Task 3

The given system of ODEs is

$$x''(t) = F u_x - K x'(t)$$

$$y''(t) = F u_y - K y'(t)$$

where K and F are given constants and $\underline{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$ is a unit vector (calculated at each time t to point from the missile to the target).

Conversion to 1st-order system

Let $\boxed{z_1(t) = x(t); \quad z_2(t) = y(t); \quad z_3(t) = x'(t); \quad z_4(t) = y'(t)}$.

Then we get the following 1st-order system of ODEs:

$$z_1'(t) = z_3(t)$$

$$z_2'(t) = z_4(t)$$

$$z_3'(t) = F u_x - K z_3(t)$$

$$z_4'(t) = F u_y - K z_4(t)$$

as stated in the assignment question.

10

```
%
%    Simulation of pursuit problem
%

% Some simulation parameters
target_number = 1;
force = 10;
K = 1;
dmin = 0.1;

tspan = [0 10]; % Set start and end times for computation
yStart = [-1.9; 1.1; 0; 0];      % rocket starting position


% Set options for the ODE solver
options = odeset('Events',@pursuit_events);

% Call the ODE solver (Runge-Kutta 4/5)
[t,y] = ode45(@missile,tspan,yStart,options,force,K,target_number,dmin);


%---------------------------------------------------------------
%
% Everything below here is just plotting & visualization
%
%---------------------------------------------------------------

% Plot missile and target positions over time.
figure(1);
N = length(t);
T = zeros(N,2); %position of target
for i = 1:N
    T(i,:)= target(t(i), target_number)';
end
plot(y(:,1),y(:,2),'rx-', T(:,1),T(:,2),'b');
xlabel('x'); ylabel('y'); axis equal;
title('Scenario (a) Trajectories');

% Plot distance from missile to target over time.
figure(2);
dist = zeros(size(t));
for i = 1:N
    relPos = T(i,:) - y(i,1:2);
    dist(i) = norm(relPos,2);
end
plot(t,dist);
xlabel('Time'); ylabel('Distance');
title('Scenario (a) Distance between missile and target');

% Animate the two trajectories
animate_pursuit(t,y,T);
```

