Simulando la aerodinámica de una esfera en un tiro parabólico

¿Es siempre el ángulo de rango máximo 45^{o} en un tiro parabólico con fricción?

Evaluación Interna Física NS

Índice

1. Introducción

Es bien sabido que el aire de la atmósfera se resiste a la velocidad de un objeto. Un ejemplo común es la forma característica que debe tener el tren de alta velocidad en España, el AVE, el cual está diseñado para velocidades de hasta 90 ms^{-1}. Un caso más pronunciado, pero menos cotidiano para un ser humano común es el mismo efecto pero en el agua. Cuando era niño en una piscina rompí una raqueta de madera porque apliqué una velocidad perpendicular a la superficie de la raqueta, creando una fuerza de resistencia suficientemente grande para romperla. Observé también que si la velocidad era paralela al plano de la raqueta, esta mostraba una menor fuerza de resistencia. En esta interna se explorarán los efectos que tienen los fluidos en los cuerpos que se mueven en ellos y se simulará mediante un programa informático propio la trayectoria de un tiro oblicuo, así como sus propiedades. Se seguirá como pregunta de investigación principal

Figura 1: AVE modelo S102, Autor: Savh, extraída de Wikimedia Commons

¿Es siempre el ángulo de rango máximo 45^o en un tiro parabólico con fricción?

donde el rango es la distancia de recorrido tras el tiro en el eje OX.

2. Exploración

Lo primero que uno se encuentra al aceptar la necesidad de considerar la resistencia del aire, o la de cualquier fluido, es el número de Reynolds

$$\text{Re} = \frac{\rho v D}{\mu}$$

donde ρ es la densidad del fluido, v la rapidez a la que viaja un objeto, D el diámetro de la tubería (a veces imaginaria) por donde viaja el objeto y μ es la viscosidad dinámica del fluido (Rott 1990). Además de para el estudio del tipo de flujo de un líquido en una tubería, el número de Reynolds también se usa en el estudio del movimiento de un cuerpo en el seno de un fluido. Con diferentes números de Reynolds, los modelos de la fuerza de resistencia aerodinámica varían. Es sabido que un objeto en movimiento en un fluido experimenta una fuerza de resistencia en la misma dirección que la velocidad pero sentido contrario, como ocurre para la fricción entre sólidos. Lo que es más complejo y no tan conocido es el valor de esa fuerza F_D.[1] Para $\text{Re} < 1$ se cumple la Ley de Stokes (Shearer y Hudson 2008) donde formula que para una esfera

$$F_D = 6\pi\mu r v \tag{1}$$

siendo r el radio de esa esfera y v la velocidad. Esta es la base del funcionamiento de un viscosímetro.

Para $\text{Re} > 1000$ en cambio se aplica la ecuación de resistencia es

$$F_D = \frac{1}{2}\rho A C_D v^2 \tag{2}$$

donde A es el área de la sección transversal perpendicular a la velocidad y C_D es el coeficiente de resistencia aerodinámico. Este último depende de la forma del objeto pero también del número de Reynolds, y, por lo tanto, de la velocidad (Bragg, Zuiden y Hermance 1974). Comúnmente se neglige esa relación con el número de Reynolds y se usa un solo coeficiente en concreto. Por ejemplo, se define que para una esfera, $C_D = 0.47$ para Re de 10^3 a 10^5 (Maroto, Duenas-Molina y Dios 2005).

[1] El subíndice $\square_D$ simboliza el arrastre, *Drag* en inglés

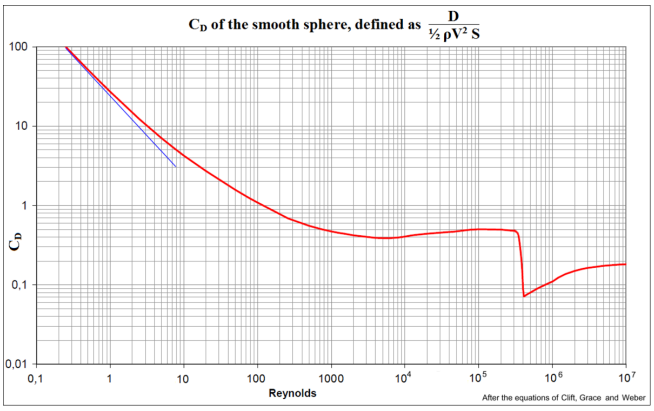


Figura 2: C_D sobre Re de una esfera, Autor: B. de Go Mars, extraída de Wikimedia Commons

La línea azul de la figura 2 tiene forma hiperbólica [2] de $C_D = 24/Re$. De hecho, si se substituye en la ecuación 2 ese valor de C_D, se llega a la Ley de Stokes, la ecuación 1. $24/Re$ es una aproximación aceptable para Re < 1. Para $1 <$ Re < 1000 no existe una aproximación clara, ya que C_D pasa de ser hiperbólico a constante.

Para llegar al tiro parabólico es necesario dejar a un lado la ley de Stokes y centrarse en la ecuación 2. Es por eso que la ecuación 2 será la base de la simulación. La razón es por el orden del número de Reynolds. Para por ejemplo una esfera de $D = 10$ cm en el aire, tenemos que Re $= 7085v$, y por lo tanto, siempre que $v > 0.14$ ms^{-1} se cumple que Re > 1000.

3. Simulando la resistencia aerodinámica

3.1. Posibilidad analítica

En un mundo de una dimensión, es posible encontrar una ecuación analítica para $v(t)$, la velocidad respecto el tiempo. Siguiendo la segunda ley de Newton

$$a_D = \frac{1}{m} F_D$$

donde m es la massa del cuerpo, a_D la aceleración de arrastre aerodinámico y F_D la fuerza de esa acceleración. Si se tiene en cuenta la gravedad, siendo negativa yendo hacia abajo, entonces la fuerza de resistencia es hacia arriba y por lo tanto positiva para el descenso del objeto. Así pues se llega a que

$$a = -g + a_D$$

siendo a la aceleración total del cuerpo y g la aceleración causada por la gravedad. Lo que es lo mismo, siguiendo la ecuación 2

$$\frac{\mathrm{d}v}{\mathrm{d}t} = -g + \frac{1}{2m} \rho A C_D v^2 \tag{3}$$

donde v es la velocidad del objeto, la cual es variable con el tiempo t. Para resolver esta ecuación diferencial ordinaria no lineal de primer orden en concreto y las expuestas posteriormente se usarán las funciones trigonométricas tanh, arctanh, tan, coth y arccoth. El resultado de la ecuación 3 queda como

$$v(t) = -\sqrt{\frac{2mg}{\rho C_D A}} \tanh\left(\sqrt{\frac{g\rho C_D A}{2m}} t + K\right)$$

[2] Si bien aparenta una línea recta, como la escala es logarítmica, en realidad es una hipérbola.

