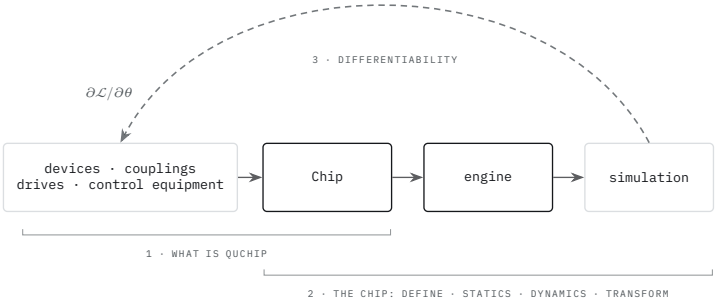
A Differentiable Toolkit for Modeling Quantum Devices

Ibraheem AlYousef

King Fahd University of Petroleum & Minerals

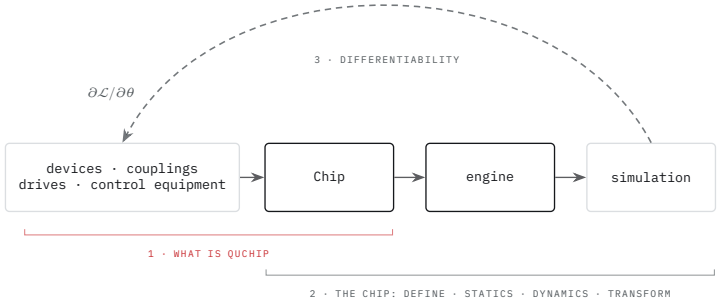
pip install quchip · arXiv:2607.17081 · quchip.org



SECTION 1

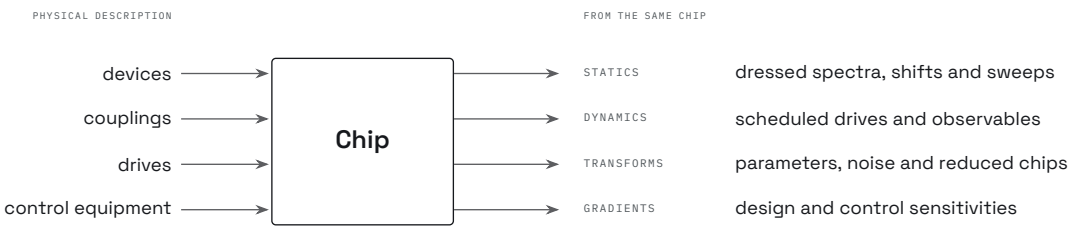
What is quchip

what must the physical model know from you?



quchip is built around the Chip

an open-source Python toolkit for the static and dynamic properties of quantum chips



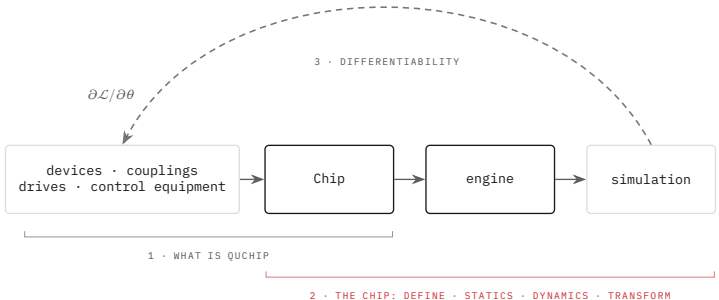
The physical model has four parts

DEVICES	the local Hamiltonians of each element	$\omega\,\hat{n} + \frac{\alpha}{2}\,\hat{n}(\hat{n}-1)$
COUPLINGS	how pairs of devices interact	$g\,(\hat{a}+\hat{a}^\dagger)(\hat{b}+\hat{b}^\dagger)$
DRIVES	how a control field couples to a device	$\varepsilon(t)\,(\hat{a}+\hat{a}^\dagger)$
CONTROL EQUIPMENT	how the classical signal chain modifies the control field	$\varepsilon \mapsto G\,\varepsilon(t-\tau), \quad \beta\,\varepsilon \to$ neighbour
THE CHIP	the assembled Hamiltonian in the selected frame and approximation	$H = \sum H_i + \sum H_{ij} + \sum s(t)\,O$

