Hamiltonian Simulation – trotter method

Input Matrix:

[[2, -1],
[-1, 2]]

As dimension matrix is 2 so it uses 1 qubit for unitary evolution.

As its not unitary matrix so it uses unitary evolution using basis gates to approximate the given matrix:

Step1: Derive the weighted Pauli gates

Weighted Pauli Gates - IXYZ = Possible paulis
I gate*matrix = gives wieght of I
x gate*matrix = gives wieghts of x paulis
y gate*matrix = gives wieghts of y paulis
z gate*matrix = gives wieghts of z paulis

Possible Basis: IXYZ

z	X	Type of Gate
FALSE	FALSE	Identity
FALSE	TRUE	X
TRUE	TRUE	Y
TRUE	FALSE	Z

Based on weight it retains the combination of paulis with which the given input matrix can be approximated.

Calculated weights:

[((4+0j), Pauli(z=[False], x=[False])),
((-2+0j), Pauli(z=[False], x=[True])),
(0j, Pauli(z=[True], x=[True])),
(0j, Pauli(z=[True], x=[False]))]

Num_qubits=1

Value for coeff = 2 ** (-num_qubits) = 1/2

Final pauli list=

Based on condition: wieght=trace*coeff and wieght>0
[[(2+0j), Pauli(z=[False], x=[False])], [(-1+0j), Pauli(z=[False], x=[True])]]

Step 2: Calculation of evolution time (t)

Final Pauli List: [[(2+0j), Pauli(z=[False], x=[False])], [(-1+0j), Pauli(z=[False], x=[True])]]

lmax=sum([abs(p[0]) for p in pauli list])
lmax=(2+1)
lmax=3

Number of qubits: 2

Evolution Time: t
t=(1 - 2 ** -numberofqubits) * 2 * np.pi / lmax

Example: number of qubits= 2
t=(1 - 2**-2)* 2 * np.pi/3
t=1.5707963267948966

Step 3: Build controlled unitary circuit

Input to Evolution Set:

First Qubit:
pauli_list: [[(2+0j), Pauli(z=[False], x=[False])], [(-1+0j), Pauli(z=[False], x=[True])]]
evo_time: -1.5707963267948966
num_time_slices: 1
controlled: True
power: 1   *- repeats the circuit 1 time*
use_basis_gates: True
shallow_slicing: False
barrier: False
cnot_qubit_pairs: [None, None]
top_xyz_pauli_indices: [-1, -1]

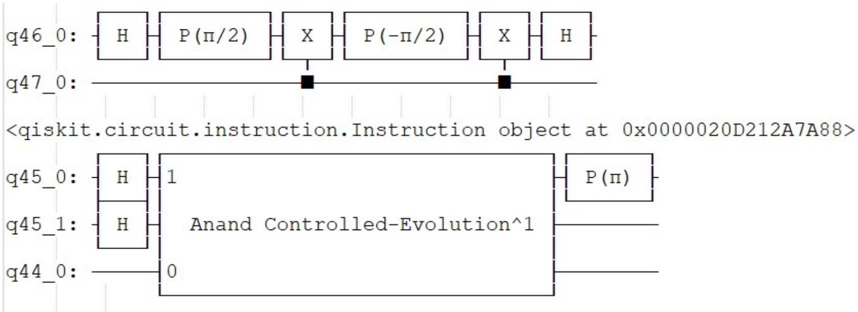
Based on x gate: value of theta for phase shift is derived from below formula

theta = (2.0 * pauli[0] * evo_time / num_time_slices).real
theta = (2.0 * -1 * -1.5707963267948966/1).real
theta = 3.141592653589793

Circuit:

H – hadmard gate
Phaseshift gate – (theta/2)
cx – controlled x
Phaseshift gate – (- theta/2)
cx – controlled x
H – hadmard gate