Y sabiendo que $v(0) = v_0$ puesto que, en general, se parte de una velocidad inicial (vertical hacia abajo, i.e. $v_0 < 0$, de momento), se puede afirmar, siguiendo el modelo de la ecuación 2, que

$$v(t) = -\sqrt{\frac{2mg}{\rho C_D A}} \tanh\left(\sqrt{\frac{g\rho C_D A}{2m}}t - \operatorname{arctanh}\left(\sqrt{\frac{\rho C_D A}{2mg}}v_0\right)\right); |v_0| < \sqrt{\frac{2mg}{\rho C_D A}} \tag{4}$$

No obstante, aún hay que tener otra cosa en cuenta. El ascenso y el descenso son distintos, y la fuerza F_D tiene signos negativo y positivo respectivamente. En la ecuación 3 se tiene en cuenta que v y g tienen el mismo signo, y por eso a_D tiene el signo inverso. Pero se puede dar el caso que $v > 0$, y que por lo tanto a_D y g deban los dos restar. Esto implica que 4 solo aplique cuando $-\sqrt{\frac{2mg}{\rho C_D A}} < v_0 \leq 0$. De la misma manera, para resolver

$$\frac{\mathrm{d}v}{\mathrm{d}t} = -g - \frac{1}{2m}\rho A C_D v^2$$

se llega a que

$$v(t) = -\sqrt{\frac{2mg}{\rho C_D A}} \tan\left(\sqrt{\frac{g\rho C_D A}{2m}}t - \arctan\left(\sqrt{\frac{\rho C_D A}{2mg}}v_0\right)\right); v(t) > 0 \tag{5}$$

Con las ecuaciones 4 y 5 se cubren los casos donde $v(t) > 0$ (porque $v_0 > 0$), y $-\sqrt{\frac{2mg}{\rho C_D A}} < v_0 \leq 0$. Si por ejemplo $v_0 > 0$ pero $v(t) \leq 0$ entonces debe existir otra ecuación. Justamente esta ecuación no será más que una translación de la ecuación 4, ya que se puede entender como que $v_0 = 0$ en el momento que se llega al máximo de altura y luego empieza a bajar desde allí. Se deberá desplazar a la derecha el zero de la función 5, así que queda

$$v(t) = -\sqrt{\frac{2mg}{\rho C_D A}} \tanh\left(\sqrt{\frac{g\rho C_D A}{2m}}t - \arctan\left(\sqrt{\frac{\rho C_D A}{2mg}}v_0\right)\right); v_0 > 0, v(t) < 0 \tag{6}$$

Finalmente, para el caso donde $v_0 < -\sqrt{\frac{2mg}{\rho C_D A}}$, lo que implica hacia abajo con rapidez que supera la terminal, se debe emplear una ecuación diferente. Si bien es cierto que la ecuación 4 da una solución a la ecuación diferencial, no es la única. Existe otra, donde

$$v(t) = -\sqrt{\frac{2mg}{\rho C_D A}} \coth\left(\sqrt{\frac{g\rho C_D A}{2m}}t - \operatorname{arccoth}\left(\sqrt{\frac{\rho C_D A}{2mg}}v_0\right)\right); v_0 < -\sqrt{\frac{2mg}{\rho C_D A}} \tag{7}$$

La diferéncia entre esta y la ecuación 4 es el dominio de las funciones arccoth y arctanh. Las dos cumplen la definición de sus respectivas derivadas, pero con dominios distintos.

Con estas 4 ecuaciones ya se puede modelar en una dimensión la fuerza aerodinámica, para todo v_0 y $v(t)$.

Si por ejemplo se tiene una esfera de PVC macizo de 1 cm de radio en caída libre en un entorno acuático, tendría una gráfica $v(t)$ como la figura 3 a diferentes v_0. Esta figura de origen propio se ha creado siguiendo las ecuaciones descritas anteriormente junto con el programa informático MATLAB, con la siguiente función como base:

```
function plotVel(k,g,v_0,t_max,col)
if(v_0 > 0)
    v_r = @(t) -sqrt(g/k)*tan(sqrt(k*g)*t-atan(sqrt(k/g)*v_0));
    fplot(v_r,[0,min(t_max,atan(sqrt(k/g)*v_0)/sqrt(k*g))],'Color',col);
    if atan(sqrt(k/g)*v_0)/sqrt(k*g) < t_max
    hold on
    v = @(t) -sqrt(g/k)*tanh(sqrt(k*g)*t-atan(sqrt(k/g)*v_0));
    fplot(v,[atan(sqrt(k/g)*v_0)/sqrt(k*g),t_max],'Color',col,'HandleVisibility','off
    ');
    end
elseif v_0<-sqrt(g/k)
```

```
    v_2 = @(t) -sqrt(g/k)*coth(sqrt(k*g)*t-acoth(sqrt(k/g)*v_0));
    fplot(v_2,[0 t_max],'Color',col);
else
    v_n = @(t) -sqrt(g/k)*tanh(sqrt(k*g)*t-atanh(sqrt(k/g)*v_0));
    fplot(v_n,[0 t_max],'Color',col);
end
end
```

