```
% NOTES: this version embeds only the sticky feature. It works properly.

clear all
close all
clc


%cjb
global  k norm_r events d_max mu lam lam_mu lam_mu_d lam_b xstore xd_maxstore x Ve repeat tst
energymodule = 1; % 0/1 = turn energy-dependent adhesion module off/on.
Fmax = 1.04E-8; % N, Value derived from literature is 1.04E-10. Please see report for calcula
events = 7E2; % number of events
repeat = 100; % number of successive runs. Increase rather than events to avoid RAM problems.
norm_r = 0.64E-6; %m typical radius of one cell, if cell-shape is defined as a sphere.
T = 300; % Environment temperature [^o Kel]
norm_ma = 1E-15; % typical mass of a cell in [kg]
kb = 1.38*10^(-23); % Boltzmann's constant [J/(K^-1)]

norm_mu = 0*.03^(1/3600); % s^-1, Typical growth rate of a solitary cell.
norm_m = 0; % mol S / mol X, Typical maintenance cost of a solitary cell.
step_alt = 0; % Defining the total number of event steps upon reaching maximum cluster size.

n = 20; % max number of initial clusters
scale = 20; % max individual dimension for initial numbers
id = fix(scale*rand(n,1));
d_max = sum(id.*[1:1:n]',1);  % number of cells present in the system, is also the maximum cl
startod = 1; % Typical concentration of starting biomass.
Ve = d_max/(startod*1e15); %  Volume of modeled environment.
mu = zeros(d_max,1); % Defining starting mu.

 for i = 1:d_max
     mu(i,1) = norm_mu*i^(1/3)-norm_m; % Defining net growth rate for each cluster size.
 end

x = zeros(length(find(id)),2,(events)*repeat); % Defining matrix containing cluster size and
xd_max = zeros(events-1,2); %Defining vector containing number of cells present in the system
xd_maxstore = []; % Defining matrix vector for storing xd_max in case of multiple runs.
t = zeros(events,1); %Defining vector containing time point for each event.
tstore = []; % Defining vector for storing t in case of multiple runs.
k = 1; %Defining current event number.
x(:,:,k) = [find(id) id(id>0)]; % Starting conditions of cluster size and number.
xstore = []; % Defining storing matrix for x. Avoids RAM problems.
tic

lam = zeros(length(find(id)),length(find(id))); % Defining lower-triangular matrix containing

for i=1:length(find(id))
    for j=1:i
        if i == j
            comb = 0.5*x(i,2,k)*(x(j,2,k)-1);
            lam(i,j) = comb*bp(x(i,1,k),x(j,1,k)); %The average arrival rate of a cluster of
        else % i \neq j
            comb = x(i,2,k)*x(j,2,k);
            lam(i,j) = comb*bp(x(i,1,k),x(j,1,k)); %The average arrival rate of a cluster of
        end
    end
end

lam_start = lam; %Stores initial arrival rates for error handling.

lam_mu = zeros(length(find(id)),1); % Defining lower-triangular matrix containing arrival rat
```

```
lam_mu_d = zeros(length(find(id)),1); % Defining lower-triangular matrix containing arrival r

for i = 1:length(find(id))
    mu = norm_mu*x(i,1,1)^(1/3)-norm_m; %Determine net growth rate for each cluster present.
    if mu > 0 % If net growth is positive, calculate arrival rate for growth.
            lam_mu(i,1) = log(1+1/(x(i,2,k)*x(i,1,k)))*mu;
    elseif mu < 0 % If net growth rate is negative (death), calculate arrival rate for death.
            lam_mu_d(i) = -log(1+1/(x(i,2,k)*x(i,1,k)))*mu;
    end
end



time = 0;
k = 0;

for rep = 1:repeat
repk = k; %Ensures continuity in case of multiple runs.
