Enseñanza de la mecánica cuántica asistida por documentos interactivos. Reporte de proyecto

Semillero Física Teórica y Computacional

16 de diciembre de 2015

Resumen

El uso de las nuevas tecnologías en la educación permite integrar herramientas interactivas que permiten un acercamiento mas simple a la exploración de conceptos y de la interpretación de como los parámetros de un sistema afectan su comportamiento. Una manera de lograrlo se puede aproximar mediante la separación del uso de las herramientas en 3 etapas, encargadas respectivamente de la visualización e interacción, la solución numérica y la exploración del concepto físico. Para este reporte se presenta como caso de estudio la exploración de las soluciones de estados ligados 1D.

1. Introducción

El uso de las herramientas computacionales a permitido transformar las metodologías de enseñanza y acercar áreas comúnmente muy teóricas y de largos desarrollos matemáticos a procesos que pueden centrarse mucho mas en el concepto físico que en la matemática requerida [1, 2]. Su aprovechamiento en su fondo requiere del desarrollo de la física computacional, que constituye el desarrollo de métodos computacionales para la solución de problemas físicos [1].

La solución completa del problema físico no constituye la sola obtención de un resultado numérico sino la adecuada representación de los mismos y su interpretación, la cual mediante visualización se puede asistir el proceso.

Alrededor de estas iniciativas, y con especial enfoque en la mecánica cuántica existen acercamientos mediante *applets* [3], aplicaciones en html5 y javascript [4], códigos con métodos simples como diferencias finitas [5] y graficación tradicional y experiencias en HUBs con NanoHUBS [6]. Se presenta un gran uso del lenguaje python para este tipo de aplicaciones cuando son de uso local (no web) [1, 2] por su sencilla sintaxis y amplio soporte.

Mas recientemente, la integración de documentos interactivos de formatos abiertos (ipython notebooks y sage notebooks) [7, 8] y cerrados (maple worksheets) [9] ha permitido llevar a un solo ambiente, continuo y natural, las simulaciones y visualizaciones con sus respectivos materiales de apoyo, con especial interés aquellos formatos libres que brindan una mayor accesibilidad.

2. Método

Se desarrollo una metodología de documentos interactivos usando el formato abierto de Jupyter Notebook (anteriormente IPython Notebook) con integración de elementos interactivos *widgets* para la exploración de parámetros [10].

El documento interactivo se divide en 3 documentos acorde a la separación metodológica de la solución del problema físico, que se expone como parte de los tópicos básicos requeridos en física computacional en distintos currículos [11, 2]

- Visualización e interacción: Se exponen las necesidades de representación de los datos y la forma de realización, así como el uso de los controles para la interacción.
- Técnicas numéricas. Se expone las técnicas de aproximación numérica y su aplicación al contexto para la búsqueda de raíces y ecuaciones diferenciales con problema de frontera, y la metodología para contribuir a la estabilidad de las soluciones mediante el rescalamiento de valores por adimensionalización de la ecuación diferencial.
- Conceptualización física. Se presentan los elementos físicos habilitando una interacción amplia de los parámetros físicos y numéricos del fenómeno expuesto. Al respecto se explora el caso de estudio de estados ligados y superposición.

La accesibilidad del recurso es posible gracias al almacenamiento gratuito y publico ofrecido por el servidor git de github[1], y la ejecución en un servidor local posible mediante la instalación libre de jupyter notebooks o de su ejecución en servicios en linea gratuitos o por suscripción como SageMath [2] o Authorea[3].

Para una mayor discusión de los tópicos de la metodología, remítase a los notebooks, cuya versión estática se anexa a este documento.

[1] https://github.com/
[2] https://cloud.sagemath.com/
[3] https://www.authorea.com/

3. Resultados

Se genera satisfactoriamente la integración en documentos interactivos de formato abierto y gratuitos de desarrollos numéricos con estrategia de visualización e interacción de conceptos de la mecánica cuántica, con la presentación de casos de estudio de estados ligados y el concepto de paquete de onda.

La estrategia planteada permite crear un apoyo al desarrollo de un curso de mecánica cuántica sin dependencia de conocimientos previos en programación ni requerimientos específicos de instalaciones, salvo contar con un navegador web y acceso a internet. Siendo así una estrategia reproducible en otras instituciones sin limitación por requerimientos de una plataforma para la ejecución de los documentos interactivos.

Se logra independizar claramente las etapas en la estrategia de solución y así permitir el abordaje del problema físico con 3 niveles según el interés que se posea en las herramientas numéricas y de visualización, o si solo se trata del apoyo como recurso para la exploración de parámetros.

Una versión estática de los notebooks se anexa al reporte. Para su versión interactiva esta puede ser descargada del repositorio github Cuantica_Jupyter[4] del semillero[5] y cargarlos en un servidor jupyter (de instalación local o de un servicio en linea como sagemath, el cual es recomendado).

Para simulaciones con tiempos de ejecución altos, el servidor notebook retira la instancia tras un cierto lapso de tiempo. Para este tipo de casos se recomienda el uso directo de ipython o python.

4. Conclusiones

Es posible integrar de una manera natural las etapas de solución de un problema físico sin generar un solapamiento de un elemento con otro, que afecte el proceso de aprendizaje en medio del desarrollo y lectura de códigos de visualización y simulación, al separar estos en términos de la presentación con su respectiva documentación que permita no solo interactuar el documento de apoyo del curso original sino también una interacción con los nuevos elementos que se incorporan.

Al no ser complejos los elementos nuevos, se puede usar el documento conceptual como apoyo a los currículos tradicionales sin requerir de algún conocimiento de programación. Los elementos nuevos en la metodológia, permiten un acercamiento para cursos con un enfoque en computación y visualización científica, para el cual si es requerido un conocimiento previo en programación.

Es posible aumentar la accesibilidad de este recurso a un mayor publico gracias a ser un formato y herramienta abierta y gratuita, permitiendo una fácil descarga e instalación en caso de uso local o el uso de servicios en linea (gratuitos o por suscripción) que ofrecen servidos de jupyter notebook, dependiendo solo de conectividad a internet sin requerimientos especiales de hardware.

Para potenciales no simétricos y con amplias diferencias entre el potencial máximo y mínimo se recomienda usar instancias directas de python e ipython, debido al comportamiento como servidor de las instancias de Jupyter, que provocan la terminación del llamado al kernel tras un *timeout*, de aproximadamente minuto y medio. Tras este periodo se recomienda reiniciar el kernel (tanto si es de ejecución local como en linea).

Referencias

[1] J.F. Rojas, M.A. Morales, A. Rangel, and I. Torres. Física computacional: Una propuesta educativa. *Revista Mexicana de Física E*, 55(1):97–111, 2 2009.

[2] Rubin H. Landau, Manuel J. Paez, Cristian Bordeianu, and Sally Haerer. Making physics education more relevant and accesible via computation and etextbooks. *Computer Physics Communications*, 182:2071–2075, 2011.

[3] Mario Belloni, Wolfgang Christian, and Anne J. Cox. *Physlet Quantum Physics 2E*. 2015.

[4] University of Colorado Boulder. Phet interactive simulations: Quantum phenomena. Online. Last Accesed date 12-12-2015.

[5] R. Garcia, A. Zozulya, and J. Stickney. Matlab codes for teaching quantum physics: Part 1. page 7, 4 2007. arXiv:0704.1622 [physics.ed-ph].

[6] G. Klimeck. Nanohub.org tutorial: Education simulation tools. In *Nano Micro Engineered and Molecular Systems*, page 41. IEEE, IEEE, 1 2007.

[7] Fernando Perez and Brian E. Granger. An open source framework for interactive, collaborative and reproducible scientific computing and education. 2013.

[8] Halen Shen. Interactive notebooks: Sharing the code. *Nature*, 515:151–152, 11 2014.

[9] Marko Horbatsch. *Quantum Mechanics Using Maple ®*. Springer, 1995.

[10] Project Jupyter. Jupyter documentation, 2015. Last Accessed date 12-12-2015.

[11] Rubin H. Landau. Computational physics, a better model for physics education? *Computing in Science and Engineering*, 8(5):50–58, 9 2006.

[4] https://github.com/fisicatyc/Cuantica_Jupyter

[5] https://github.com/fisicatyc

vis_int

December 16, 2015

1 Visualización e interacción

La visualización e interacción es un requerimiento actual para las nuevas metodologías de enseñanza, donde se busca un aprendizaje mucho más visual y que permita, a través de la experimentación, el entendimiento de un fenómeno cuando se cambian ciertas condiciones iniciales.

La ubicación espacial y la manipulación de parámetros en dicha experimentación se puede facilitar con herramientas como estas, que integran el uso de gráficos, animaciones y widgets. Este notebook, define los métodos de visualización e interacción que se usarán en otros notebooks, sobre la componente numérica y conceptual.

Esta separación se hace con el fin de distinguir claramente 3 componentes del proceso, y que faciliten la comprensión de la temática sin requerir que el usuario comprenda los 3 niveles (ya que el código es visible, y esto impactaría en el proceso de seguimiento del tema).

1.1 Funciones Matemáticas

Aunque no es parte de la visualización y de la interacción, el manejo de funciones matemáticas es requerido para estas etapas y las posteriores. Por lo que su definición es necesaria desde el principio para no ser redundante en requerir de múltiples invocaciones.

La evaluación de funciones matemáticas puede realizarse por medio del modulo `math` que hace parte de la biblioteca estandar de Python, o con la biblioteca `numpy`. Para el conjunto limitado de funciones matemáticas que requerimos y con la premisa de no realizar de formas complejas nuestros códigos, las utilidades de `numpy` no serán necesarias y con `math` y el uso de listas será suficiente.

El motivo de tener pocos requerimientos de funciones matemáticas es por el uso de métodos numéricos y no de herramientas analíticas. La idea es mostrar como con esta metodología es posible analizar un conjunto mayor de problemas sin tener que profundizar en una gran cantidad de herramientas matemáticas y así no limitar la discusión de estos temas a conocimientos avanzados de matemáticas, y más bien depender de un conocimiento básico tanto de matemáticas como de programación para el desarrollo de los problemas, y permitir simplemente la interacción en caso de solo usar estos notebooks como un recurso para el estudio conceptual. Por este último fin, se busca que el notebook conceptual posea el mínimo de código, y este se lleve sobre los notebooks de técnicas numéricas y de visualización.

```
In [1]: from math import sin, cos, tan, sqrt, log, exp, pi
```

El conjunto anterior de funciones sólo se indica por mantener una referencia de funciones para cualquier ampliación que se desee realizar sobre este, y para su uso en la creación de potenciales arbitrarios, así como en los casos de ejemplificación con funciones analíticas o para fines de comparación de resultados.

Para la implementación (partiendo de un potencial dado numéricamente), sólo se requiere del uso de `sqrt`.

El modulo de `numpy` permitiría extender la aplicación de funciones matemáticas directamente sobre arreglos numéricos, y definir estos arreglos de una forma natural para la matemática, como equivalente a los vectores y matrices a traves de la clase `array`.

1.2 Interacción

Existen multiples mecanismos para interacción con los recursos digitales, definidos de forma casi estándar en su comportamiento a través de distintas plataformas.

Dentro de la definición de los controles gráficos (widgets) incorporados en **Jupyter** en el módulo `ipywidgets`, encontramos los siguientes:

```
In [2]: import ipywidgets
        print(dir(ipywidgets))
```

```
['Accordion', 'BoundedFloatText', 'BoundedIntText', 'Box', 'Button', 'CallbackDispatcher', 'Checkbox',
```

Para nuestro uso, serán de uso principal:

- Interacciones: Son mecanismos automáticos para crear controles y asociarlos a una función. `interact`, `interactive`.
- Deslizadores: Los hay específicos para tipos de datos, y estos son `IntSlider` y `FloatSlider`.
- Botones: Elementos que permiten ejecutar una acción al presionarlos, `Button`.
- Texto: Permiten el ingreso de texto arbitrario y asociar la ejecución de una acción a su ingreso. `Text`.
- Contenedores: Permiten agrupar en un solo objeto/vista varios controles. Uno de ellos es `Box`.

```
In [3]: from ipywidgets import interact, interactive, fixed, IntSlider, FloatSlider, Button, Text, Box
```

Entre estos controles que se usan, a veces es necesario crear dependencias de sus rangos respecto al rango o propiedad de otro control. Para este fin usamos la función `link` del módulo `traitlets`. En este módulo se encuentran otras funciones utiles para manipulación de los controles gráficos.

```
In [4]: from traitlets import link
```

Tambien es necesario el uso de elementos que permitan el formato del documento y visualización de elementos y texto enriquecido, fuera de lo posible con texto plano a punta de `print` o con las capacidades de MarkDown (Nativo o con extensión). Para esto se puede extender el uso métodos para renderizado HTML y LaTeX.

```
In [5]: from IPython.display import clear_output, display, HTML, Latex, Markdown, Math
```

1.3 Visualización

Por visualización entendemos las estrategias de representación gráfica de la información, resultados o modelos. Facilita la lectura rápida de datos mediante codificaciones de colores así como la ubicación espacial de los mismos. La representación gráfica no tiene por qué ser estática, y es ahí donde las animaciones nos permiten representar las variaciones temporales de un sistema de una forma más natural (no como un gráfico respecto a un eje de tiempo, sino vivenciando un gráfico evolucionando en el tiempo).

Para este fin es posible usar diversas bibliotecas existentes en python (en sus versiones 2 y 3), siendo la más común de ellas y robusta, la biblioteca Matplotlib. En el contexto moderno de los navegadores web, es posible integrar de una forma más natural bibliotecas que realizan el almacenamiento de los gráficos en formatos nativos para la web, como lo es el formato de intercambio de datos JSON, facilitando su interacción en el navegador mediante llamados a javascript.

Así, podemos establecer preferencias, como Matplotlib para uso estático principalmente o para uso local, mientras que para interacción web, usar bibliotecas como Bokeh.

Para este caso, sin profundidad en la interacción web, se usará Matplotlib.

```
In [6]: %matplotlib inline
        import matplotlib.pyplot as plt
```

Para indicar la graficación no interactiva embebida en el documento usamos la sguiente linea

```
%matplotlib inline
```

En caso de requerir una forma interactiva embebida, se usa la linea

```
%matplotlib notebook
```

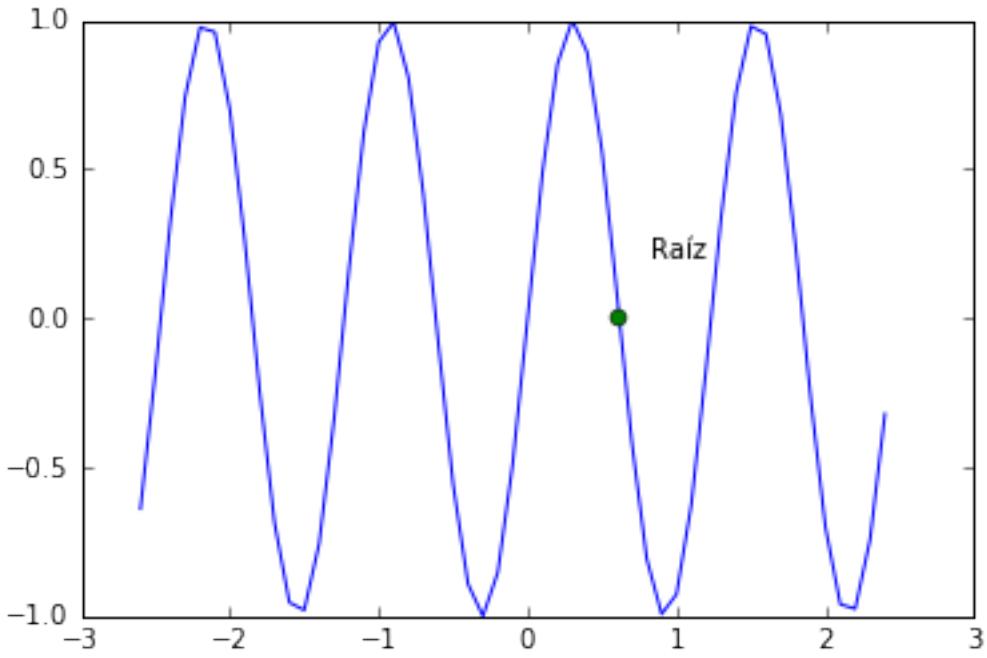
Para nuestro uso básico, todo lo necesario para gráficación se encuentra en el módulo `pyplot` de Matplotlib. Con él podemos realizar cuadrículas, trazos de curvas de diversos estilos, modificación de ejes, leyendas, adición de anotaciones en el gráfico y llenado de formas (coloreado entre curvas). Pueden consultarse ejemplos de referencia en la galería de Matplotlib y en la lista de ejemplos de la página oficial.

1.3.1 Graficación de funciones

En general nuestro ideal es poder graficar funciones que son representadas por arreglos numéricos. Las funciones continuas en su representación algebraica de discretizan, y es el conjunto de puntos interpolado lo que se ilustra. Antes de discretizar, es conveniente convertir nuestra función en una función evaluable, y asociar la dependencia solo a una variable (para nuestro caso que es 1D).

El proceso de interpolación mencionado se realiza por el paquete de graficación y nosotros solo debemos indicar los puntos que pertenecen a la función.

```
In [7]: def discretizar(funcion, a, b, n):
            dx = (b-a)/n
            x = [a + i*dx for i in range(n+1)]
            y = [funcion(i) for i in x]
            return x, y

        def graficar_funcion(x, f):
            plt.plot(x, f, '-')

        def graficar_punto_texto(x, f, texto):
            plt.plot(x, f, 'o')
            plt.text(x+.2, f+.2, texto)

In [8]: def int_raiz_sin(a:(-5.,0., .2), b:(0., 5., .2), k:(0.2, 10., .1), n:(1, 100, 1), N:(0, 10, 1)):
            f = lambda x: sin(k*x)
            x, y = discretizar(f, a, b, n)
            r = pi*(N + int(a*k/pi))/k
            graficar_funcion(x, y)
            graficar_punto_texto(r, 0, 'Raíz')
            plt.show()

        interact(int_raiz_sin)
```




```
Out[8]: <function __main__.int_raiz_sin>
```

El bloque anterior de código ilustra el uso de `interact` como mecanismo para crear controles automaticos que se apliquen a la ejecución de una función. Este permite crear de una forma simple las interacciones cuando no se requiere de personalizar mucho, ni vincular controles y se desea una ejecución automatica con cada variación de parametros. En caso de querer recuperar los valores especificos de los parametros para posterior manipulación se recomienda el uso de `interactive` o del uso explicito de los controles.

A pesar de la facilidad que ofrece `interact` e `interactive` al generar los controles automaticos, esto es poco conveniente cuando se trata de ejecuciones que toman tiempos significativos (que para escalas de una interacción favorable, un tiempo significativo son aquellos mayores a un segundo), ya que cada variación de parametros independiente, o sea, cada deslizador en este caso, al cambiar produce una nueva ejecución, y las nuevas variaciones de parámetros quedan en espera hasta terminar las ejecuciones de las variaciones individuales anteriores.

Es por esto, que puede ser conveniente definir una interacción donde los controles la unica acción que posean es la variación y almacenamiento de valores de los parametros, y sea otro control adicional el designado para indicar el momento de actualizar parametros y ejecutar.

