Métodos Computacionales

Tarea 1 — 2018–20

Los archivos:
`ApellidoNombre_temp.sh`,
`ApellidoNombre_temp.py`,
`ApellidoNombre_rectangulos.py` y
`ApellidoNombre_integrales.py`
deben estar comprimidos en `ApellidoNombre_hw1.zip` y deben descomprimirse en un directorio `ApellidoNombre_hw1`. Este directorio comprimido debe contener únicamente los archivos mencionados anteriormente y debe subirlo a SICUA antes de las 8:00 pm del lunes 10 de septiembre de 2018 (10 puntos). Recuerde que es un trabajo completamente individual (no debe usar códigos que usted no haya escrito en su totalidad ni debe recibir ayuda de nadie cuando esté escribiendo su código). Recuerde también que los códigos deben correr sin botar errores en los computadores de compufísica (Y110B). Si su código bota errores que usted no sabe cómo corregir, comente la sección de código problematica y haga un print con un mensaje que indique que hay código que usted considera que merece ser revisado en detalle durante la correción. Recuerde que es buena práctica comentar el código y elegir buenos nombres para sus variables. Eso será tenido en cuenta durante el proceso de corrección de esta tarea. No se aceptarán trabajos entregados tarde ni por otro medio distinto a SICUA.

1. (30 points) **Temperaturas globales**
   El archivo `tempdata.txt` contiene datos de temperaturas promedio mensuales y anuales de una estación meteorológica en Antártida. Por otro lado, en la página: `https://climate.nasa.gov/system/internal_resources/details/original/647_Global_Temperature_Data_File.txt` hay datos de anomalías de las temparaturas anuales globales (diferencias con un promedio de las temperaturas anuales entre 1951 y 1980). En este ejercicio ustedes deben usar un script de bash llamado `ApellidoNombre_temp.sh` y uno de python llamado `ApellidoNombre_temp.py` para analizar dichos datos.

   El script de bash debe (15pts):
   a) Crear un directorio llamado `Temperaturas`
   b) Entrar a dicho directorio
   c) Descargar los datos de `https://climate.nasa.gov/system/internal_resources/details/original/647_Global_Temperature_Data_File.txt` y copiar los datos de `tempdata.txt` en el directorio `Temperaturas`. Puede asumir que el archivo está en el directorio en el que se corre el script de bash.
   d) Para los datos de Antártida, guardar en un archivo llamado `menorque30.txt`, los años en los que en al menos uno de los meses, la temperatura fue menor a -30 grados Celsius.
   e) Para los datos de Antártida, guardar en un archivo llamado `menorque6.txt`, los años en los que las temperaturas medias de todos los meses fueron menores a -6 grados Celsius. Ojo, no incluya años en los que falten datos.
   f) Guardar en un archivo llamado `junio.txt` todas las temperaturas medias del mes junio.
   g) Imprimir en la terminal el número total de años en los que la temperatura media fue mayor a -15 grados Celsius.
   h) Borrar el archivo `tempdata.txt`.
   i) Correr el script de python.

El script de python debe (15pts):
a) Leer los datos del archivo `647_Global_Temperature_Data_File.txt` y almacenarlos en uno o varios arreglos de numpy. Estos datos tienen tres columnas, la primera corresponde a los años, la segunda a la anomalía de temperatura para dicho año y la tercera son datos de las anomalías suavizados.
b) Hacer una gráfica de anomalías de temperatura en función del año y guardar dicha gráfica sin mostrarla en `Grafica_temp.pdf`.
c) Encontrar los años para los cuales las anomalías sean mayores a 0.5 grados Celsius.
d) Encontrar los 20 años más calientes e imprimirlos en orden ascendente, de más frío a más cálido, en la terminal.
e) Hacer una gráfica (use una curva continua) de anomalías de temperatura suavizadas en función del año. A esa grafica además debe incluirle los puntos correspondientes a los 20 años más calientes. Use herramientas de matplotlib para gráficamente indicar la diferencia con respecto al año anterior. Guardar dicha gráfica sin mostrarla en `Grafica_temp2.pdf`.