11

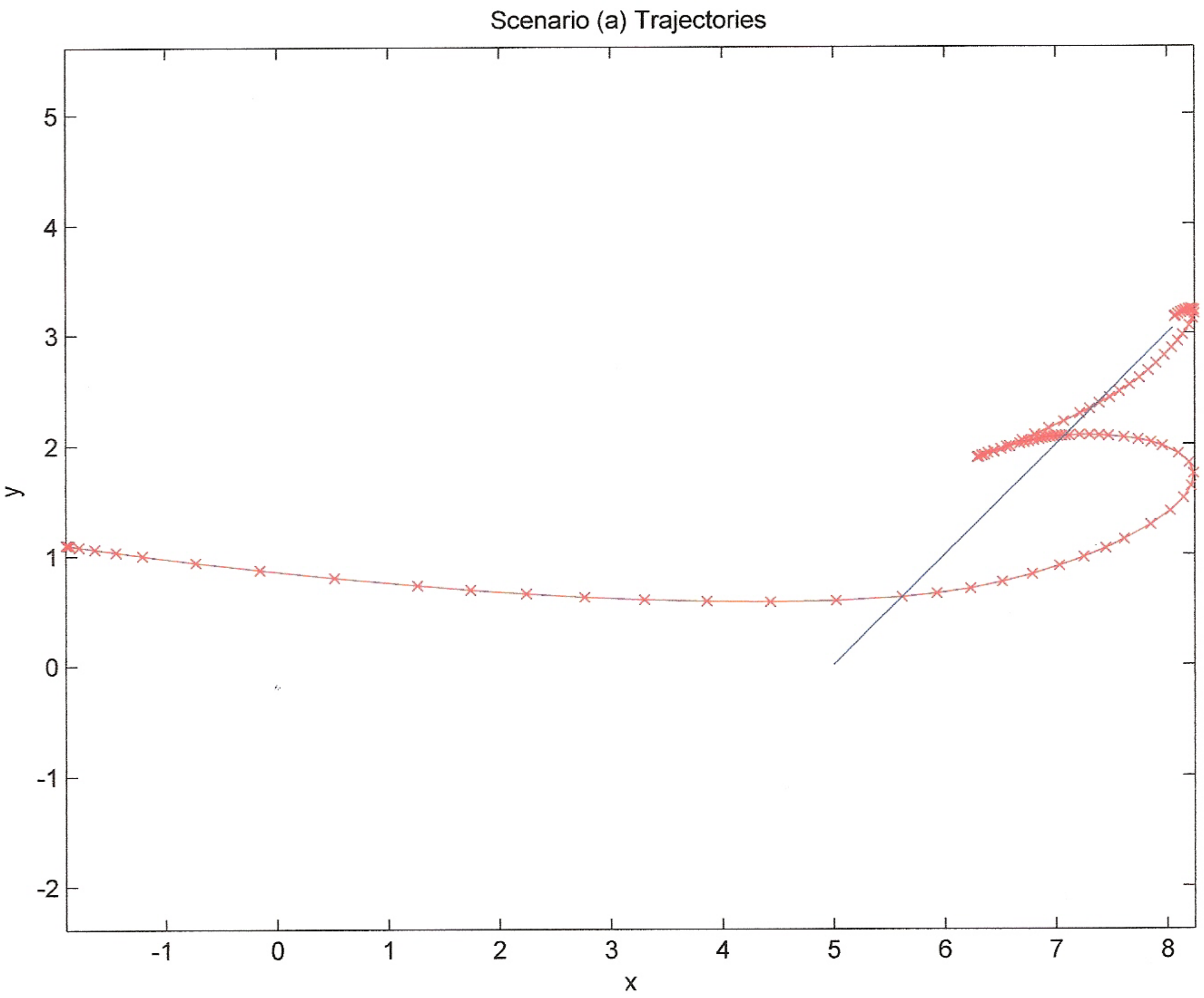
```
%
% This is the function that is called by the ode solver.
% It implements the dynamics for the missile in the pursuit problem.
%
function dzdt = missile(t,z,force,K,target_number,dmin)

    T = target(t, target_number);

    dx = T-z(1:2);  % position vector of target from pursuer
    dist = norm(dx,2);

    dzdt = zeros(4,1);
    dzdt(1:2) = z(3:4);
    dzdt(3:4) = force*dx/dist - K * z(3:4);
```

