Tensor-Programmable Quantum Circuits for Differential Equations

High Performance Computing Workshop – Leogang, 2025/02/24

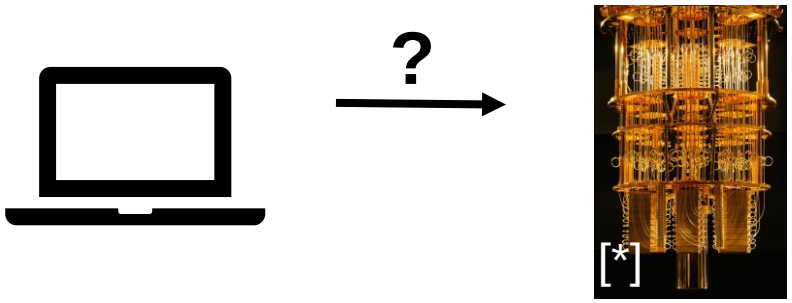
Pia Siegl, Greta Reese, Nis van Hülst, Tomohiro Hashizume, Dieter Jaksch

DLR

Computational Fluid Dynamics on Quantum Computers

- **why** should we solve **classical differential equations** on quantum computers?
  - beneficial scaling?
  - reduce computational cost?

- **how** can we solve classical partial differential equations on quantum computers?



[*] IBM-research_kryostat.jpg (2048×3128) (higgs.ch)

Example: Diffusion Equation

$$\frac{df(x,t)}{dt} = c_d \Delta f(x,t)$$

explicit Euler time steps:

$$f^{j+1} = f^j + \Delta t \cdot c_d \Delta f^j$$

spatial discretization (finite differences):

$$f^j \rightarrow \begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^j, \Delta \rightarrow \frac{1}{\Delta x^2} \begin{pmatrix} -2 & 1 & \cdots & 0 & 0 \\ 1 & -2 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & -2 & 1 \\ 0 & 0 & \cdots & 1 & -2 \end{pmatrix}$$

classical implementation:

$$\begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^{j+1} = \begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^j + \frac{\Delta t \cdot c_d}{\Delta x^2} \begin{pmatrix} -2 & 1 & \cdots & 0 & 0 \\ 1 & -2 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & -2 & 1 \\ 0 & 0 & \cdots & 1 & -2 \end{pmatrix} \begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^j$$

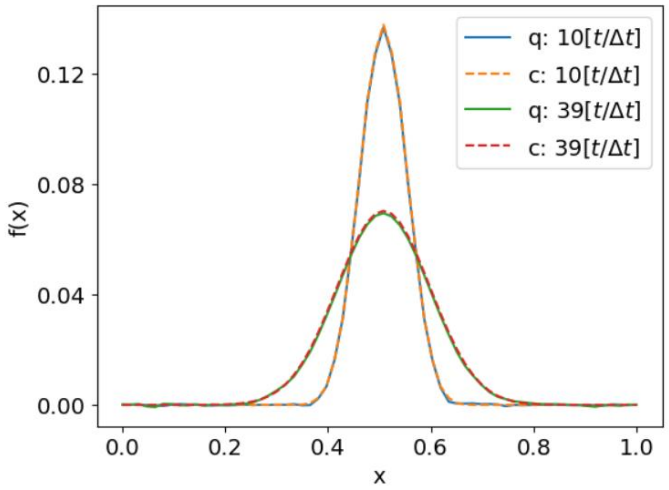
Δ = Laplace Operator
c_d = diffusion constant
Δt = time step size
Δx = spatial grid size

How to Port this on a Quantum Computer?

classical implementation:

$$\begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^{j+1} = \begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^{j} + dt \begin{pmatrix} -2 & 1 & \cdots & 0 & 0 \\ 1 & -2 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & -2 & 1 \\ 0 & 0 & \cdots & 1 & -2 \end{pmatrix} \begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^{j}$$

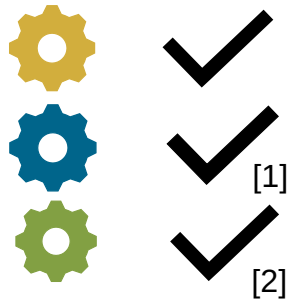
* run on simulator



goal: port this operation on the quantum computer

main steps:

