INTRODUCCIÓN A LA PROGRAMACIÓN
PRÁCTICO 9

Ejercicios más elaborados.

1 – COMPRESIÓN DE DATOS

A partir de un conjunto dado de caracteres llamado alfabeto, se puede formar una cadena combinando los elementos del mismo. Cuando se tiene que almacenar una cadena, interesa obtener una representación que consuma menos espacio. Una manera de lograrlo es observar si se repiten caracteres consecutivamente; si esto sucede, se puede armar una nueva cadena a partir de la anterior, colocando, cada vez que se repite un carácter, el mismo seguido de la cantidad de veces que se repite.

Por ejemplo, tenemos la cadena ddaadccccbbbcdb, formada a partir del alfabeto {a,b,c,d}, la cual tiene 15 caracteres. Comprimiéndola de la manera indicada anteriormente obtenemos la cadena d1a1dc3b2cdb, la cual tiene sólo 12 caracteres.

Se define la razón de compresión como la razón entre el tamaño original de la cadena y el tamaño comprimido de la misma. En el ejemplo anterior compresión = 15/12 = 1.25. Cuando se trabaja con cadenas aleatorias de distintos tamaños, la razón de compresión promedio para un alfabeto varía. Se pide estudiar la variación de la compresión promedio entre cadenas de distinto tamaños: de largo 5, 10, 20 y 50. Para el cálculo de la compresión promedio se debe generar un número grande de cadenas de un largo dado.

2 – SISTEMA GRAVITACIONAL DE N CUERPOS

Un sistema de N cuerpos con masa, interactúa a través de las Fuerzas gravitatorias. Las posiciones, velocidades, aceleraciones y fuerzas son variables en el tiempo, o sea cambian de un instante a otro en el tiempo. Resolver el movimiento (su posición a lo largo del tiempo) de cada uno de estos cuerpos al interaccionar con los demás, implica resolver la ecuación de Newton para cada cuerpo. Esto en general es muy difícil o imposible de hacer en forma analítica. Una alternativa es utilizar un método numérico. Si en un instante dado de tiempo t0, la posición, velocidad y aceleración de un cuerpo son:

$$\vec{Po} = (\text{xo, yo, zo}) \quad \vec{Vo} = (\text{vxo, vyo, vzo}) \quad \vec{Ao} = (\text{axo, ayo, azo})$$

Y en un instante t1 = t0+dt los vectores son:

$$\vec{P1} = (\text{x1, y1, z1}) \quad \vec{V1} = (\text{vx1, vy1, vz1}) \quad \vec{A1} = (\text{ax1, ay1, az1})$$

Los tres vectores varían en cada instante de tiempo entre t0 y t1 pero si consideramos el movimiento en el intervalo [t0 y t1] como rectilíneo, uniformemente acelerado, con la aceleración dada por A0 (al comienzo), entonces mientras que dt sea pequeño, la posición y velocidad calculadas en t1, serán aproximadas a las reales. Las fórmulas del MRUA nos permiten obtener:

$$\vec{P1} = \vec{Po} + \vec{Vo}\,.\,dt + \frac{\vec{Ao}\,.\,dt^2}{2} \qquad \vec{V1} = \vec{Vo} + \vec{Ao}\,.\,dt$$

Es posible que dadas las condiciones iniciales P0 , V0 , ir calculando las posiciones y velocidades

aproximadas en los tiempos t1 , t2 ... tn. En cada uno de estos tiempos es necesario calcular la aceleración con la ley de Newton de la gravedad sobre el cuerpo i:

$$\vec{A}^i = \sum_{j \neq i} \frac{G.mj.(\vec{P}^j - \vec{P}^i)}{\left|\vec{P}^j - \vec{P}^i\right|^3}$$ donde los supraíndices indican cual es el cuerpo.

La entrada de datos está dada por el sistema de efemerides de JPL horizons. El sistema permite bajar datos orbitales de distintos cuerpos celestes del sistema solar. El sistema permite bajar datos de efemérides de forma interactiva: https://ssd.jpl.nasa.gov/horizons/app.html# o de forma automática vía una api: `https://ssd.jpl.nasa.gov/api/horizons.api`.

Este ejercicio utilizará la api. Su uso consiste en una consulta https con la dirección dada arriba, y los datos requeridos se pasan como argumentos (PARAMETRO=VALOE) del url vía el método GET. Es necesario conocer los parámetros y valores para armar la url. Una vez que se tiene la url se puede utilizar wget para bajar los datos de efemérides. Algunos de los PARAMETROS y VALORES son:

format=text - Se bajan los datos en formato de texto legible.

