An Intro to the coarse-grained ME and universal Lindblad ME

Huo Chen

November 19, 2020

0.1 Model

In this tutorial, we consider a standard single qubit annealing Hamiltonian

$$H(s) = -\frac{1}{2}(1-s)\sigma_x - \frac{1}{2}s\sigma_z$$

coupling to an Ohmic bath via σ_z operator. We solve the open system dynamics via three different MEs: Redfield equation, coarse-grained ME(CGME), and universal Lindblad equation(ULE). Unlike the Redfield equation, CGME and ULE generate CP maps.

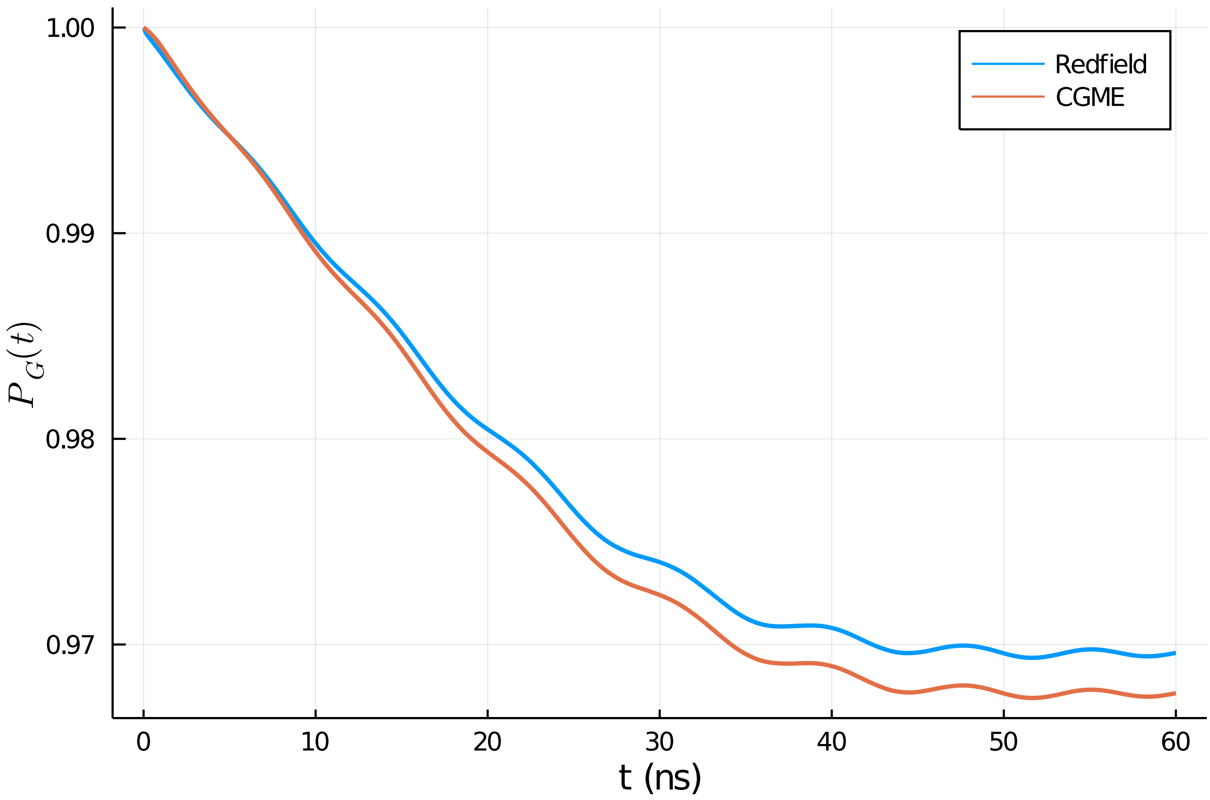
0.2 Coarse-grained ME

Coarse-grained ME is a completely positive ME obtained by applying an additional time coarse-graining approximate to the Redfield equation. More details of CGME can be found in Mozgunov and Lidar. We first solve the original Redfield equation and CGME and compare both cases' instantaneous ground state population.

```
using OrdinaryDiffEq, Plots, LaTeXStrings
using OpenQuantumTools

# Hamiltonian
H = DenseHamiltonian([(s)->1-s, (s)->s], -[σx, σz]/2, unit=:ħ)
# initial state
u0 = PauliVec[1][1]
# coupling
coupling = ConstantCouplings(["Z"], unit=:ħ)
