Universidad Complutense de Madrid

Proyecto de Innovación y Mejora de la Calidad Docente 185
(Convocatoria 2013)

Manual de uso de IPython Notebook para docentes

Autores:
E. Cabrera, E. Díaz, O. G. Calderón, S. Melle, F. Domínguez-Adame y J. M. Ezquerro

Directora:
E. Díaz

Índice general

Motivación **1**

¿Qué es IPython Notebook? **3**

Puesta en marcha y uso **7**

Propuestas Docentes **23**

Conclusiones **29**

Referencias **31**

Anexo **33**

Motivación

La necesidad de adquirir habilidades en programas de cálculo numérico y simbólico es cada vez mayor en los estudios de carreras científicas, dado el absoluto protagonismo de las tareas de programación en trabajos de investigación y en la empresa privada. En este sentido, no sólo es necesario dotar a los estudios de Grado de asignaturas dedicadas al aprendizaje de este tipo de herramientas sino integrarlos en el resto de materias, con el fin de mostrar al estudiante distintos escenarios donde su uso sea beneficioso.

Es habitual el uso de programas de ordenador para elaborar figuras que ayuden a mostrar resultados que de otro modo serían difíciles de visualizar. Sin embargo, el enriquecimiento de las asignaturas sería mucho mayor si los programas se combinaran en un mismo documento con texto explicativo, imágenes, vídeos u otros contenidos multimedia, empleando para ello plataformas como Java o Flash. De esta forma, independientemente de la disciplina que se imparta, se trabajaría en un marco común, claro y de fácil seguimiento, no sólo para usuarios más diestros sino también para los que estén comenzando a utilizar este tipo de programas. Así, el estudiante disfrutaría de un entorno con distintas herramientas combinadas que él mismo podría ampliar y modificar, con el fin de facilitar la comprensión de los conocimientos a adquirir. La integración de otros formatos docentes aparte de código ejecutable no es lo habitual en los programas usualmente utilizados, como pueden ser MATLAB o MAPLE. Por otro lado, estos programas comerciales poseen una desventaja clara para su uso entre los estudiantes, e incluso en las universidades: el coste de la licencia. Aunque haya licencias para estudiantes, más baratas a costa de limitar el uso de ciertas funciones, su pago supone una barrera inicial para un uso

amplio por parte del alumnado.

Otro punto a considerar sobre el uso de este tipo de programas comerciales es la dificultad en su instalación en diferentes sistemas operativos, lo que puede desanimar a los estudiantes que no disponen de la ayuda de un experto. Aunque el acceso a aulas de informática en las que están instalados estos programas previamente minimiza este problema, un mayor uso de este tipo de programas implica una saturación de estas aulas, y por tanto, del número de licencias accesibles que permite la Universidad.

En este manual presentamos una alternativa que solventa todas las dificultados que hemos comentado. Se trata de la plataforma IPython Notebook, basada en el lenguaje de programación Python, software libre y gratuito. Nuestra propuesta tiene como objetivo fomentar el uso integrado de códigos de programación en la presentación de contenidos para que el estudiante explore distintos casos de un problema o materia explicada en clase. Además, el empleo de este entorno de programación hace que el alumno adquiera la formación necesaria para posteriormente llegar a ser usuario de otras plataformas de uso común en el mercado, ya sean de acceso libre o de pago.

En el Anexo de este manual incluímos además varios documentos que se han diseñado para asignaturas del Grado en Óptica y Optometría de la Facultad de Óptica y Optometría y el Grado en Física de la Facultad de Ciencias Físicas de la Universidad Complutense de Madrid.

¿Qué es IPython Notebook?

Orígenes

La plataforma IPython Notebook se basa en el programa IPython [1], que es un intérprete de comandos para el lenguaje de programación Python, donde se ha potenciado la componente interactiva (de ahí la letra *I* en su nombre). IPython es software libre desarrollado por múltiples programadores, destacando Fernando Pérez de la Universidad de Berkeley (California, EEUU) [2]. Desde su versión 0.12 se ha desarrollado una interfaz, denominada Notebook, basada en un entorno computacional web que se visualiza en un navegador, y que permite la inclusión de cualquier elemento accesible a una página web, además de permitir la ejecución de código escrito en el lenguaje de programación Python. Esta interfaz se ejecuta separadamente del núcleo de ejecución de computación.

¿Por qué IPython Notebook?

Las características ya mencionadas permiten solucionar los problemas comentados anteriormente, relativos al uso de programas de cálculo en el desarrollo de un curso universitario:

- La elección de una interfaz web permite la combinación de texto, fórmulas, código, figuras, y medios audiovisuales en un único documento, lo que facilita una explicación más detallada y atractiva de los conceptos que se quieran describir.
- La separación del núcleo del sistema de la interfaz, y el hecho de que ésta se visualice en cualquier navegador moderno, permite el acceso remoto.

- Su instalación y uso es completamente gratuito, por lo que el empleo de estos documentos no se ve lastrado por la necesidad de adquirir una licencia.

El código a ejecutar ha de ser escrito en el lenguaje de programación Python. Este lenguaje se utiliza ampliamente en el ámbito científico, y su extensión no ha parado de crecer (en el año 2012 ha sido el lenguaje de programación más usado). Conocer este lenguaje puede resultar, por tanto, muy beneficioso para el estudiante en su futuro profesional. Además, posee un amplio abanico de módulos científicos que se importan en los documentos, igualando el rendimiento de programas de cálculo comerciales en prácticamente todas las facetas.

Las librerias más relevantes en el área de ciencias son:

- **SciPy**: agrupa funciones relevantes para el cálculo numérico.
- **NumPy**: proporciona funciones específicas para el cálculo numérico vectorial y matricial.
- **SymPy**: agrupa las funciones necesarias para el cálculo simbólico.
- **Matplotlib**: contiene herramientas para la elaboración de gráficos 2D y 3D (semejante a la generación de gráficos en MATLAB).
- **Mayavi**: librería específica para la creación de gráficos 3D.
- **Pandas**: librería especializada en la manipulación y análisis de datos.

Aparte de estas librerías, destacamos PyLab, parte de la librería Matplotlib, que proporciona un entorno muy parecido a MATLAB, lo que facilita enormemente la transición desde este programa comercial a la computación científica con Python.

Ventajas respecto a otros programas de cálculo

Presentamos ahora una comparación, desde el punto de vista docente, entre diversos programas de cálculo populares, tanto gratuitos (como Octave o Sage) como comerciales (como MATLAB, Mathematica o MAPLE). Puesto que la plataforma IPython Notebook se basa en el lenguaje Python, extendemos también esta comparación a

otros lenguajes de programación, como son C o Fortran. Estos datos se resumen en el Tabla 1.

	IPython Notebook	Matlab	Mathematica, Maple	Sage	C, Octave, Fortran
Texto	Sí	No	Sí	Sí	No
Multimedia	Sí	No	No	No	No
Código	Sí	Sí	Sí	Sí	Sí
Acceso Remoto	Sí	No	No	Sí	No
Gratuito	Sí	No	No	Sí	Sí

Tabla 1. *Características docentes más relevantes de varios programas de cálculo numérico y simbólico.*

A la vista de la comparación podemos concluir que la principal ventaja de IPython Notebook frente a otras soluciones es la posibilidad de mostrar diferentes tipos de medios en un mismo documento, algo en lo que fallan todas las demás alternativas. Los programas comerciales considerados han sido MATLAB, Mathematica y MAPLE, mientras que los programas basados en software libre han sido Sage y Octave. En esta comparación destaca el programa Sage por estar basado en el lenguaje Python y debido a su interfaz parecida a la de IPython Notebook. En su contra (desde el punto de vista docente) pesa que no es posible incluir otros tipos de medios más que código en sus documentos.

Puesta en marcha y uso

Instalación del software IPython Notebook

IPython Notebook es una de las posibles interfaces del intérprete IPython, por lo que para instalar esta plataforma es necesario instalar IPython. Debido a su constante desarrollo, es muy recomendable visitar la página del proyecto [1] para comprobar cuál es su versión actualizada. En dicha página también se incluyen las instrucciones para instalar el software en los distintos sistemas operativos. En las siguientes líneas resumimos esos pasos y presentamos las opciones que consideramos más sencillas para su instalación.

- **Linux:** la mayor parte de las distribuciones de Linux (Ubuntu, OpenSuse, Fedora) poseen en sus repositorios de software el paquete IPython, por lo que es aconsejable utilizar sus gestores de paquetes para instalarlo. A continuación se muestran los comandos necesarios para distintas distribuciones Linux.

  - Ubuntu:

    ```
    sudo apt-get install ipython-notebook
    ```

  - OpenSuse:

    ```
    sudo zypper install ipython python-jinja2
    python-tornado python-pygments
    ```

  - Fedora:

    ```
    sudo yum install python-ipython-notebook
    ```

  Además de dicho paquete, si queremos incluir las capacidades de los módulos científicos, debemos instalarlos aparte a través de los siguientes comandos:

  - Ubuntu:

    ```
    sudo apt-get install ipython-matplotlib python-scipy
    python-pandas python-sympy python-nose
    ```

  - OpenSuse:

    ```
    sudo zypper install python-numpy python-scipy
    python-sympy python-pandas python-matplotlib
    ```

  - Fedora:

    ```
    sudo yum install python-matplotlib scipy python-pandas
    sympy python-nose
    ```

- **Windows y OS X:** tanto en Windows como en OS X, la forma más sencilla de obtener IPython Notebook consiste en instalar alguna de las distribuciones basadas en Python que la incluyen, como son las facilitadas por Coninuum Analytics (Anaconda [3]) y Enthought Python Distribution (Canopy [4]). Estas distribuciones, aparte de IPython, facilitan los módulos científicos más usados de forma gratuita, mientras que cobran por otros más específicos y por su soporte.

  Tanto una como otra incluyen los módulos usados en las aplicaciones docentes explicadas en este manual en su versión gratuita, por lo que son igualmente recomendables. Para instalarlas, es necesario visitar la página de descarga de Anaconda [3] o bien de la versión gratuita de Enthought Canopy [4], donde se ofrecen archivos ejectuables que instalan ambos programas de forma inmediata. Debido a que son productos que incluyen muchos paquetes, su ciclo de actualización suele ser más lento que el proporcionado por el equipo de IPython. Para conseguir la última versión de IPython Notebook es recomendable actualizar el paquete de IPython incluido en estas distribuciones cada cierto tiempo. Para ello, es necesario ejecutar los siguientes comandos en un terminal (en Windows se puede abrir pinchando en *Ejecutar* y escribiendo `cmd.exe` en el cuadro de diálogo que aparece):

  - Anaconda:

```
conda update conda
conda update ipython
```

– Enthought Canopy:

```
enpkg enstaller
enpkg ipython
```

Estas distribuciones científicas de Python también están disponibles para el sistema operativo Linux, siendo su instalación similar. Por lo que suponen una alternativa a la instalación a partir de los repositorios de cada distribución.

Una vez realizada la instalación en cualquiera de los sistemas operativos mencionados, la ejecución de IPython Notebook se puede realizar abriendo un terminal y ejecutando `ipython notebook`. Igualmente se hará pinchando con el ratón en el icono correspondiente de los programas Anaconda o Canopy si se han instalado estas distribuciones.

Transformación de códigos escritos en otro lenguaje

Los profesores que imparten docencia en alguna asignatura de carácter científico usualmente han elaborado ya códigos que les asisten en la generación de figuras o que permiten profundizar en alguno de los conceptos explicados en clase. En este sentido, es interesante poder dar la opción de utilizar dichos programas o de facilitar las modificaciones necesarias para incluirlos dentro de la interfaz web de IPython Notebook.

Debido a la gran variedad de lenguajes de programación y software de cálculo numérico y simbólico, nos centraremos en este apartado en la transformación del código procedente de tres posibilidades: MATLAB (u Octave en su versión de software libre), Mathematica y el lenguaje de programación C.

- **MATLAB/Octave.** MATLAB es uno de los programas científicos más utilizados dada su potencia y versatilidad, así como su facilidad de aprendizale. Para integrar los programas ya escritos dentro de un documento de IPython

Notebook existen dos posibilidades: hacer que el propio notebook ejecute los comandos de MATLAB, o bien reescribir el código en lenguaje Python. Aunque esta última propuesta puede ser un poco más tediosa, se ganaría en legibilidad y en el uso de un único recurso, en este caso Python.

Existen diversas opciones para llevar a cabo la primera opción. En este manual nos centraremos en la extensión de IPython Notebook denominada *pymatbridge*, que proporciona un puente entre MATLAB e IPython para ejecutar directamente los comandos de aquel en un notebook. Es necesario resaltar que esta solución implica disponer de una instalación de MATLAB funcional para que se ejecuten los comandos. Es decir, la extensión llama a MATLAB para que se ejecuten los comandos, y devuelve el resultado para ser mostrado en el documento. Esta herramienta permite utilizar los archivos con extensión .m sin ningún cambio, simplemente escribiendo `%%matlab` en la primera línea de la celda que contenga el código. Las instrucciones para su instalación y su funcionamiento se pueden encontrar en la web [5]. Sin embargo, su uso queda limitado actualmente al sistema operativo Linux, aunque es previsible su actualización a versiones operativas en todos los sistemas en un futuro.

Una solución análoga se encuentra disponible para Octave, el clon de software libre de MATLAB. En esta ocasión la extensión se denomina *octavemagic* y depende del módulo `oct2py.py`, que también debe estar instalado. A diferencia de la anterior, es posible utilizarla en cualquier sistema operativo a día de hoy. Igualmente incluyendo `%%octave` en la primera línea de una celda el programa ejecutará el código Octave escrito a continuación. Como en el caso anterior, es necesario tener instalado Octave en el sistema para que la extensión funcione. Los detalles sobre este proyecto se pueden consultar en [6].

A pesar de que soluciones como las indicadas anteriormente facilitan enormemente el paso a IPython Notebook y permiten integrar códigos ya escritos con texto, imágenes y vídeos, quizás para una mayor legibilidad y para no depender de programas externos, la transformación de los códigos a lenguaje Python es la opción óptima.

En la Tabla 2 facilitamos algunos de los comandos más habituales de MATLAB y su equivalente en Python, usando alguno de los módulos de computación científica como NumPy ó SciPy. Se puede encontrar más información la equivalencia de comandos entre MATLAB y NumPy/Scipy en la página web [7].

Operaciones	**MATLAB**	**NumPy/SciPy**
Primer elemento del array `a`	`a(1)`	`a[0]`
Ãltimo elemento del array `a`	`a(end)`	`a[-1]`
Multiplicación elemento a elemento de dos arrays `a` y `b`	`a.*b`	`a*b`
Tamaño de una matriz `a`	`size(a)`	`shape(a)`
Encontrar los índices donde `a > 2`	`find(a>2)`	`nonzero(a>2)`
Máximo de una matriz `A`	`max(max(A))`	`A.max()`
Generar un vector con elementos de 0 a 5	`0:5`	`arange(6)`
Generar una matriz 3×3 de ceros	`zeros(3,3)`	`zeros((3,3))`
`a` elevado a 4	`a^4`	`a**4`
Transformada de Fourier de `a`	`fft(a)`	`fft.fft(a)`

Tabla 2. *Equivalencia de algunos comandos entre MATLAB y NumPy.*

- **Mathematica.** Mathematica es un software más orientado al cálculo simbólico, aunque en las últimas versiones sus desarrolladores han aumentado sus capacidades relativas al cálculo numérico. Quizás el módulo de Python más parecido sea Sympy, el cual permite realizar este tipo de operaciones de una forma sencilla. A pesar de que su potencia y librería de funciones no llega a la altura del software comercial Mathematica, es más que suficiente para un uso docente, así como para una gran parte de las necesidades científicas. Por otro lado, su constante desarrollo augura mejoras continuadas en el futuro.

