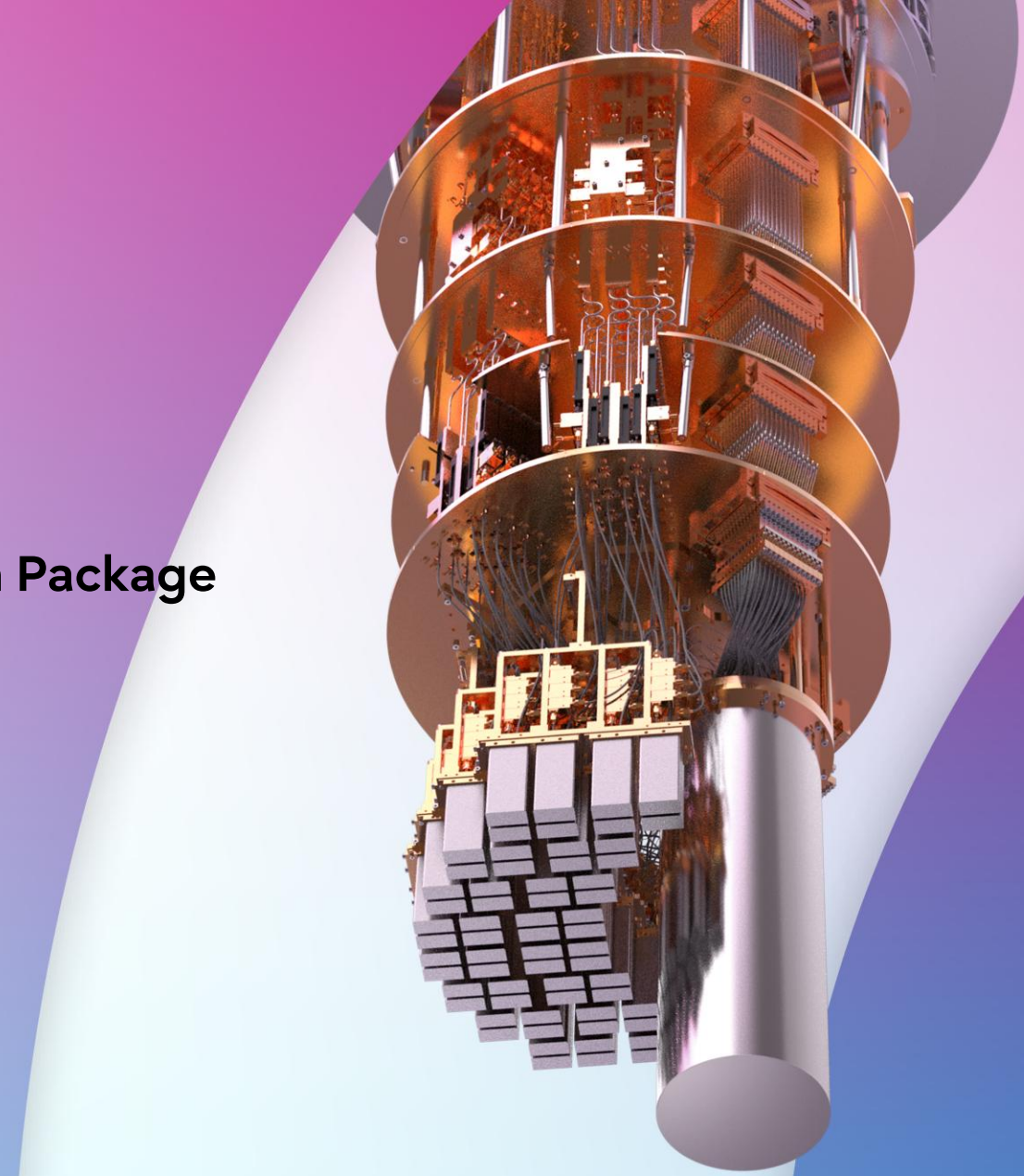
OpenQARP

Open Quantum Application Research Package



November 2026

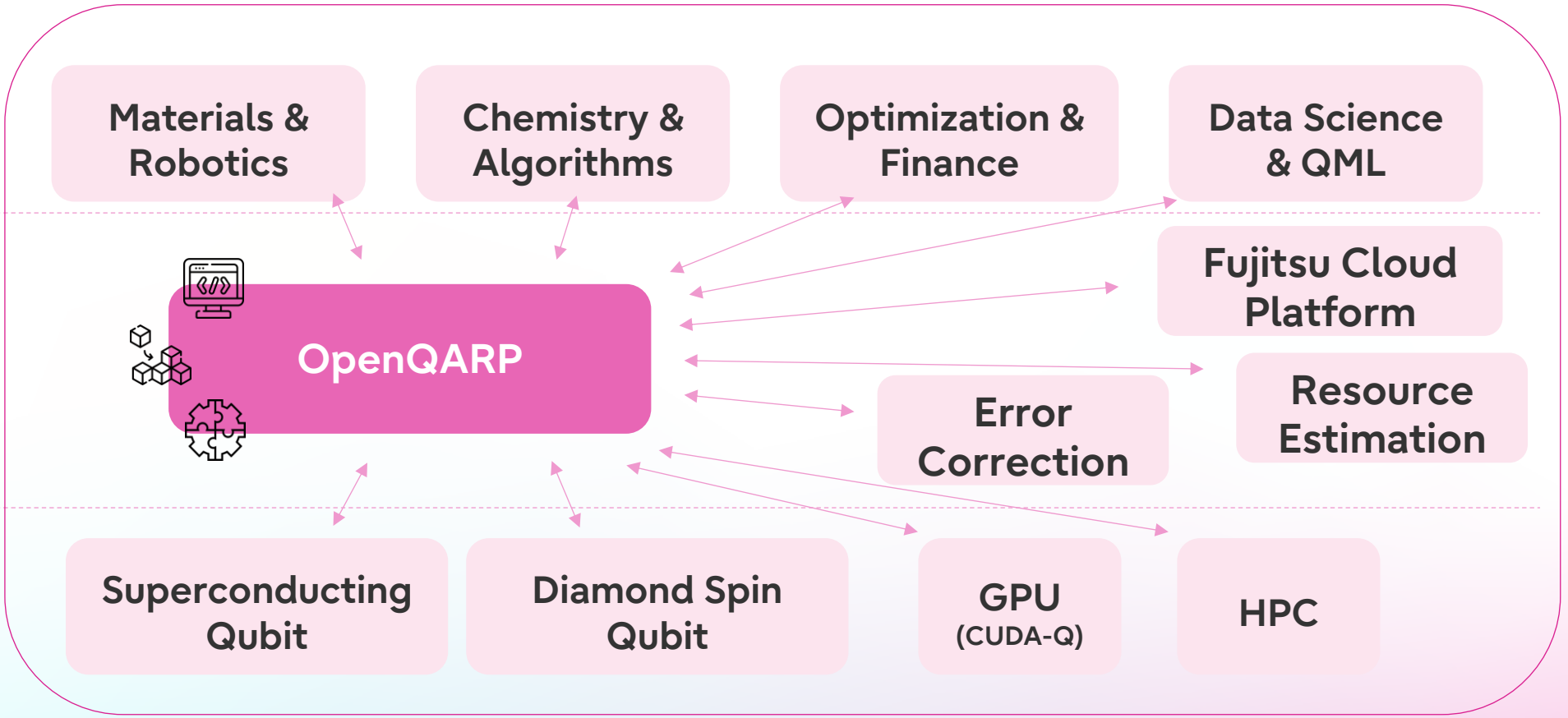
© Fujitsu 2026

Where OpenQARP fits in the Fujitsu stack

FUJITSU

September 15:
Released as OSS

$ pip install openqarp



© Fujitsu 2026

Why OpenQARP: our philosophy

FUJITSU

- **Research-first —** built by quantum researchers, for quantum researchers, to take ideas to implementation fast.
- **Full-stack —** from fermionic Hamiltonians to compiled circuits on real hardware, all in one package.
- **Backend-agnostic —** Qulacs, pytket, Qiskit and CUDA-Q backends. Write an algorithm once and run it anywhere.
- **Rapid prototyping —** composable circuit blocks, symbolic parameters and analytic gradients. Build, test, iterate.
- **Production-ready —** CI/CD, a comprehensive test suite, full documentation and versioned releases.

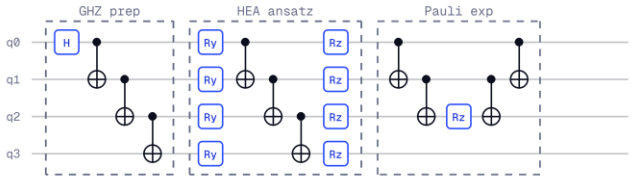
© Fujitsu 2026

© 202 Fujitsu Limited

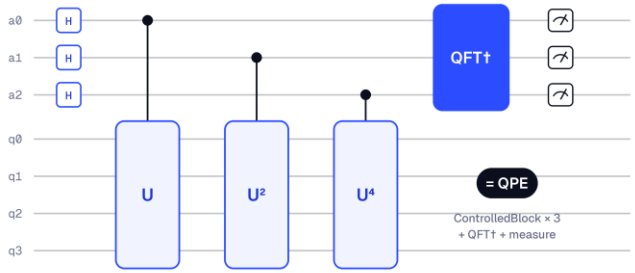
FUJITSU

A new research idea should be a new block, not a new codebase.

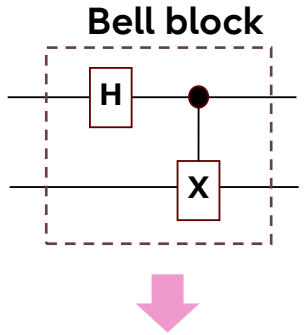
Blocks are operations, not gate lists: an ansatz layer, a Trotter step, a state preparation. They nest, they take controls, and yours behave exactly like ours.