Command=’399’ - El identificador 399 hace referencia a los datos del planeta tierra

obj_data=’yes’ - Requerido para generar los datos necesarios.

make_ephem=’yes’ - Requerido para generar los datos necesarios.

ephem_type=’vectors’ – Deseamos bajar los datos en coordenadas cartesianas.

vec_table=’2’ - Solo bajar las posiciones X,Y,Z y las velocidades VX,VY,VZ en cada instante de tiempo discreto.

Center=’500@0’ - El centro del sistema de coordenadas es el baricentro del sistema solar, 500 indica que es el centro y el 0 refiere al sistema solar (a efectos prácticos el sol).

start_time=’1986-01-01’ - Tiempo inicial de los datos a bajar.

stop_time=’1986-12-30’ – Tiempo final de los datos a bajar.

step_size=’1%20d’ - intervalo de tiempo entre datos, aquí se especifíca 1 día (url encode)

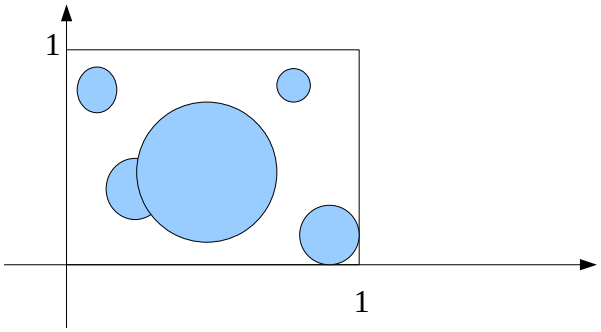
Se pide bajar los datos para la **Tierra** (command=’399’) y el cometa **Halley.** Las urls a utilizar se proveran como parte de la entrega. A modo de ejemplo para la Tierra, la url es:

https://ssd.jpl.nasa.gov/api/horizons.api?format=text&COMMAND='399'&OBJ_DATA='YES'&MAKE_EPHEM='YES'&EPHEM_TYPE='VECTORS'&VEC_TABLE='2'&CENTER='500@0'&START_TIME='1986-01-01'&STOP_TIME='1986-12-30'&STEP_SIZE='1%20d'

El objetivo del ejercicio es correr el simulador con las condiciones iniciales dadas por la primera entrada de los datos de los cuerpos bajados y generar los datos durante el período de tiempo que se corresponde con los datos (start_time, stop_time). El simulador debe generar los datos X Y Z para cada instante de timepo dado por el paso STEP_SIZE (1 día). Luego se debe comparar los datos de la simulación junto a los bajados del sistema horizons, graficando ambos en **gnuplot**. Se debe generar una imagen de salida donde se puedan comparar las orbitas simuladsa con los datos reales bajados.

3) INTEGRACIÓN MONTECARLO

El método Montecarlo sirve para calcular el área de una figura o bajo una curva de una función (integral). Es un método probabilístico que consiste en generar un conjunto de puntos aleatorios uniformemente distribuidos en una región y registrar cuantos de ellos caen dentro de la zona de interés. Entonces el área de la zona de interés será aproximada a la razón entre los puntos que cayeron dentro y los que no.



Si generamos puntos aleatorios (x,y) uniformemente distribuidos dentro del cuadrado unidad y generamos también un conjuntos de círculos aleatorios (radios y centros aleatorios) cuyos centros estén dentro del cuadrado, entonces el área aproximada de la unión de lo círculos es:

$$\text{Area} \sim \text{Nint} / \text{Ntot}$$

Nint es la cantidad de puntos que caen dentro de los círculos y Ntot la cantidad de puntos total generados.

Escribir un programa que calcule el área para el ejemplo dado. Se debe generar una cantidad determinada de círculos aleatorios con centro dentro del cuadrado unidad y los radios aleatorios deben estar en el rango: $0{,}1 < r < 0{,}25$. Notar que parte de algún círculo puede quedar fuera del cuadrado. El programa se debe detener cuando la diferencia entre las áreas calculadas en el paso N y N+1 (en valor absoluto) no supere un error dado.

Considerar como el mismo problema pero generando en el cuadrado unidad un conjunto de cuadriláteros de tamaño aleatorio, con sus lados en el rango: $0{,}1 < \text{lado} < 0{,}5$.