El ejemplo anterior se puede construir usando `FloatSlider`, `IntSlider`, `Button`, `Text`, `Box` y `display`.

```
In [9]: def raiz_sin(a, b, k, n, N, texto):
            f = lambda x: sin(k*x)
            x, y = discretizar(f, a, b, n)
            r = pi*(N + int(a*k/pi))/k
            graficar_funcion(x, y)
            graficar_punto_texto(r, 0, texto)

        a = FloatSlider(value= -2.5, min=-5., max= 0., step= .2, description='a')
        b= FloatSlider(value = 2.5, min=0., max= 5., step=.2, description='b')
        k= FloatSlider(value = 5., min=0.2, max=10., step=.1, description='k')
        n= IntSlider(value= 50, min=1, max= 100, step=1, description='n')
```

```
N= IntSlider(value=5, min=0, max=10, step=1, description='N')
texto = Text(value='Raíz', description='Texto punto')
Boton_graficar = Button(description='Graficar')

def click_graficar(boton):
    clear_output(wait=True)
    raiz_sin(a.value, b.value, k.value, n.value, N.value, texto.value)
    plt.show()

display(a, b, k, n, N, texto, Boton_graficar)
Boton_graficar.on_click(click_graficar)
```

1.3.2 Graficación de potenciales

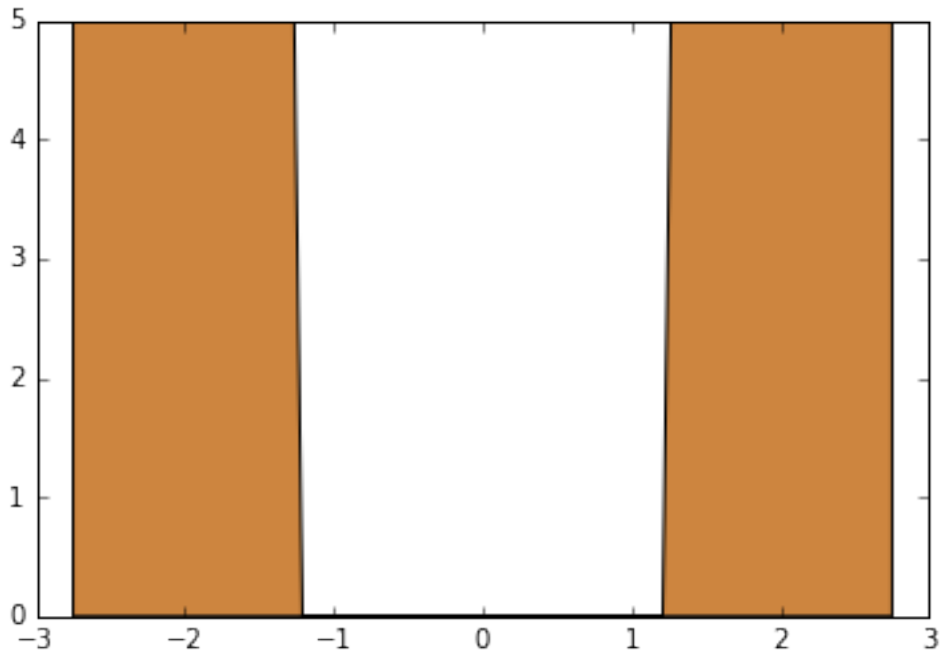
Para fines de ilustración y comprensión de los estados ligados del sistema, conviene poder ilustrar las funciones de potencial como barreras físicas. Esta noción gráfica se representa mediante el llenado entre la curva y el eje de referencia para la energía. De esta forma, al unir el gráfico con la referencia del autovalor, será claro que la energía hallada pertenece al intervalo requerido en teoría y que corresponde a un sistema ligado.

La función de graficación del potencial recibe dos listas/arreglos, uno con la información espacial y otro con la evaluación del potencial en dichos puntos. Antes de proceder con el llenado de la representación de la barrera del potencial, se crean los puntos inicial y final con el fin de crear formas cerradas distinguibles para el comando `fill`.

```
In [10]: def graficar_potencial(x, V_x):
             V_min = min(V_x)
             plt.fill_between(x, V_min, V_x, facecolor = 'peru')
```

A continuación se presenta un ejemplo interactivo de graficación del potencial finito. Se inicia con la definición del potencial, la cual se usa para generar un arreglo con la información de la evaluación del potencial en distintos puntos del espacio.

```
In [11]: def potencial(V_0, a, x):
             if abs(x) > a/2:
                 return V_0
             else:
                 return 0

         def int_potencial(V_0:(.1, 10., .1), a:(.1, 5, .1), L:(1., 10., .5), N:(10, 200, 10)):
             dx = L / N
             x = [-L/2 + i*dx for i in range(N+1)]
             y = [potencial(V_0, a, i) for i in x]
             graficar_potencial(x, y)
             plt.show()

         interact(int_potencial)
```




```
Out[11]: <function __main__.int_potencial>
```

1.3.3 Nivel de energía

Para ilustrar adecuadamente la presencia de estados ligados conviene superponer sobre la representación de la función de potencial, la referencia de energía del autovalor del sistema. Para distinguirlo, éste será un trazo discontinuo (no relleno para evitar confusión con el potencial, pero tampoco continuo para distinguirlo de la representación de las funciones de onda).

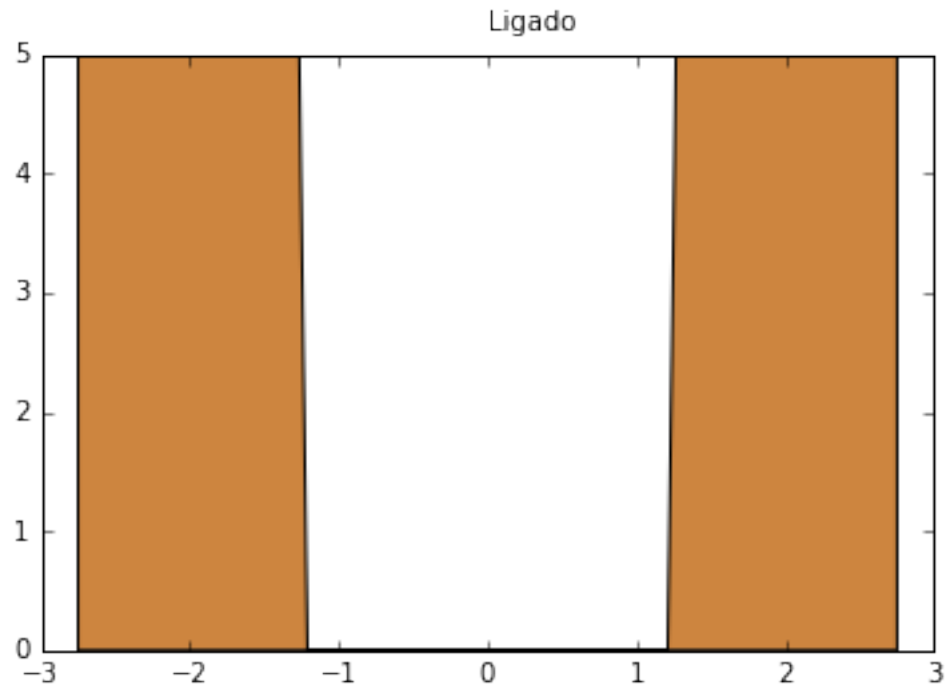
$$E \leq V_{\text{máx}}, \qquad \text{Estado ligado} \tag{1}$$

$$E > V_{\text{máx}}, \qquad \text{Estado no ligado} \tag{2}$$

Los estados no ligados son equivalentes a tener particulas libres.

```
In [12]: def graficar_autovalor(L, E):
             plt.plot([-L/2, L/2], [E, E], '--')
```

```
In [13]: def int_potencial_energia(V_0:(.1, 10., .1), E:(.1, 10., .1), a:(.1, 5, .1), L:(1., 10., .5),
             dx = L / N
             x = [-L/2 + i*dx for i in range(N+1)]
             y = [potencial(V_0, a, i) for i in x]
             graficar_potencial(x, y)
             graficar_autovalor(L, E)
             if E > V_0:
                 plt.text(0, E+0.2, 'No ligado')
             else:
                 plt.text(0, E+0.2, 'Ligado')
             plt.show()

         interact(int_potencial_energia)
```



Out[13]: <function __main__.int_potencial_energia>

1.3.4 Graficación de autofunciones

La visualización de las autofunciones (y su módulo cuadrado), nos permite reconocer visualmente la distribución de probabilidad del sistema e identificar los puntos espaciales más probables para la ubicación de la particula analizada.

Para la correcta visualización, la graficación de la función de onda debe considerar una normalización de escala, no necesariamente al valor de la unidad del eje, pero si como referencia un valor numerico comprendido por los valores máximos de potencial, que corresponden a la parte del gráfico más cercana al margen superior del recuadro de graficación. El no realizar este reescalamiento, podría afectar la visualización del potencial y de la energía, ya que el eje se reajusta a los datos máximos y mínimos.

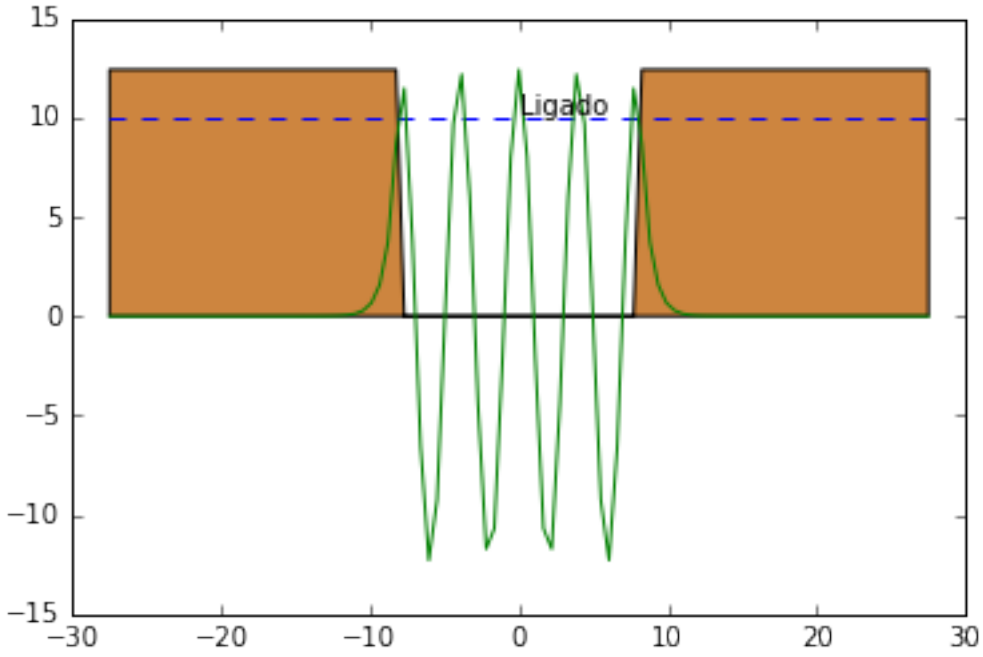
$$\psi'(x) = \frac{\psi(x)}{\max \psi(x)} V_{\text{máx}}$$

La graficación de las autofunciones es mediante el comando `plot` tradicional, y solo tiene de elemento adicional su reescalamiento con base al potencial máximo en la región de interes.

```
In [14]: def graficar_autofuncion(x, psi_x, V_max):
             psi_max = max([abs(i) for i in psi_x])
             escala = V_max / psi_max
             psi_x = [i*escala for i in psi_x]
             plt.plot(x, psi_x, '-')
```

```
In [15]: def onda(V_0, E, a, x):
             if abs(x) <= a/2:
                 return cos(sqrt(E)*x/2)
             else:
                 a2 = a/2
```

```
        k1 = sqrt(V_0 - E)
        A = cos(sqrt(E)*a2) / exp(-k1*a2)
        signo = abs(x)/x
        return A*exp(-signo*k1*x)

def int_potencial_auto_ef(V_0:(5., 20., .1), E:(.1, 20., .1), a:(2.5, 30., .1), L:(10., 100., 
    dx = L / N
    x = [-L/2 + i*dx for i in range(N+1)]
    V = [potencial(V_0, a, i) for i in x]
    f = [onda(V_0, E, a, i) for i in x]
    graficar_potencial(x, V)
    graficar_autovalor(L, E)
    graficar_autofuncion(x, f, V_0)
    if E > V_0:
        plt.text(0, E+0.2, 'No ligado')
    else:
        plt.text(0, E+0.2, 'Ligado')
    plt.show()

interact(int_potencial_auto_ef)
```



Out[15]: <function __main__.int_potencial_auto_ef>

tecnicas_numericas

December 16, 2015

Este notebook de ipython depende del modulo `vis_int`, el cual es ilustrado en el notebook de Visualización e Interacción.

```
In [1]: from vis_int import *
        import vis_int
        print(dir(vis_int))
```

```
['Box', 'Button', 'FloatSlider', 'HTML', 'IntSlider', 'Latex', 'Markdown', 'Math', 'Text', '__builtins__
```

1 Técnicas Numéricas

Para el desarrollo de los modelos expuestos en los cursos de mecánica cuántica y física moderna, se recurre frecuentemente a funciones especiales y técnicas de solución matemáticas que en su operatividad pueden distraer el objetivo del curso y el adecuado entendimiento de los conceptos físicos en medio de las herramientas matemáticas.

Con esto en mente, el uso de técnicas numéricas simples puede apoyar significativamente el desarrollo de los cursos, teniendo como ventaja la reducción a formas matemáticas simples (los métodos numéricos llevan a aproximaciones con operaciones aritmeticas) y funciones simples (las funciones se aproximan a funciones mucho más simples, generalmente polinomios). Esta reducción facilita además reducir a una sola técnica multiples desarrollos, ya que las diferencias no van tanto en los detalles del modelo original (como si dependen las soluciones análiticas) sino en los detalles del tipo de aproximación general.

Se exponen las soluciones numéricas de los siguientes problemas, útiles para el desarrollo de problemas 1D de mecánica cuántica.

1. Búsqueda de raíces.
+ Bisección.
+ Incremental.
1. Ecuaciones diferenciales con valores de frontera.
+ Método del disparo con algoritmo Numerov.
1. Adimensionalización.
+ Unidades atomicas de Rydberg.

1.1 Búsqueda de raíces

Los problemas de búsquedas de raíces corresponden a encontrar valores que al evaluarse en la función de interes generan como evaluación el valor cero. En la mecánica cuántica nos encontramos con la particularidad de requerir el calculo de raíces para determinar los autovalores de energía de un sistema en su planteamiento continuo (representación en el espacio directo). En estos sistemas de interes, de estados ligados, la energía del sistema se encuentra entre el mínimo y el máximo de la energía potencial a la que se encuentra sometido en el espacio,

$$V_{min} \leq E_n \leq V_{max}.$$

En caso de ser el máximo $V_{max} \to \infty$, el sistema posee infinitos autovalores que se encuentran con la condición $V_{min} \leq E_n$.

Para cualquiera de los casos, se presenta un interes en encontrar estos autovalores de manera ordenada, y esto lleva seleccionar los métodos de búsqueda cerrados por encima de los métodos de búsquedas abiertos, ya que en estos últimos la selección de un valor inicial no asegura la búsqueda en cercanías de este o en una dirección dada, por el contrario en los métodos cerrados se puede limitar la búsqueda a una región de la cual tenemos conocimiento que se presenta la raíz (autovalor de energía).

El uso combinado entre el método de búsqueda incremental y el método de bisección, con un paso adecuado de energía, permite cumplir con el objetivo de hallar todos los autovalores (cuando los límites de energía son finitos) del sistema de forma ordenada, y con precisión arbitraria (limitada solo por la precisión de máquina). Para ello se inicia en el intervalo de búsqueda con el método de búsqueda incremental, el cual al encontrar un intervalo candidato a raíz (un intervalo que presenta cambio de signo entre sus extremos), refina el resultado mediante la aplicación del método de bisección en el intervalo candidato.
Forma iterativa de búsqueda incremental $E_{i+1} = E_i + \Delta E$.
Forma iterativa de bisección $E_{i+1} = \frac{E_i + E_{i-1}}{2}$.

```
In [2]: def biseccion(funcion, a, b, tol_x = 1e-6, factor_ty = 1e2):
            f0 = funcion(a)
            f1 = funcion(b)
            if abs(f0) < tol_x: # Se verifica que los extremos sean raices
                return a
            elif abs(f1) < tol_x:
                return b
            else: # Si los extremos no son raices, se bisecta.
                c = (a + b) / 2.0
                f2 = funcion(c)
                while abs(f2) >= tol_x and abs(c - b) >= tol_x:
                    if f2 * f0 < 0 :
                        b = c
                        f1 = f2
                    else:
                        a = c
                        f0 = f2
                    c = (a + b) / 2.0
                    f2 = funcion(c)
                if abs(f2) < tol_x * factor_ty: # Se verifica que efectivamente sea raiz
                    return c
                else: # En caso de ser asintota vertical con cambio de signo
                    return None

        def incremental(funcion, a, b, delta_x = 1e-4, tol_x = 1e-6):
            c0 = a
            f0 = funcion(c0)
            c1 = c0 + delta_x
            c = None
            while c == None and c1 <=b: # Si no se ha hallado raíz y se esta en el intervalo, avance
                f1 = funcion(c1)
                while f0*f1 > 0 and c1 <= b:
                    c0 = c1
                    f0 = f1
                    c1 = c1 + delta_x
                    f1 = funcion(c1)
                if c1 > b: # Final del intervalo, equivalente f0*f1 > 0
                    return None
                else: # Sub-intervalo con cambio de signo
                    c = biseccion(funcion, c0, c1, tol_x) # Se invoca bisección para mejorar aproximaci
```

```
            if c == None: # Si el candidato era discontinuidad, incremental avanza
                c0 = c1
                f0 = f1
                c1 = c1 + delta_x
    return c
```

Se observa en la implementación del método de bisección, que se considera una revisión extra a los códigos tradicionales, con el fin de validad si el candidato a raíz realmente lo es. Esto se requiere ya que es posible que la función asociada a la discretización de la energía posea discontinuidades alrededor de las cuales presente cambio de signo.
Notese que la teoría clásica de métodos numéricos indica que estos métodos se aplican para funciones continuas. En este caso que esperamos discontinuidades dadas por cambios de signo a causa de divergencias al infinito, se pueden remover sistematicamente notando que a medida que se converge al candidato a raíz (tamaño de intervalo menor que la tolerancia), la evaluación de la función en este valor es significativamente mayor a la tolerancia, y cada vez su evaluación es mayor a la anterior.

$$E \in [E_i, Ei+1] \wedge \Delta E \leq tol \wedge \begin{cases} f(E) > tol, & \text{Discontinuidad} \\ f(E) \leq tol, & \text{Raíz (autovalor)} \end{cases} \tag{1}$$

Una vez se obtiene una raíz, el método de búsqueda incremental continua nuevamente avanzando hasta encontrar un próximo intervalo candidato, al cual vuelve a aplicarle el método de bisección para distinguir si es raíz o discontinuidad. Este proceso se continua hasta el límite superior para la energía, V_{max}.
Para la busqueda de un autovalor especifico, se requiere buscar todos los autovalores anteriores. De manera que se requiere de una función auxiliar que medie este progreso dada un modo. El caracter progresivo sobre las energías ofrece la ventaja sobre técnicas de autovalores, de la posibilidad de obtener los autovalores ordenados de manera natural.

```
In [3]: def raiz_n(funcion, a, b, N, delta_x = 1e-4, tol_x = 1e-6):
            c0 = a
            cont_raiz = 0
            while c0 < b and cont_raiz < N:
                c = incremental(funcion, c0, b, delta_x, tol_x)
                if c == None: # Si incremental termina en 'None', no hay más raíces
                    return None
                cont_raiz = cont_raiz + 1
                c0 = c + delta_x
            if cont_raiz == N:
                return c
            else:
                return None
```

