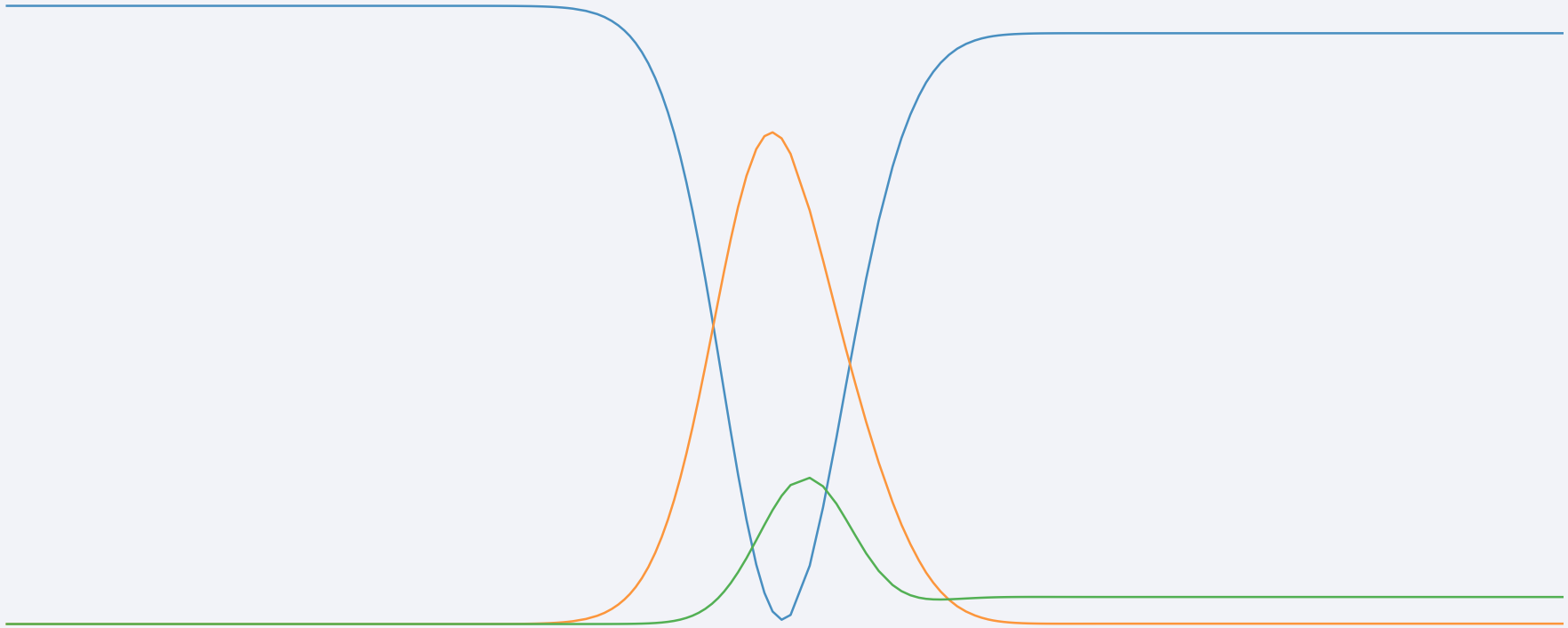
Pulse Backend in Qiskit Experiments

Mentees: R K Rupesh, JeongWon Kim

Mentor: Daniel Egger



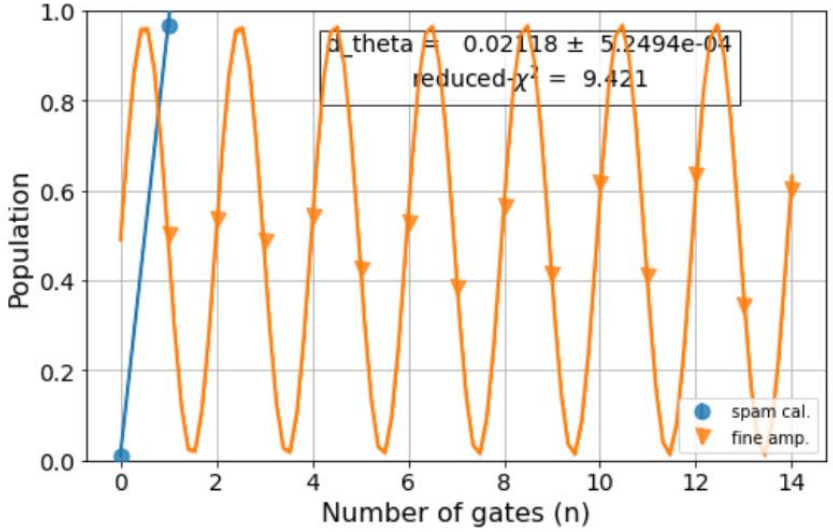
Introduction

Qiskit Experiments
a framework to

run a set of circuits
(experiment)

and

analyze the output
(analysis)