for k = 1:events-1
    k = repk + k;
    lam_b = sum(sum(lam,1),2)+sum(lam_mu)+sum(lam_mu_d);
    tp = 0; %Defining value which tracks progress towards cf
    sw = 0; %Defining value which tracks whether the specific event has been determined
    i = 1; j = 1; h=1; f=1; %Initial value for lam indexes
    te = - log(rand)/lam_b; % Time passed before occurence of event, since last event.
    cf = rand*lam_b; %Random value out of distribution of arrival rates is chosen.
    xd_max(k,1) = d_max; %stores value of d_max
    xd_max(k,2) = sum(x(:,1,k).*x(:,2,k)); %stores calculated value of d_max for error handli

    if sum(sum(lam,1),2) >= cf %If a clash has occured (not growth or death)
        while sw == 0
            if tp + sum(lam(i,:)) < cf % and the current event is not within the i row of eve
                tp = tp + sum(lam(i,:)); % the sum of all arrival rates of the i row of event
                i = i+1;                  % and we continue with the next row.
            else                         % if the current event is in the i row of events
                tp = tp + lam(i,j);      % we add the arrival rate of i,j
                if tp >= cf              % and if the sum of arrival rates now exceeds cf
                    sw = 1;              % event i,j is the current event.
                else                     % if not,
                    if i == j            %  and we have reached the end of current row (lower-
                        i = i + 1;       % we proceed with the next row
                        j = 1;
                    else % i \neq j      if we are not yet at the end of the row
                        j = j + 1;       % we proceed with the next event
                    end
                end
            end
        end

    elseif  sum(sum(lam,1),2) +sum(lam_mu) >= cf   %same, but if growth has occured.
        i = size(x,1)+1;
        tp = sum(sum(lam,1),2);      % sum of arrival rates of clashes is added.

        while sw == 0
            tp = tp + lam_mu(h);
            if tp >= cf
                sw = 1;
            else
                h = h +1;
            end
        end
```

```
    elseif sum(sum(lam,1),2)+sum(lam_mu)+sum(lam_mu_d) >= cf % same, but if death has occured
        h = size(x,1)+1;
        tp = sum(sum(lam,1),2)+sum(lam_mu);
        while sw == 0
            tp = tp + lam_mu_d(f);
            if tp >= cf
                sw = 1;
            else
                f = f + 1;
            end
        end
    else %if neither a clash, growth or death has occured -> error
            sw =3
            k
    end

    % effects of current event on distrubition of clusters.
    if i > size(x,1) % if no clash has occured...
        if h > size(x,1) % ...but death has.
            d_max = d_max - 1;
            update_lam_x(f,0,x(f,1,k)-1,-1,0,1);
        else % ...but growth has.
            d_max = d_max + 1;
            update_lam_x(h,0,x(h,1,k)+1,-1,0,1);
        end
    else % if a clash has occured
        if 1 == stick(x(i,1,k),x(j,1,k), energymodule) % we determine if the two clusters wil
            update_lam_x(i,j,x(i,1,k)+x(j,1,k),-1,-1,+1);
        else
             update_lam_x(0,0,0,0,0,0);
        end
    end

    if any(x(:,1,k) == d_max) ==1 %if any of the clusters (though it should only be one) equa
        xd_max(k+1,1) = d_max;
        xd_max(k+1,2) = sum(x(:,1,k).*x(:,2,k));
        step_alt = k
        break
    end

    time = time +te;
    t(k+1,1) = time;  %k+1 since t(1) = 0.