A continuación se ilustra el uso de la técnica con la función trascendental del problema del pozo finito simetrico con paridad par, que en la forma adimensional corresponde a:

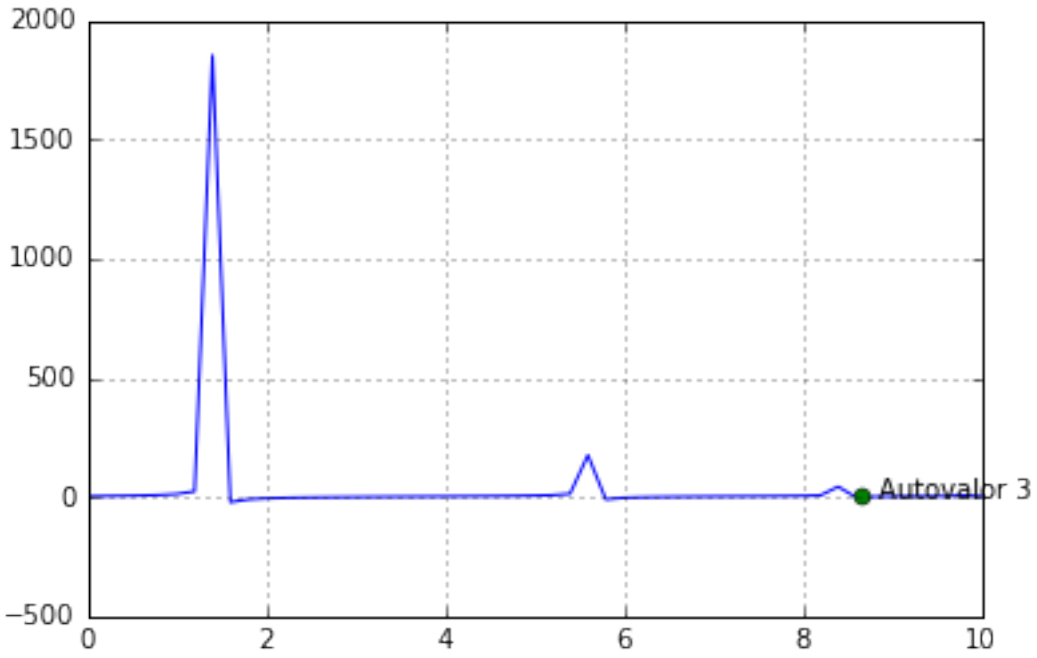
$$\sqrt{E} - \sqrt{V_0 - E} \tan\left(\frac{\sqrt{V_0 - E}a}{2}\right) = 0,$$

con a el ancho del pozo, V_0 es la profundidad del pozo (con referencia desde cero por convención).

```
In [4]: def trascendental(E, V_0, a):
            k2 = sqrt(V_0 - E)
            return sqrt(E) - k2*tan(k2*a/2)

        def int_raiz_trasc(V_0:(.1,20.,.1), a:(.1,15.,.1), N:(1, 6, 1), n:(1, 100, 1)):
```

```
        f = lambda E: trascendental(E, V_0, a)
        try:
            r = raiz_n(f, 0, V_0, N)
            E, tr = discretizar(f, 0, V_0, n)
            graficar_funcion(E, tr)
            graficar_punto_texto(r, 0, 'Autovalor ' + str(N))
            display(Latex('\(E_' + str(N) + '=' + str(r) + '\)'))
            plt.grid(True)
            plt.show()
            display(HTML('<div class="alert alert-warning">'+\
                         '<strong>Advertencia</strong> Alrededor de las discontinuidades'+\
                         ' el gráfico no es representado fielmente. </div>'))
        except ValueError:
            display(HTML('<div class="alert alert-danger">'+\
                 '<strong>Error</strong> Se evaluo la función en una discontinuidad.'+\
                 '</div>'))

interact(int_raiz_trasc)
```

(E_3=8.661879687493155)




```
<IPython.core.display.HTML object>
```

```
Out[4]: <function __main__.int_raiz_trasc>
```

1.2 Ecuaciones diferenciales con problemas de frontera

La ecuación de Schrödinger, ya sea dependiente o independiente del tiempo, es una ecuación diferencial de segundo orden. Al remover la dependencia del tiempo y disponer de problemas 1D, se tiene que la ecuación

diferencial es ordinaria. Para este tipo de ecuaciones (segundo orden, una variable) es posible aplicar métodos especificos que solución con bajo costo computacional aproximaciones de orden alto. Ejemplo de esto son los métodos de Verlet y de [Numerov](https://en.wikipedia.org/wiki/Numerov's_method), con método del disparo.

De la ecuación de Schrödinger se observa que si se reemplazan los valores por cantidades conocidas estimadas, el valor de la energía E que cumple con ser autovalor, es aquel que haga satisfacer las condiciones de frontera del problema, y por ende una forma de solucionar el problema es mediante la aplicación de un problema de busqueda de raices. De esta forma, el método del disparo lo que hace es el ajuste de E para que partiendo de una frontera, con la condicón respectiva, al propagarse hasta la otra frontera llegue con el valor de la otra condición. De no hacerlo, se cambio el valor de E y se repite el proceso.

El esquema de Numerov para la propagación es, dada una ecuación diferencial ordinaria de segundo orden sin termino lineal,

$$\frac{d^2y(x)}{dx^2} + K(x)y(x) = 0,$$

su esquema discreto se plantea como

$$y_{i+2} = \frac{\left(2 - \frac{5h^2K_{i+2}}{6}\right) y_{i+1} - \left(1 + \frac{h^2K_i}{12}\right) y_i}{\left(1 + \frac{h^2K_{i+2}}{12}\right)}.$$

Para nuestro caso, la función $K(x)$ posee dependencia de la energía, y todos los demás elementos son conocidos (la función solución, de onda en este caso, se construye iterativamente dado un valor de energía), por lo cual se puede definir una función que dada una energía como argumento, genere el valor de la función de onda en la frontera opuesta. Este valor en la frontera, por las condiciones establecidas por los potenciales y la condición de integrabilidad, debe ser $\psi(x_{izq}) = \psi(x_{der}) = 0$.

Tambien es usual usar como definición de la función, la diferencia de las derivadas logaritmicas en un punto intermedio, realizando la propagación desde ambos extremos. Por facilidad, se optará por la metodología clásica del disparo, que ofrece menor cantidad de operaciones y no presenta ambigüedades de definición, que consta de comparar con el valor esperado en la frontera opuesta, $Numerov(E) = 0$.

Este problema de búsqueda de raíces requiere conocer dos condiciones iniciales, la segunda debe tomarse acorde a la paridad de la energía buscada. Para paridad par, el segundo valor es positivo, mientras que para paridad impar el segundo valor es negativo.

La función `estacionario` se define para buscar el punto de empate adecuado para el análisis de la continuidad de la función de onda y su derivada. Como criterio, se buscan los turning points clásicos, donde $E = V(x)$.

```
In [5]: def estacionario(K, L, h):
            x = -L/2
            while x < L/2 and K(x) <= 0:
                x = x + h
            if x >= L/2:
                return L/2
            elif x == -L/2:
                return -L/2
            else:
                return x - h

        def numerov(K_ex, L, E, N, n):
            h = L / n
            K = lambda x: K_ex(E, x)
            p_est = estacionario(K, L, h)
            x = -L/2
            phi0 = 0.0
            x = x + h
```

```
    phi1 = 1e-10
    x = x + h
    while x <= p_est :
        term0 = 1 + h**2 * K(x - h) / 12
        term1 = 2 - 5 * h**2 * K( x) / 6
        term2 = 1 + h**2 * K(x + h) / 12
        aux = phi1
        phi1 = (term1 * phi1 - term0 * phi0) / term2
        phi0 = aux
        x = x + h
    phi_i_1 = phi1
    phi_i_0 = phi0
    x = L/2
    phi0 = 0.0
    x = x - h
    phi1 = 1e-10 * (-1)**(N%2 + 1)
    x = x - h
    while x > p_est :
        term0 = 1 + h**2 * K(x + h) / 12
        term1 = 2 - 5 * h**2 * K(x) / 6
        term2 = 1 + h**2 * K(x - h) / 12
        aux = phi1
        phi1 = (term1 * phi1 - term0 * phi0) / term2
        phi0 = aux
        x = x - h
    phi_d_1 = phi_i_1
    phi_d_0 = phi0 * phi_i_1 / phi1
    return (2*phi_d_1 - (phi_i_0+phi_d_0)) / (phi_d_0 - phi_i_0)


def Phi(K_ex, L, E, N, n):
    h = L / n
    K = lambda x: K_ex(E, x)
    p_est = estacionario(K, L, h)
    x = -L/2
    x_g = [x]
    phi0 = 0.0
    phi_g = [phi0]
    x = x + h
    phi1 = 1e-10
    x_g.append(x)
    phi_g.append(phi1)
    x = x + h
    while x <= p_est:
        term0 = 1 + h**2 * K(x - h) / 12
        term1 = 2 - 5 * h**2 * K(x) / 6
        term2 = 1 + h**2 * K(x + h) / 12
        aux = phi1
        phi1 = (term1 * phi1 - term0 * phi0) / term2
        x_g.append(x)
        phi_g.append(phi1)
        phi0 = aux
        x = x + h
    x = L/2
```

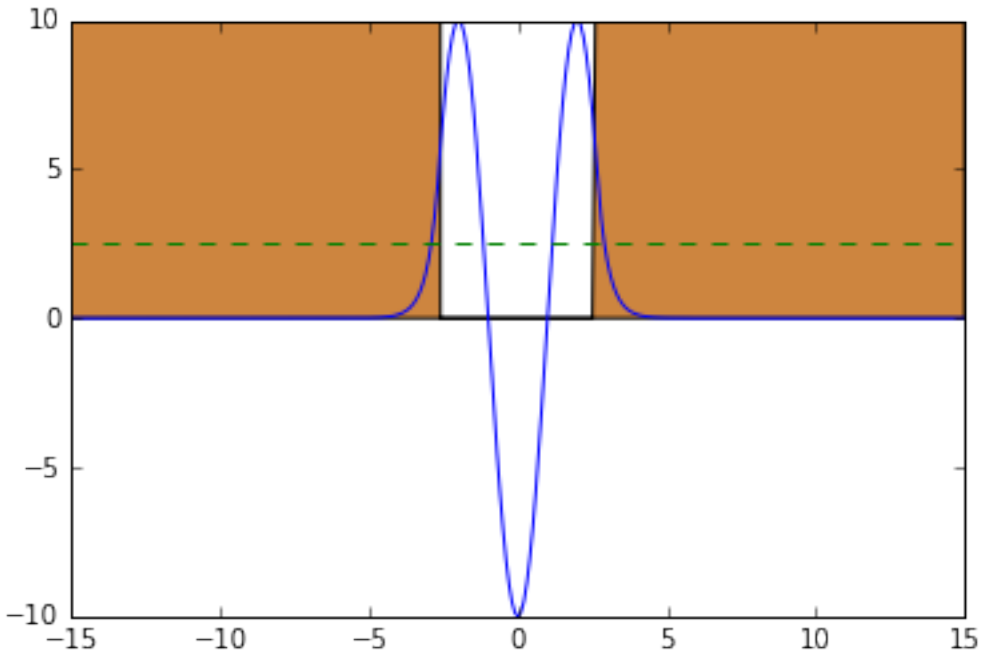
```
        phi0 = 0.0
        x_gd = [x]
        phi_gd = [phi0]
        x = x - h
        phi1 = 1e-10 * (-1)**(N%2 + 1)
        x_gd.insert(0, x)
        phi_gd.insert(0, phi1)
        x = x - h
        while x > p_est:
            term0 = 1 + h**2 * K(x + h) / 12
            term1 = 2 - 5 * h**2 * K(x) / 6
            term2 = 1 + h**2 * K(x - h) / 12
            aux = phi1
            phi1 = (term1 * phi1 - term0 * phi0) / term2
            x_gd.insert(0, x)
            phi_gd.insert(0, phi1)
            phi0 = aux
            x = x - h
        n_d = len(phi_gd)
        phi_gd = [phi_gd[i] * phi_g[-1] / phi1 for i in range(n_d)]
        x_g.extend(x_gd)
        phi_g.extend(phi_gd)
        return x_g, phi_g
```

Para la ecuación de Schrödinger, $K(x) = E - V(x)$.

```
In [6]: def K_Schr(V_0, a):
            return lambda e, x: e - potencial(V_0, a, x)
```

Para ilustrar el método del disparo, se presenta el siguiente control. La idea es ajustar para una configuración de potencial V_0, ancho a, longitud total L y numero de elementos de discretización n, la energía E adecuada para observar continuidad en la función de onda y su derivada en todo el intervalo. Dada la implementación del método se verifica dicha continuidad en el limite de la primera pared. Más adelante se define de manera general como seleccionar el punto de comparación.

```
In [7]: def disparo(V_0, a,  L, n, N, E):
            x, phi = Phi(K_Schr(V_0, a), L, E, N, n)
            V = [potencial(V_0, a, i) for i in x]
            graficar_potencial(x, V)
            graficar_autofuncion(x, phi, V_0)
            graficar_autovalor(L, E)
            plt.show()

        def presion_disparo(boton):
            disparo(V_0, a.value,  L, n.value, N, E.value)

        interact(disparo, V_0=(0., 20., .5), a=(.5, 10., .1), L=(10., 50., 5.), n=(100, 500, 50), N=fix
```




```
Out[7]: <function __main__.disparo>
```

La anterior ilustración tambien permite observar los efectos del potencial sobre un paquete de onda cuando la energía es menor o mayor que el potencial. Se puede observar como para $E > V_0$, se obtiene una función de onda oscilante en todo el intervalo, equivalente a una particula libre.

Se define la función `E_N` para el calculo de las autoenergías, que pueden incluirse en la función `Phi` para generar las autofunciones. Esta función puede explorarse en el notebook de Estados ligados.

```
In [8]: def E_N(K, E_max, L, N, n, delta_e = 1e-4, tol_e = 1e-6):
            Numerov = lambda e: numerov(K, L, e, N, n)
            return raiz_n(Numerov, tol_e, E_max, N, delta_e, tol_e)
```

```
In [9]: def Solve_Schr(Vx, E_max, L, N, n):
            x_vec, V_vec = discretizar(Vx, -L/2, L/2, n)
            V_min = min(V_vec)
            K = lambda e, x : e - Vx(x) + V_min
            E = E_N(K, E_max - V_min, L, N, n)
            if E != None:
                x_vec, phi = Phi(K, L, E, N, n)
                E = E + V_min
                display(Latex('\(E_{' + str(N) + '} = ' + str(E) + '\)'))
                V_vec = [Vx(i) for i in x_vec]
                graficar_potencial(x_vec, V_vec)
                V_max = max(V_vec)
                V_ref = max(abs(V_min), V_max)
                graficar_autofuncion(x_vec, phi, V_ref)
                graficar_autovalor(L, E)
                plt.show()
                return E, x_vec, phi
            else:
```

```
                display(HTML('<div class="alert alert-danger">'+\
                    '<strong>Error</strong> Se evaluo la función en una discontinuidad.'+\
                    '</div>'))
```

El siguiente bloque define la base de los controles, `fun_contenedor_base`, para el notebook conceptual, Estados ligados, donde los parametros de máxima energía de búsqueda, longitud de interes, número de estado y particiones son comunes.

```
In [10]: def fun_contenedor_base():
             E_max = FloatSlider(value=10., min = 1., max=20., step=1., description= '$E_{max}$')
             L = FloatSlider(value = 30., min = 10., max = 100., step= 1., description='L')
             N = IntSlider(value=1, min=1, max= 6, step=1, description='N')
             n = IntSlider(value= 300, min= 100, max= 500, step=20, description='n')
             return Box(children=[E_max, L, N, n])

         Contenedor_base = fun_contenedor_base()
         display(Contenedor_base)

In [11]: def agregar_control(base, control):
             controles = list(base.children)
             controles.append(control)
             base.children = tuple(controles)

         control_prueba = fun_contenedor_base()
         agregar_control(control_prueba, Text(description='Casilla de texto para prueba'))
         display(control_prueba)
```

La función `agregar_control` permite agregar controles adicionales a la base para crear el control especifico de los casos de prueba.

1.3 Adimensionalización

Para fines de la solución numérica, conviene definir las unidades atomicas de Rydberg. El uso de estas unidades permite hacer comparables los ordenes de maginitud tan dispares que poseen las variables involucradas y así controlar el error numérico que pueda tener el algoritmo.

$\hbar = 2m_e = \frac{e^2}{2} = 1$

Para fines de ilustración se considera que el problema se resuelve solo para electrones, de manera que el problema se hace independiente de la masa.

La energía se mide en Rydbergs y las longitudes en radios de Bohr.

$1\ Ry = 13.6\ eV, \qquad 1\ a_0 = 5.29 \cdot 10^{-11}\ m$

Realizando las sustituciones correspondienes, la forma adimensional de la ecuación Schrödinger es.

$-\frac{d^2\psi(x)}{dx^2} + V(x)\psi(x) = E\psi(x)$

estados_ligados

December 16, 2015

Este notebook de ipython depende de los modulos:

`tecnicas_numericas`, ilustrado en el notebook Técnicas numéricas.

`vis_int`, ilustrado en el notebook Visualización e Interacción (esta incluido en el `import` a `tecnicas_numericas`).

```
In [1]: from tecnicas_numericas import *
        import tecnicas_numericas
        print(dir(tecnicas_numericas))
```

```
['Box', 'Button', 'E_N', 'FloatSlider', 'HTML', 'IntSlider', 'K_Schr', 'Latex', 'Markdown', 'Math', 'Phi
```

1 Estados ligados

El presente documento cumple como función ilustrar de forma interactiva el comportamiento de las soluciones de la ecuación de Schrödinger estacionaria para estados ligados en problemas 1D con la aplicación del método de Numerov.

1.1 Simulación

La aplicación del método del disparo con el algoritmo de Numerov, implica la búsqueda de raíces para encontrar los autovalores de energía. Su forma de proceder es mediante el avance regular en pasos de energía entre un mínimo y máximo hasta encontrar un cambio de signo en la evaluación la función de onda (o criterio equivalente, como la derivada logaritmica de la misma) hasta el punto de comparación. La presencia de este cambio de signo indica que existe una energía E en el intervalo $[E_i, E_{i+1}]$ que es o raíz de la función de Numerov (por tanto autovalor del sistema) o una discontinuidad. Estas raíces y discontinuidas son asociadas a la función equivalente de cuantización de la energía, como en el problema típico de potencial finito lo es la ecuación trascendental (sin embargo, ésta no aparace explicitamente en el modelo númerico).

1.1.1 Funciones de potencial

Las soluciones de autovalores de energía de un sistema se asocian al potencial y geometría del sistema (condiciones de frontera para un problema sobre una línea por ser 1D), las cuales se deben imponer acorde a las condiciones físicas de interes. Para las condiciones de frontera sabemos que en los extremos estas deben anularse, así que solo hace falta describir el potencial al cual se esta sujeto.

