OPTIMIZACIÓN DE LA TRAYECTORIA DE UN DISPARO

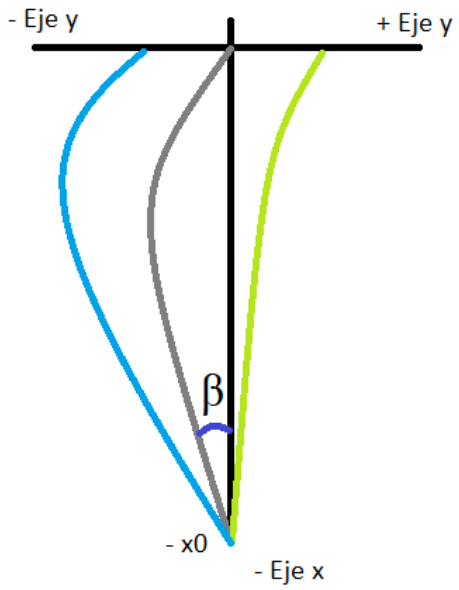
JOSÉ CARLOS RIEGO MOZAS

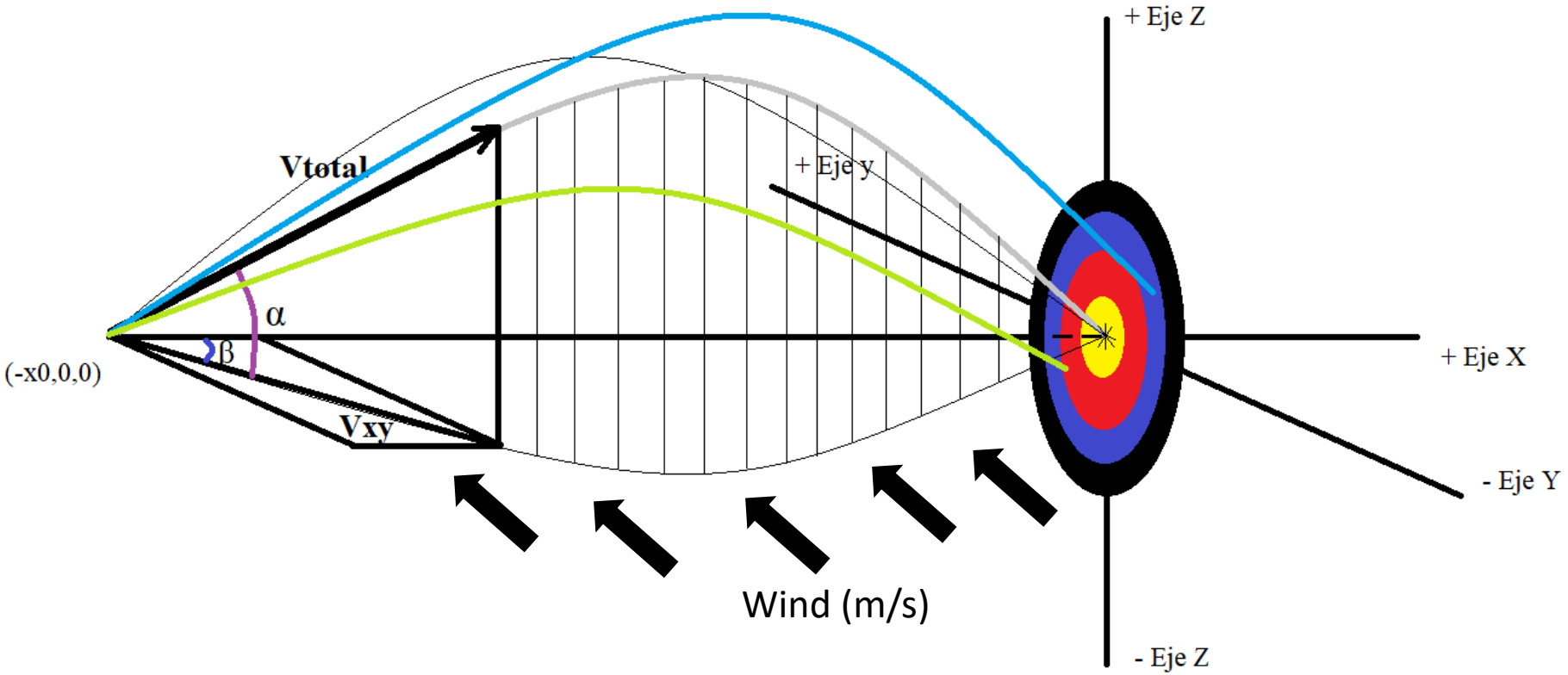
Y

PABLO RODRÍGUEZ SORIA

INTRODUCCIÓN Y OBJETIVOS

Queremos encontrar los ángulo óptimos (α y β) para los que el disparo da en el blanco (0,0,0) m.





