Dynamiqs, a library for GPU-accelerated and differentiable quantum simulation

Pierre Guilmin[12], Ronan Gautier[123], Adrien Bocquet[12], Élie Genois[3], Bogdan Agrici[1]

IEEE QCE 24', Advanced Simulations of Quantum Computations

[1]Alice & Bob [2]ENS Paris [3]University of Sherbrooke

DYNAMIQS GITHUB

ALICE & BOB

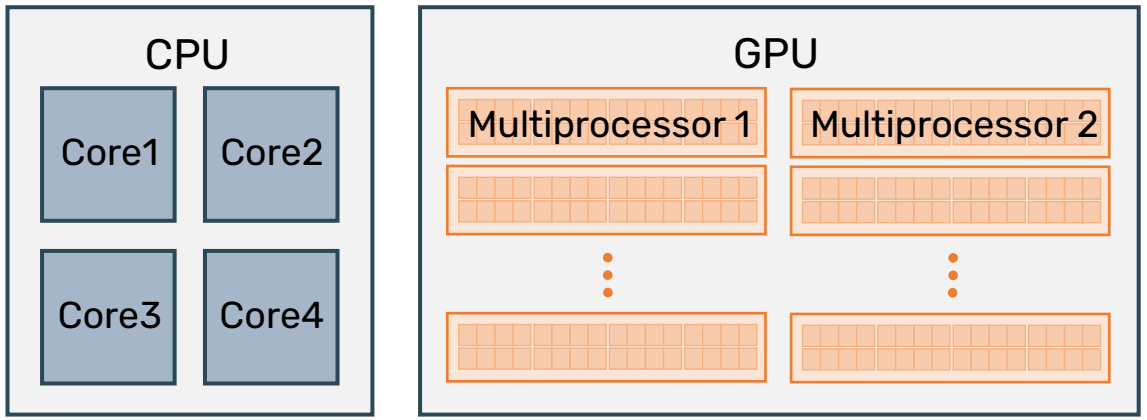
Why GPUs?

Bottleneck of solving a SE/ME/SME is **matrix products** (ODE solvers, propagator, Monte Carlo, …)

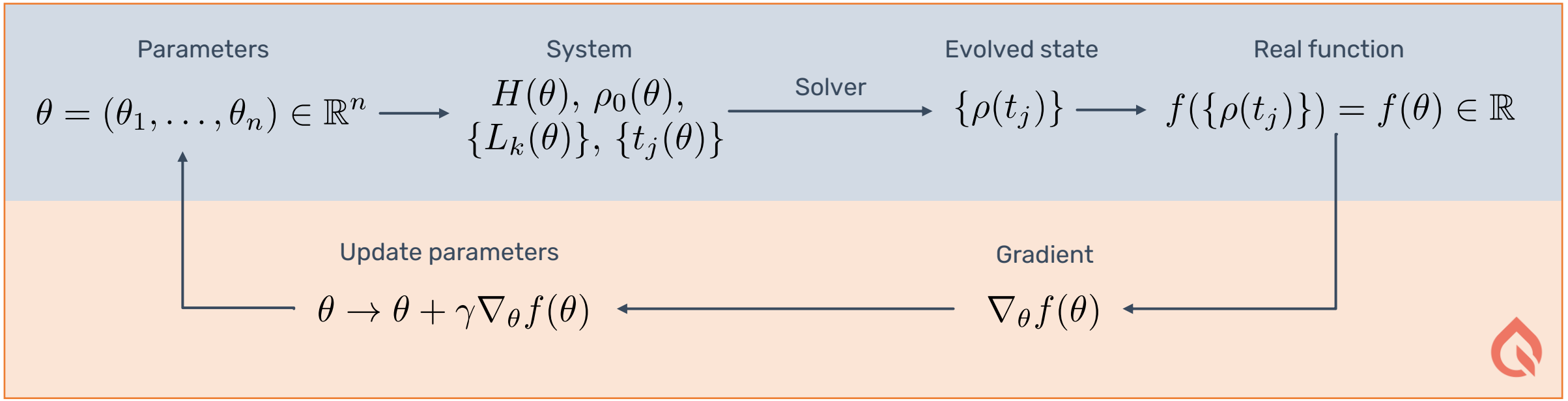
Example: Euler method for ME

$$\rho(t+dt) = \rho(t) - i[H, \rho(t)] + \sum \left(L\rho(t)L^\dagger - \tfrac{1}{2}\{L^\dagger L, \rho(t)\} \right)$$

➡ Leverage specialized hardware



Differentiable solvers



Project philosophy: fast and reliable **building block**

- Quantum optimal control
- Parameter estimation
- State tomography
- Sensitivity analysis
- …

