Métodos Numéricos para la Ciencia y la Ingeniería

Camila Castillo Pinto

01 de Octubre, 2015.

1 Introducción.

En este problema se consideraba una partícula de masa m que se movía verticalmente sólo en el eje Y rebotando contra un suelo que oscilaba sinusoidalmente con amplitud A y frecuencia w.

El choque contra el suelo es inelástico siguiendo la siguiente regla de choque:

$$v_p'(t^*) = (1+\eta)v_s(t^*) - \eta v_p(t^*) \tag{1}$$

donde t* es el instante del bote, vp y vp' son las velocidades justo antes y justo después del bote, y vs es la velocidad del suelo en ese instante y η es un coeficiente de restitución (entre 0 y 1)

Inicialmente la partícula está en contacto con el suelo y con velocidad hacia arriba mayor que la del suelo.

Los parámetros del problema son A, w, η, m, g y la condicin inicial $y(0), v(0)$, y se tomaron: $m = 1, g = 1, A = 1, y(0) = 0$. Los parámetros faltantes se fijaron durante la implementación del programa.

v(0), y .

2 Parte 1

En la Parte 1 se pedía una rutina que calculara y_{n+1}, v'_{n+1} dados y_n, v'_n, es decir, la posición y velocidad luego del n-ésimo choque.

2.1 Procedimiento

Se fijaron las condiciones iniciales para los parámetros que faltaban: $v(0) = 5, \eta = 0.15 yw = 1.66$ y se definieron límites a y b dentro de los cuales se debían buscar la intersección. El límite $a = 0.1$ se definió así ya que permite que el método para encontrar raíces no se confunda con la posición de partida (partícula pegada al suelo) como un "choque"; y el límite b se encontró mediante la fórmula:

$$b = v_0 + \sqrt{v_0^2 + 2(1+y_0)} \tag{2}$$

que viene de despejar el tiempo de viaje de la partícula hasta que llegue al punto más bajo permitido por la ecuación del suelo ($-A$).

Luego se definieron las funciones de posición y velocidad para el suelo y para la partícula, como también una función que restara ambas posiciones. Además se fijó el valor para el número de choques deseado y luego se comenzó a iterar con la posicion y velocidad inicial dadas.

Se utilizó el comando *brentq* para encontrar los ceros (o raíces) de la función que resta las posiciones, de manera que obtenemos el tiempo en el que sucede el choque entre partícula y suelo. Luego se calculan: la posición de la partícula justo en el choque (nueva posición inicial de la partícula para encontrar el siguiente choque), la velocidad del suelo y de la partícula durante el choque para poder calcular la velocidad de la partícula después del choque con la fórmula (1) (nueva velocidad inicial de la partícula para calcular el siguiente choque); con esto se tendrán los nuevos valores iniciales para calcular el siguiente choque.

Antes de volver a iterar (para pasar al siguiente choque) se deben re-calcular los valores para los límites entre los cuales trabaja *brentq* y se utiliza la misma fórmula con la que se encontraron los primeros límites (formula 2).

Notar que los intervalos re-definidos corresponden a tiempos entre 0 y el valor b, que es como si la partícula saliera de nuevo desde el tiempo 0. Para corregir esto junto con el suelo, se le agregó a la ecuación de éste una fase que llamamos *tcero*. Este *tcero* también debe ser actualizado, agregándole el valor de la intersección encontrado.

Finalmente la iteración guarda en vectores los valores de tiempo de intersección, posición inicial de la partícula luego de cada choque y su velocidad luego de cada choque.

Este proceso se repite tantas veces como choques se desee.

2.2 Resultados

Al correr el programa, éste realiza dos iteraciones y luego cae en un error que corresponde a que la función que resta las posiciones entre suelo y partícula cuando es evaluada en a y en b posee igual signo y el comando *brentq* no puede ejecutarse, es decir, no es capaz de encontrar el cero en este intervalo.

A priori, se cree que el comando *brentq* comienza a dar raíces poco exactas una vez que se empieza a iterar, por lo que llega un punto donde se posiciona a la párticula por debajo del suelo (punto a) y se intenta calcular el cero, pero tanto el punto a como el b están por debajo del suelo y es ahí donde el programa se cae.

Este mismo error ocurre cuando se cambian los párametros del problema.

No se logró arreglar este desperfecto del programa.

3 Parte 2

En la parte 2 se pedía calcular el número de choques hasta que el sistema llegase a un equilibrio.

3.1 Procedimiento

Este procedimiento no pudo ser implementado ya que el programa implementado en la parte 1 no se pudo arreglar, pero se tiene una idea sobre cómo hacerlo:

Simplemente plotear v'_n versus n y estimar para qué valor de nv'_n se ha estabilizado, con n número de choques.

De esta manera se podría ver que el sistema oscila, descendiendo hasta cierto punto donde las oscilaciones se mantienen más o menos estables.

4 Parte 3

No se implementó.

5 Parte 4

No se implementó.

6 Conclusions

De este trabajo se concluye que, si bien el programa de la parte 1 no resultó, los valores de las primeras iteraciones estaban correctos, por lo que era escencial detectar correctamente el problema que se generaba y darle una solución, pero finalmente no se logró concretar.