VIENTO (Vwind.m)

Creamos un programa para definir el viento (con dirección + eje y) que sufre la partícula en función de la altura a la que se encuentra.

```
% Vwind.m
function V = Vwind(r)
% Find the velocity of the wind in function the height
% Input:
% r(1,3) : position of the particle (m)
% Output:
% V(1,3) : velocity of the wind (m/s)

% Set values of wind
vmin = 0;                          % min value (m/s)
vmax = 50*(1e3/3600);              % max value (m/s)
vy = linspace(vmin,vmax,50);
Vx = zeros(1,length(vy));          % x-axis velocity (m/s)
Vz = Vx;                           % z-axis velocity (m/s)
Vy = vy;                           % y-axis velocity (m/s)
V = [Vx;Vy;Vz];                    % Matrix velocity (m/s)
```

→ Velocidad del viento para h = 0 m

→ Velocidad del viento para h = 50 m

→ Vector velocidad con incremento

Matriz de vientos (m/s)

VIENTO (Vwind.m)

Creamos un programa para definir el viento (con dirección + eje y) que sufre la partícula en función de la altura a la que se encuentra.

```
% Velocity in function of height
h = r(3);                          % Find height of the object
h = round(h);                      % h must be a whole number
% NOTE: becuase the height has been rounded, at height h the velocity is
% the value whose index is nearer to h
if h < 0
    V = [0;0;0];                   % There is no wind under ground
elseif h == 0
    V = [0;0;0];                   % At h=0 (ground), there is no wind
elseif h > 0 && h < length(vy)
    V = V(:,h);                    % V increases with height
elseif h >= length(vy)
    V = [0;vmax;0];                % At more than height 50 m, wind is constant (vmax)
end
end
```

Altura del objeto (eje Z)

Convertimos h en un número entero

Definimos la velocidad del viento a partir de 50 m

Ejemplo:

```
>> Vwind([4;5;2.5])

ans =

         0
    0.5669
         0
```

```
>> Vwind([4;5;3.4])

ans =

         0
    0.5669
         0
```

Matriz V:

```
V =

  Columns 1 through 4

         0         0         0         0
         0    0.2834    0.5669    0.8503
         0         0         0         0
```

FUERZA (force.m)

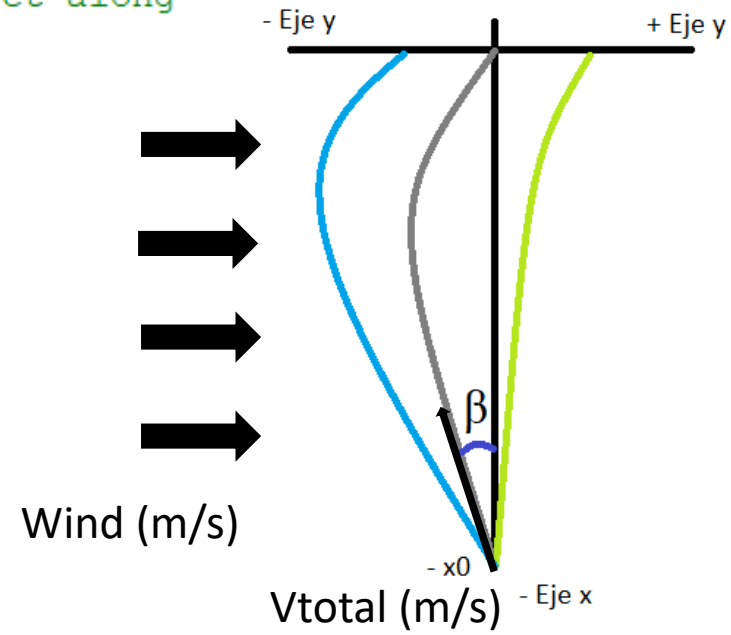
Creamos un programa para definir la fuerza que sufre la partícula.

$$F_{wind} = -A \bullet \frac{1}{2} \bullet \rho_{air}(T) \bullet Cd \bullet (v - Vwind)^2$$

$$\overrightarrow{F_{wind}} = -A \bullet \frac{1}{2} \bullet \rho_{air}(T) \bullet Cd \bullet \left(\vec{v} - \overrightarrow{V_{wind}}\right)^2 = -A \bullet 0{,}613 \bullet Cd \bullet \left|v - V_{wind}\right|^2 \bullet \frac{\left(\vec{v} - \overrightarrow{V_{wind}}\right)}{\left|v - V_{wind}\right|}$$