1) encode vector
2) apply operators
3) compute the new time-step

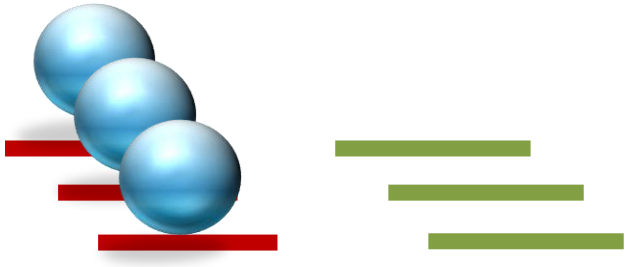

[1]
[2]


[3]

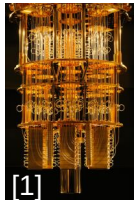
[1] Termanova et al., Quantum Tensor Programming, **New J. Phs. 26, 123019,** 2024
[2] M. Lubasch et al., *Variational quantum algorithms for nonlinear problems*, **Phys. Rev. A 101**, 2020
[3] P. Siegl et al., *Tensor-Programmable Quantum Circuits for differential equations,* https://arxiv.org/abs/2502.04425, 2025

Introduction: Bits and Qubits

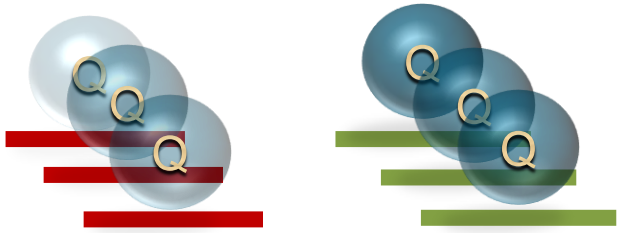
classical bit



000



quantum bit (qubit)



$$\sqrt{0.2}\,|0\rangle + \sqrt{0.8}\,|1\rangle$$

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} \begin{matrix} |000\rangle \\ |001\rangle \\ |010\rangle \\ |011\rangle \\ |100\rangle \\ |101\rangle \\ |110\rangle \\ |111\rangle \end{matrix}$$

$$a_i \in \mathbb{C}, \sum_{i=0}^{N} {a_i}^2 = 1$$

normalized vector

Encoding Field into a Quantum Computer

recap: $f^{j+1} = f^j + \Delta t \cdot c_d \Delta f^j$

$$\rightarrow \begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^{j+1} = \begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^{j} + dt \begin{pmatrix} -2 & 1 & \cdots & 0 & 0 \\ 1 & -2 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & & -2 & 1 \\ 0 & 0 & \cdots & 1 & -2 \end{pmatrix} \begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^{j}$$

classically

field: unnormalized vector $\begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}$