Currently

- Hard to test new experiments ⟹ simulate pulse schedules
- Tests hard-code the error patterns of many experiments

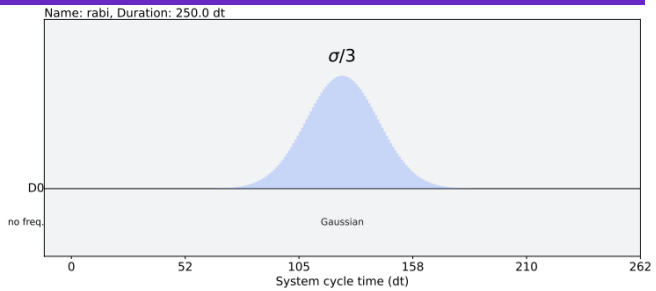
Goal

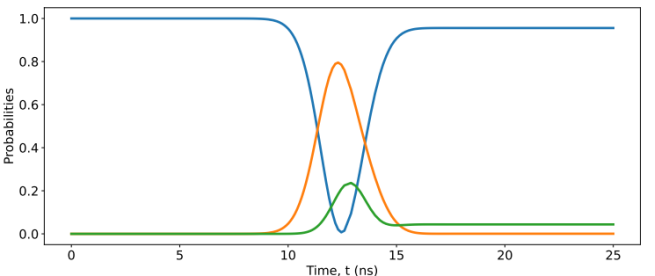
- *Realistic* backend
- *Efficient* tests based on pulse-level dynamics
- Tutorials without hardware backend

R K Rupesh, JeongWon Kim, Daniel Egger

Hamiltonian Simulation

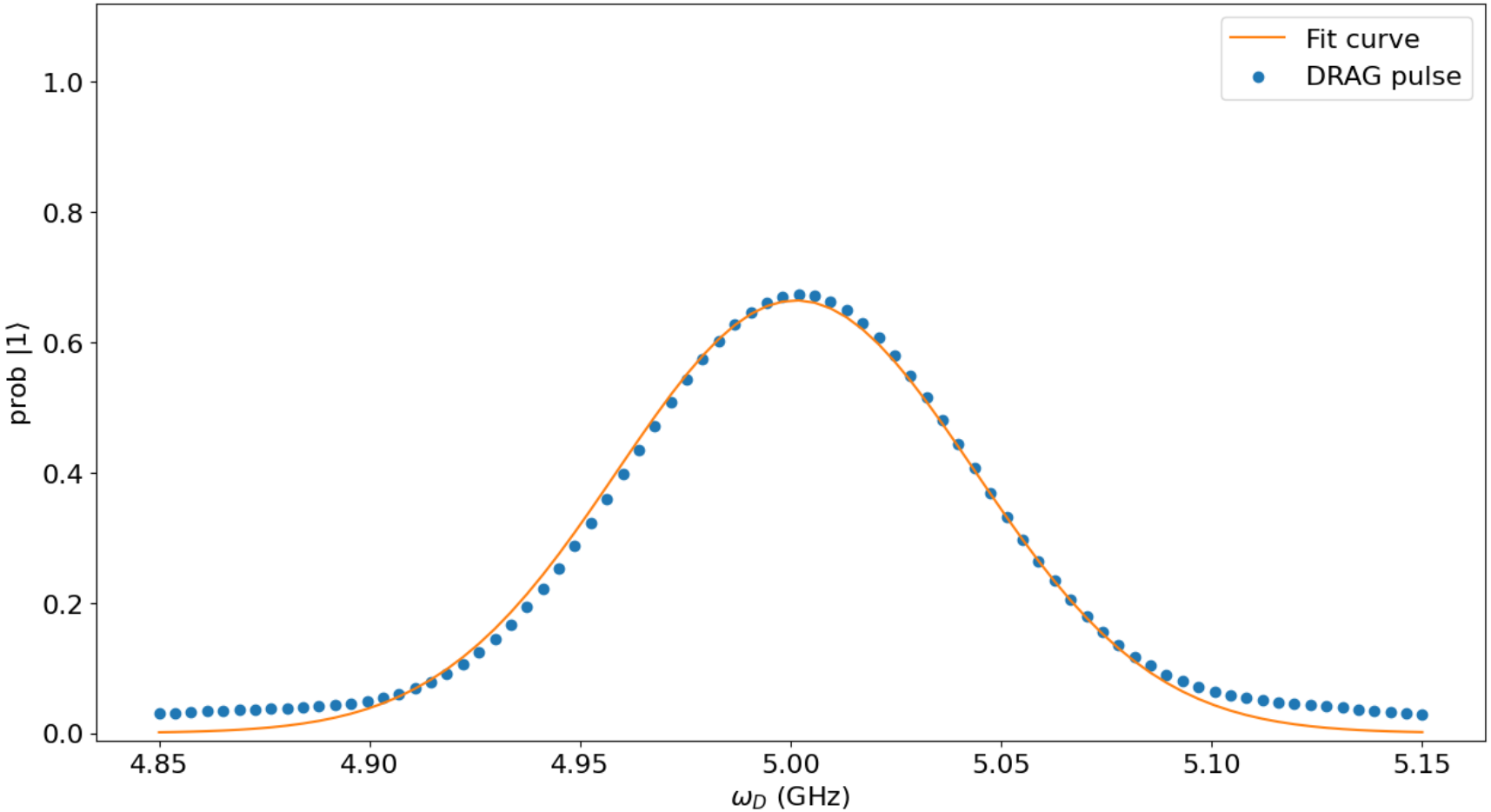
- We use a 3-level model for the qubit.
- This allows us to model leakage
- And perform DRAG experiments