  Al igual que sucede con MATLAB, la comunidad de IPython ha creado una extensión que sirve de puente entre la plataforma IPython Notebook y el programa Mathematica. Esta extensión se denomina *IPython-mathematicamagic-*

extension y su proyecto se aloja actualmente en [8]. De la misma manera que en el caso anterior, permite utilizar comandos de Mathematica directamente dentro de una celda de IPython Notebook, sin ningún tipo de transformación.

Si se escoge la opción de usar Sympy para realizar las operaciones de cálculo simbólico, a continuación se facilita la Tabla 3 con algunas de las equivalencias entre comandos de Mathematica y de Sympy. Se puede encontrar más información en la página web [9].

Operaciones	**Mathematica**	**SymPy**
Definición de un símbolo	No lo necesita	`x=Symbol('x')`
Descomposición de fracciones	`Apart[expr]`	`apart(expr,x)`
Combinación de fracciones	`Together[expr]`	`together(expr,x)`
Evaluar una expresión	`N[]`	`N()/evalf()`
Límite de una función	`Limit[expr,x->x0]`	`limit(expr,x,x0)`
Diferenciación	`D[expr,var]`	`diff(expr,var)`
Expansión en serie de potencias	`Series[f,{x,x0,n}]`	`f.series(x,x0,n)`
Integral indefinida	`Integrate[f,x]`	`integrate(f,x)`
Integral definida	`Integrate[f,{x,a,b}]`	`integrate(f,(x,a,b))`
Solución de ecuaciones algebraicas	`Solve[expr,x]`	`solve(expr,x)`

Tabla 3. *Equivalencia de comandos entre Mathematica y Sympy.*

- **Lenguaje C/C++.** El lenguaje de programación C/C++ es ya "veterano" y, a pesar de ello, su extrema rapidez, especialmente al ejecutar bucles, lo hace uno de los lenguajes de referencia en computación científica junto con el lenguaje Fortran. Al ser Python un lenguaje interpretado, la ejecución de los códigos es más lenta que con un lenguaje compilado como es C. A cambio, se gana en legibilidad y facilidad de programación, lo cual ahorra también un tiempo tanto en el desarrollo de códigos como en el mantenimiento y búsqueda

de errores o problemas graves que podrían acontecer por la ejecución de un programa mal diseñado. Más importante desde un punto de vista docente es la facilidad de aprendizaje del lenguaje, mucho menor para C/C++ y Fortran, lo que limita su uso generalizado para generar códigos sencillos que den apoyo a la explicación de otros conceptos. Sin embargo, la búsqueda de una mayor rapidez a la hora de ejecutar ciertos programas, ha hecho que se desarrollen varias opciones para reutilizar código C en Python. A continuación describimos algunas de ellas.

- **Weave** es parte del módulo Scipy, y permite escribir código C o C++ directamente dentro de un programa de Python. Para su uso se necesita disponer de un compilador de C/C++ instalado, además de la instalación de Python. Weave se puede utilizar de dos formas diferentes: con la función `inline` y con la función `blitz`. Con la primera de ellas, el código en C/C++ se escribe directamente como una cadena de caracteres (entre comillas), dentro de la función `inline()`. La primera vez que corre el programa, Weave se encarga de compilar ese código (llamando al compilador que se tenga instalado) y genera una librería que en sucesivas ejecuciones no necesitará ser compilada, y por tanto disminuirá sensiblemente el tiempo de ejecución. Por otro lado, la función `blitz()` se encarga de transformar expresiones del módulo NumPy a código C/C++ que realiza exactamente la misma función, pero de manera mucho más rápida. En este caso, la ventaja principal es que el usuario no necesita saber programar en C/C++, sino únicamente usar las expresiones del módulo NumPy.
- **Cython** ha ganado en popularidad en los últimos años y presenta una solución muy flexible para ganar algo de tiempo en la ejecución de programas escritos en Python. A diferencia de Weave, Cython es un lenguaje de programación en sí mismo, basado en Python, y que lo engloba. Puede ejecutar código escrito en Python, o bien modificarlo incluyendo definición de variables con declaración de tipos, lo que permite agilizar la ejecución. El uso de Cython requiere un conocimiento más avanzado que

lo expuesto en el presente Manual, por lo que no se abordará aquí. Para más información se puede acudir a la página web del proyecto [10].

En estas líneas únicamente se ha dado una muestra de las capacidades de Python para reciclar códigos escritos en otras plataformas. Estas capacidades no acaban en las opciones descritas, y otros lenguajes y software científico como Fortran, IDL ó R también disponen de una amplia comunidad que aporta soluciones para llevar a cabo la migración de los códigos a Python y a los documentos de IPython Notebook. Para más información acerca de distintas extensiones y opciones, puede consultarse la página web [11].

Apariencia del Notebook

Presentamos ahora brevemente la interfaz web de esta herramienta y los principales elementos necesarios para su uso (véase Figura 1). El documento se divide en distintas celdas, el comportamiento de las cuales puede seleccionarse entre distintas opciones: título (de diversos tamaños), texto (plano o enriquecido utilizando el lenguaje Markdown) o código. Aparte de estas opciones, las librerías específicas permiten importar imágenes o vídeos, incluir enlaces a recursos web o incluso incrustar una página web completa dentro del documento.

Comandos generales

Puesto que IPython Notebook y Python son proyectos de software libre, existe una gran cantidad de información acerca de ellos disponible en la red. Gran parte de este material se puede encontrar directamente a través de la pestaña Help que se encuentra en el navegador de cualquier documento de IPython Notebook. Por eso aquí sólo mencionaremos los aspectos que consideramos más relevantes para un uso extendido en aplicaciones docentes del área de ciencias.

- Inclusión de texto.
  - **Celdas de texto**: el texto se incluye en lenguaje Markdown que permite darle formato al texto de una manera sencilla. Además, se pueden

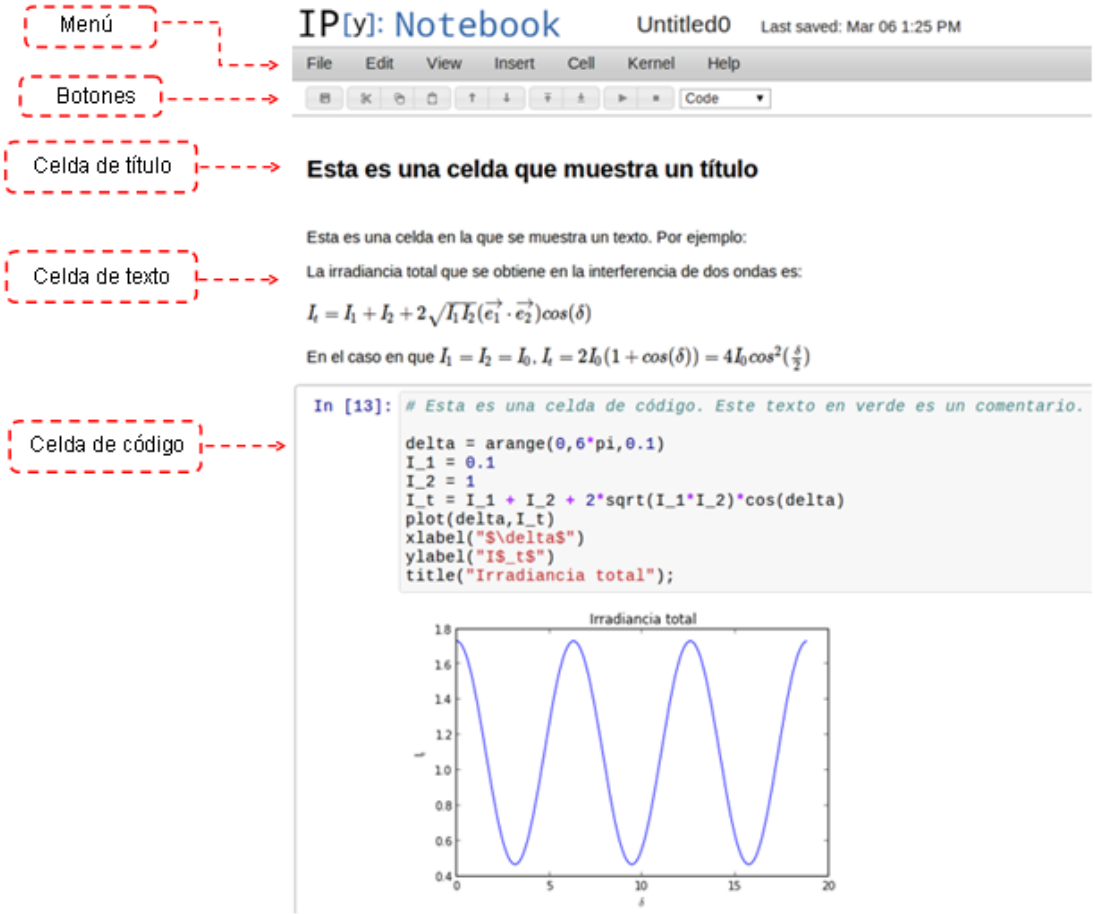


Figura 1. *Ejemplo de un documento IPython Notebook dividido en celdas*

incluir expresiones matemáticas en código LaTeX gracias a la inclusión de MathJax, que procesa la expresión escrita en LaTeX y la muestra de manera legible para el usuario.

Las fórmulas dentro de una línea de texto se escriben entre símbolos dólar

$ . . . $

Las fórmulas en líneas separadas del texto se escriben entre símbolos dólar doble

$$ . . . $$

Además, en este tipo de celdas es posible incluir lenguaje HTML directamente para tener más posibilidades de formato.

  - **Comentarios en celdas de código**: si el comentario ocupa una línea se escribe delante ♯ . . . , si ocupa varias líneas se escribe entre comillas " . . . "

- Gestión de librerías

  - **Importar todas las funciones de una librería**: si usamos la librería numpy como ejemplo habría que escribir

    ```
    from numpy import *
    ```

    Existen funciones que aparecen en dos librerías diferentes y que actúan de forma diferente, por lo que a veces resulta útil importar una librería con un prefijo distintivo. Además, y especialmente en el caso de la escritura de archivos ejecutables en Python, es una buena práctica de programación incluir este prefijo para facilitar una lectura posterior del archivo y encontrar posibles errores en él. En cualquier caso, si elegimos el prefijo np para la librería numpy ha de escribirse

    ```
    import numpy as np
    ```

  - **Importar una función particular**: para importar una única función concreta de una librería, por ejemplo la función coseno de la librería numpy, escribiríamos

    ```
    from numpy import cos
    ```

  - **Consejo para aplicaciones docentes**: en general, para tratar cualquier ejercicio de asignaturas de ciencias, lo más cómodo es posiblemente utilizar el entorno PyLab, el cual, una vez ejecutado, carga en el sistema las funciones más utilizadas y permite escribir código de una manera muy similar a MATLAB. Además, para que las figuras ejecutadas en el notebook aparezcan en el mismo documento y no en una ventana aparte es necesario utilizar el comando inline. En resumen, para cubrir estos aspectos se escribiría al comienzo del notebook

    ```
    %pylab inline
    ```

- Comandos de código numérico:

- **Obtener ayuda**: para pedir información de alguna función concreta, por ejemplo de la función coseno, se escribiría

  `help(cos)` ó `help?`

- **Escritura en pantalla**: para que al ejecutar la celda de código se muestre un texto o el valor de una variable, llamémosla x por ejemplo, se escribiría

  ```
  print('La variable x tiene el valor')
  print(x)
  ```

- **Incluir una imagen de archivo**: es necesario importar la función Image. Su sintaxis es (en este caso se considera que el archivo de la imagen estaría en la misma carpeta que el del notebook, aunque es también posible llamar a una imagen alojada en Internet),

  ```
  from IPython.display import Image
  Image(filename="FiguraTest")
  ```

  Como alternativa, una imagen se puede incluir directamente en una celda de texto de tipo Markdown, con la sintaxis propia de Markdown, o bien si se quiere editar el tamaño o su posición dentro de la celda, incluyendo lenguaje HTML. La forma de incluir una imagen en Markdown se realiza de la siguiente forma:

  ```
  ![Texto Alternativo](files/FiguraTest)
  ```

  En el anterior comando, la parte de texto alternativo indica el texto que aparece en sustitución de la imagen si hay algún problema al cargarla. La parte entre paréntesis es la ruta de la imagen (la cual puede ser una dirección web). En ella destaca que para incluir una imagen que se encuentre en el mismo directorio en donde tenemos el notebook, debemos incluir la ruta con `files/` en ella.

- **Incluir una web**: IPython Notebook permite incrustar una página web dentro del notebook, indicando la anchura y la altura del marco en el que aparece dicha web. Para ello, debemos importar la función HTML. Un ejemplo de su uso sería el siguiente,

```
from IPython.display import HTML
HTML('<iframe
src=http://www.wikipedia.org width=800px height=400px >')
```

- **Incluir un video de YouTube**: La inclusión de vídeos permite incrementar las posibilidades docentes. Dentro de las distintas opciones, YouTube dispone de una cantidad ingente de vídeos científicos. Para mostrar uno de ellos disponemos de la función Youtube que se usa de la siguiente manera, utilizando como identificador el código que YouTube asigna a cada vídeo:

  ```
  from IPython.display import Youtube
  Youtube("codigovideo")
  ```

- **Bucle**: nos fijamos en el bucle más común que es el bucle `for`. Si queremos por ejemplo escribir en pantalla los números del 1 al 10 escribiríamos

  ```
  for j in range(1,10):
     print(j)
  ```

- **Condiciones**: tratamos los tres casos más generales. La condición más sencilla suele escribirse con el comando `if` con la sintaxis que se muestra en el siguiente ejemplo

  ```
  if j>0:
     print('La variable j es positiva')
  ```

  Para una condición que incluya dos casos se utiliza el `if ... else ...` de la siguiente manera

  ```
  if j>0:
     print('La variable j es positiva')
  else:
     print('La variable j es negativa o cero')
  ```

  Por último, el comando `while` que se utiliza en condiciones relacionadas con bucles se utilizaría como en el siguiente ejemplo

  ```
  j=-5
  ```

```
while j<0:
   print('La variable j todavía es negativa')
   j++
```

– **Definición de funciones**: para definir una función particular se utiliza el comando `def`. Por ejemplo si queremos definir la función escalón,

```
def fun(x):
   if(x>0):
     return 1
   else:
     return 0
```

– **Vectores y matrices**: hay diferentes maneras de crear vectores y matrices así que aquí sólo consideramos algunos ejemplos.

Para crear un vector de números con un número fijo de elementos (100 en el ejemplo) con valores comprendidos entre un valor inicial y otro final (0 y 100 por ejemplo) es muy útil la función `linspace` con la siguiente sintaxis

```
x=linspace(0,10,100)
```

Si en vez del número de elementos, queremos especificar el paso entre los distintos elementos del vector, podemos utilizar la función `arange`. Por ejemplo, si queremos un paso de 1 y generar un vector entre los valores 0 y 10, tendremos que escribir:

```
arange(0,11,1)
```

(Nótese que el segundo elemento de la función `arange` ha de ser el valor final que busquemos más el paso del vector.)

Otras funciones útiles para generar matrices de dimensión `m` $\times$ `n` pueden ser las siguientes:

* Si todos los elementos son igual a cero.

  ```
  zeros((m,n))
  ```

* Si todos los elementos son igual a uno.

`ones((m,n))`

* Si los elementos no tienen un valor inicial predeterminado.