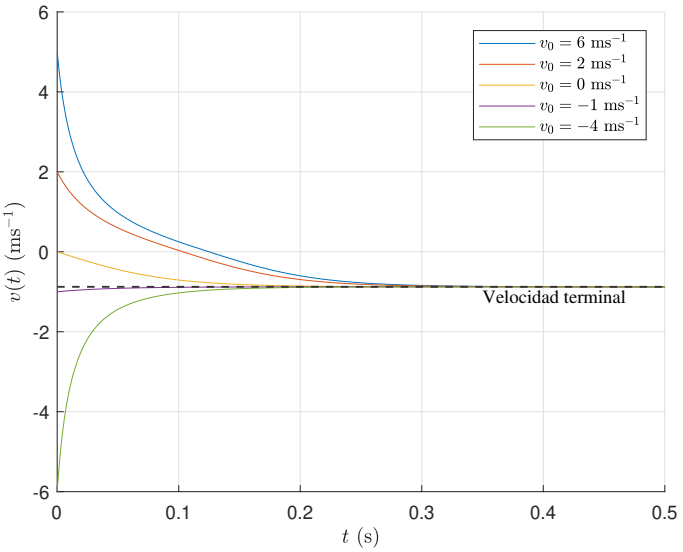


Figura 3: Velocidad de una esfera en caída libre con velocidad inicial

$v(t)$ tiende a una constante cuando $t \to \infty$, donde F_D es tan grande como su peso. Concretamente esa constante se llama velocidad terminal, y tiene fórmula[3]

$$v_t = -\sqrt{\frac{2mg}{\rho C_D A}}$$

ya que el límite cuando $x \to \infty$ de $\tanh(x)$ y $\coth(x) = 1$, lo cual corresponde a cuando $a_D = g$ en la ecuación 3.

Se debe tener en cuenta que para estas soluciones analíticas se menosprecia la altura. Por ejemplo para la atmósfera de la Tierra a 200 m la aceleración de la gravedad ha disminuido un 0.006 %, la densidad de la atmósfera un 1.88 %, y la viscosidad dinámica un 0.35 % (ToolBox 2005). A esa distancia se pueden asumir g, ρ y C_D (derivado de μ) constantes. Para modelar un objeto que debe ir a una altura considerablemente más grande se debería tener en cuenta cómo esos parámetros varían.

3.2. Aproximando a dos dimensiones

Para responder a la pregunta de investigación es necesario modelar con 2 dimensiones, no obstante el precio a pagar con esta dimensión de más es la incapacidad de resolver la ecuación $v(t)$ analíticamente. La idea es encontrar $v_x(t)$ y $v_y(t)$ mediante un programa informático escrito en Processing (Java) el cual se puede consultar en el apartado del apéndice A.1. Para empezar se tiene el diagrama de cuerpo libre del objeto que se quiere estudiar (omitiendo la gravedad).

[3] Alternativamente se acostumbra a presentar sin el signo negativo.

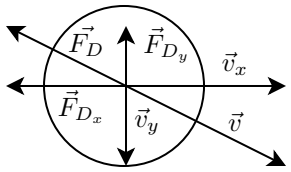


Figura 4: $\vec{v}$ y $\vec{F}_D$ de un cuerpo

De la figura 4 se puede encontrar que

$$\frac{v}{|v_x|} = \frac{F_D}{|F_{D_x}|}$$

aislando para F_{D_x}

$$F_{D_x} = -F_D \frac{v_x}{v} = -\frac{1}{2}\rho A C_D v_x v$$

donde $v = \sqrt{v_x^2 + v_y^2}$, puesto que v_x y v_y son las componentes del vector $\vec{v}$. Por lo tanto,

$$F_{D_x} = -\frac{1}{2}\rho A C_D v_x \sqrt{v_x^2 + v_y^2}$$

F_{D_y} se encuentra de una manera similar. Teniendo en cuenta que para a_y se debe restar el valor de g, con dos dimensiones espaciales, para representar lo mismo que antes en la ecuación 3 nos queda un sistema de ecuaciones diferenciales:

$$\begin{cases} \dfrac{\mathrm{d}v_x}{\mathrm{d}t} = -\dfrac{1}{2m}\rho A C_D v_x \sqrt{v_x^2 + v_y^2} \\ \dfrac{\mathrm{d}v_y}{\mathrm{d}t} = -g - \dfrac{1}{2m}\rho A C_D v_y \sqrt{v_x^2 + v_y^2} \end{cases} \tag{8}$$

Es este sistema el que no tiene solución analítica (Nave 1998). La razón por la cual no se distingue entre subida o bajada, o sea, restar F_D o sumar, es porque el signo ya está presente dentro de la variable. El término $v_x\sqrt{v_x^2 + v_y^2}$ será positvo si $v_x > 0$ y negativo si $v_x < 0$. El programa en Java aproximará numéricamente el sistema. Para ello se emplea el método de integración de Euler, donde se discreta el tiempo con pequeños incrementos. Al ser un algoritmo, es relativamente sencillo utilizar un programa que lo logre. Discretando se llega a que

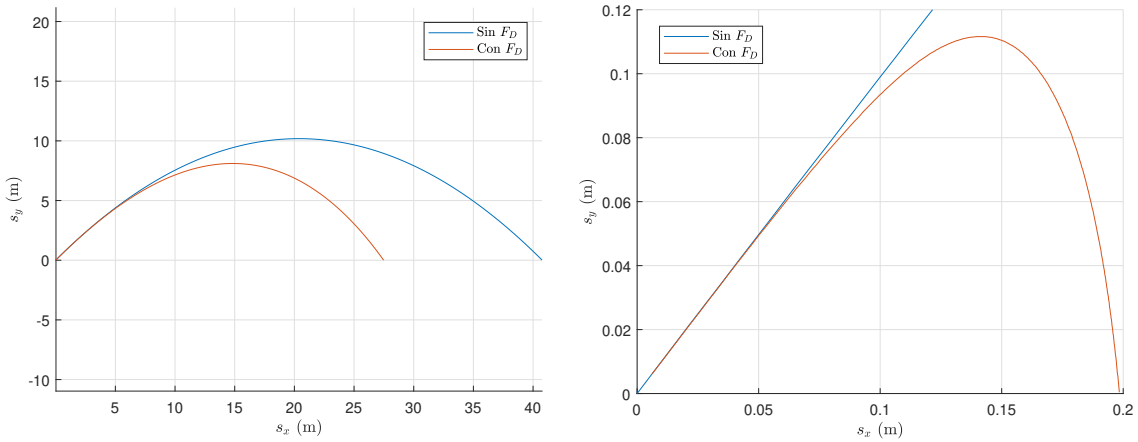
$$\begin{cases} v_{x_{n+1}} = v_{x_n} - \dfrac{1}{2m}\rho A C_D v_{x_n} \sqrt{v_{x_n}^2 + v_{y_n}^2}\,\Delta t \\ v_{y_{n+1}} = v_{y_n} + \left(-g - \dfrac{1}{2m}\rho A C_D v_{y_n} \sqrt{v_{x_n}^2 + v_{y_n}^2}\right)\Delta t \\ s_{x_{n+1}} = s_{x_n} + v_{x_n}\Delta t \\ s_{y_{n+1}} = s_{y_n} + v_{y_n}\Delta t \end{cases} \tag{9}$$

