







GTNN_Headless_Sim (Calls: 1, Time: 14.696 s)

Generated 27-Jan-2025 13:21:43 using performance time.
Function in file [C:\Users\zaida\Desktop\Thesis\GTNN\GTNN_clean\old tool\GTNN_Headless_Sim.m](#)
[Copy to new window for comparing multiple runs](#)

Parents (calling functions)
No parent

Lines that take the most time					
Line Number	Code	Calls	Total Time (s)	% Time	Time Plot

145	$G_p = v_p - \text{netI} + Q \cdot (v_p - v_n) + P_{\text{sip}};$	1000	4.584	31.2%	
146	$G_n = v_n + \text{netI} - Q \cdot (v_p - v_n) + P_{\text{sin}};$	1000	4.554	31.0%	
45	$Q = 0.5 \cdot \text{randn}(\text{nNeuron}, \text{nNeuron});$	1	1.125	7.7%	
148	$v_p = a \cdot v_p + (dt/\tau) \cdot ((v_p.^2 - v_{\text{max}}^2) \cdot G_p) ./ \dots$	1000	0.524	3.6%	
149	$v_n = a \cdot v_n + (dt/\tau) \cdot ((v_n.^2 - v_{\text{max}}^2) \cdot G_n) ./ \dots$	1000	0.472	3.2%	
All other lines			3.436	23.4%	
Totals			14.696	100%	

Children (called functions)

Code Analyzer results

Coverage results

Total lines in function	203
-------------------------	-----

Non-code lines (comments, blank lines)	75
Code lines (lines that can run)	128
Code lines that did run	102
Code lines that did not run	26
Coverage (did run/can run)	79.69 %

Function listing

Time	Calls	Line
------	-------	------

```

1 function GTNN_Headless_Sim()
2 % GTNN_Headless_Sim - A "headless" version of the GTNN demo that
3 % runs the simulation purely in code, with no GUI or figures.
4
5 % ----- USER PARAMETERS -----
6 nNeuron      = 10000;      % Number of neurons
7 useGPU        = true;      % Set this to 'true' if you want to run on the GPU
8 plotMembrane  = true;      % If you only want to store results, not plot them
9 T             = 1000;      % Total simulation steps (in "ms" units)
10 nSpeed        = 1;        % Speed multiplier (like the slider in the GUI)
11 dt            = 0.001;     % Simulation timestep
12 tau           = 0.01;     % Time constant
13 eta           = 0.1;      % Learning rate
14 learnFlag     = false;     % Whether to enable learning or not
15 dataflag      = false;     % If true, will use external 'userdata.mat'
16 repeatdata    = 100;      % # timesteps each data vector is applied
17
18 % If not using external data, specify any DC/AC drive here:
19 I_input       = zeros(nNeuron,1); % DC input
20
21 %Example stimulation on neuron 3:
22 I_input(3)    = 0.5;
23
24 ac_amp        = zeros(nNeuron,1); % AC amplitude
25 freq          = 5*ones(nNeuron,1); % AC frequency
26 a             = ones(nNeuron,1);  % "alpha" parameter for each neuron
27 % -----
28
29 % If you want to load user data for training:
30 if dataflag
31     load userdata.mat; % This file must be in the same folder
32     [dataLen,dataDim] = size(userdata);
33 else
34     userdata = [];
35     dataLen = 0;
36     dataDim = 0;
37 end
38
39 % Initialize synaptic weight matrix Q, connectivity mask M, identity I
40 Q = zeros(nNeuron,nNeuron);
41 M = ones(nNeuron,nNeuron);
42 I = eye(nNeuron);
43
44 % (Optional) pick any custom Q here. E.g. uncomment:
45 Q = 0.5*randn(nNeuron,nNeuron);
46 Q(logical(eye(size(Q)))) = 0; % zero diagonal
47
48 % Make sure everything is on the GPU if desired:
49 if useGPU

```

```

0.339      1  50      Q      = gpuArray(Q);
0.260      1  51      M      = gpuArray(M);
0.420      1  52      I      = gpuArray(I);
< 0.001    1  53      I_input = gpuArray(I_input);
< 0.001    1  54      a      = gpuArray(a);
< 0.001    1  55      ac_amp  = gpuArray(ac_amp);
< 0.001    1  56      freq   = gpuArray(freq);
< 0.001    1  57  end
58
59  % State variables
< 0.001    1  60  vp   = -0.5*ones(nNeuron,1,'like',Q); % Positive membrane potential branch
< 0.001    1  61  vn   = -0.5*ones(nNeuron,1,'like',Q); % Negative membrane potential branch
< 0.001    1  62  Psip = zeros(nNeuron,1,'like',Q);      % +spike flags
< 0.001    1  63  Psin = zeros(nNeuron,1,'like',Q);      % -spike flags
64
65  % Logging for energy/spikes. We store the spiking energy in S_av, etc.
< 0.001    1  66  win    = 900;          % time window for smoothing
< 0.001    1  67  S_hist  = zeros(1,win,'like',Q);
< 0.001    1  68  S_av    = zeros(1,T,'like',Q);
< 0.001    1  69  spikeEnergy = 0;
70
71  % Extra parameters from the GUI version
< 0.001    1  72  Lambda = 5; % threshold parameter
< 0.001    1  73  vmax   = 1; % max membrane potential
< 0.001    1  74  vth    = 0; % threshold for spiking
< 0.001    1  75  C      = 1; % amplitude assigned to Psip / Psin upon spike
76
77  % Create variables for user-data iteration
< 0.001    1  78  currentIndex = 1;
< 0.001    1  79  currentCount = 0;
< 0.001    1  80  output      = zeros(dataLen,1); % if you're going to store data usage
81
82  % Pre-allocate a buffer for logging membrane potentials (optional).
83  % If you only need final Q or final spiking rates, you can skip this.
0.004      1  84  ylog = zeros(nNeuron, T, 'like', Q);
85
86  % ----- MAIN SIMULATION LOOP -----
87  % We do T steps, each possibly with multiple internal updates (nSpeed).
< 0.001    1  88  fprintf('Starting simulation with %d neurons for %d steps...\n',nNeuron,T);
< 0.001    1  89  iter = 1;
90
< 0.001    1  91  for t = 1:T
92
93      % Repeat "nSpeed" times each ms-step if desired
0.001      1000  94      for subIter = 1:nSpeed
95
96          % Decide on input current (from either data or user-specified AC/DC)
< 0.001      1000  97          if dataflag
98              % If using training data from 'userdata'