end

if step_alt > 0 %extra ensurance of break
    break
end

xd_maxstore = [xd_maxstore ; xd_max];
tstore = [tstore ; t];

end

toc
%clustergraph
figure(1)
yaxdim = 2;
subplot(3,3,1)
bar(x(1:find(x(:,1,1),1,'last'),1,1),x(1:find(x(:,1,1),1,'last'),2,1),'g')
xlabel('Cluster size')
```

```
ylabel('Amount of clusters')
title('t = 0')
xlim([0 xd_max(k,1)])
set(gca, 'XTick', 0:500:xd_max(k,1))

subplot(3,3,2)
bar(x(1:find(x(:,1,fix(k/8)),1,'last'),1,fix(k/8)),x(1:find(x(:,1,fix(k/8)),1,'last'),2,fix(k
xlabel('Cluster size')
ylabel('Amount of clusters')
title(['t = ' num2str(fix((k)/8),3)])
xlim([0 xd_max(k,1)])
set(gca, 'XTick', 0:500:xd_max(k,1))

subplot(3,3,3)
bar(x(1:find(x(:,1,fix(2*k/8)),1,'last'),1,fix(2*k/8)),x(1:find(x(:,1,fix(2*k/8)),1,'last'),2
xlabel('Cluster size')
ylabel('Amount of clusters')
title(['t = ' num2str(fix(2*(k)/8),3)])
xlim([0 xd_max(k,1)])
set(gca, 'XTick', 0:500:xd_max(k,1))

subplot(3,3,4)
bar(x(1:find(x(:,1,fix(3*k/8)),1,'last'),1,fix(3*k/8)),x(1:find(x(:,1,fix(3*k/8)),1,'last'),2
xlabel('Cluster size')
ylabel('Amount of clusters')
title(['t = ' num2str(fix(3*(k)/8),3)])
xlim([0 xd_max(k,1)])
set(gca, 'XTick', 0:500:xd_max(k,1))

subplot(3,3,5)
bar(x(1:find(x(:,1,fix(4*k/8)),1,'last'),1,fix(4*k/8)),x(1:find(x(:,1,fix(4*k/8)),1,'last'),2
xlabel('Cluster size')
ylabel('Amount of clusters')
title(['t = ' num2str(fix(4*(k)/8),3)])
xlim([0 xd_max(k,1)])
set(gca, 'XTick', 0:500:xd_max(k,1))

subplot(3,3,6)
bar(x(1:find(x(:,1,fix(5*k/8)),1,'last'),1,fix(5*k/8)),x(1:find(x(:,1,fix(5*k/8)),1,'last'),2
xlabel('Cluster size')
ylabel('Amount of clusters')
title(['t = ' num2str(fix(5*(k)/8),3)])
xlim([0 xd_max(k,1)])
set(gca, 'XTick', 0:500:xd_max(k,1))

subplot(3,3,7)
bar(x(1:find(x(:,1,fix(6*k/8)),1,'last'),1,fix(6*k/8)),x(1:find(x(:,1,fix(6*k/8)),1,'last'),2
xlabel('Cluster size')
ylabel('Amount of clusters')
title(['t = ' num2str(fix(6*(k)/8),3)])
xlim([0 xd_max(k,1)])
set(gca, 'XTick', 0:500:xd_max(k,1))

subplot(3,3,8)
bar(x(1:find(x(:,1,fix(7*k/8)),1,'last'),1,fix(7*k/8)),x(1:find(x(:,1,fix(7*k/8)),1,'last'),2
xlabel('Cluster size')
ylabel('Amount of clusters')
title(['t = ' num2str(fix(7*(k)/8),3)])
xlim([0 xd_max(k,1)])
set(gca, 'XTick', 0:500:xd_max(k,1))
```

```
subplot(3,3,9)
bar(x(1:find(x(:,1,k),1,'last'),1,k),x(1:find(x(:,1,k),1,'last'),2,k),'g')
xlabel('Cluster size')
ylabel('Amount of clusters')
title(['t = ' num2str(fix(k),3)])
xlim([0 xd_max(k,1)])
set(gca, 'XTick', 0:500:xd_max(k,1))

xavgcluster = xd_maxstore(1:k,1)./permute(sum(x(:,2,1:k)), [3 2 1]);

if step_alt > 0
xavgcell(step_alt+1) = d_max;
xavgcluster(step_alt+1) = d_max;
k = k +1;
end

figure(2)
plot(t(1:1:k),xavgcluster','go', 'MarkerSize', 1);
xlabel('time')
ylabel('Average cluster size');

%Settling

%Compare settling behaviour at eventpoint tp with settling behaviour of
%initial clusters.

