```
%
% ENGR 1221
% Final Project Simulation
%

% inputs

% Open loop Steering constant (u)
% Constant Speed (v)
% Auto Constants (A,B,C)
% Polynomial Constants(third, second, first, zeroth)
% Final Sim Time (tFinal)

% Outputs
% Position vectors (x, y), RMS Error (E)

%Units: kg, m, s



function [x, y, E] = Simulation(u, v, A, B, C, third, second, first, zeroth,
tFinal)


%Initialize Constants

m = 1700;
J = 2000;
c1 = 100000;
c2 = 80000;
a = 1.5;
b = 1.3;

%Initialize error
Etot = 0;

%define dt
tInterval = 0.01;

%initialize position vectors
x = zeros(1, tFinal*(1/tInterval));
y = zeros(1, tFinal*(1/tInterval));
t = zeros(1, tFinal*(1/tInterval));

% dv_n = zeros(1, tFinal*(1/tInterval));
% domega = zeros(1, tFinal*(1/tInterval));
% dx = zeros(1, tFinal*(1/tInterval));
% dy = zeros(1, tFinal*(1/tInterval));
%
% v_n = zeros(1, tFinal*(1/tInterval));
% v_t = zeros(1, tFinal*(1/tInterval));
%
% omega = zeros(1, tFinal*(1/tInterval));
% theta = zeros(1, tFinal*(1/tInterval));

%Initialize state variables
dvn = 0;
dvt = 0;
vt = v;
```

```
vn = 0;
dx = 0;
dy = 0;
dom = 0;
om = 0;
auto = false;
ds = 0;

%Create symbolic polynomial
syms z f(z) df(z)

f(z) = third * z^3 + second * z^2 + first * z + zeroth;
df(z) = diff(f(z), z);

%Choose between autonomous and manual based off of u value
if isnan(u)
    %Initialize with Polynomial origin and angle
    thet = atand(double(df(0)));
    y(1) = double(f(0));
    auto = true;
else
    %Initialize with Polynomial origina and angle
    thet = 0;
    y(1) = double(f(0));
end

for k = 1:length(t)-1

    ds = abs(y(k) - double(f(x(k))));

    Etot = Etot + sqrt(ds.^2);
    if auto
        u = double(C*ds + B * (atand(df(x(k) + A)) - thet));
    end

    %Calculating velocities, angles, and positions using Euler's method
    dvn = tInterval*((-(c1 + c2)*vn + (-a*c1 + b*c2)*om)/(m*vt) + (c1 / m) * u);
    dom =  tInterval*((-(a*c1 - b*c2) * vn - (a^2 * c1 + b^2 * c2)*om)/(J*vt) + (a
* c1 / J) * u);
    om = dom + om;
    thet = rad2deg(om)*tInterval + thet;
    vn = vn + dvn;
    vt = sqrt(v^2 - vn^2); % vt^2 + vn^2 = v^2 because of pythagorean theorem
    dx = tInterval*(vt*cosd(thet)+vn*sind(thet));
    dy = tInterval*(vn*cosd(thet)+vt*(sind(thet)));
    x(k+1) = x(k) + dx;
    y(k+1) = y(k) + dy;



end
E = double(Etot / sqrt(abs(length(t))-1)); %Root Mean square





end
```