2. (30 points) **Geometría (20pts)**
Escriba un programa en Python llamado `ApellidoNombre_rectangulos.py` que, dados dos rectángulos centrados en puntos con coordenadas (x_1, y_1) y (x_2, y_2), cuyos lados ($a1 = b1$ y $c1 = d1$) y ($a2 = b2$ y $c2 = d2$) están orientados paralelos a x o a y, imprima uno de los siguientes mensajes:
`los rectangulos no se intersectan`
`los rectangulos se tocan en un solo punto de coordenadas x0, y0`
`los rectangulos se tocan en un segmento de recta horizontal/vertical x=x0/y=y0"`
`los rectangulos se intersectan formando un rectangulo de lados`XXX".
Donde x0, y0 y z0 o XXX son valores encontrados por su código. Su código debe funcionar para todas las configuraciones posibles.

3. (30 points) **Integrales**
Escriba un programa en Python llamado `ApellidoNombre_integrales.py` que tenga sus implementaciones propias de los siguientes métodos de integración:
- Trapezoide
- Simpson
- Monte Carlo
y Valores medios
Cada método debe estar definido como una función dentro del script.
a) En la primera parte debe calcular la integral de la función cos(x) en el intervalo $(-\pi/2, \pi)$ para un 10001 puntos. Imprima un mensaje para cada método así:
`metodo:, valor de la integral, error`.
donde el error los debe calcular como $\mid (I_N - I_A) \mid /I_A$ donde I_N es la integral calculada numéricamente e I_A es la integral analítica.
b) En la segunda parte deben calcular cómo varía el error en función del número de puntos N para los distintos métodos. Para esto construyan un arreglo de 6 valores N entre 101 y $(1+10^7)$ (usen numpy.logspace y piensen por qué toman números impares) y hagan una gráfica del error en función del número de puntos. Use una escala logarítmica loglog y guárde la gráfica sin mostrarla en `ApellidoNombre_int_error.pdf`.

c)1- En la tercera parte deben integrar la siguiente función:

$$\int_0^1 \frac{dx}{\sqrt{\sin(x)}} \tag{1}$$

Para ello, pruebe los cuatro métodos de integración desarrollados en las secciones anteriores. Esta prueba es para que ustedes vean qué métodos no funcionan y debe estar comentada en el código final que envién con su tarea.

2- Es altamente probable que el resultado sea distinto para los tres métodos, por su manejo de la singularidad en 0, o que tengan errores por la división por cero. El propósito de esta parte del taller es probar distintas maneras de resolver el problema de la singularidad. El primer truco es tratar de quitar el punto problemático. En vez de que sea infinito, obligue al programa para que sea un valor grande no infinito, por ejemplo 10^6. ¿Qué tan grande es el error ahora? Imprima:

El nuevo valor de la integral usando el método de Simpson cambiando infinito por 10^6 *el valor de la integral es XXXX.*

3- Otro truco es no evaluar la función en 0 sino en un valor un poco mayor a cero, por ejemplo 10^{-6}. ¿Qué tan grande es el error ahora? Imprima:

El nuevo valor de la integral usando el método de Simpson evaluando la función en 10^-6 *y no en cero el valor de la integral es XXXX*

4- Finalmente un truco que funciona mucho mejor que los anteriores es restar la singularidad. Para valores pequeños $sin(x) \approx x$, con lo que podemos hacer el siguiente truco:

$$\int_0^1 \left(\frac{1}{\sqrt{\sin(x)}} - \frac{1}{\sqrt{x}} \right) dx + \int_0^1 \frac{dx}{\sqrt{x}} \tag{2}$$

El segundo valor lo podemos calcular de manera analítica, y el primer valor numéricamente, aprovechando que para valores pequeños de x las singularidades se cancelan. Vuelva a hacer la integral con esta nueva función **(calculando la otra integral analíticamente)**. Imprima el valor de la integral "restando la singularidad el resultado es XXXX".

Un valor de referencia para la integral es 2.03480532.

5- Imprima cuál de los métodos se acerca más a este valor.