`empty((m,n))`

- **Creación de un gráfico**: un ejemplo para la realización de un gráfico básico, de la función coseno en este caso, sería

```
x=linspace(0,10,100)
y=cos(x)
plot(x,y)
xlabel('X')
ylabel('Y')
title('PlotTest')
```

Aquí las últimas tres líneas definen el título del eje x, del eje y y del gráfico respectivamente.

Como se ve en los ejemplos presentados, en IPython Notebook no es necesario separar cada línea de código por ningún signo de puntuación. Sin embargo, sí es muy importante respetar el indentado que el propio lenguaje incluye en la sintaxis de las funciones. Esta característica propia de Python facilita la legibilidad y limpieza del código, por lo que puede ser una ayuda para los estudiantes.

Por otro lado, cada celda de código se puede ejecutar utilizando las teclas mayúscula + intro o pulsando el boton de play que aparece en el navegador del notebook.

Exportar a otros formatos

Aunque los documentos generados por IPython Notebook son fácilmente accesibles a través de un navegador, en ocasiones es necesario o útil disponer de otros formatos, por ejemplo, utilizar un archivo Python ejecutable. Con el fin de facilitar la conversión de los *notebooks*, IPython dispone desde la versión 1.0 (versión actual a día de hoy) de la herramienta **nbconvert**, que permite exportar los documentos hechos

en IPython Notebook a otros formatos. Concretamente, **nbconvert** proporciona la conversión a documentos estáticos (es decir, las celdas no pueden ejecutarse), en formato HTML, LaTeX, Markdown, reStructuredText, script de Python ejecutable y, finalmente, en formato presentación. El uso de esta herramienta requiere la instalación de un nuevo paquete denominado *Pandoc*, que puede instalarse a través del gestor de paquetes de las distintas distribuciones Linux o bien, a través de su página web [12].

El funcionamiento de esta herramienta requiere por el momento el uso de una terminal o consola, aunque en un futuro se encuentra planeada su inclusión en el propio notebook como opción en uno de sus menús. Su sintaxis es

```
$ ipython nbconvert --to FORMAT notebook.ipynb
```

donde `FORMAT` es el formato que se desea obtener. Por ejemplo, si queremos convertir nuestro documento a un archivo en LaTeX, escribiríamos:

```
$ ipython nbconvert --to latex notebook.ipynb
```

Una vez generado el archivo .tex podemos convertirlo a pdf mediante, por ejemplo, pdflatex o cualquier editor de LaTeX. Si queremos realizar realizar la conversión a formato pdf directamente, podemos añadirlo a la anterior instrucción, quedando,

```
$ ipython nbconvert --to latex notebook.ipynb --post PDF
```

También es posible modificar las plantillas de nuestro archivo LaTeX generado a formato libro por ejemplo añadiendo `--template book`. Es necesario hacer notar que el formato presentación no genera un archivo PowerPoint o similar, sino una presentación Reveal.js HTML que requiere para su visualización que el ordenador muestre la presentación en el navegador. Este requisito se cumple si añadimos a la línea anterior la opción `--post serve`. Este tipo de presentaciones, aunque más difíciles de mostrar por la necesidad de que el ordenador actúe como servidor de esa página en HTML, dan un mayor número de opciones que las presentaciones tradicionales y son más sencillas de compartir a través de diferentes sistemas operativos y plataformas.

Aparte de estas opciones, podemos exportar el documento a formato PDF mediante la opción de imprimir a un archivo de este tipo desde el propio navegador, aunque el resultado es menos óptimo que la conversión a través del documento

LaTeX.

Para más información, se pueden consultar los siguientes enlaces [13, 14].

Propuestas Docentes

Consideraciones generales

Las Comisiones que han elaborado los protocolos del Nuevo Espacio Europeo de Educación Superior (EEES) destacan la importancia de la innovación docente para la mejora de las clases presenciales teóricas y prácticas, complementadas con una fuerte participación en seminarios y ejercicios realizados fuera del aula. En particular, el desarrollo de seminarios interactivos permite al alumno adquirir un conocimiento teórico a la vez que realiza actividades prácticas al respecto. En el área de ciencias es importante, además, dotar a los estudiantes de recursos de programación para favorecer el aprendizaje de técnicas numéricas imprescindibles en problemas científicos de alta complejidad, optimizando el tiempo que se emplea en la adquisición de conocimientos y minimizado el empleado en los cálculos que demuestran los aspectos teóricos fundamentales. En este marco, hemos propuesto el software IPython Notebook para el diseño de seminarios interactivos, constituyendo una nueva herramienta docente utilizable en el aula y puesta a disposición a través del Campus Virtual. Destacamos que en esta plataforma de cálculo simbólico y numérico, las líneas de texto explicativo, las líneas de comandos y la representación gráfica coexisten en el mismo entorno, facilitando así que los estudiantes puedan manejarlo aún sin conocer en detalle todas sus funcionalidades. El uso de estos seminarios puede realizarse a diferentes niveles de profundidad según las necesidades docentes y la motivación del alumno.

- El primer nivel consistirá en el uso de las explicaciones, apoyadas con figuras obtenidas de las simulaciones.

- En un siguiente nivel se plantearán ejercicios en los que el estudiante utilizará las simulaciones como una *caja negra*, es decir, cambiando los parámetros y obteniendo los resultados sin conocer el método de cálculo.

- En el nivel más avanzado los estudiantes podrán modificar los códigos para ampliar las simulaciones mostradas, adquiriendo así nuevas competencias en el cálculo numérico.

IPython Notebook como herramienta docente, aunque puede ser de utilidad en cualquier disciplina universitaria, es de especial relevancia en titulaciones de naturaleza científica, técnica o relativas a ciencias de la salud, donde usualmente se requiere una potente herramienta de cálculo numérico, simbólico o estadístico. Sin embargo, no es menos importante destacar otra de las ventajas principales para la comunidad universitaria, y es que establece una pasarela hacia el conocimiento de herramientas de software libre, cuya filosofía (y no solo por su menor coste económico) consideramos más afín a la enseñanza universitaria.

Nos gustaría destacar que la herramienta IPython Notebook no circunscribe su aplicación únicamente a las tareas docentes. Los módulos y funciones que se utilizan son también válidos para el desarrollo de programas más avanzados orientados a la investigación, pudiendo sustituir en gran parte el uso de otros programas comerciales de cálculo numérico y simbólico. Así pues, su aprendizaje resulta útil para la adquisición de competencias y capacidades para el ejercicio profesional futuro. En la web [15] se puede encontrar información sobre la gran cantidad de proyectos científicos y docentes que usan Python y la herramienta IPython para realizar y compartir sus cálculos. Entre ellos, cabe destacar el sistema *Ganga*, desarrollado en el CERN para el control de trabajos en red de los experimentos en el LHC, o en aplicaciones docentes como el curso de biología computacional en la Universidad de Michigan State, basado en la herramienta IPython Notebook. Por todo esto, el aprendizaje de esta herramienta y la base del lenguaje de programación Python permite a los alumnos y profesores interaccionar con otros proyectos científicos y docentes muy activos en otras partes del mundo.

Beneficios para los estudiantes

Detallamos ahora los aspectos del apredizaje que son potenciados por los seminarios interactivos desarrollados con IPython Notebook y los beneficios que este tipo de seminarios conllevan para los estudiantes::

- Favorecen el aprendizaje autónomo de los estudiantes a diferentes niveles de complejidad según los intereses y las necesidades de cada alumno.
- Permiten un aprendizaje más atractivo, ya que estos seminarios digitales pueden incluir imágenes y vídeos de alta calidad y enlaces a otras webs de interés.
- Facilitan el proceso activo de adquisición de competencias en el cálculo numérico.
- Promueven el aprendizaje de código de texto en formato LaTeX que se utiliza para la elaboración de documentos científicos de amplia difusión, incluso en imprenta.
- Promueven el conocimiento y aprendizaje de nuevos lenguajes de programación basados en software libre que no conllevan coste económico y que utilizan formatos estándar. Es decir, pueden ser fácilmente utilizados por otras personas y exportados con mayor facilidad.
- Ofrecen nuevas posibilidades de uso del Campus Virtual, con el que los estudiantes están cada vez más familiarizados.
- Permiten realizar tareas de evaluación y autoevaluación en remoto de una forma más flexible y adecuada a las necesidades particulares de cada alumno. Esto liberará parte del tiempo de clase presencial que podrá ser usado para ampliar las explicaciones del docente.
- Facilitan su utilización por parte de un mayor número de alumnos, ya que están disponibles sin necesidad de instalar ningún software comercial y se puede acceder a ellos a través de dispositivos electrónicos más dinámicos, como los teléfonos inteligentes o las tabletas.

Ejemplos de propuestas docentes

A continuación se presentarán los aspectos más relevantes de algunos de los documentos elaborados durante el Curso 2012/2013 para las asignaturas de Física y Óptica Física II correspondiente al Grado de Óptica y Optometría de la Facultad de Óptica y Optometría, y para la asignatura de Física de Estado Sólido del Grado de Física impartido en la Facultad de Ciencias Físicas, ambas adscritas a la Universidad Complutense de Madrid. Estos documentos se incluyen en el Anexo a este manual y se presentan como una pequeña muestra de las capacidades de IPython Notebook dentro de las tareas docentes. Es necesario indicar sin embargo que al incluirlos en formato pdf, son únicamente una versión estática que no permite ni la ejecución de código ni mostrar otros elementos dinámicos como aplicaciones en Java o vídeos incrustados dentro del documento.

Estos documentos no pretenden cubrir todos los conceptos explicados en los temas a los que hacen referencia dentro de las asignaturas mencionadas anteriormente. Tampoco deberían entenderse como documentos cerrados, sino, como cualquier material docente, en constante proceso de actualización y mejora para futuros cursos.

- **Interferómetro de Young**. Quizás el experimento más utilizado para mostrar las interferencias de ondas electromagnéticas y por tanto el carácter ondulatorio de la luz. Este documento explicativo incide en el desarrollo del cálculo que lleva a las expresiones utilizadas en las prácticas del laboratorio de la asignatura y presenta un código que el estudiante puede modificar para mostrar las franjas de interferencia observadas en distintas situaciones. Se plantean preguntas sobre cómo varían los resultados frente a posibles cambios de las condiciones del problema inicial. Por ejemplo, se pregunta cómo cambiaría el patrón de interferencia si se usase luz blanca en lugar de luz monocromática y se incluye una imágen con el resultado. Por último, para ampliar la información y dar soporte visual al proceso físico, se incluye un enlace a un vídeo alojado en YouTube y una aplicación de Java en la que se pueden cambiar condiciones del problema y obtener resultados visuales inmediatos.

- **Anillos de Newton**. Este documento es un ejercicio donde los estudiantes

han de calcular el radio de curvatura de una lente mediante la técnica interferométrica de los anillos de Newton a partir de la fotografía de un patrón de interferencia medido en el laboratorio. Al principio del notebook se incluye una imagen con el esquema del dispositivo experimental donde se originan los anillos de Newton y una fotografía real del patrón de interferencia que se observa en ese dispositivo con una escala de longitudes. Esta imagen puede ser tomada por cada estudiante en su sesión de laboratorio por ejemplo. Con esa información y el aumento producido por el dispositivo, los estudiantes pueden medir el radio de diferentes anillos oscuros del patrón interferencial. Por último, se incluye un código para determinar a partir de esos datos el radio de la superficie de la lente en cuestión mediante un ajuste lineal de la posición de los mínimos de irradiancia.

- **Filtros interferenciales**. Este documento presenta un ejercicio en donde el estudiante debe buscar en Internet un láser de alta potencia que emita en el visible, y encontrar en una página web (facilitada por el profesor) un filtro interferencial que proteja al usuario del haz emitido por dicha fuente. La capacidad de incrustar una página web en el notebook permite que el estudiante no tenga que salir del documento para buscar el filtro. Además, mediante un código que se incluye, el estudiante puede importar y representar los datos de la transmitancia de dicho filtro que la empresa facilita en la web. Después de navegar por una web comercial donde el estudiante aprende a buscar la información que necesita, y adquiere una idea de los precios de los dispositivos ópticos, emite al final un juicio sobre la bondad del filtro escogido para bloquear la radiación del láser.

- **Tubo de Quincke**. Este ejercicio presenta los conceptos más importantes que permiten estudiar el tubo de Quincke y su utilidad para medir la velocidad del sonido en aire. Como introducción se incluye una explicación con gráficas y cuestiones sobre dos ondas armónicas que interfieren en el espacio. Se estudia cómo la interferencia depende del desfase entre las dos ondas, considerando particularmente el desfase debido a la diferencia de camino recorrido por ca-

da una de las ondas. Después se presenta el tubo de Quincke incluyendo una imágen con el esquema del dispositivo experimental y se aplica la teoría ya explicada para contestar ciertas cuestiones. Destacamos que se incluye una animación en la que se representan las dos ondas viajeras en el espacio, y la onda estacionaria resultante de su interferencia en cada instante de tiempo. También se propone un pequeño experimento con sus correspondientes cálculos, realizable a través de una aplicación que se incrusta a través de la web que la aloja. Por último se incuye otra página web donde aparecen indicaciones para que los estudiantes interesados sepan como construir un tubo de Quincke.

- **Estados electrónicos de una cadena finita de pozos idénticos**. Este documento se dedica fundamentalmente a ciertos aspectos básicos del cálculo numérico. En él se plantea cómo construir la matriz correspondiente a un Hamiltoniano de enlace fuerte con interacción a primeros vecinos. En la parte diagonal se incluye el perfil de potencial por la cadena de pozos discretizada y la parte no diagonal la constituye un factor relativo a la interacción con vecinos próximos. El código presenta cómo diagonalizar la matriz y obtener las autoenergías y autoestados del Hamiltoniano. Los estudiantes pueden elegir el número y anchura de los pozos y el número del autoestado para conocer su energía y representar en una misma figura, el perfil de potencial y el autoestado. A través de este cálculo numérico se plantean diferentes ejercicios a los estudiantes para que observen cómo se distribuyen las bandas de energía y las funciones de onda en un sólido ordenado generado con la unión de diferentes átomos idénticos (simulados por pozos). Por último, se incluye una aplicación Java que estudia el mismo sistema y permite una variación de un número mayor de parámetros para que los estudiantes observen efectos diferentes.

Conclusiones

Tras la evaluación del uso de documentos interactivos basados en la herramienta IPython Notebook para aplicaciones docentes, hemos mostrado que esta plataforma ofrece un entorno rico en contenidos que facilitan el aprendizaje, integrando de manera natural simulaciones realizadas en Python. Así, el estudiante adquiere competencias en cálculo numérico a la vez que profundiza en los conceptos explicados en el aula. El acceso remoto y la ausencia de coste asociado a su licencia facilitan sin lugar a dudas la implantación de estas herramientas en los contenidos curriculares de los Grados en Ciencias.

Los documentos interactivos se pueden utilizar como texto explicativo y ampliado de los conceptos presentados en el aula, como ejercicio que evalúe el profesor o como ejercicio de autoevaluación del alumno. Dependiendo de la respuesta del alumnado, podría incluso involucrarse a los estudiantes en la elaboración de apuntes dinámicos (a la manera de una *Wiki*) que puedan ser mejoras en cursos sucesivos.

