Laboratorio 2: aproximación numérica de derivadas

¡Bienvenidos al segundo laboratorio!

La clase pasada aprendimos a aproximar derivadas numéricamente utilizando la expansión de Taylor y diferencias finitas. Durante esta sesión, vamos a implementar algunas de las técnicas que vimos en esa clase. Eso si, antes de escribir cualquier línea de código, repasaremos un poco de la teoría.

Repaso teoría

Malla computacional

Digamos que existe una función $f(x)$ cuya expresión no conocemos, pero si sabemos su valor numérico en puntos discretos del dominio x_i para $i = 0, 1, 2$. Nuestra misión es encontrar una buena aproximación de la derivada en el punto x_i a partir de la los datos disponibles.

La figura anterior es una representación de esta situación: no tenemos una expresión para la función $f(x)$ (línea punteada), pero si conocemos el valor de $f(x_i)$ en los puntos x_i. A partir de esto queremos calcular la derivada $\partial f / \partial x$, que es la pendiente (m) de la recta roja (tangente a la curva).

Llamaremos *malla computacional* a la colección de puntos x_i, y es ahí donde queremos calcular $\partial f / \partial x$. Para facilitar la discusión, digamos que los puntos x_i están separados una misma distancia h, lo que implica que tenemos una *malla regular*.

Expansion de Taylor

Utilizaremos la expansión de Taylor para encontrar una expresión que aproxime $\partial f/\partial x$ en los puntos x_i. Usando solamente los términos de primer orden, la expansión de Taylor nos permite aproximar el valor de $f(x_i + h) = f(x_{i+1})$ a partir de $f(x_i)$:

$$f(x_{i+1}) \approx f(x_i) + h\frac{\partial f}{\partial x}\bigg|_{x_i} + O(h^2).$$

Despejando, queda

$$\frac{\partial f}{\partial x}\bigg|_{x_i} \approx \frac{f(x_{i+1}) - f(x_i)}{h} + O(h)$$

¿Ven como usamos los valores en x_i y x_{i+1} para calcular la derivada en x_i? Esto se conoce como *diferencia adelantada*. Si tomamos el valor en x_{i-1} en vez de x_{i+1} llegamos a la fórmula de *diferencia atrasada*:

$$\frac{\partial f}{\partial x}\bigg|_{x_i} \approx \frac{f(x_i) - f(x_{i-1})}{h} + O(h).$$

Otra opción es restar las expansiones de Taylor para $f(x_{i-1})$ y $f(x_{i+1})$, lo que nos lleva a *diferencia centrada*:

$$\frac{\partial f}{\partial x}\bigg|_{x_i} \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2).$$

¿Se fijaron que el error de la aproximación en diferencias centradas es $O(h^2)$? En este caso, el término $\frac{h^2}{2}\frac{\partial^2 f}{\partial x^2}$ de la expansión de Taylor se cancela al hacer la resta. Esto implica que con diferencia centrada el error decae con h^2 a medida que h se hace más chico. Decimos entonces que diferencia centrada es una aproximación de *segundo orden*, mientras que diferencia adelantada y atrasada son de *primer orden*.

La imagen es una representación gráfica de cada una de las aproximaciones. La pendiente de la recta roja es la derivada aproximada con diferencia atrasada, la pendiente de la recta azul es la aproximación con diferencia adelantada y la negra con diferencia centrada.

Derivadas de segundo orden

Consideremos la expansión de Taylor para $f(x)$ evaluados en x_{i+1} y x_{i-1} hasta los términos de segundo orden:

$$f(x_{i+1}) \approx f(x_i) + h\frac{\partial f}{\partial x}\bigg|_{x_i} + \frac{h^2}{2}\frac{\partial^2 f}{\partial x^2}\bigg|_{x_i} + \frac{h^3}{3!}\frac{\partial^3 f}{\partial x^3}\bigg|_{x_i} + O(h^4),$$

$$f(x_{i-1}) \approx f(x_i) - h\frac{\partial f}{\partial x}\bigg|_{x_i} + \frac{h^2}{2}\frac{\partial^2 f}{\partial x^2}\bigg|_{x_i} - \frac{h^3}{3!}\frac{\partial^3 f}{\partial x^3}\bigg|_{x_i} + O(h^4)$$

Para aproximar la primera derivada con diferencia centrada nosotros restamos estas dos expresiones, pero ahora nos conviene sumarlas. Esto nos da:

$$\frac{\partial^2 f}{\partial x^2}\bigg|_{x_i} \approx \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{h^2} + O(h^2)$$

lo que también es una aproximación de segundo orden (el error cae con h^2).

