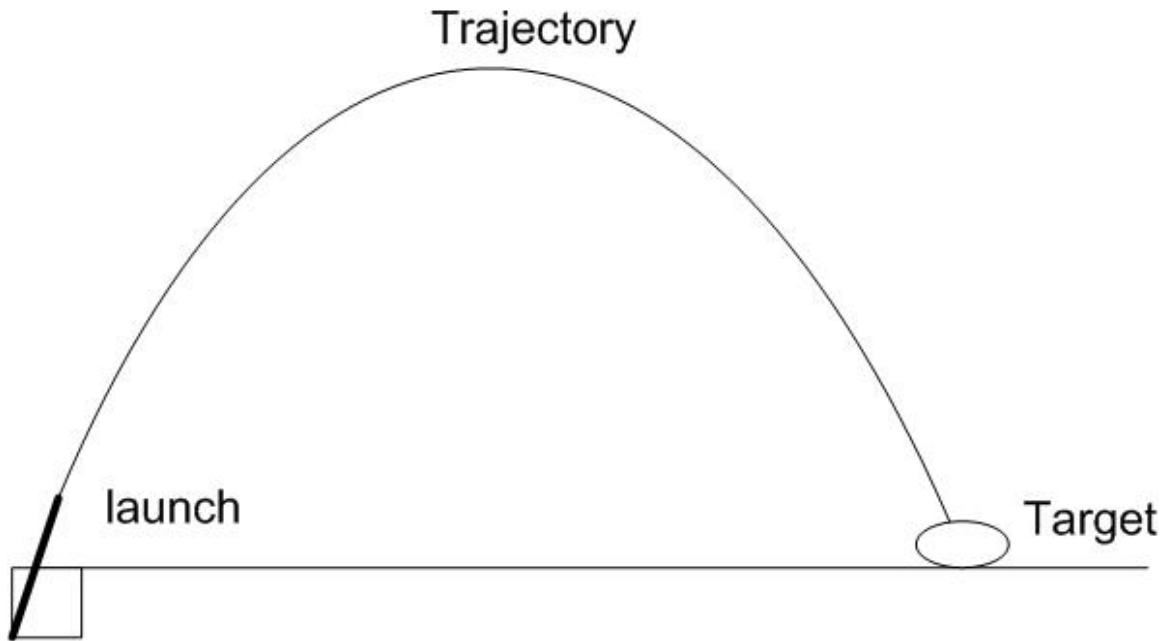


CIEG 675

Homework #8 Due **Wednesday April 29, 2009**

Just one problem. Recall when you were a kid (ok you are too young)... There used to be a game on the old RadioShack computers where you were responsible for providing an initial trajectory angle to a munitions expert so they could try to hit a target a known distance away with a mortar round (see picture below). You can either make the initial velocity constant or have the user provide as an input (I used 250 m/s). Redevelop that game in matlab where the inputs to your function are shot angle and target distance. Assume for simplicity launch and target are located at the same elevation unless you want to have elevation be an input as well. Assume a HIT if the mortar strikes within 10 m of the target. To make it fun, plot intermediate steps on the screen by investigating the usage of **pause(n)**. Neglect air drag.



solns next page

```

function mortar_game(angle, dist2targ, targelev, velocity)

% a game to determine hit of a target some distance away where the user
% specifies an initial trajectory angle and a velocity if they wish
% target and shot assumed to be at same elevation.
% target assumed hit if within 5 m.
%
% input:
% angle: initial angle of trajectory (degrees)
% dist2target: distance to target in meters
% targelev: is the target elevation
% velocity: initial speed in m/s of munition. default is 250 m/s
%

if nargin ==3
    velocity = 250; % m/s
end

v0=velocity; % simplify variable name
ang=angle*pi/180; % get to radians
d=dist2targ;
E=targelev;

% constants
g=9.81; % acceleration due to gravity
dist4success = 10;

%The equation for x motion is
% x-x0=v0cos(theta0)t % since no horizontal acceleration
% y-y0=v0sin(theta0)t-0.5gt^2

% you can combine these by removing t to get
% y=tan(theta0)x- gx^2/(2(v0cos(theta0))^2 )

% the range (total horizontal distance travelled)
% can be found by setting x-x0 to R and y-y0 to 0 as
% R=v0^2/g * sin(2(theta0));

% knowing these, lets still press ahead and do it the slow loop way
% for us, set x0 and y0 to 0

% R=v0^2/g * sin (2*ang); % but only for 0 elevation change from launch point to %
target
R1=(2*v0^2*(cos(ang))^2*tan(ang)/g + sqrt( 4*v0^4 * (cos(ang))^4 *
(tan(ang))^2/g^2 - 8*v0^2*(cos(ang))^2*E/g ) )/2; % pos root

```

```

R2=(2*v0^2*(cos(ang))^2*tan(ang)/g - sqrt( 4*v0^4 * (cos(ang))^4 *
(tan(ang))^2/g^2 - 8*v0^2*(cos(ang))^2*E/g  ) )/2; % neg root

% check for imaginary
if imag(R1)~=0
    R1=0;
end
if imag(R2)~=0
    R2=0;
end

R=max(R1,R2); % maximum travel distance.

x=linspace(0,R,100); % create a vector form 0 to R with 100 elements.
y=tan(ang)*x - g*x.^2/(2*(v0*cos(ang))^2  ) ;

%% make the game board
clf
plot(0,0,'ks','markersize',10);
hold on
plot(d,E,'bd','markersize',10,'markerfacecolor','b');
axis([-10 max(max(x)+50,d+50) min(-10,E) max(y)+50]) % so axes dont bounce
around
xlabel('Range (m)');
ylabel('Elevation (m)');

% make plot
hit=0;
k=0;
while hit==0 & k<length(x)
    k=k+1; % counter
    plot(x(k),y(k),'ro','markersize',6,'markerfacecolor','r');
    pause(0.05)
    %determine whether or not the region was hit

    dist=sqrt( (x(k)-d)^2 + (y(k)-E)^2);

    if dist<dist4success
        hit=1;
    end % you were close enough for assignment

end % while loop

if hit==1

```

```
disp('Congrats, you hit the target')
else
    txt=sprintf('not such a good shot, you were %f m away',dist);
    disp(txt);
end % if loop
```