Qubit-Spectroscopy

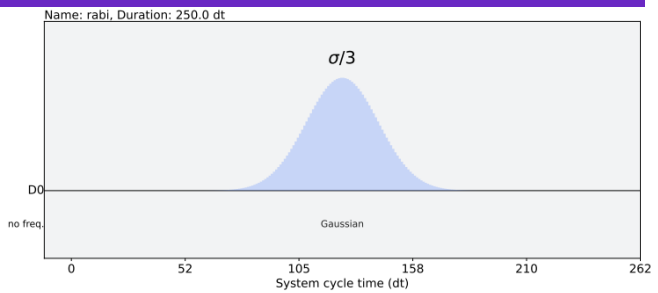
$\omega_{01} = 5.0GHz \quad \Delta = -0.25GHz \quad \lambda = 0.8GHz \quad dt = 0.1ns \quad T = 250dt = 25.0ns \quad \sigma * amp = 5.0 * 0.05$

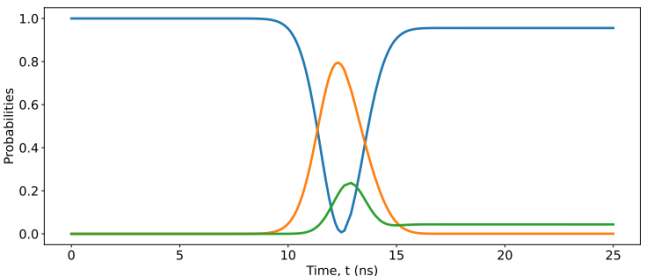


$$H = \hbar \sum_{j=1,2} \omega_j \Pi_j + \varepsilon(t)\lambda_j\left(\sigma_j^+ + \sigma_j^-\right)$$

Hamiltonian Simulation

- We use a 3-level model for the qubit.
- This allows us to model leakage
- And perform DRAG experiments