Gates, grouped into reusable blocks



Controlled blocks → phase estimation




```
bell = SimpleBlock(2, name="bell")
bell.h(0); bell.cx(0, 1); bell.build()
exact = StateVector(ket=bell, operator=H)
shots = Sampler(ket=bell, n_shots=4000)
engine = QarpEngine(seed=42)
engine.build([exact, shots])
results = engine.run()
```

First-class

A block you write gets the same control, adjoint, compile and export as ours.

Compile once

θ stays symbolic in the C++ core: build U(θ) once, rebind it at every optimizer step.

Phase-exact

Blocks keep their exact global phase, so controlling one never shifts eigenvalues.

30+ algorithms, one stack

Ground state	VQE · ADAPT-VQE · QITE
Excited states	VQD · SSVQE · QSE
Spectra	QPE · DOS-QPE · QMEGS · MMQCELS
Optimization	QAOA · PCE and variants
Search & sampling	Grover · Shor · QC-QMC · VFF

© Fujitsu 2026

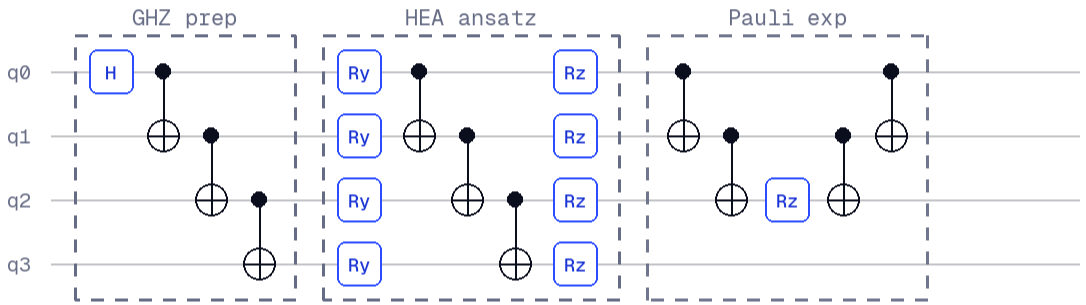
OpenQARP

Quantum applications, built in layers.

open source · Apache 2.0 · Fujitsu Research of Europe

```
$ pip install openqarp
```

1 Blocks what the circuit does



Blocks nest: prep + ansatz + Pauli exponential → one CompositeBlock U(θ)

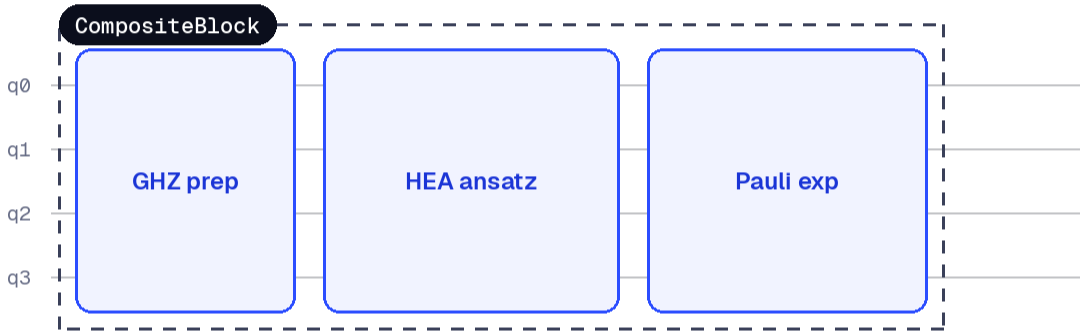
OpenQARP

Quantum applications, built in layers.

