**Appendix C – Codes accompanying Question 3**

|  |
| --- |
| function xdot = NL\_DynModel(t,x,u,Pnoise)  %NL\_DynModel  % input: t - time model; x - state vector; u - control input vector;  % Pnoise - process noice vector  % output: \_\_\_  % Function input for ode45 for NL dynamics model    % u = [v\_g, phi\_g, v\_a, w\_a]';  % x = [xi\_g eta\_g theta\_g xi\_a eta\_a theta\_a]';  % ~w = [w\_x,g w\_y,g w\_w,g w\_x,a w\_y,a w\_w,a]';  L = 0.5;    xdot = [u(1)\*cos(x(3)) + Pnoise(1);  u(1)\*sin(x(3)) + Pnoise(2);  (u(1)/L)\*(tan(u(2))) + Pnoise(3);  u(3)\*cos(x(6)) + Pnoise(4);  u(3)\*sin(x(6)) + Pnoise(5);  u(4) + Pnoise(6)];  end |
| function [y] = NL\_MeasModel(x,Mnoise)  %NL\_MeasModel  % input: x - state vector; Mnoise - measurement noice vector  % output: y - sensor readings;  % Uses NL state inputs and measurement noise vector to get sensor  % readings    % y = [gamma\_ag rho\_ga gamma\_ga xi\_a eta\_a]';  % x = [xi\_g eta\_g theta\_g xi\_a eta\_a theta\_a]';  % x = [1 2 3 4 5 6 ]';    y = [atan2(x(5)-x(2),x(4)-x(1)) - x(3);  sqrt((x(1)-x(4))^2 + (x(2)-x(5))^2);  atan2(-x(5)+x(2),-x(4)+x(1)) - x(6);  x(4);  x(5)];    y = y + Mnoise;    end |
| function [A\_t,B\_t,C\_t] = Linearize(x,u)  %Linearize  % input: x - nominal state vector; u - nominal control input;  % output: A\_t - A tilde Matrix; B\_t - B tilde Matrix; C\_t - C tilde  % Matrix  % Obtain the CT linearized state perturbation matrices    % u = [v\_g, phi\_g, v\_a, w\_a]';  % x = [xi\_g eta\_g theta\_g xi\_a eta\_a theta\_a]';    L = 0.5;    A\_t = [0 0 -u(1)\*sin(x(3)) 0 0 0;  0 0 u(1)\*cos(x(3)) 0 0 0;  0 0 0 0 0 0;  0 0 0 0 0 -u(3)\*sin(x(6));  0 0 0 0 0 u(3)\*cos(x(6));  0 0 0 0 0 0];      B\_t = [cos(x(3)) 0 0 0;  sin(x(3)) 0 0 0;  tan(u(2))/L (u(1)/L)\*sec(u(2))^2 0 0;  0 0 cos(x(6)) 0;  0 0 sin(x(6)) 0;  0 0 0 1];    % x = [xi\_g eta\_g theta\_g xi\_a eta\_a theta\_a]';    C11 = (x(5)-x(2))/((x(5)-x(2))^2 + (x(4)-x(1))^2);  C12 = -(x(4)-x(1))/((x(5)-x(2))^2 + (x(4)-x(1))^2);  C13 = -1;  C14 = -(x(5)-x(2))/((x(5)-x(2))^2 + (x(4)-x(1))^2);  C15 = (x(4)-x(1))/((x(5)-x(2))^2 + (x(4)-x(1))^2);  C21 = (x(1)-x(4))\*((x(1)-x(4))^2 + (x(2)-x(5))^2)^-0.5;  C22 = (x(2)-x(5))\*((x(1)-x(4))^2 + (x(2)-x(5))^2)^-0.5;  C24 = -(x(1)-x(4))\*((x(1)-x(4))^2 + (x(2)-x(5))^2)^-0.5;  C25 = -(x(2)-x(5))\*((x(1)-x(4))^2 + (x(2)-x(5))^2)^-0.5;  C31 = -(x(2)-x(5))/((x(2)-x(5))^2 + (x(1)-x(4))^2);  C32 = (x(1)-x(4))/((x(2)-x(5))^2 + (x(1)-x(4))^2);  C34 = (x(2)-x(5))/((x(2)-x(5))^2 + (x(1)-x(4))^2);  C35 = -(x(1)-x(4))/((x(2)-x(5))^2 + (x(1)-x(4))^2);  C36 = -1;  C44 = 1;  C55 = 1;    C\_t = [C11 C12 C13 C14 C15 0;  C21 C22 0 C24 C25 0;  C31 C32 0 C34 C35 C36;  0 0 0 C44 0 0;  0 0 0 0 C55 0];  end |
| function [x\_DTL, dx\_DTL, y\_DTL, dy\_DTL] = DT\_L\_Model(t,Dt,x\_NL,y\_NL,x\_pert,u\_nom)  %DT\_L\_Model  % input: t - time; Dt - time increment; x\_NL - nominal state trajectory;  % y\_NL - nominal sensor readings; x\_pert - initial state  % perturbation; u\_nom = nominal control input;  % output: x\_DTL = discrete time state; dx\_DTL = discrete time state  % perturbation; y\_DTL = discrete time sensor; dy\_DTL = discrete  % time sensor perturbation  % Function for linear DT model    % u = [v\_g, phi\_g, v\_a, w\_a]';  % x = [xi\_g eta\_g theta\_g xi\_a eta\_a theta\_a]';  % ~w = [w\_x,g w\_y,g w\_w,g w\_x,a w\_y,a w\_w,a]';    x\_nominal = [x\_NL(1,:); x\_NL(2,:);  wrapToPi(x\_NL(3,:)); x\_NL(4,:);  x\_NL(5,:); wrapToPi(x\_NL(6,:))];  y\_nominal = y\_NL;    F = zeros(6,6,length(t));  H = zeros(5,6,length(t));    % Evaluate F and H matrices with predef nominal state trajectories  for i=1:length(t)  [A\_t, ~, C\_t] = Linearize(x\_nominal(:,i),u\_nom);  F(:,:,i) = eye(6) + A\_t\*Dt;  H(:,:,i) = C\_t;  end    dx\_DTL = zeros(6,length(t));  dy\_DTL = zeros(5,length(t));  x\_DTL = zeros(6,length(t));  y\_DTL = zeros(5,length(t));  dx\_DTL(:,1) = x\_pert;  dy\_DTL(:,1) = H(:,:,1)\*x\_pert;  x\_DTL(:,1) = x\_nominal(:,1)+dx\_DTL(:,1);  y\_DTL(:,1) = y\_nominal(:,1)+dy\_DTL(:,1);    for i=2:length(t)  dx\_DTL(:,i) = F(:,:,i)\*dx\_DTL(:,i-1);  dy\_DTL(:,i) = H(:,:,i)\*dx\_DTL(:,i);  x\_DTL(:,i) = x\_nominal(:,i)+dx\_DTL(:,i);  y\_DTL(:,i) = y\_nominal(:,i)+dy\_DTL(:,i);  end  end |