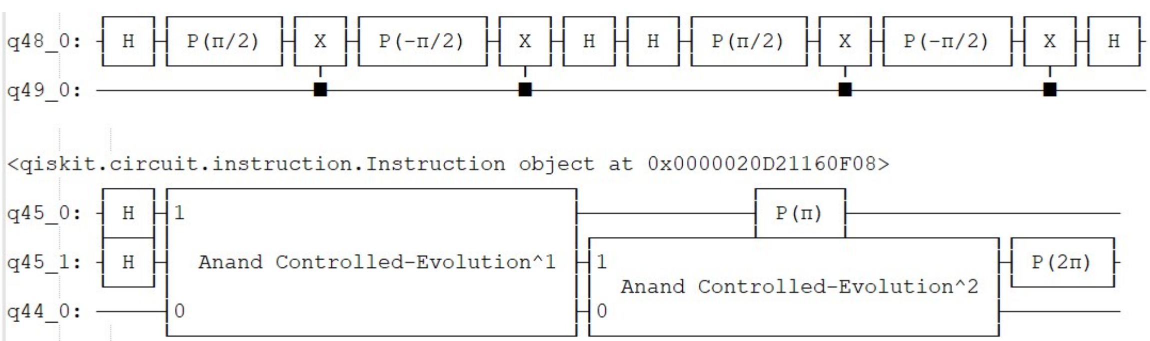
Repeat this circuit based on the power – that forms the controlled unitary evolution circuit.



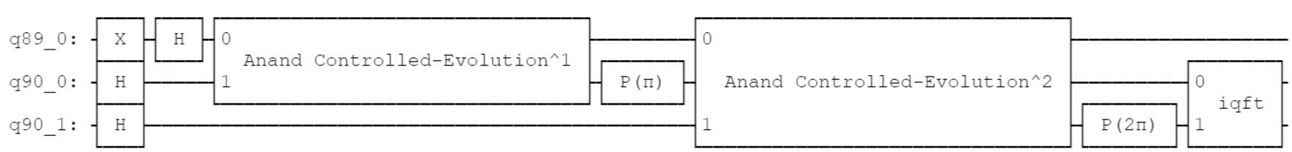
Second qubit:

Evolution Set
pauli_list: [[(2+0j), Pauli(z=[False], x=[False])], [(-1+0j), Pauli(z=[False], x=[True])]]
evo_time: -1.5707963267948966
num_time_slices: 1
controlled: True
power: 2 - *Repeats the circuit two times*
use_basis_gates: True
shallow_slicing: False
barrier: False
cnot_qubit_pairs: [None, None]
top_xyz_pauli_indices: [-1, -1]

Based on power it repeats this evolution circuit as controlled unitary.



Final Circuit:



Input Matrix:

```
[[ 2 -1  0  0]
 [-1  2 -1  0]
 [ 0 -1  2 -1]
 [ 0  0 -1  2]]
```

As dimension matrix is 4 so it uses 2 qubits for unitary evolution.

Calculated Paulis:

[[(2+0j), Pauli(z=[False, False], x=[False, False])],
 [(-1+0j), Pauli(z=[False, False], x=[True, False])],
 [(-0.5+0j), Pauli(z=[False, False], x=[True, True])],
 [(-0.5+0j), Pauli(z=[True, True], x=[True, True])]]

Weighted Pauli List: 40 combinations

Final:
[[(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(-0.6579630871775028+0j), Pauli(z=[False, False], x=[False, False])], [(0.3289815435887514-0j), Pauli(z=[False, False], x=[True, False])], [(0.1644907717943757-0j), Pauli(z=[False, False], x=[True, True])], [(0.1644907717943757-0j), Pauli(z=[True, True], x=[True, True])], [(0.1644907717943757-0j), Pauli(z=[True, True], x=[True, True])], [(0.1644907717943757-0j), Pauli(z=[False, False], x=[True, True])], [(0.3289815435887514-0j), Pauli(z=[False, False], x=[True, False])], [(-0.6579630871775028+0j), Pauli(z=[False, False], x=[False, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.20724538589718786+0j),

