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

run_all (Calls: 2, Time: 110.447 s)

Generated 05-Aug-2024 10:27:44 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

This function changed during profiling or before generation of this report. Results may be incomplete or inaccurate.

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
82	`% parpool`	1	0.388 s	0.4%	
66	`omega_j = sort(2*hbar*w*rand(N...`	1	0.001 s	0.0%	
70	`rho0 = zeros(N+1);`	1	0.000 s	0.0%	
74	`te_results = zeros(N, Nr);`	1	0.000 s	0.0%	
77	`gge_results = zeros(N+1, Nr);`	1	0.000 s	0.0%	
All other lines			110.058 s	99.6%	
Totals			110.447 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
parallel_function	function	1	108.695 s	98.4%	
legend	function	1	0.582 s	0.5%	
parpool	function	1	0.388 s	0.4%	
prepareAxes	function	3	0.280 s	0.3%	
addpath	function	1	0.112 s	0.1%	
rng	function	1	0.044 s	0.0%	
xlabel	function	1	0.030 s	0.0%	
notifyUI	function	1	0.025 s	0.0%	
RandStream.RandStream>RandStream.create	class method	1	0.024 s	0.0%	
close	function	1	0.024 s	0.0%	

title	function	1	0.023 s	0.0%	
hold	function	2	0.020 s	0.0%	
analytical	function	1	0.012 s	0.0%	
ylabel	function	1	0.009 s	0.0%	
clearvars	function	1	0.006 s	0.0%	
sliced_type_check	function	3	0.004 s	0.0%	
colon_range_check	function	1	0.002 s	0.0%	
pwd	function	1	0.000 s	0.0%	
fullfile	function	1	0.000 s	0.0%	
Self time (built-ins, overhead, etc.)			0.167 s	0.2%	
Totals			110.447 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 = 'multicore';
                    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
```

```
                    % A flag that indicates the consideration of internal
                    % couplings of the TLSs in the bath. Use 0 for no
                    % internal coupling, 1 to include internal coupling
< 0.001        1    mutual = 1;

                    % Sets the magnitude of the internal coupling strength.
                    % Taken to be w/(5*sqrt(2)) in the example case.
                    % For weak coupling regime, smaller of the frequency of the qubit,
                    % but is it enough small? Physical explanation for the choosen value?
< 0.001        1    gamma = w/(5*sqrt(2));

                    % The final time at which the populations are calculated.
< 0.001        1    tmax = 8000000000;

                    % Construct a N-by-1 column vector with (sorted) uniformly distributed
                    % random numbers in [0, 2*hbar*w]. It will be the diagonal elements of
                    % the bath Hamiltonian, representing the energy levels hbar*frequencies
                    % of the spins of the bath hbar*omega_j where j is in [1, N].
                    % The energy levels are sorted to reflect the ordered energy spectrum of
                    % the physical systems.
                    % This is a constant random vector during the iterations.
< 0.001        1    omega_j = sort(2*hbar*w*rand(N,1));

                    % The initial state of the system, bath in the ground state
                    % and qubit excited
< 0.001        1    rho0 = zeros(N+1);
< 0.001        1    rho0(N+1, N+1) = 1;

                    % The array for collecting the results of long time evolution
< 0.001        1    te_results = zeros(N, Nr);

                    % The array for collecting the results of the GGE prediction
< 0.001        1    gge_results = zeros(N+1, Nr);


< 0.001        1    if strcmp(type, 'multicore')
                        % Initiate the parallel poll. In local environment uncomment the next line...
  0.388        1        % parpool
                        % ... and comment the next line.
                        % initParPool()
                        % Initialize the random number generator with the Multiplicative lagged
                        % Fibonacci generator, for multiple workers in parallel
                        s = RandStream.create('mlfg6331_64','NumStreams', Nr,'Seed',...
                        'shuffle', 'CellOutput',true);
                        % Iterrate Nr times
                        parfor idx = 1:Nr
                        RandStream.setGlobalStream(s{idx});
                        H = total_hamiltonian (N,w,mutual,gamma, omega_j);
                        [vel, el] = diagonal (H);
                        E1 = time_evolution (N, hbar, tmax, vel, el, rho0);
                        nau = GGE (N, vel);

                        te_results(:, idx) = E1;
                        gge_results(:, idx) = nau;
                        end
                    else
                        % Iterrate Nr times
                        for idx = 1:Nr
                        H = total_hamiltonian (N,w,mutual,gamma, omega_j);
                        [vel, el] = diagonal (H);
                        E1 = time_evolution (N, hbar, tmax, vel, el, rho0);
                        nau = GGE (N, vel);

                        te_results(:, idx) = E1;
                        gge_results(:, idx) = nau;
```

```
    end
end

% Get the mean of the iterations
te_results_mean = sum(te_results, 2) / Nr;
gge_results_mean = sum(gge_results, 2) / Nr;

% The analytical GGE prediction for the populations
[nl, omega] = analytical (N, w, gamma);

% Plotting
% (i) Numerical long-time evolution
% (ii) Numerical GGE
% (iii) Analytical

a1 = semilogy(omega_j, te_results_mean, 'o', "Color", 'b');
hold on
a2 = plot(omega_j, gge_results(1:N), 'x', "LineWidth", 1.1, "Color","g");
a3 = plot(omega, nl, "LineWidth", 1.2, "Color", "r");

out1 = sprintf('Long-time evolution for %d spins with %d iterations', N, Nr);
xlabel("$\omega/\Omega$", 'Interpreter',"latex", 'FontSize',18)
ylabel("$n$", 'Interpreter',"latex", 'FontSize',18)
title(out1);
legend([a1(1), a2(1), a3(1)], 'Long-time evolution', 'Numerical GGE', ...
    'Analytical GGE', 'location', "northwest")
%ylim([0.5*10^(-5),10^(-1)])
hold off

profile viewer

% Save the image
relativeFolder = 'output';
filename = sprintf('time_evolution_%d_%d.png', N, Nr);
fullFolderPath = fullfile(pwd, relativeFolder);
fullFilePath = fullfile(fullFolderPath, filename);

% Ensure the directory exists
if ~exist(fullFolderPath, 'dir')
    mkdir(fullFolderPath);
end

% Define characteristics for the image
exportgraphics(gcf, fullFilePath, 'Resolution', 300);

% If multicore in local environment unccoment the following line
% delete(gcp('nocreate'));

% Output display
disp('The simulation for')
disp(out1)
disp(['was completed in:', ' ', num2str(toc), ' seconds'])
disp(['using parallelisation type', ' ', type])
disp(['with', ' ', getenv('SLURM_CPUS_PER_TASK'), ' ', 'CPUs'])
```