MODELO MOLECULAR DE SDS PARA REDUCIR EL TIEMPO DE FORMACIÓN MICELAR EN SOLUCIÓN ACUOSA USANDO DINÁMICA MOLECULAR

M.E. Monroy-González[a], A.M. Franco-Riesta[a]*

Departamento de Ciencias, Instituto Tecnológico de Monterrey Campus Santa Fe, Av Carlos Lazo 100, Santa Fe, La Loma, Álvaro Obregón, Ciudad de México, CDMX. Código Postal 01389, MÉXICO. *a01029647@tec.mx

Introducción

El proyecto "Modelación de gol de campo" se enfoca en el desarrollo de un dispositivo diseñado para simular y estudiar el movimiento de proyectiles bajo la influencia de la gravedad, lo que resulta en trayectorias parabólicas. El objetivo principal de este proyecto es proporcionar una herramienta que nos permitiera explorar y comprender los principios detrás de los movimientos parabólicos, además de evaluar la influencia de variables como el ángulo de lanzamiento, la velocidad inicial y la masa del proyectil. El lanzador ofrece la capacidad de controlar y ajustar múltiples parámetros para llevar a cabo experimentos repetibles. Este dispositivo es capaz de lanzar proyectiles con precisión, lo que facilita la recopilación de datos y el análisis de resultados. Adicionalmente, como parte de este proyecto, se lleva a cabo una simulación detallada del tiro parabólico utilizando el programa GeoGebra. El simulador consiste en un conjunto de ecuaciones paramétricas y deslizadores que permiten ajustar las condiciones variables del viento y sus ángulos. De igual manera, se modeló un poste de anotación por medio de vectores fijos para así visualizar el trazado del proyectil.

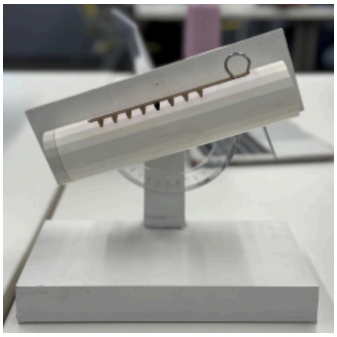
Metodología

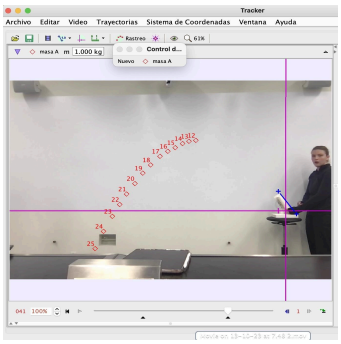
Para ambas partes se requirió de un manejo y desarrollo de ecuaciones cinemáticas, al igual que un entendimiento de la Ley de Hooke (Khan Academy, s.f.) y factores externos como velocidad de viento.

$$y = y_o + \vec{v}_0 sin(\theta)t + \frac{1}{2}\vec{a}t^2 \qquad \int_0^{\vec{\Delta x}} \frac{k\vec{\Delta x}}{m} - \vec{g}(sin(\theta) + \mu_k cos(\theta))d\vec{x} = \int_0^{\vec{v}_f} \vec{v}d\vec{v}$$

$$\Sigma\vec{F}_x = -m\vec{g}sin(\theta) + k\vec{\Delta x} - \mu_k\vec{N} = m\vec{a}_x \qquad v_f = \sqrt{\frac{k}{m}\vec{\Delta x}^2 - 2\vec{g}(sin(\theta) + \mu_k cos(\theta))\vec{\Delta x}}$$

El desarrollo del prototipo físico comenzó con el diseño del lanzador utilizando Tinkercad, una herramienta de modelado 3D. Una vez completado el diseño, procedimos a imprimir el lanzador en 3D para obtener un prototipo físico. Para que el prototipo funcionase se requirió un resorte con una constante elástica (k) lo suficientemente pequeña para permitir la compresión manual, pero lo bastante grande para proporcionar la energía necesaria para un disparo. El resorte era muy pequeño, por lo que se optó por atar dos resortes en serie para obtener la longitud necesaria y así lograr un disparo efectivo. Ya teniendo el arreglo construido, se realizó una grabación clara y estable de una pequeña bola de 0.0027kg siendo disparada por la compresión máxima del resorte, y su trayectoria fue analizada en el software Tracker.



