```
function [t, y] = backward_euler_newton(f, df, tspan, ic, nsteps, tol)
    t0 = tspan(1);
    tf = tspan(end);
    h = (tf - t0) / nsteps;
    t = zeros(nsteps+1,1);
    y = zeros(nsteps + 1,length(ic));
    y(1) = ic;
    % Main loop for Backward Euler method
    for j = 1:nsteps
        t(j+1) = t(j) + h;
        g = @(y_jplus1) y_jplus1 - h*f(t(j+1), y_jplus1)- y(j);
        g_prime = @(y_jplus1) 1 - h*df(t(j+1), y_jplus1); % Derivative of g with
respect to y
        % Initial guess for Newton's method
        x0 = y(j);
        % Apply Newton's method to solve for y(j+1)
        d = newtons_method(g, g_prime, x0, tol);
        y(j+1)= d(end); % Assuming newtons_method is implemented as discussed
    end
end
```