An Intro to HOQST - Lindblad equation

Huo Chen

April 18, 2021

This tutorial demonstrates how to solve the time-independent Lindblad equation using HOQST.

0.1 Model setup

We consider the Lindblad equation of the following form:

$$\dot{\rho} = -i[H, \rho] + \sum_i \gamma_i \left(L_i \rho L_i^\dagger - \frac{1}{2} \left\{ L_i^\dagger L_i, \rho \right\} \right) .$$

In this example, we choose a constant Hamiltonian

$$H(s) = \sigma_z \ ,$$

a single Lindblad operator $L = \sigma_z$ and a single rate γ. The entire evolution can be defined by:

```
using OpenQuantumTools, OrdinaryDiffEq, Plots
# define the Hamiltonian
H = DenseHamiltonian([(s)->1.0], [σz], unit=:ħ)
# define the initial state
u0 = PauliVec[1][1]*PauliVec[1][1]'
# define the Lindblad operator
# the rate and Lindblad operator can also be time-dependent functions
lind = Lindblad(0.1, σz)
# combine them into an Annealing object
annealing = Annealing(H, u0, interactions = InteractionSet(lind))
```

```
Annealing with OpenQuantumBase.DenseHamiltonian{ComplexF64} and u0 Matrix{C
omplexF64}
u0 size: (2, 2)
```

0.2 Dynamics

The solution of the Lindblad ME can be obtained by calling `solve_lindblad`:

```
# define total annealing/evolution time
tf = 10
# solve the Lindblad equation
sol = solve_lindblad(annealing, 10, alg=Tsit5());
```

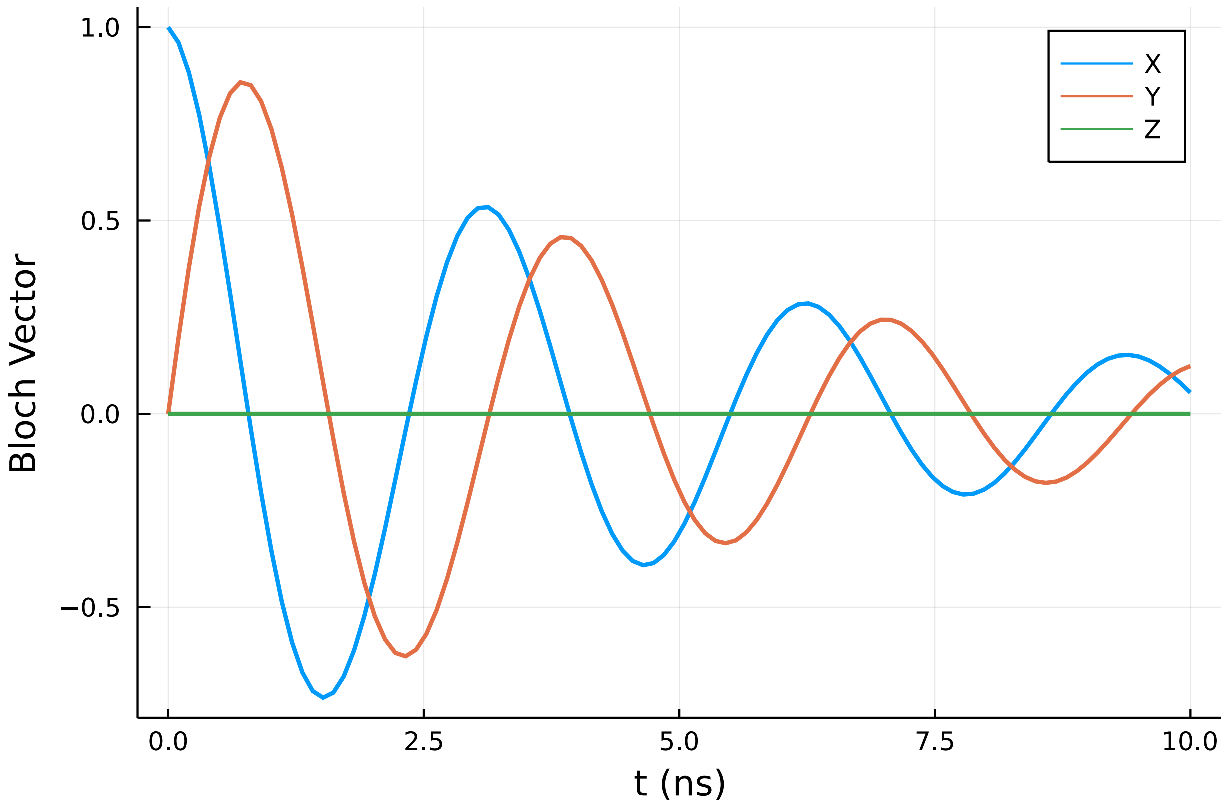
In the following code blocks, we show how to extract useful quantities like the Bloch vector or density matrix elements from the simulation results:

```
t_axis = range(0, 10, length=100)
bloch_vector = []
for t in t_axis
    # matrix_decompose projects a matrix onto a list of basis elements
    push!(bloch_vector, 2*real.(matrix_decompose(sol(t), [σx, σy, σz])))
end

off_diag = []
for t in t_axis
    push!(off_diag, abs(sol(t)[1,2]))
end
```

We first plot the Bloch vector representation of the qubit along the evolution:

```
plot(t_axis, [c[1] for c in bloch_vector], label="X", linewidth=2)
plot!(t_axis, [c[2] for c in bloch_vector], label="Y", linewidth=2)
plot!(t_axis, [c[3] for c in bloch_vector], label="Z", linewidth=2)
xlabel!("t (ns)")
ylabel!("Bloch Vector")
```



Then, we plot the absolute value of the off-diagonal element $|\rho_{01}|$ and compare it with the analytical solution:

```
plot(t_axis, off_diag, linewidth=2, label="ME")
plot!(t_axis, 0.5*exp.(-0.2*t_axis), linestyle=:dash, linewidth=3, label="Analytical")
xlabel!("t (ns)")
ylabel!("|ρ_0_1(t)|")
```