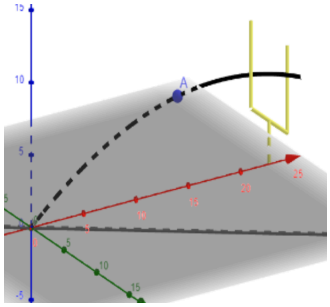


De esta forma, se obtuvo la velocidad de disparo experimental, y con las fórmulas anteriores, se calculó la velocidad teórica para así obtener el margen de error entre ambas. Como complemento, utilizando ecuaciones paramétricas en 3 dimensiones e introduciendo un factor externo, velocidad de viento a un ángulo, se desarrolló un simulador y un programa en Python (que utiliza la librería Sympy (Sympy, s.f.) que indica aproximadamente la configuración del simulador para determinada condición del ambiente al resolver el sistema de ecuaciones 3x3 tomando como parámetros ingresados por el usuario la velocidad del viento, su ángulo, la velocidad de pateo, y la constante de arrastre por viento.

$$Pos = \begin{cases} x(t) = \vec{v_0}\cos(\theta)\sin(\phi)\,t + cte[\vec{u}tsin(\gamma)] \\ y(t) = \vec{v_0}\cos(\theta)\cos(\phi)\,t + cte[\vec{u}tcos(\gamma)] \\ z(t) = y_0 + \vec{v_0}\sin(\theta)\,t - \dfrac{\vec{g}t^2}{2} \end{cases}$$

```
from sympy import symbols, cos, sin, nsolve
from mpmath import radians
# Recibe v0, u, cte, c
# Calcula a, b, t
v0 = float(input("Velocidad inicial de la patada (ms^-1) [0 < v0]? \n\t>>> "))
u = float(input("Velocidad del viento (ms^-1) [u < 0]? \n\t>>> "))
cte = float(input("Constante de arrastre del viento? [0 < 1]? \n\t>>> "))
c = float(input("Ángulo de dirección de viento [0-360]?\n\t>>> "))
a, b, t = symbols('a b t', real = True)
eq_x = - 22.5 + v0 * cos(radians(a)) * sin(radians(b)) * t + cte * u * t * sin(radians(c))
eq_y = - 0 + v0 * cos(radians(a)) * cos(radians(b)) * t + cte * u * t * cos(radians(c))
eq_z = - 5.97 + v0 * sin(radians(a)) * t - 0.5 * 9.81 * t ** 2
solution = nsolve((eq_x, eq_y, eq_z), (a, b, t), (1,1,1))
a_res, b_res, t_res = solution

if a_res > 360:
    a_res = a_res % 360
if b_res > 360:
    b_res = b_res % 360
print("\nEl ángulo de disparo a = %s°. \nEl ángulo de compensación lateral b = %s°." %(a_res, b_res))
print("El tiempo de la trayectoria es de %s segundos." %(t_res))
```



Resultados

Utilizando los métodos mencionados, se obtuvo una constante de elongación con valor de 2.77 N/m, y por ende, una velocidad de disparo máximo teórico de $3.72ms^{-1}$. Esto, comparado con el valor experimental obtenido con Tracker de $4.765ms^{-1}$, difiere en un 27.98%. Así mismo, el programa desarrollado en Python permite, recibiendo valores del entorno como datos de viento, conocer una aproximación numérica a cómo debiera patear el jugador el balón en términos de ángulos de disparo y ajuste, cuyos efectos son visualizables en el simulador Geogebra.

$$\vec{v}_E^{\,2} = \vec{v}_{0x}^{\,2} + \vec{v}_{0y}^{\,2} \qquad \vec{v}_T = \sqrt{\frac{2.77}{0.0027}(0.1231)^2 - 2(9.81)[sin(40°) + 0.07cos(40°)](0.1231)}$$

$$\vec{v}_T = 3.723ms^{-1}$$

$$\vec{v}_E = \sqrt{\vec{v}_{0x}^{\,2} + \vec{v}_{0y}^{\,2}}$$

$$\vec{v}_E = 4.765ms^{-1}$$

$$\varepsilon_{\%} = \frac{\left|\vec{v}_E - \vec{v}_T\right|}{\vec{v}_E} = \frac{|3.723 - 4.765|}{3.723}(100) = 27.98\%$$

Conclusiones

En este trabajo se construyó un arreglo propulsor de proyectiles y se desarrolló un simulador matemático en conjunto con un programa en Python que plantea y modela una solución a la incógnita de cómo

debiera ajustarse un tiro habiendo factores externos ambientales. Posibles mejoras involucran precisión en las mediciones, o un proceso de construcción más refinado. Incluso se podría automatizar con sensores y actuadores un disparador electrónico que en conjunto con el programa pueda ajustarse por sí mismo. En base a este conjunto de herramientas es posible, fundamentado en matemáticas y física, predecir y ajustar diseños y sus comportamientos al entorno externo para así cumplir objetivos.

Referencias

1. Sympy. (s.f.). Solveset Python documentation.
2. Khan Academy. (s. f.). *¿Qué es la Ley de Hooke? (Artículo)*. https://es.khanacademy.org/science/physics/work-and-energy/hookes-law/a/what-is-hookes-law