% Create an i x n array, where i = n + 1 due to the presence of the i = 0

% case.

final\_array = zeros(9,8);

% Enter the number of channels to be simulated (iprim), the average

% duration of each event (event\_average), the standard deviation of events

% (event\_stdev), open probability, and the number of events to simulate.

% Finally, create an array with 2i columns and 2\*event number rows. Odd

% rows correspond to closed-to-open transitions, even rows to the inverse.

for iprim = 1:8

number\_of\_channels = iprim;

event\_average = 30;

event\_stdev = 10;

open\_probability = 0.3;

number\_of\_events\_to\_simulate = 2000;

array\_of\_simulations = zeros(2\*number\_of\_events\_to\_simulate,number\_of\_channels\*2);

% Generate open events for each independent channel, then determine the

% total duration of open and closed events based on open probability.

% Then, randomly distribute timepoints (closedeventmarkers) across the

% closed-event dwell time. Open events are initiated at these markers.

for i = 1:number\_of\_channels;

open = normrnd(event\_average,event\_stdev,[number\_of\_events\_to\_simulate,1]);

closedbase = sum(open);

closedbase = round(closedbase \* (1-open\_probability)/(open\_probability));

closedeventmarkers = randperm(closedbase,number\_of\_events\_to\_simulate);

closedeventmarkers = closedeventmarkers';

closedeventmarkers = sort(closedeventmarkers);

% Calculate the length of closed events through the time difference between

% two markers. Then combine open and closed event lengths into a single

% array (final\_durations), in which each opening event is followed by

% closure.

closedfinal = closedeventmarkers(2:end) - closedeventmarkers(1:end-1);

closedfinal = [closedeventmarkers(1,1);closedfinal];

final\_durations = zeros(2\*number\_of\_events\_to\_simulate,2);

for ib = 1:number\_of\_events\_to\_simulate;

final\_durations(ib\*2-1,1) = closedfinal(ib);

final\_durations(ib\*2-1,2) = 0;

final\_durations(ib\*2,1) = open(ib);

final\_durations(ib\*2,2) = 1;

combine\_part1 = zeros(2\*number\_of\_events\_to\_simulate,2);

end;

% Add up the durations to generate a final timeline of open and closed

% events (array\_of\_simulations) for each independent channel.

for ic = 1:2\*number\_of\_events\_to\_simulate;

combine\_part1(ic,1) = sum(final\_durations(1:ic,1));

combine\_part1(ic,2) = final\_durations(ic,2);

end

array\_of\_simulations(:,2\*i-1) = combine\_part1(:,1);

array\_of\_simulations(:,2\*i) = combine\_part1(:,2);

end

% Limit the total analysis time to the shortest total observation duration

% across all channels. Typically, all observation durations are roughly

% identical.

baselinearray=array\_of\_simulations(:,1:2:end);

baseline = min(baselinearray(2\*number\_of\_events\_to\_simulate,:));

baseline = floor(baseline) - 1;

% At each time point across the observation period, check how many channels

% are open (great\_sum) and record this in an array.

giant\_array = zeros(1,baseline);

for id = 1:baseline;

great\_sum = 0;

for ie = 1:number\_of\_channels

findentry = find(array\_of\_simulations(:,2\*ie-1) > id,1);

findlevel = array\_of\_simulations(findentry,2\*ie);

great\_sum = great\_sum + findlevel;

end

giant\_array(1,id) = great\_sum;

end

% For each combination of n and i, count and record the time points in

% which i channels are open. This array provides the p(i) distribution for

% each n.

dwelltimes = zeros(number\_of\_channels+1,1);

for id = 1:number\_of\_channels+1,

dwelltimes(id,1) = sum(giant\_array(:) == id-1);

final\_array(id,iprim) = dwelltimes(id,1);

end

end