

```
In [7]: # Preamble: minimal imports + local include
using Random
using Statistics
using Printf
using Plots
using Logging

include("formatting.jl")

if !isdefined(Main, :System1D)
    include("../src/System1D.jl")
end
using .System1D: Hamiltonian, TrialWF, Walker, DMCPParams, DMCSim, Importance
```

## System: Fixed-Node DMC for 1D Harmonic Oscillator (Odd Parity)

### Potential

$$V(x) = \frac{1}{2}\omega^2 x^2$$

### Trial wavefunction (odd parity)

$$\psi_T(x) = x e^{-\alpha x^2/2}$$

The fixed node at  $x = 0$  enforces odd parity. This cell defines the Hamiltonian and trial wavefunction, initializes walkers, and runs fixed-node DMC.

```
In [8]: # System: fixed-node DMC for a single particle in a 1D HO (odd parity)
omega = 1.0
V(R) = 0.5 * omega^2 * R[1]^2
H = Hamiltonian(1, 0.5, V)

alpha = omega
logpsi(R) = begin
    x = R[1]
    ax = abs(x)
    return ax == 0 ? -Inf : log(ax) - 0.5 * alpha * x^2
end

gradlogpsi(R) = begin
    x = R[1]
    ax = abs(x)
    s = sign(x)
    return ax == 0 ? [0.0] : [(s / ax) - alpha * x]
end

lapllogpsi(R) = begin
    x = R[1]
    ax = abs(x)
```

```

    return ax == 0 ? -Inf : -1.0 / ax^2 - alpha
end

signpsi(R; tol=1e-12) = abs(R[1]) < tol ? 0.0 : sign(R[1])

trial = TrialWF(logpsi, gradlogpsi, lapllogpsi, signpsi)
guiding = ImportanceGuiding(trial, H)

# Walkers and params
targetN = 1000
walkers = [Walker([randn()]) for _ in 1:targetN]

E_ref = 1.5 * omega
params = DMCPParams(0.005, 400, 50, targetN, E_ref, 0.1, 10, 50)

# Run
rng_sim = MersenneTwister(7)
sim = DMCSim(H, params, walkers, rng_sim; guiding=guiding, nodepolicy=FixedN
run_simulation!(sim; snapshot_steps=[params.nsteps])

ET_history = sim.ET_history
population_history = sim.population_history
energy_mean_history = sim.energy_mean_history
energy_variance_history = sim.energy_variance_history

nothing

```

## Results

### Exact odd-parity (first excited) energy

$$E_1 = \frac{3}{2}\omega$$

This cell reports the post-equilibration energy and plots the reference energy, population, mean local energy, and local-energy variance over time, plus the final walker density.

```

In [10]: # Results and plots
t = (0:params.nsteps) .* params.dt

nequil = params.nequil
start_idx = min(nequil + 1, length(energy_mean_history))
postE = energy_mean_history[start_idx:end]

Ebar = mean(postE)
SEM = (length(postE) > 1) ? std(postE) / sqrt(length(postE)) : NaN

Ebar_str, SEM_str = format_with_uncertainty(Ebar, SEM)
println("Fixed-node H0 (odd parity) DMC energy (post-eq, nequil=$(nequil)):
E_final_str, var_str = format_pair_sigfig(energy_mean_history[end], energy_v
println("Final step: E = $(E_final_str), Var(E_L) = $(var_str)")
println("Reference energy: $(format_sigfig(E_ref; sigfigs=5))")