open source · Apache 2.0 · Fujitsu Research of Europe

```
$ pip install openqarp
```

1 Blocks what the circuit does



Blocks nest: prep + ansatz + Pauli exponential → one CompositeBlock U(θ)

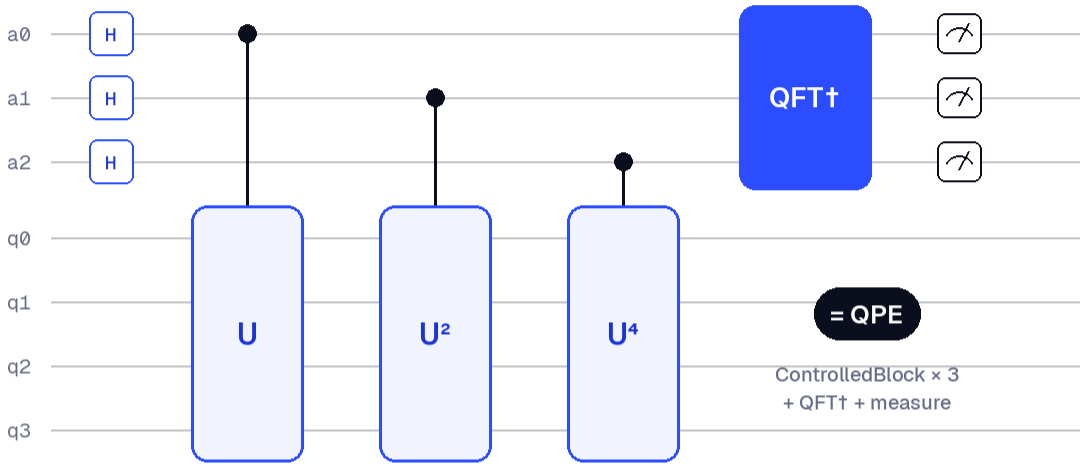
OpenQARP

Quantum applications, built in layers.

open source · Apache 2.0 · Fujitsu Research of Europe

```
$ pip install openqarp
```

1 Blocks what the circuit does



Control any block → a phase-estimation ladder, correct by construction

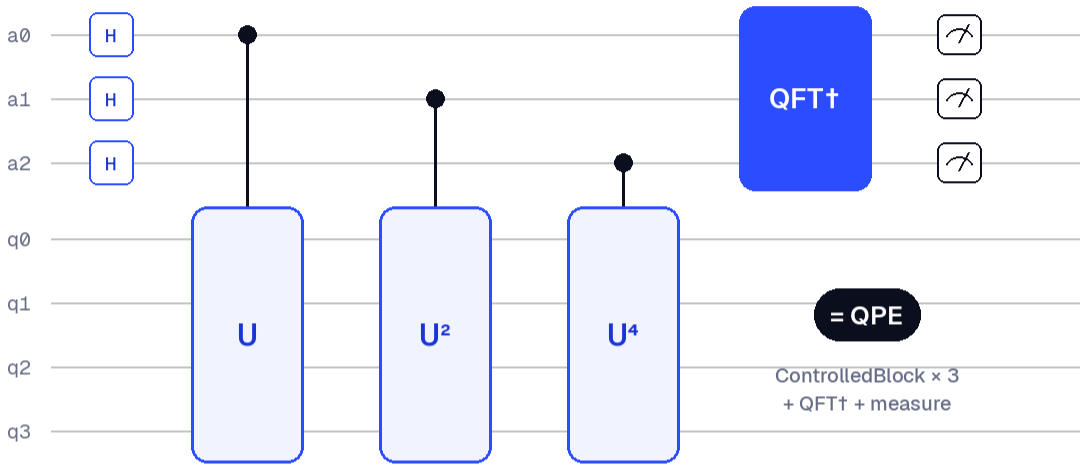
OpenQARP

Quantum applications, built in layers.

open source · Apache 2.0 · Fujitsu Research of Europe

```
$ pip install openqarp
```

1 Blocks what the circuit does



70+ blocks · symbolic parameters · exact phases · compile once, reuse

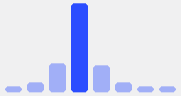
2 Primitives what to extract

StateVector `-1.136761`

exact amplitudes · ⟨ψ|O|ψ⟩

Sampler

n_shots = 4000 → distribution



PauliAveraging `≈ -1.1368`

⟨H⟩ from shots · shadows

Overlaps `⟨φ|O|ψ⟩`

Hadamard · SWAP · mirror test

Name the quantity, not the procedure

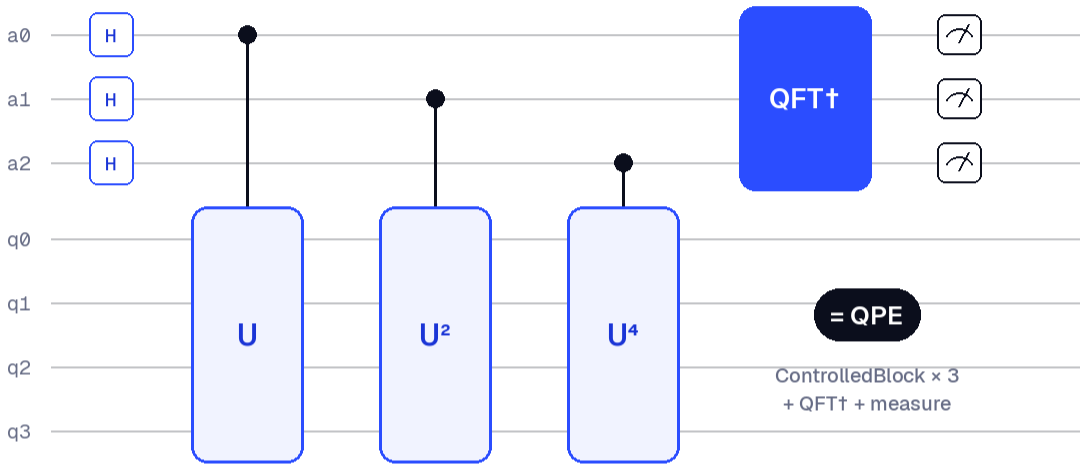
OpenQARP

Quantum applications, built in layers.

