Classical stochastic noise – the spin-fluctuator model

Huo Chen

November 22, 2020

0.1 Spin-fluctuator model (Classical 1/f noise)

The total Hamiltonian in this example is

$$H(s) = -Z + \frac{1}{2}\sum_i n_i(s) Z \ ,$$

where $n_i(s)$ is the telegraph process that switches randomly between $\pm b_i$ with a rate γ_i. The summation $\sum_i n_i(s)$ generates the "1/f" noise approximately.

We choose the initial state to be

$$|\phi(0)\rangle = |+\rangle \ .$$

0.1.1 EnsembleFluctuator

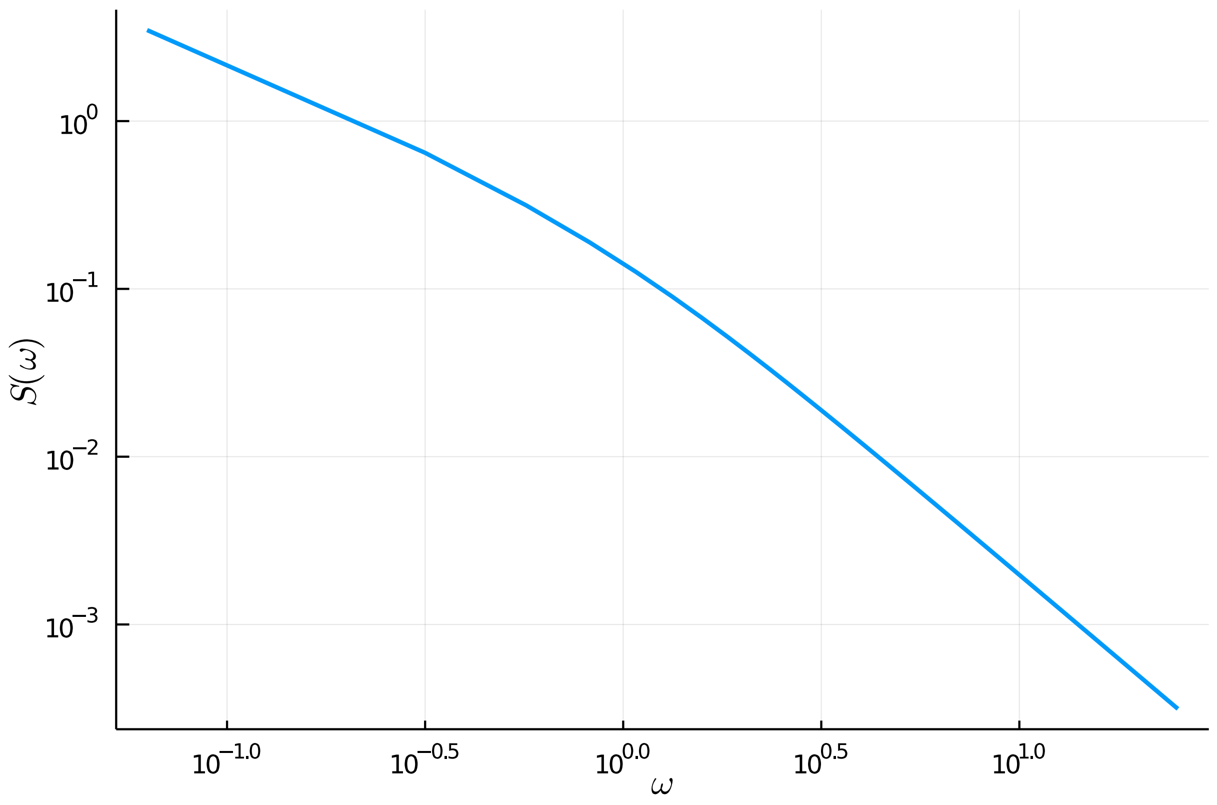
We first construct the fluctuator bath object using `EnsembleFluctuator` and plot its spectral density:

```
using OpenQuantumTools, LaTeXStrings
using OrdinaryDiffEq, Plots, StatsBase

# calculate the mean and standard deviation of mean estimator from a sample
function mean_std(sample)
    m, v = mean_and_std(sample, corrected=true)
    m, v/sqrt(length(sample))
end

# All the value created in the following codes is in angular frquencies unit
num = 10
bvec = 0.2 * ones(num)
γvec = log_uniform(0.01, 1, num)
fluctuator_ensemble = EnsembleFluctuator(bvec, γvec);

plot(fluctuator_ensemble, :spectrum, 2*π*range(0.01, 4, length=100), xscale=:log10,
yscale=:log10, linewidth=2, label="")
xlabel!(L"\omega")
ylabel!(L"S(\omega)")
```