SECTION 2

The Chip

define · statics · dynamics · transform

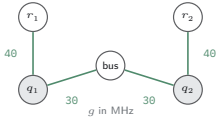


Each component contributes a Hamiltonian term

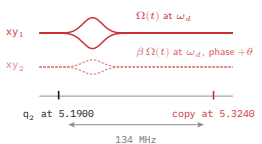
$$H_{\text{lab}}(t) = \sum_{k=1}^{2}\left[\omega_k \hat{n}_k + \tfrac{\alpha_k}{2}\hat{n}_k(\hat{n}_k - 1)\right] + \sum_{j\in\{b, r_1, r_2\}} \omega_j \hat{n}_j + \sum_{\text{edges}} g\,(\hat{a} + \hat{a}^\dagger)(\hat{b} + \hat{b}^\dagger) + \sum_{k=1}^{2} \varepsilon_k(t)\, i(\hat{a}_k - \hat{a}_k^\dagger)$$

DEVICES

COUPLINGS

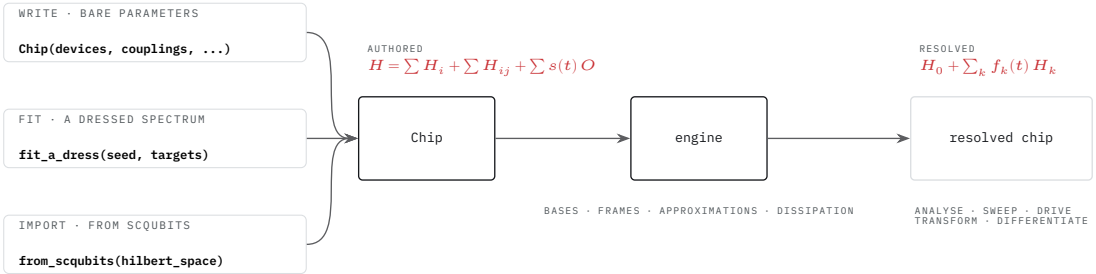


CONTROL LINES




```
q1  = DuffingTransmon(freq=5.3262, anharmonicity=-0.261, levels=4)
q2  = DuffingTransmon(freq=5.1919, anharmonicity=-0.263, levels=4)
bus, r1, r2 = Resonator(freq=6.30, levels=4), Resonator(freq=6.559, levels=3), Resonator(freq=6.658, levels=3)
couplings = [Capacitive(q1, bus, g=0.030), Capacitive(bus, q2, g=0.030),
             Capacitive(q1, r1, g=0.040), Capacitive(q2, r2, g=0.040)]
xy1, xy2 = ChargeDrive(target=q1), ChargeDrive(target=q2)
beta, theta = [[1, 0.14], [0.16, 1]], [[0, 1.885], [3.31, 0]]
eq = ControlEquipment([xy1, xy2]); eq.set_crosstalk_matrix(beta, theta)
chip = Chip([q1, q2, bus, r1, r2], couplings=couplings, control_equipment=eq, frame="rotating", approximation=RWA())
```

Defining and resolving a Chip



fit targets: freq · α · χ · ZZ · exchange · g import: HilbertSpace → Chip

The Chip's statics

SPECTRAL OBSERVABLES

dressed transitions, conditional shifts, χ and ZZ

$\tilde{\omega}_{01} \quad \tilde{\omega}_{01}\,|\,q_2{=}1 \quad \tilde{\omega}_{12} \quad \chi \quad \zeta_{ZZ} \quad \widetilde{|q_1{=}1\rangle}$

SWEEP · FIG 1

vary a bare parameter and track the dressed spectrum

HAMILTONIAN

in the selected frame after the RWA

$$\sum_i \omega_i \hat{n}_i + \tfrac{\alpha_i}{2}\hat{n}_i(\hat{n}_i - 1) + \sum_i g_i\,(\hat{a}_i + \hat{a}_i^\dagger)(\hat{b} + \hat{b}^\dagger)$$