open source · Apache 2.0 · Fujitsu Research of Europe

```
$ pip install openqarp
```

1 Blocks what the circuit does



70+ blocks · symbolic parameters · exact phases · compile once, reuse

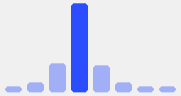
2 Primitives what to extract

StateVector `-1.136761`

exact amplitudes · ⟨ψ|O|ψ⟩

Sampler

n_shots = 4000 → distribution



PauliAveraging `≈ -1.1368`

⟨H⟩ from shots · shadows

Overlaps `⟨φ|O|ψ⟩`

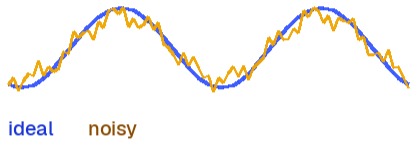
Hadamard · SWAP · mirror test

Name the quantity, not the procedure

3 Engines how it runs

QarpEngine + NoiseModel

depolarizing · Pauli · bit-flip · amp. damping




```
engine = QarpEngine(noise_model=noise)
engine.build(primitives); engine.run()
```

Swap the engine, keep the program

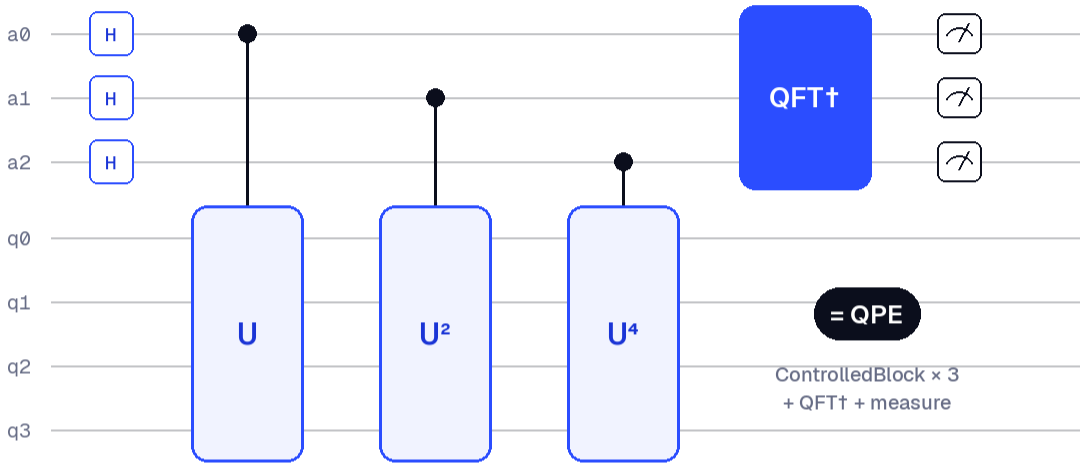
OpenQARP

Quantum applications, built in layers.

open source · Apache 2.0 · Fujitsu Research of Europe

```
$ pip install openqarp
```

1 Blocks what the circuit does



70+ blocks · symbolic parameters · exact phases · compile once, reuse

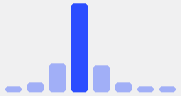
2 Primitives what to extract

StateVector `-1.136761`

exact amplitudes · ⟨ψ|O|ψ⟩

Sampler

n_shots = 4000 → distribution



PauliAveraging `≈ -1.1368`

⟨H⟩ from shots · shadows

Overlaps `⟨φ|O|ψ⟩`

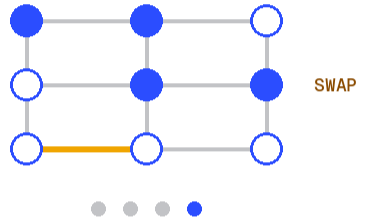
Hadamard · SWAP · mirror test

Name the quantity, not the procedure

3 Engines how it runs

Device-aware compile

gate set · coupling map · SABRE routing




```
# + device: gate set, coupling, noise
engine.build(primitives); engine.run()
```

Swap the engine, keep the program

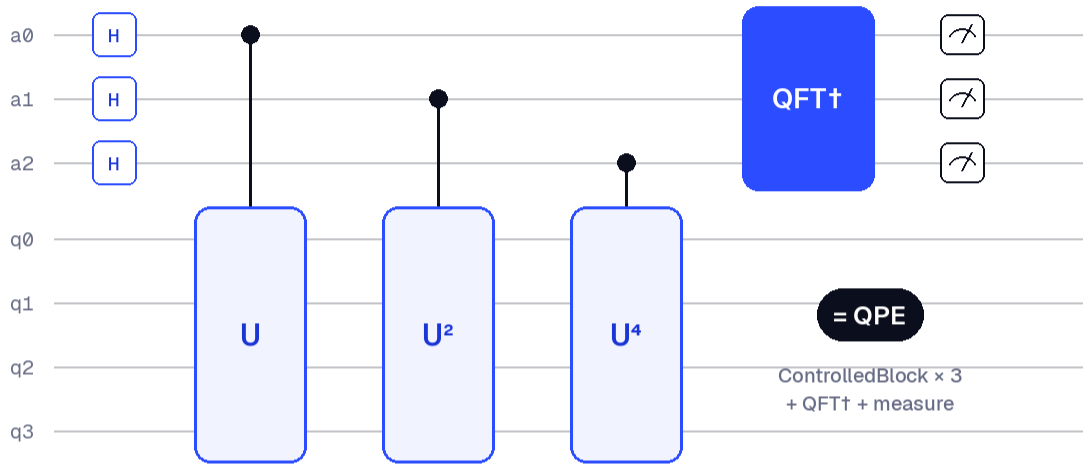
OpenQARP

Quantum applications, built in layers.

open source · Apache 2.0 · Fujitsu Research of Europe

```
$ pip install openqarp
```

