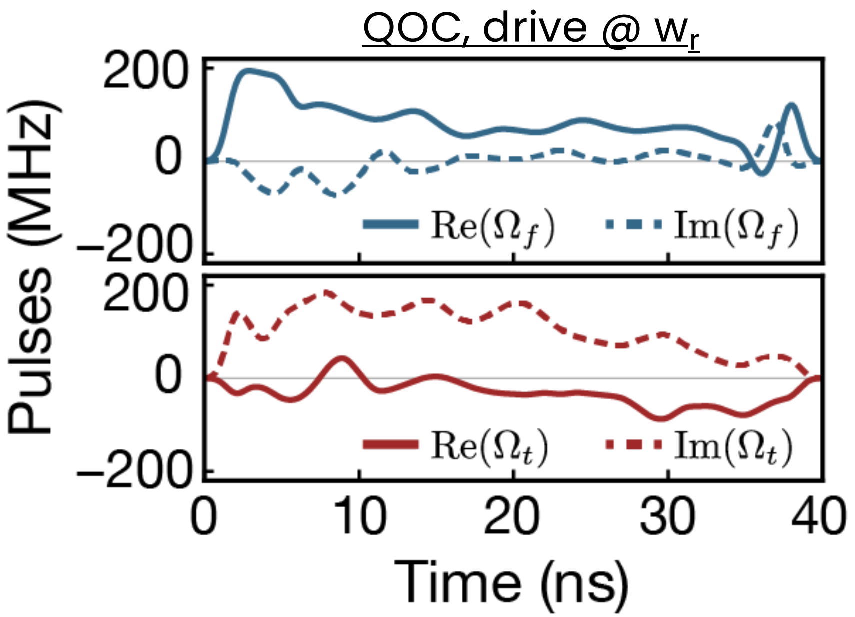
Optimizing towards a two-step pulse

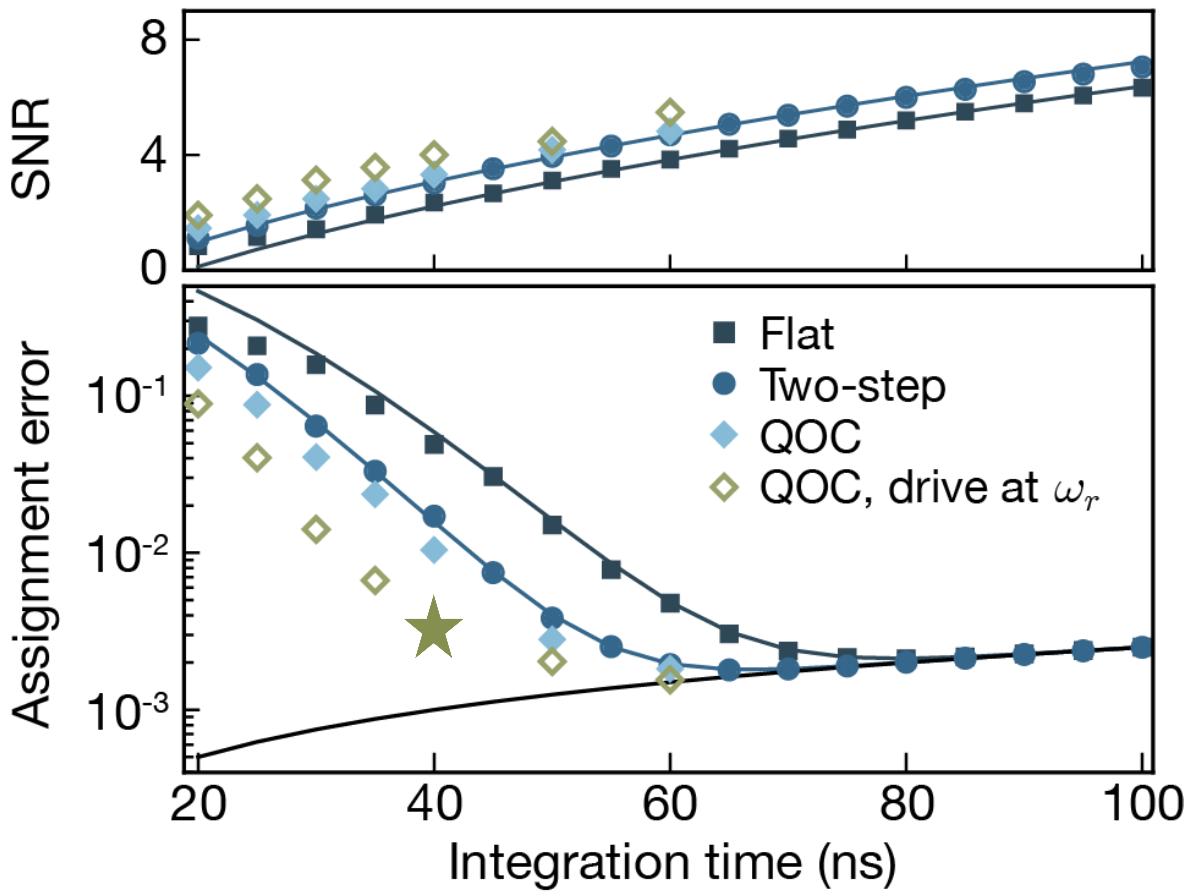


- Envelope similar to two-step (Walter, PRApplied 2017)
- Already optimal
- Limited by strength of dispersive coupling