Pozo infinito y finito

El potencial del pozo infinito es descrito como:

$$V(x) = \begin{cases} 0 & |x| < \frac{a}{2} \\ \infty & |x| \geq \frac{a}{2} \end{cases} \tag{1}$$

Para el potencial de un pozo finito su potencial se puede describir como:

$$V(x) = \begin{cases} 0 & |x| < \frac{a}{2} \\ V_0 & |x| \geq \frac{a}{2} \end{cases} \tag{2}$$

Estos pozos son los casos básicos de estudio por la facilidad para su desarrollo análitico e interpretación sencilla. Se puede ver en estos casos de estudio aplicaciones en ...

```
In [2]: def V_inf(x):
            return 0

        def V_fin(V_0, a, x):
            if abs(x) < a/2:
                return 0
            else:
                return V_0
```

Para efectos numéricos el infinito se traslada a una longitud grande comparativamente al ancho del pozo, la cual se designará como L. En el caso de que $a = L$, corresponde justamente al pozo infinito, de manera que la simulación de estos dos casos requiere un solo control y es basado en el potencial finito.

```
In [3]: control_pozo = fun_contenedor_base()
        agregar_control(control_pozo, FloatSlider(value = 5.2, min = .5, max= 10., step= .1, descriptio
        pozo_link = link((control_pozo.children[1], 'min'), (control_pozo.children[4], 'value'))
        boton_pozo = Button(description='Simular pozo')
        def click_pozo(boton):
            V_max = control_pozo.children[0].value
            L = control_pozo.children[1].value
            N = control_pozo.children[2].value
            n = control_pozo.children[3].value
            a = control_pozo.children[4].value
            Vx = lambda x: V_fin(V_max, a, x)
            Solve_Schr(Vx, V_max, L, N, n)
            clear_output(wait=True)

        boton_pozo.on_click(click_pozo)
        display(control_pozo, boton_pozo)
```

Potencial armonico

El potencial armónico cumple con la descripción dada por

$$V(x) = \frac{\omega^2 x^2}{4} \tag{3}$$

```
In [4]: def V_arm(omega, x):
            return omega**2 * x**2 / 4
```

```
In [5]: control_arm = fun_contenedor_base()
        agregar_control(control_arm, FloatSlider(value = 1., min = .1, max= 4., step= .1, description='
        boton_arm = Button(description='Simular potencial')
        def click_arm(boton):
            E_max = control_arm.children[0].value
            L = control_arm.children[1].value
            N = control_arm.children[2].value
            n = control_arm.children[3].value
            omega = control_arm.children[4].value
            Vx = lambda x: V_arm(omega, x)
```

```
        Solve_Schr(Vx, E_max, L, N, n)
        clear_output(wait=True)

    boton_arm.on_click(click_arm)
    display(control_arm, boton_arm)
```

Potencial arbitrario

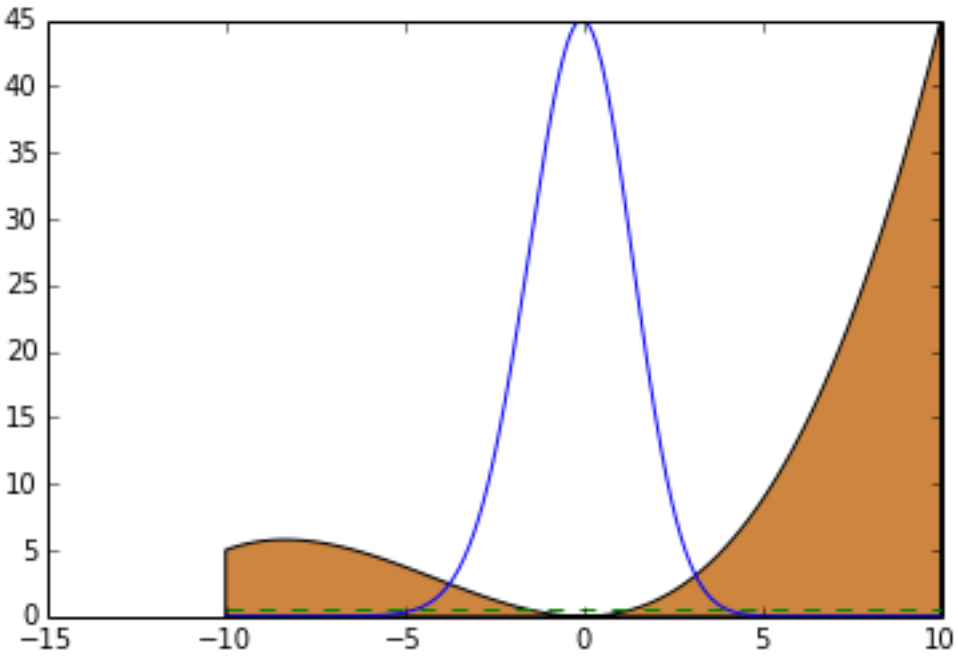
El problema 1D puede ser resuelto para un problema de potencial arbitrario $V(x)$, donde `str_potencial` es un string que representa la función del potencial. Debe tenerse en cuenta, tanto para este caso como los anteriores, que numéricamente el infinito es representado como una escala mayor por un cierta cantidad que la escala de interes del sistema.

Actividad : Proponga una función potencial de interes y desarrolle el bloque de código requerido para simularlo con este notebook. Use como base las funciones desarrolladas en los notebooks y el bloque siguiente.

El bloque siguiente ilustra un problema de potencial armónico con anarmonicidad.

In [6]:
```
control_arb = fun_contenedor_base()
E_max = control_arb.children[0]
L = control_arb.children[1]
N = control_arb.children[2]
n = control_arb.children[3]
n.value = 300
L.value = 20.

str_potencial = Text(value='x**2 / 4 + x**3 / 50', description= 'Potencial')
str_potencial.funcion = lambda x: eval(str_potencial.value)
agregar_control(control_arb, str_potencial)
# Ingrese un texto en formato python con dependencia solo de 'x'.

def ingreso_potencial(str_potencial):
    str_potencial.funcion = lambda x: eval(str_potencial.value)
    Vx = str_potencial.funcion
    h = L.value / n.value
    V_vec = [Vx(-L.value/2 + h*i) for i in range(n.value + 1)]
    V_min = min(V_vec)
    V_max = max(V_vec)
    dV = (V_max - V_min) / 50
    E_max.step = dV
    E_max.min = V_min
    E_max.max = V_max + (V_max - V_min)
    E_max.value = V_max

ingreso_potencial(str_potencial)
boton_arb = Button(description='Simular potencial')

def click_arbitrario(boton):
    Vx = str_potencial.funcion
    Solve_Schr(Vx, E_max.value, L.value, N.value, n.value)
    clear_output(wait=True)

str_potencial.on_submit(ingreso_potencial)
boton_arb.on_click(click_arbitrario)
display(control_arb, boton_arb)
```

(E_{1} = 0.48729084375)



Para potenciales no simétricos y con amplias diferencias entre el potencial máximo y mínimo se recomienda usar instancias directas de Python e IPython, debido al comportamiento como servidor de las instancias de Jupyter, que provocan la terminación del llamado al kernel tras un *timeout*, de aproximadamente minuto y medio. Tras este periodo se recomienda reiniciar el kernel (tanto si es de ejecución local como en línea).

El caso con los siguientes parámetros toma 256 s su solución y no es posible realizarlo en el notebook.
$+ L = 20 + N = 1 + n = 300 + V(x) = \frac{x^2}{4} + 0.3x^3 + E_{max} = 325$

Se encuentra $E_1 = -241.0251$ y la función de onda graficada a continuación.

In [ ]:

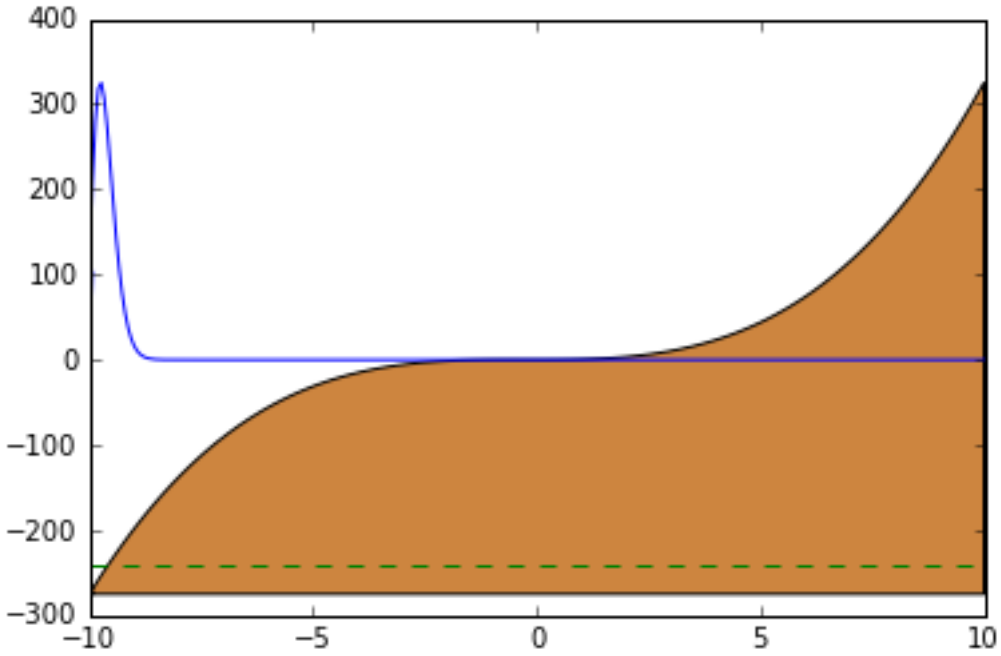


Figure 1: Función de onda solución

Superposicion

December 16, 2015

1 SUPERPOSICIÓN DE ONDAS

Se comprende claramente que una onda consiste en la propagación de una perturbación a través de cualquier medio, implicando el transporte de energía y no el de materia. Sin embargo al hablar de superposición de onda estamos considerando la presencia de dos o más ondas en un mismo medio que se superponen en la misma región del espació y cuyo resultado es la formación de una nueva onda debido a la suma algebraica de sus ondas consecutivas, siendo esta una situación que sugiere el principio de superposición.

Una de las propiedades de las ondas es que se extiende por todo el espacio. Matemáticamente y para un caso monodimensional, se expresa de la siguiente manera:

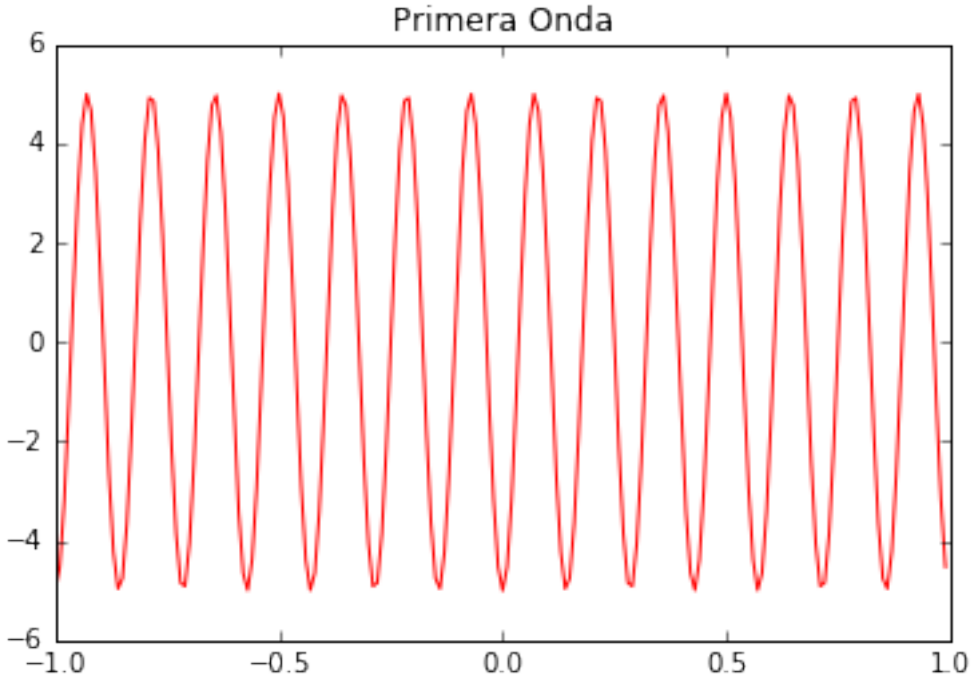
$$E(x,t) = E_0 cos(kx - \omega t) \tag{1}$$

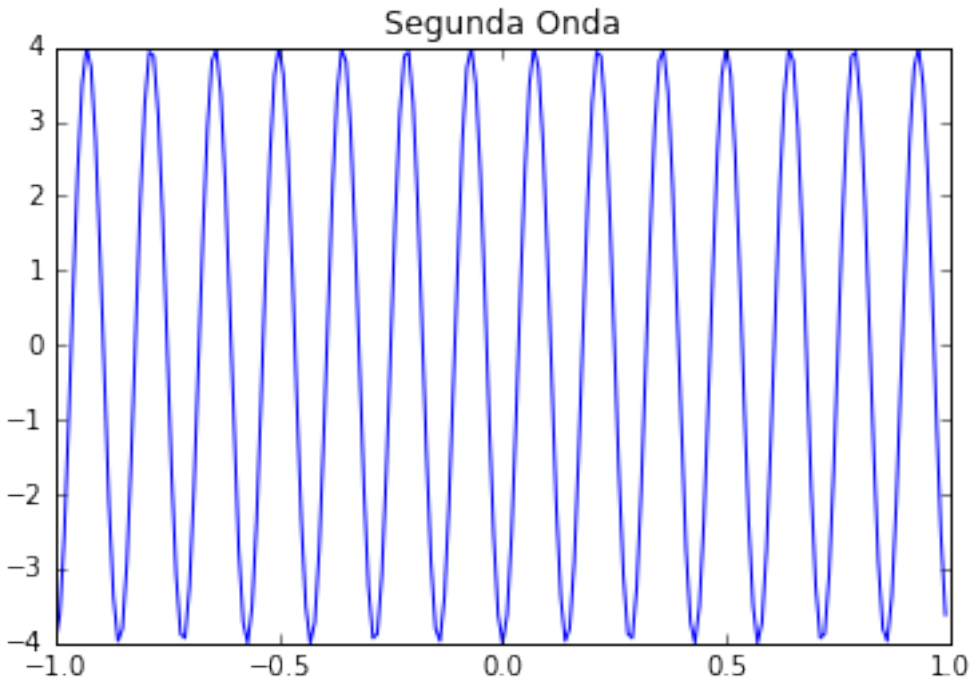
Donde $\underline{E_0}$ corresponde a la amplitud, $\underline{k}$ corresponde al número de onda, y $\underline{\omega}$ corresponde a la frecuencia angular.

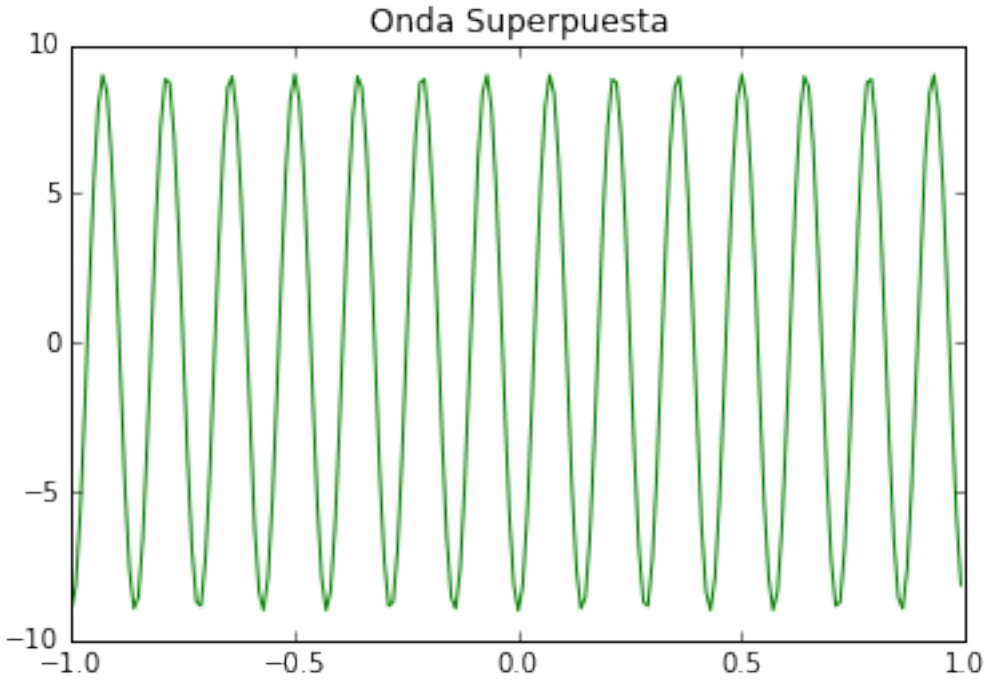
In [1]:
```
"""Bibliotecas"""

import matplotlib.pyplot as plt
from numpy import *
from ipywidgets import *
from IPython.display import * # Importa los métodos de renderizado
from math import factorial,exp
%matplotlib inline
#se crea un arreglo para definir en que valores se va a trabajar
t=arange(-1,1,0.01)
```

In [2]:
```
#se define la función que va a crear ambas ondas y la resultante

def pltsin(n1,f,n2):
    """
    n1= Amplitud de la primera onda
    n2= Amplitud de la segunda onda
    f= Frecuencia de la primera onda
    f2= Frecuencia de la segunda onda
    """
    plt.figure(1)
    plt.plot(t,n1*cos(2*pi*t*f),'r')#ecuación de la primera onda
    plt.title('Primera Onda')
    plt.show()
    # Se grafica la primera onda
    plt.figure(2)
    plt.plot(t,n2*cos(2*pi*t*f))#ecuación de la segunda onda
    plt.title('Segunda Onda')
    plt.show()
```

```
    #Se grafica la segunda onda
    plt.figure(3)
    plt.plot(t,(n2+n1)*cos(2*pi*t*f),'g')#ecuación de la superposición
    plt.title('Onda Superpuesta')
    plt.show()
    #Se grafica la superposición de ambas ondas
interact(pltsin,n1=(-10,10,1),f=(1,10,1),n2=(-10,10,1))
```