Problema: Aproximación de la derivada de $\arctan(x)$

Para probar las técnicas que aprendimos en clase, vamos a aproximar las derivadas de la función $\arctan(x)$ en puntos entre -10 y 10, y con diferentes valores de h.

Preparando la solución

Partamos importando algunas librerías: `numpy` contiene operaciones sobre arreglos que son muy útiles, y `pyplot` de `matplotlib` nos permitirá graficar nuestros resultados. La línea que dice `%matplotlib inline` es requerida por IPython Notebooks para mostrar las imágenes de `matplotlib`, sin embargo, no es necesaria en el caso que hagan el código en un archivo de texto aparte. La función `rcParams` controla la tipografía en las imágenes generadas por `matplotlib`.

In [2]:
```
import numpy as np
from matplotlib import pyplot
%matplotlib inline
from matplotlib import rcParams
rcParams['font.family'] = 'serif'
rcParams['font.size'] = 16
```

Usemos cinco tamaños de malla: $h = 0.5, 0.25, 0.125,$, y generémoslas usando la función `numpy.arange` (http://docs.scipy.org/doc/numpy/reference/generated/numpy.arange.html#numpy.arange). Guardemos las mallas en un arreglo bidimensional `X`, donde `X[i]` tiene la malla correspondiente a `h[i]`, para `i=0,1,2,...`.

Fíjense que cada elemento de `X` es un arreglo con diferentes dimensiones. La forma más fácil de lidiar con esto es inicializándolo con la función `numpy.empty` (http://docs.scipy.org/doc/numpy/reference/generated/numpy.empty.html).

In [11]:
```
h = np.array([0.5, 0.25, 0.125, 0.0625, 0.03125, 0.03125/2.])
x_start = -10.
x_end   =  10.

X = np.empty(len(h), dtype=object)
for i in range(len(h)):
    X[i] = np.arange(x_start, x_end+h[i], h[i])

X.shape
```

Out[11]: (6,)