Bibliografía

[1] http://ipython.org

[2] http://fperez.org

[3] http://continuum.io/downloads

[4] https://www.enthought.com/products/epd/free/

[5] https://github.com/arokem/python-matlab-bridge

[6] http://nbviewer.ipython.org/url/github.com/ipython/ipython/raw/master/examples/note

[7] http://wiki.scipy.org/NumPy_for_Matlab_Users

[8] https://github.com/bjedwards/IPython-mathematicamagic-extension

[9] https://github.com/sympy/sympy/wiki/SymPy-vs.-Mathematica

[10] http://cython.org

[11] https://github.com/ipython/ipython/wiki/Extensions-Index

[12] http://johnmacfarlane.net/pandoc/installing.html

[13] http://ipython.org/ipython-doc/stable/interactive/nbconvert.html#nbconvert

[14] http://www.slideviper.oquanta.info/tutorial/slideshow_tutorial_slides.html?transition=no

[15] http://wiki.ipython.org/Projects_using_IPython

Anexo

Ejemplos de propuestas docentes

A continuación se incluyen algunos ejemplos de material docente generado a modo de ilustración de algunas de las capacidades de la interfaz web Notebook del software IPython. Estos documentos son únicamente versiones estáticas, y por tanto, algunas de las funcionalidades dinámicas, como la inclusión de páginas web, vídeos, aplicaciones Java o animaciones, son imposibles de mostrar. Sin embargo, el código que las genera sí se encuentra incluido, permitiendo reproducirlo en una instalación propia de IPython.

ExperimentoYoung

Eduardo Cabrera Granado

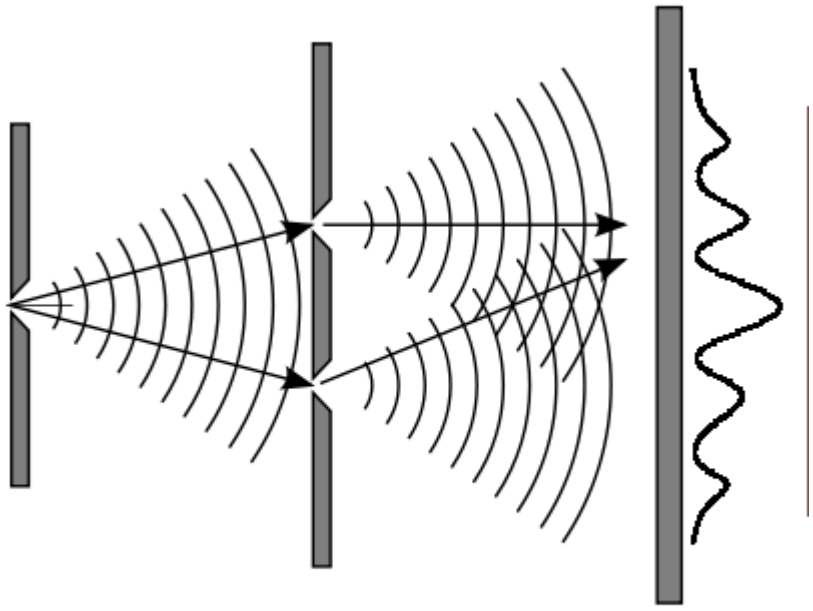
November 11, 2013

Part I

Experimento de Young

In [50]:
```
from IPython.display import Image
Image(filename="EsquemaYoung.png")
```

Out [50]:



Young's Two-slit Experiment

Consideremos el montaje detallado en la Figura 1. Una onda esférica (o bien una onda plana, el tratamiento es equivalente),incide en una pantalla sobre la cual se han realizado dos aperturas S_1 y S_2 muy próximas entre sí (llamaremos a la distancia entre ellas a). Estas aperturas actúan como dos fuentes secundarias de radiación, generando a su vez dos ondas esféricas que se superponen en el espacio que hay detrás de ellas. Si observamos la distribución de irradiancia en una pantalla situada a una cierta distancia D, ¿qué nos encontraremos?.Las dos ondas que se generan en S_1 y S_2 pueden escribirse como:

$$\vec{E_1} = E_{01}\vec{e_1}e^{i(kr_1-\omega t+\phi_1)}$$
$$\vec{E_2} = E_{02}\vec{e_2}e^{i(kr_2-\omega t+\phi_2)}$$

En el caso planteado, $\phi_1 = \phi_2$ ya que el frente de ondas llega simultáneamente a S_1 y S_2. Por tanto, los podemos fijar arbitrariamente a cero. Por otro lado, $r_{1,2}$ es el camino que recorre la onda desde $S_{1,2}$ hasta el punto de observación P.

La superposición de estas dos ondas, nos dará la expresión ya conocida,

$$I_t = I_1 + I_2 + 2\sqrt{I_1 I_2}(\vec{e_1}\cdot\vec{e_2})cos(\delta)$$

con $\delta = k(r_1 - r_2) + \phi_1 - \phi_2$

En esta expresión podemos hacer alguna que otra simplificación,

- $\vec{e_1}\cdot\vec{e_2} = 1$ porque las ondas las consideramos polarizadas linealmente en la misma dirección.
- $I_1 = I_2$ en caso de que no haya ningún filtro en S_1 ó S_2.
- $\phi_1 - \phi_2 = 0$ ya que el frente de ondas llega simultáneamente a S_1 y S_2. Nótese que si colocásemos por ejemplo una pieza de un material transparente antes de una de las dos aperturas, tendríamos un desfase en una de las dos ondas y esta diferencia ya no sería nula.

Como vemos, es la diferencia de caminos $r_1 - r_2$ la que determina el valor de la irradiancia final en el punto P. Vamos a calcularla.Según la figura, $\Delta = r_1 - r_2$ lo podemos escribir como $\Delta = asen(\theta)$. Si éste ángulo es pequeño (lo que significa que la distancia entre las fuentes y la pantalla de observación sea grande comparada con la separación entre las fuentes), esta expresión la podemos simplificar,

$$\Delta = asen(\theta) \simeq atan(\theta) = a\frac{x}{D}$$

.

Y por tanto,

$$\delta = k\frac{ax}{D} = \frac{2\pi ax}{\lambda D}$$

En estas expresiones, x es la distancia del punto P de observación al eje mientras que D es la distancia entre el plano que contiene a las fuentes y la pantalla de observación, donde se encuentra P.

Ahora estamos en disposición de contestar a la pregunta que nos planteábamos antes, ¿cómo es la distribución de irradiancia en la pantalla de observación?. Vemos que la irradiancia total posee un término $cos(\frac{2\pi ax}{\lambda D})$ por lo que veremos en la pantalla una distribución cosenoidal, con máximos de irradiancia cuando $\delta = 2m\pi, , m = 0, \pm1, \pm2...$ y mínimos de irradiancia cuando $\delta = (2m+1)\pi, , m = 0, \pm1, \pm2....$ Las posiciones x a las que corresponden estas condiciones serán,

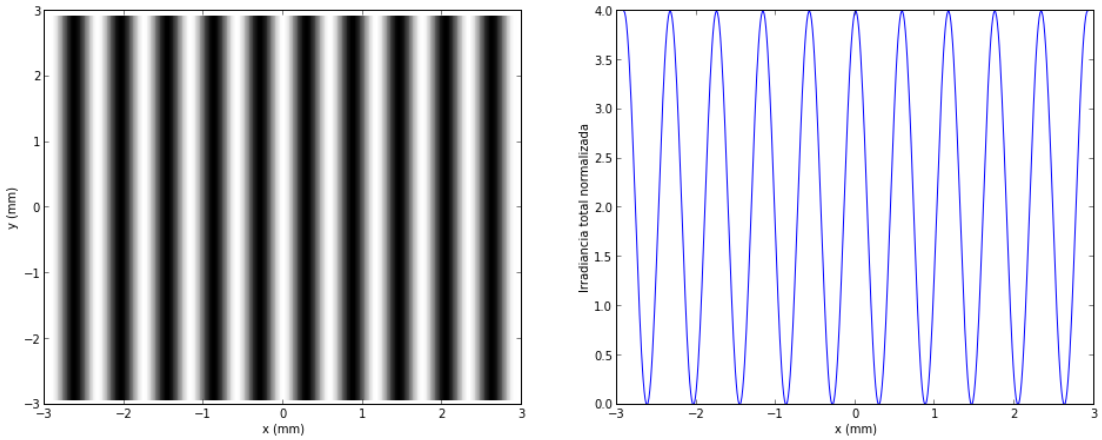
- Máximos de irradiancia. $\delta = 2m\pi \implies \frac{ax}{D} = m \implies x_{max} = \frac{m\lambda D}{a}$
- Mínimos de irradiancia. $\delta = (2m+1)\pi \implies \frac{ax}{D} = (m+1/2) \implies x_{min} = \frac{(m+1/2)\lambda D}{a}$

Si dibujamos en 2D la distribución de irradiancia en la pantalla, obtenemos lo siguiente.

In [7]:
```
%pylab inline
Lambda = 5e-7 # longitud de onda de la radiación de 500 nm
k = 2.0*pi/Lambda
D = 3.5# en metros
a = 0.003 # separación entre fuentes de 3 mm
interfranja = Lambda*D/a
print "interfranja",interfranja*1e3, "(mm)" # muestra el valor de la inte

x = linspace(-5*interfranja,5*interfranja,500)
I1 = 1 # Consideramos irradiancias normalizadas a un cierto valor.
```

```
I2 = 1

X,Y = meshgrid(x,x)
delta = k*a*X/D
Itotal = I1 + I2 + 2.0*sqrt(I1*I2)*cos(delta)

subplot(121)
pcolormesh(x*1e3,x*1e3,Itotal,cmap = 'gray')
xlabel("x (mm)")
ylabel("y (mm)")
subplot(122)
plot(x*1e3,Itotal[x.shape[0]/2,:])
xlabel("x (mm)")
ylabel("Irradiancia total normalizada")
figsize(16,6)

# Calculo del contraste
C = (Itotal.max() - Itotal.min())/(Itotal.max() + Itotal.min())
print "Contraste", C
```

```
interfranja 0.583333333333 (mm)
Contraste 0.999980181859
```



Como podemos ver, el valor de la interfranja (distancia de máximo a máximo o de mínimo a mínimo), es

$$Interfranja = \frac{\lambda D}{a}$$

que en el caso representado, con $a = 3$ mm, $D = 3.5$ m y $\lambda = 500$ nm, toma un valor de 0.58 mm.**Cuestiones**

1. Disminuye el valor de I1 o I2 y observa cómo cambia el patrón de interferencia. ¿Cómo cambia el valor del contraste?
2. Supongamos que colocamos una pieza de un material transparente antes de la apertura S_1. ¿En qué cambiaría el tratamiento dado en el bloque de código mostrado?. Si el material tiene un índice de refracción $n = 1.8$ y un espesor $d = 1$ mm, modifique el código mostrado para mostrar el nuevo patrón de interferencia.

¿Qué diferencia habría si en vez de iluminar con luz monocromática iluminamos con luz blanca?

Podemos verlo en la siguiente imagen

In [51]:
```
from IPython.display import Image
Image(filename="FranjasYoung.png")
```

Out [51]:

Como se puede observar, en el caso de luz blanca, cada una de las longitudes de onda que la componen forma un sistema de franjas con los máximos situados en posiciones distintas y con una interfranja diferente. Esto dificulta enormemente la visualización de la interferencia y nos llevará a definir el concepto de luz coherente e incoherente.

0.1 Para saber un poco más. Otros recursos en la red.

Video de la UNED sobre Thomas Young y su experimento.

In [6]:
```
from IPython.display import YouTubeVideo
YouTubeVideo("B34bAGtQL9A")
```

Out [6]: <IPython.lib.display.YouTubeVideo at 0x987c18c>

Aplicación en java de la Universidad de Barcelona. Se puede modificar la distancia entre el plano que contiene las rendijas y la pantalla, la separación entre las rendijas, etc. Podéis observar cómo cambia el patrón de interferencia al modificar estos parámetros. En general, la serie de aplicaciones sobre óptica es excelente así que merece la pena jugar con ellas.

In [1]:
```
from IPython.core.display import HTML
HTML('<iframe src=http://www.ub.edu/javaoptics/applets/YoungEn.html width
```

Out [1]: <IPython.core.display.HTML at 0xaccfb4c>

Anillos_de_Newton

Oscar Gómez Calderón

November 12, 2013

1 Caracterización de una lente oftálmica mediante la técnica de los Anillos de Newton

Grupo de trabajo

En esta celda los integrantes del grupo: *modificar el texto*

- Juan Antonio Fernández
- Alberto Pérez
- Juan

Incluir las direcciones de correo electrónico

Introducción. Anillos de Newton

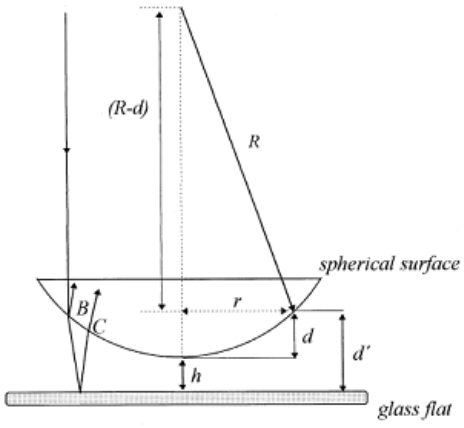
Vamos a determinar el radio de curvatura de una lente plano-convexa empleando la técnica interferométrica conocida como Anillos de Newton. El esquema del montaje experimental se muestra en la siguiente figura donde la superficie esférica de la lente (*spherical surface*) se coloca sobre una superficie plana (*glass flat*). Así se forma entre ambas superficies una lámina de aire de espesor variable. Al iluminar la lente con una haz colimado y monocromático (de longitud de onda λ) aparecen las franjas de interferencias en forma de anillos concéntricos. El radio r de los anillos oscuros es

$r^2 = m\lambda R - 2hR$

donde R es el radio de curvatura de la lente, m indica el número del anillo oscuro, y h considera que el contacto entre las dos superficies no es perfecto.

In [5]:
```
from IPython.core.display import Image,display
Image(filename="SetupAnillosNewton.png")
```

Out [5]:

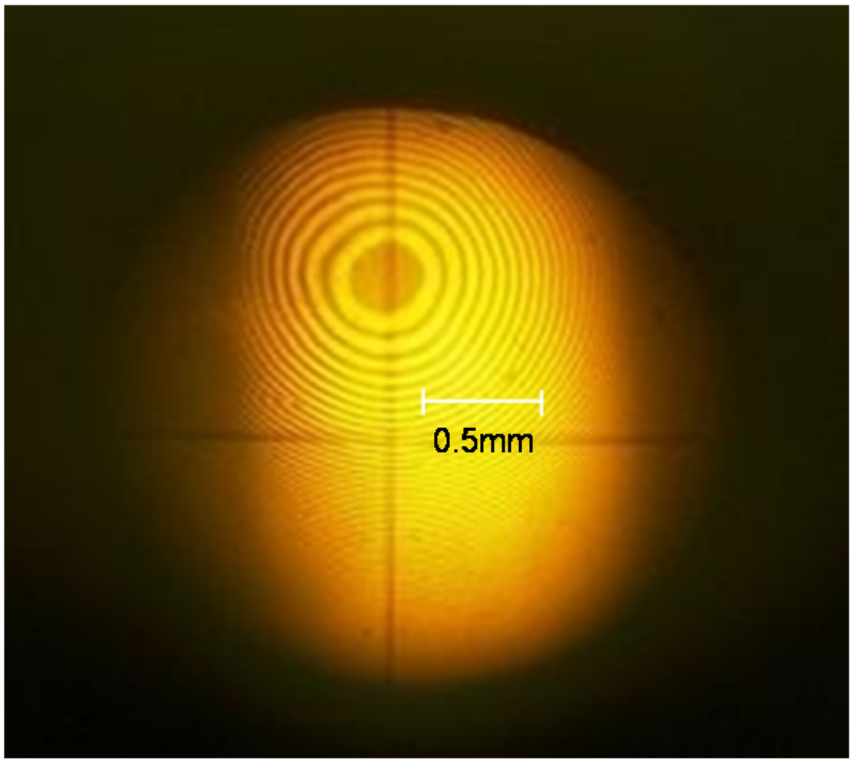


Tarea 1. Patrón de interferencias. Determinar el radio de los anillos oscuros.