1 Blocks what the circuit does



70+ blocks · symbolic parameters · exact phases · compile once, reuse

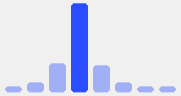
2 Primitives what to extract

StateVector `-1.136761`

exact amplitudes · ⟨ψ|O|ψ⟩

Sampler

n_shots = 4000 → distribution



PauliAveraging `≈ -1.1368`

⟨H⟩ from shots · shadows

Overlaps `⟨φ|O|ψ⟩`

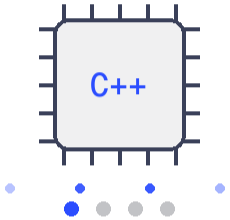
Hadamard · SWAP · mirror test

Name the quantity, not the procedure

3 Engines how it runs

QarpEngine

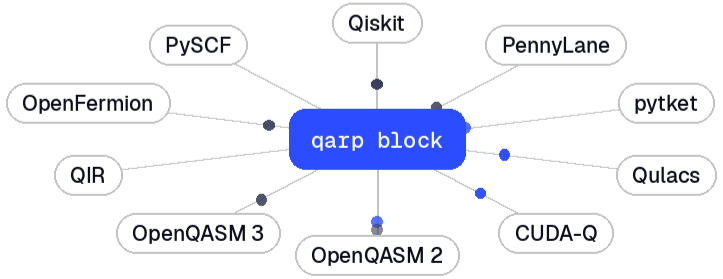
exact state vector · C++ core · gate fusion




```
engine = QarpEngine(seed=42)
engine.build(primitives); engine.run()
```

Swap the engine, keep the program

4 Interoperability emit / absorb



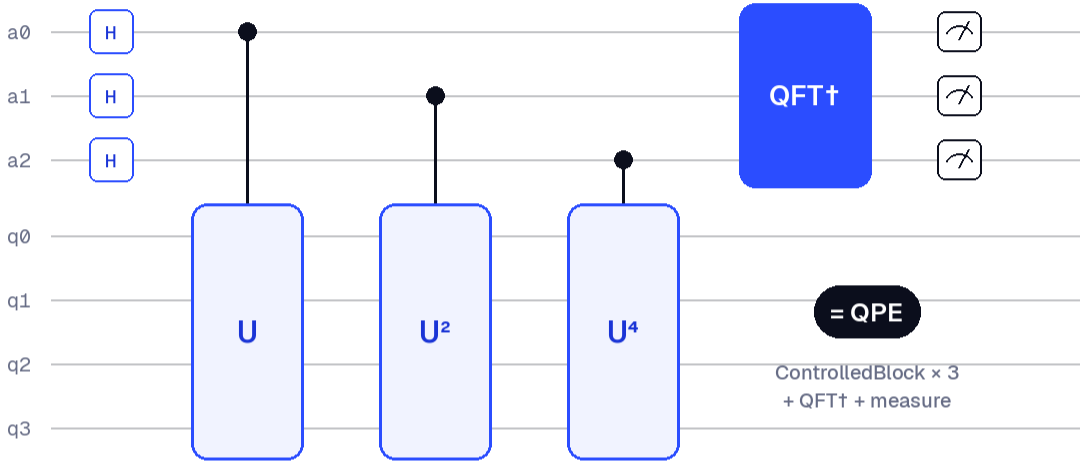
OpenQARP

Quantum applications, built in layers.

open source · Apache 2.0 · Fujitsu Research of Europe

```
$ pip install openqarp
```

1 Blocks what the circuit does

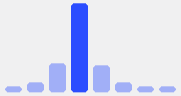


70+ blocks · symbolic parameters · exact phases · compile once, reuse

2 Primitives what to extract

StateVector `-1.136761`
exact amplitudes · ⟨ψ|O|ψ⟩

Sampler
n_shots = 4000 → distribution



PauliAveraging `≈ -1.1368`
⟨H⟩ from shots · shadows

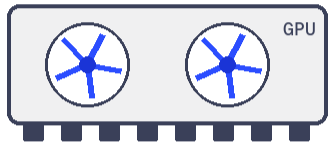
Overlaps `⟨φ|O|ψ⟩`
Hadamard · SWAP · mirror test

Name the quantity, not the procedure

3 Engines how it runs

CudaqEngine

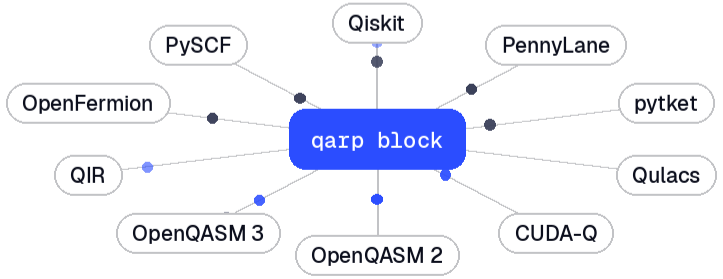
GPU state vector · optional extra



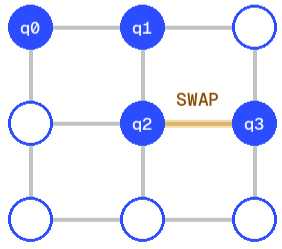

```
engine = CudaqEngine()
engine.build(primitives); engine.run()
```

Swap the engine, keep the program

4 Interoperability emit / absorb



5 Devices & compilation devices are data



coupling map · SABRE routing

noise models · circuit cutting

resource estimation

logical

routed

Clifford+T

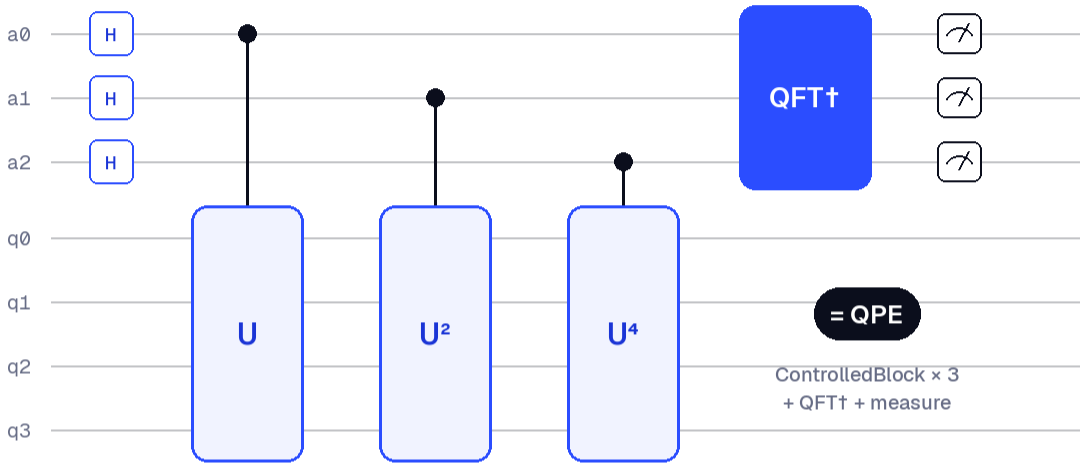
OpenQARP

Quantum applications, built in layers.