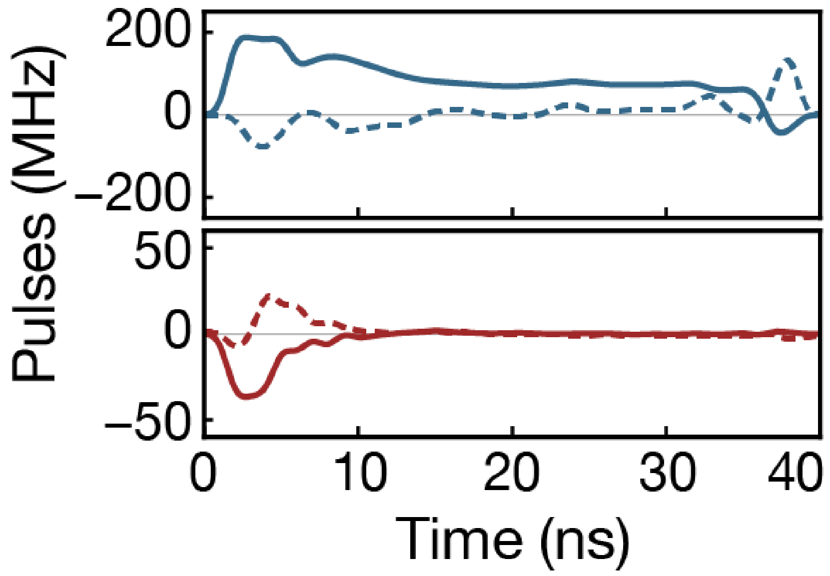
Multichannel driving

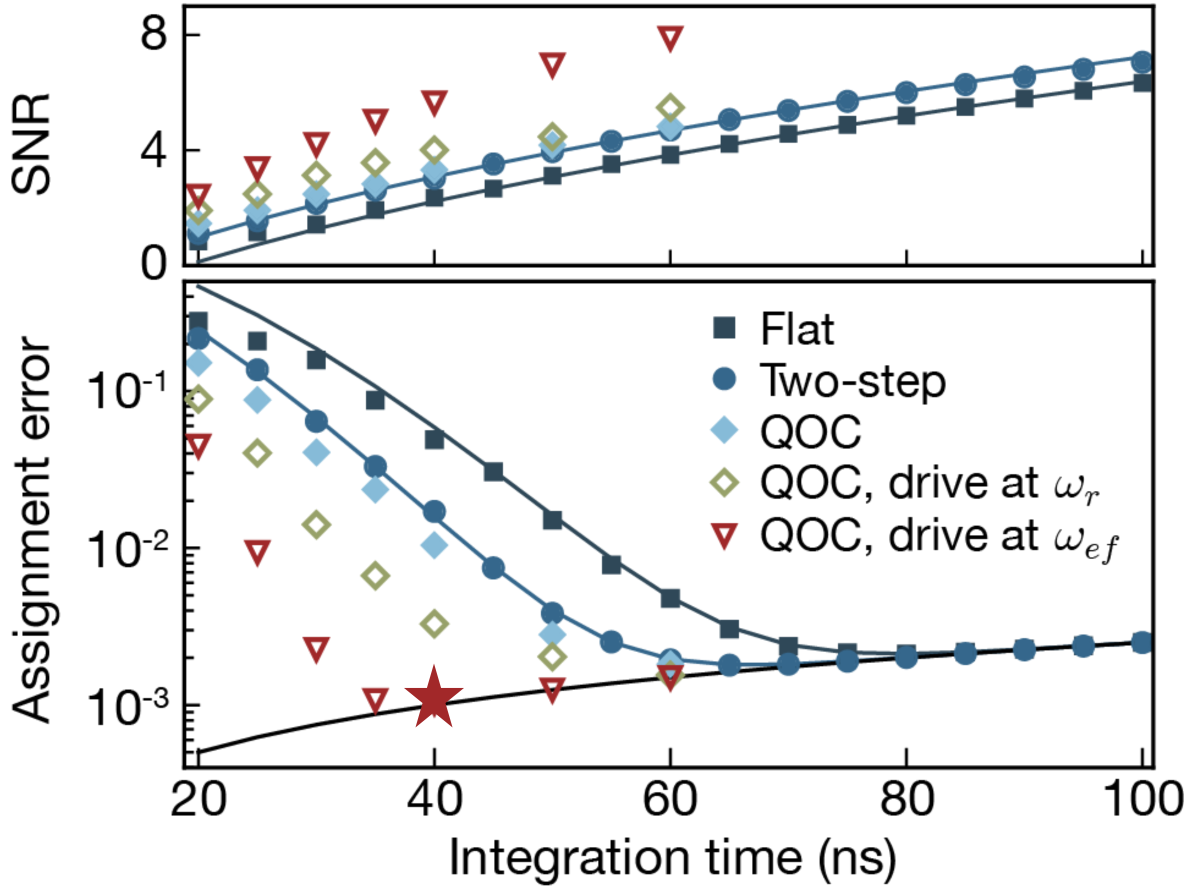




- Need another ingredient
- Similar to (Ikonen, PRL 2019) & (Touzard, PRL 2019)
- Drive transmon at w_r → displace origin of resonator phase-space

Shelving





- Shelving (Elder, PRX 2020) & (Hann, PRA 2018)
  → use |f> state with larger coupling
- Filter envelope similar to two-step
- 10ns pi-pulse with DRAG & stark-shift
- **x2 improvement in readout time**

01. Optimal control w/ **adjoint state method**: low-memory, fast, generic

02. **Fast transmon readout** with additional drive on the transmon

03. **Realistic pulses** and known strategies found by optimizer

ALICE & BOB TALKS

Purcell filter trajectories

|g> and |e> trajectories in the Purcell filter → enhanced integrated distance