Universidad de Guanajuato
División de Ciencias e Ingenierías
Mecáanica Analítica
Agosto-Diciembre 2018

Tarea 1

Equipo Cuchara

Rodrigo Aguilar Meneses
María José Fonseca Vázquez
Omar Alejandro Lezama Gallegos
Profesor: Dr. Gustavo Niz Quevedo

14 de septiembre de 2018

Resumen

Se utilizaron métodos numericos en lenguaje Python para generar un código que permita la aproximación de la solución de una ecuación trascendental y, a partir de dicha solución, se calcularon otras variables de interés para el análisis.

1. Introducción

Una ecuación trascendental es aquella cuya solución no es posible encontrar mediante métodos analíticos. Por ejemplo, si se tiene una variable en su forma lineal igualada con una exponencial o un logaritmo de la misma variable, ya que al despejar alguna de las dos, la otra no será útil para realizar un despeje, como suele hacerse habitualmente.

Para poder encontrar soluciones a este tipo de ecuaciones, se puede recurrir a diversos métodos, como por ejemplo el de perturbaciones, expandiendo en series, o también se pueden utilizar métodos numéricos para encontrar las soluciones de las ecuaciones.

Existen diversos métodos numéricos, como la secante, bisección, y Newton-Rhapson, por mencionar algunos.

A continuación, se muestra la resolución de una de ecuación trascendental, apoyándose de métodos numéricos. Se realizó un sólo código para la resolución de todos los incisos, de nombre *sol_num.py*. Los archivos generados para cada inciso se mencionan en su respectiva sección.

2. Resolución del Problema

La ecuación que se debe resolver es la siguiente:

$$T = \frac{kV + g}{gk}(1 - e^{-kT}) \tag{1}$$

Esta ecuación se obtiene evaluando la función de la posición de un proyectil en $y = 0$, de tal manera que se obtiene su alcance horizontal máximo. Resolviendo la ecuación se obtiene el tiempo de vuelo T. Como se observa, al despejar de un lado de la ecuación para T, ya no es posible trabajar con la variable que se encuentra del otro lado.

2.1. Inciso a

Usando algún algoritmo recursivo, crea un código que calcule T para diferentes valores de k, el ángulo y la velocidad inicial.

Para calcular los valores de T para diferentes valores de k, el ángulo, y la velocidad inicial, se consideró como constante todo lo que multiplica al factor de $(1 - e^{kt})$, a esa constante se le llamó G. Se definió una función llamada *constantG* para calcular su valor, para ello se iteran diferentes valores de k, v_0 y θ, y se tiene por separado la otra parte de la función que es $(1 - e^{kt})$. Se tienen 3 listas de los valores que se tomarán para k, v_0 y para θ. Con esto, se calculan todas las combinaciones de estos valores y se resuelve para el tiempo con un triple ciclo anidado en el que se iteran las 3 cifras, se calcula la constante G para cada uno de estos casos, y después que calcula el tiempo de vuelo revisando la intersección entre las dos funciones (la lineal y la constante por la exponencial). Para revisar esta intersección se usa un método numérico implemenado en la función *checkIntersection*

en el que se evalúa la diferencia absoluta entre ambas funciones. Si ésta es menor a la tolerancia (0.025) el punto de intersección es una solución para el tiempo de vuelo. La función que evalúa si se tiene una intersección se itera con un incremento de 0.05 para valores de t, lo que junto con la tolerancia da una buena aproximación. Los datos generados para este inciso se guardan en el archivo *dataA.txt*.

2.2. Inciso b

Con la velocidad inicial de 500m/s y un ángulo inicial de 65 grados, graficar el rango contra k para (k = 0, k = 0.05 y otros 3 valores entre 0 y 1). Compararlo con la aproximación vista en clase basado en teoría de perturbaciones.