open source · Apache 2.0 · Fujitsu Research of Europe

```
$ pip install openqarp
```

1 Blocks — what the circuit does

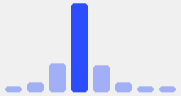


70+ blocks · symbolic parameters · exact phases · compile once, reuse

2 Primitives — what to extract

StateVector `-1.136761`
exact amplitudes · ⟨ψ|O|ψ⟩

Sampler
n_shots = 4000 → distribution



PauliAveraging `≈ -1.1368`
⟨H⟩ from shots · shadows

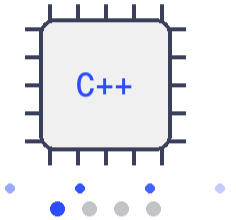
Overlaps `⟨φ|O|ψ⟩`
Hadamard · SWAP · mirror test

Name the quantity, not the procedure

3 Engines — how it runs

QarpEngine

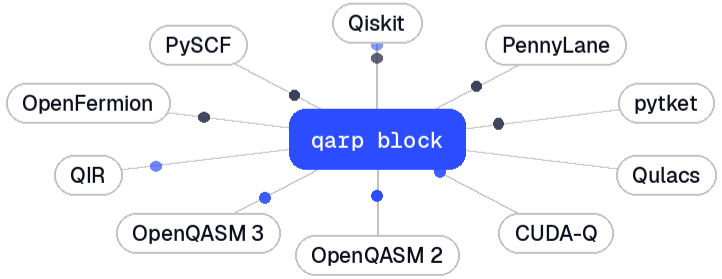
exact state vector · C++ core · gate fusion



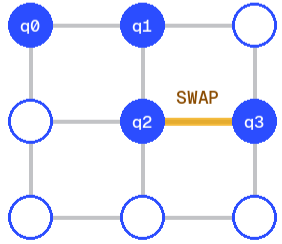

```
engine = QarpEngine(seed=42)
engine.build(primitives); engine.run()
```

Swap the engine, keep the program

4 Interoperability — emit / absorb



5 Devices & compilation — devices are data



coupling map · SABRE routing

noise models · circuit cutting

resource estimation

logical

routed

Clifford+T

6 Composite algorithms — primitives + optimizer

`primitive` —E(θ)→ `optimizer` COBYLA · SPSA · Adam —θ→

VQE · ADAPT-VQE · VQD · SSVQE · QSE · QITE · QPE
DOS-QPE · QMEGS · MMQCELS · QAOA · PCE · Grover
Shor · Amp. estimation · QC-QMC · VFF · Projected VQE

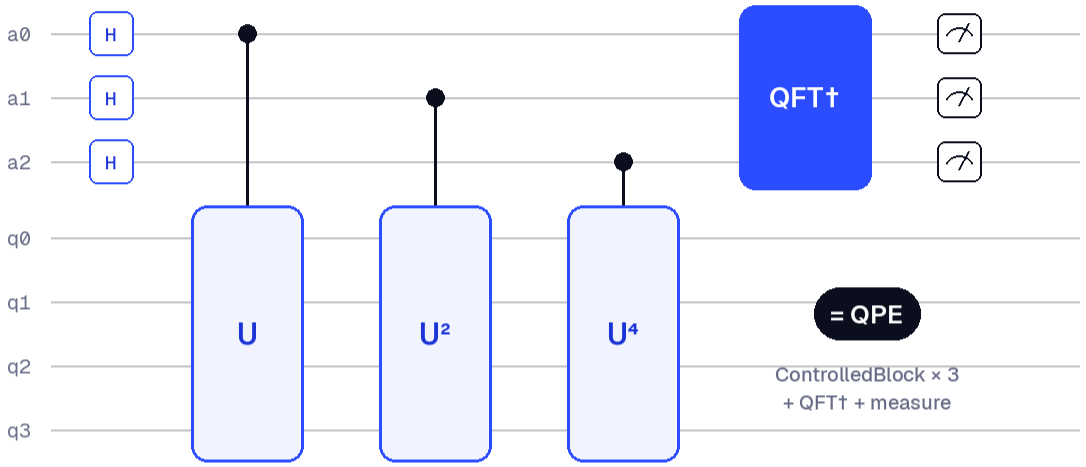
OpenQARP

Quantum applications, built in layers.

open source · Apache 2.0 · Fujitsu Research of Europe

```
$ pip install openqarp
```

1 Blocks — what the circuit does

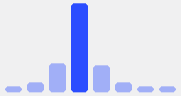


70+ blocks · symbolic parameters · exact phases · compile once, reuse

2 Primitives — what to extract

StateVector `-1.136761`
exact amplitudes · ⟨ψ|O|ψ⟩

Sampler
n_shots = 4000 → distribution



PauliAveraging `≈ -1.1368`
⟨H⟩ from shots · shadows

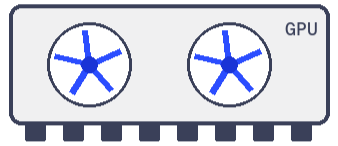
Overlaps `⟨φ|O|ψ⟩`
Hadamard · SWAP · mirror test

Name the quantity, not the procedure

3 Engines — how it runs

CudaqEngine

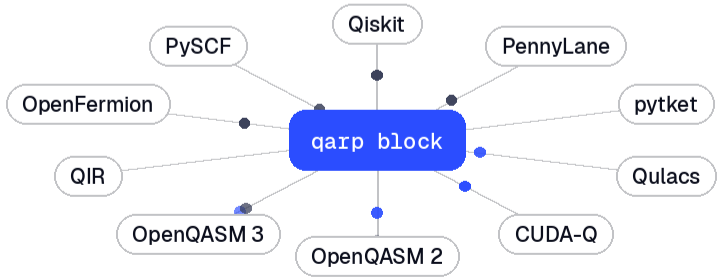
GPU state vector · optional extra



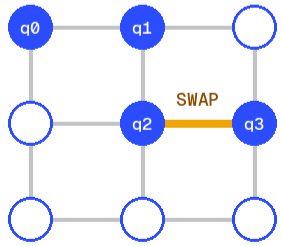

```
engine = CudaqEngine()
engine.build(primitives); engine.run()
```