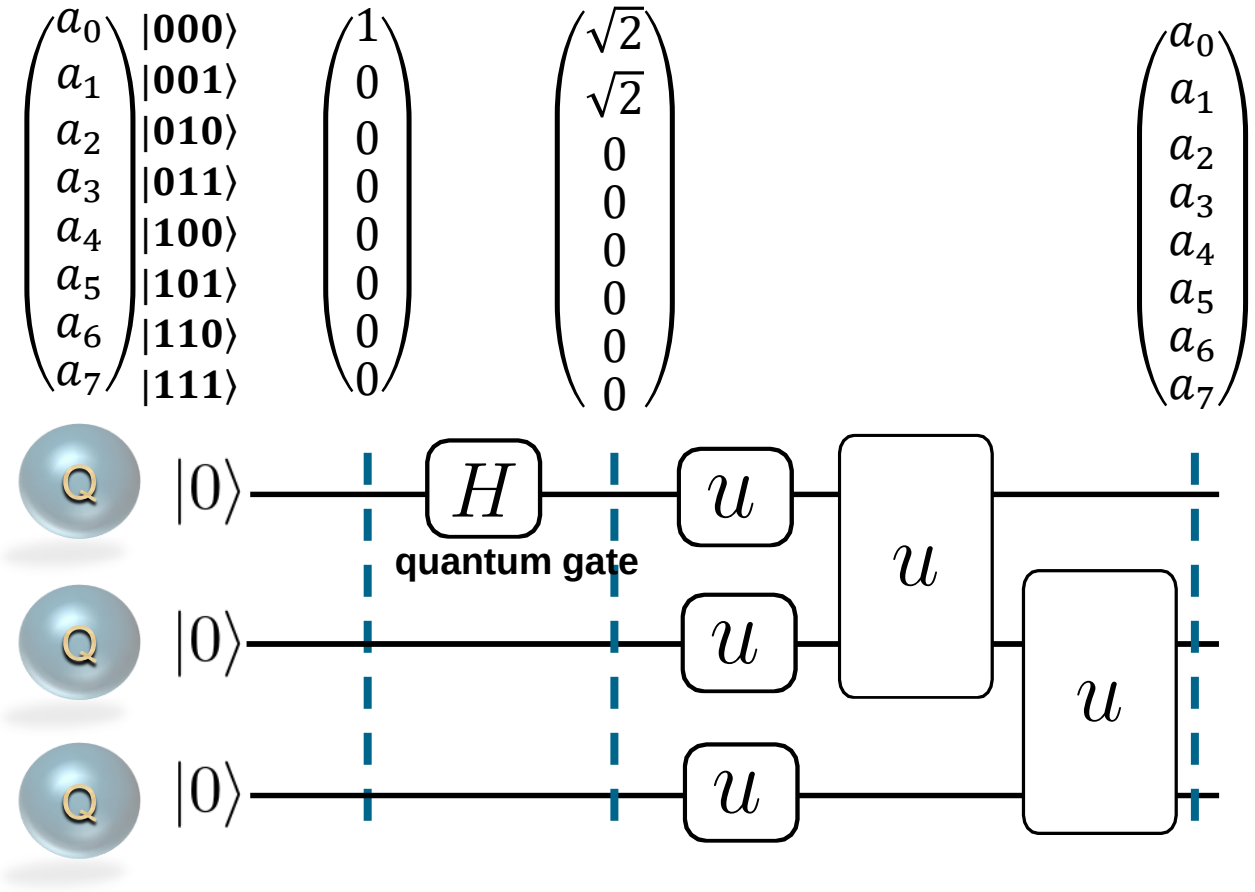
quantum computer

state: normalized vector $\begin{pmatrix} a_0 \\ \vdots \\ a_N \end{pmatrix}$

connecting both worlds

field: $\begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix} = \theta_0 \begin{pmatrix} a_0 \\ \vdots \\ a_N \end{pmatrix}$

