Hamiltonian Simulation H=Z

§4.7 Simulation of quantum systems, Nielsen & Chuang

Notebook: Óscar Amaro, July 2021 @ GoLP-EPP
Contact: oscar.amaro@tecnico.ulisboa.pt

Introduction

Quantum circuit for simulating the Hamiltonian H=⊗Z for time Δt.

Get the Quantum Mathematica package at https://homepage.cem.itesm.mx/lgomez/

In[•]:=
```
(* import package *)
Needs["Quantum`Computing`"];
```

n=1

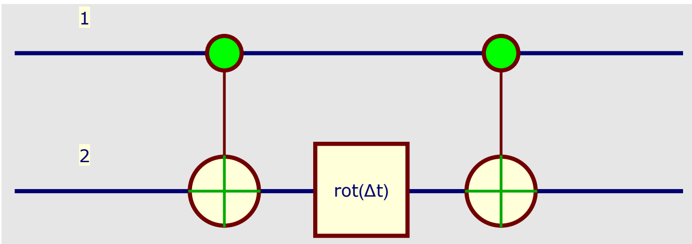
In[∘]:=
```
Clear[Δt, I2, Z, RZ, U1, U2, rot, psi]
I2 = PauliMatrix[0];
Z = PauliMatrix[3];

(* direct method *)
U1 = MatrixExp[-I KroneckerProduct[Z, I2] Δt];

(* define Rz rotation *)
SetQuantumGate[rot, 1,
  Function[{q1},
   Function[{Δt},
    e^(-i Δt) |0_q̂1⟩ · ⟨0_q̂1| + e^(+i Δt) |1_q̂1⟩ · ⟨1_q̂1| ]]];
QuantumTableForm[rot_1̂[Δt]];
QuantumMatrixForm[rot_1̂[Δt]];
QuantumMatrixForm[C^{3̂}[NOT_4̂]];

(* build circuit *)
circ = C^{1̂}[NOT_2̂] · rot_2̂[Δt] · C^{1̂}[NOT_2̂];
(* plot *)
QuantumPlot[circ]
U2 = QuantumMatrix[circ];
(* compare unitaries *)
Print["U1=", U1]
Print["U2=", U2]
(* define test ket. We are only interested on the effect of the operators
 on an arbitrary ket (where the ancilla is supposed to start as 0) *)
psi = ArrayFlatten[KroneckerProduct[{a[0], a[1]}, {1, 0}], 1];
Print["ψ=", psi]
(* the non-zero entries of this ket
  will select the relevant entries of the operators *)
(* because of zero entries in U1,U2,
we need to add ϵ to the operators and then take the limit ϵ→0 *)
Limit[(U1.psi + ϵ) / (U2.psi + ϵ), ϵ → 0]
(* the resulting kets are the same *)
```

Out[∘]=



```
U1={{e^{-i Δt}, 0, 0, 0}, {0, e^{-i Δt}, 0, 0}, {0, 0, e^{i Δt}, 0}, {0, 0, 0, e^{i Δt}}}
U2={{e^{-i Δt}, 0, 0, 0}, {0, e^{i Δt}, 0, 0}, {0, 0, e^{i Δt}, 0}, {0, 0, 0, e^{-i Δt}}}
ψ={a[0], 0, a[1], 0}
```

Out[•]= {1, 1, 1, 1}

n=2

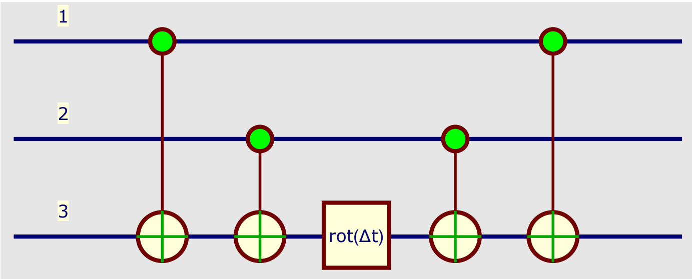
```
Clear[Δt, I2, Z, RZ, U1, U2, rot, psi]
I2 = PauliMatrix[0];
Z = PauliMatrix[3];

(* direct method *)
U1 = MatrixExp[-I KroneckerProduct[Z, Z, I2] Δt];

(* define Rz rotation *)
SetQuantumGate[rot, 1,
  Function[{q1},
   Function[{Δt},
    e^{-i Δt} |0_q1̂⟩ · ⟨0_q1̂| + e^{+i Δt} |1_q1̂⟩ · ⟨1_q1̂|]]];
QuantumTableForm[rot_1̂[Δt]];
QuantumMatrixForm[rot_1̂[Δt]];
QuantumMatrixForm[C^{3̂}[NOT_4̂]];

(* build circuit *)
circ = C^{1̂}[NOT_3̂] · C^{2̂}[NOT_3̂] · rot_3̂[Δt] · C^{2̂}[NOT_3̂] · C^{1̂}[NOT_3̂];
(* plot *)
QuantumPlot[circ]
U2 = QuantumMatrix[circ];
(* compare unitaries *)
psi = ArrayFlatten[KroneckerProduct[{a[0], a[1]}, {a[2], a[3]}, {1, 0}], 1];
Limit[(U1.psi + ϵ) / (U2.psi + ϵ), ϵ → 0]
```

Out[•]=



Out[•]= {1, 1, 1, 1, 1, 1, 1, 1}

n=3

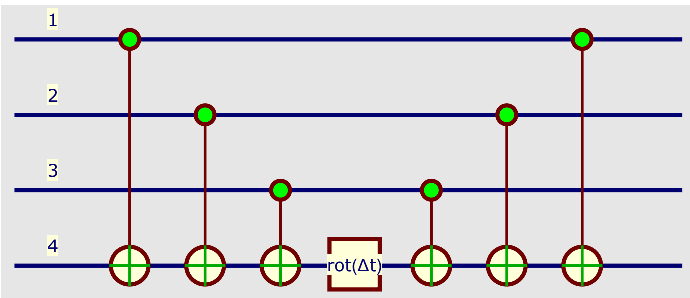
In[•]:=
```
Clear[Δt, I2, Z, RZ, U1, U2, psi]
I2 = PauliMatrix[0];
Z = PauliMatrix[3];

(* direct method *)
U1 = MatrixExp[-I KroneckerProduct[Z, Z, Z, I2] Δt];

(* define Rz rotation *)
SetQuantumGate[RZ, 1,
  Function[{q1},
   Function[{Δt},
    e^(-i Δt) |0_q̂1⟩ · ⟨0_q̂1| + e^(+i Δt) |1_q̂1⟩ · ⟨1_q̂1| ]]];
QuantumTableForm[rot_1̂[Δt]];
QuantumMatrixForm[rot_1̂[Δt]];
QuantumMatrixForm[C^{3̂}[NOT_4̂]];

(* build circuit *)
circ =
  C^{1̂}[NOT_4̂] · C^{2̂}[NOT_4̂] · C^{3̂}[NOT_4̂] · rot_4̂[Δt] · C^{3̂}[NOT_4̂] · C^{2̂}[NOT_4̂] · C^{1̂}[NOT_4̂];

(* plot *)
QuantumPlot[circ]
U2 = QuantumMatrix[circ];
(* compare unitaries *)
psi = ArrayFlatten[
   KroneckerProduct[{a[0], a[1]}, {a[2], a[3]}, {a[4], a[5]}, {1, 0}], 1];
Limit[(U1.psi + ϵ) / (U2.psi + ϵ), ϵ → 0]
```

Out[•]=


Out[•]= {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}