Se debe graficar el rango contra k, variando k, se tiene un valor constante de la velocidad inicial y el ángulo inicial. Se tiene una función con la que se calcula el rango en un tiempo t, llamada *horizontal*. Haciendo uso del tiempo de vuelo T calculado en iteraciones, se evalúa *horizontal* en t = T, y el resultado se almacena en la variable *maxRange*. Los resultados se guardan en un archivo de nombre dataB.txt.

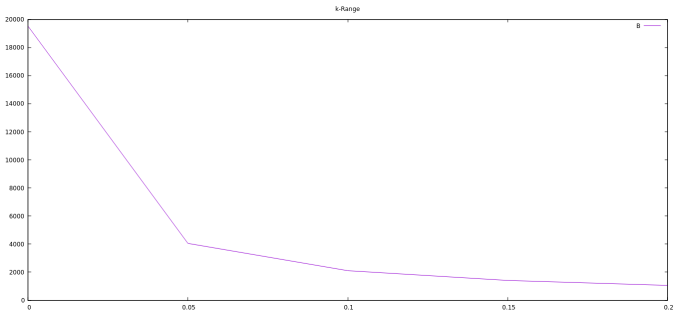

Figura 1: Gráfica del rango como función de k

En la Figura 1 se observa que a medida que la constante k incrementa el rango disminuye. Lo anterior coincide con lo esperado, dado que hay una resistencia del aire, mientras más grande sea ésta menor distancia recorrerá el proyectil: en los siguientes gráficos se aprecia mejor la variación de la trayectoria del proyectil a medida que la resistencia del aire aumenta.

En la sección de conclusiones se compara con la teoría de perturbaciones. Los datos generados para este inciso se guardan en *dataB.txt*.

2.3. Inciso c

Usando los mismos datos iniciales del punto anterior, graficar Distancia Vertical contra Distancia Horizontal para k = 0, y otros 4 valores entre 0 y 1.

Se hicieron dos funciones llamadas *generate_horizontal_data* y *generate_vertical_data*, que con el ángulo y velocidad inicial constantes, y para cada k, itera t de 0 a T y genera un archivo con los valores calculados para $x(t)$ y $y(t)$ usando las funciones *horizontal* y *vertical*. Para $k = 0$ se utilizan las funciones *horizontalk*0 y *verticalk*0.

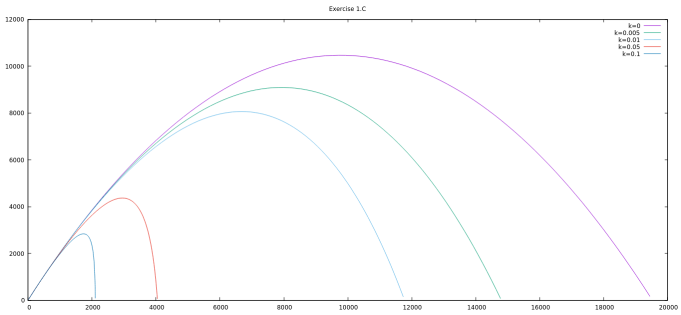

Figura 2: Distancia horizontal contra distancia vertical.

En la Figura 2 se graficaron 5 valores para k. A medida de que k aumenta, se reduce bastante la magnitud de la trayectoria en ambos ejes. El caso ideal, en el que k es igual a cero también se incluye en la gráfica, en él, la forma de la curva corresponde a una parábola que es la predicción mediante el modelo clásico usando los principios de Newton. Los datos generados para este inciso se guardan en el archivo *dataC.txt*.

2.4. Inciso d

Usando los mismos datos iniciales que en los puntos anteriores, graficar Altura contra Tiempo, Velocidad Horizontal contra Tiempo, y Velocidad Vertical contra Tiempo para k = 0, y otros 4 valores entre 0 y 1.