Y en el código

```
1  void update(float dt){
2
3     // Actualizar primero el vector pos para que al sobreescribir vel no integre por
      arriba
4     pos.x+=vel.x*dt;
5     pos.y+=vel.y*dt;
6     // k = 1/(2*m) * rho * A * C_D
7     float dv_x = -vel.x*sqrt(vel.x*vel.x + vel.y * vel.y)*k;
8     float dv_y = - container.gravity  -vel.y*sqrt(vel.x*vel.x + vel.y * vel.y)*k;
9
10    vel.x+=dv_x*dt;
11    vel.y+=dv_y*dt;
12 }
```

Concretamente este método es análogo a integrar haciendo rectángulos *por abajo*, formalmente, suma de Riemann por la izquierda.

Teniendo $s_{x_0} = s_{y_0} = 0$ y $v_{x_0} = v\cos\theta$, $v_{y_0} = v\sin\theta$, donde v y θ son la rapidez y el ángulo iniciales, ya se puede aproximar mediante un programa informático. Para la misma esfera del ejemplo anterior pero en la atmósfera terrestre, partiendo de que $v = 20$ ms^{-1}, $\theta = \frac{\pi}{2}$ rad y $\Delta t = (600 \text{ Hz})^{-1}$ (o $0.001\overline{6}$ s), su trayectoria forma una curva ya no tan parecida a la de una parábola (Fig. 5a). Su rango y máximo de altura son intuitivamente menores. Además, hay cierta asimetría entre el ascenso y el descenso. Se puede exacerbar esa asimetría con valores de F_D propios del agua. Cabe mencionar que en este caso (y cualquiera con esta fricción) ya no se habla de un tiro parabólico, puesto que la trayectoria del proyectil no es una parábola.



(a) En el aire

(b) En el agua

Figura 5: Trayectoria de un proyectil con y sin fricción

La gráfica 5b ilustra cómo, en el descenso, la velocidad se desvía *antes y más* de la parábola debido a fuerzas de resistencia aerodinámica grandes. La resistencia según la componente x de la velocidad se reduce casi a cero, anulando la componente x de la aceleración, mientras que la componente y de la resistencia aerodinámica aumenta debido a la aceleración de la gravedad. Sin embargo, este caso, que se refiere al agua, no es realista ya que no considera la fuerza de empuje de Arquímedes.

De la gráfica 5a deducimos que el rango es menor simplemente porque la trayectoria naranja cruza en eje $y = 0$ para un valor de x menor que el azul. Asimismo, vemos que la altura máxima es también menor. Cabe puntualizar que esta gráfica no muestra los tiempos de ascenso, descenso ni total. Sin embargo, observamos que, claramente, en el descenso la gráfica naranja decrece de manera más rápida que la azul.

Para interpretar las diferencias entre las trayectorias se puede recurrir a la derivada, la cual ha sido calculada usando incrementos (partiendo de la gráfica 5b).

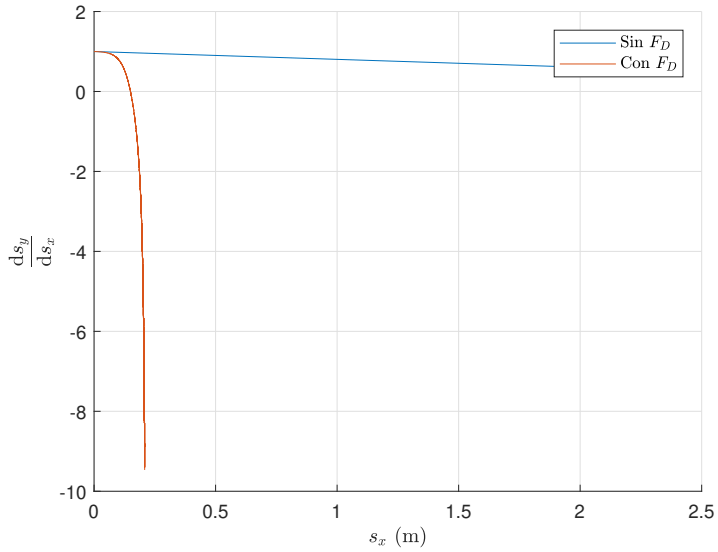


Figura 6: Derivada de la trayectoria de un proyectil ($\partial_{s_x} s_y$)

En este caso la figura 6 no es, por ejemplo, una representación de la velocidad respecto el tiempo, puesto que se deriva la función discreta $s_y(s_x)$ respecto s_x de la misma manera que se derivaría la función $\sin(x)$ respecto x obteniendo $\cos(x)$. La idea de esto es conseguir una visión del incremento de la función, la cual es apreciable al compararla con la de una parábola, que debe ser una recta.

El código de Matlab usado para el cálculo de la derivada es el siguiente

```
function dydx = discrete_derivative(x,y)
n = length(x);
dydx = zeros(1,n-1);
for i = 1:n-1
    dydx(i) = (y(i+1) - y(i)) / (x(i+1) - x(i));
end
end
```

Al principio sí que aparenta una recta, no obstante, decrece más rápidamente que la recta azul. Este comportamiento corresponde a el poco incremento de s_x el cual ya ha sido mencionado. Como

$$\int_0^{s_x} \partial_{s_x} s_y \mathrm{d}s_x = s_y = 0$$

sea u ese valor que cumple que

$$\int_0^{u} \partial_{s_x} s_y \mathrm{d}s_x = -\int_u^{s_x} \partial_{s_x} s_y \mathrm{d}s_x$$

u es ese valor de s_x donde hay el máximo en s_y, de otra manera, que $\partial_{s_x} s_y$ cruza $x = 0$. En la figura 6 se aprecia como u está más cerca de s_x (el punto donde ya no hay más valores para la función) que de 0, y por lo tanto, en menos distancia horizontal recorre la misma vertical, así pues, el descenso debe ocurrir más rápidamente que el ascenso. La cuestión es que siempre $u \geq \frac{s_x}{2}$ porque $\partial_{s_x} s_y$ decrece más rápido que la línea recta azul.

