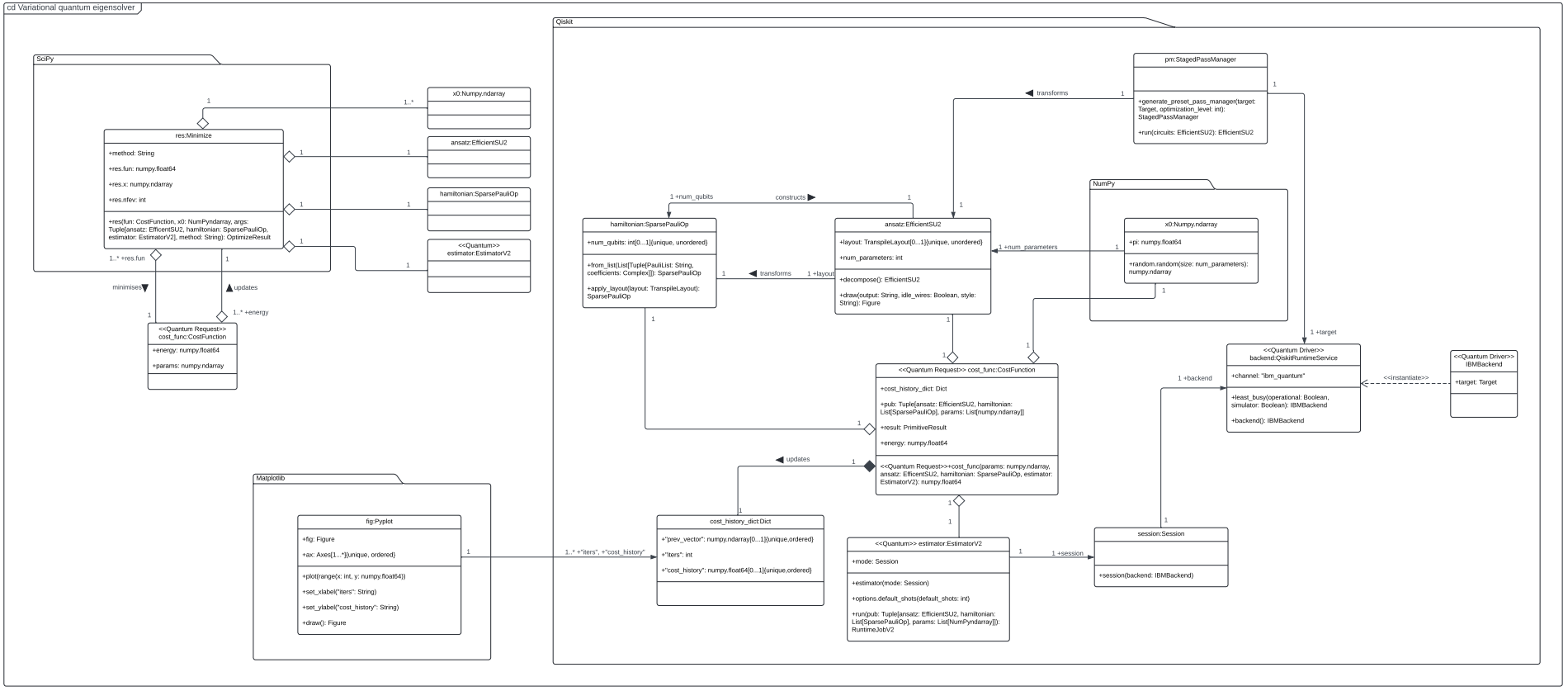
cd Variational quantum eigensolver
SciPy
res:Minimize
+method: String
+res.fun: numpy.float64
+res.x: numpy.ndarray
+res.nfev: int
+res(fun: CostFunction, x0: NumPyndarray, args: Tuple[ansatz: EfficientSU2, hamiltonian: SparsePauliOp, estimator: EstimatorV2], method: String): OptimizeResult
x0:Numpy.ndarray
ansatz:EfficientSU2
hamiltonian:SparsePauliOp
<<Quantum>>
estimator:EstimatorV2
1..* +res.fun
minimises
updates
1..* +energy
<<Quantum Request>>
cost_func:CostFunction
+energy: numpy.float64
+params: numpy.ndarray
Matplotlib
fig:Pyplot
+fig: Figure
+ax: Axes[1...*][unique, ordered]
+plot(range(x: int, y: numpy.float64))
+set_xlabel("iters": String)
+set_ylabel("cost_history": String)
+draw(): Figure
1..* +"iters", +"cost_history"
Qiskit
pm:StagedPassManager
+generate_preset_pass_manager(target: Target, optimization_level: int): StagedPassManager
+run(circuits: EfficientSU2): EfficientSU2
transforms
1 +num_qubits
constructs
hamiltonian:SparsePauliOp
+num_qubits: int[0...1][unique, unordered}
+from_list(List[Tuple[PauliList: String, coefficients: Complex]]): SparsePauliOp
+apply_layout(layout: TranspileLayout): SparsePauliOp
transforms
1 +layout
ansatz:EfficientSU2
+layout: TranspileLayout[0...1][unique, unordered}
+num_parameters: int
+decompose(): EfficientSU2
+draw(output: String, idle_wires: Boolean, style: String): Figure
1 +num_parameters
NumPy
x0:Numpy.ndarray
+pi: numpy.float64
+random.random(size: num_parameters): numpy.ndarray
1 +target
<<Quantum Driver>>
backend:QiskitRuntimeService
+channel: "ibm_quantum"
+least_busy(operational: Boolean, simulator: Boolean): IBMBackend
+backend(): IBMBackend
<<instantiate>>
<<Quantum Driver>>
IBMBackend
+target: Target
1 +backend
<<Quantum Request>> cost_func:CostFunction
+cost_history_dict: Dict
+pub: Tuple[ansatz: EfficientSU2, hamiltonian: List[SparsePauliOp], params: List[numpy.ndarray]]
+result: PrimitiveResult
+energy: numpy.float64
<<Quantum Request>>+cost_func(params: numpy.ndarray, ansatz: EfficientSU2, hamiltonian: SparsePauliOp, estimator: EstimatorV2): numpy.float64
updates
cost_history_dict:Dict
+"prev_vector": numpy.ndarray[0...1][unique,ordered}
+"iters": int
+"cost_history": numpy.float64[0...1][unique,ordered}
<<Quantum>> estimator:EstimatorV2
+mode: Session
+estimator(mode: Session)
+options.default_shots(default_shots: int)
+run(pub: Tuple[ansatz: EfficientSU2, hamiltonian: List[SparsePauliOp], params: List[NumPyndarray]]): RuntimeJobV2
1 +session
session:Session
+session(backend: IBMBackend)