Vector unitario

```
% myforce.m
function f = force(m,A,Cd,r,v,Vwind)
% Find the total force due to wind and gravity applied in an object along
% the trajectory
% Input:
%   m : mass of the object (kg)
%   A : area of the object (m^2)
%   Cd : drag coefficient
%   den : air density (Kg/m^3)
%   r(3,1) : position of the object (m)
%   v(3,1) : velocity of the object (m/s)
%   Vwind(3,1) : velocity of the wind (m/s)
% Output:
%   f(3,1) : total force on the object (N)
```



FUERZA (force.m)

```
% Set Planet's surface gravity
% Earth
R = 6371e3;             % Earth's radius (m)
M = 5.972e24;           % Earth's mass (Kg)
G = 6.672e-11;          % Universal gravitation constant (m^2/kg*s^-2)
den = 1.23;             % Air density at T = 25°C (Kg/m^3)

% Mars
% R = 3389e3;           % Earth's radius (m)
% M = 6.39e23;          % Earth's mass (Kg)
% G = 6.672e-11;        % Universal gravitation constant (m^2/kg*s^-2)
% den = 1.30;           % Air density at T = 0°C (Kg/m^3)

% Set force
f = -A*0.5*den*Cd*(norm(v-Vwind))^2*(v-Vwind)/norm(v-Vwind);

% Add gravity force on the z-axis
g = -G*M/R^2;                          % Planet's surface gravity (m/s^2)
f_g = m*g;                             % Gravity force (N)
f(3,:) = f(3,:) + f_g;                 % Final z-axis force
```

$$Eje\ z$$

$$\downarrow$$

$$F_g = g \bullet m = -G \bullet \frac{M \bullet m}{R^2}$$

$$\downarrow$$

$$g = -G \bullet \frac{M}{R^2}$$

$$F_{total} = F_{wind} + F_g$$

PUNTO DE IMPACTO (ImpactPoint.m)

Creamos un programa para hallar el punto de impacto en los ejes Y y Z (manteniendo la condición x=0) para 2 ángulos cualesquiera.(α y β).

```
% impactPoint.m
function [y,z,t01] = ImpactPoint(alpha,beta,m,Cd,A,x0,Vtotal,tmax,dt)
% Find the impact point in the y and z axis (where x = 0).
% Input:
%   m : mass of the object (kg)
%   A : area of the object (m^2)
%   Cd : drag coefficient
%   x0 : initial position in x-axis (m)
%   Vtotal : module of the initial velocity of the object (m/s)
%   alpha : angle formed by the direction of the shot with the xy plane
%   (rad)
%   beta : angle formed by the projection of the shot direction with the
%   x-axis (rad)
%   tmax : total time of the trajectory (s)
%   dt : interval of time (s)
% Output:
%   y : impact point in the y axis (m)
%   z : impact point in the z axis (m)
```

PUNTO DE IMPACTO (ImpactPoint.m)

Integramos la trayectoria llamando a rungeKutta

```
% Set initial params
r0 = [x0;0;0];                                                                    % Initial position (m)
v0 = [Vtotal*cos(alpha)*cos(beta); Vtotal*cos(alpha)*sin(beta); Vtotal*sin(alpha)]; % Initial velocity (m/s)

% Integrate trajectory of the object
myforce = @(r,v,t) force(m,A,Cd,r,v,Vwind(r));
[r] = rungeKutta(myforce,m,r0,v0,dt,tmax);
r = squeeze(r);                  % Get all the columns together in a matrix
```