0.1.2 Free Evolution

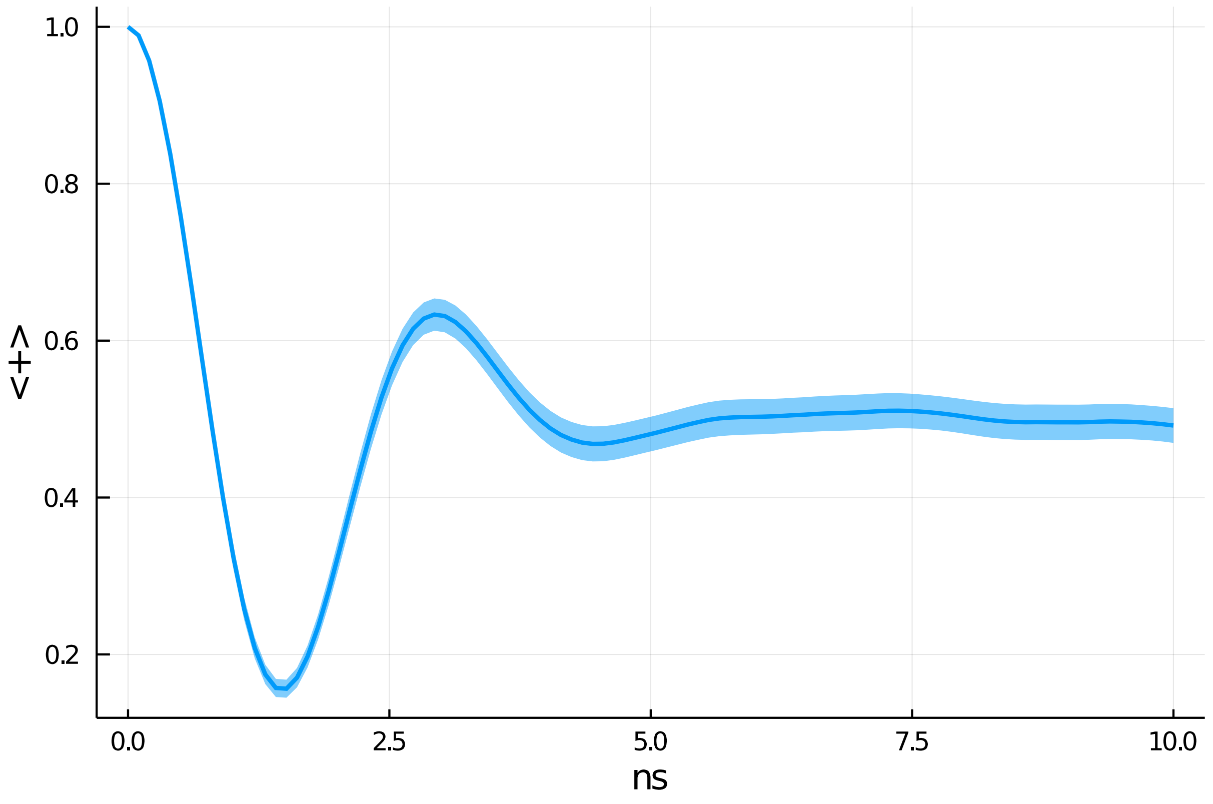
We then calculate the dynamics of the free evolution:

```
H = DenseHamiltonian([(s)->1], [σz], unit=:ħ)
coupling = ConstantCouplings([0.5*σz], unit=:ħ)
u0 = PauliVec[1][1]
annealing = Annealing(H, u0, coupling = coupling, bath=fluctuator_ensemble)
tf = 10
# create object for parallel simulation
prob = build_ensembles(annealing, tf, :stochastic)
# we run each trajectories in serial for this example
sol = solve(prob, Tsit5(), EnsembleSerial(), trajectories=1000, reltol=1e-6,
saveat=range(0,tf,length=100))
```

After the solution is obtained, we plot $\langle + \rangle$ during the evolution:

```
t_axis = range(0,tf,length=100)
es = []
err = []
for (i, s) in enumerate(t_axis)
    sample = [abs2(u0'*so[i]) for so in sol]
    pop, pop_std = mean_std(sample)
    push!(es, pop)
    push!(err, 2*pop_std)
end

plot(t_axis, es, ribbon=err, linewidth=2, label="")
xlabel!("ns")
ylabel!("<+>")
```