En la siguiente figura se puede observar el patrón de interferencias obtenido con nuestra lente oftálmica.

In [2]:
```
from IPython.core.display import Image,display
Image(filename="ImagenAnillosNewton.png")
```

Out [2]:



Empleando dicha imagen vamos a medir el diámetro de los anillos oscuros. Se pueden medir directamente en la pantalla del ordenador o imprimiendo la figura en una hoja de papel. Escribir en la siguiente tabla el diámetro de los anillos oscuros (añadir tantas filas como sean necesarias manteniendo el formato)

Número del anillo | Diámetro del anillo (mm)

1 | 23

2 | 31

3 | 37

4 | 43

Estos diámetros tienen el aumento β correspondiente al sistema óptico de medida y al tamaño de la figura en la pantalla del ordenador o en la hoja de papel. Teniendo en cuenta la escala de referencia que aparece en la figura calcular el aumento de los diámetros medidos $\beta = 46$ (*escribir el valor y el procedimiento empleado*). Esta medida debe realizarse en las mismas condiciones que las de los diámetros. Usando dicho aumento podemos obtener el radio real de los anillos oscuros

Radio = (Diámetro / 2) / β *Escribir el valor*

Escribir otra tabla con los valores finales de los radios de los anillos oscuros

Número | Radio (mm)

1 | 0.25

2 | 0.337

3 | 0.4022

4 | 0.4674

Tarea 2. Análisis de los datos. Ajuste lineal de los radios de los anillos oscuros

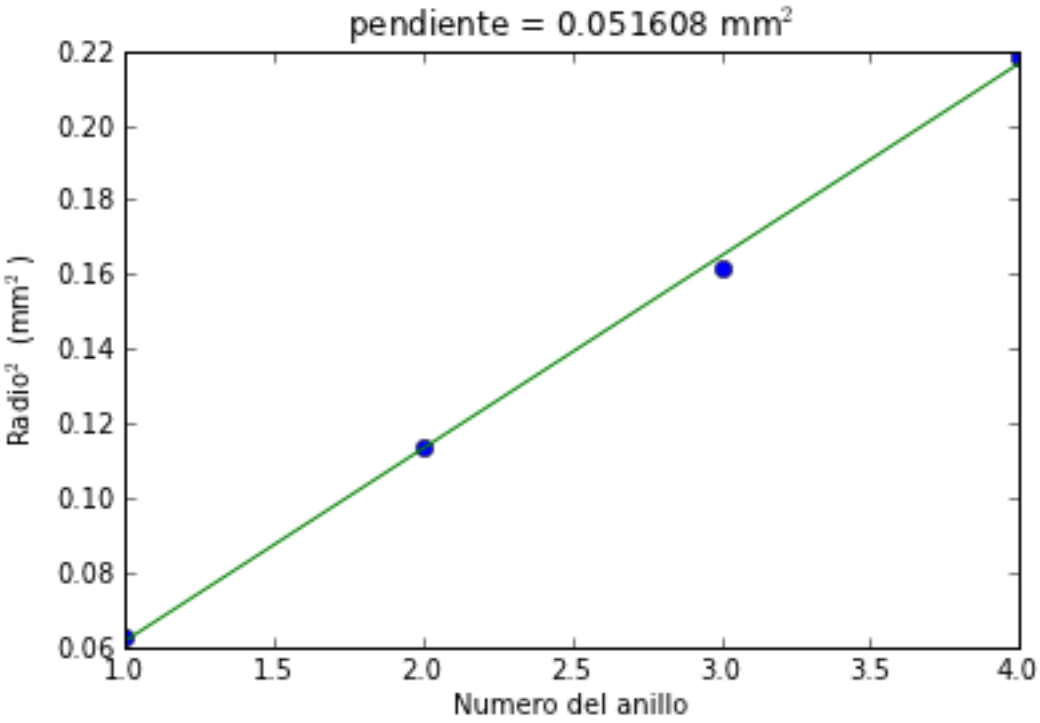
Empleando los radios de los anillos oscuros obtenidos en la **Tarea 1** vamos a representar gráficamente el radio al cuadrado en función del número del anillo oscuro. Dicha representación debería darnos una dependencia lineal cuya pendiente es el radio de curvatura multiplicado por la longitud de onda de la luz empleada en el experimento, es decir,

pendiente = λ R

La luz empleada consiste en un LED cuya longitud de onda se encuentra en λ=650 nm.

En la siguiente celda de código se representan los datos y se realiza el ajuste lineal para obtener el valor de la pendiente (aparece escrito en la figura). Dicho valor tendrá las unidades de los radios al cuadrado, es decir, si los radios de los anillos se introducen en mm, entonces la pendiente tendrá dimensiones de mm^2.

In [15]:
```
%pylab inline
####
# Parámetros a modificar. INICIO
####

radio= array([ 0.25, 0.337, 0.4022, 0.4674 ]) # Incluir los radios de los

####
# Parámetros a modificar. FIN
####

#########################################################################

#####
# Ajuste lineal de los datos
####
m=linspace(1,size(radio),size(radio)) # Vector con los números de los ani
radio2=radio*radio # Vector con los radios de los anillos al cuadrado
a,b = polyfit(m,radio2,1) # Ajuste de los datos a una recta donde a es la

#####
# Dibujamos los datos junto con la recta del ajuste
####
plot(m,radio2,'o',m,a*m+b,'-')
xlabel('Numero del anillo');ylabel('Radio$^2$ (mm$^2$)') # Escribimos los
te = "pendiente = %f mm$^2$" % a;title(te); # Se muestra el valor de la pe
```



A la vista de los datos experimentales y la recta de ajuste comentar *(modificando el texto de este apartado)* si se cumple la ley teórica de los radios de los anillos oscuros.

Empleando el valor de la pendiente y la longitud de onda calcular el radio de curvatura de la lente R= (*modificar este texto*)

TrabajoFiltros

Oscar Gómez Calderón, Eduardo Cabrera Granado

November 11, 2013

1 TRABAJO PROPUESTO SOBRE FILTROS INTERFERENCIALES

Consultar el manual de uso de los cuadernos interactivos (notebooks) que se encuentra disponible en el Campus Virtual

Grupo de trabajo

En esta celda los integrantes del grupo: *modificar el texto*

- Juan Antonio Fernández
- Alberto Pérez
- Juan

Incluir las direcciones de correo electrónico

Introducción

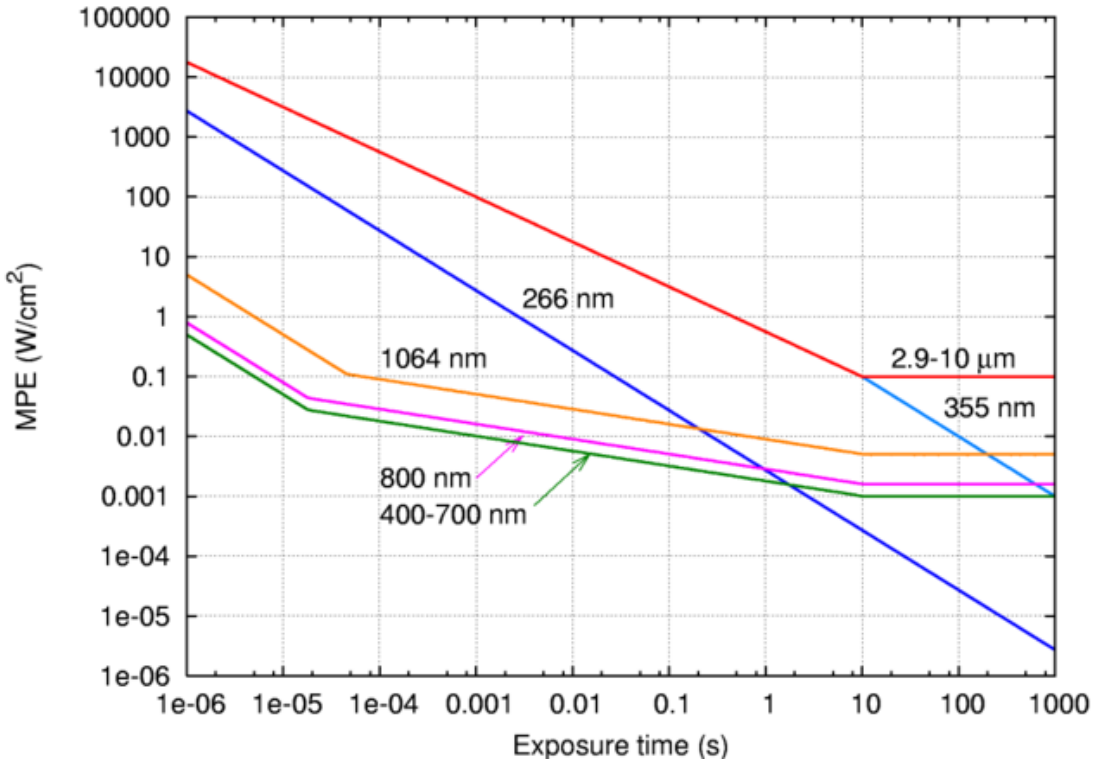
El trabajo consiste en encontrar un filtro interferencial comercial que sirva para proteger el ojo de la radiación visible de un puntero láser de alta potencia. El trabajo se divide en las siguientes tareas:

1.1 Tarea 1. Exposición máxima permisible (MPE)

La exposición máxima permisible (MPE *maximum permissible exposure*) es la máxima densidad de potencia o de energía (W/cm^2 o J/cm^2) de un haz de luz que puede alcanzar el ojo humano sin producir daño. La MPE se mide en la córnea, y depende de la longitud de onda de la radiación y del tiempo de exposición. En la siguiente figura se muestra la MPE en la córnea (en unidades de irradiancia (W/cm^2)) en función del tiempo de exposición para distintos rangos del espectro electromagnético. Figura de http://en.wikipedia.org/wiki/Laser_safety

In [5]:
```
from IPython.core.display import Image
Image("http://upload.wikimedia.org/wikipedia/commons/thumb/2/28/IEC60825_
```

Out [5]:



Tarea 1 (a). Irradiancia máxima

Como estamos considerando el haz láser de un puntero que emite en el visible, como tiempo de exposición emplearemos el tiempo que se tarda en cerrar el párpado. Así con este tiempo de exposición estimar de la gráfica la irradiancia máxima que puede alcanzar el ojo.

Escribir el tiempo de exposición empleado y el correspondiente valor de la irradiancia.

- Tiempo de exposición (parpadeo) = s
- Irradiancia máxima permisible = W/cm^2

Tarea 1 (b). Potencia máxima

Vamos a considerar que el haz que alcanza nuestro ojo está colimado con un tamaño equivalente al de nuestra pupila. Empleando dicho tamaño calcular la potencia máxima que puede alcanzar nuestro ojo sin provocar daño.

Escribir el tamaño de la pupila considerado, las operaciones y el resultado final de la potencia (en mW)

- Diámetro o radio de la pupila = mm
- Cálculos intermedios
- Potencia máxima permisible = mW

1.2 Tarea 2. Elección del puntero láser

Buscar en internet información sobre un puntero láser visible que sea de alta potencia. Verificar que dicho puntero láser puede provocar daño ocular (teniendo en cuenta el resultado de la **Tarea 1 (b)**)

Escribir aquí las características técnicas de dicho láser (potencia, longitud de onda, etc.) y el precio. Incluir referencia sobre la página web http:\

Vamos a incrustar en el notebook la página web empleada. Para ello escribimos la dirección de la página web en la celda de código siguiente.

In [1]:
```
####
# Parámetros a modificar. INICIO
####

web_laser = 'http://www.punterolaser.com' # Incluir la dirección de la pág

web_anchura = '1100' # Valor en pixeles de la anchura de la página web inc

web_altura = '800' # Valor en pixeles de la altura de la página web incrus

####
# Parámetros a modificar. FIN
####

#########################################################################
texto_web_laser='<iframe src= '+web_laser+' width='+web_anchura+'px, heigh
from IPython.display import HTML
HTML(texto_web_laser)
```

Out [1]: <IPython.core.display.HTML at 0xa83c46c>

1.3 Tarea 3. Elección del filtro interferencial

Vamos a buscar en internet un filtro interferencial comercial que permita evitar el riesgo de daño ocular para el puntero láser seleccionado. Se tratará de un filtro que bloquee la longitud de onda del puntero láser.

Tarea 3 (a). Busqueda e información del filtro interferencial

Vamos a emplear la información accesible en la casa **Semrock** (http://www.semrock.com/filters.aspx)

Seleccionar en esta página web un filtro adecuado. Pinchar sobre cada filtro (sobre la curva de transmitancia, sobre el *Part Number*, o sobre *Show Product Detail*) para obtener más información. Escribir aquí las características más relevantes del filtro seleccionado: transmitancia T, absorbancia o densidad óptica OD, rango de longitudes de onda, precio, etc..

Vamos a incrustar en el notebook la página web con la información detallada del filtro seleccionado. Para ello escribimos la dirección de dicha página web en la celda de código siguiente.

In [20]:
```
####
# Parámetros a modificar. INICIO
####

web_filtro = 'http://www.semrock.com/FilterDetails.aspx?id=LP02-224R-25' #

web_anchura = '1100' # Valor en pixeles de la anchura de la página web inc

web_altura = '800' # Valor en pixeles de la altura de la página web incrus

####
# Parámetros a modificar. FIN
####

#########################################################################
```

```
texto_web_filtro='<iframe src= '+web_filtro+' width='+web_anchura+'px, hei
from IPython.display import HTML
HTML(texto_web_filtro)
```

Out [20]: <IPython.core.display.HTML at 0xc37b6ec>

Tarea 3 (b). Verificación del filtro

Empleando el dato de la transmitancia (T) a la longitud de onda del puntero láser comprobar que dicho filtro evitará el riesgo de lesión.

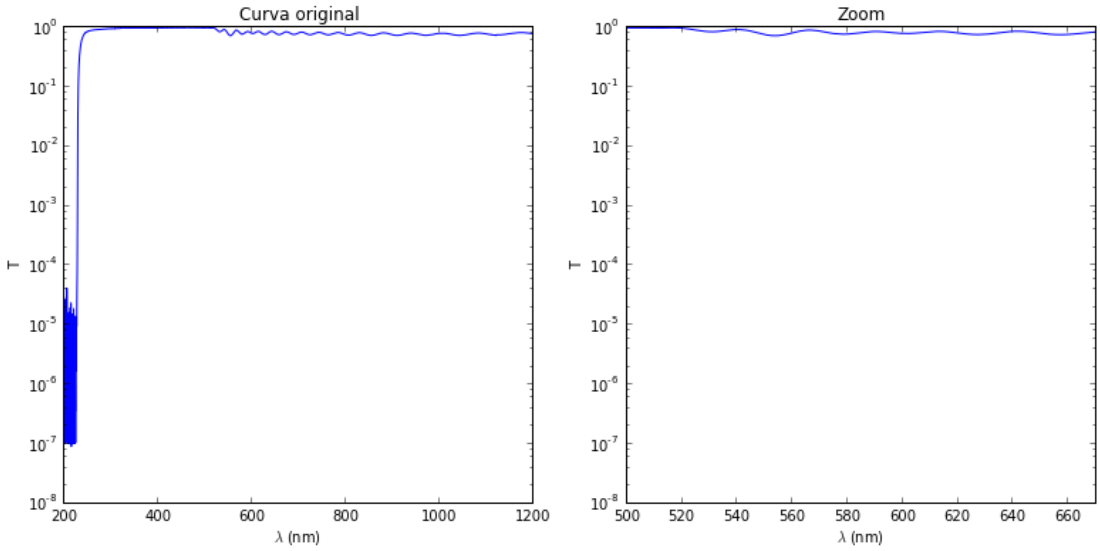
Para ello vamos a usar los datos de la transmitancia del filtro seleccionado que aparecen en la página web de Semrock. Para cargar dichos datos en nuestro notebook se emplea el código que identifica a dicho filtro y que se obtiene automáticamente de la dirección de la página web seleccionada en el apartado anterior (**Tarea 3(a)**).

