Métodos Numéricos para la Ciencia e Ingeniería

Pablo Aníbal Peña Rojas

September 25, 2015

1 Introducción

El trabajo presentado consiste en utilizar métodos númericos propios para calcular la Luminosidad total del sol y la radiación de él tanto con algoritmos propios como con los de las librerías de python y así poder determinar la precisión y efectividad de cada uno (tiempo que tarda el computador en computar los algoritmos). Con estos datos se calculará el radio solar. Todo esto utilizando un entorno de trabajo profesional para proyectos en los cuales participan múltiples programadores (Github) [1].

2 Procedimiento

2.1 Parte 1 - Constante Solar

Es dado el archivo "sunAMO.dat" [2], el cual contiene el espectro del Sol, medido justo afuera de nuestra atmósfera en $\frac{[W]}{m^2 nm}$/ en relación a la longitud de onda. Se lee el archivo con python y se plotea un gráfico que relacione ambas en unidades astronómicas con el fin de obtener la constante solar.

2.2 Parte 2 - Flujo del Sol

Se utiliza un algoritmo propio para integrar el flujo según la longitud de onda y así obtener la luminosidad en todo el espectro de luz al momento de llegar a la tierra. Para la luminosidad total del sol, que corresponde a la superficie de la esfera (de la cual nos llega tan sólo un punto) multiplicada por el flujo incidente (integral calculada con el algoritmo propio) se utilizó el siguiente cálculo $L_{total} = 4\pi d^2_{tierra-sol} * flujo$, donde $d_{tierra-sol} = 1UA$.

2.3 Parte 3

Se intenta de medir la radiación de cuerpo negro (asumiendo la temperatura efectiva del sol $T_eff = 5778[K]$) con la ecuación:

$$B_\lambda(T) = \frac{2\pi hc^2/\lambda^5}{e^{hc/\lambda k_B T} - 1}$$

donde h es la constante de Planck, c es la velocidad de la luz en el vacío, k_B es la constante de Boltzmann, T es la temperatura del cuerpo negro y λ es la longitud de onda. Para ello se integró numéricamente la función de Planck para

estimar la radiación de cuerpo negro. Para ello se utiliza la integral equivalente dada

$$P = \frac{2\pi h}{c^2}\left(\frac{k_B T}{h}\right)^4 \int_0^\infty \frac{x^3}{e^x - 1}$$

, que sin embargo, tiene puntos en los que diverge ($x = 0$), para ello se utilizó el cambio de variable $y = arctan(x)$ quedando una constante multiplicada por

$$I = \int_0^{\frac{\pi}{2}} \frac{\tan(x)^3}{\cos(x)^2 * \exp(\tan(x)) - 1)}$$

Finalmente, con esos datos (radiación de cuerpo negro total(bbr) y flujo) se calcula el radio del sol con la siguiente ecuación:

$$\sqrt{AU^2 \frac{L_{puntual}}{bbr}}$$

2.4 Parte 4

Se calculan las mismas integrales con el paquete de numpy, las funciones .trapz y .quad, que corresponden al método de los trapecios y a el método de Planck, respectivamente. Finalmente, se comparan los valores obtenidos con numpy y con los métodos propios, tanto en resultados como en tiempo de ejecución. El tiempo de ejecución se midió utilizando la función %timeit de ipython.

3 Resultados

3.1 Parte 1

El gráfico es el siguiente:

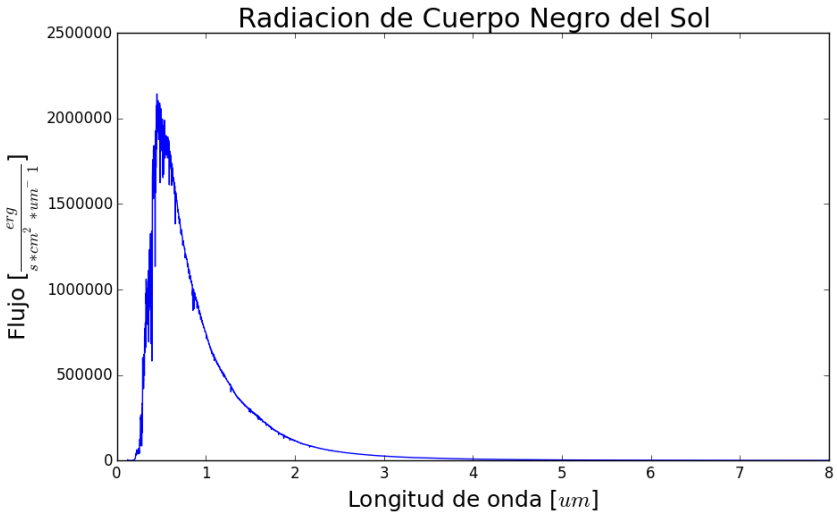

Figure 1: Longitud de Onda[μn] vs Flujo $\frac{erg}{scm^2 \mu n}$

3.2 Parte 2

Para la integral se utilizó el método de los trapecios y el código fue el siguiente [4]:

```
for i in range(len(wavelength)-1):
    trapecitos= (wavelength[i+1]-wavelength[i])*(flujo[i]+flujo[i+1])/2.0
    Integral_1+=trapecitos
```

Y el resultado de L_{total} fue de $3.8418486670 * 10^{33} \frac{erg}{s}$, Comparando el resultado con la librería de python astropy, que trae incluida la luminosidad del sol (constants.au), hubo una diferencia de $4 * 10^{30} \frac{erg}{s}$, que si bien es de 3 órdenes de magnitud menor, aún así no es despreciable. Se presume que la diferencia se debe a la inexactitud de la integral utilizada.

3.3 Parte 3

El resultado de la integral de la radiación de cuerpo negro hubo que multiplicarla por las constantes y quedó en cgs como $63200674870.1 \frac{erg}{s}$. El código utilizado para calcularla fue:

```
for i in range(len(planx)-1):
    x2=planx[i+1]
    x1=planx[i]
    y2=blackbody_function[i+1]
    y1=blackbody_function[i]
    kkk=((x2-x1)/2)*(y2+y1)
    Integral_2+=kkk
```

Y utilizando la fórmula $\sqrt{AU^2 \frac{L_{puntual}}{bbr}}$ dió que el radio del sol era de 69551153468.8 $[cm]$. Contrastando con el valor del radio solar de la libreria constants de astropy, tenemos que la diferencia es de 353468 [cm], un poco más de 3 [km], lo que es un error bastante satisfactorio.

3.4 Parte 4

Los resultados de las integrales, con todos los métodos se muestran en la siguiente tabla [3]:

Table 1: Comparación de Valores

Tabla de valores	Método Propio	integrate.trapz	integrate.quad
Integral del Espectro de Luz Solar $\frac{erg}{(scm^2)}$	1366090.79684	1366090.79684	
Integral de la Radiación de Cuerpo Negro $\frac{erg}{s}$	6.49393890472	6.49393890472	6.49393940227

Y para los tiempos se hizo también una tabla en nanosegundos, el método utilizado fue %timeit, por lo que python ejecutó varias veces las integrales y promedió los tiempos de demora para obtener mayor exactitud. Los tiempos son por iteración y eran 10^7 iteraciones, por lo que si se quiere obtener los datos en segundos, basta multiplicar por 100:

Table 2: Comparación de tiempos

Tabla de Tiempos [ns * 10^7 loops]	Método Propio	integrate.trapz	integrate.quad
Integral del Espectro de Luz Solar $\frac{erg}{(scm^2)}$	30.4	32.4	
Integral de la Radiación de Cuerpo Negro $\frac{erg}{s}$	34.8	30.3	33.2

4 Conclusiones

Los resultados abordados fueron bastante similares, el problema de la Luminosidad del Sol (muy distinto al de astropy) no se debió a la inexactitud de la integral, puesto que el valor es el mismo que con el método integrate.trapz (véase tabla 1). El radio del sol varió en unos despreciables 3 [km]. Por otro lado, los tiempos no fueron tan disímiles, para la primera integral, el método de python fue 20 milisegundos más lento (no es mucho). Para la segunda integral los tiempos fueron más variados, pero nada muy acentuado: Método propio fue de 0.348[s], método trapz fue de 0.303[s] y el quad de 0.332[s], unos 35 milisegundos más rápido. Para que el tiempo de ejecución se note, deben ser programas con miles de millones de datos. Si es vez de hacer 10^7 iteraciones, hubiesemos hecho 10^10, el programa hubiera tardado 3 segundos más en calcular.

References

[1] https://github.com

[2] http://www.pveducation.org/pvcdrom/appendices/standard-solar-spectra

[3] Tablas hechas en http://www.tablesgenerator.com/

[4] Códigos hechos con http://pygments.org/demo/2746442/