0.1.3 Pulses in the middle

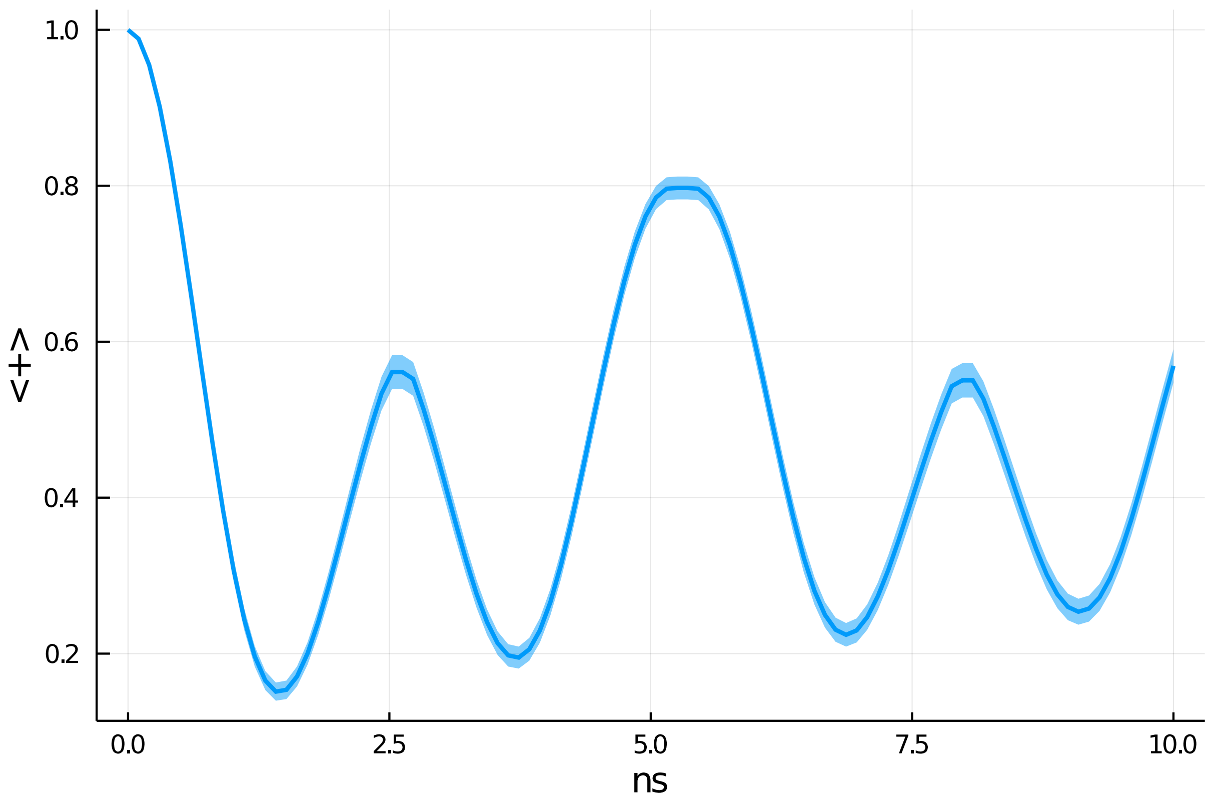
We can also apply instantaneous pulses during the middle of the evolution using `InstPulseControl`. In the following example, we apply X pulses at $s = 0.25, 0.5$ and 0.75.

```
cb = InstPulseCallback([0.25, 0.5, 0.75] .* tf, (c, x) -> c .= σx * c)
sol = solve(prob, Tsit5(), EnsembleSerial(), trajectories=1000, reltol=1e-6,
saveat=range(0,tf,length=100), callback=cb)
```

Again, we plot the $\langle + \rangle$ w.r.t the evolution time:

```
s_axis = range(0,1,length=100)
es = []
err = []
for (i, s) in enumerate(s_axis)
    sample = [abs2(u0'*so[i]) for so in sol]
    pop, pop_std = mean_std(sample)
    push!(es, pop)
    push!(err, 2*pop_std)
end

plot(tf*s_axis, es, ribbon=err, linewidth=2, label="")
xlabel!("ns")
ylabel!("<+>")
```



1.0
0.8
0.6
0.4
0.2
<+>
0.0
2.5
5.0
7.5
10.0
ns