↓ engine $\quad \hat{H}_0 + \sum_k f_k(t)\,\hat{H}_k \quad$ 4 dropped terms

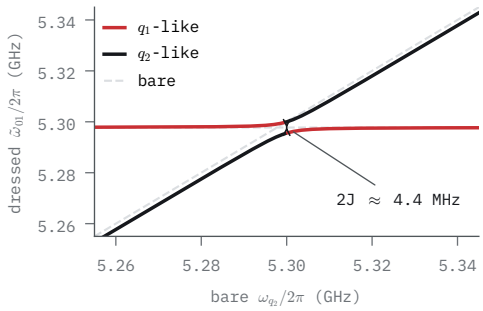


FIG 1 · BARE ω_{q_2} SWEPT ACROSS q_1

The Chip's dynamics

SCHEDULE

pulses on named control lines

OBSERVABLES · FIG 2

solve in the required frame; report X, Y and Z in the device frame

BATCH · FIG 3

sweep pulse parameters or initial states

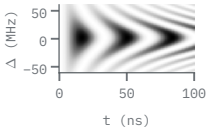


FIG 3 · ONE BATCH

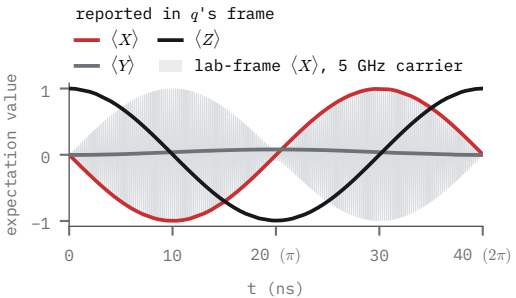


FIG 2 · LAB-FRAME SOLVE, REPORTED IN THE QUBIT FRAME

SOLVERS · QuTiP 5 [Lambert et al., Phys. Rep. 1153, 1 (2026)] · dynamiqs [Guilmin et al., 2025, in preparation]

Chip transformations

REBIND · SAME STRUCTURE

change physical parameters or replace device noise and baths; topology unchanged

REDUCE · APPROXIMATE

`eliminate` removes a selected device or coupling and retains its effective shifts, losses and couplings

$$\frac{g^2}{\Delta} \to \omega \qquad \left(\frac{g}{\Delta}\right)^2 \kappa \to \Gamma_1 \qquad \frac{g_1 g_2}{\Delta} \to J$$

g/Δ = 0.05 · 0.05 · 0.03 · 0.03; block gap ≥ 1.1 GHz

REDUCE FOR A SEQUENCE · ACTIVE PATCH

keep the driven neighbourhood; eliminate the other connected devices and retarget the controls

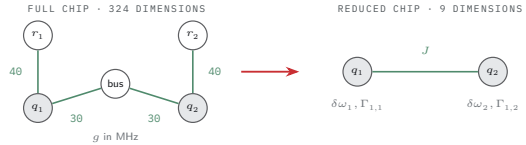
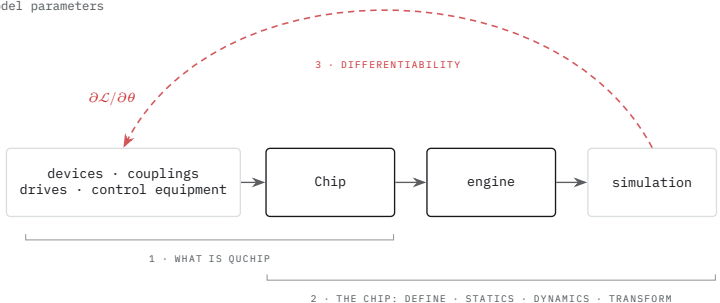


