

```
%  
% Offline simulator for Dynamic Neural Fields, 1-layer architecture with  
% memory trace  
%  
% The model consists of one dynamic field with associated memory trace:  
% field_u - dynamic field (acts both excitatory and inhibitory)  
% memTrace_u - memory trace of layer u  
%  
% Parameters for each layer have the corresponding letter as index, with mu  
% for the parameters of the memory traces  
%  
%  
% Connections in the model:  
% u->u (exc.), u->u (inh)  
% u->mu (exc.), mu->u (exc.)  
%  
% Connection parameters within layer u are designated with indices exc or  
% inh, kernels and connection parameters between two layers are indexed  
% with the the indices for both layers, with the layer that receives the  
% input written first, e. g.:  
% kernel_uu: interaction kernel for connection u->u (exc + inh)  
% kernel_mu: interaction kernel from u to its memory trace  
%  
% The connection from a memory trace to the associated layer does not use  
% an interaction kernel, the activity of the memory trace is simply added  
% to the activity of the field (accordingly, m does not need a resting  
% level or a steepness parameter for the sigmoid function).  
  
%%%%%%%%%
```

```
% model parameters %
%%%%%%%%%%

fieldSize = 181; % must be odd

tau_u = 20; % time constant of dynamic field
tau_mu = 1000; % time constant of memory traces
beta_u = 5; % steepness parameter of sigmoid function
h_u = -5; % resting level

% c: strength of interaction kernel
% sigma: width of interaction kernel
% g: strength of global inhibition
c_exc = 0; sigma_exc = 5;
c_inh = 0; sigma_inh = 10; g_inh = 0;
c_mu = 0; sigma_mu = 10;

q_u = 0; % noise levels

%%%%%%%%%%
% simulation time course %
%%%%%%%%%%

nTrials = 1;
tMax = 250;

%tStoreFields = [100, 200]; % at these time steps the field activities will be stored
tStoreFields = 1:tMax; % use this to store field activities at all time steps
```

```
% for example
tStimulusStart = 50;
tStimulusEnd = 150;
stim1 = 6*gauss(1:fieldSize, 60, 5); % a localized input
stim2 = 6*gauss(1:fieldSize, 120, 10); % a broad localized input
stim = 5*gauss(1:fieldSize, 90, 5) + 1; % a hill of input with homogeneous boost

%%%%%%%%%%%%%%
% initialization %
%%%%%%%%%%%%%%

halfField = floor(fieldSize/2);

% create row vectors for field activities
field_u = zeros(1, fieldSize);
memTrace_u = zeros(1, fieldSize);

% create matrices to store field activities at different times
history_s = zeros(nTrials * length(tStoreFields), fieldSize);
history_u = zeros(nTrials * length(tStoreFields), fieldSize);
history_mu = zeros(nTrials * length(tStoreFields), fieldSize);

% index of the current position in the history matrices
iHistory = 1;

% set up the interaction kernels
kernel_uu = c_exc * gauss(-halfField:halfField, 0, sigma_exc) ...
    - c_inh * gauss(halfField:halfField, 0, sigma_inh) - g_inh;
kernel_mu = c_mu * gauss(-halfField:halfField, 0, sigma_mu);
```

```
%%%%%%%%%%
% simulation %
%%%%%%%%%%

% loop over trials
for i = 1 : nTrials

    % prepare matrix that holds the stimulus for each time step
    stimulus = zeros(tMax, fieldSize);

    % if needed, create a new stimulus pattern for every trial
    stimPos = round(fieldSize * rand);
    stimT = 6*gauss(1:fieldSize, stimPos, 5);

    % write the stimulus pattern into the stimulus matrix for all time steps
    % where it should be active
    for j = tStimulusStart : tStimulusEnd
        stimulus(j, :) = stimT;
    end

    % reset field activities to resting levels
    field_u(1:fieldSize) = h_u;

    % loop over time steps
    for t = 1 : tMax
        % calculation of field outputs
        output_u = sigmoid(field_u, beta_u, 0);
    end
end
```

```
% circular padding of outputs for convolution
output_u_padded = [output_u(halfField+2:fieldSize), output_u, output_u(:, 1:halfField)];

% get endogenous input to fields by convolving outputs with interaction kernels
conv_uu = conv2(1, kernel_uu, output_u_padded, 'valid');
conv_mu = conv2(1, kernel_mu, output_u_padded, 'valid');

% create field noise for this timestep
noise_u = q_u * randn(1, fieldSize);

% update field activities
field_u = field_u + 1/tau_u * (-field_u + h_u + stimulus(t, :) ...
    + conv_uu + memTrace_u) + noise_u;
memTrace_u = memTrace_u + 1/tau_mu * (-memTrace_u + conv_mu);

% store field activities at the selected time steps
if any(tStoreFields == t)
    history_s(iHistory, :) = stimulus(t, :);
    history_u(iHistory, :) = field_u;
    history_mu(iHistory, :) = memTrace_u;
    iHistory = iHistory + 1;
end
end
end

%%%%%%%%%%
% visualization of results %
%%%%%%%%%%
```

```
% note: you can also access and plot all stored results afterwards from the
% Matlab prompt

% view plots of all stored field activities iteratively
nStoredFields = nTrials * length(tStoreFields);
if 1
    figure;
    for i = 1 : nStoredFields
        plot(0:fieldSize-1, zeros(1, fieldSize), ':k', 1:fieldSize, history_s(i, :), '--g', ...
            1:fieldSize, history_mu(i, :) + h_u, '-c', 1:fieldSize, history_u(i, :), '-b');
        set(gca, 'XLim', [0 fieldSize-1], 'YLim', [-10 10]);
        ylabel('activity u');
        pause(0.01);
    end
end

% view evolution of field activities in each trial as mesh plot
nFieldsPerTrial = length(tStoreFields);
if 0
    figure;
    for i = 1 : nTrials
        subplot(2, 1, 1);
        mesh(1:fieldSize, tStoreFields, history_u((i-1)*nFieldsPerTrial+1 : i*nFieldsPerTrial, :));
        zlabel('activity u');
        subplot(2, 1, 2);
        mesh(1:fieldSize, tStoreFields, history_mu((i-1)*nFieldsPerTrial+1 : i*nFieldsPerTrial, :));
        zlabel('memory trace u');
        pause
    end
end
```

end

```
% view mesh plot of all stored field activities together
if 0
    figure;
    subplot(2, 1, 1);
    mesh(1:fieldSize, 1:nStoredFields, history_u(:, :));
    zlabel('activity u');
    subplot(2, 1, 2);
    mesh(1:fieldSize, 1:nStoredFields, history_mu(:, :));
    zlabel('memory trace u');
end
```