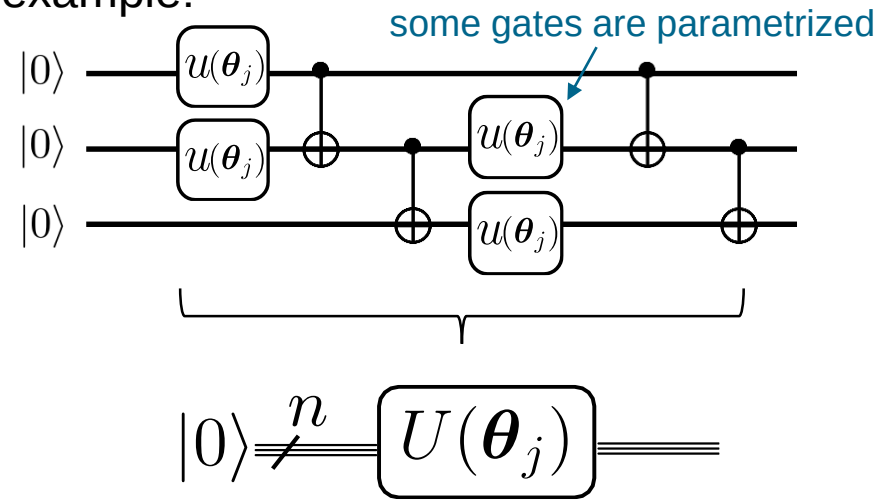
Encoding Field into a Quantum Computer



quantum circuit

here:
encode field with classically parametrized ansatz

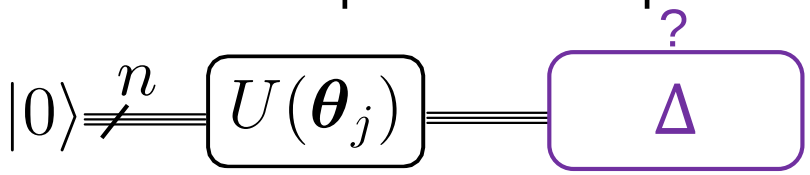
example:



classical parameters: θ_j

Porting Operators on Quantum Computers

goal: matrix vector multiplication on quantum computer



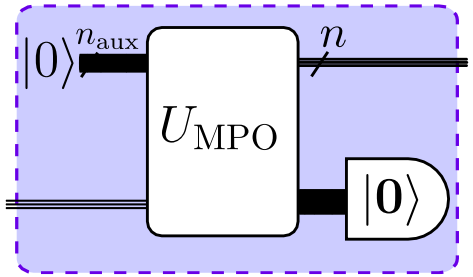
but: gates are always unitary
(**norm-conserving + reversible**)

classically
operator: non-unitary

quantum computer
gates: always unitary

connecting both worlds

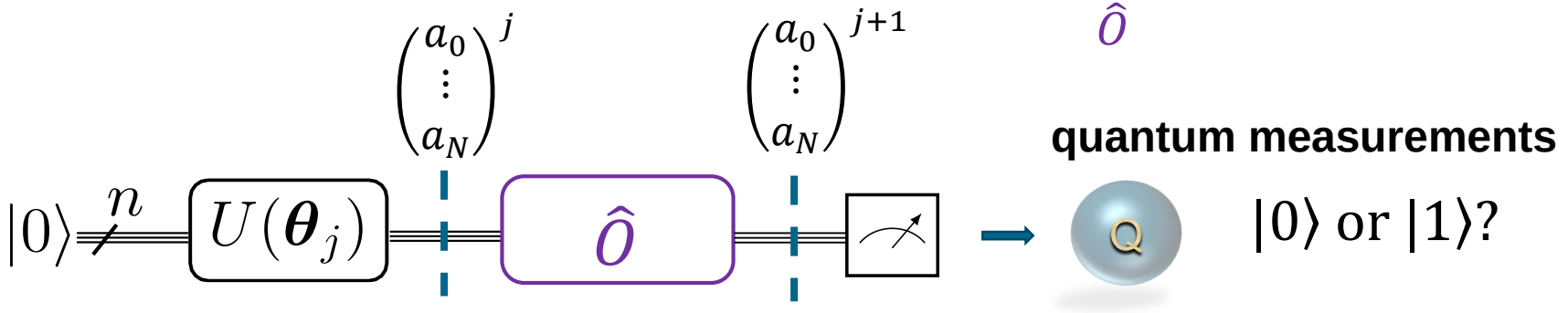
probabilistic application of Δ:
- success probability α_{succ}
- using extra qubits + measurements
- tensor networks



[1] Termanova et al., **New J. Phs. 26, 123019,** 2024

Computing the New Time Step

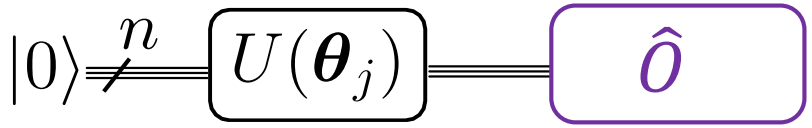
recap: explicit Euler time stepping $f^{j+1} = f^j + \Delta t \cdot c_d \, \Delta f^j = \underbrace{(I_n + c\,\Delta)}_{\hat{O}} f^j$ $\qquad c = \Delta t \cdot c_d$



state read out is expensive!