En la ecuación 9 se depende de 4 constantes. Es común agruparlas todas en una sola, un constante k para poder modificarla artificialmente. Si bien es cierto que C_D no es exactamente una constante, puesto que depende de Re, que depende a su vez de v, se asumirá su invariancia. Así pues se define que $k = \frac{1}{2m}\rho A C_D$, lo que queda como

$$a_D = kv^2$$

Esta misma k es la k usada en la función update(float dt) del código. Para el ejemplo de la figura 5a, $k = 0.016$. Cuando no hay fricción $k = 0$ A partir de ahora se hablará de resistencia aerodinámica de orden k para referirse a F_D.

3.3. Propiedades derivables de la experimentación

Tras la experimentación con la aproximación de la trayectoria se puede responder a la pregunta inicial respecto a sus propiedades. Se puede hallar la relación entre el rango y el ángulo de tiro en el caso del tiro parabólico, se sobreentiende que, sin resistencia aerodinámica.

$$s_x(\theta, v) = \frac{v^2}{g}\sin(2\theta) \tag{10}$$

Esto se encuentra partiendo de las ecuaciones de MRUA

$$x = v\cos\theta t$$

$$y = v\sin\theta t - \frac{1}{2}gt^2$$

$$y = 0 \Rightarrow \frac{1}{2}gt^2 = v\sin\theta t$$

$$\frac{1}{2}gt = v\sin\theta; t \neq 0$$

$$t = \frac{x}{v\cos\theta}$$

$$\frac{1}{2}g\frac{x}{v\cos\theta} = v\sin\theta$$

$$x = \frac{1}{g}v^2 2\cos\theta\sin\theta = \frac{v^2}{g}\sin 2\theta$$

Restringiendo $0 \leq \theta \leq \frac{\pi}{2}$ se puede encontrar fácilmente que el máximo de $s_x(\theta, v)$ es para $\theta = \frac{\pi}{4}$.

$$\frac{\partial s_x}{\partial\theta} = 0$$

$$2\frac{v^2}{g}\cos 2\theta = 0$$

$$\cos 2\theta = 0$$

$$\theta = \frac{1}{2}\operatorname{arc}\cos 0 = \frac{\pi}{4}$$

Puesto que v y g solo escalan el gráfico, estos no interfieren en el máximo de la función. Si la rapidez es el doble, para el mismo ángulo, el rango se cuadripilca. Por otro lado, es inversamente proporcional a g; por ejemplo: dado v en la Luna donde g es aproximadamente 1/6 de la terrestre, el alcance o rango se multiplicaría por 6. No obstante, esto no debe ser necesariamente igual para $k \neq 0$. Según el programa usado, $s_x(\frac{\pi}{4}, 20) = 40.7886$ m con $\Delta t = 10^{-7}$ para $k = 0$, lo que es un error de aproximación del $3 \cdot 10^{-6}$ % respecto al valor analítico de la ecuación 10.

Construyamos una función $\varsigma(\theta, v, k)$ que retorne el rango del proyectil a un ángulo θ, con rapidez v y con un orden de fricción k. $\varsigma(\theta, v, 0)$ debe ser igual a $s_x(\theta, v)$. Para $\theta = \frac{\pi}{4}$, $1 \leq v \leq 20$ ms^{-1} y $0 \leq k \leq 1$, la función $\varsigma(\theta, v, k)$ tiene la forma de la figura 7.

El equivalente en el programa es

```
1 double range (double vel, double theta, double k, double g) {
2
3   double dt = 1e-7d;
4   Container c = new Container(0, g); // Simplemente contiene la gravedad y la
      densidad. La densidad es 0 porque ahora se define respecto k en el objeto Body b
5   Body b = new Body(new PVector_d(0, 0), PVector_d.fromAngle(-radians(theta)).mult(
      vel), k, c);
6   b.update(dt); // Se ejecuta una vez para que b.pos.y < 0 y no = 0
7   while (b.pos.y<0)b.update(dt); // Se recurre hasta que b.pos.y pase el 0
8   return b.pos.x;
9 }
```

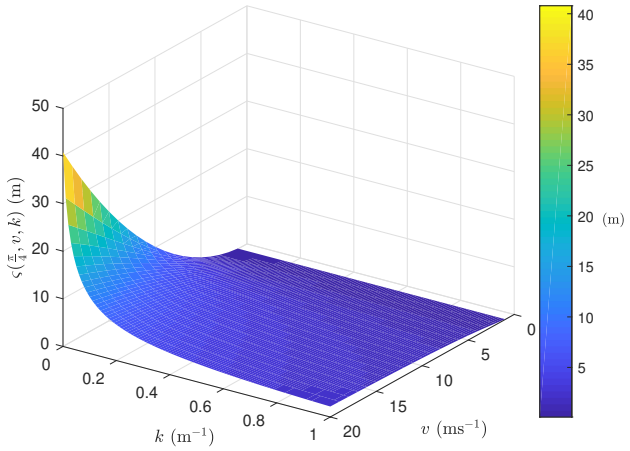
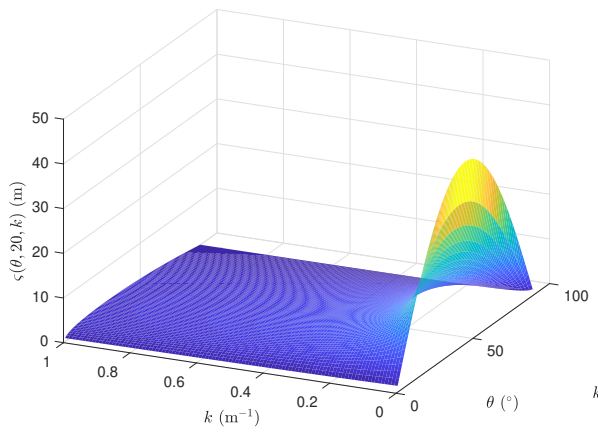
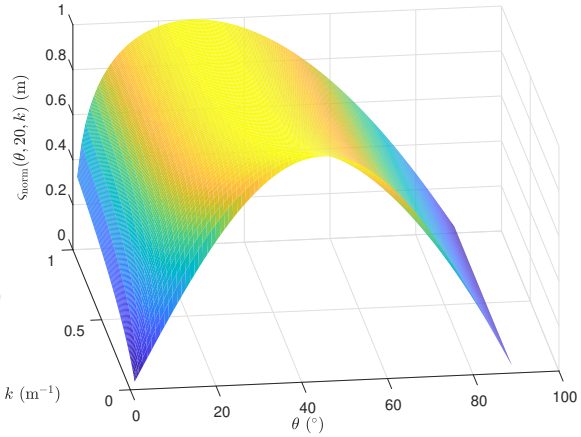