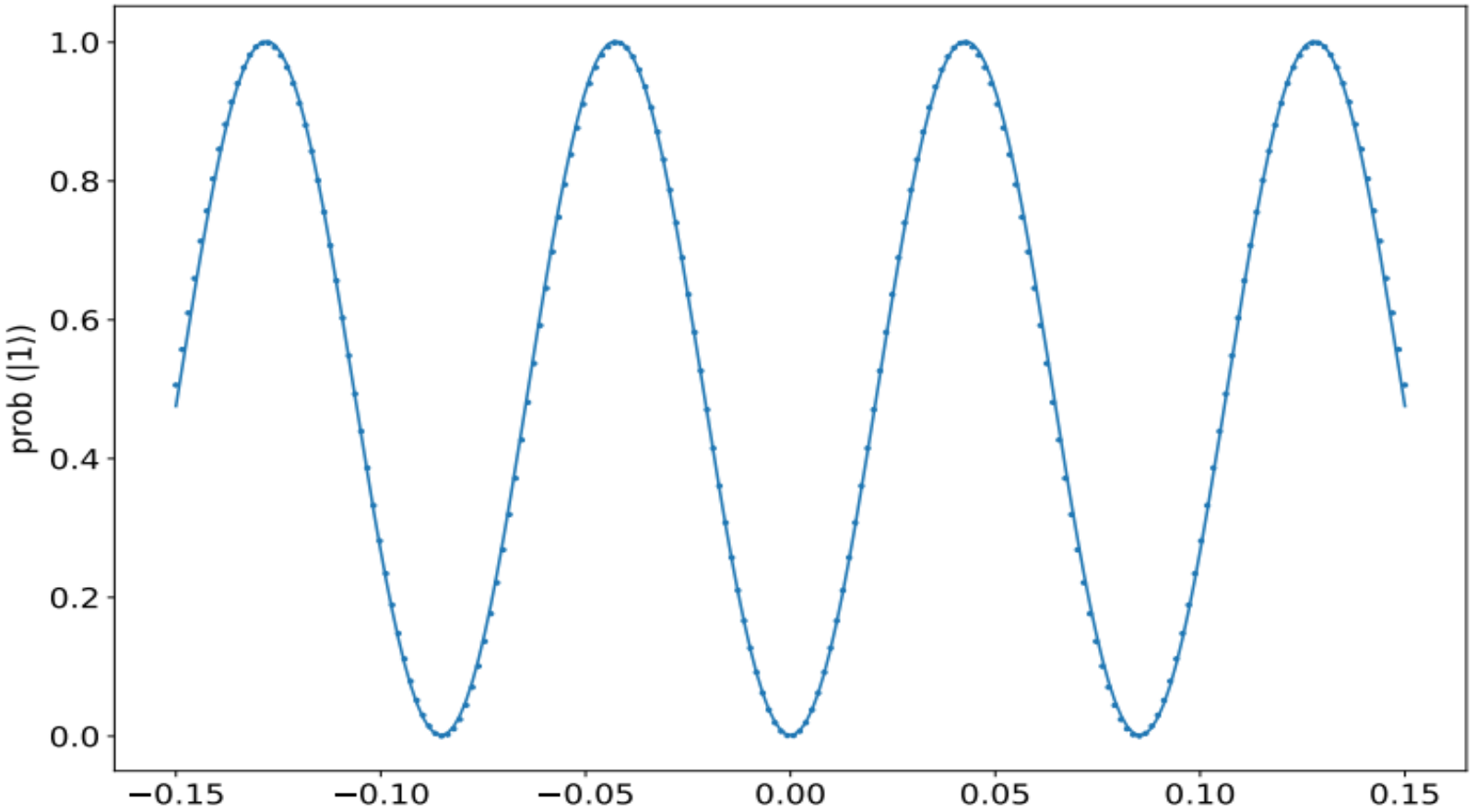




Rabi Experiment (RoughAmp)

Qiskit

$\omega_{01} = \omega_D = 5.0GHz \quad \Delta = -0.25GHz \quad \lambda = 0.8GHz \quad dt = 0.1ns \quad T = 250dt = 25.0ns \quad \sigma = 5.0ns$



$$H = \hbar \sum_{j=1,2} \omega_j \Pi_j + \varepsilon(t)\lambda_j\left(\sigma_j^+ + \sigma_j^-\right)$$