[2] we do:

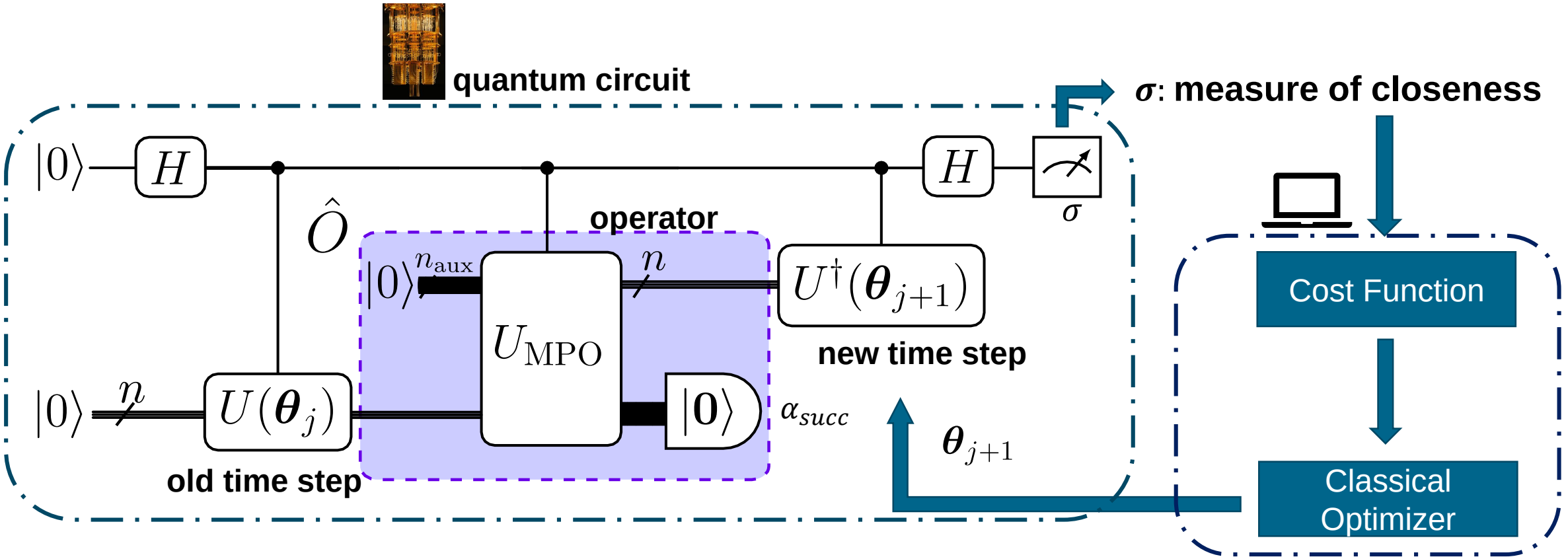
create a **variational ansatz** $U^\dagger(\boldsymbol{\theta}_{j+1})$

compare with 

[2] M. Lubasch et al., *Variational quantum algorithms for nonlinear problems*, **Phys. Rev. A 101**, 2020