Figura 7: $\varsigma(\frac{\pi}{4}, v, k)$, $1 \leq v \leq 20$, $0 \leq k \leq 1$

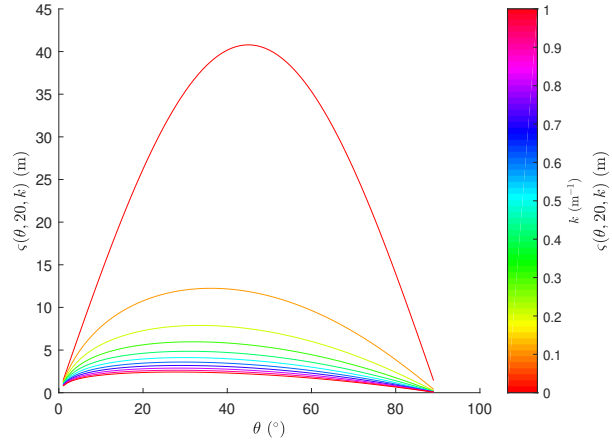
Como resulta deducible, una menor velocidad inicial y una mayor fricción contribuyen a que el rango del proyectil disminuya. Si bien esta función ς no aporta visualmente algún detalle significativo, es necesario para hallar el alcance en la función del ángulo de tiro, lo que se hará a continuación para una rapidez inicial v dada.



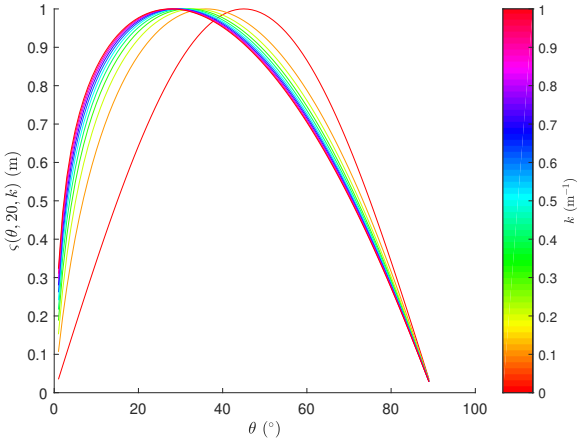
(a) ς cruda



(b) ς normalizada



(c) ς cruda en 2 dimensiones



(d) ς normalizada en 2 dimensiones

Figura 8: $\varsigma(\theta, 20, k)$, $0 \leq \theta \leq 90°$, $0 \leq k \leq 1$

La figura 8a muestra el rango en función del ángulo inicial de un proyectil con k variable. Representa la misma función que la fig. 7 pero esta vez v es constante, mientras que en el otro gráfico θ es constante. A parte de la notable disminución general del rango, también disminuye el ángulo para el rango máximo, que para $k = 0.5$ a $v = 20$ ms^{-1} es de aproximadamente 30.02°. Para encontrar este número se prueban iterativamente distintos ángulos y se escoge el de mayor rango. Se debe poder realizar para cualquier valor v. Para mostrar esta variación de ángulo donde el rango es máximo se ha normalizado la función (fig. 8a) respecto θ variable. Lo que significa que para una misma k, se computa el valor máximo con diferentes θ y se divide ese valor a toda la función de esa k concreta.

Construyamos ahora una función $\phi(v, k)$ que retorne el ángulo de rango máximo teniendo en cuenta la velocidad inicial v y el orden de fricción k. Según el ejemplo anterior se sabe que $\phi(20, 0.5) \approx 30.02°$. En el programa

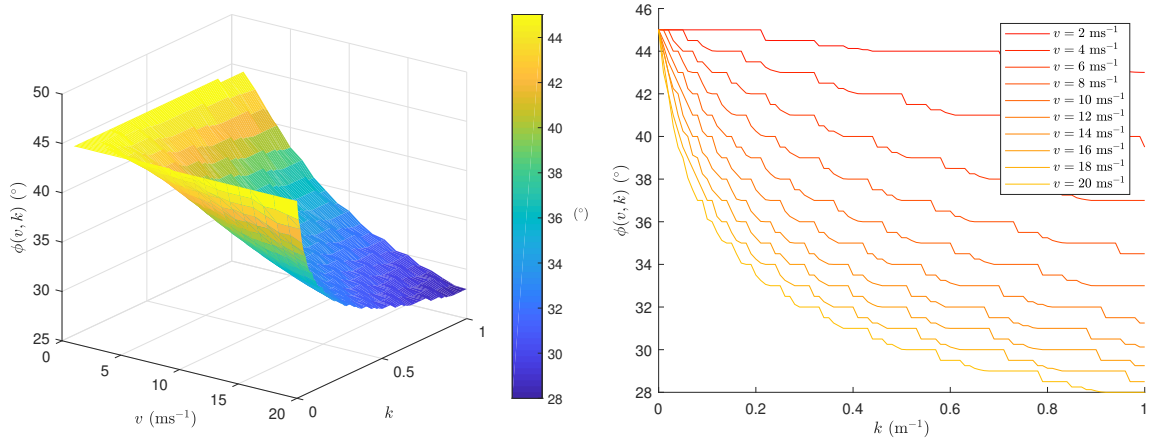
```
double highestAngle(double vel, double k, double g) {

  double d_theta = 1; // Incremento inicial del angulo
  double c_theta = 0; // angulo actual (current theta)
  double c_iter = 0; // iteracion actual (current interation)
  double m_iter = 10; // maximo de iteraciones
  double p_s = -100; // Rango previo (previous s), se inicializa a algo negativo
    relativamente grande para que el siguiente sea mayor siempre

  while (c_iter < m_iter) {

    c_theta+=d_theta;
    double s = range(vel, (c_theta), k, g);
    if (s < p_s) { // Si se sobrepasa, se vuelve atras y el incremento se hace mas
    pequeno
      c_theta-=d_theta;
      d_theta/=2;
      c_iter++;
    } else p_s = s;
  }
  return c_theta;
}
```

Una aproximación de esta función ϕ queda representada en la figura 9.