Pauli(z=[False, False], x=[True, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])]]

Step 2: Calculation of evolution time (t)

Pauli list: [[(2+0j), Pauli(z=[False, False], x=[False, False])], [(-1+0j), Pauli(z=[False, False], x=[True, False])], [(-0.5+0j), Pauli(z=[False, False], x=[True, True])], [(-0.5+0j), Pauli(z=[True, True], x=[True, True])]]

lmax=sum([abs(p[0]) for p in pauli list])
`lmax= 4`

Number of qubits: 5

Evolution Time: t
t=(1 - 2 ** -numberofqubits) * 2 * np.pi / lmax

Example: number of qubits= 5
t=(1 - 2**-5)* 2 * np.pi/4
t=`1.521708941582556`

Step 3: Build controlled unitary circuit

Input to Evolution Set:

First qubit:
pauli_list: [[(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(-0.6579630871775028+0j), Pauli(z=[False, False], x=[False, False])], [(0.3289815435887514-0j), Pauli(z=[False, False], x=[True, False])], [(0.1644907717943757-0j), Pauli(z=[False, False], x=[True, True])], [(0.1644907717943757-0j), Pauli(z=[True, True], x=[True, True])], [(0.1644907717943757-0j), Pauli(z=[True, True], x=[True, True])], [(0.1644907717943757-0j), Pauli(z=[False, False], x=[True, True])], [(0.3289815435887514-0j), Pauli(z=[False, False], x=[True, False])], [(-0.6579630871775028+0j), Pauli(z=[False, False], x=[False, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True,

True])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])], [(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])], [(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])], [(0.4144907717943757+0j), Pauli(z=[False, False], x=[False, False])]]

evo_time: -1.521708941582556
num_time_slices: 1
controlled: True
power: 1
use_basis_gates: True
shallow_slicing: False
barrier: False

Based on x gate: value of theta for phase shift is derived from below formula

Loops thru all 40 gates:

```
[(-0.20724538589718786+0j), Pauli(z=[False, False], x=[True, False])]
```

theta = (2.0 * pauli[0] * evo_time / num_time_slices).real
theta = (2.0 * -0.20724538589718786 * -1.521708941582556/1).real
theta = 0.6307343136429562

Circuit:

H – hadmard gate
Phaseshift gate – (theta/2)
cx – controlled x
Phaseshift gate – (- theta/2)
cx – controlled x
H – hadmard gate

```
[(-0.10362269294859393+0j), Pauli(z=[False, False], x=[True, True])]
```

theta = (2.0 * pauli[0] * evo_time / num_time_slices).real
theta = (2.0 * -0.10362269294859393* -1.521708941582556/1).real
theta = 0.3153671568214781

H – hadmard gate
Phaseshift gate – (theta/2)
cx – controlled x
Phaseshift gate – (- theta/2)
cx – controlled x
H – hadmard gate

[(-0.10362269294859393+0j), Pauli(z=[True, True], x=[True, True])]

It's a y gate – when encountered y gate it always applies unitary

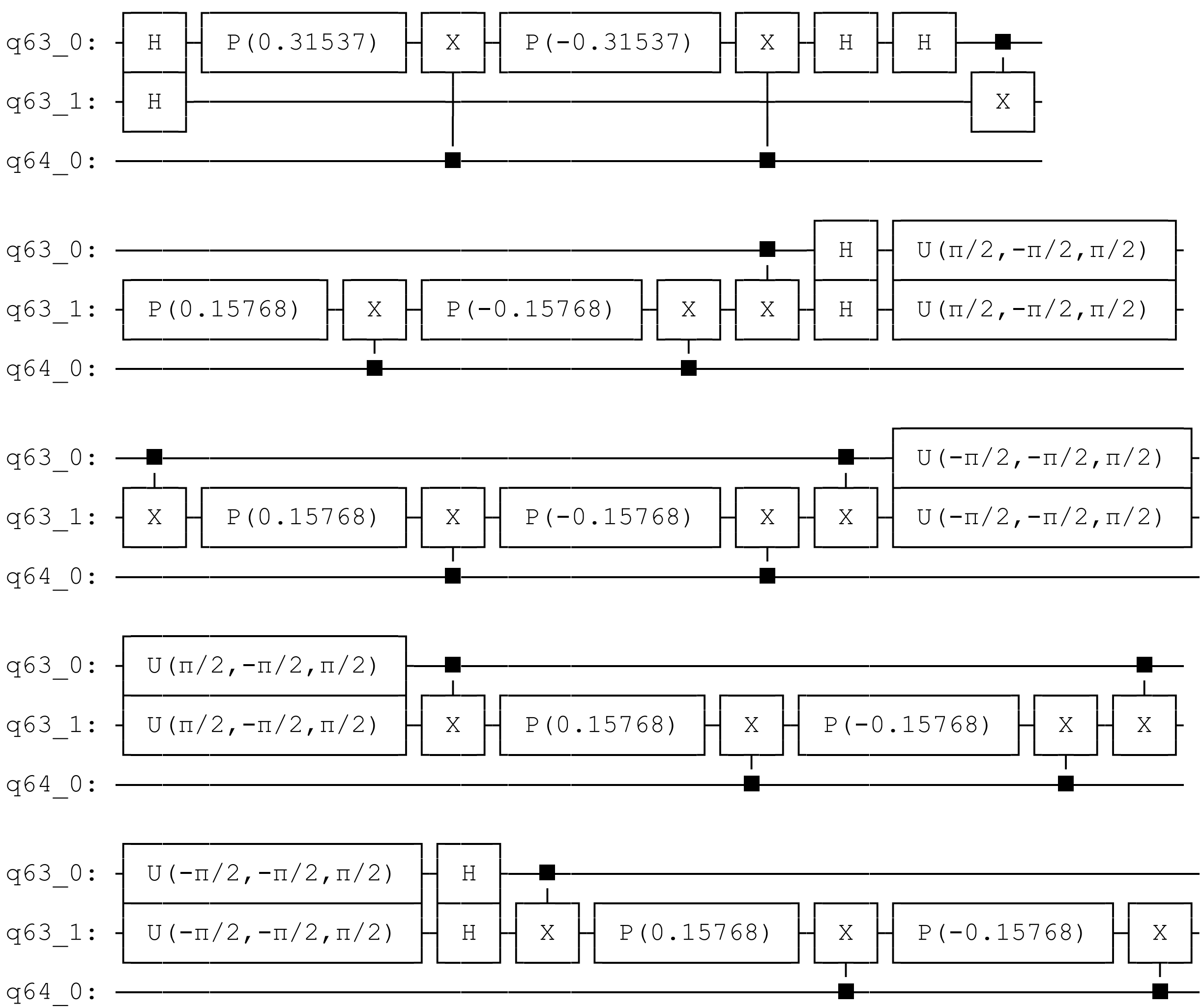
U gate:

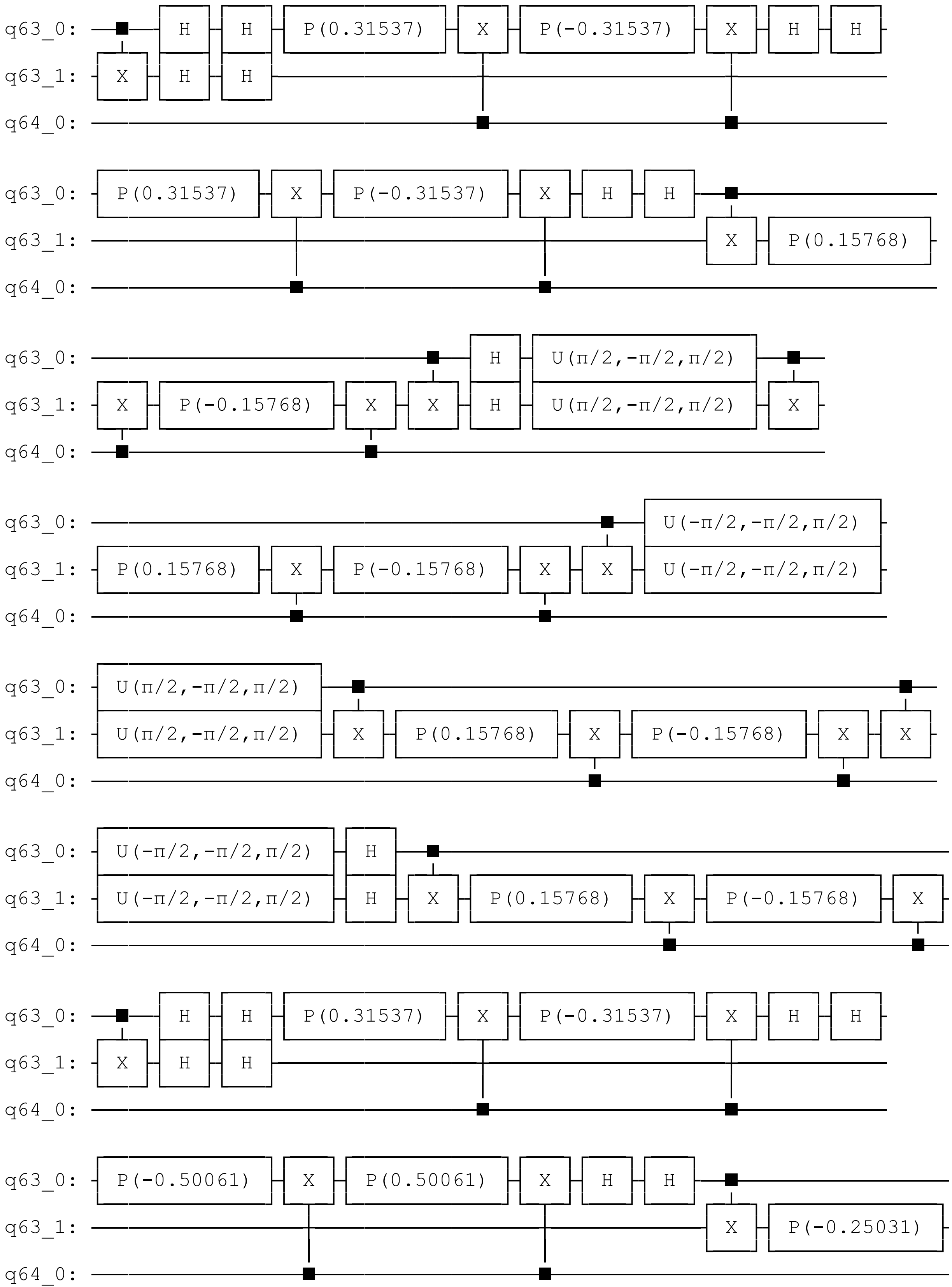
$$U(\theta, \phi, \lambda) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -e^{i\lambda}\sin(\frac{\theta}{2}) \\ e^{i\phi}\sin(\frac{\theta}{2}) & e^{i(\phi+\lambda)}\cos(\frac{\theta}{2}) \end{pmatrix}$$

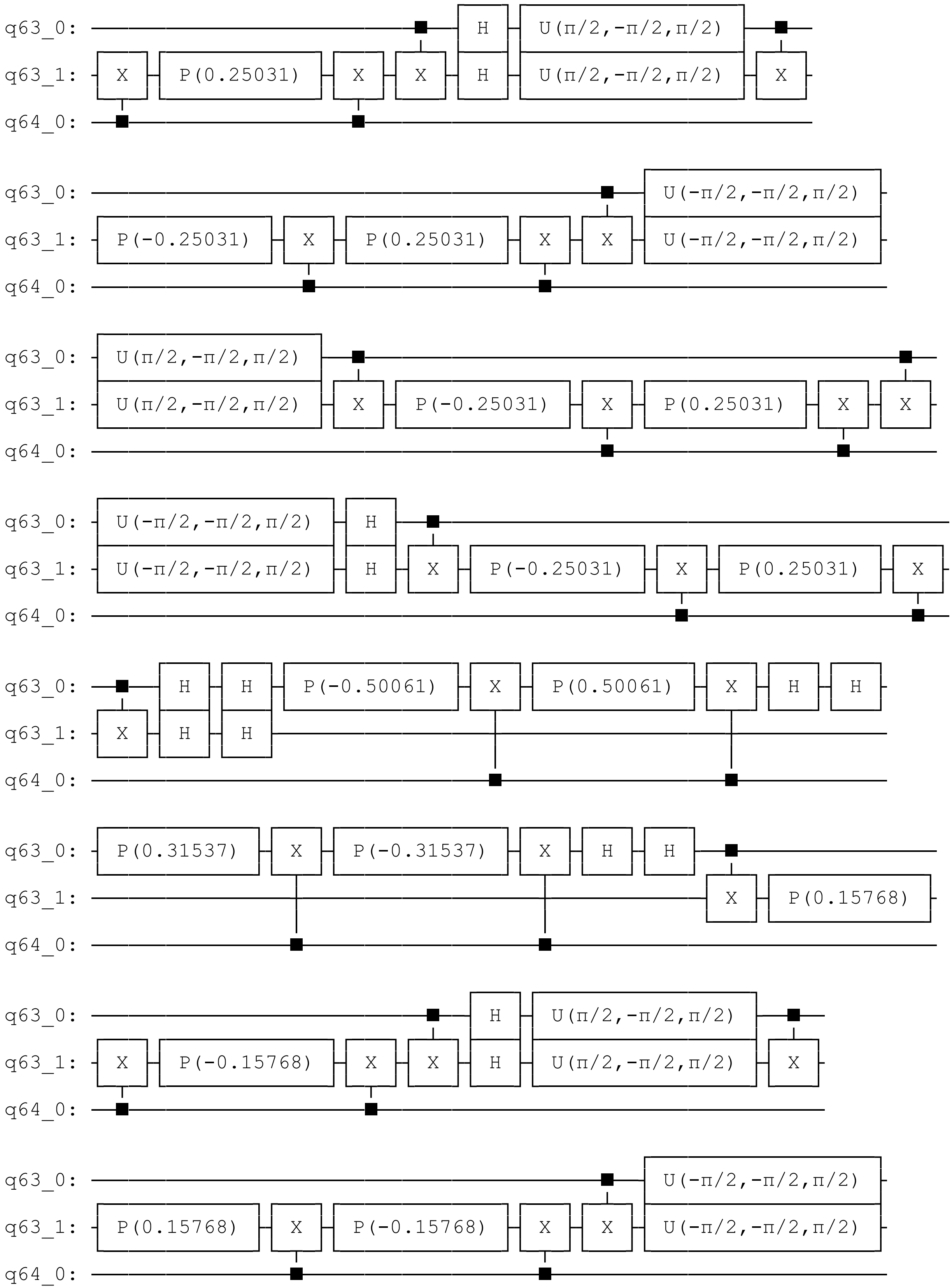
```
    u(-pi / 2, -pi / 2, pi / 2, qubit)
```

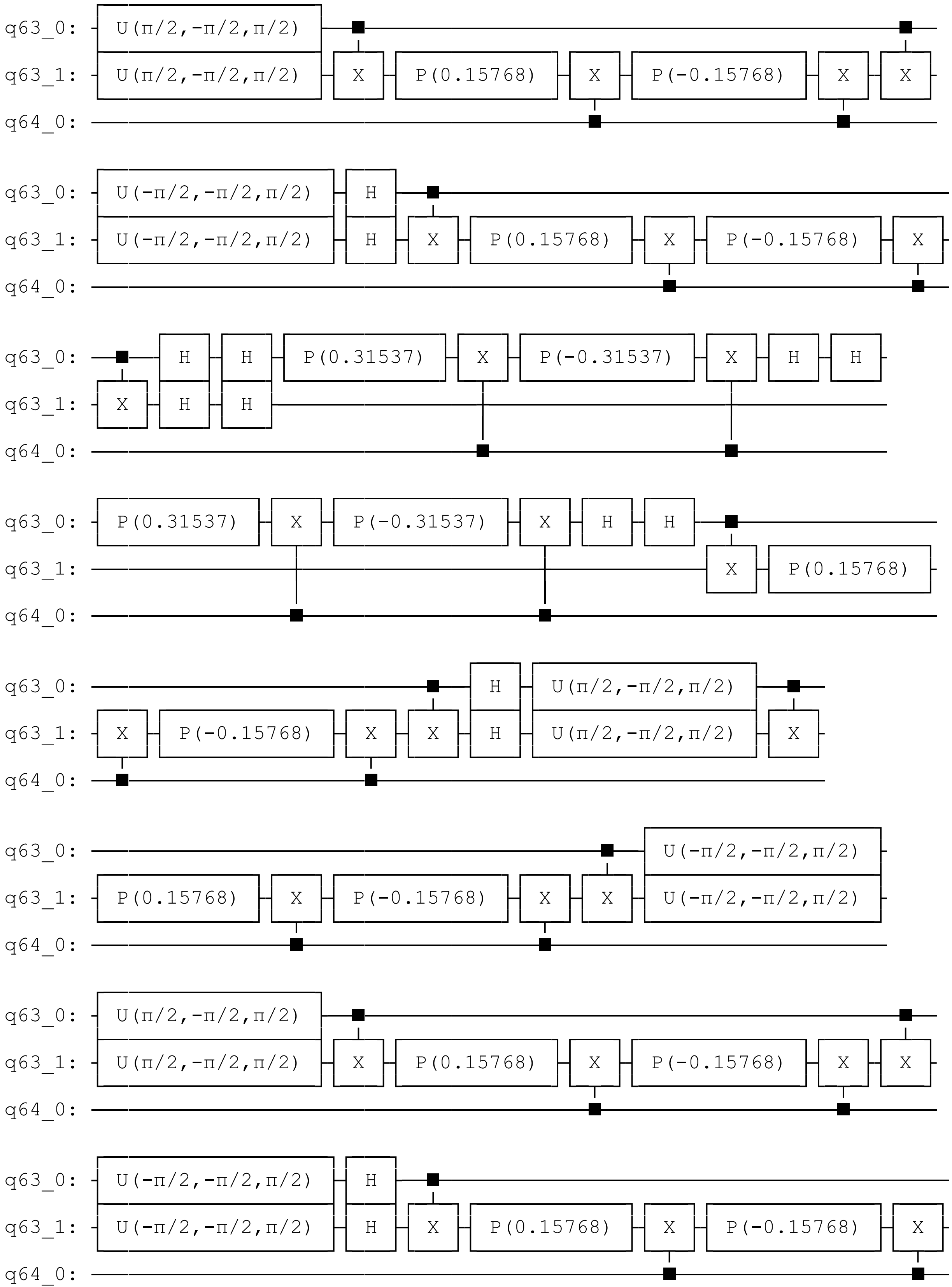
Finally, after looping thru all base gates:

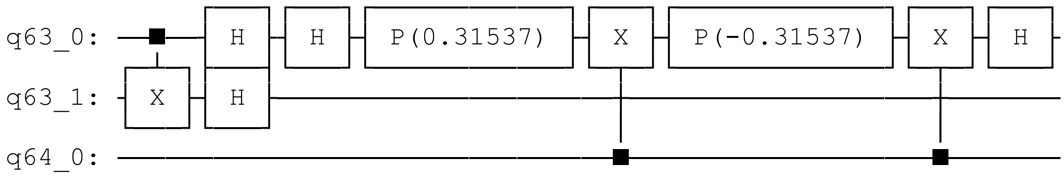
Final unitary evolution is as mentioned below:




q63_0:
q63_1:
q64_0:
X
H
H
H
H
P(0.31537)
X
P(-0.31537)
X
H
H
q63_0:
q63_1:
q64_0:
P(0.31537)
X
P(-0.31537)
X
H
H
X
P(0.15768)
q63_0:
q63_1:
q64_0:
X
P(-0.15768)
X
X
H
H
U(π/2,-π/2,π/2)
U(π/2,-π/2,π/2)
X
q63_0:
q63_1:
q64_0:
P(0.15768)
X
P(-0.15768)
X
X
U(-π/2,-π/2,π/2)
U(-π/2,-π/2,π/2)
q63_0:
q63_1:
q64_0:
U(π/2,-π/2,π/2)
U(π/2,-π/2,π/2)
X
P(0.15768)
X
P(-0.15768)
X
X
q63_0:
q63_1:
q64_0:
U(-π/2,-π/2,π/2)
U(-π/2,-π/2,π/2)
H
H
X
P(0.15768)
X
P(-0.15768)
X
q63_0:
q63_1:
q64_0:
X
H
H
H
H
P(0.31537)
X
P(-0.31537)
X
H
H
q63_0:
q63_1:
q64_0:
P(-0.50061)
X
P(0.50061)
X
H
H
X
P(-0.25031)


q63_0: H U(π/2,-π/2,π/2)
q63_1: X P(0.25031) X X H U(π/2,-π/2,π/2) X
q64_0:
q63_0: U(-π/2,-π/2,π/2)
q63_1: P(-0.25031) X P(0.25031) X X U(-π/2,-π/2,π/2)
q64_0:
q63_0: U(π/2,-π/2,π/2)
q63_1: U(π/2,-π/2,π/2) X P(-0.25031) X P(0.25031) X X
q64_0:
q63_0: U(-π/2,-π/2,π/2) H
q63_1: U(-π/2,-π/2,π/2) H X P(-0.25031) X P(0.25031) X
q64_0:
q63_0: H H P(-0.50061) X P(0.50061) X H H
q63_1: X H H
q64_0:
q63_0: P(0.31537) X P(-0.31537) X H H
q63_1: X P(0.15768)
q64_0:
q63_0: H U(π/2,-π/2,π/2)
q63_1: X P(-0.15768) X X H U(π/2,-π/2,π/2) X
q64_0:
q63_0: U(-π/2,-π/2,π/2)
q63_1: P(0.15768) X P(-0.15768) X X U(-π/2,-π/2,π/2)
q64_0:


q63_0:
q63_1:
q64_0:
U(π/2,-π/2,π/2)
U(π/2,-π/2,π/2)
X
P(0.15768)
X
P(-0.15768)
X
X
U(-π/2,-π/2,π/2)
U(-π/2,-π/2,π/2)
H
H
X
P(0.15768)
X
P(-0.15768)
X
X
H
H
H
H
P(0.31537)
X
P(-0.31537)
X
H
H
P(0.31537)
X
P(-0.31537)
X
H
H
X
P(0.15768)
X
P(-0.15768)
X
X
H
H
U(π/2,-π/2,π/2)
U(π/2,-π/2,π/2)
X
P(0.15768)
X
P(-0.15768)
X
X
U(-π/2,-π/2,π/2)
U(-π/2,-π/2,π/2)
U(π/2,-π/2,π/2)
U(π/2,-π/2,π/2)
X
P(0.15768)
X
P(-0.15768)
X
X
U(-π/2,-π/2,π/2)
U(-π/2,-π/2,π/2)
H
H
X
P(0.15768)
X
P(-0.15768)
X



Similarly, the number of qubits used is 5. So, it applies same unitary evolution on qubits as 2**i

Repeat evolution circuit = 1,2,4,8,16