Swap the engine, keep the program

4 Interoperability — emit / absorb



5 Devices & compilation — devices are data



coupling map · SABRE routing

noise models · circuit cutting

resource estimation

logical

routed

Clifford+T

6 Composite algorithms — primitives + optimizer

`primitive` ⇄ E(θ) / θ ⇄ `optimizer` COBYLA · SPSA · Adam

VQE · ADAPT-VQE · VQD · SSVQE · QSE · QITE · QPE
DOS-QPE · QMEGS · MMQCELS · QAOA · PCE · Grover
Shor · Amp. estimation · QC-QMC · VFF · Projected VQE

70+ blocks **30+** algorithms & primitives **10+** frameworks & formats **60+** runnable notebooks **5,000+** tests vs independent oracles

openqarp.com

Quantum software blocks

FUJITSU

State Preparation

- ComputationalBasisState Block
- CsfState Block
- CVQRAMState Block
- CVOQRAMState Block
- DickeState Block
- GHZLikeState Block
- HypergraphState Block
- LowRankState Block
- MappedONVState Block
- MpsState Block
- MultiOnvState Block
- PiecewiseLinearState Block
- SlaterDeterminant Block
- SparseState Block
- UniformSuperposition Block
- SynthesizedState Block

Basic

- Any Block
- Composite Block
- Conditional Block
- Controlled Block
- Measure Block
- Reset Block
- Simple Block

Primitives

- Agate Block
- AmplitudeAmplification Block
- AmplitudeEstimation Block
- BlockEncoding Block
- BrickworkEntangling Block
- BrickworkPCE Block
- CostOperator Block
- DOSQPE Block
- Xn Block
- Givens Block
- Grover Block
- HaarRandom Block
- HadamardTest Block
- HEA Block
- Hn Block
- Identity Block
- InterferometricState Block
- Layer Block
- LinearEntangling Block
- MixedOperator Block
- ModularMultiplication Block
- OrbitalRotation Block
- OrderFinding Block
- Pauli Block
- PhaseShift Block
- ProjectControlPhase Block
- Projector Block
- QAOA Block
- QFT Block
- QPE Block
- QROM Block
- QSP Block
- QSVT Block
- Qubitization Block
- Readout Block
- Reflection Block
- RSP Block
- SPA Block
- Select Block
- SWAPTest Block
- SynthesizedTimeEvolution Block
- SynthesizedUnitary Block
- Trotter Block
- UCC Block
- UPCCD Block
- VUMPOBrickwork Block