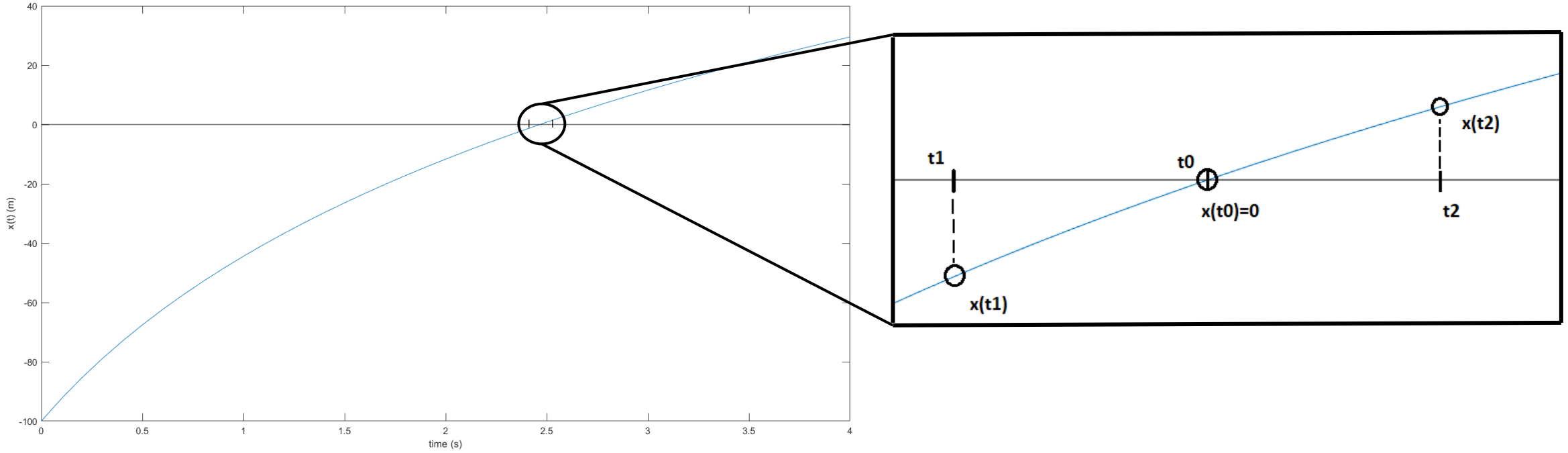
Función squeeze: agrupa todas las matrices columnas de posición para cada instante generadas por rungeKutta.

PUNTO DE IMPACTO (ImpactPoint.m)

- REGLA DE LA PALANCA: para una función f(x), hallamos la x_0 tal que f(x) = 0 de la siguiente forma:

$$x_0 = \frac{x_1 \bullet f(x_2) - x_2 \bullet f(x_1)}{f(x_2) - f(x_1)} \Longrightarrow t_0 = \frac{t_1 \bullet x(t_2) - t_2 \bullet x(t_1)}{x(t_2) - x(t_1)}$$

En nuestro caso, f(x) = x(t)



PUNTO DE IMPACTO (ImpactPoint.m)

REGLA DE LA PALANCA:

```
% Set vector of times
nt = size(r,2);                     % Find number of columns of r
t = linspace(0,tmax,nt);            % Set vector of times 't', with the same length as r
it = find(r(1,:)<0,1,'last');       % Find the index of r in wich there is the last negative value of x before zero

% Set it for when it = nt (it+1 doesn't exist)
if it == nt
    it = it -1;
end

% Lever rule
x1 = r(1,it);                                   % Last x before 0 (m)
x2 = r(1,it+1);                                 % First x after 0 (m)
t1 = t(it);                                     % Time for x = x1 (s)
t2 = t(it+1);                                   % Time for x = x2 (s)
t01 = (t1*x2-t2*x1)/(x2-x1);                    % Time for x = 0 (s)
v01 = (r(:,it+1)-r(:,it))/(t(it+1)-t(it));      % Velocity at x = 0 (m/s)
r01 = r(:,it)+v01*(t01-t(it));                  % Position vector at time t01 (m)
y = r01(2);                                     % Impact point in the y axis (when t = t01) (m)
z = r01(3);                                     % Impact point in the z axis (when t = t01) (m)

end
```

ÁNGULOS α_o Y β_o (bisectionRoot.m)

Creamos un programa para hallar los ángulos α_o y β_o óptimos para que impacte en el blanco (0,0,0) m.

```
% bisectionRoot.m
function [alpha0,beta0] = bisectionRoot(alpha01,alpha02,beta01,beta02,dangleMax,m,Cd,A,x0,Vtotal,tmax,dt)
% Find a root alpha0 and beta0 of function func, bracketed between alpha1
% and alpha2 & beta1 and beta2, respectively.
% In order to find the optimized angles (alpha0 and beta0) at which the
% object impacts in the required point (0,0,0), set an initial beta
% angle and find the alpha0 that satisfies z(alpha)=0. Then, set the
% last alpha0 as constant and find the beta0 that satisfies y(beta)=0.
% Repeat this process until the difference between alpha0 and its previous
% value is less than 1e-6. Do the same with beta0.
% Input:
%   m : mass of the object (kg)
%   A : area of the object (m^2)
%   Cd : drag coefficient
%   x0 : initial position in x-axis (m)
%   Vtotal : module of the initial velocity of the object (m/s)
%   tmax : total time of the trajectory (s)
%   dt : interval of time (s)
%   alpha1, alpha2 : lower and upper bounds of root (rad)
%   beta1, beta2 : lower and upper bounds of root (rad)
%   dangleMax  : max. error in value of root alpha0 and beta0 (rad)
% Output:
%   alpha0 : root of z(alpha), i.e. z(alpha0)=0
%   beta0 : root of y(beta), i.e. y(beta0)=0
```

