uniandes

# Laboratorio de Métodos Computacionales

Tarea 11 - C++: ODE

01-2018

uniandes

La solución a este taller debe subirse por SICUA antes de terminada la clase. Los archivos código fuente deben subirse en un único repositorio en Github llamado `NombreApellido_hw11`, por ejemplo yo debería subir mis archivos a un repositorio llamado `JesusPrada_hw11` (10 puntos). En la actividad de sicua deben enviar el link de la tarea en su repositorio.

**Recuerden que es un trabajo individual y debe ser realizado en scripts de python (.py) y C++ (.cpp).**

**NO SE ACEPTAN TRABAJOS TARDE ENVIADOS AL CORREO.**

1. (0 points) **COMENTARIOS Y OUTPUTS**

   **El programa de C++ será interactivo a traves del manejo de inputs y outputs con el usuario. Todo debe quedar claro para un usuario que no programó el script, no sólo por medio de la documentación o los comentarios.**

   **SI EL CÓDIGO NO ESTÁ COMENTADO (INCLUYENDO MENSAJES EN CONSOLA) Y NO CORRE O ARROJA RESULTADOS ERRÓNEOS, NO SE REVISARÁ EL ORIGEN DE SUS ERRORES**

2. (60 points) **Sistema gravitatorio unidimensional**

   Se quiere conocer cómo sería la trayectoria de una partícula de masa $m$ con velocidad y posición inicial $v, x$ respectivamente, bajo la atracción gravitacional de una partícula de masa $M >> m$ ubicada en el origen. Asuma que la diferencia de masas es tan grande que la partícula de masa $M$ NO se mueve.

   Cree un programa de C++ llamado (NA = SUS iniciales de NombreApellido) `NA_gravity.cpp` que evolucione el sistema con Runge-Kutta o Leapfrog. El programa debe inicializar las siguientes constantes:

   - $G = 10$
   - $M = 1000$

   y deberá preguntar el resto de información $(m, x, v)$ al usuario, incluyendo también el $dt$ o $h$ para su método de integración.

   El programa deberá imprimir la posición, velocidad y el tiempo (en ese orden) de la partícula en cualquier formato que puedan leer fácilmente después y deberá detenerse cuando llegue al 1% de la distancia inicial.

   Corra su programa y redirija el output a un archivo llamado `tray.txt`.

**PROHIBIDO USAR CÓDIGO DE OTRAS FUENTES**.

**Ayuda:** Tenga cuidado con el loop while. Puede poner un un seguro tal que si su programa falla en llegar a la condición establecida, el loop termine de todas maneras.

3. (20 points) **Programa de Python que grafica**

   Cree un programa de Python llamado (NA = SUS iniciales de NombreApellido) `NA_graph.py` que lea los resultados del archivo llamado `tray.txt`, con el formato que ustedes le dieron y:

   - Haga una gráfica de $x$ Vs $t$ y la guarde en `pos.png`
   - Haga una gráfica de $v$ Vs $t$ y la guarde en `vel.png`
   - Haga una gráfica de $v$ Vs $x$ y la guarde en `phase.png`
   - Para todas las gráficas le ponga nombre a los ejes, título y leyenda.

4. (20 points) **Makefile**

   Cree un makefile que dados los programas de python y C++ únicamente, los acople de tal manera que al correr el make file, se obtengan las gráficas.

5. (40 points) **BONO TODO O NADA**

   Haga un programa funcional que evolucione y grafique la posición de la partícula en dos dimensiones dada una velocidad tangencial. Deberá incluir un programa de C++ que evolucione el sistema y un programa de Python que grafique, además de lo requerido, $y$ Vs $t$ y $y$ Vs $x$.