Se piden un total de tres gráficas, por tanto fueron hechas tres funciones, una para cada gráfica. Como se parte de la misma velocidad inicial y ángulo, la función es muy parecida. Las funciones son: *generate_height_time_data*, *generate_horizontal_velocity_time_data* y *generate_vertical_velocity_time_da* Las tres funciones hacen básicamente lo mismo. La que corresponde a altura vs tiempo (generate_height_time_data) calcula la altura vertical iterando t para diferentes valores de k, y lo guarda en el archivo de texto correspondiente. Para las dos funciones restantes se sigue la misma idea, calculándolo para la velocidad en x y y, respectivamente. Cabe mencionar que para los casos especiales $k = 0$ se usa un ciclo con funciones diferentes dentro de las mismas funciones *generate*.

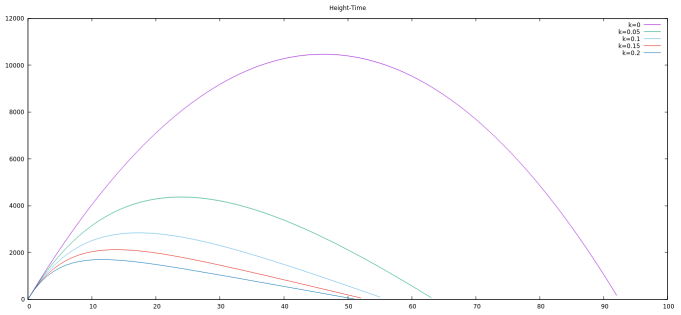

Figura 3: Altura como función del tiempo

En la Figura 3 se observa la altura como función del tiempo. En el caso ideal se tiene una parábola, el proyectil alcanza su punto máximo y cae en forma simétrica a la trayectoria que siguió al subir.

Debido a la resistencia del aire, para valores de k diferentes de cero se tiene que el proyectil describe una curva muy similar a una parábola en los primeros valores del tiempo, pero al llegar al punto máximo tarda más tiempo en caer conforme aumenta la resistencia del aire.

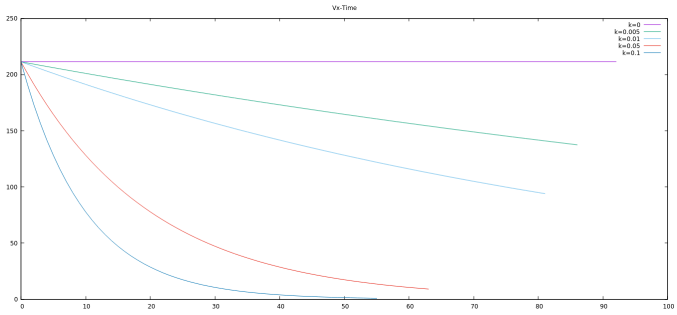

Figura 4: Velocidad horizontal como función del tiempo.

La velocidad horizontal como función del tiempo se aprecia en la Figura 4. Cuando no hay resistencia la velocidad a lo largo de este componente es constante, como es de esperar. Cuando se tienen valores de k diferentes de cero la gráfica tiene una forma similar a una exponencial decayente, es decir, para valores de resistencia cada vez más grandes, la velocidad en el eje x disminuye considerablemente. Cabe mencionar que con valores de k pequeños, el decaimiento tiene la forma de una recta.

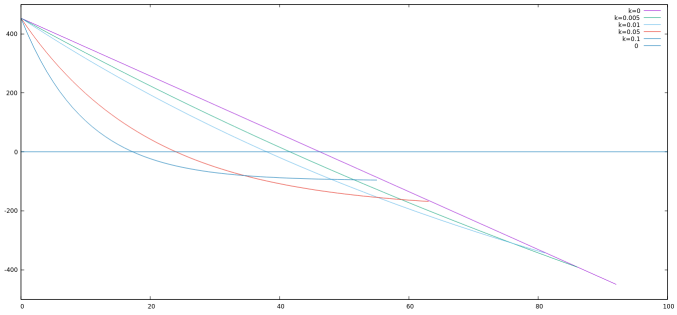

Figura 5: Velocidad vertical como función del tiempo.