ÁNGULOS α_o Y β_o (bisectionRoot.m)

```
% Set lastalpha and lastbeta for first iteration
lastalpha = inf;
lastbeta = inf;
error = 1e-6;                          % Set error

while true
if ~exist('beta0')    % Set beta angle for first iteration
    beta = pi/180;    % default value
else
    beta = beta0;     % Set beta0 as constant to find alpha0
end

% Set alpha1 and alpha2 as constant for every iteration
alpha1 = alpha01;
alpha2 = alpha02;

% Check bounds
[~,z1,~] = ImpactPoint(alpha1,beta,m,Cd,A,x0,Vtotal,tmax,dt);
[~,z2,~] = ImpactPoint(alpha2,beta,m,Cd,A,x0,Vtotal,tmax,dt);
```

Establecemos los primeros anteriores alfa y beta para la primera iteración

→ Establecemos β constante cualquiera para la primera iteración

→ Establecemos β_0 constante par las demás iteraciones

Establecemos los límites de alfa constantes para todas las iteraciones

Obtiene los puntos z1 = y(α_1, β) e z2 = y(α_2, β)

ÁNGULOS α_o Y β_o (bisectionRoot.m)

```
% Bisection iteration
while (abs(alpha2-alpha1)>dangleMax)
    alpha0 = (alpha1+alpha2)/2;
    [~,z0,~] = ImpactPoint(alpha0, beta,m,Cd,A,x0,Vtotal,tmax,dt);
    if (z0*z1>0)
        alpha1 = alpha0;
        z1 = z0;
    else
        alpha2 = alpha0;
        z2 = z0;
    end
end % while
alpha0 = (alpha1*z2-alpha2*z1) / (z2-z1);    % Find the root
alpha = alpha0;                              % Set alpha0 as constant to find beta0

% Set beta1 and beta2 as constant for every iteration
beta1 = beta01;
beta2 = beta02;

% Check bounds
[y1,~,~] = ImpactPoint(alpha, beta1,m,Cd,A,x0,Vtotal,tmax,dt);
[y2,~,~] = ImpactPoint(alpha, beta2,m,Cd,A,x0,Vtotal,tmax,dt);
```

Halla el ángulo α_o tal que $z(\alpha_o, \beta) = 0$

Establecemos el ángulo α_o hallado como constante para la siguiente iteración

Establecemos los límites de beta constantes para todas las iteraciones

Obtiene los puntos y1 = y(α, β_1) e y2 = y(α, β_2)

ÁNGULOS α_o Y β_o (bisectionRoot.m)

```
% Bisection iteration
while (abs(beta2-beta1)>dangleMax)
    beta0 = (beta1+beta2)/2;
    [y0,~,~] = ImpactPoint(alpha, beta0,m,Cd,A,x0,Vtotal,tmax,dt);
    if (y0*y1>0)
        beta1 = beta0;
        y1 = y0;
    else
        beta2 = beta0;
        y2 = y0;
    end
end % while
beta0 = (beta1*y2-beta2*y1) / (y2-y1);     % Find the root

% Stop iteration when the difference between alpha0 and its previous
% value is less than the error (same with beta0)
if abs(lastalpha-alpha0)<error && abs(lastbeta-beta0)<error
    break
end

% Set new lastalpha and lastbeta
lastalpha = alpha0;
lastbeta = beta0;
end % while true
end % function
```

Halla el ángulo β_o tal que $y(\alpha, \beta_2) = 0$

Para cuando la diferencia entre el alfa nuevo y el anterior es menor que el error establecido

Establece los nuevos 'lastalpha' y 'lastbeta' como los nuevos α_o y β_o obtenidos

TRAYECTORIA ÓPTIMA – TIRO BALÍSTICO (impact.m)

Creamos un script donde establecemos todos los datos y representaremos la trayectoria

```
%% DATA - BALLISTIC SHOT %%
clear all
m = 269e-3;          % Mass of the bullet (kg)
Vtotal = 300;        % Initial velocity of the bullet (m/s)
A = 1e-4;            % Section area of the bullet (m^2)
Cd = 0.1;            % Aerodinamic coefficient
tmax = 3.5;          % Maximum time of integration of the trajectory (s)
dt = 0.1;            % Time interval's length (s)
x0 = -1000;          % Initial position in the x-axis (m)

% Initial angles for bisectionRoot (rad)
alpha01 = 0;         % Lower bound of root (z(alpha))
alpha02 = pi/2;      % Upper bound of root (z(alpha))
beta01 = -pi/2;      % Lowerr bound of root (y(beta))
beta02 = +pi/2;      % Upper bound of root (y(beta))
dangleMax = 1e-6;    % Maximum error in value of root alpha0 and beta0 (rad)
```

