

Links are disabled because this is a static copy of a profile report







run_all (Calls: 1, Time: 66.486 s)

Generated 15-Jul-2024 19:40:42 using performance time.

script in file D:\Aalto\2324\BScThesis\FullRepo\parallelsimulations_finitebath\src\run_all.m

Copy to new window for comparing multiple runs

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
102	[vel, e1] = diagonal (H);	25	23.551 s	35.4%	
104	nau = GGE (N, vel);	25	22.203 s	33.4%	
103	E1 = time_evolution (N, hbar, ...	25	17.378 s	26.1%	
101	H = total_hamiltonian (N,w,mul...	25	2.294 s	3.5%	
132	legend([a1(1), a2(1), a3(1)], ...	1	0.588 s	0.9%	
All other lines			0.472 s	0.7%	
Totals			66.486 s	100%	

Function listing

```
time    Calls    line
        6 clearvars
        7 close all
        8 clc
        9
       10 % Enable long format for higher accuracy in the calculations
       11 format long
       12
       13 % Initialize the random number generator based on the current time
       14 rng("shuffle");
       15
       16 % Define parallelisation type. Accepted values are 'modular', 'GPU',
       17 % 'multicore'.
       18 type = 'GPU';
       19
       20 % Add the folders of the parallelisation in the path
       21 addpath(fullfile(pwd, type));
       22
       23 % Begin timing
       24 tic;
       25 profile on
       26
       27 % Defining example variables of the problem
       28
       29 % The total number of two level systems (TLSs) in the bath.
       30 % The intially excited state, the qubit, is not considered to be
       31 % part of the bath. Therefore N+1 is the overall number of TLSs
```

```

< 0.001      1      32 N = 1500;
               33
               34 % Number of independent, random iterations
< 0.001      1      35 Nr = 25;
               36
               37 % The frequency of the qubit.
               38 % Take it normalized to 1 for simpler calculations
< 0.001      1      39 w = 1;
               40
               41 % The reduced Planck's constant.
               42 % Take it normalized to 1 for simpler calculations
< 0.001      1      43 hbar = 1;
               44
               45 % A flag that indicates the consideration of internal
               46 % couplings of the TLSs in the bath. Use 0 for no
               47 % internal coupling, 1 to include internal coupling
< 0.001      1      48 mutual = 1;
               49
               50 % Sets the magnitude of the internal coupling strength.
               51 % Taken to be w/(5*sqrt(2)) in the example case.
               52 % For weak coupling regime, smaller of the frequency of the qubit,
               53 % but is it enough small? Physical explanation for the choosen value?
< 0.001      1      54 gamma = w/(5*sqrt(2));
               55
               56 % The final time at which the populations are calculated.
< 0.001      1      57 tmax = 8000000000;
               58
               59 % Construct a N-by-1 column vector with (sorted) uniformly distributed
               60 % random numbers in [0, 2*hbar*w]. It will be the diagonal elements of
               61 % the bath Hamiltonian, representing the energy levels hbar*frequencies
               62 % of the spins of the bath hbar*omega_j where j is in [1, N].
               63 % The energy levels are sorted to reflect the ordered energy spectrum of
               64 % the physical systems.
               65 % This is a constant random vector during the iterations.
< 0.001      1      66 omega_j = sort(2*hbar*w*rand(N,1));
               67
               68 % The initial state of the system, bath in the ground state
               69 % and qubit excited
< 0.001      1      70 rho0 = zeros(N+1);
< 0.001      1      71 rho0(N+1, N+1) = 1;
               72
               73 % The array for collecting the results of long time evolution
< 0.001      1      74 te_results = zeros(N, Nr);
               75
               76 % The array for collecting the results of the GGE prediction
< 0.001      1      77 gge_results = zeros(N+1, Nr);
               78
               79
< 0.001      1      80 if strcmp(type, 'multicore')
               81         % Initiate the parallel poll.
               82         initParPool()
               83         % Initialize the random number generator with the Multiplicative lagged
               84         % Fibonacci generator, for multiple workers in parallel
               85         s = RandStream.create('mlfg6331_64', 'NumStreams', Nr, 'Seed', ...
               86         'shuffle', 'CellOutput', true);
               87         % Iterate Nr times
               88         for idx = 1:Nr
               89             RandStream.setGlobalStream(s{idx});
               90             H = total_hamiltonian (N,w,mutual,gamma, omega_j);
               91             [vel, el] = diagonal (H);
               92             E1 = time_evolution (N, hbar, tmax, vel, el, rho0);
               93             nau = GGE (N, vel);
               94
               95             te_results(:, idx) = E1;
               96             gge_results(:, idx) = nau;

```

```

97     end
< 0.001    1    98 else
99         % Iterrate Nr times
< 0.001    1   100     for idx = 1:Nr
2.294    25   101     H = total_hamiltonian (N,w,mutual,gamma, omega_j);
23.551    25   102     [vel, el] = diagonal (H);
17.378    25   103     E1 = time_evolution (N, hbar, tmax, vel, el, rho0);
22.203    25   104     nau = GGE (N, vel);
105
< 0.001    25   106     te_results(:, idx) = E1;
< 0.001    25   107     gge_results(:, idx) = nau;
< 0.001    25   108     end
< 0.001    1   109 end
110
111 % Get the mean of the iterations
< 0.001    1   112 te_results_mean = sum(te_results, 2) / Nr;
< 0.001    1   113 gge_results_mean = sum(gge_results, 2) / Nr;
114
115 % The analytical GGE prediction for the populations
0.009    1   116 [nl, omega] = analytical (N, w, gamma);
117
118 % Plotting
119 % (i) Numerical long-time evolution
120 % (ii) Numerical GGE
121 % (iii) Analytical
122
0.264    1   123 a1 = semilogy(omega_j, te_results_mean, 'o', "Color", 'b');
0.015    1   124 hold on
0.006    1   125 a2 = plot(omega_j, gge_results(1:N), 'x', "LineWidth", 1.1, "Color","g");
0.004    1   126 a3 = plot(omega, nl, "LineWidth", 1.2, "Color", "r");
127
< 0.001    1   128 out1 = sprintf('Long-time evolution for %d spins with %d iterations', N, Nr);
0.049    1   129 xlabel("\Omega/\Omega$", 'Interpreter',"latex", 'FontSize',18)
0.008    1   130 ylabel("$n$", 'Interpreter',"latex", 'FontSize',18)
0.019    1   131 title(out1);
0.588    1   132 legend([a1(1), a2(1), a3(1)], 'Long-time evolution', 'Numerical GGE', ...
133         'Analytical GGE', 'location', "northwest")
134 %ylim([0.5*10^(-5),10^(-1)])
0.003    1   135 hold off
136
0.084    1   137 profile viewer
138
139 % Save the image
140 relativeFolder = 'output';
141 filename = sprintf('time_evolution %d %d.png', N, Nr);
142 fullFolderPath = fullfile(pwd, relativeFolder);
143 fullFilePath = fullfile(fullFolderPath, filename);
144
145 % Ensure the directory exists
146 if ~exist(fullFolderPath, 'dir')
147     mkdir(fullFolderPath);
148 end
149
150 % Define characteristics for the image
151 exportgraphics(gcf, fullFilePath, 'Resolution', 300);
152
153 % Output display
154 disp('The simulation for')
155 disp(out1)
156 disp(['was completed in:', ' ', num2str(toc), ' seconds'])
157 disp(['using parallelisation type', ' ', type])
158 disp(['with', ' ', getenv('SLURM_CPUS_PER_TASK'), ' ', 'CPUs'])

```