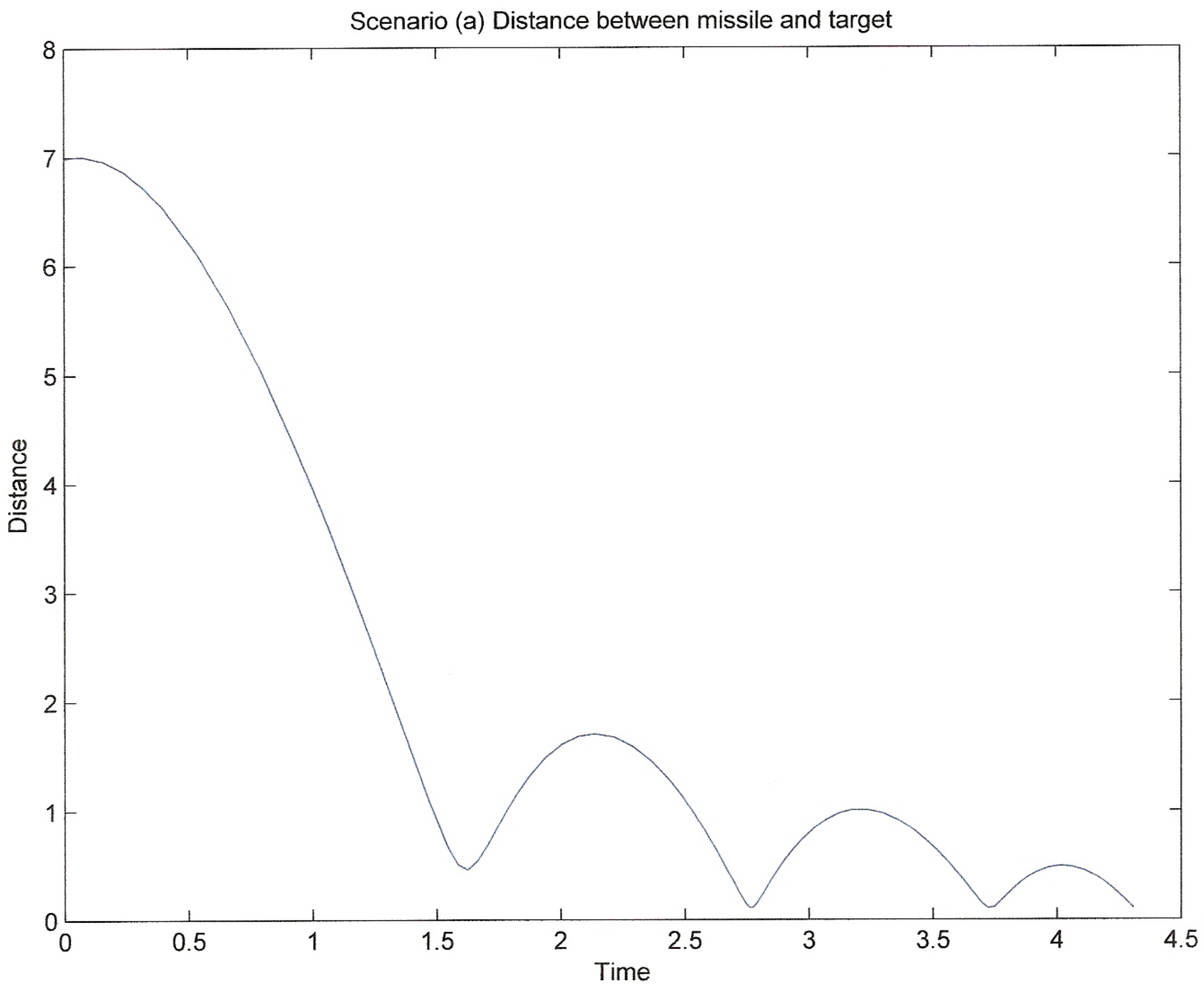
12

```
% function Tposition = target(t, target_number)
%
% This is the target function for the pursuit
% problem. It is really two different targets.
% A particular target is selected by setting the
% input argument "target_number" to 1 or 2.
%
function Tposition = target(t, target_number)

    Tposition = zeros(2,1);

    if target_number == 1  % target flies at unit speed on straight line
        Tposition(1) = t/sqrt(2)+5;
        Tposition(2) = t/sqrt(2);
    elseif target_number == 2
        Tposition(1) = 1 - 2 * abs(mod(t,2) - 1);
        Tposition(2) = 1+t;
    else
        disp('target_number not set');
    end
```