TRAYECTORIA ÓPTIMA – TIRO BALÍSTICO (impact.m)

Encontramos los ángulos α_0 *y* β_0 con bisectionRoot.m

```
%% FIND alpha0 and beta0 %%
[alpha0,beta0] = bisectionRoot(alpha01,alpha02,beta01,beta02,dangleMax,m,Cd,A,x0,Vtotal,tmax,dt);
[y,z,t01] = ImpactPoint(alpha0,beta0,m,Cd,A,x0,Vtotal,tmax,dt);
% Print results
fprintf('Impact point in y-axis: %e m\n',y);
fprintf('Impact point in z-axis: %e m\n',z);
fprintf('Total time: %2.3f s\n',t01);
fprintf('Angle alpha0 formed by the direction of the shot with the xy plane is: %2.6f°\n',alpha0*(180/pi));
fprintf('Angle beta0 formed by the projection of the shot direction with the x-axis: = %2.6f°\n',beta0*(180/pi));
```

Integramos la trayectoria llamando a rungeKutta

```
%% FIND OPTIMIZED TRAJECTORY %%
% Set initial params
r0 = [x0;0;0];                                                          % Initial position (m)
v0 = [Vtotal*cos(alpha0)*cos(beta0); Vtotal*cos(alpha0)*sin(beta0); Vtotal*sin(alpha0)];  % Initial velocity (m/s)

% Integrate trajectory
myforce = @(r,v,t) force(m,A,Cd,r,v,Vwind(r));
[r] = rungeKutta(myforce,m,r0,v0,dt,tmax);
```

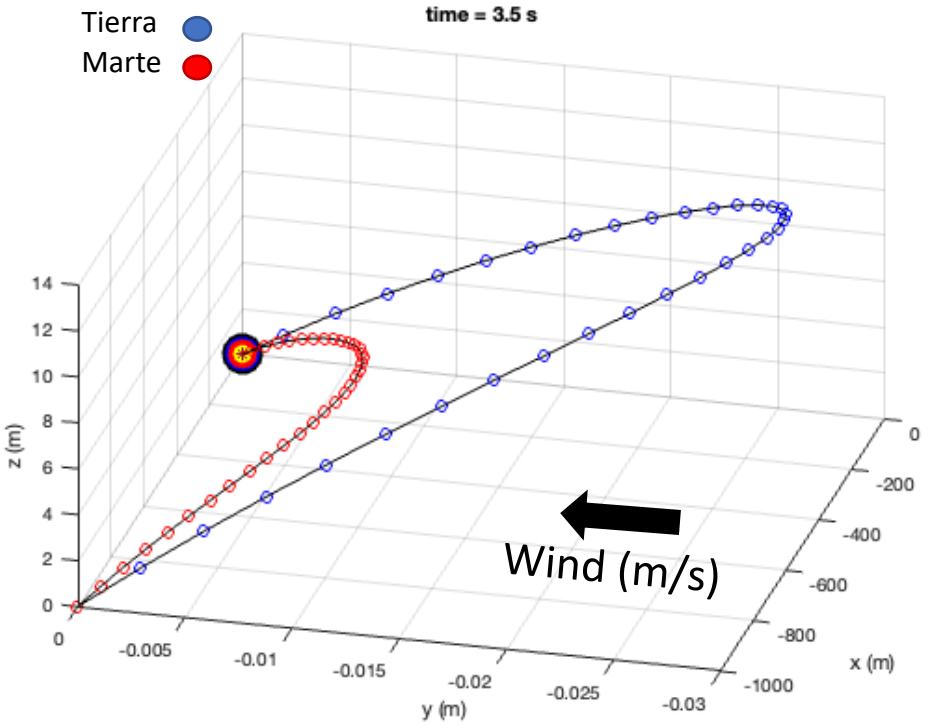
TRAYECTORIA ÓPTIMA – TIRO BALÍSTICO (impact.m)

```
%% PLOT THE TRAJECTORY %%
% Plot as a video
figure(1)
nt = size(r,3);
t = linspace(0,tmax,nt);
for it = 1:nt
    plot3(r(1,:,it),r(2,:,it),r(3,:,it),'ro'); hold on
    % Plot target
    scatter3(0,0,0,500,'filled','k'); hold on
    scatter3(0,0,0,350,'filled','b'); hold on
    scatter3(0,0,0,250,'filled','r'); hold on
    scatter3(0,0,0,100,'filled','y'); hold on
    xlabel('x (m)')
    ylabel('y (m)')
    zlabel('z (m)')
    grid on
    xlim([x0,0])
    title(['time = ',num2str(t(it)),' s'])
    pause(dt)
end

% Find optimise time for ploting
it = find(t>t01,1,'first');

% Plot total trajectory
r = squeeze(r);              % Get all the columns together in a matrix
plot3(r(1,1:it),r(2,1:it),r(3,1:it),'k-');
hold on
plot3(0,0,0,'*');
```