(a) En 3 dimensiones

(b) En 2 dimensiones con v discreta

Figura 9: $\phi(v, k)$, $1 \leq v \leq 20$ ms^{-1}, $0 \leq k \leq 1$

Tiene sentido que $\phi(v, 0) = 45°$ para todo v. Para $v = 0$, teniendo en cuenta que el desplazamiento sería 0, técnicamente cualquier ángulo sería el de rango máximo, por eso en la figura 9 se representa para $v \geq 1$. La función disminuye cuando k o v aumentan. Para entender la disminución respecto k se puede pensar que cuando hay fricción del aire, la componente v_x de la velocidad de un proyectil

se ve más afectada que la componente v_y debido a que la fuerza de arrastre actúa en contra de la dirección del movimiento en la componente v_x. Esto significa que la componente v_x de la velocidad disminuye más rápidamente que la componente v_y debido a la fuerza de arrastre. Para compensar esta disminución en la componente v_x y lograr el alcance máximo, se necesita asignarle más velocidad inicial en comparación con la componente v_y. Lo que implica un ángulo más pequeño. De la misma manera, con una v inicial más grande, también ocurre lo mismo, ya que la fuerza de arrastre del aire se vuelve más significativa a velocidades más altas. Por lo tanto, para compensar la disminución más rápida de la componente v_x de la velocidad debido a la fricción del aire, se requiere un ángulo de lanzamiento más pequeño.

4. Discusiones

Si bien es cierto que para llegar al resultado final, (la figura 9) no eran necesarias las 4 ecuaciones analíticas, ya que se partía del enunciado del sistema de ecuaciones diferenciales, estas han servido como escalón, rompiendo un problema en piezas más sencillas. También es directamente admirable tener la capacidad de poder usar ecuaciones analíticas, aunque estas sean una mezcla de funciones trigonométricas. No obstante, el modelo persenta demasiadas limitaciones, solo es aplicable para esferas, con área transversal constante independientemente del ángulo. Como posible continuación al estudio, se podría explorar cómo afecta un área transversal variable, como por ejemplo un cubo. También se podría comparar el hecho de tener aletas en un cohete, y como esas redirigen la trayectoria. Aunque para un cohete se debería tener en cuenta el cambio de densidad, viscosidad... Sin embargo, como un primer modelo es suficientemente aceptable.

El hecho de haber necesitado de una aproximación numérica cuya precisión depende directamente del poder computacional del ordenador donde se corre, recuerda al método de los elementos finitos usado en la ingeniería donde tiene una aplicación directa.

5. Conclusiones

Respondiendo directamente a la pregunta *¿es siempre el ángulo de rango máximo 45^o en un tiro parabólico con fricción?* se ha concluido que no siempre el ángulo es 45^{o}. De hecho, nunca es 45^{o} fuera de las simplificaciones. Además se ha observado que este ángulo de rango máximo no solo depende del orden de fricción k sino que también de la propia velocidad a la cual se lanza el proyectil. No obstante, no queda claro de qué manera disminuye, si bien si que tiene que ser menor a $45°$. Por ejemplo, una posibilidad es que para $k \to \infty$ haya una asíntota horizontal en $\theta = 0$. De la misma manera, también parece que haya una asíntota para $v \to \infty$ pero la forma de la asíntota es distinta.

6. Referencias

Bragg, GM, L van Zuiden y CE Hermance (1974). "The free fall of cylinders at intermediate Reynold's numbers". En: *Atmospheric Environment (1967)* 8.7, págs. 755-764.

Maroto, JA, J Duenas-Molina y J de Dios (2005). "Experimental evaluation of the drag coefficient for smooth spheres by free fall experiments in old mines". En: *European journal of physics* 26.3, pág. 323.

Nave, Carl Rod (1998). *Hyperphysics*. URL: `http://hyperphysics.phy-astr.gsu.edu/hbase/Mechanics/quadrag.html`.

Rott, Nikolaus (1990). "Note on the history of the Reynolds number". En: *Annual review of fluid mechanics* 22.1, págs. 1-12.

Shearer, Scott A y Jeremy R Hudson (2008). "Fluid mechanics: stokes' law and viscosity". En: *Measurement Laboratory* 3.

ToolBox, Engineering (2005). *International Standard Atmosphere*. [Online; accessed 4 April 2023]. URL: `https://www.engineeringtoolbox.com/international-standard-atmosphere-d_985.html`.