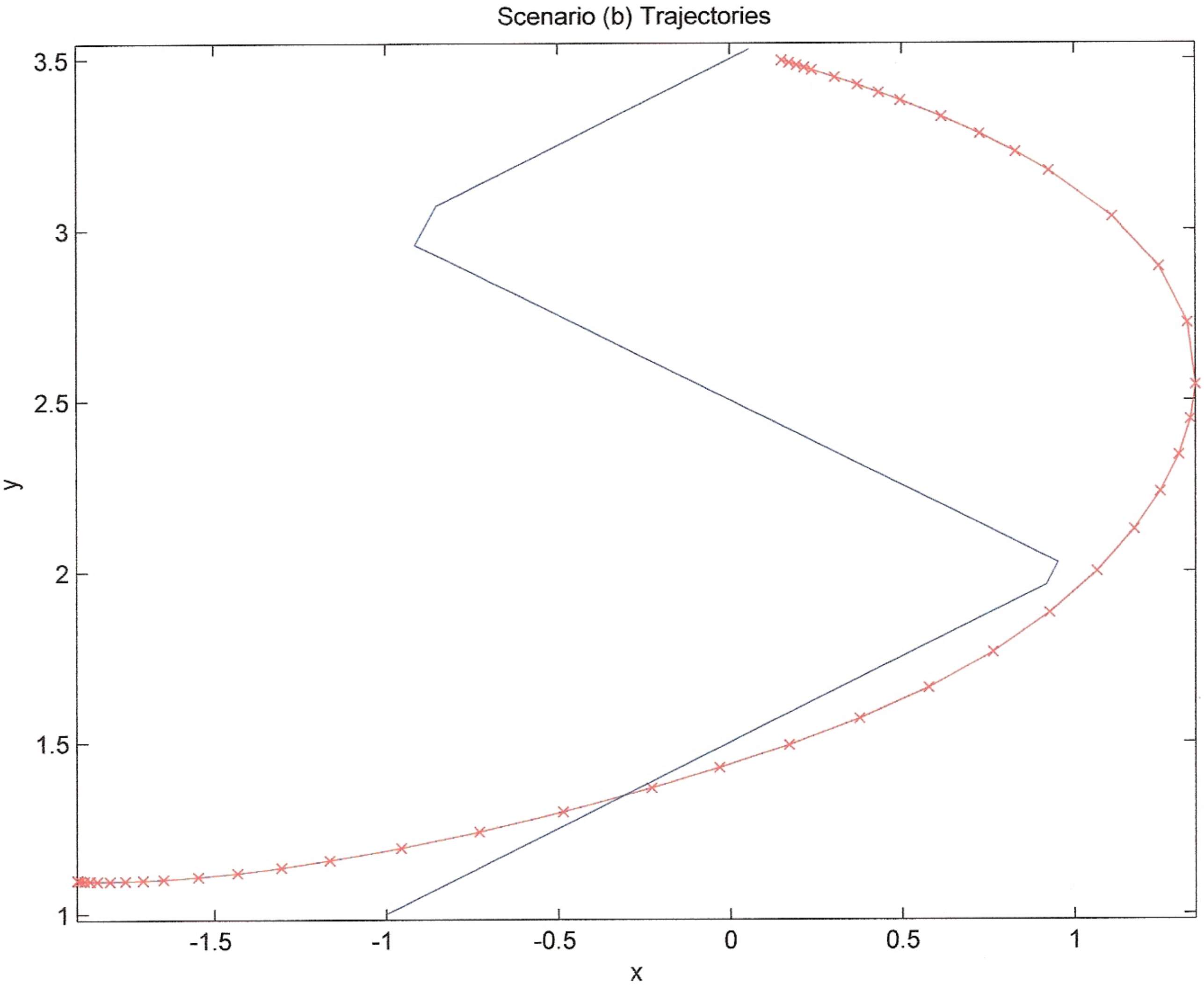
(13)