Diana



TRAYECTORIA ÓPTIMA – TIRO BALÍSTICO (impact.m)

RESULTADOS:

Disparo en la TIERRA

$$g_T = 9.8166\ m/s^2$$

$$\rho_{air} = 1{,}23\ Kg/m^3$$

```
Impact point in y-axis: -4.770490e-17 m
Impact point in z-axis: 4.551914e-14 m
Total time: 3.377 s
Angle alpha0 formed by the direction of the shot with the xy plane is: 3.179948°
Angle beta0 formed by the projection of the shot direction with the x-axis: = -0.005223°
```

Disparo en MARTE

$$g_M = 3.7120\ m/s^2$$

$$\rho_{air} = 1{,}30\ Kg/m^3$$

```
Impact point in y-axis: -2.363561e-17 m
Impact point in z-axis: 2.220446e-16 m
Total time: 3.375 s
Angle alpha0 formed by the direction of the shot with the xy plane is: 1.201403°
Angle beta0 formed by the projection of the shot direction with the x-axis: = -0.001658°
```

TRAYECTORIA ÓPTIMA – TIRO CON ARCO (impact.m)

Creamos un script donde establecemos todos los datos y representaremos la trayectoria

```
%% DATA - ARCHERY %%
clear all
m = 20e-3;          % Mass of the arrow (kg)
Vtotal = 80;        % Initial velocity of the arrow (m/s)
A = 40e-4;          % Section area of the arrow (m^2
Cd = 0.1;           % Aerodinamic coefficient
tmax = 4;           % Maximum time of integration of the trajectory (s)
dt = 0.1;           % Time interval's length (s)
x0 = -100;          % Initial position in the x-axis (m)

% Initial angles for bisectionRoot (rad)
alpha01 = 0;          % Lower bound of root (z(alpha))
alpha02 = pi/2;       % Upper bound of root (z(alpha))
beta01 = -pi/2;       % Lowerr bound of root (y(beta))
beta02 = +pi/2;       % Upper bound of root (y(beta))
dangleMax = 1e-6;     % Maximum error in value of root alpha0 and beta0 (rad)
```

TRAYECTORIA ÓPTIMA – TIRO CON ARCO (impact.m)

Encontramos los ángulos α_0 *y* β_0 con bisectionRoot.m

```
%% FIND alpha0 and beta0 %%
[alpha0,beta0] = bisectionRoot(alpha01,alpha02,beta01,beta02,dangleMax,m,Cd,A,x0,Vtotal,tmax,dt);
[y,z,t01] = ImpactPoint(alpha0,beta0,m,Cd,A,x0,Vtotal,tmax,dt);
% Print results
fprintf('Impact point in y-axis: %e m\n',y);
fprintf('Impact point in z-axis: %e m\n',z);
fprintf('Total time: %2.3f s\n',t01);
fprintf('Angle alpha0 formed by the direction of the shot with the xy plane is: %2.6f°\n',alpha0*(180/pi));
fprintf('Angle beta0 formed by the projection of the shot direction with the x-axis: = %2.6f°\n',beta0*(180/pi));
```

Integramos la trayectoria llamando a rungeKutta

```
%% FIND OPTIMIZED TRAJECTORY %%
% Set initial params
r0 = [x0;0;0];                                                          % Initial position (m)
v0 = [Vtotal*cos(alpha0)*cos(beta0); Vtotal*cos(alpha0)*sin(beta0); Vtotal*sin(alpha0)];  % Initial velocity (m/s)

% Integrate trajectory
myforce = @(r,v,t) force(m,A,Cd,r,v,Vwind(r));
[r] = rungeKutta(myforce,m,r0,v0,dt,tmax);
```

