```
function [Ui] = inputcontrol_Algorithm1(num_nodes, nodes, Nei_agent, n, epsilon
    , r, d, p_nodes)
    %{
    This function is to find alpha and beta neighbors
    Created by Anthony Bugatto

    Inputs: positions of nodes (nodes),
            indices of alpha neighbors (Nei_agent)
            (n)
            (epsilon)
            active range for alpha agents (r)
            (d)
            (k_scale)
            (p_nodes)

    Outputs: controlled acceleration (Ui)

    %}
    %+++++++++++++++++Constants++++++++++++++++++++

    c_a1 = 30;
    c_a2 = 2*sqrt(c_a1);
    a = 5;
    b = 5;
    c = abs(a - b) / sqrt(4*a*b);
    r_sig = sigma_norm(r);
    d_sig = sigma_norm(d);

    %+++++++++++++++++BUILDING THE ADJACENCY MATRCEES+++++++++++++++++++++

    n_ij = zeros(num_nodes,num_nodes,n); %gradient matrix 1x2
    for i = 1:num_nodes
        for j = 1:num_nodes
            q = norm(nodes(j,:) - nodes(i,:));
            sig_grad = (nodes(j,:) - nodes(i,:)) / (1 + epsilon * sigma_norm
                (nodes(j,:) - nodes(i,:)));

            if q < r && q ~= 0 %is zero otherwise
                n_ij(i,j,:) = sig_grad;
            end
        end
    end

    %+++++++++++++++BUILDING CONTROL ACCELERATION Ui++++++++++++++++++++++

    U = zeros(num_nodes, n); %100x3 matrix for accelerations

    gradient = 0;
    conscensus = 0;
    a_ij = zeros(num_nodes,num_nodes);
    for i = 1:num_nodes %loop through all i in Ui matrix
        for j = 1:size(Nei_agent{i}) % loop through all neighbors in neighbor
            matrix for each i
            Nei_val = Nei_agent{i}(j);
            if(i ~= Nei_val)
                %phi is the time differential of the smooth pairwise
```

```
                %   attractive/repulsive  potential
                z = sigma_norm(nodes(Nei_val,:) - nodes(i,:)); %parameter for
                    phi_alpha
                z_phi = z - d_sig; %parameter for phi
                rho_h = bump(z / r_sig);
                sigmoid = (z_phi + c) / sqrt(1 + (z_phi + c)^2);
                phi = .5 * ((a + b) * sigmoid + (a - b));

                phi_alpha = rho_h * phi;

                a_ij(i,Nei_val) = rho_h;
                %implement the algorithm for the fragmenting control law:
                %
                % Ui = c_a1*SUM[phi_alpha * nij) + c_a2*SUM[aij * (pj - pi)]
                %

                gradient = phi_alpha * [n_ij(i,Nei_val,1) n_ij(i,Nei_val,2)];
                conscensus = a_ij(i,Nei_val) * (p_nodes(Nei_val,:) - p_nodes(i
                    ,:));
            end
        end

        U(i,:) = (c_a1 * gradient) + (c_a2 * conscensus);
    end

    Ui = U;
end
```