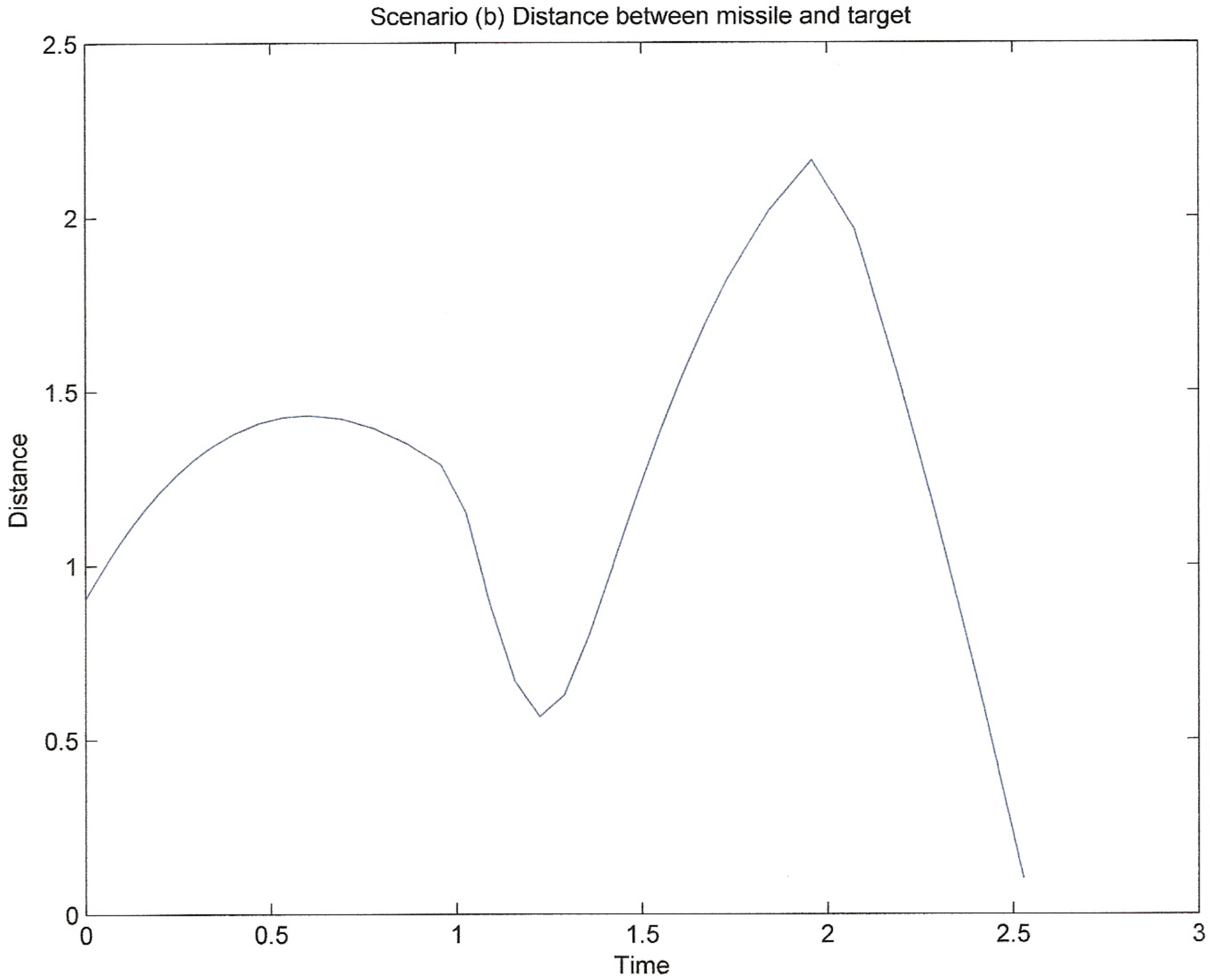
```
%
% This is a terminal event function, called by the ode solver.
%
function [value,isterminal,direction] = pursuit_events(t,P,force,K,target_number,dmin)

    T = target(t, target_number);

    dx = T - P(1:2);
    dist = norm(dx,2);

    value = dist - dmin;

    if value <= 0
        disp('Target acquired!');
    end

    isterminal = 1;    % Stops the integration
    direction = -1;    % Negative direction
```