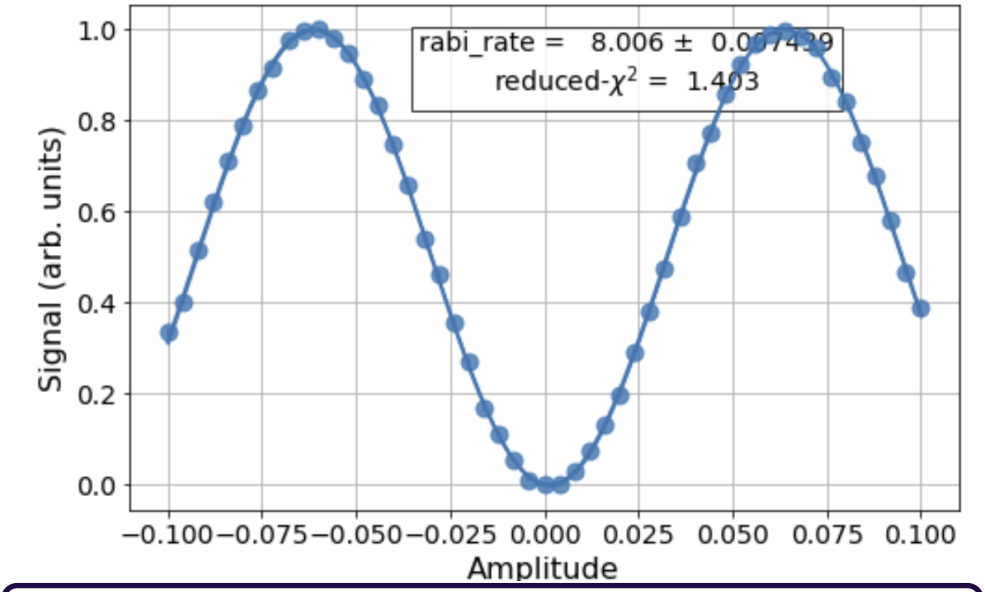
Current testing framework

Circuits

Calibration Experiment in Qiskit-Experiment

- ✓ FrequencyCal, RoughFrequencyCal, FineFrequencyCal
- ✓ RoughDragCal, Fine(X)(SX)DragCal
- ✓ Fine(X)(SX)AmplitudeCal, Rough(XSX)AmplitudeCal, FineSXAmplitudeCal

MockIQBackend (MockIQExperimentHelper) : Hard codes an error pattern



Results : IQ data, counts (qiskit-experiment analysis)

Current testing framework

Qiskit

Circuits

Calibration Experiment in Qiskit-Experiment

- ✓ FrequencyCal, RoughFrequencyCal, FineFrequencyCal
- ✓ RoughDragCal, Fine(X)(SX)DragCal
- ✓ Fine(X)(SX)AmplitudeCal, Rough(XSX)AmplitudeCal, FineSXAmplitudeCal

MockIQBackend (MockIQExperimentHelper) : Hard codes an error pattern

```
def compute_probabilities(self, circuits: List[QuantumCircuit]) -> List[Dict[str, float]]
    """Returns the probability based on the rotation angle and amplitude_to_angle."""
    amplitude_to_angle = self.amplitude_to_angle
    output_dict_list = []
    for circuit in circuits:
        probability_output_dict = {}
        amp = next(iter(circuit.calibrations["Rabi"].keys()))[1][0]

        # Dictionary of output string vectors and their probability
        probability_output_dict["1"] = np.sin(amplitude_to_angle * amp) ** 2
        probability_output_dict["0"] = 1 - probability_output_dict["1"]
        output_dict_list.append(probability_output_dict)
    return output_dict_list
```

Results : IQ data, counts (qiskit-experiment analysis)

Current testing framework

Qiskit

Calibration Experiment in Qiskit-Experiment
- ✓ FrequencyCal, RoughFrequencyCal, FineFrequencyCal
- ✓ RoughDragCal, Fine(X)(SX)DragCal
- ✓ Fine(X)(SX)AmplitudeCal, Rough(XSX)AmplitudeCal, FineSXAmplitudeCal

Circuits

MockIQBackend (MockIQExperimentHelper) :
Hard codes an error pattern

```
def compute_probabilities(self, circuits: List[QuantumCircuit]) -> List[Dict[str, float]]
    """Returns the probability based on the rotation angle and amplitude_to_angle."""
    amplitude_to_angle = self.amplitude_to_angle
    output_dict_list = []
    for circuit in circuits:
        probability_output_dict = {}
        amp = next(iter(circuit.calibrations["Rabi"].keys()))[1][0]

        # Dictionary of output string vectors and their probability
        probability_output_dict["1"] = np.sin(amplitude_to_angle * amp) ** 2
        probability_output_dict["0"] = 1 - probability_output_dict["1"]
        output_dict_list.append(probability_output_dict)
    return output_dict_list
```

Results : IQ data, counts (qiskit-experiment analysis)

Calibration Experiment in Qiskit-Experiment
- ✓ FrequencyCal, RoughFrequencyCal, FineFrequencyCal
- ✓ RoughDragCal, Fine(X)(SX)DragCal
- ✓ Fine(X)(SX)AmplitudeCal, Rough(XSX)AmplitudeCal, FineSXAmplitudeCal

Circuits

New PulseBackend () : PulseSimulator
Extract Pulse Schedule →
Simulate with Qiskit-Dynamics

Results : IQ data, counts(qiskit-experiment analysis)