```

```

Logging.with_logger(Logging.NullLogger()) do
  redirect_stderr(devnull) do
    p1 = plot(t, ET_history, xlabel="step", ylabel="E_T", title="Referer
    display(p1)

    p2 = plot(t, population_history, xlabel="step", ylabel="N_w", title=
    display(p2)

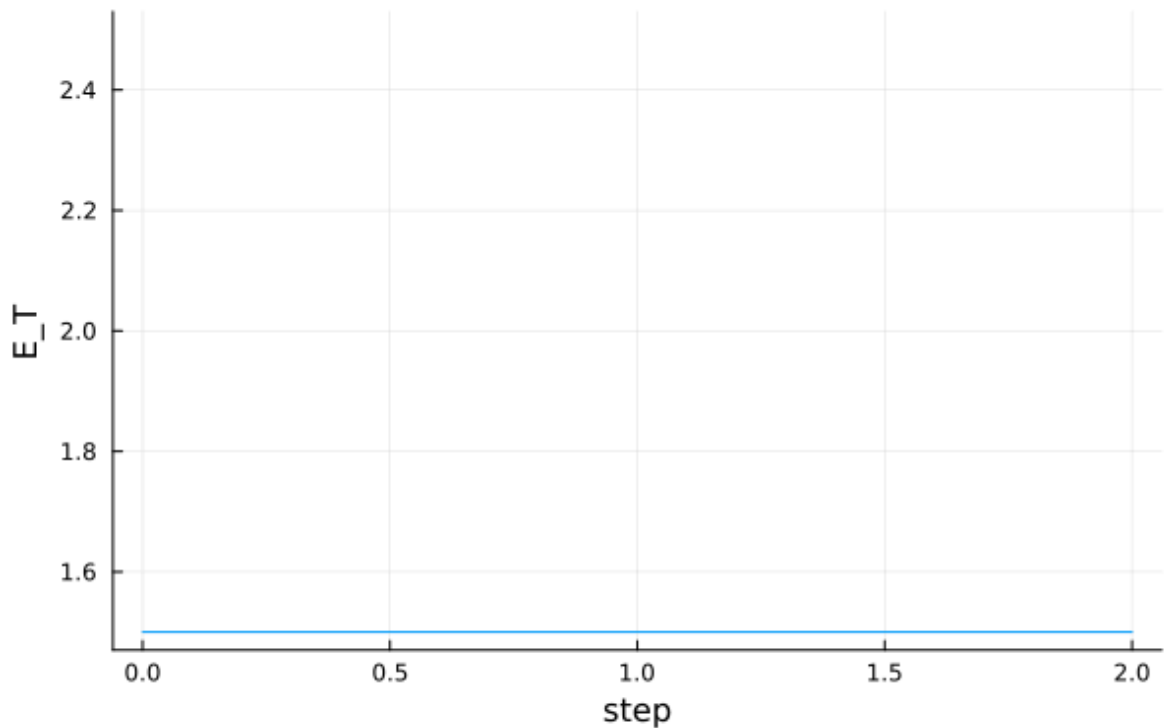
    p3 = plot(t, energy_mean_history, xlabel="step", ylabel="⟨E_L⟩", tit
    plot!(p3, [t[1], t[end]], [E_ref, E_ref], ls=:dash, label="ref")
    display(p3)

    p4 = plot(t, energy_variance_history, xlabel="step", ylabel="Var(E_L
    display(p4)

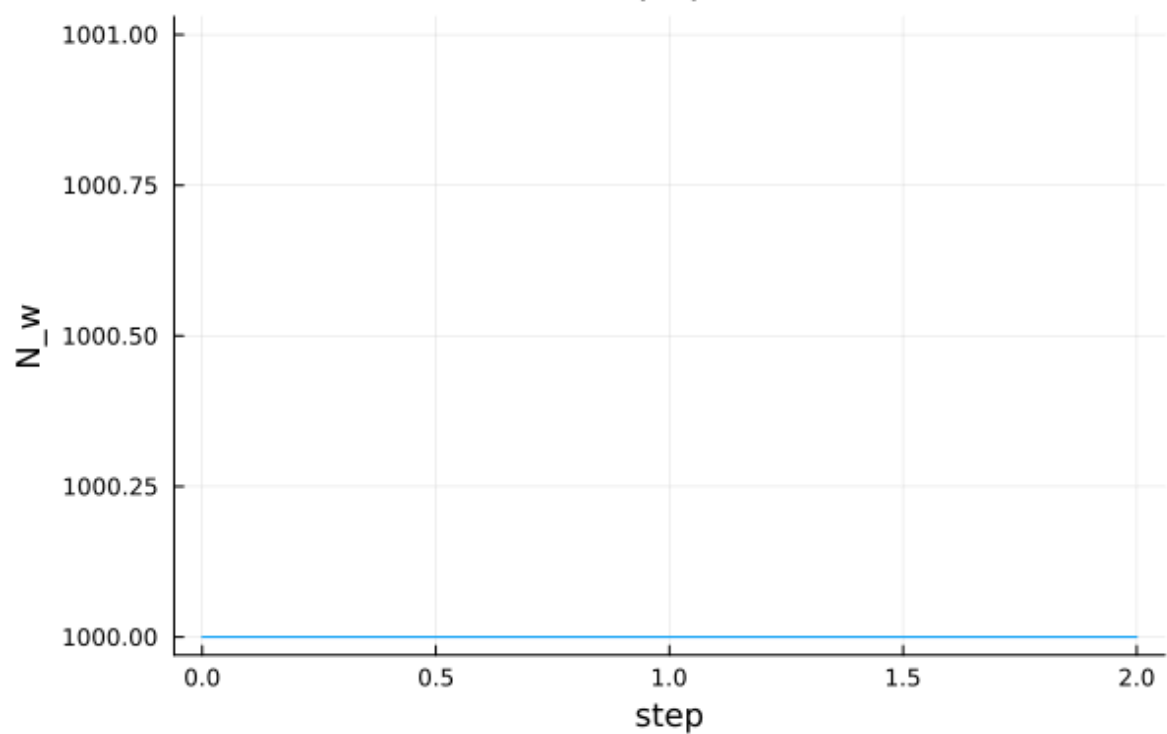
    if !isempty(sim.walker_positions_history)
      snap = sim.walker_positions_history[end]
      p5 = plot_snapshot_1d_density(snap; nbins=120, title="Final walk
      display(p5)
    end
  end
end

