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

run_all (Calls: 1, Time: 66.923 s)
Generated 05-Aug-2024 10:13:54 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
104	`[vel, el] = diagonal (H);`	25	23.765 s	35.5%	
106	`nau = GGE (N, vel);`	25	22.556 s	33.7%	
105	`E1 = time_evolution (N, hbar, ...`	25	17.355 s	25.9%	
103	`H = total_hamiltonian (N,w,mut...`	25	2.365 s	3.5%	
134	`legend([a1(1), a2(1), a3(1)], ...`	1	0.460 s	0.7%	
All other lines			0.421 s	0.6%	
Totals			66.923 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
diagonal	function	25	23.762 s	35.5%	
GGE	function	25	22.555 s	33.7%	
time_evolution	function	25	17.352 s	25.9%	
total_hamiltonian	function	25	2.282 s	3.4%	
legend	function	1	0.460 s	0.7%	
prepareAxes	function	3	0.226 s	0.3%	
xlabel	function	1	0.025 s	0.0%	
title	function	1	0.020 s	0.0%	
hold	function	2	0.018 s	0.0%	
analytical	function	1	0.015 s	0.0%	
ylabel	function	1	0.005 s	0.0%	
Self time (built-ins, overhead, etc.)			0.204 s	0.3%	

Totals			66.923 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. In local environment uncomment the next line...
                   82     % parpool
                   83     % ... and comment the next line.
                   84     initParPool()
                   85     % Initialize the random number generator with the Multiplicative lagged
                   86     % Fibonacci generator, for multiple workers in parallel
                   87     s = RandStream.create('mlfg6331_64','NumStreams', Nr,'Seed',...
                   88     'shuffle', 'CellOutput',true);
                   89     % Iterrate Nr times
                   90     parfor idx = 1:Nr
                   91     RandStream.setGlobalStream(s{idx});
                   92     H = total_hamiltonian (N,w,mutual,gamma, omega_j);
                   93     [vel, el] = diagonal (H);
                   94     E1 = time_evolution (N, hbar, tmax, vel, el, rho0);
                   95     nau = GGE (N, vel);
                   96
                   97     te_results(:, idx) = E1;
                   98     gge_results(:, idx) = nau;
                   99     end
< 0.001       1   100 else
                  101     % Iterrate Nr times
< 0.001       1   102     for idx = 1:Nr
  2.365      25   103     H = total_hamiltonian (N,w,mutual,gamma, omega_j);
 23.765      25   104     [vel, el] = diagonal (H);
 17.355      25   105     E1 = time_evolution (N, hbar, tmax, vel, el, rho0);
 22.556      25   106     nau = GGE (N, vel);
                  107
< 0.001      25   108     te_results(:, idx) = E1;
< 0.001      25   109     gge_results(:, idx) = nau;
< 0.001      25   110     end
< 0.001       1   111 end
                  112
                  113 % Get the mean of the iterations
< 0.001       1   114 te_results_mean = sum(te_results, 2) / Nr;
< 0.001       1   115 gge_results_mean = sum(gge_results, 2) / Nr;
                  116
                  117 % The analytical GGE prediction for the populations
  0.015       1   118 [nl, omega] = analytical (N, w, gamma);
                  119
                  120 % Plotting
                  121 % (i) Numerical long-time evolution
                  122 % (ii) Numerical GGE
                  123 % (iii) Analytical
                  124
  0.227       1   125 a1 = semilogy(omega_j, te_results_mean, 'o', "Color", 'b');
  0.015       1   126 hold on
  0.007       1   127 a2 = plot(omega_j, gge_results(1:N), 'x', "LineWidth", 1.1, "Color","g");
  0.004       1   128 a3 = plot(omega, nl, "LineWidth", 1.2, "Color", "r");
                  129
< 0.001       1   130 out1 = sprintf('Long-time evolution for %d spins with %d iterations', N, Nr);
```

```
0.025       1   131 xlabel("$\omega/\Omega$", 'Interpreter',"latex", 'FontSize',18)
0.005       1   132 ylabel("$n$", 'Interpreter',"latex", 'FontSize',18)
0.020       1   133 title(out1);
0.460       1   134 legend([a1(1), a2(1), a3(1)], 'Long-time evolution', 'Numerical GGE', ...
                135     'Analytical GGE', 'location', "northwest")
                136 %ylim([0.5*10^(-5),10^(-1)])
0.004       1   137 hold off
                138
0.092       1   139 profile viewer
                140
                141 % Save the image
                142 relativeFolder = 'output';
                143 filename = sprintf('time_evolution_%d_%d.png', N, Nr);
                144 fullFolderPath = fullfile(pwd, relativeFolder);
                145 fullFilePath = fullfile(fullFolderPath, filename);
                146
                147 % Ensure the directory exists
                148 if ~exist(fullFolderPath, 'dir')
                149     mkdir(fullFolderPath);
                150 end
                151
                152 % Define characteristics for the image
                153 exportgraphics(gcf, fullFilePath, 'Resolution', 300);
                154
                155 % If multicore in local environment unccoment the following line
                156 % delete(gcp('nocreate'));
                157
                158 % Output display
                159 disp('The simulation for')
                160 disp(out1)
                161 disp(['was completed in:', ' ', num2str(toc), ' seconds'])
                162 disp(['using parallelisation type', ' ', type])
                163 disp(['with', ' ', getenv('SLURM_CPUS_PER_TASK'), ' ', 'CPUs'])
```