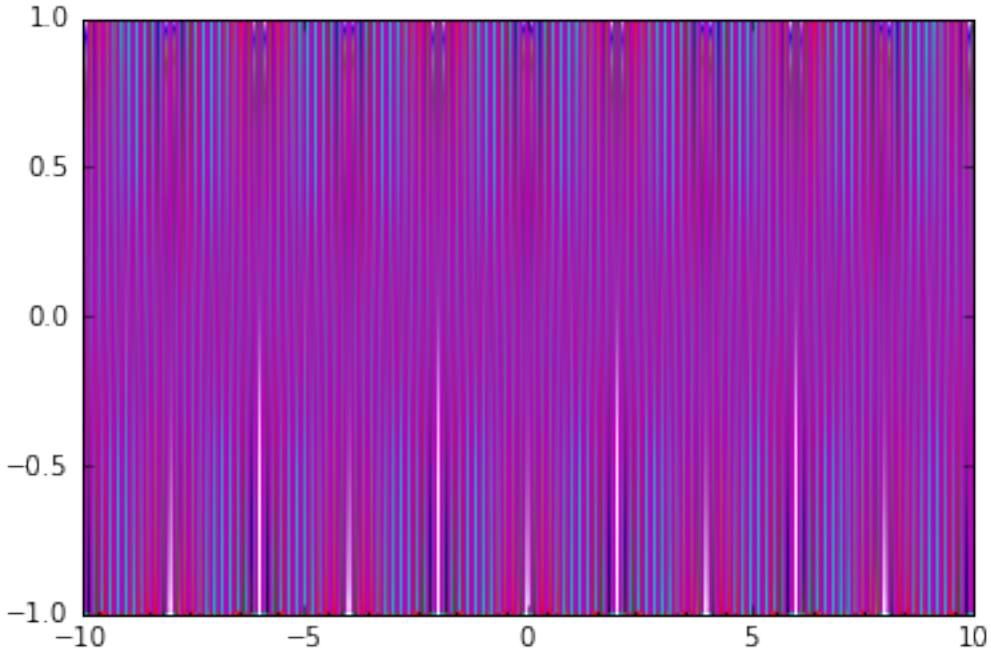


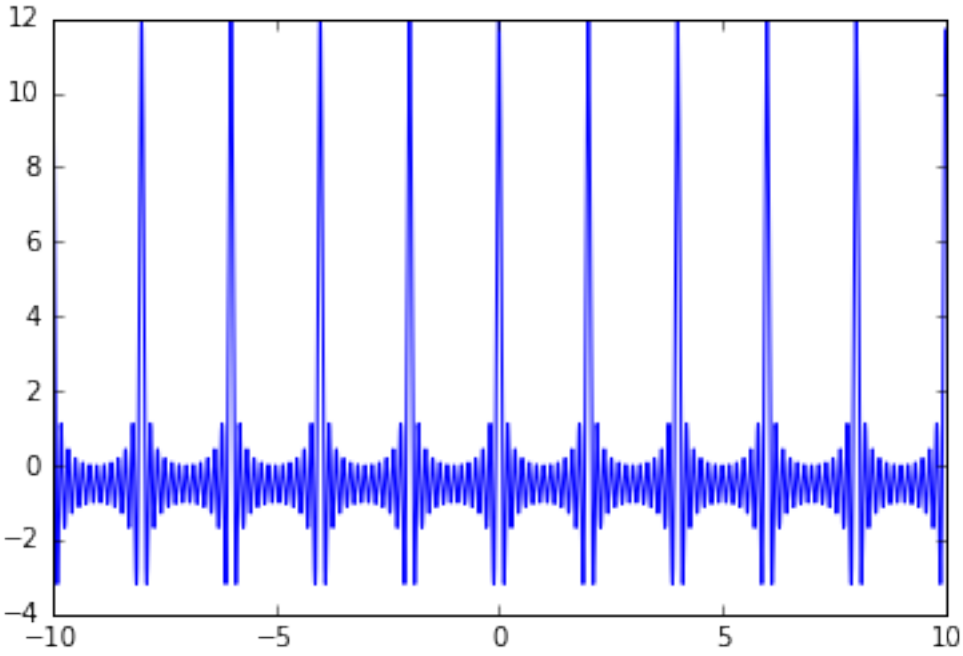




Las ondas visualizadas y las descritas en la ecuación 1, sin embargo, no se pueden representar a la forma de De Broglie, ya que debe haber una superposición de ondas con mismas amplitudes pero con diferentes números de ondas para que la onda resultante concentra la intensidad en ciertas regiones del espacio, frente a otros espacios donde se han amortiguado.

In [8]:
```
k=arange(-10,10,0.01)
def try2(n):
    res=0
    for i in range(1,n+1):

        plt.figure(6)
        plt.plot(k,cos(i*k*pi))
        res=res+cos(i*k*pi)
    plt.show()
    plt.figure(6)
    plt.plot(k,res)
    plt.show()

interact(try2,n=(1,12,1))
```





Se concluye que a medida que se va aumentando el número de ondas, las interferencias entre ellas hacen que se aumente el espacio de las regiones de esfuerzo y se obtenga finalmente una región con un único grupo central denominado paquete de ondas.

Para que exista una superposición de ondas que satisfaga el principio de De Broglie, la onda resultante debe cumplir con la siguiente condición:

"La una nueva onda debe resultar del producto entre una onda portadora de longitud de onda $\lambda = \frac{2\pi}{k}$ y un factor que modula la amplitud en forma de otra onda envolvente de longitud $\lambda- = \frac{2\pi}{k-}$.

1.1 Referencias

1. Superposición de ondas. [Citado el 13 de diciembre del 2015] [en línea]. http://www.sc.ehu.es/sbweb/fisica/cuantica/tunel/tunel.htm
2. Principio de superposición de ondas. [Citado el 13 de diciembre del 2015] [en línea]. https://es.wikipedia.org/wiki/Principio_de_superposici%C3%B3n_de_ondas
3. Paquetes de ondas. [Citado el 13 de diciembre del 2015] [en línea]. http://wdb.ugr.es/~bosca/Fisica-Cuantica/?p=403

Blackbody_radiation

December 16, 2015

1 La radiación de cuerpo negro - Fenómeno cuántico

Juan José Cadavid Muñoz - Semillero de Física Teórica y Experimental - Universidad EAFIT 2015.

1.1 Sobre el origen del fenómeno

La radiación de cuerpo fue un problema que surge en un contexto donde la física clásica está en auge. El mundo podía ser resumido por los deterministas en: La mecánica, termodinámica y electromagnetismo. Tras el éxito del modelo electromagnético de Maxwell, se intentó dar respuesta a cómo los cuerpos calientes emitían radiación.

Una de las primeras observaciones fue realizada en 1972 por Thomas Wedgood, quien observó que al tomar diferentes cuerpos sólidos de diferente composición, todos se tornaban de color rojizo a la misma temperatura. Más de 50 años después, los avances en la espectroscopía permitieron detallar que todo cuerpo luminiscente al calentarse emitía un espectro continuo.

1.2 La idea de absorción y emisión

En 1859, Gustav Kirchhoff demostró un teorema derivado de la termodinámica, en el que define que si un cuerpo está en equilibrio térmico con la radiación, la potencia de emisión (E_f $[W/m^2]$) era proporcional a la potencia de absorción, estableciendo la expresión:

$$E_f = J(f,T)A_f \tag{1}$$

donde $J(f,T)$ es una función de emisión universal que depende de la frecuencia de la radiación f y la temperatura T del cuerpo. A_f es la capacidad de absorción del cuerpo. Cuando el cuerpo tiene toda la absorción, $A_f = 1$. El teorema describe que si un cuerpo a una temperatura fija, éste cuerpo emite a una cierta frecuencia que es exacta a la que puede ser absorbida. Éste planteamiento, permitió reversar el problema del cuerpo negro para estudiar no cuánto emite sino cuánto absorbe. De ésta manera se plantea la idea de una **cavidad de cuerpo negro**.

1.3 La cavidad de cuerpo negro

Un cuerpo es un objeto idealizado en la capacidad de absorción. De ahí deriva su nombre, al tener capacidad absoluta de absorción $A_f = 1$ no hay luz que pueda ser percibida y por ende es negro. Pero al ser un absorbente absoluto, por el teorema de Kirchhoff, también es un emisor absoluto **un cuerpo blanco**, por lo que ambos problemas son equivalentes.

Una cavidad de cuerpo negro, es un sólido hueco que tiene las mismas propiedades del cuerpo negro, y en la superficie posee un orificio. Bajo la idea de Kirchhoff, toda la radiación que emite un cuerpo caliente saldrá de la cavidad,y por ende toda la radiación que incide sobre la cavidad será absorbida de una manera particular. Kirchhoff retó a experimentalistas y teóricos a entender y encontrar la forma de emisión/absorción en lo que él denominó como *hohlraumstrahlung*. Retó que guiaría posteriormente al inicio de la teoría cuántica.

Entonces la nueva pregunta que se plantea es sobre ¿Cómo la cavidad absorbe toda la radiación? La respuesta a ésta pregunta sería también equivalente a ¿Cómo la cavidad emite toda la radiación? Más adelante se discutirá las aproximaciones que se realizaron para resolver ésta pregunta.

1.4 Absorción de la radiación y naturaleza cuántica

En el contexto del problema, se tenia en conocimiento que el calor podía generar vibraciones en las moléculas y átomos de un sólido, y también se conocía que los átomos eran un conjunto de cargas eléctricas. Sin embargo los efectos atómicos se desconocía, principalmente porque los modelos clásicos fallaban en predecir éstos efectos, por lo que en sí del átomo se conocía parte de su naturaleza eléctrica. Del análisis en las ecuaciones de Maxwell se explicaba que si se tienen portadores de cargas oscilantes, campos electromagnéticos variables en el tiempo eran generados.

Con éstas dos ideas, en el panorama pintaba una idea de que si un cuerpo era calentado, las vibraciones térmicas en la escala atómica, era capaz de que los portadores de carga oscilaban y por ende eran los generadores de perturbaciones ondulatorias electromagnéticas. Las herramientas matemáticas permitían una aproximación en la combinación de la teoría electromagnética y la termodinámica clásica, y por su parte Kirchhkof había planteado un problema que llevaría la combinación de éstas nociones para entender la emisión de la cavidad. Lo anterior era el panorama que tenían los físicos a finales del siglo XIX para éste problema.

La primera aproximación fue experimental. Obtener una curva característica de un cuerpo negro, que con la ayuda de la ya desarrollada espectroscopia, permitiría establecer el comportamiento de los datos, por lo que se conocía una forma experimental del espectro de emisión de diferentes longitudes de onda, para algunas temperaturas.

El planteamiento entonces vendría en encontrar una forma analítica que ajustara los datos, por lo que con los hallazgos experimentales surgieron dos observaciones importantes:

1.4.1 Ley de Stefan - Boltzmann

Una de las contribuciones importantes sobre el estudio del cuerpo negro, fue la ley de Stefan propuesta por Josef Stefan, quien experimentalmente encontró una forma para calcular la cantidad de potencia total emitida por unidad de área en un cuerpo negro. La relación describe:

$$E_{total} = \int_0^\infty E_f \mathrm{d}f = \sigma\, T^4 \tag{2}$$

Posteriormente, Ludwig Boltzmann llega a éste mismo resultado, combinando la termodinámica y las ecuaciones de Maxwell, por lo que ésta ley se le conoce también como la ley de Stefan-Boltzmann. La estrategia de Boltzmann fue considerar la densidad de energía de un campo electromagnético en un volumen y unirla con la densidad de entropía. En éste caso Boltzmann estaba resolviendo el problema termodinámico de un pistón con un gas, pero consideraba la radiación como el gas en el pistón. El detalle de la deducción puede encontrarse en éste vínculo: Black Body Radiation.

1.4.2 Ley de desplazamiento de Wien

La segunda contribución al problema fue propuesta por Wilhelm Wien, siguiendo en parte a la idea de tratar la densidad de energía de la cavidad. La idea de Wien era describir el corrimiento del pico espectral mayor respecto al aumento de temperatura y de ésta manera observar cómo el cuerpo cambiaba el color que era emitido, comenzando por el rojo, que sería el primer colo que se vería temperaturas de $900K$ hasta llegar a temperaturas de $10000K$ para observar el azul. La ley establece:

$$f_{max} \propto T \therefore f_{max} = 5.879 \times 10^{10} T \tag{3}$$

donde $5.879 \times 10^{10}[Hz\, T^{-1}]$ es la constate de proporción experimental propuesta, tras el barrido de temperaturas en el experimento de Wien. Para 1893, él deduce analíticamente la conjetura de la Ley de radiación de Wien estableciendo la relación entre la densidad de energía como:

$$\rho(f,T) = \alpha\, f^3 \mathrm{e}^{-\beta\, f\, T^{-1}} \tag{4}$$

1.5 El modelo teórico y los hallazgos experimentales

En 1895, Wien y Lummer realizaron un experimento en la universidad de Berlin, con el que contaban con un horno virtualmente cerrado, salvo por el pequeño orificio para dejar salir la radiación y de ésta forma obtener una aproximación a la cavidad de cuerpo negro.

En el experimento, se instaló una red de difracción a la salida de la cavidad para observar las diferentes longitudes de ondas emitidas por la cavidad. Para determinar la densidad de energía un detector de intensidad era desplazado en cada longitud difractada para determinar su valor de intensidad. En el experimento sólo se medio la radiación visible, eliminando las altas frecuencias a través de reflexiones en diferentes cristales, de ésta manera tener fiabilidad en la energía medida correspondía al espectro visible.

La medición del detector no era directamente sobre la densidad de energía $\rho(f,T)$ sino sobre una fracción de ésta. Debido a que hay una cantidad de entrada y salida de densidad, sólo la mitad salía de la cavidad, al igual que una porción atravesaba el área transversal A, implicando que la intensidad registrada dependía del orificio.

Otro factor era que la radiación que era emitida en la cavidad, no necesariamente pasaba a través del orificio,sino que era un tipo de emisión polar que dependía de un factor $\cos(\theta)$, siendo θ el ángulo entre el eje del orificio y un punto de emisión. Al integrar la forma de emisión se obtiene un factor de $1/4$. Por consiguiente la distribución de energía vista a la salida del horno correspondía a:

$$J(f,T) = \frac{A\mathrm{c}}{4}\rho(f,T) \tag{5}$$

Otro experimento realizado por el espectroscopista alemán Friedrich Paschen, con cuerpos a $1500K$ permitió realizar un barrido espectral de pequeñas longitudes de onda hasta el espectro del infrarrojo en longitudes de onda entre $1 - 4\mu\, m$; principalmente en las regiones de la máxima frecuencia de emisión. Con los valores experimentales obtenidos y la ley de Wien, se encontró buena correspondencia en el modelo y los datos experimentales en ésta zona.

En 1900 se realizaron dos nuevos experimentos. Lummer y Pringsheim aumentaron el rango del espectro de emisión hasta la longitud de onda de $18\mu\, m$. Posteriormente Rubens y Kurlbaum llevaron hasta longitudes de onda de $60\mu\, m$. Éstas regiones en el experimento, a diferencia de la zona de la versión de Paschen, estaban mucho más alejadas de la frecuencia de máxima emisión. Al contrastar los valores con el modelo de Wien, observaron que no había un ajuste correcto para éstas regiones.

La parte experimental demostró que todavía no había un análisis completo en los modelos teóricos. Sin embargo éste paso fue pionero en los conllevaría a la solución, al menos analítica del enigma del cuerpo negro. Para ésta época surgen dos prominentes figuras de la física en la búsqueda de la respuesta: John William Strutt (Baron Rayleigh) y Max Planck.

1.6 El comportamiento discreto de la materia en la equipartición de energía

La mayoría de los experimentalistas y teóricos habían centrado su enfoque en la forma de la radiación emitida. Rayleigh había centrado su idea en el comportamiento electromagnético en el interior de la cavidad, y para 1900 ya comenzado a plantear el comportamiento de la radiación ondulatoria y la interacción con las paredes internas.

1.6.1 La deducción electromagnética de una cavidad resonante de Rayleigh

La propuesta de Rayleigh partía de entender la radiación como un conjunto de ondas estacionarias encerradas en una cavidad cúbica de longitud a. Por consiguiente un oscilador electromagnético. La pregunta a resolver sería ¿Cuántas diferentes ondas estacionarias hay en la cavidad? La idea era similar al problema que él había resuelto 25 años antes, sobre las ondas sonoras en una cavidad cúbica.

El oscilador de Rayleigh consideró que la radiación electromagnética y sus componentes, se podrían describir como un oscilador clásico unidimensional. La cantidad y tipo de ondas posibles debían satisfacer

el criterio de los modos normales posibles en la cavidad y las frecuencias permitidas de oscilación. Ésta restricción se establece, respectivamente, como:

$$(k_x, k_y, k_z) = \frac{\pi}{a}(l, m, n) \therefore \frac{c}{2\pi}(k_x^2 + k_y^2 + k_z^2)^{1/2} \tag{6}$$

Lo que Rayleigh encontró fue una expresión para la densidad de estados de radiación en el horno; la cantidad de modos permisibles en la cavidad:

$$N(f)\mathrm{d}f = \frac{8a^3\pi\, f^2}{c^3}\mathrm{d}f \tag{7}$$

1.6.2 La deducción termodinámica y de la mecánica estadística de Planck

Planck por su parte, no tomó una aproximación tan directa como Rayleigh. En lugar de retomar el electromagnetismo, Plack desde hacía unos años trabajaba en un complejo y abstracto modelo conformado por la termodinámica, las leyes de la entropía, la mecánica estadística y parte del electromagnetismo.

Planck conocía la correspondencia de la ley de Wien para altas frecuencias por lo que intentó deducir ésta ley partiendo principalmente de la mecánica estadística y las leyes de Maxwell. Finalmente Planck encuentra un modelo analítico que tenía correspondencia en los experimentos de Rubens. Sin embargo la interpretación de Planck sobre el fenómeno le daría más peso y ayudará a plantear ideas del comportamiento.

La idea de que en las paredes de la cavidad habían osciladores y cada uno a diferentes frecuencias. La radiación que emitían era igual en frecuencia a la frecuencia de vibración. Al aplicar la restricción del equilibrio térmico en la cavidad, encuentra una forma general de la densidad de potencia espectral:

$$\rho(f,T)\mathrm{d}f = \bar{E}N(f)\mathrm{d}f \tag{8}$$

Ésta forma describe que la densidad volumétrica de energía depende de la forma y cantidad de oscilaciones en los resonadores en las paredes de la cavidad, $N(f)$, evaluada entre las frecuencias f y Δf. El segundo factor de dependencia es la energía promedio $\bar{E}$ en cada resonador. Planck demuestra con las restricciones termodinámicas que el número de oscilaciones posibles en la cavidad unitaria es:

$$N(f)\mathrm{d}f = \frac{8\pi\, f^2}{c^3}\mathrm{d}f \tag{9}$$

Tanto Rayleigh como Planck acuerdan en la forma en que el comportamiento ondulatorio debe comportarse. Aunque ambos desconocían que habían llegado al mismo resultado y por diferentes razonamientos. Al sustituir el número de modos, la densidad de energía en la cavidad quedaría de la forma:

$$\rho(f,T)\mathrm{d}f = \bar{E}\frac{8\pi\, f^2}{c^3}\mathrm{d}f \tag{10}$$

Lo que quedaría pendiente sería la distribución de energía en los resonadores de la cavidad. Como se verá a continuación sobre la energía hay dos percepciones: Energía continua y energía discreta.