4) CAMINO ALEATORIO

El camino aleatorio (“random walk” en inglés) es un modelo matemático que muestra como se comportan o distribuyen ciertos valores o parámetros de un conjunto de variables aleatorias. Si consideramos tirar una moneda (solo puede salir cara o cruz), el resultado de una de las caras es una variable aleatoria. Si tiramos una secuencia de k monedas y vemos el resultado, obtendremos algo parecido a lo siguiente (cara o cruz lo tomo como 1 o 0):

0 1 0 0 0 1 1 1 0 1 0 1 ....

Si realizamos muchas pruebas de estas secuencias, algunas tendrán más unos y otras más ceros. Para saber como se distribuye la cantidad de unos (o ceros) se considera la suma total de cada secuencia. Queremos estudiar como se distribuye esta suma para un número grande de secuencias de un largo dado y a su vez como varían estas distribuciones cuando cambia el largo de la secuencia.

Se puede crear un programa que genere N pruebas de cadenas de K números aleatorios y guarde en un archivo los porcentajes de las distintas sumas de los K valores que se dan, por ejemplo un archivo para K=5 y N=10 contiene lo siguiente:

# suma	porcentaje
0	10
1	15.3
2	30.7
3	25
4	12
5	7

El programa debe contemplar la posibilidad de variar los valores de K y N, esto es desde donde comienza K y el espaciamiento y que se genere los archivos correspondientes de forma que sea posible evaluarlos e identificar los parámetros (K y N) que los generaron para usarlo posteriormente de forma automática con otro programa, script, etc.

El programador tiene la libertad de implementar lo pedido de la forma que considere mas conveniente (elección de las variables aleatorias binarias generadas, identificar los parámetros de los archivos por el nombre del archivo o un encabezado inicial dentro del mismo, etc.).

5) Para calcular la trayectoria de una partícula sometida a fuerzas arbitrarias podemos usar una aproximación discreta de la tercera ley de Newton.Las posiciones,velocidades, aceleraciones y fuerzas son variables en el tiempo, o sea cambian de un instante a otro en el tiempo. Resolver el movimiento (su posición a lo largo del tiempo) de cada uno de estos cuerpos al interaccionar con los demás, implica resolver la ecuación de Newton para cada cuerpo. Esto en general es muy difícil o imposible de hacer en forma analítica. Una alternativa es utilizar un método numérico. Si en un instante dado de tiempo t0, la posición, velocidad y aceleración de un cuerpo son:

$\vec{Po}$ = (xo, yo, zo) $\vec{Vo}$ = (vxo, vyo, vzo) $\vec{Ao}$ = (axo, ayo, azo)

Y en un instante t1 = t0+dt los vectores son:

$\vec{P1}$ = (x1, y1, z1) $\vec{V1}$ = (vx1, vy1, vz1) $\vec{A1}$ = (ax1, ay1, az1)

Los tres vectores varían en cada instante de tiempo entre t0 y t1 pero si consideramos el movimiento en el intervalo [t0 y t1] como rectilíneo, uniformemente acelerado, con la aceleración dada por A0 (al comienzo), entonces mientras que dt sea pequeño, la posición y velocidad calculadas en t1, serán aproximadas a las reales. Las fórmulas del MRUA nos permiten obtener:

$$\vec{P1} = \vec{Po} + \vec{Vo}\,.\,dt + \frac{\vec{Ao}\,.\,dt^2}{2} \qquad \vec{V1} = \vec{Vo} + \vec{Ao}\,.\,dt$$

Es posible que dadas las condiciones iniciales P0 , V0 , ir calculando las posiciones y velocidades aproximadas en los tiempos t1 , t2 ... tn. En cada uno de estos tiempos es necesario calcular la aceleración con la ley de Coulomb sobre la partícula de carga q1,generada por otra fija de carga q2:

$$\vec{F}^{i} = \frac{K \,.\, q1 \,.\, q2 \,.\, (\vec{P}^{j} - \vec{P}^{i})}{\left|\vec{P}^{j} - \vec{P}^{i}\right|^{3}}$$ con K=8.99*10^{9} $Nm^{2}C^{-2}$

En base a esto se pide estudiar la “dispersión de Rutherford” cuando se dispara una partícula cargada de carga q1 con P1o=(-xo,b), Vo=(vo,0) y para q2 con P2o=(0,0) fija. Para esto se disparan varias partículas con b (parámetro de impacto) variando entre 0 y bmax. Se pide registrar los ángulos de las partículas dispersadas pasando la carga fija (x>0) y a una distancia R de la partícula fija. Considerar los siguientes parámetros:

q1,q2=1.6*$10^{-}19$
vo=10^{7}
xo=10^-4
bmax en [0,10^-12]
R=10^-12
dt=10^-13

Todo en el sistema mks.