El último caso de este inciso se observa en la Figura 5, en ella se muestra el valor $v_y = 0$, pues la velocidad en este componente es positiva hasta que llega a su punto máximo, y negativa cuando pasa del punto máximo al eje de las abscisas. Si no hubiera fricción con el aire, la forma de la velocidad es una línea recta, la velocidad es máxima al inicio de la trayectoria y antes del impacto con el suelo, llega a su punto máximo en la mitad de su rango en x. Pero cuando se tienen valores de k mayores a cero, se nota que el punto máximo (el punto donde la velocidad es cero) se alcanza mucho antes, es decir, la influencia del aire impide que la trayectoria del proyectil sea mayor, tanto en altura como en rango.

Los datos generados se guardan en tres archivos correspondientes para cada gráfica: *data_height_time.txt*, *horizontal_velocity_time.txt*, y *vertical_velocity_time.txt*.

2.5. Inciso e

Buscar el ángulo que da la distancia máxima numéricamente para k = 0, y otros 4 valores entre 0 y 1.

Se escribió una función *generate_angle_distance_data*, en la que se tiene una lista vacía *rangeList* donde se guardarán los rangos máximos que se evaluarán iterando el ángulo de 0 a 90 para distintos valores de k. En la iteración se calcula el tiempo de vuelo para calcular el rango máximo, y cuando se llena el arreglo, se guarda el mejor ángulo en la variable *bestAngle* utilizando $bestAngle = rangeList.index(max(rangeList))$. Esto se repite en instrucciones especiales para el caso $k = 0$ dentro de la misma función *generate*.

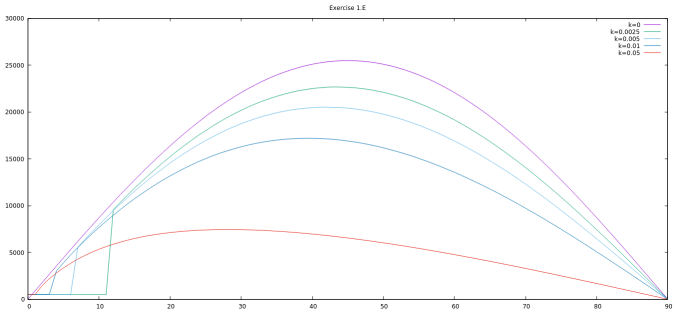

Figura 6: Distancia máxima respecto al ángulo.

El método anterior se ilustra en la Figura 6, para cada valor de k se tiene un ángulo al cual la distancia es máxima. La aproximación no es muy buena para valores pequeños de θ, por lo que se observan algunas líneas rectas al inicio de las curvas con valores más grandes de k. Los ángulos encontrados son los siguientes, para cada valor de k mostrado se tienen ángulos de: 45, 43, 42, 39 y 28 grados, respestivamente; por lo que el ángulo de inclinación para alcanzar un punto máximo en el recorrido del proyectil disminuye conforme la resistencia del aire aumenta. Para $k = 0$ se tiene que el mejor ángulo es de 45 grados, lo que concuerda con lo ya resuelto previamente para el caso sin resistencia del aire. Los datos generados para este inciso se guardan en el archivo *data_angle_distance.txt*

3. Conclusiones

Se comprobaron mediante el uso de métodos numéricos las predicciones que se tienen tanto en el caso ideal como en los modelos más realistas de la mecánica Newtoniana. Se desarrollaron nuevas habilidades para poder visualizar, mediante la programación adecuada, diferentes conceptos físicos y la relación entre ellos.

En general, con el programa realizado se puede hacer una comparación entre la solución numérica y el método de perturbación revisado en clase. Si bien la solución numérica no es una descripción completamente apegada a la reallidad, es una mejor herramienta para aproximar el rango y el tiempo de vuelo. Esto se ve reflejado en la gráfica del inciso b), en la Figura 1 , donde se tiene un decaimiento exponencial del rango en función de la constante k, mientras que en el método perturbativo es un decaimiento lineal que no se ajusta a las observaciones.

Más aún, los autores consideramos que el método numérico es más sencillo de manejar, puesto que se puede tomar un intervalo más grande para valores de k, y no se requiere despreciar tantas cantidades como en el método perturbativo.