In [1]:

```
# configure jax to use 64 bit mode
import jax
jax.config.update("jax_enable_x64", True)

# tell JAX we are using CPU
jax.config.update('jax_platform_name', 'cpu')

# import Array and set default backend
from qiskit_dynamics.array import Array
Array.set_default_backend('jax')
```

In [2]:

```
from qiskit_dynamics.array import wrap

jit = wrap(jax.jit, decorator=True)
```

In [3]:

```
import numpy as np
import jax.numpy as jnp
from qiskit.quantum_info import Operator
from qiskit_dynamics import Solver, Signal
from qiskit_dynamics.array import Array

X = Operator.from_label('X')
Z = Operator.from_label('Z')
L = np.outer([1.,0.],[0.,1.])
```

In [4]:

```
#list of Hamiltonian parameters
parameters = [0.5,1.,1.]
```

In [5]:
```
r, w, B = parameters
static_hamiltonian = B * 2 * np.pi * w * Z/2
hamiltonian_operators = [2 * np.pi * r * X/2]

evals = np.asarray(np.real(np.sort(np.linalg.eigvals(static_hamiltonian)))) / (2*np.pi)
ω = evals[1]-evals[0]

#sets up a solver instance with the current Hamiltonian
solver = Solver(
        static_hamiltonian=static_hamiltonian,
        hamiltonian_operators=hamiltonian_operators,
        rotating_frame=static_hamiltonian,
        static_dissipators = [L],
        in_frame_basis=False,
        rwa_cutoff_freq=3*ω,
        rwa_carrier_freqs=[ω]
        )
```

In [6]:
```
#density matrix for the zero state, to be used as the initial state in the simulations
p0 = np.outer(np.array([0., 1.], dtype=complex), np.array([0., 1.], dtype=complex))
```

Utilities

In [7]:
```
#constructs a 'Signal' object given some input amplitude and frequency
def signal_from_input(pulse_input):
    amp = Array(pulse_input[0])
    w = pulse_input[1]
    signal = [Signal(amp, carrier_freq = 1.)]
    signal[0].carrier_freq = w
    return signal
```

In [8]:
```
#updates the Hamiltonian given some input parameters
def update_static_hamiltonian(parameters):
    #directly updates the hamiltonian
    w = parameters[0]
    r = parameters[1]
    B = parameters[2]
    solver.model.static_hamiltonian = B * 2 * np.pi * w * jnp.asarray(Z)/2
    solver.model.rwa_carrier_freqs = w
```

In [9]:
```
#updates the lindblad operators given an input parameter
def update_lindbladian(parameters):
    #directly updates the hamiltonian
    a = parameters[0]
    nL = a * np.outer([1.,0.],[0.,1.])
    solver.model.lindblad_dissipators = [nL]
```

In [10]:
```
#updates the hamiltonian operators given input parameters
def update_hamiltonian_operators(parameters):
    #directly updates the hamiltonian
    w = parameters[0]
    r = parameters[1]
    B = parameters[2]
    solver.model.hamiltonian_operators = [2 * np.pi * r * jnp.asarray(X)/2]
```

Simulation with updating Hamiltonian parameters

In [11]:
```
#Simulations the evolution of the system under a hamiltonian and 'Signal' determined by inputs
def sim_function_1(pulse_input, hamiltonian_parameters):
    # define a constant signal
    signals = signal_from_input(pulse_input)
    update_static_hamiltonian(hamiltonian_parameters)

    # simulate and return results
    results = solver.solve(
        t_span=[0, 50.],
        y0=p0,
        signals=signals,
        t_eval=np.linspace(0, 50., 100),
        method='jax_odeint'
    )

    return results.y
```

In [12]:
```
fast_sim_1 = jit(sim_function_1)
```

In [13]:
```
#changing the inputs alters the result and is jit-able
%time ys1 = fast_sim_1([1., 1.], [1., 1., 5.]).block_until_ready()

print(ys1[-1])
```

```
CPU times: user 750 ms, sys: 13.6 ms, total: 763 ms
Wall time: 754 ms
[[0.99817336+2.15522128e-16j 0.04268088+5.97891268e-04j]
 [0.04268088-5.97891268e-04j 0.00182664+1.01330241e-18j]]
```

Simulation with updating Lindblad dissipators - does not work

In [14]:
```python
#same function, this time trying to update the lindblad dissipators instead of the static_hamiltonian
def sim_function_2(pulse_input, lindblad_parameters, hamiltonian_parameters):
    # define a constant signal
    signals = signal_from_input(pulse_input)
    #update_lindbladian(lindblad_parameters)
    update_hamiltonian_operators(hamiltonian_parameters)

    # simulate and return results
    results = solver.solve(
        t_span=[0, 5000.],
        y0=p0,
        signals=signals,
        t_eval=np.linspace(0, 5000., 100),
        method='jax_odeint'
    )

    return results.y
```

In [15]:
```python
fast_sim_2 = jit(sim_function_2)
```

In [16]:
```python
#It is not possible to alter the lindblad or hamiltonian operators
%time ys2 = fast_sim_2([1., 1.], [5.], [1., 1., 5.]).block_until_ready()

print(ys2[-1])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<timed exec> in <module>

~/.local/lib/python3.10/site-packages/qiskit_dynamics/array/wrap.py in wrapped_decorated(*f_args, **f_kwargs)
    161                 f_args = _wrap_args(f_args)
    162                 f_kwargs = _wrap_kwargs(f_kwargs)
--> 163                 result = _wrap_function(decorated)(*f_args, **f_kwargs)
    164                 return Array._wrap(result)
    165

~/.local/lib/python3.10/site-packages/qiskit_dynamics/array/wrap.py in wrapped_func(*args, **kwargs)
    103             args = _wrap_args(args)
    104             kwargs = _wrap_kwargs(kwargs)
--> 105             result = _wrap_array_function(func)(*args, **kwargs)
    106             return Array._wrap(result)
    107

~/.local/lib/python3.10/site-packages/qiskit_dynamics/array/wrap.py in wrapped_function(*args, **kwargs)
     57
     58         # Evaluate function with unwrapped inputs
---> 59         result = func(*args, **kwargs)
     60
     61         # Unwrap result

    [... skipping hidden 12 frame]

~/.local/lib/python3.10/site-packages/qiskit_dynamics/array/wrap.py in wrapped_function(*args, **kwargs)
     57
     58         # Evaluate function with unwrapped inputs
---> 59         result = func(*args, **kwargs)
     60
     61         # Unwrap result

/tmp/ipykernel_48700/404429297.py in sim_function_2(pulse_input, lindblad_parameters, hamiltonian_parameters)
      4     signals = signal_from_input(pulse_input)
      5     #update_lindbladian(lindblad_parameters)
----> 6     update_hamiltonian_operators(hamiltonian_parameters)
      7
      8     # simulate and return results

/tmp/ipykernel_48700/296514943.py in update_hamiltonian_operators(parameters)
      5     r = parameters[1]
      6     B = parameters[2]
----> 7     solver.model.hamiltonian_operators = [2 * np.pi * r * jnp.asarray(X)/2]

AttributeError: can't set attribute 'hamiltonian_operators'
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_48700/3353554311.py in <module>
      2 get_ipython().run_line_magic('time', 'ys2 = fast_sim_2([1., 1.], [5.], [1., 1., 5.]).block_until_ready()')
      3
----> 4 print(ys2[-1])

NameError: name 'ys2' is not defined
```

In [ ]: