sim-isi

March 30, 2024

[23]:
```
from qiskit import *
import matplotlib.pyplot as plt

simulator = Aer.get_backend('qasm_simulator')
```

1 Simulation d'un modèle de Ising sur un ordinateur quantque

[24]:
```
a = 1
J = 1
dt = 0.1

N = 3
```

L'opérateur d'évolution du hamiltonien

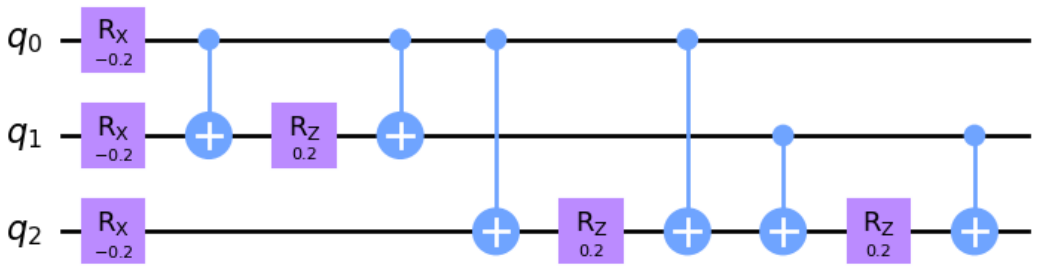
$$H = \sum_{(i,j)} J Z_i Z_j - a \sum_i X_i$$

peut être implément via le circuit ci-dessous qui donne donc $U = e^{iHt}$ (cf. https://share.phys.ethz.ch/~alps/cqp.pdf)

[25]:
```
# one dt circuit operation
ope = QuantumCircuit(N)

for i in range(N):
    ope.rx(-2*a*dt, i)

for i in range(N):
    for j in range(i+1, N):
        ope.cx(i, j)
        ope.rz(2*J*dt, j)
        ope.cx(i, j)

ope_gate = ope.to_gate()

ope.draw(output='mpl')
```

```
c:\Python310\lib\site-packages\qiskit\visualization\circuit\matplotlib.py:266:
FutureWarning: The default matplotlib drawer scheme will be changed to "iqp" in
a following release. To silence this warning, specify the current default
explicitly as style="clifford", or the new default as style="iqp".
  self._style, def_font_ratio = load_style(self._style)
```

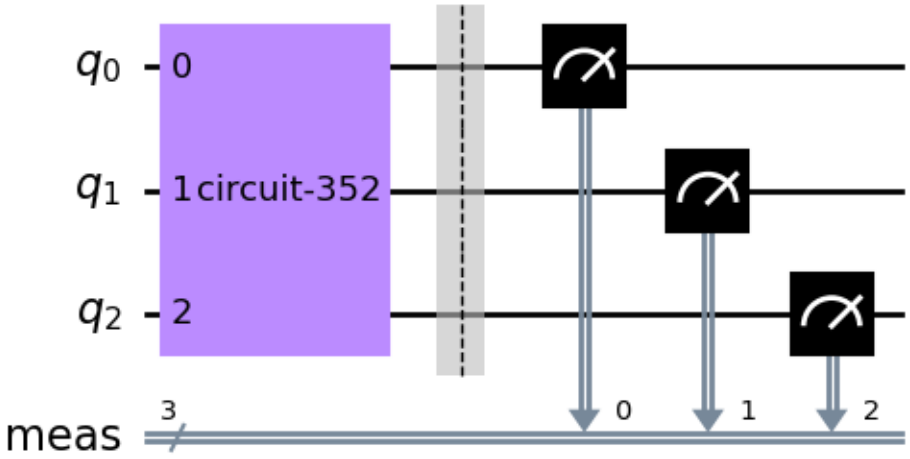
[25]:



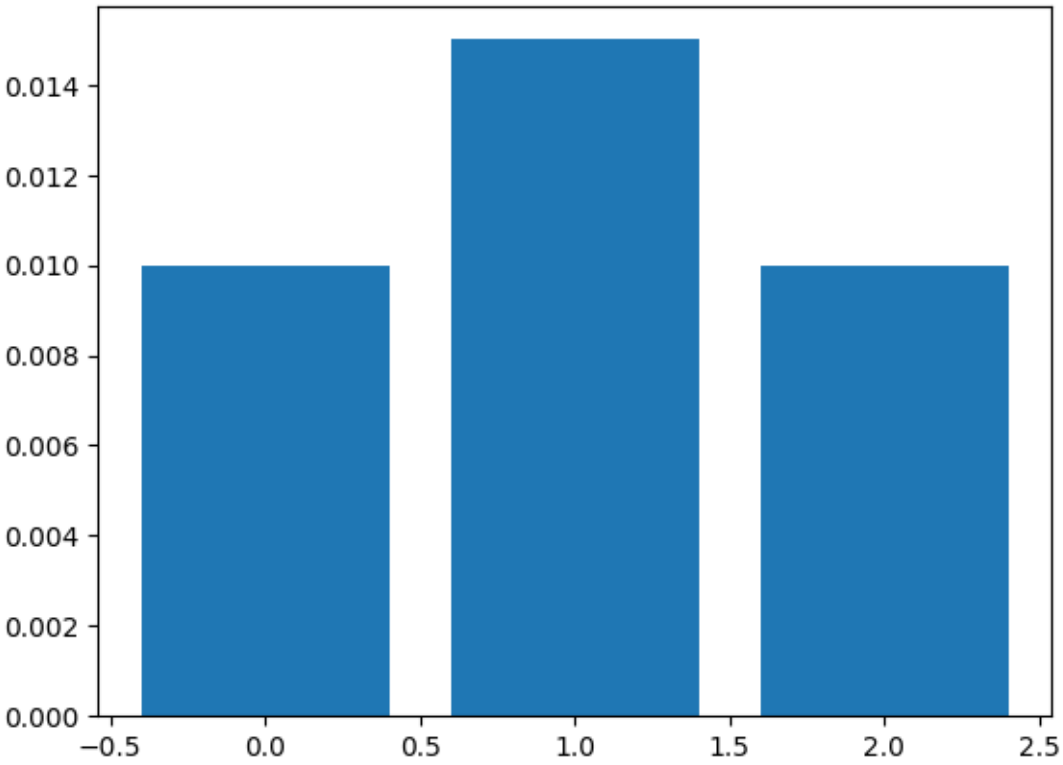
1.1 Évolution sur 1 Δt de l'état $|000\rangle$

[26]:
```
test = QuantumCircuit(N)

test.append(ope_gate, range(N))

test.measure_all()

test.draw(output='mpl')
```

[26]:

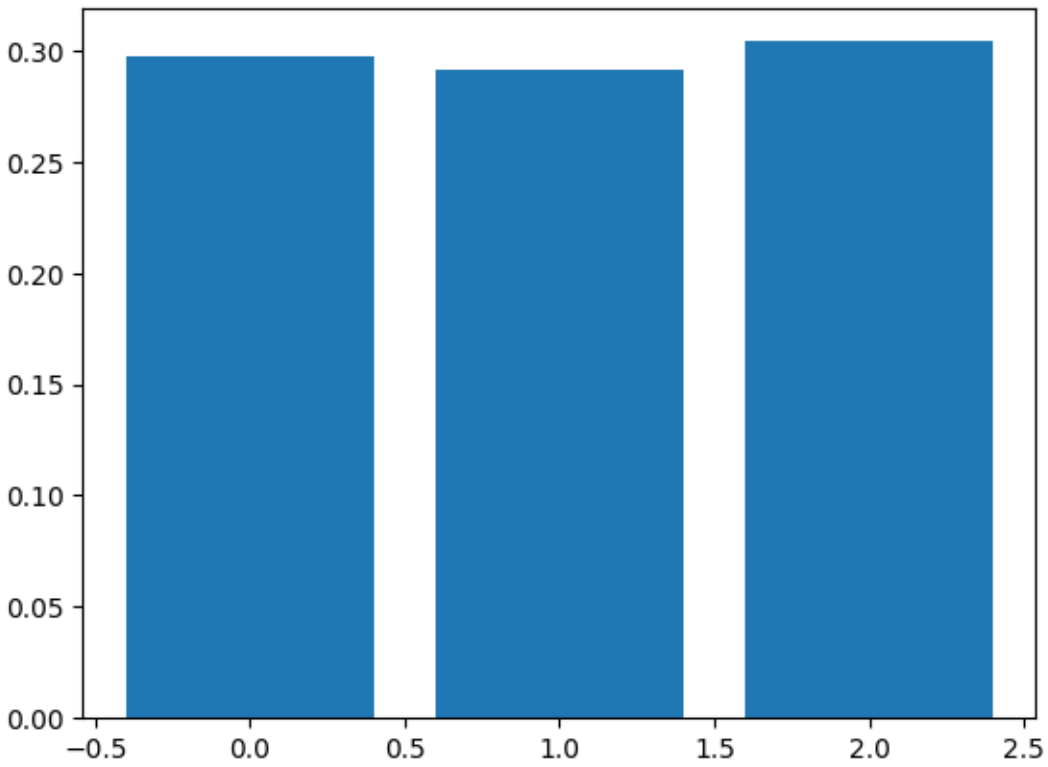


[27]:
```
shots = 1000
job = execute(test, simulator, shots=shots)
result = job.result()
counts = result.get_counts(test)

ones = [0 for i in range(N)]
for k in counts.keys():
    for i in range(len(k)):
        if k[i] == '1':
            ones[i] += counts[k]

ones = [ones[i]/shots for i in range(N)]

# display a plot of bar of height ones[i] for each i
plt.bar(range(N), ones)
plt.show()
```



1.2 De même après 10 Δt

[28]:
```
time_it = 10

test = QuantumCircuit(N)

for i in range(time_it):
    test.append(ope_gate, range(N))

test.measure_all()

shots = 1000
job = execute(test, simulator, shots=shots)
result = job.result()
counts = result.get_counts(test)

ones = [0 for i in range(N)]
for k in counts.keys():
    for i in range(len(k)):
        if k[i] == '1':
            ones[i] += counts[k]

ones = [ones[i]/shots for i in range(N)]

# display a plot of bar of height ones[i] for each i
plt.bar(range(N), ones)
plt.show()
```



1.3 Évolution de $|101\rangle$ sur 30 Δt

[29]:
```
def ising(iter):
    test = QuantumCircuit(N)

    test.x(0)
    test.x(2)

    for i in range(iter):
        test.append(ope_gate, range(N))

    test.measure_all()

    shots = 1000
    job = execute(test, simulator, shots=shots)
    result = job.result()
    counts = result.get_counts(test)

    ones = [0 for i in range(N)]
    for k in counts.keys():
        for i in range(len(k)):
            if k[i] == '1':
```

```
                ones[i] += counts[k]

    ones = [ones[i]/shots for i in range(N)]
    return ones
```

[41]:
```
steps = 30
evolve = [ising(i) for i in range(steps)]

plt.figure(figsize=(20, 20))
for i in range(steps):
    plt.subplot(int(steps/5), 5, i+1)
    plt.bar(range(N), evolve[i])
    plt.ylim(0, 1)
    plt.title('step ' + str(i+1))
plt.show()
```