En la siguiente celda de código se representa la transmitancia del filtro en escala logarítmica en función de la longitud de onda (en nm). A la izquierda se muestra la curva original obtenida de Semrock y a la derecha se muestra un zoom de la misma en la región que nosotros elijamos.

Esta gráfica nos permite obtener el valor de la transmitancia a la longitud de onda de nuestro puntero láser, por ello debemos escribir el valor de dicha longitud de onda en el código, y el se encargará de calcular la transmitancia. El resultado aparece en la parte superior de las gráficas.

In [22]:
```
%pylab inline
####
# Parámetros a modificar. INICIO
####

longitud_de_onda_laser = 530 # Incluir el valor de la longitud de onda del

# Pintamos la gráfica original y un zoom empleando el rango de valores sig
longitud_de_onda_minina = 500 # Incluir el valor de la longitud de onda mi
longitud_de_onda_maxima = 670 # Incluir el valor de la longitud de onda ma
transmitancia_minina = 1e-8 # Incluir el valor de la transmitancia mímina
transmitancia_maxima = 1 # Incluir el valor de la transmitancia máxima pa

####
# Parámetros a modificar. FIN
####

#########################################################################
from numpy.core.defchararray import find
indice_igual=find(web_filtro,'=')
codigoID = web_filtro[indice_igual+1:-3]
Codigo_Filtro = codigoID
filename = 'http://www.semrock.com/_ProductData/Spectra/'+Codigo_Filtro+'_
data=genfromtxt(filename,dtype=float,skip_header=4) # Carga los datos
longitud_de_onda=data[:,0];transmitancia=data[:,1];
figure(figsize(13,6))
subplot(1,2,1)
semilogy(longitud_de_onda,transmitancia)
xlabel('$\lambda$ (nm)');ylabel('T');title('Curva original')
subplot(1,2,2)
semilogy(longitud_de_onda,transmitancia)
xlabel('$\lambda$ (nm)');ylabel('T');title('Zoom')
axis([longitud_de_onda_minina, longitud_de_onda_maxima, transmitancia_mini

from scipy.interpolate import interp1d
f_transm = interp1d(data[:,0],data[:,1])
transm_para_laser = f_transm(longitud_de_onda_laser)
print "Transmitancia para la longitud de onda del puntero láser"
print transm_para_laser
```

```
Transmitancia para la longitud de onda del puntero láser
0.84435
```



Empleando el valor de la transmitancia del filtro a la longitud de onda del puntero láser verificar que el filtro evitará el riesgo de daño ocular. Escribir aquí la estimación realizada.

Tubo de Quincke-ALUMNOS

Sonia Melle Hernández

September 27, 2013

Part I

TUBO DE QUINCKE: INTERFERENCIA DE ONDAS SONORAS

NOMBRE ALUMNO:

CORREO ELECTRÓNICO: _____

1 INTRODUCCIÓN

Un foco F_1 emite una onda armónica de *amplitud* A y *frecuencia* ω. Esta onda se propaga con *velocidad*ˆ $v = \omega/k$, donde k es el *número de onda* definido como $k = 2\pi/\lambda$, siendo λ la *longitud de onda*. Consideramos que la onda sale del foco con un *desfase inicial* δ_0. La función de ondas de esta onda armónica viene dada por:

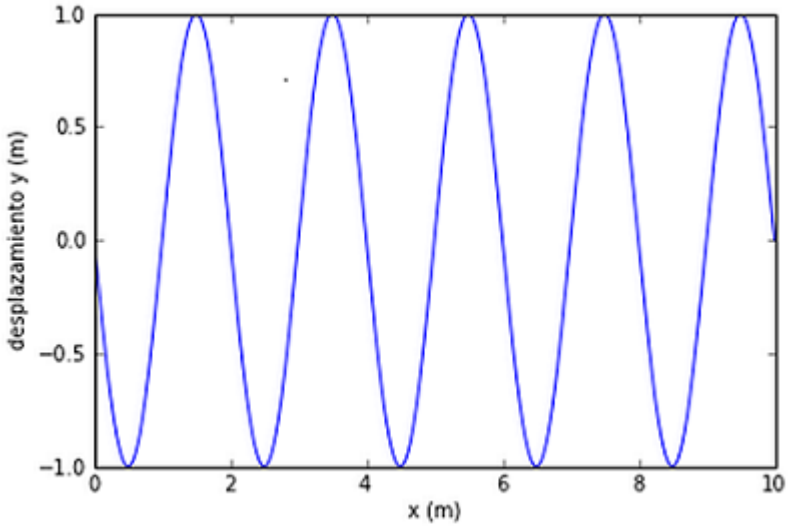
$$y(x,t) = Asin(kx - \omega t + \delta_0)$$

1.1 A) ONDA ARMÓNICA

Supongamos que la frecuencia de la onda es de 170 Hz y su perfil espacial en el instante t = 0 es el representado a continuación:

In [2]:
```
from IPython.display import Image
Image(filename="C:\USUARIOS\NotebooksIPython\onda_armonica.png")
```

Out [2]:



PREGUNTA 1: Calcular la velocidad de propagacón de la onda = m/sPREGUNTA 2: Escribir la expresión general de la función de onda

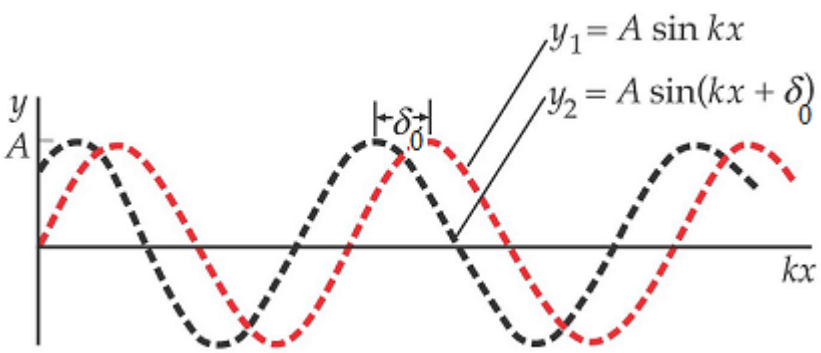
$$y(x,t) =$$

1.2 B) INTERFERENCIA DE DOS ONDAS ARMÓNICAS

1.3 B1) ONDAS QUE RECORREN EL MISMO CAMINO

In [3]:
```
from IPython.display import Image
Image(filename="C:\USUARIOS\NotebooksIPython\interferencia_focos_mismo_pur
```

Out [3]:



Consideramos la interferencia de dos ondas armónicas emitidas por dos focos, F_1 y F_2, situados en el mismo punto del espacio. Consideramos que estas ondas tienen igual amplitud A y frecuencia ω y se propagan con la misma velocidad $v = \omega/k$. Consideramos además que la onda que emite F_2 sale con una diferencia de fase δ_0 respecto a la que emite F_1. La función de ondas de estas ondas vendrá dada por:

$$y_1(x,t) = Asin(kx - \omega t)$$
$$y_2(x,t) = Asin(kx - \omega t + \delta_0)$$

Ambas se propagan en la misma dirección de forma que interfieren en todos los puntos del espacio. La superposición de ambas dará lugar a una onda cuya expresión vendrá dada por:

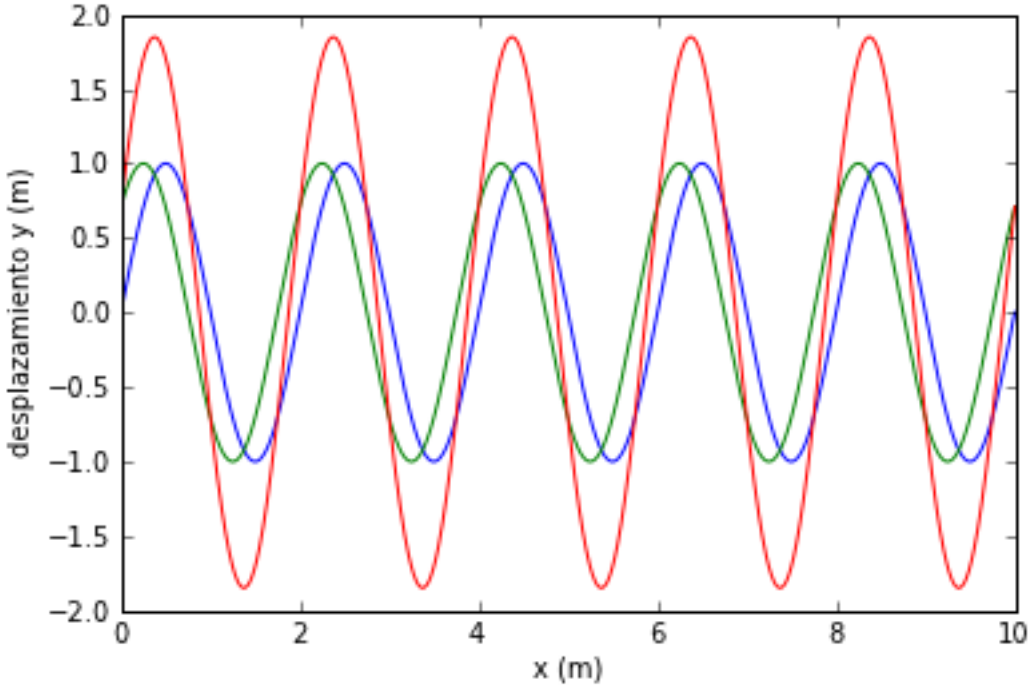
$$y(x,t) = 2Acos(\delta_0/2)sin[kx - \omega t + \delta_0/2]$$

de forma que la *amplitud de la onda resultante* de la interferencia $A_R = 2Acos(\delta_0/2)$, tomará distintos valores dependiendo del valor del desfase inicial δ_0. Así:

- si $\delta_0 = 2m\pi, , m = 0, \pm1, \pm2, ..., \implies$ amplitud máxima $A_R = \pm 2A$, $\implies$ *interferencia constructiva.*
- si $\delta_0 = (2m+1)\pi, , m = 0, \pm1, \pm2, ..., \implies$ amplitud nula $A_R = 0$, $\implies$ *interferencia destructiva.*

Representamos a continuación una foto instantánea de las ondas que interfieren (curvas verde y azul) y la onda resultante de dicha interferencia (curva roja).

In [15]:
```
##############################
#MAGNITUD CUYO VALOR QUEREMOS VARIAR
delta0=                                          #desfase incial de la onda y2
##############################
x=linspace(0,10,1000)
Lambda=2                                         #longitud de onda (m)
k=2*pi/Lambda                                    #número de onda (m^-1)
A=1                                              #amplitud (m)
y1=A*sin(k*x)                                    #función de onda de la onda y
y2=A*sin(k*x+delta0)                             #función de onda de la onda y
plot(x,y1)
plot(x,y2)
xlabel("x (m)")
ylabel("desplazamiento y (m)")
delta=delta0                                      #desfase final
AR=2*A*cos(delta0/2)                              #amplitud de la onda resulta
y=AR*sin(k*x+delta0/2)                            #función de onda de la onda
plot(x,y);
```



PREGUNTA 3: Variar en el script anterior el valor del desfase incial entre ambas ondas y, a partir de la figura, obtener la amplitud de la onda resultante en los siguientes casos:

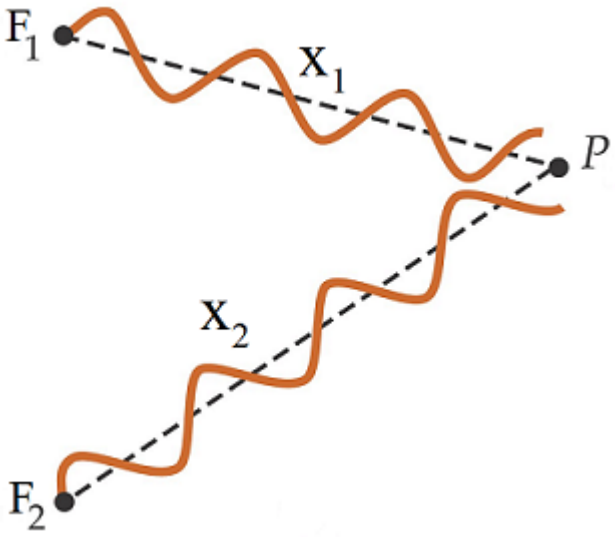
- **cuando ambas ondas salen en oposición de fase** ($\delta_0 = \pi$) $\implies$ A_R **=**
- **cuando ambas ondas están inicialmente en fase** ($\delta_0 = 0$) $\implies$ A_R **=**
- **cuando ambas ondas salen en cuadratura** ($\delta_0 = \pi/2$) $\implies$ A_R **=**
- **cuando el desfase inicial entre ambas ondas es** $\delta_0 = \pi/4$ $\implies$ A_R **=**

1.4 B2) ONDAS QUE RECORREN DISTINTO CAMINO

Consideramos ahora dos ondas que recorren distinta distancia x_1 y x_2 hasta llegar al punto P donde interfieren. Llamamos x_1 a la distancia desde F_1 al punto P y x_2 a la distancia desde F_2 al punto P (ver Figura).