Current testing framework

Qiskit

Circuits

Calibration Experiment in Qiskit-Experiment
- ✓ FrequencyCal, RoughFrequencyCal, FineFrequencyCal
- ✓ RoughDragCal, Fine(X)(SX)DragCal
- ✓ Fine(X)(SX)AmplitudeCal, Rough(XSX)AmplitudeCal, FineSXAmplitudeCal

MockIQBackend (MockIQExperimentHelper) :
Hard codes an error pattern

```
def compute_probabilities(self, circuits: List[QuantumCircuit]) -> List[Dict[str, float]]
    """Returns the probability based on the rotation angle and amplitude_to_angle."""
    amplitude_to_angle = self.amplitude_to_angle
    output_dict_list = []
    for circuit in circuits:
        probability_output_dict = {}
        amp = next(iter(circuit.calibrations["Rabi"].keys()))[1][0]

        # Dictionary of output string vectors and their probability
        probability_output_dict["1"] = np.sin(amplitude_to_angle * amp) ** 2
        probability_output_dict["0"] = 1 - probability_output_dict["1"]
        output_dict_list.append(probability_output_dict)
    return output_dict_list
```

Results : IQ data, counts (qiskit-experiment analysis)

Circuits

Calibration Experiment in Qiskit-Experiment
- ✓ FrequencyCal, RoughFrequencyCal, FineFrequencyCal
- ✓ RoughDragCal, Fine(X)(SX)DragCal
- ✓ Fine(X)(SX)AmplitudeCal, Rough(XSX)AmplitudeCal, FineSXAmplitudeCal

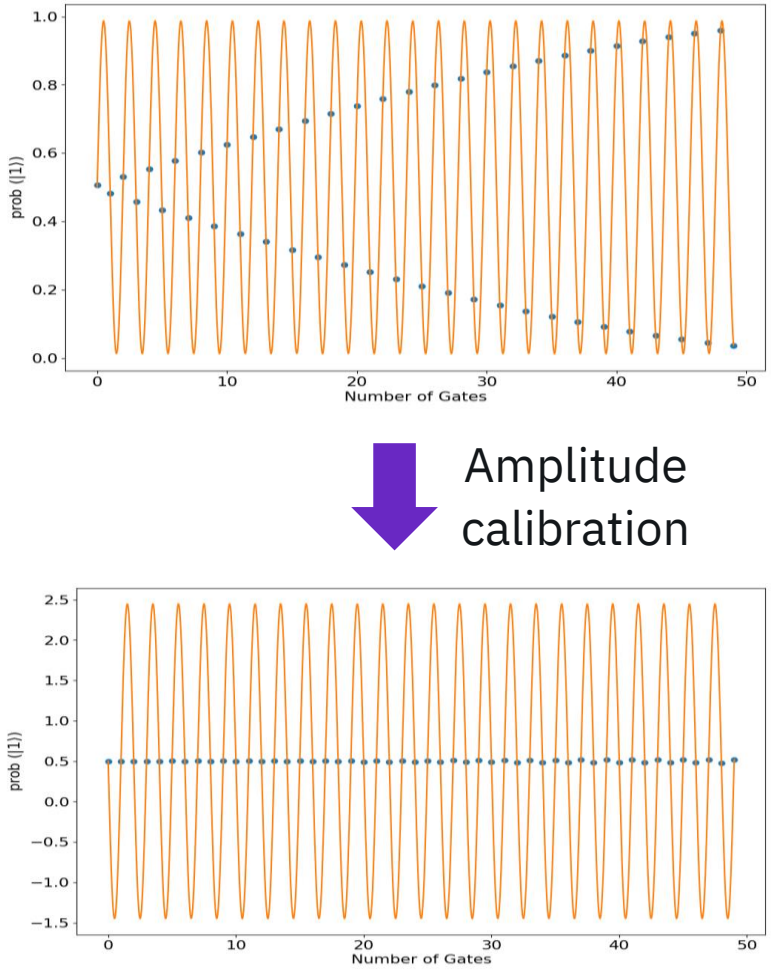
New PulseBackend () : PulseSimulator
Extract Pulse Schedule →
Simulate with Qiskit-Dynamics

1. Extract pulse schedules from calibrations in QuantumCircuit
2. Simulate the pulse schedules to obtain time evolution operators with Qiskit-Dynamics
3. Evolve ground state using time evolution operator
4. Generate count/IQ data from evolved state

Results : IQ data, counts(qiskit-experiment analysis)

Conclusions

Pulse simulation of fine amplitude calibration



Main Goals

- Realistic Tests
- Tutorials without Hardware Backend

Where We are

- Pulse Backend structure
- Rabi, Spectroscopy, FineAmp