1.6.3 La catástrofe ultravioleta y la ley de Rayleigh-Jeans

Rayleigh y el astrónomo-físico inglés James Jeans, determinaron el término de la energía utilizando la distribución de Maxwell-Boltzaman. El modelo describe cuál es la probabilidad P de encontrar un sistema individual en una energía E, cuando éste hace parte de un conjunto de sistemas a una temperatura T. Planteando:

$$P(E) = P_0\mathrm{e}^{-(E-E_0)/k_BT} \tag{11}$$

donde $ P_{0}$ es la probabilidad de encontrar el sistema en la energía inicial E_0. Por consiguiente la energía promedio normalizada en cada sistema sería:

$$\bar{E} = \frac{\sum E P(E)}{\sum P(E)} \tag{12}$$

Con éste planteamiento Rayleigh y Jeans consideraron una forma clásica en que cualquier resonador podría tener ésta energía a cualquier frecuencia. Por consiguiente se planteó una forma continua de la energía:

$$\bar{E} = \frac{\int_0^\infty E e^{-E/k_B T} \mathrm{d}E}{\int_0^\infty e^{-E/k_B T} \mathrm{d}E} = k_B T \tag{13}$$

Con ésta consideración, encontraron que ésta energía en cada oscilador era dependiente de la temperatura e independiente de la frecuencia. Por lo que cada resonador tendría la misma energía independiente de las oscilaciones. Con éste término, la densidad de energía de la cavidad sería:

$$\rho(f,T)\mathrm{d}f = k_B T \frac{8\pi f^2}{c^3} \mathrm{d}f \tag{14}$$

Sin embargo al evaluar éste modelo para altas frecuencias, en cuerpos a más de $1000K$, el modelo diverge a medida que la longitud de onda se aproxima a 0, prediciendo energía infinita en la región del ultravioleta. Posteriormente Paul Ehrenfest se refirió a éste como la catástrofe-ultravioleta. El modelo para bajas frecuencias tiene correspondencia.

1.6.4 La energía discreta del oscilador y la "suerte" de Planck

Planck por su parte, seguía trabajando en las consideraciones de la energía de los resonadores, apoyado de la mecánica estadística. Su idea principal era en mantener los estados discretos. Planck plantea la energía de manera discreta:

$$\bar{E} = \frac{\sum_{n=0}^{\infty} (nhf)(P_0 e^{-nhf/k_B T})}{\sum_{n=0}^{\infty} P_0 e^{-nhf/k_B T}} = \frac{hf(1-e^{-hf/k_B T})(e^{-hf/k_B T})}{(1-e^{-hf/k_B T})^2} = \frac{(hf)(e^{-hf/k_B T})}{(1-e^{-hf/k_B T})} = \frac{hf}{e^{-hf/k_B T}-1} \tag{15}$$

El modelo considera restricciones no unicamente en la frecuencia sino en una cantidad de energía. El sistema no podría tener más energía que hf. Cada elemento del sistema debería tener una energía asociada que depende; a diferencia del modelo de Rayleigh-Jeans, de la frecuencia y la temperatura. Planck con ésta forma de la energía, interpola la información experimental de Rubens y Kurlbaum del *Restrahlen* (experimento de los rayos residuales) junto con el modelo de las altas frecuencias de Wien.

Un domingo de octubre de 1900, Planck escribe en una carta dirigida Rubens en la que contenía la formula que se conoce como la ley de emisión de Planck:

$$\rho(f,T)\mathrm{d}f = \frac{8\pi f^2}{c^3} \frac{hf}{e^{-hf/k_B T}-1} \mathrm{d}f \tag{16}$$

El modelo, matemáticamente, resuelve la catástrofe ultravioleta al atenuar el término f^2, para las altas frecuencias. Rubens verifica el total acuerdo con los datos experimentales.

En 1918 el comité del premio Nobel decide que Max Planck es merecedor del premio nobel por "contribuciones en el avance de la física y el descubrimiento de la energía cuantizada". Planck en 1920 en su discurso menciona su insatisfacción. Si bien el modelo ajusta el experimento perfectamente, no fue más que una "conjetura con suerte". La interpretación física sobre el cuerpo negro, Planck aún la desconocía.

1.7 El gas de radiación de Einstein.

En 1905, Einstein promueve una de las revoluciones más grandes de la física. En éste año, Einstein nuevamente retoma el estudio sobre el cuerpo negro, tras casi 5 años de silencio en el tema. Tomando los resultados de Rayleigh para regiones de bajas frecuencias y los resultados de Planck para las altas frecuencias, se percata de una analogía con la densidad de energía en la distribución clásica de los gases:

$$f(E) = \left[4\pi \left(\frac{m}{2\pi\, k_b T}\right)^{3/2} v^2\right] E\mathrm{e}^{-E/k_B T} \tag{17}$$

donde m es la masa de las partículas y v es la velocidad. En el caso de las altas frecuencias de la ley de Planck, ésta tiene la forma:

$$\rho(f,T) = \frac{8\pi\, f^2 hf}{c^3}\mathrm{e}^{-hf/k_B T} \tag{18}$$

Einstein propone que en las altas frecuencias la radiación se podría imaginar como un gas de partículas independientes, cuya energía sería $E = hf$. Reescribiendo la expresión anterior:

$$\rho(f,T) = \left[\frac{8\pi\, f^2}{c^3}\right] E\mathrm{e}^{-hf/k_B T} \tag{19}$$

Por consiguiente la analogía entre las expresiones es cercana. Para éste gas la frecuencia es proporcional al número de onda y al momentum. Para los átomos la velocidad es proporcional al momentum. Por tanto ambas expresiones se encuentran en el espacio del momentum. Ésta observación conllevó a Einstein a afirmar que "la radiación en el interior de la cavidad de por sí está cuantizada". Por consiguiente la cuantización en la energía no era algo exclusivo de los radiadores de Planck, sino también de la radiación; paquetes de radiación.

2 Métodos numéricos en el problema de cuerpo negro

2.1 Sobre el código

El siguiente código, es un ejercicio sobre métodos numéricos para resolver numéricamente algunos de los planteamientos anteriormente vistos. En éste notebook se utilizan funciones nativas de python y otras funciones importadas del repositorio remoto, por lo que para la ejecución de nuevos parámetros o modificación, es necesario abrir éste archivo en una consola interactiva de Jupyter.

La sección computacional se deriva en cinco subsecciones:

- Ley de Planck de la radiación espectral de cuerpo negro
- Determinación por gradiente de los máximos de emisividad de la distribución
- Regresión linear en la ley de desplazamiento de Wien
- Interpolación y comparación de la ley de Rayleigh-Jeans
- Ley de Stefan-Boltzmann

```
In [1]: # Import Modules
        import os
        import matplotlib.pyplot as plt
        import matplotlib.patches as mpatches
        import time
        import numpy as np
        import pandas as pd
        from math import pi

        %matplotlib inline
```

2.2 Ley de Planck de la radiación espectral de cuerpo negro

La ley de Planck permite determinar la densidad de energía emitida por un un cuerpo negro, acorde a la temperatura.

$$\rho(\lambda, T) = \frac{2\,hc^2}{\lambda^5 \left[\exp\left(\frac{hc}{\lambda\, k_b T}\right) - 1\right]} \tag{20}$$

Donde λ corresponde a la longitud de onda, T corresponde a la temperatura del cuerpo, h corresponde a la constante de Planck, c corresponde a la velocidad de la luz y k_B corresponde a la constante de Boltzmann cuyo valor es igual a 1.3805·10-23 J/K.

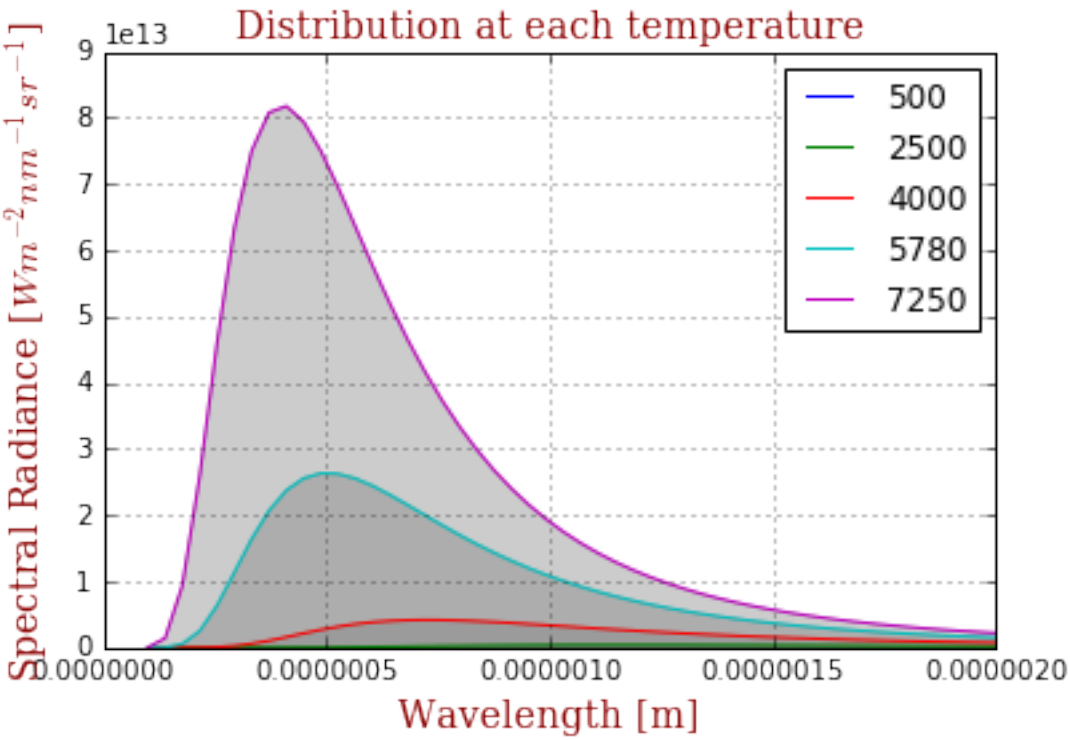
El siguiente código ilustrara en una gráfica la relación entre la intensidad de radiación y la longitud de onda, aplicando la ley de Planck. Se analizara la presente gráfica partiendo de la variación de la temperatura en el cuerpo negro, esto con el fin de comprobar computacionalmente la relación inversa que se presenta entre la temperatura y la longitud de onda, por lo que a mayores temperatura menor será el tamaño de la longitud de onda, y mayor la intensidad de radiación.

In [2]:
```
# %load https://raw.githubusercontent.com/fisicatyc/numericos-interactivo/master/project_2014_n
"""
    Description: function comptraprule is part of the Composite Integration
    Techniques and can be thought as an upgrated version of Trapezoidal rule
    integration technique that aproximates a defined integal evaluated in [a,b]
    with better presicion in larger intervals. It's truncation error is of order
    h^2 and defined by a second order derivative. Similar to compmidtrule but
    request more operations Presicion is similar to Composite midpoint rule.
    Difference from the other methods is the need of only one integration interval
    therefore the number of subintervals can be even or odd.

    Inputs: lowerlimit - float - defines first limit of integration.
            upperlimit - float - defines last limit of integration.
            redc - integer - integer that reduces space grid in domain.
            function - function type object - evaluates f(x) at x.

    Outputs: integ - float - defined integral aproximation

    Example line: integ = comptraprule(-2, 2, 40, (lambda x: 3*x**4));

    Dependencies: None.

    Version: 1.2 for Python 3.4

    Definitions were taken from:
        Richard L. Burden, J. Douglas Faires. "Numerical Analysis" 9th ed.
        "Chapter 4 - Numerical Differentiation and Integration".
        Cengage Learning. 2010. pp: 153 - 156.

    Author: J.J. Cadavid - SFTC - EAFIT University.
    Contact: jcadav22@eafit.edu.co

    Date: 28/12/2014.
"""

def comptraprule (lowerlimit, upperlimit, redc, function):

#  Input error verification
    if redc % 2 != 0:
        raise Exception('Reduced inverval - redc- must be an even integer');

    test = lambda: None;
```

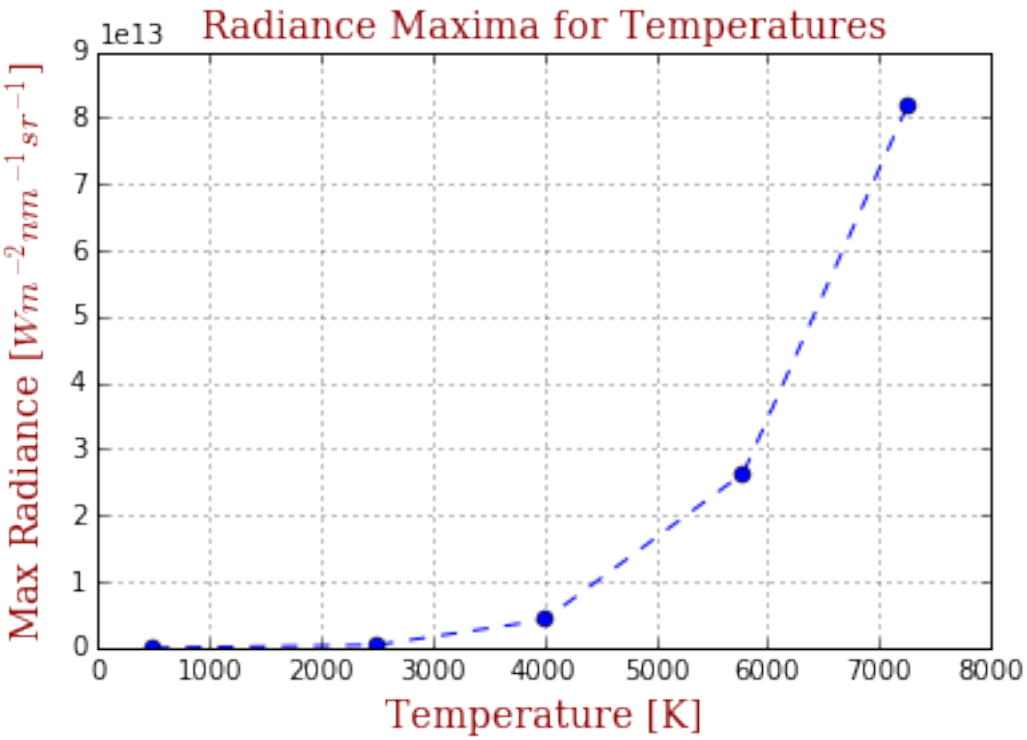
```
        if isinstance(function,type(test)) == 0:
            raise Exception("function must be lambda object-type");

    # Zero initialing
        sumf = 0.0;

    # Space grid size
        h = (upperlimit - lowerlimit)/redc;

    # Main loop - Composite trapezoidal rule approximation .
        for k in range(1, redc):
            sumf = sumf + 2*function(lowerlimit + k*h);

        integ = 0.5*h*(sumf + function(upperlimit) + function(lowerlimit));

        return(integ);
```

In [3]:
```
# Stefan-Boltzmann Constant
SIGMA = 5.670e-8;

# Planck parameters
h = 6.62607004e-34;
c = 3.0*10**8;
Kb = 1.38064852e-23;
TemperatureVect = [500,2500,4000,5780,7250];
VectSize = len(TemperatureVect);

# Data initialization
dataSize = 50;
IntegSamples = 100 # Defines the number of samples
RadianceMat = [[0 for x in range(dataSize)] for x in range(VectSize)];
MaxDataMat = [[0 for x in range(3)] for x in range(VectSize)];
waveLengthMin = 100e-9;
waveLengthMax = 2000e-9;
waveLengthVect = np.linspace(waveLengthMin,waveLengthMax,dataSize);

# Planck's Law function
for n in range(0,VectSize):
    B = lambda waveLength: (2*h*c**2)/waveLength**5*(1/(np.exp((h*c)/(waveLength*Kb*Temperature
    RadianceMat[n][:] = B(waveLengthVect);
    MaxDataMat[n][2]= comptraprule(waveLengthMin, waveLengthMax, IntegSamples, B);
```

In [4]:
```
# Plot parameters
font = {'family' : 'serif','color'  : 'darkred', 'weight' : 'normal', 'size'   : 14};
for n in range(0,VectSize):
    plt.plot(waveLengthVect,RadianceMat[n][:], label = TemperatureVect[n],);
    plt.fill_between(waveLengthVect,RadianceMat[n][:], alpha=0.05*n, facecolor='k');
plt.axis('auto');
plt.title('Distribution at each temperature', fontdict=font);
plt.xlabel('Wavelength [m]', fontdict=font);
plt.ylabel('Spectral Radiance [$W m^{-2}nm^{-1}sr^{-1}$]', fontdict=font);
plt.grid();
plt.legend();
plt.show();
```




```
In [5]: # Data analysis
        for n in range(0,VectSize):
            CurrentData = RadianceMat[n][:];
            MaxDataMat[n][0] = waveLengthVect[CurrentData.index(max(CurrentData))];
            MaxDataMat[n][1] = max(CurrentData);

        MaxDataArry = np.asarray(MaxDataMat)
        TempArry = np.asarray(TemperatureVect);

        plt.plot(TemperatureVect,MaxDataArry[:,1], marker='o', linestyle='--', color='b');
        plt.axis('auto');
        plt.title('Radiance Maxima for Temperatures', fontdict=font);
        plt.xlabel('Temperature [K]', fontdict=font);
        plt.ylabel('Max Radiance [$W m^{-2}nm^{-1}sr^{-1}$]', fontdict=font);
        plt.grid();
        plt.show();

        Frame = pd.DataFrame(MaxDataMat,index=TemperatureVect, columns=['Max Radiance WL [m]','Max Radi
        Frame
```



Out[5]:

	Max Radiance WL [m]	Max Radiance	Total Energy
500	2.000000e-06	2.082200e+06	0.358755
2500	1.146939e-06	3.990157e+11	445878.365127
4000	7.204082e-07	4.184867e+12	3949892.500594
5780	4.877551e-07	2.631612e+13	18902104.924807
7250	4.102041e-07	8.174296e+13	48105063.417381

2.3 Determinación por gradiente de los máximos de emisividad de la distribución

Acorde la gráfica radiancia máxima, se puede observar una relación entre los máximos de cada temperatura. Ésta gráfica muestra el comportamiento que la ley de desplazamiento Wien para diferentes temperaturas. Aunque ésta ley solo predice el máximo para cada temperatura, al tener los diferentes máximos se puede ver un comportamiento exponencial en la densidad de energía asociada. Sin embargo, la forma de encotrarlos fue direcamten comparando el máximo valor en el arreglo de datos.

Éstos máximos para cada distribución pueden obtenerse también al entender el comportamiento de los cambios en la función, principalmente en los valores de las pendientes. Si se observa la gáfica de las distribuciones de la densidad de energía, desde las longitudes de onda mayores hasta las pequeñas, la función decrece (menor energía para longitudes de ondas grandes). Por consiguiente ésta región tiene pendiente negativa. Sin embargo, si se observa en la región de las longitudes menores hacía la parte visible del espectro (La transición del ultravioleta al azul), se observa que la pendiente aumenta. (Gran parte de la densidad de energía del espectro está en la región visisble).