# bath
bath = Ohmic(1e-4, 4, 16)
annealing = Annealing(H, u0; coupling=coupling, bath=bath)

tf = 60
U = solve_unitary(annealing, tf, alg=Tsit5(), abstol=1e-8, reltol=1e-8)
U = InplaceUnitary(U)

@time solr = solve_redfield(annealing, tf, U, alg=Tsit5())
# we set the integration error tolerance to 1e-5 for speed
@time solc = solve_cgme(annealing, tf, U, alg=Tsit5(), int_atol=1e-5, int_rtol=1e-5)
plot(solr, H, [0], 0:0.01:tf, linewidth=2, xlabel="t (ns)", ylabel="\$P_G(t)\$",
label="Redfield")
```

```julia
plot!(solc, H, [0], 0:0.01:tf, linewidth=2, label="CGME")
```

```
0.290015 seconds (3.08 M allocations: 80.058 MiB)
 51.352793 seconds (489.73 M allocations: 19.392 GiB, 5.07% gc time)
```



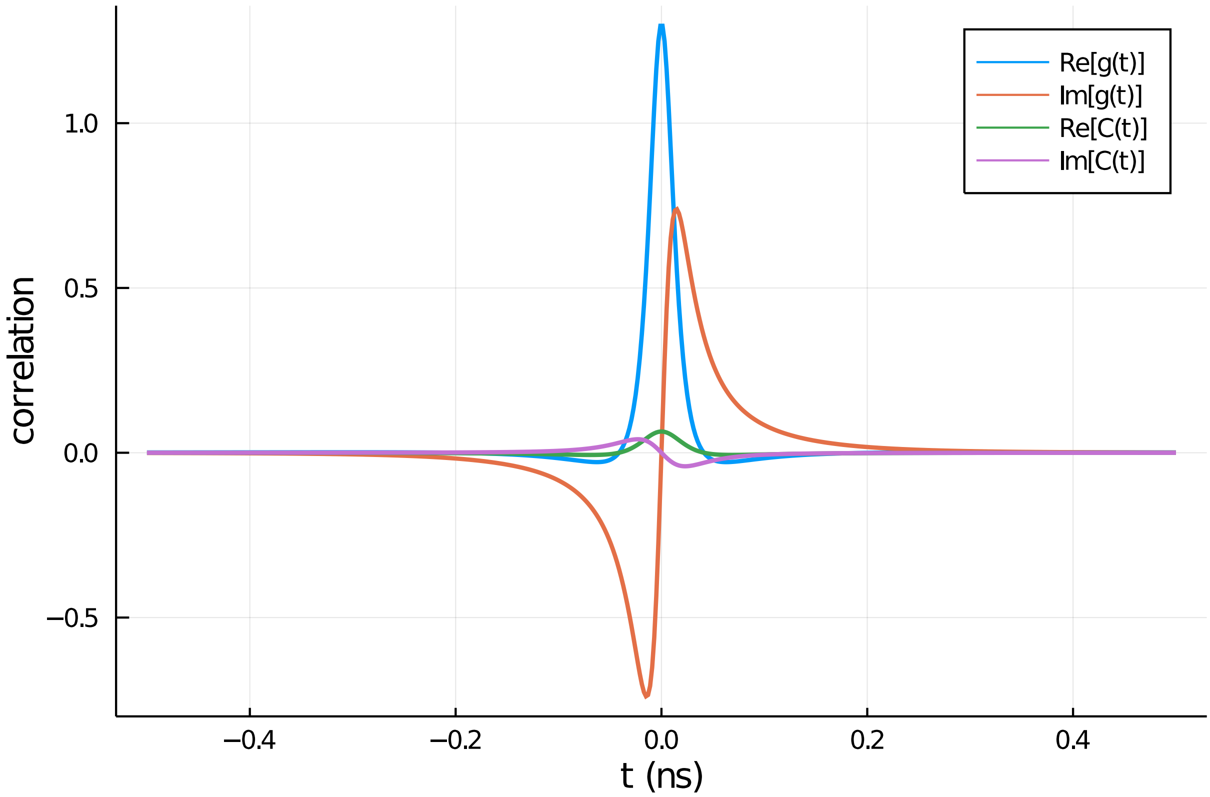
0.3 Universal Lindblad equation

Universal Lindblad equation(ULE) is a different CP ME proposed in Nathan and Rudner. Unlike the Redfield and CGME, it depends on the jump correlator, which is the inverse Fourier transform of the square root of the noise spectrum:

$$g(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \sqrt{\gamma(\omega)} e^{i\omega t} \mathrm{d}\omega \ .$$

