Métodos Numéricos para la Ciencia e Ingeniería:
Informe Tarea1

Martín Panza

25 de Agosto, 2015

1. Pregunta 1

1.1 Introducción

Mediante un archivo 'sun_AM0.dat' con datos del espectro del Sol: flujo y longitud de onda cuyas unidades son $W \cdot m^{-2} \cdot nm^{-1}$ y nm respectivamente; se pide realizar un gráfico con ambas series de datos. Se requiere utilizar cgs para las unidades de flujo y Angstrom o micron para la longitud de onda.

1.2 Procedimiento

Para abordar este problema se hizo uso de la instrucción np.genfromtxt() para obtener los datos desde el archivo disponible. A continuación se crearon dos arreglos y se les cambiaron las unidades de $W \cdot m^{-2} \cdot nm^{-1}$ a $g \cdot s^{-3} \cdot cm^{-1}$ respectivamente para luego graficar utilizando plt.plot desde el módulo matplotlib.

1.3 Resultados

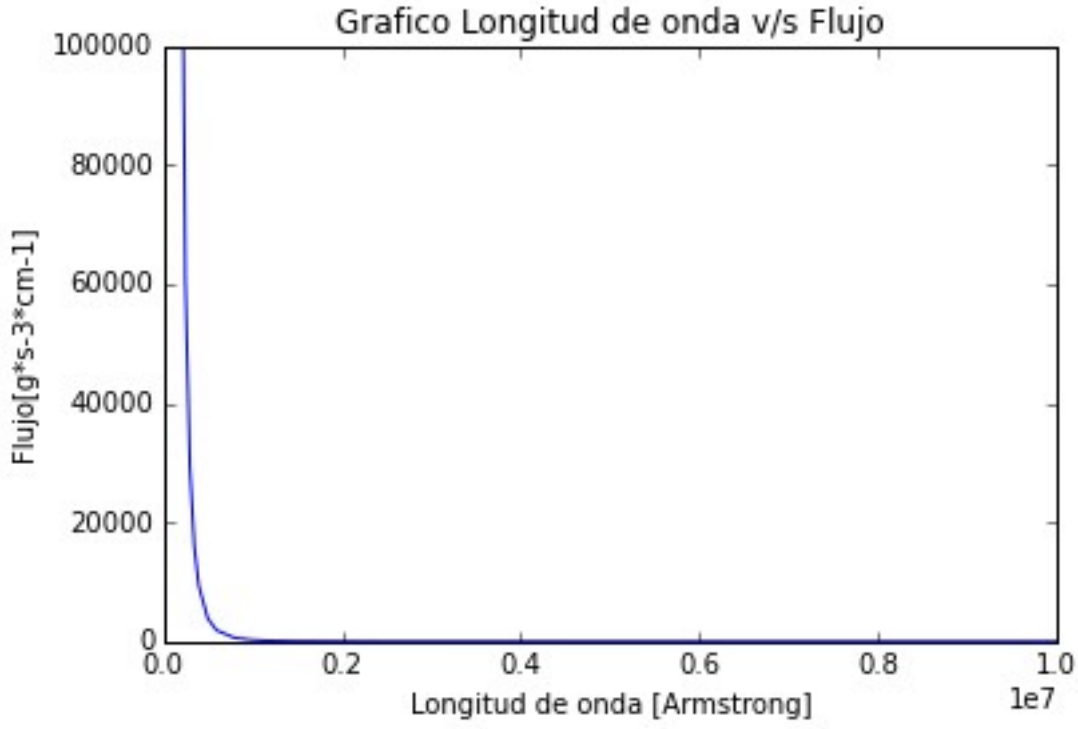

Figura 1: Espectro del sol

1.4 Conclusiones

Se puede observar en la Figura 1 que el flujo demuestra un comportamiento asintótico decreciente al aumentar la longitud de onda.

2. Pregunta 2

2.1 Introducción

Se pidió integrar el espectro en longitud de onda para calcular la luminosidad total del sol (energía por unidad de tiempo). Por lo tanto, se debía escoger un método apropiado. El algoritmo debía ser escrito por el alumno.

2.2 Procedimiento

Para integrar el espectro en longitud de onda escogí el método del trapecio debido a que era más fácil de implementar a dos arreglos que era lo que se tenía, además de que los puntos en el eje x no eran equidistantes. La división que utilicé para el método fue de 1696 que es la máxima posible para tener una mayor precisión. Elegí transformar las unidades de vuelta a $W \cdot m^{-2} \cdot nm^{-1}$ para el flujo y nm para la longitud de onda.
Por último, la luminosidad del sol corresponde a la fórmula $L = P \cdot S$ con P la potencia que fue la integral calculada y S la superficie de una esfera de radio $1{,}496 \cdot 10^{11} [m^2]$

2.3 Resultados

La potencia calculada fue 1366.09079684 $W \cdot m^{-2}$
Por lo que para la luminosidad se obtuvo 3.84195803333e+26 W

3. Pregunta 3

3.1 Introducción

Fue dada la función Planck que determina la radiación de un cuerpo negro en unidades de energía por unidad de tiempo por unidad de área por unidad de longitud de onda.