In [4]:
```
from IPython.display import Image
Image(filename="C:\USUARIOS\NotebooksIPython\interferencia.png")
```

Out [4]:



Escribimos las funciones de ondas de las ondas emitidas por ambos focos en el punto donde interfieren, punto P, teniendo en cuenta que la onda y_2 se emite con una diferencia de fase incial δ_0 respecto de la onda y_1 y que la distancia que recorre y_2 es mayor que la que recorre y_1 (es decir, $x_2 > x_1$):

$$y_1(x_P, t) = Asin(kx_1 - \omega t)$$
$$y_2(x_P, t) = Asin(kx_2 - \omega t + \delta_0)$$

La superposición de estas dos ondas en el punto P vendrá dada por la expresión:

$$y(x_P, t) = 2Acos(\delta/2)sin[(k(x_1 + x_2)/2) - \omega t + \delta_0/2]$$

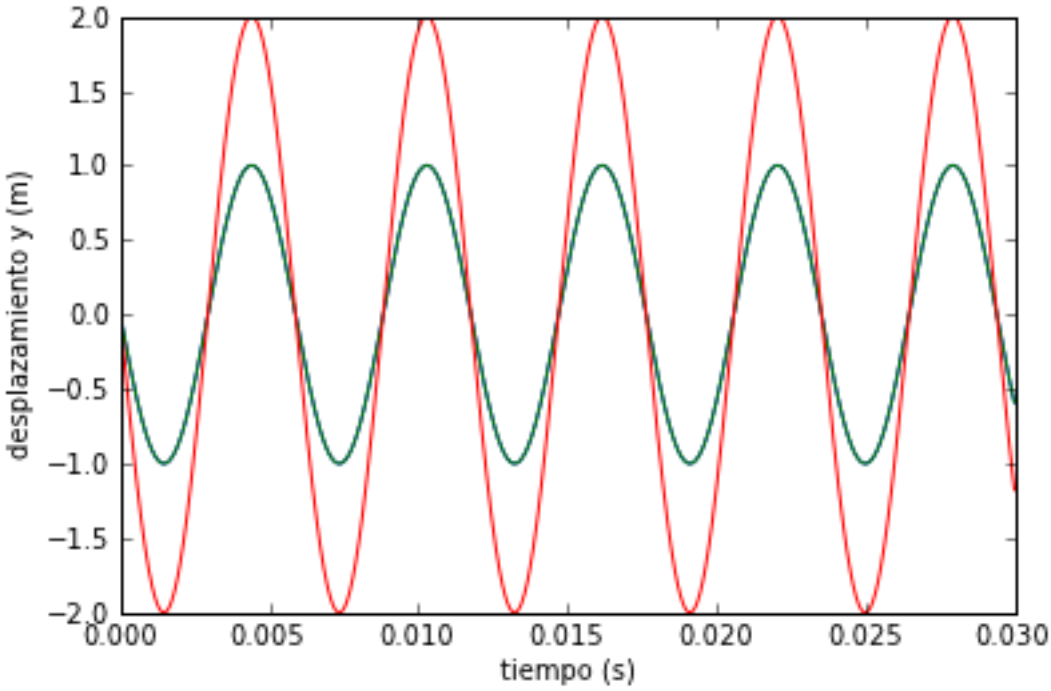
donde $\delta = k(x_2 - x_1) + \delta_0$, es el desfase final, suma del desfase inicial entre ambas ondas δ_0 y el desfase acumulado por la diferencia entre las distancias que recorren ambas ondas desde sus focos hasta el punto donde interfieren (distancia que se conoce como *diferencia de camino* $\Delta x = x_2 - x_1$;) . El valor de la amplitud de la onda resultante en el punto P, que viene dada por $A_R = 2Acos(\delta/2)$;, dependerá por tanto del valor del *desfase final* δ. Así:

- si $\delta = 2m\pi, , m = 0, \pm1, \pm2, ...,$ $\implies$ amplitud máxima $A_R = \pm 2A$, $\implies$ *interferencia constructiva.*
- si $\delta = (2m + 1)\pi, , m = 0, \pm1, \pm2, ...,$ $\implies$ amplitud nula $A_R = 0$, $\implies$ *interferencia destructiva.*

Representamos la evolución temporal de las ondas que interfieren en el punto P donde se produce la interfrencia (curvas verde y azul) y la evolución temporal de la onda resultante de la interferencia de ambas en el mismo punto P (curva roja).

In [18]:
```
##############################
#MAGNITUDES CUYO VALOR QUEREMOS VARIAR
delta0=                                    #desfase incial de la onda y2 r
Deltax=                                    #diferencia de camino entre la
##############################
t=linspace(0,0.03,1000)
x1=20                                       #camino que recorre la onda y1
x2=Deltax+x1                                #camino que recorre la onda y2
Lambda=2                                    #longitud de onda (m)
k=2*pi/Lambda                               #número de onda (m^-1)
A=1                                         #amplitud de cada onda viajera
```

```
f=170                                          #frecuencia de las ondas viaje
y1=A*sin(k*x1-2*pi*f*t)                        #función de onda de la onda y1
y2=A*sin(k*x2-2*pi*f*t+delta0)                 #función de onda de la onda y2
plot(t,y1)
xlabel("tiempo (s)")
ylabel("desplazamiento y (m)")
plot(t,y2)
delta=k*Deltax+delta0                          #defase final (rad)
AR=2*A*cos(delta/2)                            #amplitud de la onda resultant
y=AR*sin(k*(x2+x1)/2-2*pi*f*t+delta0/2)        #función de onda de la onda re
plot(t,y);
```



PREGUNTA 4: Para un desfase incial nulo, variar en el script anterior la diferencia de camino Δx con el fin de obtener interferencia constructiva o destructiva en el punto P. Anotar los valores elegidos y razonar el resultado obtenido:

- $\Delta x =$ $\Longrightarrow$ **Interferencia constructiva**
- $\Delta x =$ $\Longrightarrow$ **Interferencia destructiva**

PREGUNTA 5: Si el desfase inicial es $\pi/2$, elegir un valor de Δx para el que la interferencia sea constructiva y otro para el que la interferencia sea destructiva. Anotar los valores elegidos y razonar el resultado obtenido:

- $\Delta x =$ $\Longrightarrow$ **Interferencia constructiva**
- $\Delta x =$ $\Longrightarrow$ **Interferencia destructiva**

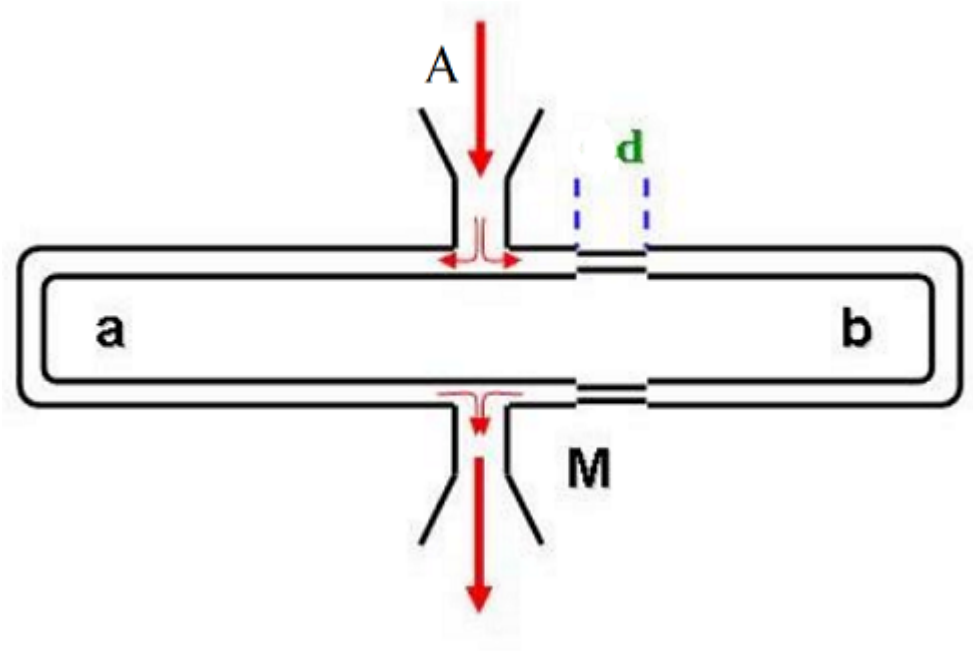
---

2 TUBO DE QUINCKE: INTERFERENCIA DE ONDAS SONORAS. VELOCIDAD DEL SONIDO

Una onda de sonido es una onda mecánica longitudinal cuya frecuencia entra dentro del rango audible por el oído humano, entre 20 Hz (sonidos graves) y 20 kHz (sonidos agudos).Con el fin de determinar la velocidad del sonido en un medio, vamos a utilizar un dispositivo sencillo denominado Tubo de Quincke y teniendo en cuenta el fenómeno de interferencia de ondas sonoras.

In [5]:
```
from IPython.display import Image
Image(filename="C:\USUARIOS\NotebooksIPython\TuboQuincke.png")
```

Out [5]:



El tubo de Quincke está formado por dos tubos en forma de U (tubos a y b en la figura). Estos tubos tienen dos extremos abiertos (A y M en la figura). En el extremo A se sitúa un altavoz que genera una onda de sonido, y en el extremo M se sitúa un micrófono que registra el sonido que le llega. Hacemos vibrar el diafragma del altavoz con una frecuencia y amplitud conocidas. Esta vibración se transmitirá a las moléculas de aire próximas al diafragma generándose una onda armónica de sonido que viajará por ambos brazos en sentido contrario de forma que el sonido recogido por el micrófono será la interferencia de ambas ondas en el punto donde se sitúa el micrófono. El tubo a es un tubo de longitud fija, mientras que el b es un tubo deslizante. Modificando la longitud del tubo deslizante, se modifica el camino recorrido por una de las ondas, y con él, el desfase final entre las dos perturbaciones en el punto donde está el micrófono. Así, si desplazamos hacia la derecha el tubo b una distancia d, aumentamos la longitud del camino recorrido por la onda que se desplaza por este tubo en una distancia $2d$ respecto a la que se deplaza por el tubo a, es decir, $x_2 = x_1 + 2d$. Por tanto, modificando la longitud del tubo deslizante, escucharemos máximos (interferencia constructiva), y mínimos (interferencia destructiva) de sonido según sea el valor del desfase total. En nuestros cálculos vamos a considerar que la amplitud de ambas ondas y_1 e y_2 es la misma. Esto es una aproximación pues sabemos que a medida que aumente la longitud del tubo b la amplitud de la onda sonora que se propaga por dicho tubo disminuirá.

Este experimento es parecido al experimento de Young en Óptica, en donde lo que interfieren son dos ondas electromagnéticas procedentes de dos fuentes puntuales y la interferencia de ambas se registra en una pantalla. En ese caso, conociendo la distancia entre los máximos o mínimos de intensidad, puede extraerse información de la frecuencia o longitud de onda de la luz.

2.1 A) ONDA ESTACIONARIA

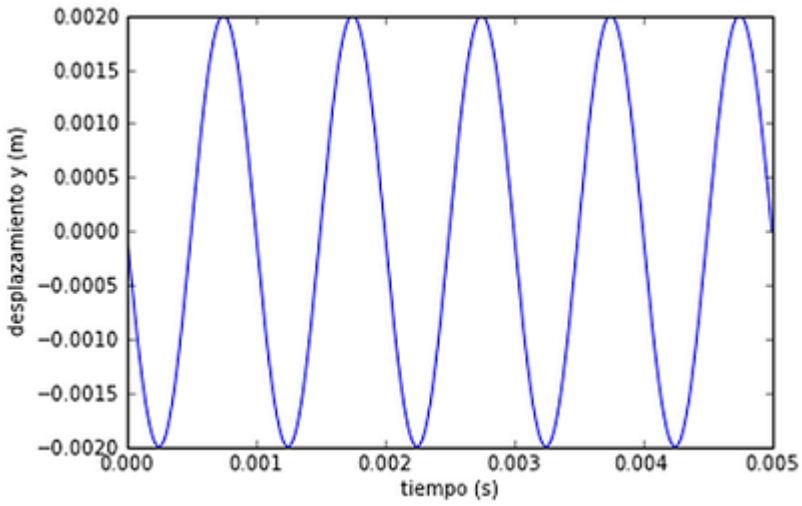
La onda de sonido generada por el altavoz viajará por ambos tubos en sentido contrario. Vamos a considerar que el micrófono está en el punto $x = 0$; y vamos a escribir las funciones de onda de las ondas que viajan por ambos tubos de la forma:

$$y_1(x,t) = Asin(kx - \omega t)$$
$$y_2(x,t) = Asin(-kx - \omega t + \delta)$$

donde x es la distancia al micrófono y δ la diferencia de fase acumulada por la onda y_2 debida a la diferencia de camino entre ambas ondas , $\delta = k\Delta x = k(x_2 - x_1)$; .La figura representa la evolución temporal de la onda y_1 cuando llega al micrófono (punto de interferencia), es decir, en $x = 0$.

In [6]:
```
from IPython.display import Image
Image(filename="C:\USUARIOS\NotebooksIPython\onda_altavoz.png")
```

Out [6]:



PREGUNTA 6: A la vista de la figura , y teniendo en cuenta que la onda se propaga a 340 m/s, escribir la función de onda para la onda de sonido generada por el altavoz en cualquier punto x del recorrido, tomando como origen de distancias la posición del micrófono:

$$y_1(x,t) =$$

PREGUNTA 7: Considerar una molécula en un punto dentro del tubo a situado a 2 cm del micrófono. Hallar la velocidad transversal y longitudinal máximas de esa molécula. Comentar el resultado.

- $v_{tranv-max}$ **=**
- $v_{long-max}$ **=**

Como hemos dicho, la interferencia de ambas ondas dará lugar a una *onda estacionaria*. En la siguiente animación están representadas las ondas viajeras en azul y rojo y la onda estacionaria que generan en negro.

In [4]:
```
from IPython.core.display import HTML
HTML('<iframe src=https://upload.wikimedia.org/wikipedia/commons/7/7d/Star
```

Out [4]: <IPython.core.display.HTML at 0x4a19350>

PREGUNTA 8: Teniendo en cuenta las expresiones de y_1 e y_2, escribir la función de ondas de la onda resultante de la interferencia de las dos ondas que llegan al micrófono en función de la distancia d que desplazamos el tubo corredizo y de la distancia x al micrófono.

$$y(x,t) =$$

Podéis consultar el desplazamiento de las moléculas de aire en el applet al final de la siguiente página web. En este caso el altavoz está colocado en el lado inferior del tubo de Quincke de forma que el movimiento de un pequeño émbolo representa las oscilaciones de la membrana del altavoz. El micrófono está situado en el lado superior del tubo, de forma que la oscilación del émbolo representa las oscilaciones debidas a la interferencia de ambas ondas. En la parte central del applet, se representa el movimiento armónico que viaja hasta la posición del micrófono por el tubo izquierdo (en color rojo) y por el tubo derecho (en color azul). Cuando la onda se propaga por el tubo, las moléculas de aire vibran. Los puntos de color rojo y azul en los tubos, representan estas moléculas.

In [20]:
```
from IPython.core.display import HTML
HTML('<iframe src=http://www.sc.ehu.es/sbweb/fisica/ondas/quincke/quincke.
```

Out [20]: <IPython.core.display.HTML at 0x4c03230>

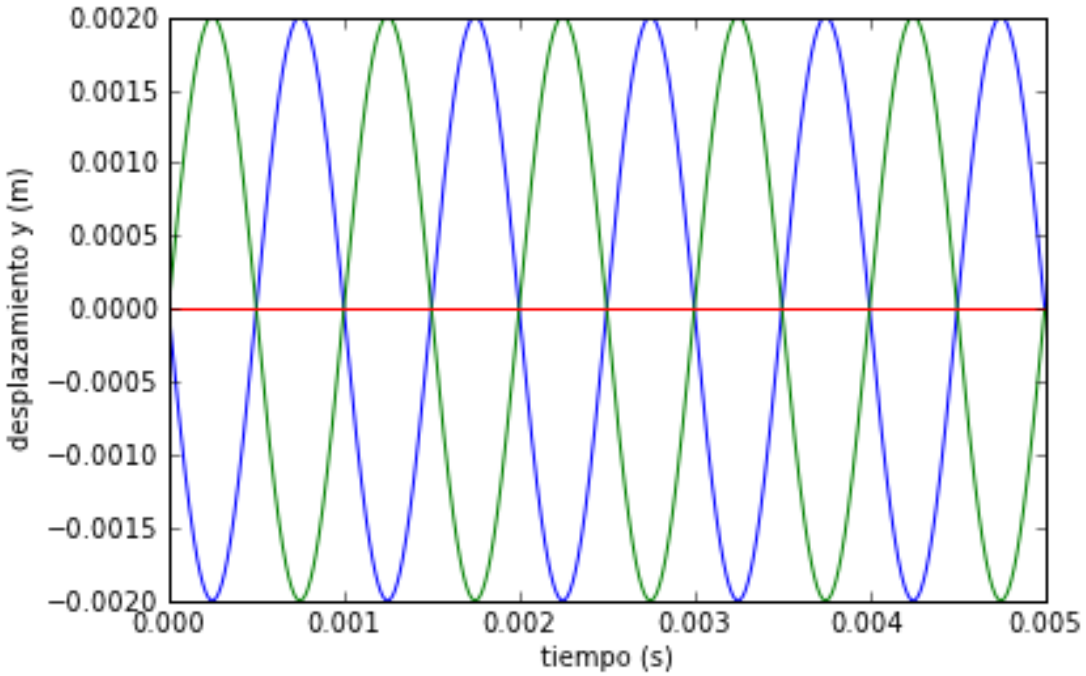
PREGUNTA 9: Una vez fijada la frecuencia actuando en la barra de desplazamiento titulada *Frecuencia*, pulsar el botón *Inicio*. A continuación desplazar el tubo corredizo una distancia determinada hasta conseguir que las ondas viajeras que aparecen en la zona central del applet lleguen al micrófono en oposición de fase, es decir, que el máximo de una coincida con mínimo de la otra. Para desplazar el tubo corredizo hay que arrastrar con el puntero del ratón el círculo de color rojo, situado en la regla (parte inferior derecha del applet). Pulsar el botón *Empieza* y comprobar que, en este caso, no hay desplazamiento del émbolo de presión situado en el lado superior, es decir, no se oirá nada en el micrófono. A continuación, variar la distancia que desplaza el tubo corredizo hasta conseguir que ambas ondas lleguen al micrófono en fase (máximo de una coincidiendo con máximo de la otra). Pulsar el botón *Empieza* y comprobar que la amplitud sonora será máxima (máximo desplazamiento del émbolo en el punto donde está el micrófono). Anotar dos posiciones consecutivas del brazo deslizante (en cm) en la regla para las cuales la amplitud de las oscilaciones del émbolo en la parte superior del applet, que representa el micrófono, sea máxima. Comprobar que relación existe entre la distancia entre dos máximos consecutivos y longitud de onda. Para esta pregunta supondremos que la velocidad del sonido en las condiciones del experimento es de 340 m/s.