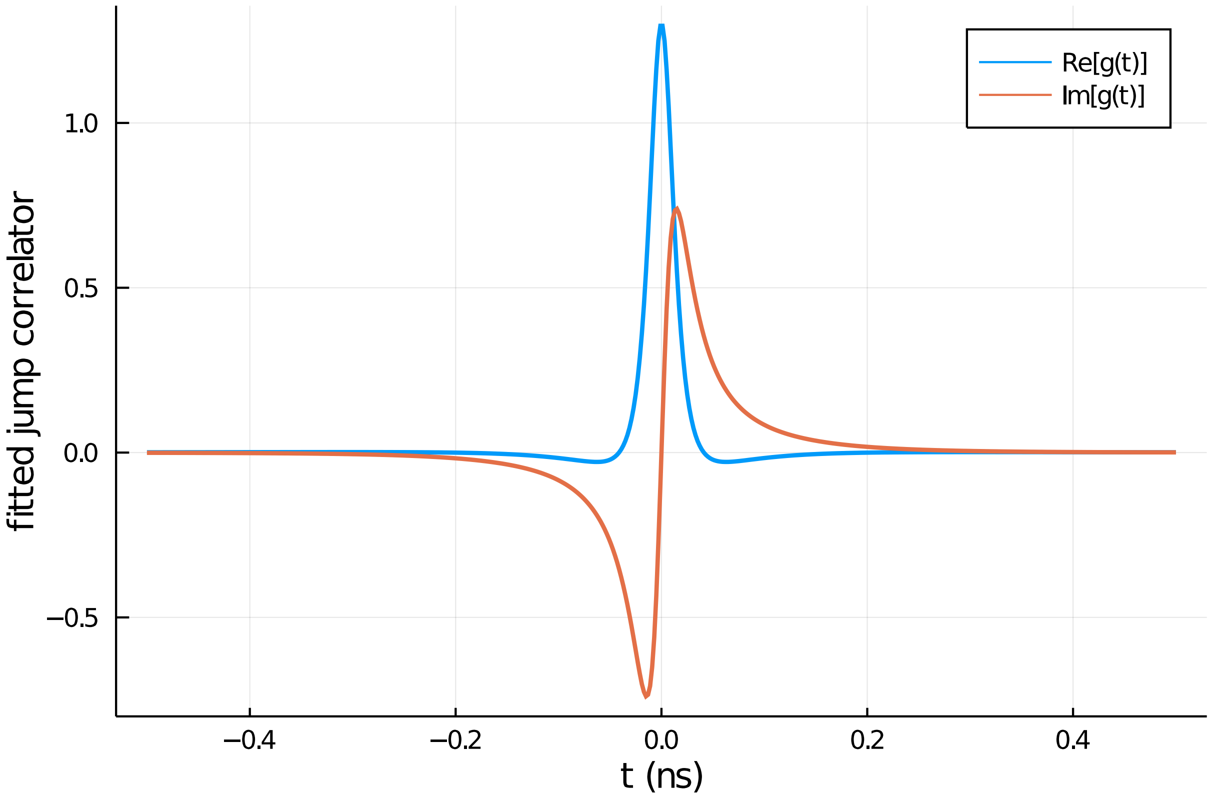
Let's first see how it looks compared with the two-point correlation function $C(t)$:

```julia
using QuadGK

g(t) = quadgk((w)->sqrt(γ(w, bath))*exp(1.0im*w*t)/2/π, -Inf, Inf)[1]

t = range(-0.5,0.5,length=500)
g_value = g.(t)
c_value = [correlation(x, bath) for x in t];

plot(t, real.(g_value), label="Re[g(t)]", linewidth=2)
plot!(t, imag.(g_value), label="Im[g(t)]", linewidth=2)
plot!(t, real.(c_value), label="Re[C(t)]", linewidth=2)
plot!(t, imag.(c_value), label="Im[C(t)]", linewidth=2)
xlabel!("t (ns)")
ylabel!("correlation")
```



From the above picture, we can see that the jump correlator and two-point correlation function roughly have the same time scale. To avoid recalculating the inverse Fourier transform within the solver, we can precalculate $g(t)$ and construct interpolation from these pre-computed values. This procedure can be done by the following code block:

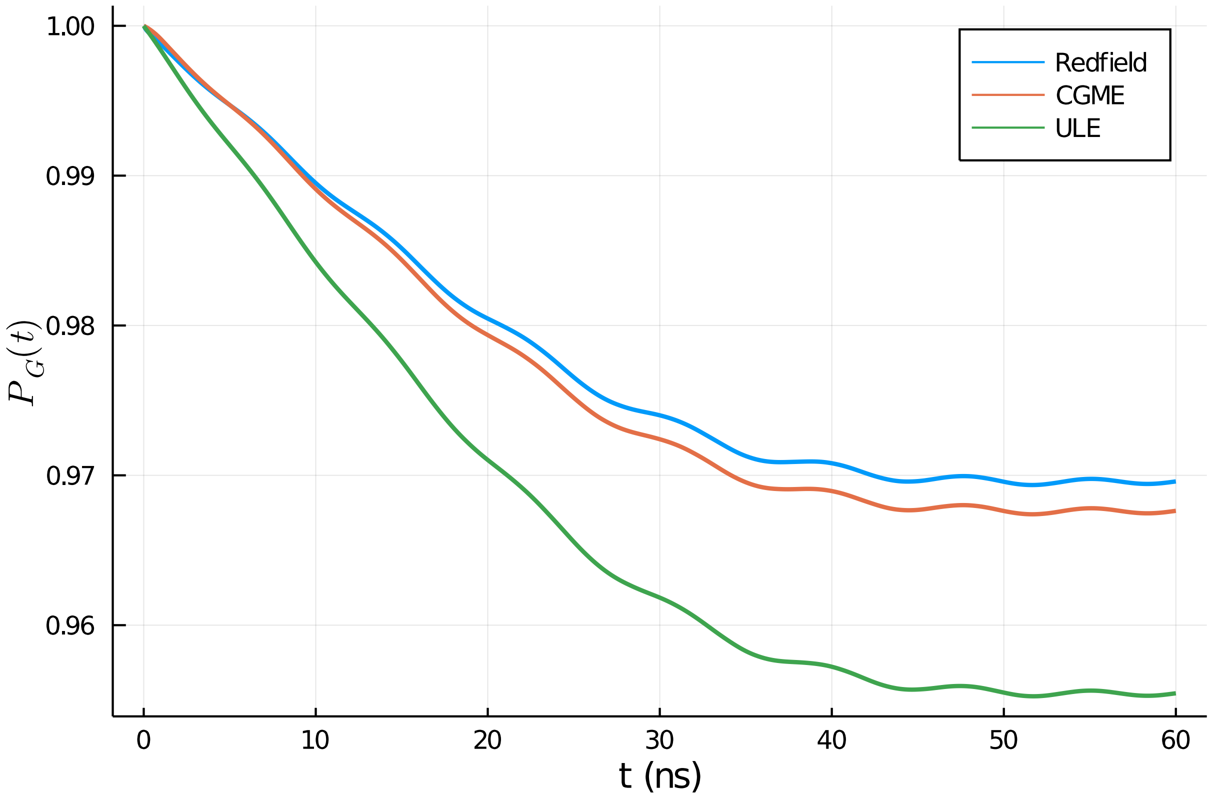
```
t = range(-4,4,length=2000)
g_value = g.(t)
gf = construct_interpolations(t, g_value, extrapolation = "flat")

t = range(-0.5,0.5,length=500)
g_value = gf.(t)
plot(t, real.(g_value), label="Re[g(t)]", linewidth=2)
plot!(t, imag.(g_value), label="Im[g(t)]", linewidth=2)
xlabel!("t (ns)")
ylabel!("fitted jump correlator")
```



Finally we solve ULE and compare the result with the Redfield equation and CGME:

```
ubath = ULEBath(gf)
annealing = Annealing(H, u0; coupling=coupling, bath=ubath)
@time solu = solve_ule(annealing, tf, U, alg=Tsit5(), int_atol=1e-5, int_rtol=1e-5)
plot(solr, H, [0], 0:0.01:tf, linewidth=2, xlabel="t (ns)", ylabel="\$P_G(t)\$",
label="Redfield")
plot!(solc, H, [0], 0:0.01:tf, linewidth=2, label="CGME")
plot!(solu, H, [0], 0:0.01:tf, linewidth=2, label="ULE")
```

```
0.506539 seconds (5.09 M allocations: 133.383 MiB, 2.78% gc time)
```


$P_G(t)$
1.00
0.99
0.98
0.97
0.96
0
10
20
30
40
50
60
t (ns)
Redfield
CGME
ULE