Full Quantum Circuit

combining tensor network based operator encoding with variational time stepping



[3] P. Siegl et al. (2025), https://arxiv.org/abs/2502.04425

Necessary Correction

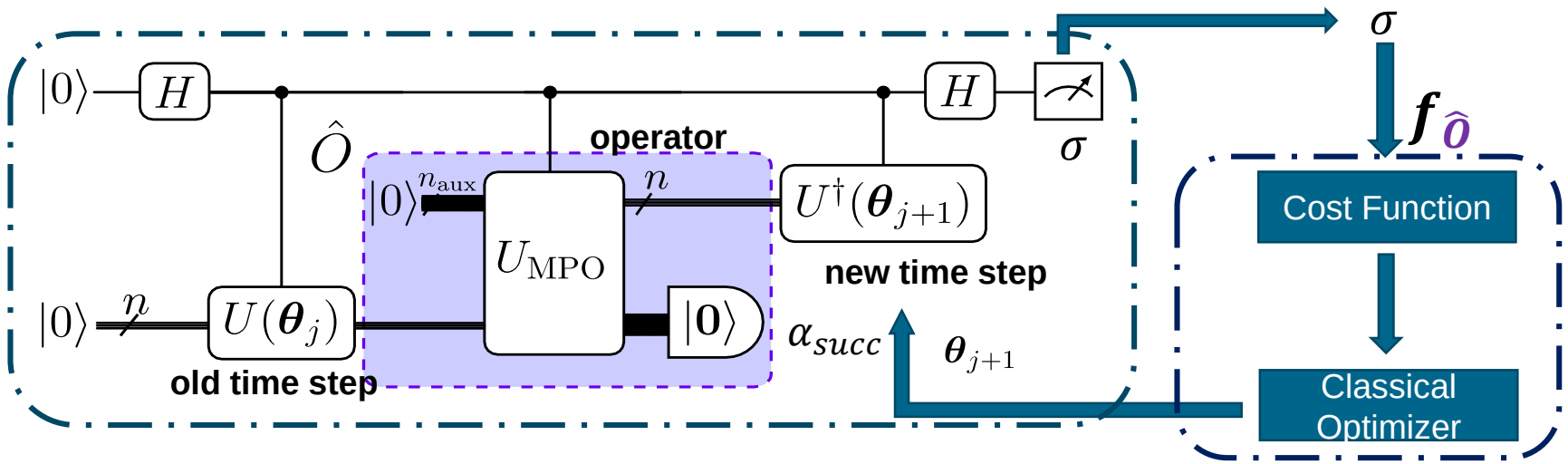
problem: solution has wrong norm field: $\begin{pmatrix} f_0 \\ \vdots \\ f_N \end{pmatrix}^j = \theta_0 \begin{pmatrix} a_0 \\ \vdots \\ a_N \end{pmatrix}^j \rightarrow \theta_0 = \theta_0^j$

reason:

$\hat{O}_{classic} \begin{pmatrix} n_0 \\ \dots \\ n_N \end{pmatrix} = \begin{pmatrix} \dots \\ \dots \\ \dots \end{pmatrix}$, $\hat{O}_{quantum} \begin{pmatrix} n_0 \\ \dots \\ n_N \end{pmatrix} = \begin{pmatrix} \dots \\ \dots \\ \dots \end{pmatrix}$,

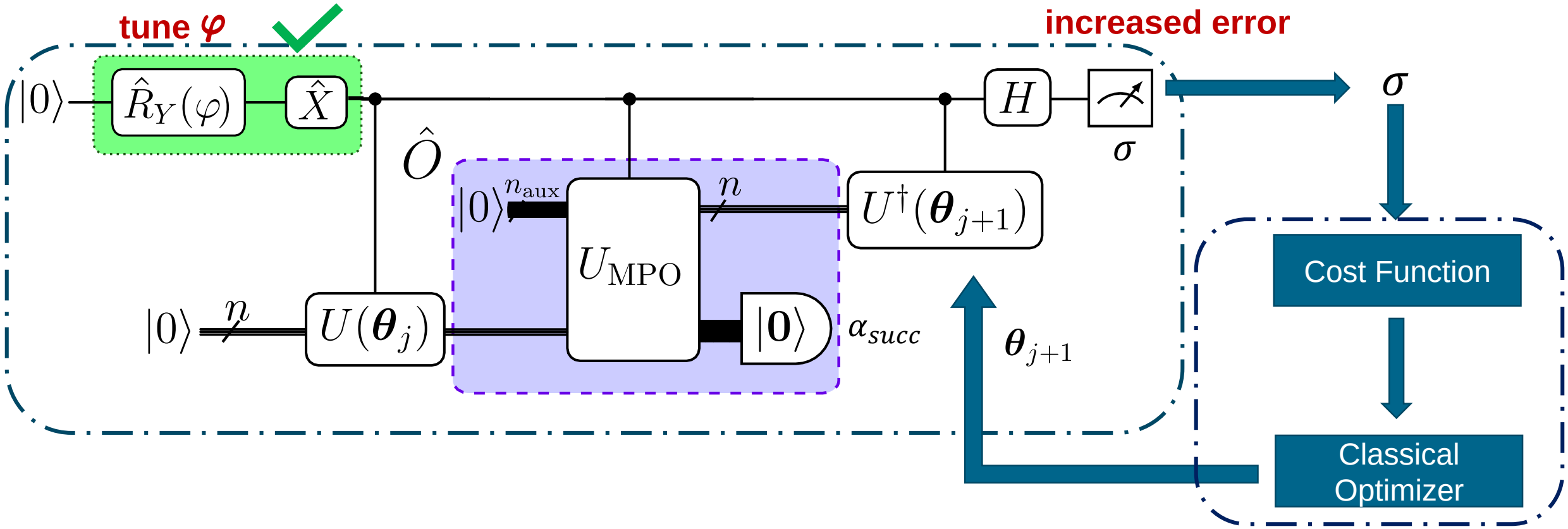
$\begin{pmatrix} \dots \\ \dots \\ \dots \end{pmatrix} = \boldsymbol{f}_{\hat{O}} \begin{pmatrix} \dots \\ \dots \\ \dots \end{pmatrix}$,