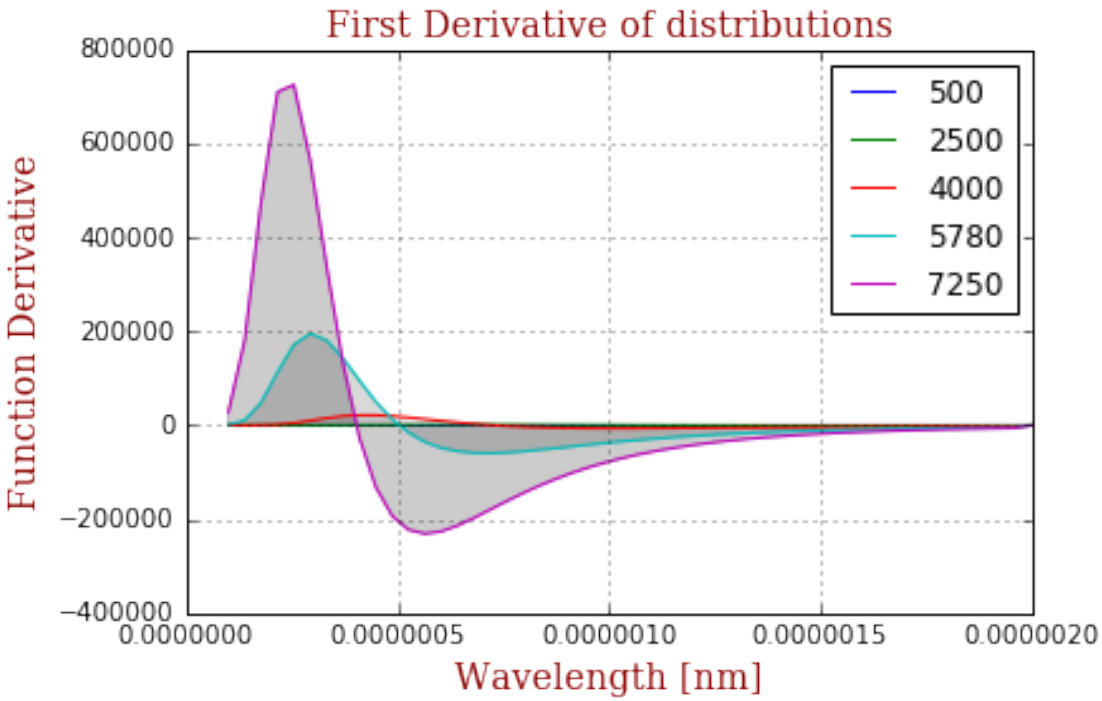
Los cambios en el signo de la pendiente indican que hay un cruce por cero, o por consiguiente que un máximo (o un mínimo) local está presente. El algoritmo propuesto evalúa localmente las derivadas de toda la función utilizando diferencias finitas de primer orden centrada. El código es el siguiente:

In [6]:
```
# %load https://raw.githubusercontent.com/fisicatyc/numericos-interactivo/master/project_2014_n
#!/usr/bin/python
"""
    Description: class centraldiff defines a set of numerical methods based on
    finite differences of central step. Capable of aproximating derivatives at
    a given point inside an interval. Central step does not evaluate derivatives
    at endpoints. Precision is highly dependant on interval size, bigger intervals
    lead to highter truncation errors. Minor error types are of h^2 order.
    Analitical errors can be calculated using fourth derivatives. Highter
    derivative orders can lead to high truncation errors, so results are likely
    to differ a lot from analytical results, especially with interpolated data.

    FCDA or First Centered Difference Aproximation, calculates the first
    derivative in the interval. Dependencies: testlambda, getarray.

    Method inputs: n - integer - Defines number of n+1 elements in list.
                   lowerlimit - float - first term of interval.
                   upperlimit - float - last term of interval.
                   function - function type object - evaluates f(x) at x.

    Method output: diff_array - list - list of derivated values in (a,b).

    call sequence example: centraldiff.FCDA(4, 0, 2, (lambda x: 4*sin(x)**2));

    Dependencies: None.

    Version: 1.2 for Python 3.4

    Definitions are taken from:
        Jaan Kiusalaasr. "Numerical Methods in Engineering With Python 3".
        3th ed. "Chapter 5 - Numerical Differentiation".
        Cambridge University Press. 2013. PP. 183 - 185.

    Author: J.J. Cadavid - SFTC - EAFIT University.
    Contact: jcadav22@eafit.edu.co

    Date: 28/12/2014.
"""

def FCDA(n,lowerlimit, upperlimit, function, x, flag = False):
    test = lambda: None;
    if isinstance(function,type(test)) == 0:
        raise Exception("function must be lambda object-type");

    h = upperlimit-lowerlimit;

    if flag:
        hc = h/(n-1);
        arrayStart = lowerlimit;
        x_array = [arrayStart + hc*x for x in range(n)];
        diff_array = [0]*n;

        for i in range(n):
            diff_array[i] = (function(x_array[i] + h) - \
```

```
                                function(x_array[i] - h))/2*h;
            else:
                diff_array = (function(x + h) - \
                                function(x - h))/2*h;
            return(diff_array);
```

In [7]:
```
DerivMat = [[0 for x in range(dataSize)] for x in range(VectSize)];
# First derivative for each Planck's Law at different temperatures
for n in range(0,VectSize):
    DB = lambda waveLength: (2*h*c**2)/waveLength**5*(1/(np.exp((h*c)/(waveLength*Kb*Temperatur
    for b in range(0,dataSize-1):
        DerivMat[n][b] = FCDA(n, waveLengthVect[b], waveLengthVect[b+1],DB,waveLengthVect[b]);
```

In [8]:
```
for n in range(0,VectSize):
    plt.plot(waveLengthVect,DerivMat[n][:], label = TemperatureVect[n],);
    plt.fill_between(waveLengthVect,DerivMat[n][:], alpha=0.05*n, facecolor='k');
plt.axis('auto');
plt.title('First Derivative of distributions', fontdict=font);
plt.xlabel('Wavelength [nm]', fontdict=font);
plt.ylabel('Function Derivative', fontdict=font);
plt.grid();
plt.legend();
plt.show();
```



Tras encontrar las derivadas de toda la función, se puede observar una región de interés. Con una lectura de derecha a izquierda de la gráfica se observa una pendiente que se vuelve más negativa. Ésto indica que el cambio negativo y es más abrupoto mayor para las longitudes de onda que comienzan a alejarse del espectro visible, indicando que tienen menor densidad asociada.

Al llegar a la zona del espectro visible, la cantidad de energía aumenta rápidamente; pendiente positiva "empinada". Es en ésta región de aumento aproximadamente constante de energía donde ocurre el cambio

de signo, y gráficamente se ve como el cruce por cero del eje vertical. Es en éste valor donde se encuentra el máximo de emisividad.

El siguiente paso está en encotrar los indices del arreglo que tengan la pendiente negativa más pronunciada y la pendiente positiva más pronunciada. Ésta aproximación tiene en consideración que la función tiene un único máximo que es global y no hay mínimos. De ésta forma se garantiza que el valor entre éstas pendientes es el máximo que se busca.

La comparación de valores busca un primer máximo y lo compara con el valor vecino del arreglo. En el momento de que encuentra el valor más cercano a cero, se considera como el máximo de la función, devolviendo las indices de las posiciones que corrspondend a los máximos de las distribuciones. El algoritmo compara todos los valores indpependientemente de haber en contrado el máximo.

Los resultados son presentados en la tabla al final de la celda.

In [9]:
```
# Find Derivative Zeros - all zeros are assumed to be between global maximum and minimum
MinDataMat = [[0 for x in range(2)] for x in range(VectSize)];

# Loop to find individual maximum at each temperature
CurrentMax = 0;
for n in range(0,VectSize):
    CurrentData = DerivMat[n][:];
    MaxIndex = CurrentData.index(max(CurrentData));
    MinIndex = CurrentData.index(min(CurrentData));
    TGTIntval = DerivMat[n][MaxIndex:MinIndex]
    IntvalMin = min(abs(i) for i in TGTIntval)
    try:
        a = CurrentData.index(IntvalMin);

    except:
        a = CurrentData.index(-IntvalMin);
    MinDataMat[n][0] = waveLengthVect[a];
    MinDataMat[n][1] = RadianceMat[n][a]
    CurrentData = RadianceMat[n][a];


    if CurrentMax < CurrentData:
        MaxRadiance =  RadianceMat[n][a];
        MaxWlength = waveLengthVect[a];
        Temperature = TemperatureVect[n];
    else:
        CurrentMax = RadianceMat[n][a];

print("The Black Body with Temperature: %s, has the Maximum radiance: %s which is reached at Wa

FrameB = pd.DataFrame(MinDataMat,index=TemperatureVect, columns=['Max Radiance WL [m]','Max Radi
FrameB
```

```
The Black Body with Temperature: 7250, has the Maximum radiance: 8.17429629438e+13 which is reached at W
```

Out[9]:

	Max Radiance WL [m]	Max Radiance
500	1.961224e-06	1.727468e+06
2500	1.146939e-06	3.990157e+11
4000	7.204082e-07	4.184867e+12
5780	4.877551e-07	2.631612e+13
7250	4.102041e-07	8.174296e+13

2.4 Regresión linear en la ley de desplazamiento de Wien

La ley de desplazamiento de Wien permite calcular la posición del espectro máximo en un cuerpo negro. Una ley que refleja de manera operacional la relación inversamente proporcional existente entre la temperatura y la longitud de onda, del cual confirma que ha mayor temperatura menor la longitud de onda que emite. Matemáticamente la ley se expresa de la siguiente manera.

$$\lambda_{max} = \frac{0,0028976}{T} \tag{21}$$

donde T corresponde a la temperatura del cuerpo y λ corresponde a la longitud de onda en el pico máxima del espectro de emisión.

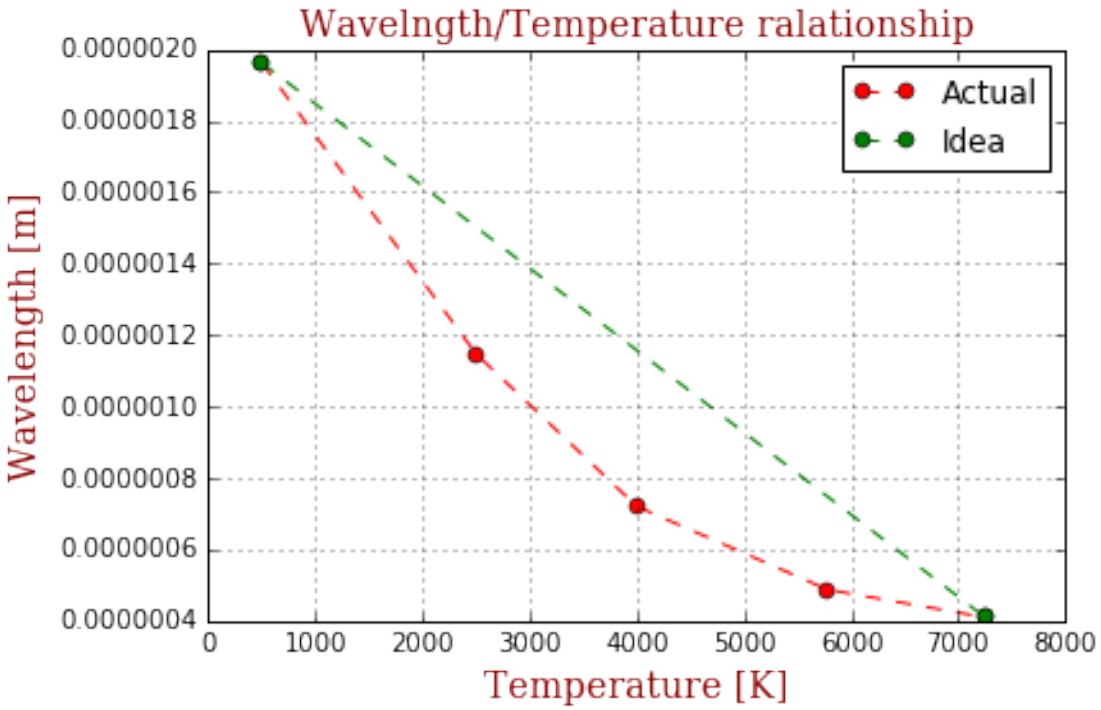
Con los datos computados de los valores Máximos de la función se desea realizar una regresión lineal que permita determinar una función que aproxime el modelo. Sin embargo ésta aproximación no tiene validez cuando la longitud de onda se vuelve pequeña, principalmente por el comportamiento asintótico de la función, cuya indeterminación ocurre en $\lambda = 0$.

En la figura siguiente, se muestra el comportamiento no lineal de los máximos (linea roja), pero se desea aproximar con una función de la forma:

$$f(x) = ax + b \tag{22}$$

Ésta recta del gráfico, únicamente se traza desde el primer y último dato del arreglo de máximos. Por consiguiente entre más grande sea la diferencia entre la primera y última temperatura, menor sera el comportamiento lineal.

```
In [10]: MinDataArry = np.asarray(MinDataMat);
         plt.plot(TemperatureVect,MinDataArry[:,0], marker='o', linestyle='--', color='r',label='Actual
         plt.plot([TemperatureVect[0],TemperatureVect[VectSize-1]],[MinDataArry[0,0],MinDataArry[VectSi
         plt.axis('auto');
         plt.title('Wavelngth/Temperature ralationship', fontdict=font);
         plt.xlabel('Temperature [K]', fontdict=font);
         plt.ylabel('Wavelength [m]', fontdict=font);
         plt.grid();
         plt.legend();
         plt.show();
```



Para el ajuste de la curva, se reapiza una regresión por mínimos cuadrádos de un polinomio de grado m. En nuestro caso, un polinomio de grado 1. También es posible utilizar otro tipo de funciones para el ajuste. De ésta forma se realiza una minimización, cuya representación forma es:

$$S(a_0, a_1, a_2 ... a_m) = \sum_{i=1}^{n} [y_i - f(x_i)]^2 \therefore \frac{\partial S}{\partial a_i} = 0 \tag{23}$$

donde $f(x)$ es la función del ajuste y n es la cantidad total de información para el ajuste. De tal forma, se minimiza sobre los m coeficientes del polinomio, tal que produzca el menor residuo (discrepancia) entre la información que se está ajustando. El problema del ajuste linear se plantea como:

$$S(a, b) = \sum_{i=1}^{n} [y_i - f(x_i)]^2 = \sum_{i=1}^{n} (y_i - a - bx_i)^2 \tag{24}$$

La solución a los parámetros que satisfacen las derivadas son:

$$a = \frac{\bar{y} \sum x_i^2 - \bar{x} \sum x_i y_i}{\sum x_i^2 - n\bar{x}^2} \qquad b = \frac{\sum x_i y_i - \bar{x} \sum y_i}{\sum x_i^2 - n\bar{x}^2} \tag{25}$$

Sin embargo ésta definición de los parámetros es más sensible a errores de redondeo. Por tanto, se pueden calcular como:

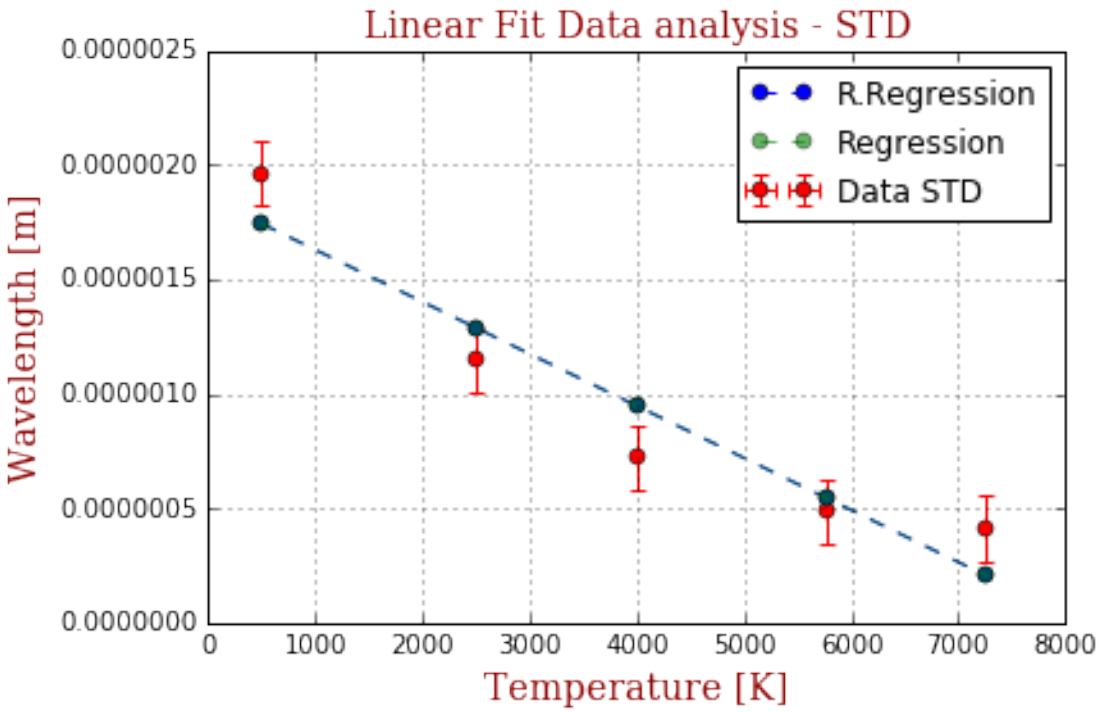
$$b = \frac{\sum y_i (x_i - \bar{x})}{\sum x_i (x_i - \bar{x})} \qquad a = \bar{y} - \bar{x} b \tag{26}$$

Éstas definiciones son aplicadas en la siguiente celda.

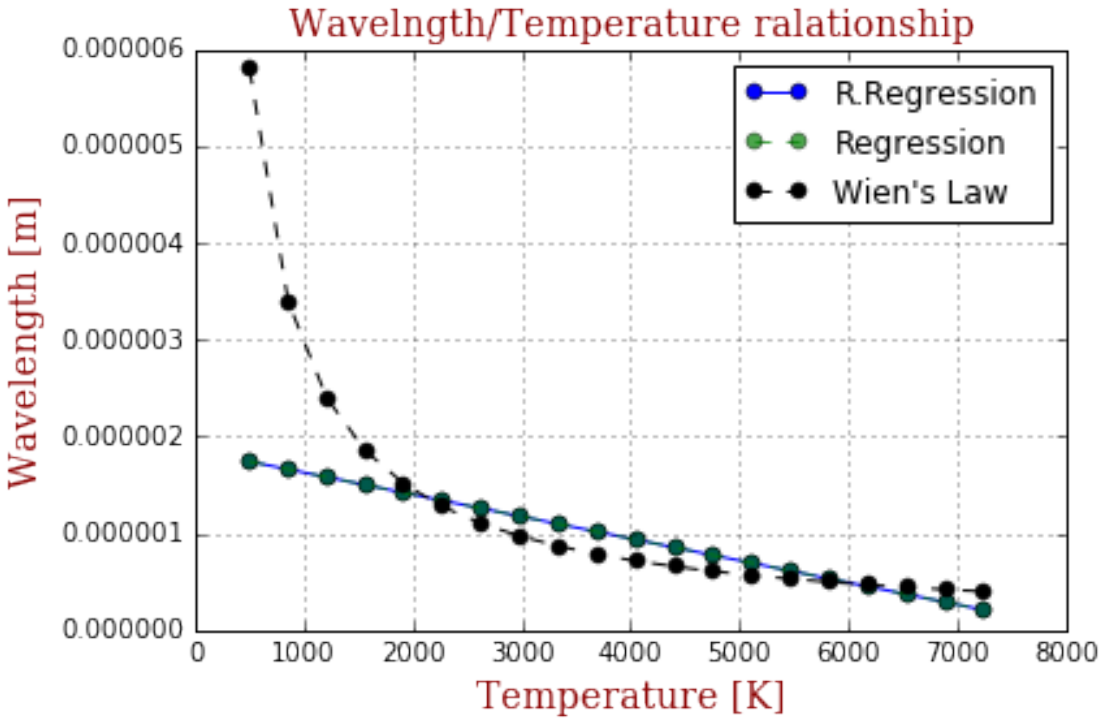
```
In [11]: # Defining the parameters of fitting curve - A linear polynomial
         Tmean = sum(TempArry)/VectSize;
         WLmean = sum(MinDataArry[:,0])/VectSize;
```

```
SqrTemp = sum(TempArry**2);

SigmaX = np.sqrt(1/VectSize*sum(TempArry - Tmean)**2);
SigmaY = np.sqrt(1/VectSize*sum(MinDataArry[:,0] - WLmean)**2);

# Round-off susceptible fit
ar = (WLmean*SqrTemp - Tmean* sum(TempArry*MinDataArry[:,0]))/(SqrTemp-VectSize*Tmean**2);
br = (sum(TempArry*MinDataArry[:,0]) - Tmean*sum(MinDataArry[:,0]) )/(SqrTemp-VectSize*Tmean**
Yr = lambda Temp: ar + br*Temp;

# Less Round-off susceptible fit
b = sum(MinDataArry[:,0]*(TempArry - Tmean))/sum(TempArry*(TempArry - Tmean));
a = WLmean - Tmean*b;
Y = lambda Temp: a + b*Temp;

# Wien's Displacement Law:
WLw = lambda Temp: 2.89776829e-3/Temp; # [m]K/K
```

In [12]:
```
# Plot regression results
DataScale = 10e14;
plt.errorbar(TempArry,MinDataArry[:,0], xerr=SigmaX*DataScale, yerr=SigmaY*DataScale,fmt='o',
plt.plot(TempArry,Yr(TempArry), marker='o', linestyle='--', color='b',label='R.Regression');
plt.plot(TempArry,Y(TempArry), marker='o', linestyle='--', color='g',label='Regression',alpha=

plt.title('Linear Fit Data analysis - STD', fontdict=font);
plt.xlabel('Temperature [K]', fontdict=font);
plt.ylabel('Wavelength [m]', fontdict=font);
plt.grid();
plt.legend();
plt.show();
```



In [13]:
```
# Temperature array
TempData = np.linspace(min(TempArry),max(TempArry),20);

# Plot Data
plt.plot(TempData,Yr(TempData), marker='o', linestyle='-', color='b',label='R.Regression');
plt.plot(TempData,Y(TempData), marker='o', linestyle='--', color='g',label='Regression',alpha=
plt.plot(TempData,WLw(TempData), marker='o', linestyle='--', color='k',label="Wien's Law");
plt.axis('auto');
plt.title('Wavelngth/Temperature ralationship', fontdict=font);
plt.xlabel('Temperature [K]', fontdict=font);
plt.ylabel('Wavelength [m]', fontdict=font);
plt.grid();
plt.legend();
plt.show();
```



Como se observa en la figura anterior, el polinomio como la ley de Wien, difieren particularmente para la región en que el comportamiento asintótico es más evidenciado. Para las regiones más distantes de la función se observa que hay cercanía en las formas de la función y la regresión.