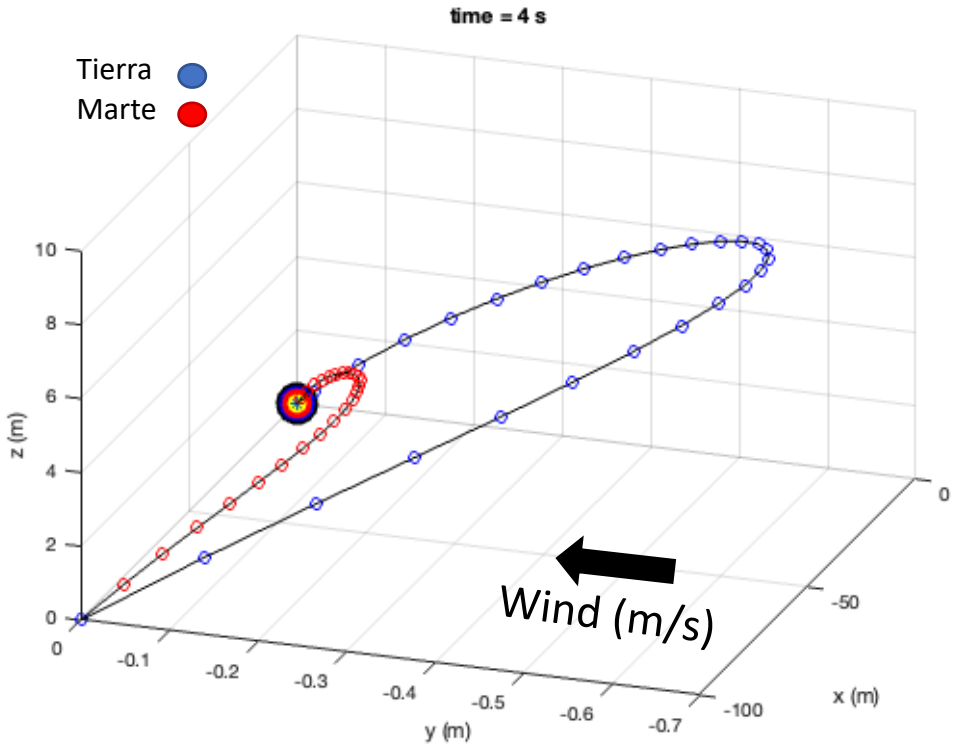
TRAYECTORIA ÓPTIMA – TIRO CON ARCO (impact.m)

```
%% PLOT THE TRAJECTORY %%
% Plot as a video
figure(1)
nt = size(r,3);
t = linspace(0,tmax,nt);
for it = 1:nt
    plot3(r(1,:,it),r(2,:,it),r(3,:,it),'ro'); hold on
    % Plot target
    scatter3(0,0,0,500,'filled','k'); hold on
    scatter3(0,0,0,350,'filled','b'); hold on
    scatter3(0,0,0,250,'filled','r'); hold on
    scatter3(0,0,0,100,'filled','y'); hold on
    xlabel('x (m)')
    ylabel('y (m)')
    zlabel('z (m)')
    grid on
    xlim([x0,0])
    title(['time = ',num2str(t(it)),' s'])
    pause(dt)
end

% Find optimise time for ploting
it = find(t>t01,1,'first');

% Plot total trajectory
r = squeeze(r);              % Get all the columns together in a matrix
plot3(r(1,1:it),r(2,1:it),r(3,1:it),'k-');
hold on
plot3(0,0,0,'*');
```

Diana



TRAYECTORIA ÓPTIMA – TIRO CON ARCO (impact.m)

RESULTADOS:

Disparo en la TIERRA

$$g_T = 9.8166\ m/s^2$$

$$\rho_{air} = 1{,}23\ Kg/m^3$$

```
Impact point in y-axis: -2.021300e-14 m
Impact point in z-axis: -3.297161e-09 m
Total time: 2.547 s
Angle alpha0 formed by the direction of the shot with the xy plane is: 12.577226°
Angle beta0 formed by the projection of the shot direction with the x-axis: = -0.941738°
```

Disparo en MARTE

$$g_M = 3.7120\ m/s^2$$

$$\rho_{air} = 1{,}30\ Kg/m^3$$

```
Impact point in y-axis: -3.703635e-15 m
Impact point in z-axis: 7.469025e-13 m
Total time: 2.580 s
Angle alpha0 formed by the direction of the shot with the xy plane is: 4.886793°
Angle beta0 formed by the projection of the shot direction with the x-axis: = -0.259168°
```

BIBLIOGRAFÍA

- https://es.wikipedia.org/wiki/Aire (Valores densidad del aire en función de la temperatura)
- https://es.wikipedia.org/wiki/Resistencia_aerodin%C3%A1mica#:~:text=Se%20denomina%20forma%20aerodin%C3%A1mica%2C%20o,del%20cuerpo%20respecto%20del%20medio.&text=En%20el%20caso%20del%20agua,ejemplo%2C%20se%20denomina%20forma%20hidrodin%C3%A1mica. (Fuerza debida al viento)
- https://es.wikipedia.org/wiki/Gravedad (Fuerza gravitatoria)