FIG 4 · 3×3 LATTICE, g = 12 MHz · FOUR FOLDS ON RECORD

SECTION 3

Differentiability

derivatives with respect to the physical model parameters



Gradients through statics, dynamics and calibration

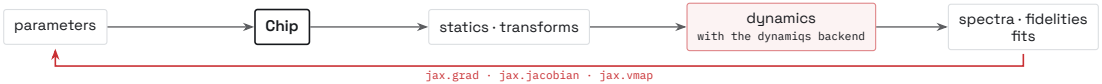


FIG 5 · STATIC CHIP

$\Delta p_i/p_i = 1\,\% \;\rightarrow\; \Delta\zeta_{ZZ}$

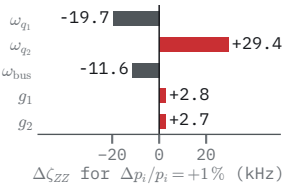


FIG 6 · DRIVEN CHIP

pulse error $\rightarrow$ ΔF_π

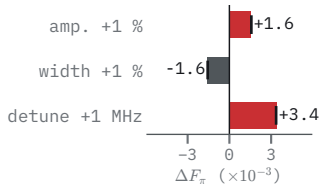
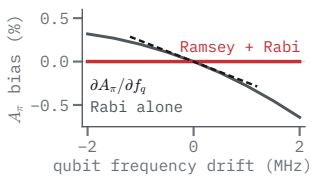


FIG 7 · CALIBRATION ROUTINE

qubit drift $\rightarrow$ ΔA_π



Conclusions

1 The Chip stores the physical model

built-in or user-defined devices · couplings · drives · dissipation

2 The engine resolves it for each calculation

dressed spectra and sweeps · driven dynamics · reductions with each fold recorded

3 Gradients pass through complete calculations

spectra · time-domain solves · calibration fits

quchip is open source and available on PyPI. If quchip cannot yet model your chip, let's talk about it.

```
pip install quchip
```

github.com/quchip/quchip

arXiv:2607.17081

Thank you

quchip.org/sqa-2026 · the talk and the examples

Local basis projection and reference-frame transformation

LOCAL SOLVER BASIS

1 · AUTHORED DEVICE

one isolated device: Hamiltonian H_i and operators O_i

2 · DIAGONALIZE AND DEFINE P_i

$$H_i|n_i\rangle = E_{i,n}|n_i\rangle$$

$$P_i = [|0_i\rangle, \dots, |d_i - 1\rangle]$$

3 · PROJECT BEFORE ASSEMBLY

$$H_i, O_i \longmapsto P_i^\dagger (H_i, O_i) P_i$$

basis="native": keep authored basis · basis="eigen": use P_i

same P_i projects H_i, drives and dissipation

couplings and global dressing come afterward

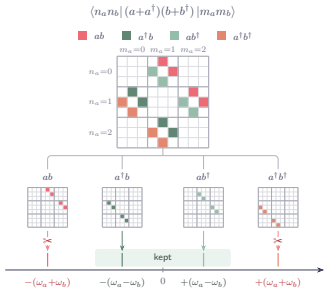
REFERENCE FRAME

$$a_i \longmapsto a_i \, e^{-i 2\pi f_{\mathsf{f},i} t}$$

$$f_i n_i \longmapsto (f_i - f_{\mathsf{f},i}) n_i$$

Lab, rotating and supplied reference frequencies set each term's phase and detuning. Term selection occurs later, when the approximation is applied.

Structural RWA from operator-band decomposition



BAND DECOMPOSITION OF ONE COUPLING TERM

$$O = \sum_{\mathbf{w}} O_{\mathbf{w}}$$

RWA : $\sum_i w_i = 0$

EXACT

keeps every operator band

RWA

keeps structurally stationary bands

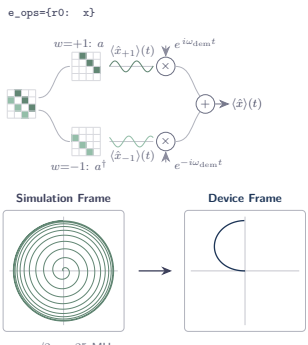
KEEP_BANDS

uses an explicit structural selection

discarded bands remain inspectable

SCOPE Floquet and higher-order corrections require a separate calculation.