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

Donde h, c y kB son las constantes de Planck, la velocidad de la luz en el vacío y de Boltzmann respectivamente. T es la temperatura del cuerpo negro, que en el caso del sol es de 5778 K; y λ es la longitud de onda.
Se pedía escribir un algoritmo para integrar numéricamente la función de Planck para estimar la energía total por unidad de tiempo emitida por un cuerpo negro con la temperatura efectiva del sol, para luego compararla con la obtenida en la pregunta 2 y estimar el radio efectivo del sol. El algoritmo debía permitir el ir refinando el valor de la integral para una tolerancia del error asociado. Fue dado también el resultado analítico de la integral con el objetivo de comparar los resultados.

3.2 Procedimiento

Se indicó que la integral de la función de Planck correspondía a la siguiente fórmula:

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

Sin embargo, para poder realizar esa integral impropia se debió realizar el cambio de variable: y=arctan(x), resultando la integral impropia de la siguiente manera:

$$P=\frac{2\pi h}{c^2}\cdot\left(\frac{kBT}{h}\right)^4\int_0^{\pi/2}\frac{\tan^3(y)\,sec^2(y)}{e^{\tan(y)}-1}$$

El método utilizado para la integración numérica fue el método de Simpson compuesto. Sin embargo, este método no permite evaluar la función en los límites de la integral, por lo que en aquellos puntos utilicé el método del punto medio.
Luego, se implementó una instrucción para obtener el error de la integración numérica respecto al resultado analítico y permitir un cambio a la partición hecha sobre el eje x para encontrar el mejor resultado.
Para encontrar el radio efectivo del sol se utilizó la siguiente fórmula:

$$P=\sigma\cdot T_{eff}{}^4\cdot\left(\frac{R_s}{a_0}\right)$$

Con σ la constante de Stefan-Boltzmann, T_{eff} la temperatura efectiva del sol 5778 K, R_s radio del sol y a_0 radio de una esfera $1{,}496\cdot10^{11}[m^2]$

3.3 Resultados

Resultado analítico de la integral de la función de Planck: 63200679.7124

n (partición)	Error
3	68.6781848038%
6	6.96948069324%
12	6.78281849002%
24	0.563402791478%
48	0.0269411132858%
96	0.000113156017647%
192	5.55307588002e-06%
384	6.13782759729e-07%

Tabla 1: comparación entre integración numérica y resultado analítico de la función de Planck para distintas particiones.

Como se observa en la Tabla1, 384 posee el menor error.
El valor para esta partición fue 63200680.1003

Al compararla con lo obtenido en la pregunta 2: 1366.09079684
Se observa que la integral en la pregunta 3 es 46263.8415714 veces mayor.

El resultado obtenido para el radio efectivo del sol fue: 1.49599910381e+11 m

3.4 Conclusiones

El error obtenido para la partición adecuada de la integración numérica fue muy bajo, que permite confianza en el método. Sin embargo, para ciertas particiones los errores son muy altos. Es vital buscar el n indicado. Claramente para n más grande el método es más preciso.

4. Pregunta 4

4.1 Introducción

Para esta parte, se pidió calcular las integrales de las preguntas anteriores 2 y 3, esta vez con métodos incluidos en el modulo scipy con el fin de compararlos. También se pidió la comparación entre la velocidad de ejecución de los algoritmos.

4.2 Procedimiento

Para calcular las integrales se utilizaron los siguientes métodos: scipy.integrate.trapz para los arreglos de la pregunta 2, y scipy.integrate.quad para la función de Planck en la pregunta 3. Sin embargo, este último no podía calcular la integral en los puntos límites por lo que se le ingresaron valores aproximados.
Para calcular la velocidad de ejecución se utilizó en la consola de ipython el comando %timelimit.

4.3 Resultados

Valor de la integral:

scipy.integrate.trapz: 1366.09079684
algoritmo: 1366.09079684
error: 0%

scipy.integrate.quad: 63200676.48016987
algoritmo: 63200680.1003
error:0.000108041744724%

Tiempo de ejecución:

scipy.integrate.trapz: 152 ns
algoritmo: 61.2 ns

scipy.integrate.quad: 155ns
algoritmo: 182.4ns (60.6 método de simpson+60.9 método del punto medio * 2)

4.4 Conclusiones

Según los resultados obtenidos, pude observar que en general mis funciones se ejecutaban más rápido que las de scipy. Esto puede deberse a que ellas son más complejas. Sin embargo, al llamar a más funciones pierdo tiempo de ejecución. La precisión de las funciones es muy satisfactoria.