```

```

99         if currentCount > repeatdata
100             if learnFlag
101                 currentIndex = randi(dataLen,1); % pick random pattern
102             else
103                 currentIndex = currentIndex + 1;
104                 if currentIndex > dataLen
105                     currentIndex = 1;
106                 end
107             end
108             currentCount = 0;
109         end
110         currentCount = currentCount + 1;
111         if useGPU
112             netI = gpuArray(netI);
113         end
114
115         % "userdata" might have fewer dims than nNeuron, so pad
116         if dataDim >= nNeuron
117             netI = userdata(currentIndex,1:nNeuron).';
118         else
119             netI = [userdata(currentIndex,:), ...
120                 zeros(1,nNeuron - dataDim)].';
121         end
122         % Optionally add a constant offset:
123         netI(end) = 0.1; % from the old GUI example
124     else
125         % If not using external data, just AC+DC
126         netI = I_input + ac_amp .* sin(2*pi*freq*iter/1000);
127     end
128
129     % Convert to GPU if we are using GPU mode:
130
131
132     % -- Core spiking logic from the original code --
133
134     % 1) Check for spiking
135     spikedP = (vp > vth);
136     spikedN = (vn > vth);
137     Psip(spikedP) = C;
138     Psin(spikedN) = C;
139     vp(spikedP) = vth;
140     vn(spikedN) = vth;
141
142     % 2) Optionally record the new membrane potential for debugging
143     ylog(:,t) = vp; % or something else (like y in GUI code)
144     % 3) Calculate the gradient
145     Gp = vp - netI + Q*(vp - vn) + Psip;
146     Gn = vn + netI - Q*(vp - vn) + Psin;
147     % 4) Update vp, vn

```



```

0.524 1000 148 vp = a.*vp + (dt/tau)*(((vp.^2 - vmax^2).*Gp)./(-vp.*Gp + Lambda*vmax));
0.472 1000 149 vn = a.*vn + (dt/tau)*(((vn.^2 - vmax^2).*Gn)./(-vn.*Gn + Lambda*vmax));
150
151 % 5) Compute spiking energy
0.248 1000 152 numSpikes = sum(spikedP) + sum(spikedN);
0.091 1000 153 S_hist = [S_hist(2:end), numSpikes]; % shift in sliding window
0.130 1000 154 spikeEnergy = sum(S_hist)/(2*win*nNeuron);
0.056 1000 155 S_av(t) = spikeEnergy;
156
157 % 6) Learning
< 0.001 1000 158 if learnFlag
159     % Weight update (Psp-Psin)*(vp-vn)'
160     Q = Q + 0.5*eta * M .* ((Psp - Psin)*(vp - vn)');
< 0.001 1000 161 end
162
163 % 7) Reset spike flags
0.054 1000 164 Psp(:) = 0;
0.042 1000 165 Psin(:) = 0;
166
167 % Optional: store some result if dataflag==1 and learnFlag==0
< 0.001 1000 168 if dataflag && ~learnFlag
169     output(currentIndex) = spikeEnergy;
< 0.001 1000 170 end
171
< 0.001 1000 172 iter = iter + 1;
< 0.001 1000 173 end % of sub-iterations
< 0.001 1000 174 end % of T steps
175
176 % Pull back GPU arrays to CPU memory if needed
< 0.001 1 177 if useGPU
0.231 1 178     Q = gather(Q);
0.024 1 179     ylog = gather(ylog);
< 0.001 1 180     S_av = gather(S_av);
< 0.001 1 181 end
182
183 % Report final connectivity matrix, energy, etc.
< 0.001 1 184 fprintf('Simulation complete.\n');
0.003 1 185 fprintf('Final mean spiking energy = %g.\n', mean(S_av(end-50:end)));
186
187 % If you want a quick plot at the end, optionally do it here:
< 0.001 1 188 if plotMembrane
0.076 1 189     figure;
0.101 1 190     subplot(1,2,1);
0.184 1 191     imagesc(Q + eye(nNeuron));
0.247 1 192     colorbar; title('Final Connectivity (Q+I)');
193
0.029 1 194     subplot(1,2,2);
0.012 1 195     plot(S_av, 'LineWidth', 2);
< 0.001 1 196     set(gca, 'YScale', 'log');

```

```
0.012      1  197      xlabel('Time step'); ylabel('Spiking Energy');
0.012      1  198      grid on; title('Spiking Energy (log scale)');
< 0.001    1  199      end
                200
                201      % If needed, save results
                202      % save('GTNN_results.mat','Q','ylog','S_av','-v7.3');
0.070      1  203      end
```

Local functions in this file are not included in this listing.