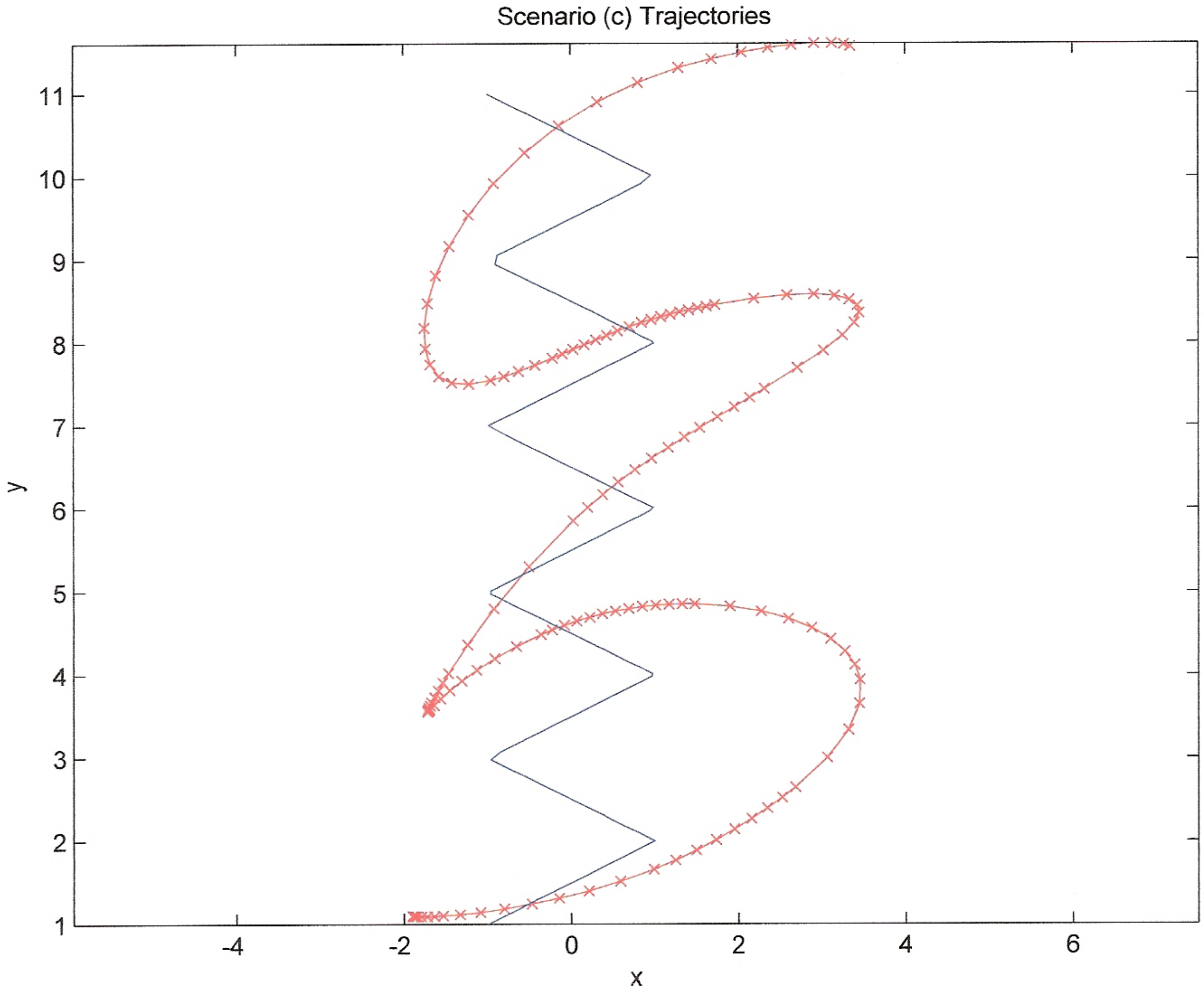

Scenario (a) Trajectories
y
5
4
3
2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8
x


Scenario (a) Distance between missile and target
Distance
Time
0
1
2
3
4
5
6
7
8
0.5
1.5
2.5
3.5
4.5