```

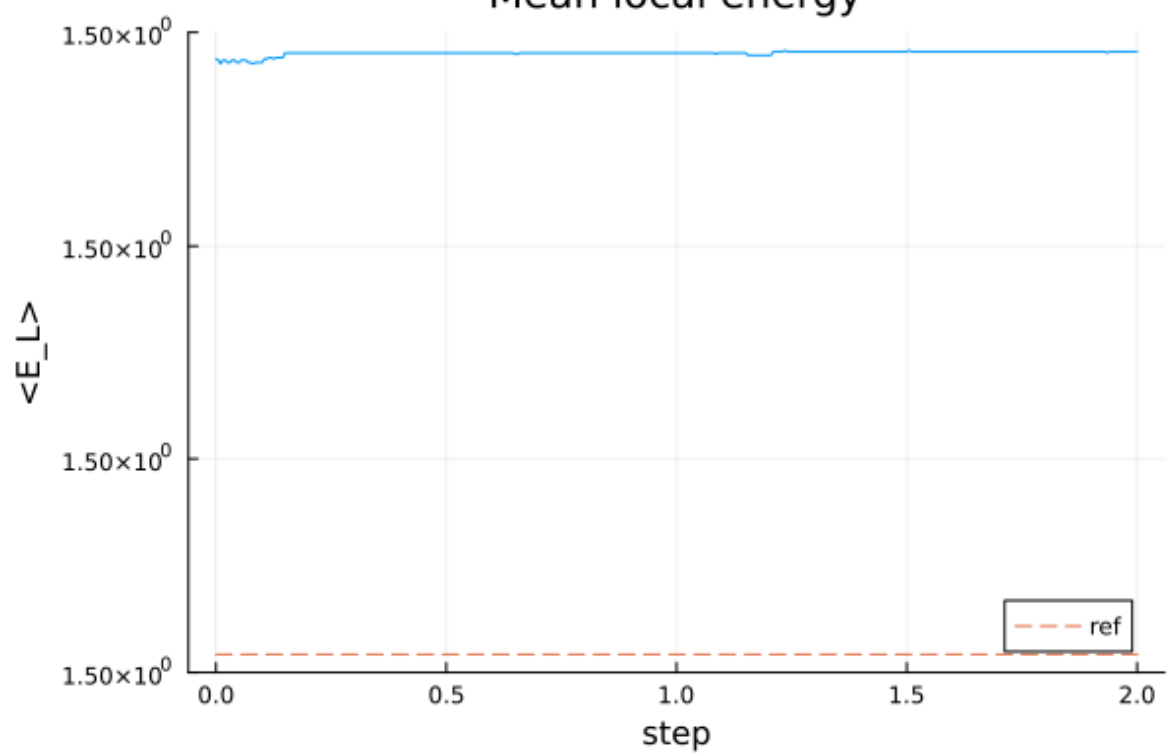
Reference energy

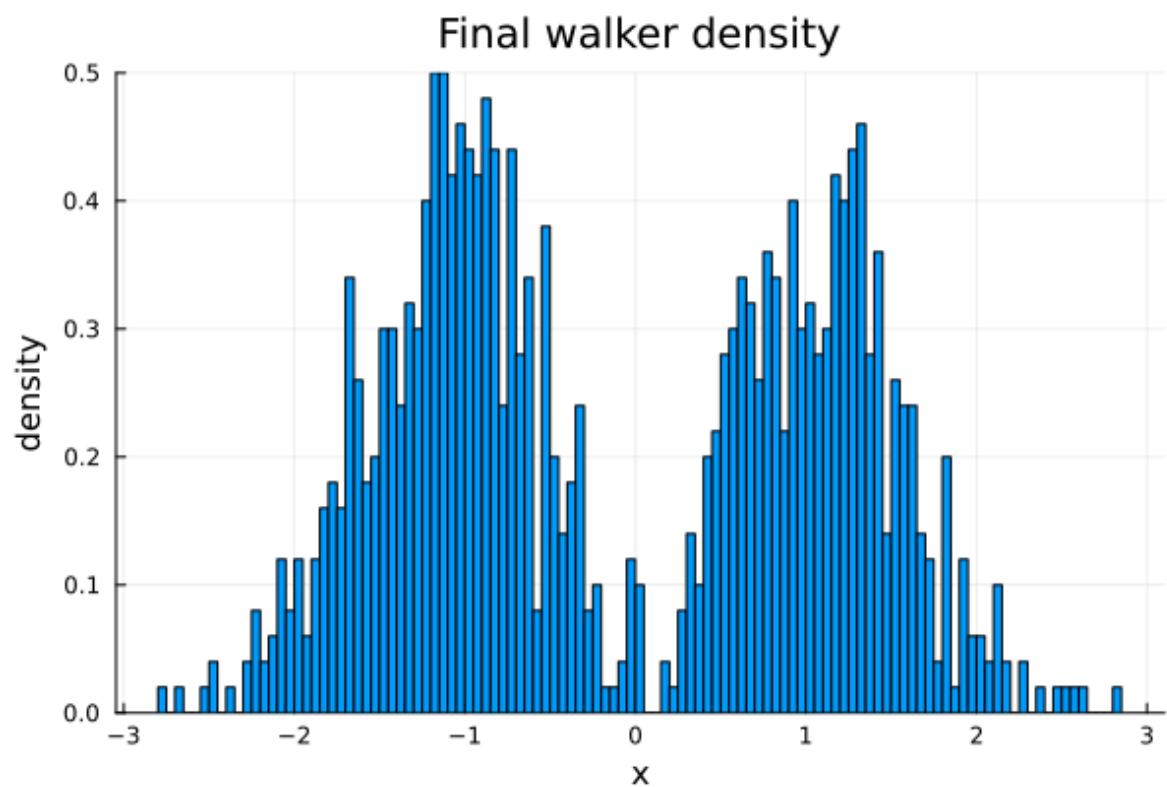
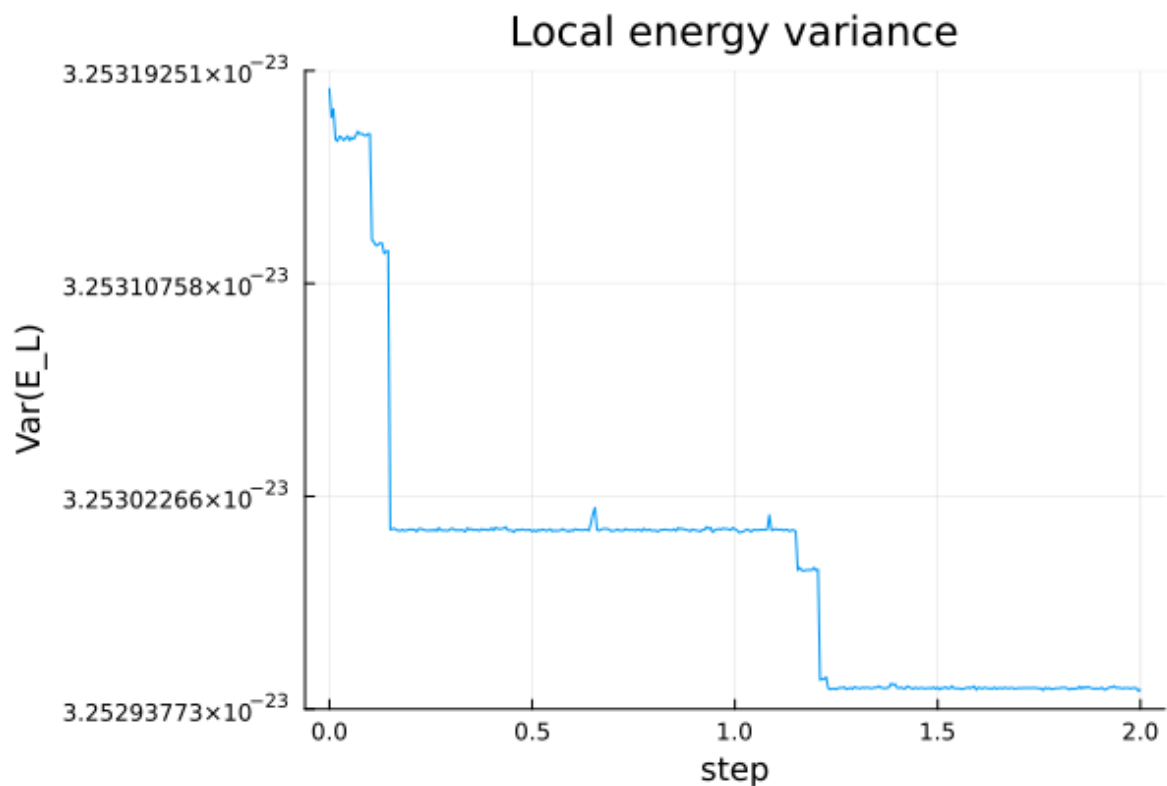


Walker population



Mean local energy





Fixed-node H0 (odd parity) DMC energy (post-eq, nequil=50):  $E = 1.50000000000011191 \pm 0.00000000000000008$   
 Final step:  $E = 1.500000000000110800257858$ ,  $\text{Var}(E_L) = 0.000000000000000000000033$   
 Reference energy: 1.5000