A. Apéndice

A.1. Código empleado

fric_sim_pde.pde

```
public static final double AIR_DENSITY = 1.293d; // kg m^-3
public static final double WATER_DENSITY = 1000d; // kg m^-3
public static final double EARTH_GRAVITY = -9.80665d; // m s^-2
final int M = 45;
double theta = radians(45);
double vel = 10;
void settings() {
  fullScreen();
  PVector_d init_v = PVector_d.fromAngle(-theta).mult(vel);
  sphere = new Sphere(1e-2f, 0.0058, 0, 0, init_v.x, init_v.y, air);
  sphere_no_res = new Sphere(1e-2f, 0.0058, 0, 0, init_v.x, init_v.y, air);
  sphere_no_res.affectAirResistance(false);
}

void setup() {
  surface.setTitle("Fluid resistance simulator");
  background(51);
  frameRate(30);
  sphere.track(color(74, 127, 237));
  sphere_no_res.track(color(237, 74, 74));
}

Container air = new Container(WATER_DENSITY, EARTH_GRAVITY);
Sphere sphere;
Sphere sphere_no_res;

double dt = 1f/600;
int N = 10;
void draw() {
  stroke(238);
  background(51);
  translate(M, height-M);
  strokeWeight(1);
  textAlign(CENTER, CENTER);
  int A = (int)(xToM(width+M));
  for (int m = 0; m<=A; m+=max(int(A/10f), 1)) {
    float x = (float)mToX(m);
    stroke(238);
    line(x, 0, x, 10);
    text(m, x, 18);
```

```
    stroke(100);
    line(x, 0, x, -height);
  }
  textAlign(RIGHT, BASELINE);
  A = (int)(-xToM(-height+M));
  for (int m = 0; m<=A; m+=max(int(A/10f), 1)) {
    float y = (float)mToY(m);
    stroke(238);
    line(0, y, -10, y);
    text(m, -20, y);
    stroke(100);
    line(0, y, width, y);
  }

  stroke(238);
  line(0, 0, width, 0);
  line(0, 0, 0, -height);
  for (int i = 0; i<N; i++) {
    if (sphere.pos.y<=0) sphere.update(dt/N);
    if (sphere.pos.y<=0)sphere_no_res.update(dt/N);
    //else exit();
  }
  sphere.showTrack();
  sphere_no_res.showTrack();
}

double xToM(double screenX) {
  return (screenX-M) / zoom_scale / pixels_in_m;
}

double yToM(double screenY) {
  return -(screenY-height+M) / zoom_scale / pixels_in_m;
}

double mToX(double m) {
  return m * zoom_scale * pixels_in_m;
}

double mToY(double m) {
  return  -m * zoom_scale * pixels_in_m;
}
```

Body.pde

```
double pixels_in_m = 6240;
double zoom_scale = 5e-1f;
double scale = pixels_in_m * zoom_scale;

class Body {
  double mass;
  double corss_area;
  double dragg_coff;
  PVector_d pos, vel;
  boolean k_def = false;
  double k;

  boolean affect_air_resistance = true;
  boolean log = false;
  Tracker tracker;

  Body(){

  }

  Body(PVector_d pos,PVector_d vel,double k,Container c){
    k_def = true;
    this.pos = pos;
    this.vel = vel;
```

```
25      this.k = k;
26      this.container = c;
27    }
28
29    void affectAirResistance(boolean affect_air_resistance){
30      this.affect_air_resistance = affect_air_resistance;
31    }
32
33    double k(){
34      return k_def? k : 1f/(2*mass) * corss_area * dragg_coff*container.density;
35    }
36
37    Container container;
38
39    void track(color col){
40      tracker = new Tracker(this, col);
41      tracker.track(pos);
42    }
43
44    void showTrack(){
45      tracker.showTrack();
46    }
47
48    void show() {
49    }
50
51    void update(double dt){
52
53      double dv_x = affect_air_resistance ? -vel.x*Math.sqrt(vel.x*vel.x + vel.y * vel.
        y)*k() : 0;
54      double dv_y = affect_air_resistance ? - container.gravity  -vel.y*Math.sqrt(vel.x
        *vel.x + vel.y * vel.y)*k() : - container.gravity;
55
56      vel.x+=dv_x*dt;
57      vel.y+=dv_y*dt;
58
59      pos.x+=vel.x*dt;
60      pos.y+=vel.y*dt;
61
62      if(tracker != null){
63        tracker.track(pos);
64      }
65    }
66  }
67
68  class Sphere extends Body {
69    double r;
70
71    Sphere(double r, double m, double x, double y, double vx, double vy, Container
        container) {
72
73      pos = new PVector_d(x, y);
74      vel = new PVector_d(vx, vy);
75
76      mass = m;
77      dragg_coff = 0.47;
78      corss_area = sphere_corss_area(r);
79
80      this.container = container;
81      this.r = r;
82    }
83
84    void show(){
85      super.show();
86      ellipseMode(RADIUS);
87      fill(100);
88      stroke(238);
89      ellipse((float)(pos.x*scale),(float)(pos.y*scale),(float)(r*scale),(float)(r*
```

```
    scale));
  }

  double sphere_corss_area(double radius) {
    return (double)(Math.PI * radius * radius);
  }
}
```

Container.pde

```
class Container{
  double density;
  double gravity;

  Container(double density, double gravity){
    this.density = density;
    this.gravity = gravity;
  }
}
```

---

propos.pde

```
double radians(double degrees){
  return degrees*Math.PI/180.0d;
}

double range (double vel, double theta, double k, double g) {
  double dt = 1e-4d;//1e-7d;
  Container c = new Container(0, g);
  Body b = new Body(new PVector_d(0, 0), PVector_d.fromAngle(-radians(theta)).mult(
    vel), k, c);
  b.update(dt);
  while (b.pos.y<0)b.update(dt);
  return b.pos.x;
}
//Angle where range is bigger based in initial modulous velocity, friction constant k
    and gravity g
double highestAngle(double vel, double k, double g) {
  double d_theta = 1;
  double c_theta = 0;
  double c_iter = 0;
  double m_iter = 10;
  double p_s = -100;

  while (c_iter < m_iter) {
    c_theta+=d_theta;
    double s = range(vel, (c_theta), k, g);
   // println(s);
    if (s < p_s) {
      c_theta-=d_theta;
      d_theta/=2;
      c_iter++;
    } else {
      p_s = s;
    };
  }
  return (c_theta);
}
```

---

s_tracker.pde

```
class Tracker{
  Body b;
  color col;
  Tracker(Body b, color col){
    this.col = col;
```

```
6       this.b = b;
7       this.b.tracker = this;
8     }
9     ArrayList<Double> x_s = new ArrayList<Double>();
10    ArrayList<Double> y_s = new ArrayList<Double>();
11
12    void track(PVector_d pos){
13      x_s.add(pos.x);
14      y_s.add(pos.y);
15    }
16
17    void showTrack(){
18      noFill();
19      strokeWeight(3);
20      stroke(col);
21
22      beginShape();
23      for(int i = 0; i<x_s.size(); i++){
24        float x = (float)(x_s.get(i) * scale);
25        float y = (float)(y_s.get(i)*scale);
26
27        vertex(x,y);
28      }
29      endShape();
30    }
31
32  }
```

Nota: la clase PVector_d es una copia de la nativa de Processing PVector, pero con precisión double en vez de float.