© Fujitsu 2026

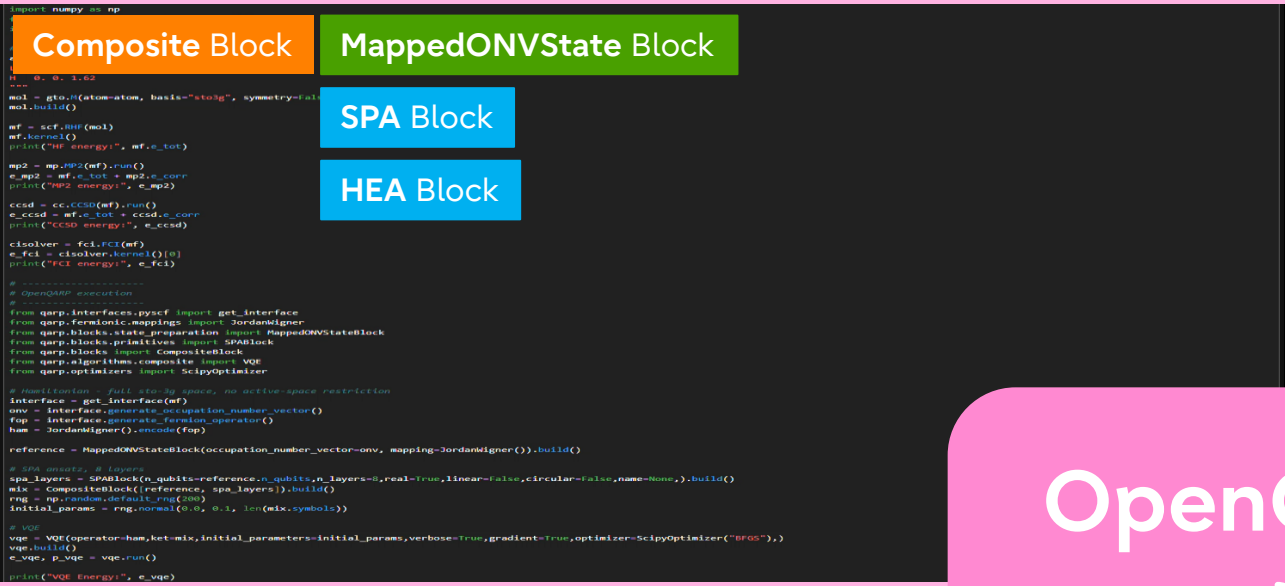
Quantum Chemistry

Composite Block

MappedONVState Block

SPA Block

HEA Block



Finance

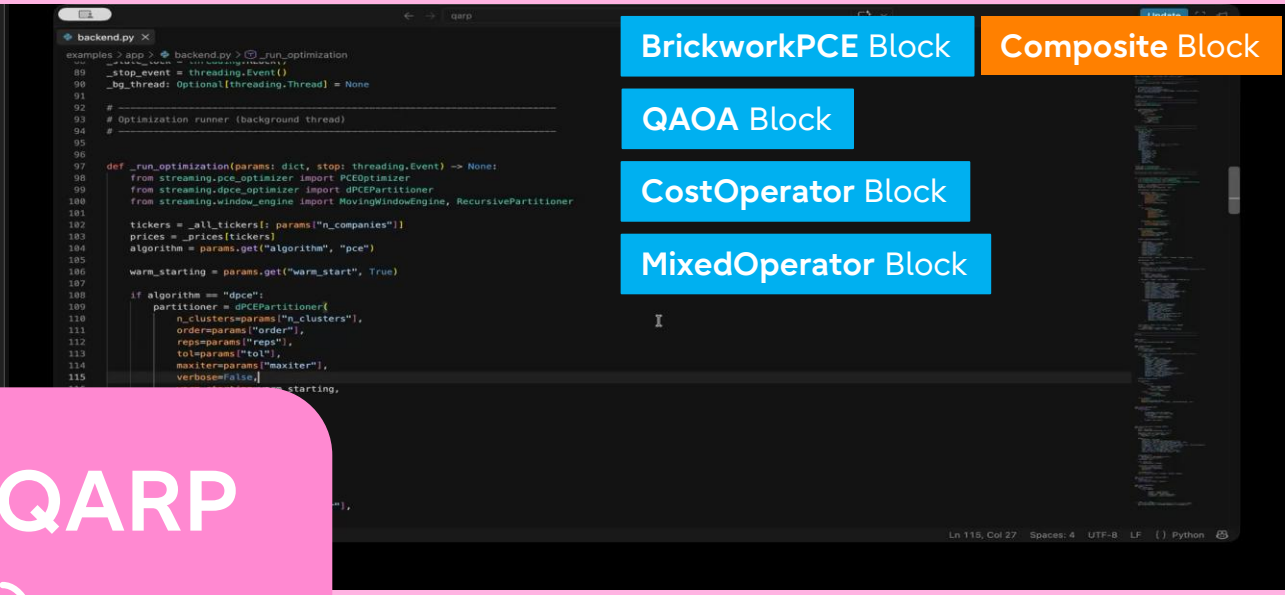
BrickworkPCE Block

Composite Block

QAOA Block

CostOperator Block

MixedOperator Block



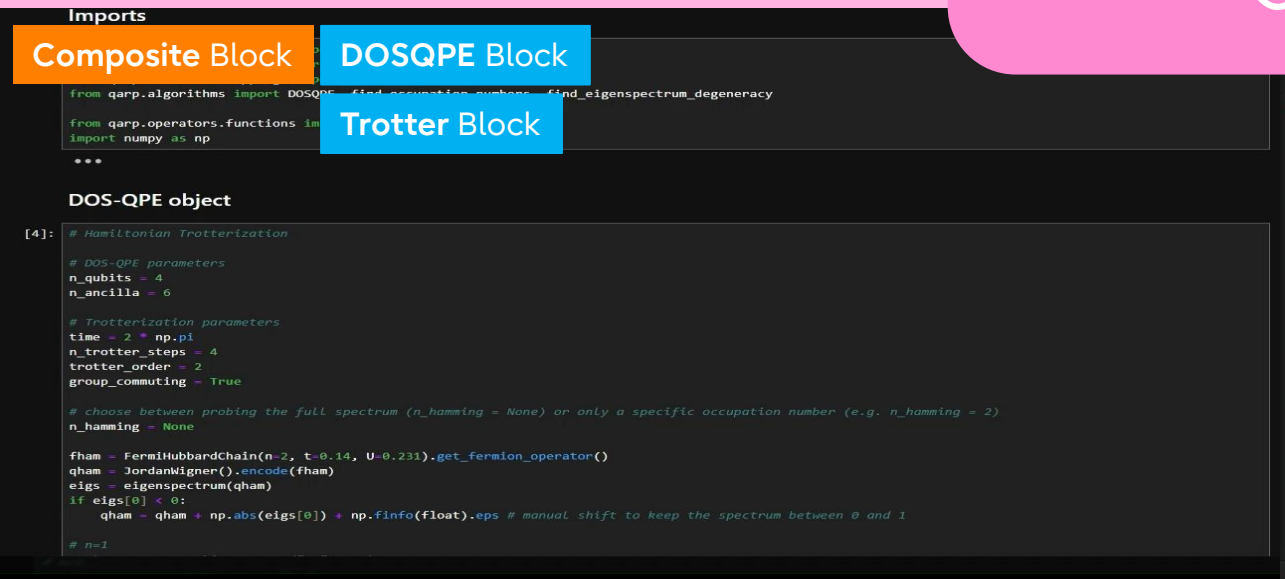
OpenQARP

Spectral Calculations

Composite Block

DOSQPE Block

Trotter Block



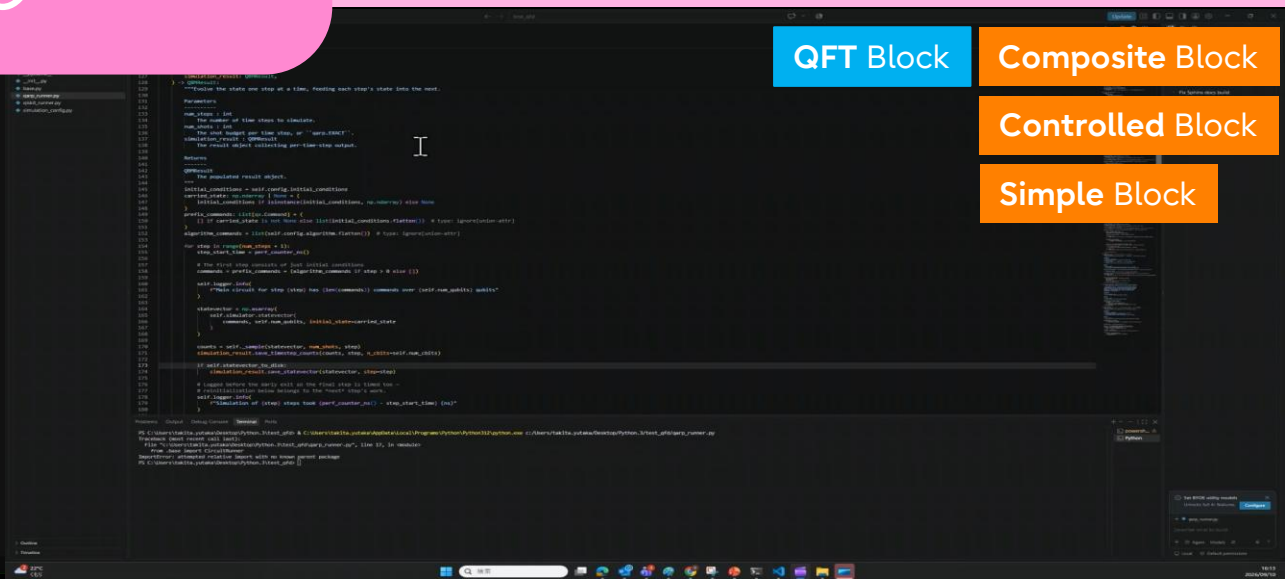
CFD

QFT Block

Composite Block

Controlled Block

Simple Block



OpenQARP is open source

FUJITSU

Bring an algorithm, a workload — or your CV.

Gartner Featured in “First Take”
*Jointly with Diamond Spin and Yaqumo

50+ media features

openqarp@fujitsu.com

$ pip install openqarp

openqarp.com

arXiv paper

GitHub

© Fujitsu 2026

Thank you

© Fujitsu 2026