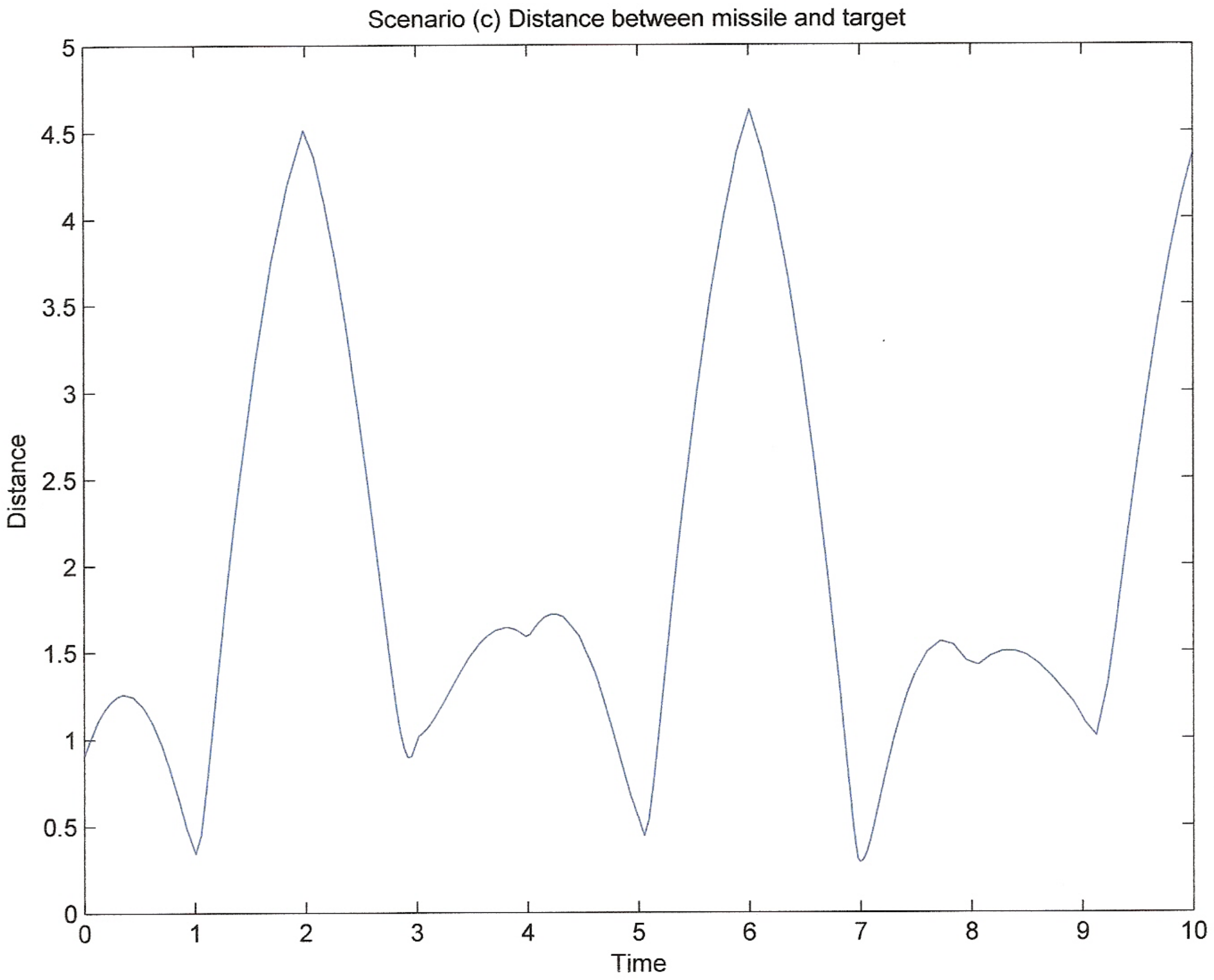
16




Scenario (b) Distance between missile and target
Distance
Time
0
0.5
1
1.5
2
2.5
3



Scenario (c) Trajectories
y
x
11
10
9
8
7
6
5
4
3
2
1
-4
-2
0
2
4
6


Scenario (c) Distance between missile and target
Distance
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
1
2
3
4
5
6
7
8
9
10