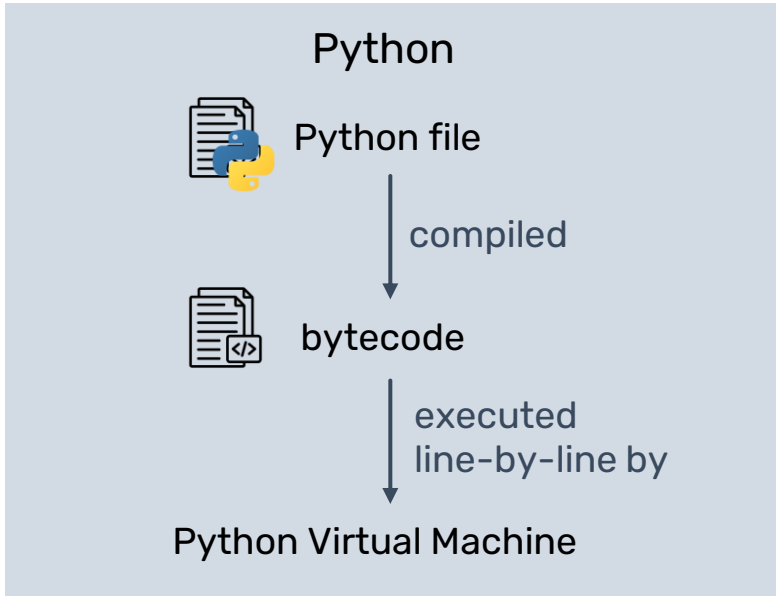
Computing gradients:

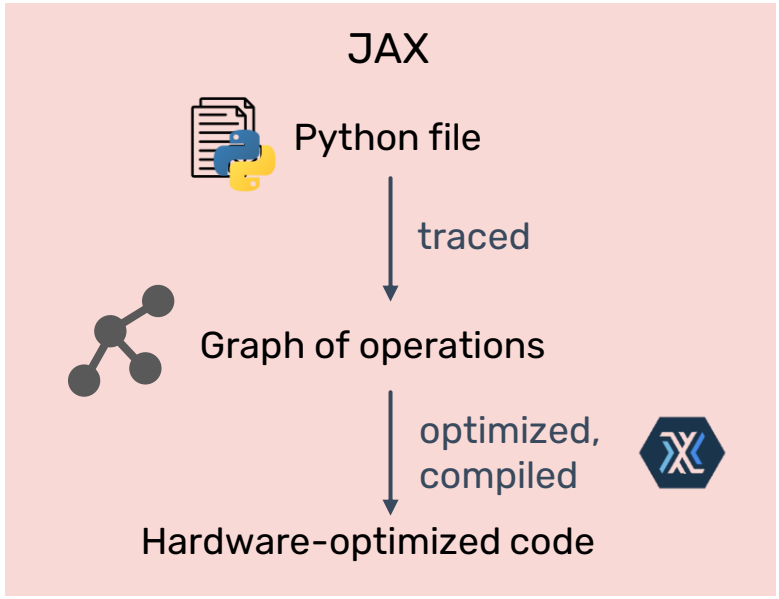
- Automatic differentiation
  → Fast and reliable, but large memory
- Adjoint state method*
  → Low memory, but slower
- Recursive checkpointing
  → Very strong tradeoff (recommended)

*see Gautier, Genois and Blais, arXiv:2403.14765

Under the hood: JAX and diffrax

Linear algebra on GPUs + automatic differentiation
→ same tools as machine learning
→ Dynamiqs built on JAX (Google) and Diffrax (Patrick Kidger)





- Interpretation overhead
- Dynamic typing
- No low-level optim.