En éste caso también se observa que las dos formas de regresión coinciden, implicando que no hay errores por redondeo en los datos. La regresión indica que la ley de Desplazamiento crece lentamente para altas temperaturas por lo que la forma de la función es aproximadamente recta, y la función de ajuste es válida para éstos valores.

2.5 Interpolación y comparación de la ley de Rayleigh-Jeans

La ley de Raleigh-jean consiste en describir la radiación espectral de un cuerpo negro a una temperatura dada y variando las longitudes de ondas. A continuación se expresa matemáticamente la presente ley.

$$B = \frac{2ckT}{\lambda^4} \tag{27}$$

El código en la siguiente sección calcula la interpolación de Newton y compara directamente la relación con la ley de Rayleigh-Jeans.

In [14]:
```
## module newtonPoly - From Numerical Methods in Engineering with Python 3 by Jaan Kiusalass -
"""
p = evalPoly(a,xData,x).
Evaluates Newton's polynomial p at x. The coefficient
vector {a} can be computed by the function 'coeffts.

a = coeffts(xData,yData).
Computes the coefficients of Newton's polynomial.
"""
def evalPoly(a,xData,x):
    n = len(xData) - 1 # Degree of polynomial
    p = a[n]
    for k in range(1,n+1):
        p = a[n-k] + (x -xData[n-k])*p
    return p

def coeffts(xData,yData):
    m = len(xData) # Number of data points
    a = yData.copy()
    for k in range(1,m):
        a[k:m] = (a[k:m] - a[k-1])/(xData[k:m] - xData[k-1])
    return a
```

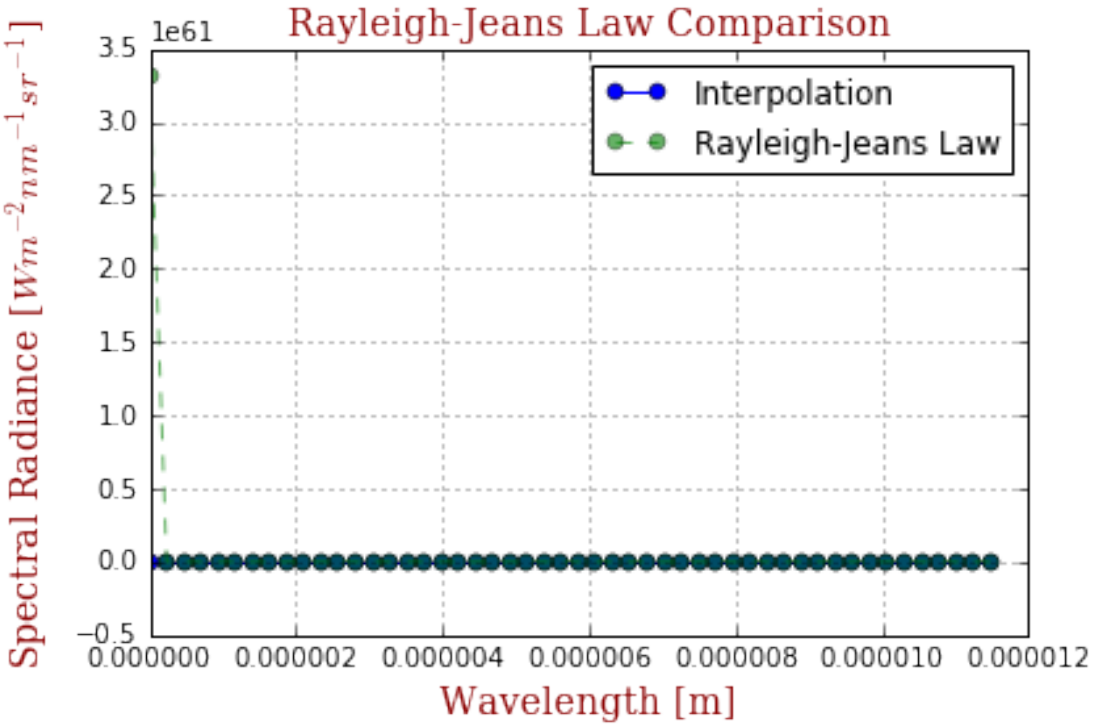
In [15]:
```
# Fixed temperature
FxdTemp = 4000; #[K]
WLSet = [1.146939e-5,4.146939e-06,6.204082e-07,1e-18];
SetSize = len(WLSet);
FxdData = [[0 for x in range(2)] for x in range(SetSize)];

BF = lambda waveLength: (2*h*c**2)/waveLength**5*(1/(np.exp((h*c)/(waveLength*Kb*FxdTemp))-1))

for n in range(SetSize):
    FxdData[n][0] = WLSet[n];
    FxdData[n][1] = BF(WLSet[n]);

FxdDataArry = np.asarray(FxdData);
```

In [16]:
```
# Compute interpolant coefficients and polynomial
Coeff = coeffts(FxdDataArry[:,0],FxdDataArry[:,1]);
PolyData = [0]*dataSize;
print("Polynomial coefficients: %s" %(Coeff));

# Evaluate set with polynomial
WLSet = np.linspace(WLSet[0],WLSet[SetSize-1],dataSize)
for n in range(dataSize):
```

```
            PolyData[n] = evalPoly(Coeff,FxdDataArry[:,0],WLSet[n]);

        # Rayleigh-Jeans Law
        BRJ = lambda waveLength: (2*c*Kb*FxdTemp)/waveLength**4;
```

```
Polynomial coefficients: [  1.63006572e+09  -9.38689237e+15   1.00112532e+23   1.65069346e+29]
```

```
In [17]: # Plot Data
         plt.plot(WLSet,PolyData, marker='o', linestyle='-', color='b',label='Interpolation');
         plt.plot(WLSet,BRJ(WLSet), marker='o', linestyle='--', color='g',label='Rayleigh-Jeans Law',al
         plt.axis('auto');
         plt.title('Rayleigh-Jeans Law Comparison', fontdict=font);
         plt.xlabel('Wavelength [m]', fontdict=font);
         plt.ylabel('Spectral Radiance [$W m^{-2}nm^{-1}sr^{-1}$]', fontdict=font);
         plt.grid();
         plt.legend();
         plt.show();
```



2.6 Ley de Stefan-Boltzmann

Como se observó en la la ley de Planck una cavidad de cuerpo negro puede emitir en todo el espectro. Un de las propieades que se puede analizar es la Potencia de Emisividad de la cavidad, definiendo la cantidad de energía por unidad de área se emite. En el caso de tener una cavidad con un ángulo sólido de emisión hemisférico, se puede establecer la relación:

$$E_{total} = 2\pi\, hc^2 \int_0^{\infty} \frac{\mathrm{d}\lambda}{\lambda^5 \left[\exp\left(\frac{hc}{\lambda\, k_b T}\right) - 1\right]} d\nu = \sigma T^4 \tag{28}$$

2.6.1 Solución numérica de la constante de Stefan-Boltzmann

Para resolver numéricamente ésta expresión es necesario considerar que se trata de una integral impropia, puesto que sus limite superior de integración no está definido y posee una singularidad en el limite inferior. Por éste motivo las técnicas como el trapecio o las reglas de Newton-Cotes no pueden ser utilizadas, principalmente porque dependen de los limites para definir el dominio de integración.

Otra estrategia está en la integración de punto medio de Gauss-Legendre, sin embargo éste método no arroja resultas acertados sino se utiliza altos costos computacionales. Para plantear una solución a éste planteamiento, primero se realizará un cambio de variable en los argumentos de la función, para obtener un problema análogo:

$$u = \frac{hc}{\lambda\, k_B T} \longrightarrow \mathrm{d}\lambda = \frac{-hc}{\lambda^2\, k_B T} \therefore \tag{29}$$

$$E = \frac{2\pi\,(k_B T)^4}{c^2 h^3} \int_0^\infty \frac{u^3}{\exp(u) - 1} \mathrm{d}u = \sigma T^4 \tag{30}$$

El segundo paso sugiere cambiar el dominio de integración infinito a un dominio finito, aplicando un segundo cambio de variable, tal que:

$$t = u^{-1}, dt = -u^{-2} \therefore du = -u^{-2}, dt = -t^{-t} \tag{31}$$

de ésta manera se puede aplicar:

$$\int_a^\infty f(u)du = \int_{1/a}^0 -t^{-2} f(t^{-1})dt = \int_0^{1/a} t^{-2} f(t^{-1})dt \tag{32}$$

Sin embargo, nótese que en éste caso, el parámetro a equivale a cero, por lo que definiría nuevamente un limite indeterminado. Por éste motivo, otra aproximación que se puede realizar sería evadir la singularidad, desplazando el dominio de integración tal que $a = 0.1$. De ésta forma se puede establecer:

$$E = \frac{2\pi\,(k_B T)^4}{c^2 h^3} \int_{0.1}^{10} \frac{t^{-5}}{\exp(t^{-1}) - 1} \mathrm{d}u \tag{33}$$

El integrando planteado, puede ser evaluado numéricamente porlos convecionales métodos de intervalo cerrado o el método de Gauss-Lagendre. Para éste caso se utilizará éste último y el trapecio compuesto. Para el caso del punto medio, se puede establecer que:

$$\int_a^b f(x)\mathrm{d}x = \sum_{i=1}^{n} c_i f(r_i) \tag{34}$$

donde c_i y r_i son el i-ésimo coeficiente de interpolación y raíz del i-ésimo polinomio de Legendre. En el código siguiente, se realiza la apróximación numérica de estos valores utilizando el método de Newton-Rapson. En éste caso se utilizará $n = 2$ para garantizar que la función del integrando sea interpolable por un polinomio menor al grado $2n$.

In [18]:
```
# Original code: http://w3mentor.com/learn/python/scientific-computation/gauss-legendre-m-poin

    ''' x,A = gaussNodes(m,tol=10e-9)
        Returns nodal abscissas {x} and weights {A} of
        Gauss-Legendre m-point quadrature.
    '''
    from math import cos,pi
    from numpy import zeros

    def gaussNodes(m,tol=10e-9):

        if m<2:
```

```
            raise Exception("m parameter must be greater than 1.");

        def legendre(t,m):
            p0 = 1.0; p1 = t;
            for k in range(1,m):
                p = ((2.0*k + 1.0)*t*p1 - k*p0)/(1.0 + k )
                p0 = p1; p1 = p
            dp = m*(p0 - t*p1)/(1.0 - t**2)
            return p,dp;

        A = zeros(m);
        x = zeros(m);
        nRoots = (m + 1)//2          # Number of non-neg. roots

        for i in range(nRoots):
            t = cos(pi*(i + 0.75)/(m + 0.5))  # Approx. root
            for j in range(30):
                p,dp = legendre(t,m);         # Newton-Raphson
                dt = -p/dp; t = t + dt        # method
                if abs(dt) < tol:
                    x[i] = t; x[m-i-1] = -t
                    A[i] = 2.0/(1.0 - t**2)/(dp**2) # Eq.(6.25)
                    A[m-i-1] = A[i];
                    break;
        return x,A
    nRoots = 2;
    [roots,Coeff] = gaussNodes(nRoots);
    print('Number of roots: ', nRoots);
    print('Roots =',roots);
    print('Coefficients = ',Coeff);
```

```
Number of roots:  2
Roots = [ 0.57735027 -0.57735027]
Coefficients =  [ 0.99999997  0.99999997]
```

```
In [19]: Integrand = lambda t: (1/(t**5))/(np.exp((1/t)-1));
         Sm = 0;
         for index in range(0,nRoots-1):
             Sm = Coeff[index]*Integrand(roots[index]) + Sm;
         sigmaGL = (2*pi*Kb**4)/(h**3*c**2)*(Sm);
         sr = comptraprule(0.1, 10, 10, Integrand);
         sigmaT = (2*pi*Kb**4)/(h**3*c**2)*(sr);
```

2.6.2 Solución analítica de la constante de Stefan-Boltzmann

Para resolver la forma analítica y comparar los resultados, la clave de la integración está en tener la serie:

$$\frac{\mathrm{e}^{-x}}{1-\mathrm{e}^{-x}} = \sum_{n=1}^{\infty} \mathrm{e}^{-nx} \tag{35}$$

Para llevar la expresión a esa forma, el integrando se divide por $\exp u$ obteniendo:

$$E = \frac{2\pi\,(k_B T)^4}{c^2 h^3} \int_0^{\infty} \frac{u^3}{\mathrm{e}^u - 1}\mathrm{d}u = \frac{2\pi\,(k_B T)^4}{c^2 h^3} \int_0^{\infty} \frac{u^3 \mathrm{e}^{-u}}{1-\mathrm{e}^u}\mathrm{d}u = \frac{2\pi\,(k_B T)^4}{c^2 h^3} \sum_{n=1}^{\infty} \int_0^{\infty} u^3 \mathrm{e}^{-nu} \tag{36}$$

Ésta expresión puede ser evaluada utilizando la función gamma, si se considera:

$$t = nu \longrightarrow u^3 = \frac{t^3}{n^3} \therefore \mathrm{d}t = n\mathrm{d}u \longrightarrow \mathrm{d}u = \frac{\mathrm{d}t}{n} \tag{37}$$

Planteando:

$$\Gamma(Z) = (Z-1)! = \int_0^\infty t^{z-1}\mathrm{e}^{-t}\mathrm{d}t = \int_0^\infty \frac{1}{n^4}t^3\mathrm{e}^{-t}\mathrm{d}t \therefore 3 = Z-1 \longrightarrow Z = 4 \therefore \Gamma(4) = 3! = 6 \longrightarrow \int_0^\infty \frac{1}{n^4}t^3\mathrm{e}^{-t}\mathrm{d}t = \frac{6}{n^4} \tag{38}$$

Por consiguiente:

$$E = \frac{2\pi\,(k_BT)^4}{c^2h^3}\sum_{n=1}^{\infty}\int_0^\infty u^3\mathrm{e}^{-nu} = \frac{2\pi\,(k_BT)^4}{c^2h^3}\sum_{n=1}^{\infty}\frac{6}{n^4} \tag{39}$$

La parte de la sumatoria, puede verse como un caso particular de la función zeta $\zeta(s)$ de Riemann, tal que:

$$E = \frac{2\pi\,(k_BT)^4}{c^2h^3}\sum_{n=1}^{\infty}\frac{6}{n^4} = \frac{2\pi\,(k_BT)^4}{c^2h^3}6\zeta(4) = \frac{2\pi\,(k_BT)^4}{c^2h^3}6\frac{\pi^4}{90} = \frac{2\pi^5k_B^4T^4}{15c^2h^3} \therefore \sigma = \frac{2\pi^5k_B^4}{15c^2h^3} \longrightarrow E(T) = \sigma\,T^4 \tag{40}$$

Ésta expresión se conoce como la ley de Stefan-Boltzmann.

2.6.3 Comparación

Sustituyendo el conjunto de constantes, el valor numérico es $\sigma = 5.670e{-}8\,[W\,K^4\,m^{-2}]$. Comparando éste valor con el numérico se obtiene:

```
In [20]: ErR = lambda VR,VO: abs(VR-VO)/VR*100;
         print('Valor analítico: %1.3e'% SIGMA);
         print('Método de punto medio: %1.3e. Error relativo: %1.2f%%.' % (sigmaGL, ErR(SIGMA,sigmaGL)))
         print('Método del trapecio %1.3e. Error relativo: %1.2f%%.' % (sigmaT, ErR(SIGMA,sigmaT)));
```

```
Valor analítico: 5.670e-08
Método de punto medio: 6.537e-08. Error relativo: 15.29%.
Método del trapecio 5.982e-08. Error relativo: 5.51%.
```

2.7 Referencias y otros recursos

1. Serway, Raymond A. Física Moderna. Capítulo 3: "La teória cuántica de la Luz", (Ch3), Apéndice Web Segundo. Thomson Learning,inc. 2005.
2. Michael Fowler. Modern Physics. "Black Body Radiation". University of Virginia 2008. URL
3. Rafael Zamora Ramos. Estudio de la radiación del cuerpo negro. Determinación de la constante de Wien
4. Universidad del País Vasco - Campus de Gipuzkoa. La radiación de cuerpo negro
5. Maria de Fátima Oliveira Saraiva y Kepler de Souza Oliveira Filho - Kepler de Souza Oliveira Filho. Radiación de cuerpo negro
6. Jaan Kiusalaasr. Numerical Methods in Engineering With Python 3. 3th ed. "Chapter 5 - Numerical Differentiation". Cambridge University Press. 2013. PP. 183 - 185.
7. Richard L. Burden, J. Douglas Faires. Numerical Analysis 9th ed. "Chapter 4 - Numerical Differentiation and Integration". Cengage Learning. 2010. pp: 153 - 156.
8. Ronald E. Walpole, Roanoke College, Raymond H. Myers, Sharon L. Myers.Probability and statistics for engineers and scientists. Chapter 11. Simple Linear Regression and Correlation. Pearson Education. 2012.