- **Distancia 1** $d_1 =$
- **Distancia 2** $d_2 =$
- **Diferencia de distancia=** $d_2 - d_1$
- **Longitud de onda=**
- **Relación entre diferencia de distancia y longitud de onda=**

La figura representa la evolución temporal de la onda resultante de la interferencia de las dos ondas que llegan al micrófono en el punto donde está el micrófono (x = 0);. Recordar que en el tubo de Quincke la diferencia de camino entre y_1 e y_2 es $\Delta x = 2d$.

In [27]:
```
##############################
#MAGNITUD CUYO VALOR QUEREMOS VARIAR
d=                                  #distancia que desplazo el brazo d
##############################
t=linspace(0,0.005,1000)
f=1000                              #frecuencia emitida por el altavoz
v=340                               #velocidad del sonido en el aire a
Lambda=v/f                          #longitud de onda
```

```
k=2*pi*f/v                                  #número de onda
x= 0                                        #distancia al micrófono
Deltax=2*d                                  #diferencia de camino que recorre
A=0.002                                     #amplitud de la onda de sonido em
delta=k*(Deltax)                            #desfase debido a diferencia de c
y1=A*sin(k*x-2*pi*f*t)
y2=A*sin(-k*x-2*pi*f*t+delta)
plot(t,y1)
xlabel("tiempo (s)")
ylabel("desplazamiento y (m)")
plot(t,y2)
AR=2*A*cos(delta/2-k*x)                     #amplitud de la onda resultante d
y=AR*sin(-2*pi*f*t+delta/2)                 #función de onda de la onda result
plot(t,y);
```



PREGUNTA 10: Variar el valor de d y anotar dos valores para los que la amplitud de la onda resultante sea constructiva y destructiva. Justificar el resultado obtenido.

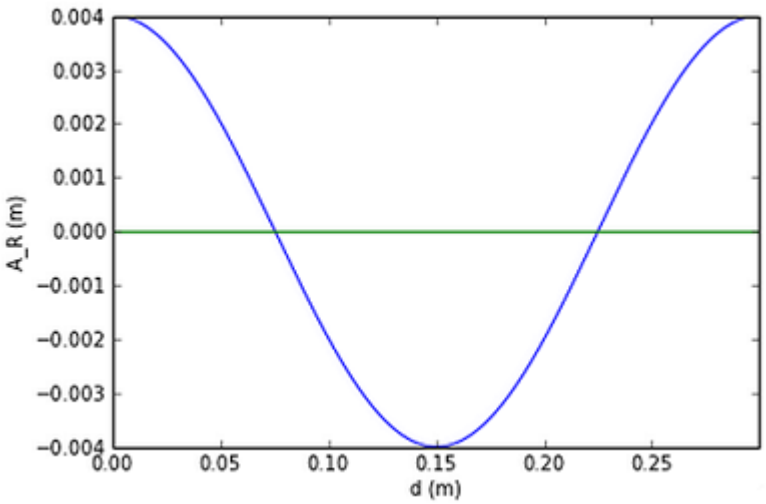
- **Interferencia destructiva $\Longrightarrow$ d = metros**
- **Interferencia constructiva $\Longrightarrow$ d = metros**

2.2 B) VELOCIDAD DEL SONIDO

Supongamos ahora que variamos la temperatura del tubo y consideramos una frecuencia de la onda sinusoidal que genera el altavoz de f= 1.16 kHz. Medimos la distancia entre los mínimos de amplitud sonora que recoge el micrófono y obtenemos la siguiente gráfica:

In [7]:
```
from IPython.display import Image
Image(filename="C:\USUARIOS\NotebooksIPython\distancia_entre_minimos.png")
```

Out [7]:



PREGUNTA 11: A la vista de la gráfica, calcular la velocidad del sonido en este caso particular. Explicar cómo se ha obtenido. v **= m/s.PREGUNTA 12: Consultar (por ejemplo en las transparencias de clase) la expresión de la velocidad del sonido en el aire. Escribirla a continuación. Hallar la temperatura a la que se realizó el experimento.** T **= KPREGUNTA 13: Si enfriamos el tubo 20 ºC ¿Cuál sería la distancia entre los mínimos de amplitud?** d **= cm**

2.3 INFORMACIÓN ADICIONAL

Una forma sencilla de fabricar el tubo puede encontarse en el enlace siguiente:

In [65]:
```
from IPython.core.display import HTML
HTML('<iframe src=http://sudandolagotagorda.blogspot.com.es/2010/06/tubo-
```

Out [65]: <IPython.core.display.HTML at 0x4d7c670>

EstadosElectronicosCadenaFinitaPozos

Elena Díaz

November 11, 2013

Part I

Estados electrónicos de una cadena finita de pozos idénticos

En este documento se define el Hamiltoniano asociado a una cadena finita de NW pozos idénticos por discretización y diagonalización del sistema.

Se puede modificar el número de pozos, la anchura de pozos y barreras y el valor del potencial en ambas regiones.

Los autoestados y autoenergías se ordenan en orden creciente de energía desde level= 0,1,2,3,. . .

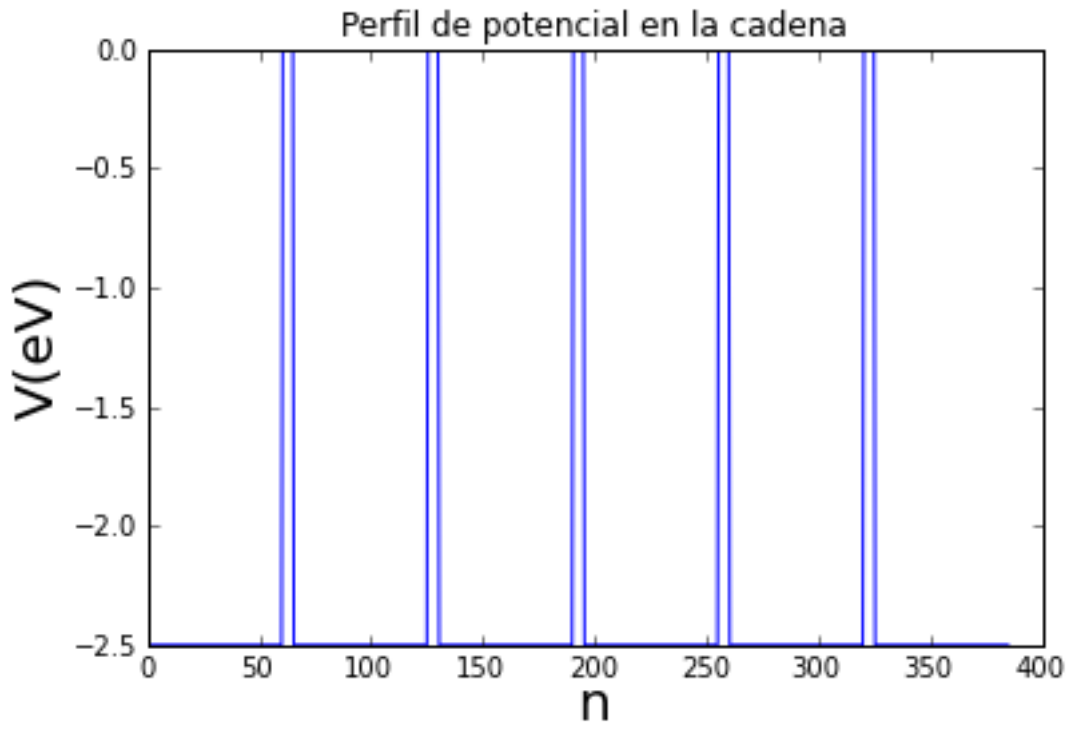
Modificando el valor de la variable level el programa presenta en pantalla el valor de la energía de dicho estado y su amplitud de probabilidad a lo largo de la cadena.

In [7]:
```
%pylab inline

a = 60. #60well width A (Amstrong)
Va = -2.5 #eV well potential
b = 5. #10barrier witdh A
Vb = 0. #eV barrier potential
NW = 6; #number of wells
NB = NW - 1; #number of barriers
factor = 0.0176; #2m/hbar2 ev^-1 A^-2
L = NW*a + NB*b #total length of the system
s=0.5 #length step A
n = L/0.5 #number of sites
N=int(n)
na = a/s #number of sites in a well
nb = b/s #number of sites in a barrier

# Definición y representación del potencial creado por la cadena de pozos
i=linspace(1,N,N)
x=s*i
VW=Va*ones(na)
VB=Vb*ones(nb)
V=VW
for j in range(0, NB):
    V=concatenate((V,VB,VW),axis=0)

plot(x,V)
```

```
xlabel('n',fontsize=20)
ylabel('V(eV)',fontsize=20)
title('Perfil de potencial en la cadena')
```

```
Welcome to pylab, a matplotlib-based Python environment [backend:
module://IPython.zmq.pylab.backend_inline].
For more information, type 'help(pylab)'.
```

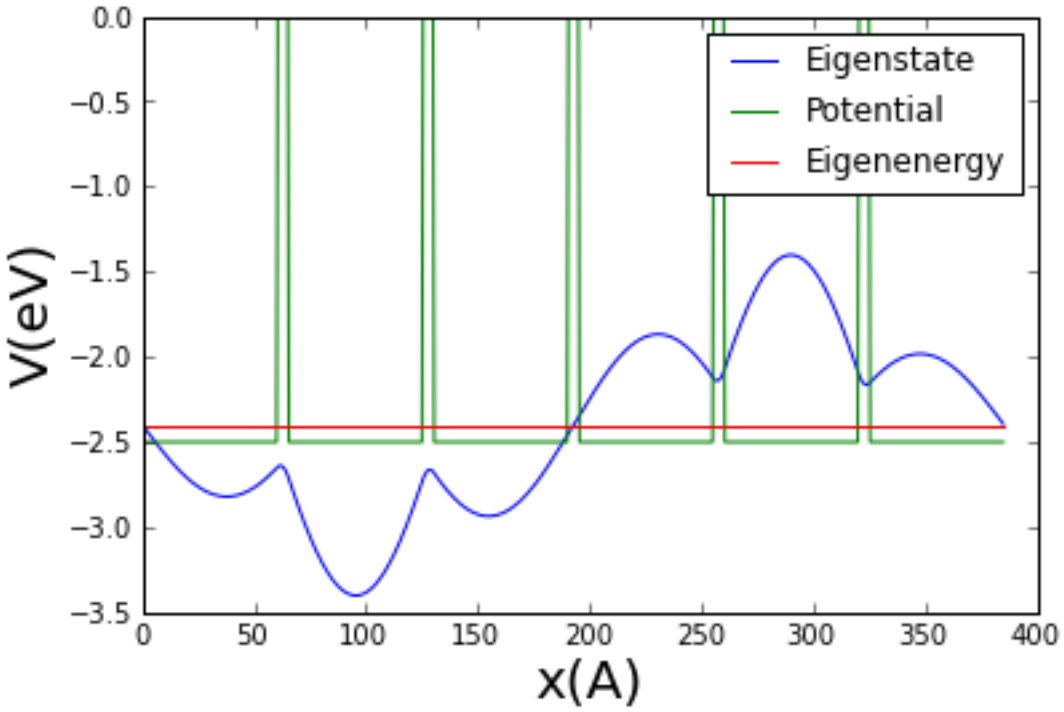
Out [7]: <matplotlib.text.Text at 0x384ab50>



In [13]:
```
# Definición de la matriz del Hamiltoniano del sistema
DiagElements=2. + factor*V*s*s
M=identity(N)*DiagElements

for j in range (0,N):
    for k in range (0,N):
        if (j==(k+1) or j==(k-1)):
            M[j,k]=-1.

# Solución del problema de autovalores y autovectores del Hamiltoniano
a,w = eig(M)
e=a/(factor*s*s)
E=empty(N)
W=empty([N,N])
index=argsort(e)
for j in range(0,N):
    E[j]=e[index[j]]

for j in range(0, N):
    for k in range(0,N):
        W[j,k]=w[k,index[j]]
# Los autovalores y autovectores del Hamiltoniano ordenados según la ener
```

```
level=1 # Nivel de autovalor y autovector que se dibuja
plot(x, E[level] + W[level]/amax(absolute(W[level])), label="Eigenstate")
plot(x, V, label="Potential")
plot(x, E[level]*ones(N), label="Eigenenergy")
xlabel('x(A)',fontsize=20)
ylabel('V(eV)',fontsize=20)
legend(loc=0); # upper left corner
print('Level')
print(level)
print('Eigenenergy')
print(E[level])
```

```
Level
1
Eigenenergy
-2.40137413911
```



0.1 Ejercicios

Obtenga todos los estados ligados, con energía menor que cero, del sistema para un número de pozos NW=1, NW=2, NW=5, NW=8. Realice un gráfico donde en el eje x represente el número de pozos del sistema y en el eje y todas las energías ligadas obtenidas. Estudie la formación de bandas de energías.

Represente el estado fundamental, level=0, para una cadena de NW=1, NW=4, NW=8 pozos. Observe las similitudes y diferencias con las funciones de Bloch como son los autoestados de una cadena infinita.

Represente el estado de mínima energía de cada una de las bandas en una cadena de NW=8 pozos. Observe las similitudes y diferencias con las funciones de Bloch como son los autoestados de una cadena infinita. (Ayuda: observe los estados ligados de un único pozo)

Represente las seis primeros autoestados de una cadena de NW=8 pozos. Obtenga el número de nodos y la longitud de onda de estos estados. Exprese una ecuación que relacione la longitud de onda con el nivel del autoestado y el tamaño del sistema.

0.2 Para saber más...

En el siguiente enlace se puede acceder a una simulación interactiva realizada en la "University of Colorado at Boulder" donde podreis estudiar los autoestados y autoenergías de una cadena finita

de átomos para diferentes perfiles de potencial y en presencia de un campo eléctrico.

In [1]:
```
from IPython.display import HTML
HTML('<div style="position: relative; width: 300px; height: 232px;"><a hre
```

Out [1]:
```
<IPython.core.display.HTML at 0x3835390>
```