Observable bands are demodulated after the solve



OBSERVABLE BANDS

$$\langle O \rangle(t) = \sum_{\mathbf{w}} e^{+i\,2\pi\,\mathbf{w}\cdot\mathbf{f}_{\mathrm{dem}}\,t} \langle O_{\mathbf{w}} \rangle_{\mathrm{solve}}(t)$$

$\mathbf{f}_{\mathrm{dem}} = \mathbf{f}_{\mathrm{reference}} - \mathbf{f}_{\mathrm{solve}}$

$\mathbf{w} = 0$

populations and number-like observables are unchanged

$\mathbf{w} = \pm 1$

transverse observables regain the phase required by the reporting frame

POST-PROCESSING · PHASE EACH OBSERVABLE BAND, THEN SUM

Validate the reduced model before and after the solve

BEFORE THE SOLVE

$$324 \longrightarrow 9$$

WEAK MIXING

$g/\Delta = 0.05, 0.05, 0.03, 0.03$

SPECTRAL SEPARATION

eliminated block at least 1.1 GHz away

AFTER THE SOLVE

Run the same pulse sequence on the full and reduced models.

COMPARE RETAINED POPULATIONS

$$\max_t |P_{\text{full}}(t) - P_{\text{red}}(t)| \leq 6 \times 10^{-3}$$

The residual checks the approximation over the pulse, not only at the final time.

IF THE ELIMINATED BLOCK IS NOT WELL SEPARATED, USE THE FULL MODEL

A new component defines one local physics method

declare parameter fields · return an operator expression in the component's local space

DEVICE · local_hamiltonian

```
class Resonator(FockDevice):
  freq: Scalar = parameter(
    unit="GHz")
  def local_hamiltonian(self, op, p):
    return p.freq * op.n
```

Fields become constructor arguments and parameter bindings.

COUPLING · interaction

```
class Exchange(CouplingModel):
  g: Scalar = parameter(unit="GHz")
  def interaction(self, a, b, p):
    return p.g * (a.a * b.adag
         + a.adag * b.a)
```

Author the complete interaction; the chip-level approximation comes later.

DRIVE · hamiltonian

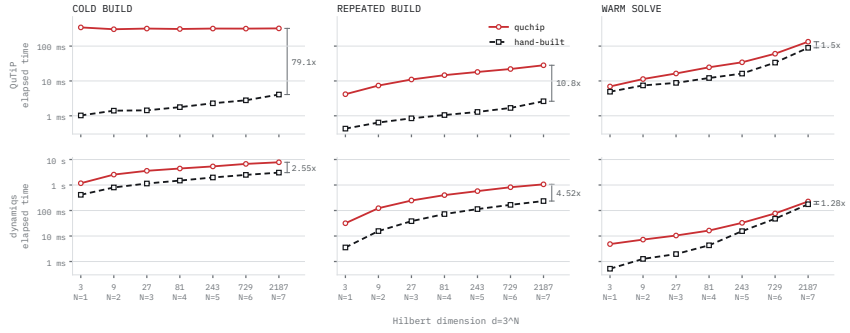
```
class ChargePhaseDrive(DeviceDrive):
  def hamiltonian(self, target, signal):
    charge = target.charge_coupling_operator()
    phase = target.phase_coupling_operator()
    return signal.i*charge - signal.q*phase
```

The method receives the delivered I/Q signal; override `signal` only to reshape it.

ENGINE RESPONSIBILITIES

parameter validation · basis projection · tensor embedding · frames and RWA · backend lowering

Elapsed time shows which gaps close as the model grows



SOLID = QUCHIP · DASHED = HAND-BUILT · BRACKETS SHOW THE $N = 7$ RATIO

imports excluded · first solve / JAX compilation not shown · quchip v0.2.0 · parity $< 10^{-5}$ · CI 32280028936 · commit 6085f82