h = .25; % m, average starting height of cells at moment of settling.
ts = 3600; % seconds, sinking time
tp = k; %number of the event of which cluster dimensions are used for settling calculation
hx = []; % m, average height at t=ts of each cluster size
hxi = []; %m, average height at t=ts of each cluster size if initial cluster dimensions are u

V_P = 3e-18;        % typical volume of a cell, 3 um^3
p_P = 1.11e3;       % density of a cell 1.1 g/mL --> 1e-3/1e-6 kg/m^3
p_F = 1.05e3;      % medium density 1.13 kg/m3
V_Fp = V_P * (1/pi*sqrt(18) -1 );    % fraction of volume consisting of medium in packed stat
g = 9.81;           % m/s^2
nu = 1e-3;          % 1 g/m/s2 --> 1e3 kg/m/s2 dynamic viscosity

for nn = 1:find(x(:,1,tp), 1,'last')

        n = x(nn,1,tp);
        % calculate volume
        V = n * V_P + (n-1) * V_Fp;
        m = n * V_P * p_P + n * V_Fp * p_F;
        r = (3*V/4/pi)^1/3;
        % Stoke's law
        w = 2*(m/V - p_F) * g * r^2 / ( 9 * nu );
        if h - w*ts >= 0
            hx = [hx; (h - w*ts)];
        else
            hx = [hx; 0 ];
        end
end

maj = 0; % Defining value for majority of biomass
%starting from the largest cluster, we sum the number of cells of each
%cluster size until we have more than 99% of biomass. The settling velocity
%of the smallest cluster needed for 99% biomass is the limiting velocity.

for i = size(x,1):-1:1
```

```
    maj = maj + x(i,1,tp).*x(i,2,tp)./xd_max(tp,1);
    if maj >= 0.99
        n = x(i, 1, tp);
        % calculate volume
        V = n * V_P + (n-1) * V_Fp;
        m = n * V_P * p_P + n * V_Fp * p_F;
        r = (3*V/4/pi)^1/3;
        % Stoke's law
        w = 2*(m/V - p_F) * g * r^2 / ( 9 * nu );
        hourtsx = h/(w*3600);
        break
    end
end

figure(4)
scatter( x(x(:,1,tp)>0, 1,tp) , hx(:,1), x(1:find(x(:,1,tp), 1,'last'),1,tp).*x(1:find(x(:,1,
hold on
ylim([0 h])
xlabel( 'cluster size' );
ylabel( 'height (m)' );
grid on;
box on;


for nn = 1:find(x(:,1,1), 1,'last')

        n = x(nn,1,1);
        % calculate volume
        V = n * V_P + (n-1) * V_Fp;
        m = n * V_P * p_P + n * V_Fp * p_F;
        r = (3*V/4/pi)^1/3;
        % Stoke's law
        w = 2*(m/V - p_F) * g * r^2 / ( 9 * nu );
        if h - w*ts >= 0
            hxi = [hxi; (h - w*ts)];
        else
            hxi = [hxi; 0 ];
        end
end

maj = 0; %majority of biomass

for i = size(x,1):-1:1
    maj = maj + x(i,1,1).*x(i,2,1)./xd_max(1,1);

    if maj >= 0.99
        n = x(i, 1, 1);
        % calculate volume
        V = n * V_P + (n-1) * V_Fp;
        m = n * V_P * p_P + n * V_Fp * p_F;
        r = (3*V/4/pi)^1/3;
        % Stoke's law
        w = 2*(m/V - p_F) * g * r^2 / ( 9 * nu );
        hourtsx = h/(w*3600);
        break
    end
end

figure(4)
scatter( x(x(:,1,1)>0, 1,1) , hxi(:,1), x(1:find(x(:,1,1), 1,'last'),1,1).*x(1:find(x(:,1,1),
hold on
```

```
ylim([0 h])
xlabel( 'cluster size' );
ylabel( 'height (m)' );
grid on;
box on;
```