can be computed **from $\boldsymbol{\alpha_{succ}}$**



Beneficial Correction

problem: extra measurements $|0\rangle$ increase measurement error of 



[3] P. Siegl et al. (2025), https://arxiv.org/abs/2502.04425

Example: Euler Equation

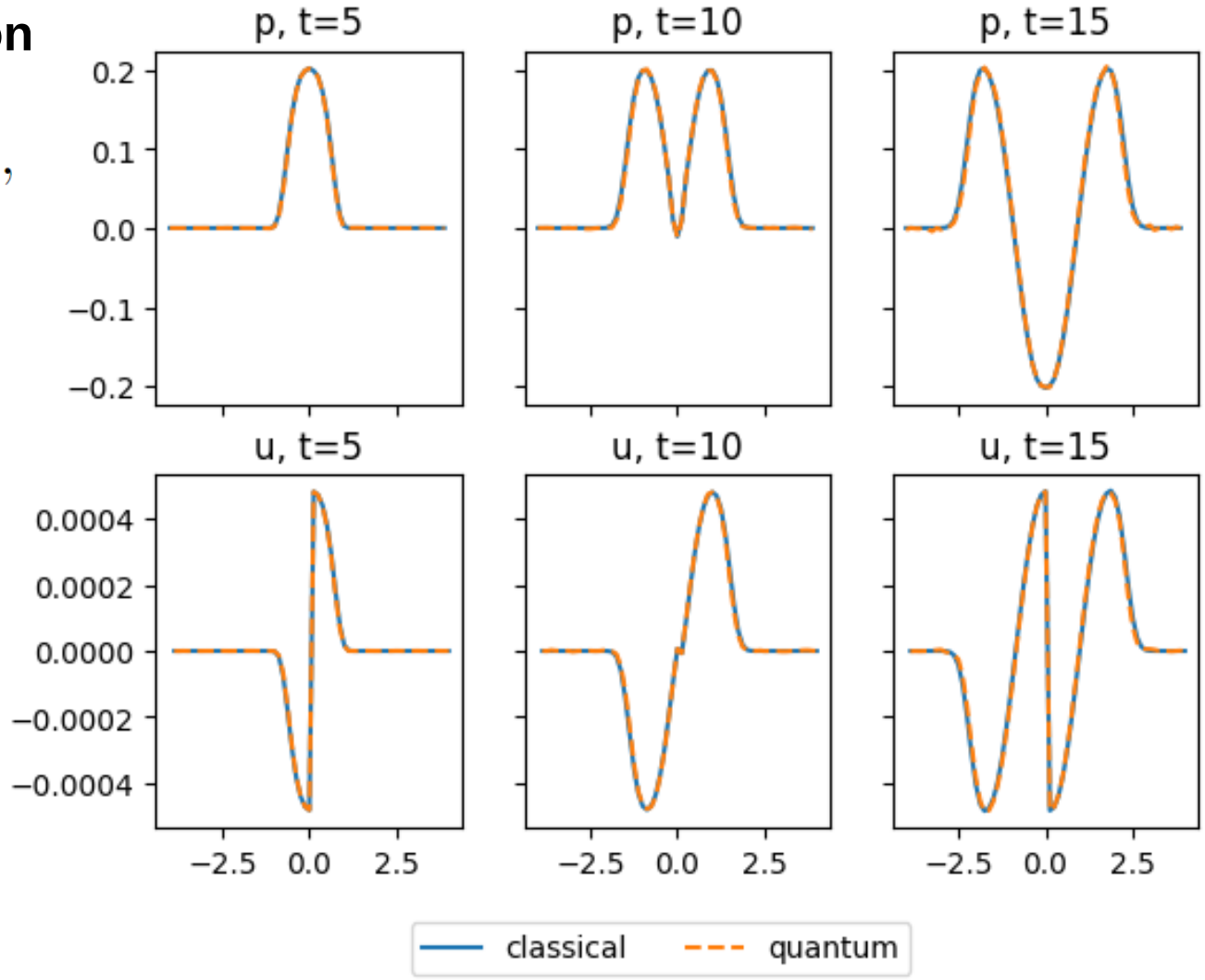
1D linear incompressible Euler equation

$$\frac{\partial p}{\partial t} = -\bar{\rho}c^2\left(\frac{\partial u}{\partial x}\right) + f(x,t) - \gamma(x)p,$$
$$\frac{\partial u}{\partial t} = -\frac{1}{\bar{\rho}}\frac{\partial p}{\partial x} - \gamma(x)u,$$

source

sponge

- 4th order Runge Kutta
- run on quantum simulator



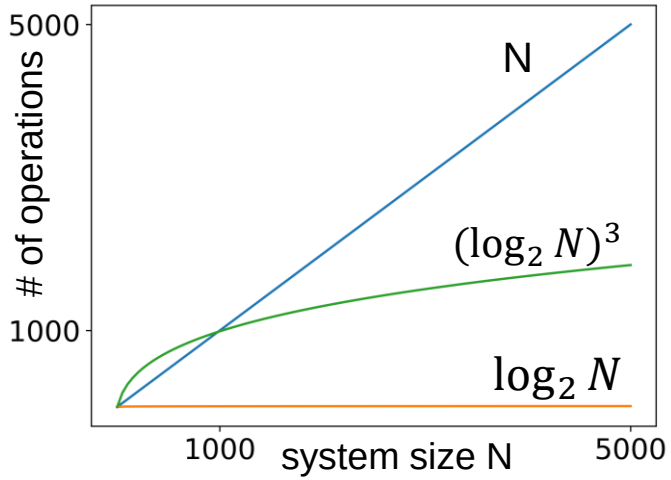
[3] P. Siegl et al. (2025), https://arxiv.org/abs/2502.04425

Scaling with System Size

! advantage expected for **large scale simulations**

classically

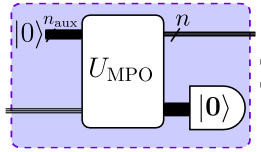
matrix – vector multiplication: N^2
sparse matrix-vector multiplication: $>N$



quantum computer → expected:

$U(\boldsymbol{\theta}_j)$: poly($\log_2 N$)

→ dependence on amount of randomness expected



: $\log_2 N$

→ tensor network algorithms can help to estimate scaling

? how does the training scale with $\#\theta_j$?
→ expected: $\#\theta_j$ increases poly($\log_2 N$)
→ trainability can be assured for larger circuit sizes

Advantages of Approach

- very easy incorporation of various operators with little additional cost:
  - ➢ different boundary conditions, higher order accuracies
  - ➢ flexible incorporation of non-derivative operators.

- possible reduction of computational cost

🎓 big advantage identified: in-the loop quantification of quality of solution.
  - ➢ **currently working on quantifying this and increasing the accuracy of this.**

Summary

- why should we solve partial differential equations on quantum computers?
  - **promising scaling: advantages for large scale simulations?**
- can we solve classical partial differential equations on quantum computers?
  - **implementation of broad range of differential equations possible**
  - **in-the loop quantification of quality of solution possible**

Questions?