- Fused operations
- Memory access optimisations
- Loop unrolling

All ODE solving is handled by Diffrax, a specialized library built on JAX

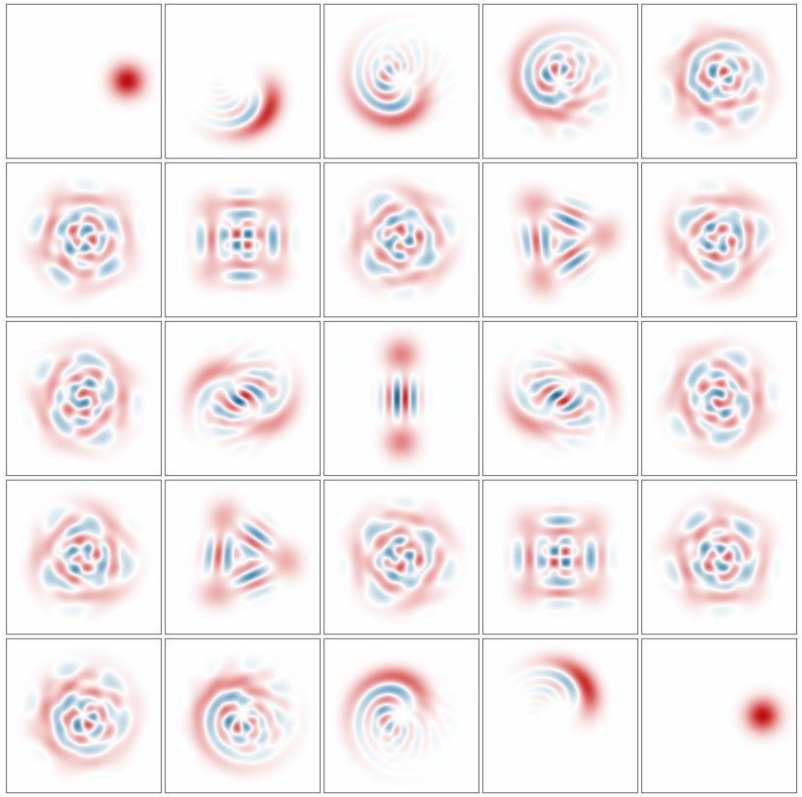
A QuTiP-like API

```python
import dynamiqs as dq
import numpy as np
dq.set_layout('dense')

# define model
n = 16                                  # Hilbert space dimension
a = dq.destroy(n)                       # annihilation operator
H = a.dag() @ a.dag() @ a @ a           # Kerr Hamiltonian
psi0 = dq.coherent(n, 2.0)              # coherent state
tsave = np.linspace(0, np.pi, 101)      # save times

# run simulation
result = dq.sesolve(H, psi0, tsave)

# plot results
dq.plot_wigner_mosaic(result.states, n=25, nrows=5, xmax=3.5)
```



- QuTiP-like API, with **small** differences when appropriate (e.g. time-dependence)
- **Compatible with QuTiP** objects
- Smoothly runs on GPUs, computes gradients, or **set global settings** (matrix layout, precision)

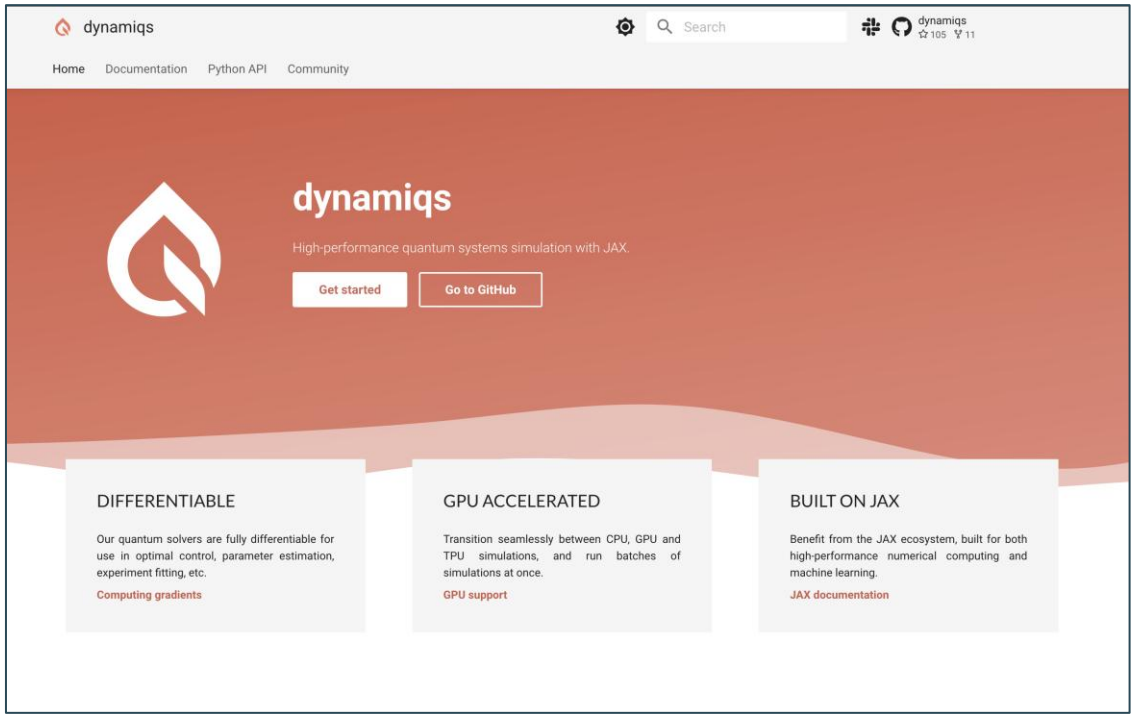
Dynamiqs in a nutshell

An open-source Python library based on JAX, for the simulation of

- the Schrödinger equation
- the Lindblad master equation
- stochastic master equations
- …

With

- CPU and **GPU** support
- **Batching**
- End-to-end **differentiability**
- Tailored **sparse** support
- QuTiP-like **API**



www.dynamiqs.org