Aseguremonos que está todo bien: deben ser puntos equidistantes entre -10 y 10

In [6]:
```
for i in range(len(h)):
    print ('\nMalla para h[%i]=%1.3f'%(i,h[i]))
    print (X[i])
```

```
Malla para h[0]=0.500
[-10.   -9.5  -9.   -8.5  -8.   -7.5  -7.   -6.5  -6.   -5.5  -5.   -4.5
  -4.   -3.5  -3.   -2.5  -2.   -1.5  -1.   -0.5   0.    0.5   1.    1.5
   2.    2.5   3.    3.5   4.    4.5   5.    5.5   6.    6.5   7.    7.5
   8.    8.5   9.    9.5  10. ]

Malla para h[1]=0.250
[-10.    -9.75  -9.5   -9.25  -9.    -8.75  -8.5   -8.25  -8.    -7.75
  -7.5   -7.25  -7.    -6.75  -6.5   -6.25  -6.    -5.75  -5.5   -5.25
  -5.    -4.75  -4.5   -4.25  -4.    -3.75  -3.5   -3.25  -3.    -2.75
  -2.5   -2.25  -2.    -1.75  -1.5   -1.25  -1.    -0.75  -0.5   -0.25
   0.     0.25   0.5    0.75   1.     1.25   1.5    1.75   2.     2.25
   2.5    2.75   3.     3.25   3.5    3.75   4.     4.25   4.5    4.75
   5.     5.25   5.5    5.75   6.     6.25   6.5    6.75   7.     7.25
   7.5    7.75   8.     8.25   8.5    8.75   9.     9.25   9.5    9.75
  10.  ]

Malla para h[2]=0.125
[-10.     -9.875  -9.75   -9.625  -9.5    -9.375  -9.25   -9.125  -9.
  -8.875  -8.75   -8.625  -8.5    -8.375  -8.25   -8.125  -8.     -7.875
  -7.75   -7.625  -7.5    -7.375  -7.25   -7.125  -7.     -6.875  -6.75
  -6.625  -6.5    -6.375  -6.25   -6.125  -6.     -5.875  -5.75   -5.625
  -5.5    -5.375  -5.25   -5.125  -5.     -4.875  -4.75   -4.625  -4.5
  -4.375  -4.25   -4.125  -4.     -3.875  -3.75   -3.625  -3.5    -3.375
  -3.25   -3.125  -3.     -2.875  -2.75   -2.625  -2.5    -2.375  -2.25
  -2.125  -2.     -1.875  -1.75   -1.625  -1.5    -1.375  -1.25   -1.125
  -1.     -0.875  -0.75   -0.625  -0.5    -0.375  -0.25   -0.125   0.
   0.125   0.25    0.375   0.5     0.625   0.75    0.875   1.      1.125
   1.25    1.375   1.5     1.625   1.75    1.875   2.      2.125   2.25
   2.375   2.5     2.625   2.75    2.875   3.      3.125   3.25    3.375
   3.5     3.625   3.75    3.875   4.      4.125   4.25    4.375   4.5
   4.625   4.75    4.875   5.      5.125   5.25    5.375   5.5     5.625
   5.75    5.875   6.      6.125   6.25    6.375   6.5     6.625   6.75
   6.875   7.      7.125   7.25    7.375   7.5     7.625   7.75    7.875
   8.      8.125   8.25    8.375   8.5     8.625   8.75    8.875   9.
   9.125   9.25    9.375   9.5     9.625   9.75    9.875  10.   ]

Malla para h[3]=0.062
[-10.      -9.9375  -9.875   -9.8125  -9.75    -9.6875  -9.625   -9.5625
  -9.5     -9.4375  -9.375   -9.3125  -9.25    -9.1875  -9.125   -9.0625
  -9.      -8.9375  -8.875   -8.8125  -8.75    -8.6875  -8.625   -8.5625
  -8.5     -8.4375  -8.375   -8.3125  -8.25    -8.1875  -8.125   -8.0625
  -8.      -7.9375  -7.875   -7.8125  -7.75    -7.6875  -7.625   -7.5625
  -7.5     -7.4375  -7.375   -7.3125  -7.25    -7.1875  -7.125   -7.0625
  -7.      -6.9375  -6.875   -6.8125  -6.75    -6.6875  -6.625   -6.5625
  -6.5     -6.4375  -6.375   -6.3125  -6.25    -6.1875  -6.125   -6.0625
  -6.      -5.9375  -5.875   -5.8125  -5.75    -5.6875  -5.625   -5.5625
  -5.5     -5.4375  -5.375   -5.3125  -5.25    -5.1875  -5.125   -5.0625
  -5.      -4.9375  -4.875   -4.8125  -4.75    -4.6875  -4.625   -4.5625
  -4.5     -4.4375  -4.375   -4.3125  -4.25    -4.1875  -4.125   -4.0625
  -4.      -3.9375  -3.875   -3.8125  -3.75    -3.6875  -3.625   -3.5625
  -3.5     -3.4375  -3.375   -3.3125  -3.25    -3.1875  -3.125   -3.0625
  -3.      -2.9375  -2.875   -2.8125  -2.75    -2.6875  -2.625   -2.5625
  -2.5     -2.4375  -2.375   -2.3125  -2.25    -2.1875  -2.125   -2.0625
  -2.      -1.9375  -1.875   -1.8125  -1.75    -1.6875  -1.625   -1.5625
  -1.5     -1.4375  -1.375   -1.3125  -1.25    -1.1875  -1.125   -1.0625
  -1.      -0.9375  -0.875   -0.8125  -0.75    -0.6875  -0.625   -0.5625
```

```
 -0.5     -0.4375  -0.375   -0.3125  -0.25    -0.1875  -0.125   -0.0625
  0.       0.0625   0.125    0.1875   0.25     0.3125   0.375    0.4375
  0.5      0.5625   0.625    0.6875   0.75     0.8125   0.875    0.9375
  1.       1.0625   1.125    1.1875   1.25     1.3125   1.375    1.4375
  1.5      1.5625   1.625    1.6875   1.75     1.8125   1.875    1.9375
  2.       2.0625   2.125    2.1875   2.25     2.3125   2.375    2.4375
  2.5      2.5625   2.625    2.6875   2.75     2.8125   2.875    2.9375
  3.       3.0625   3.125    3.1875   3.25     3.3125   3.375    3.4375
  3.5      3.5625   3.625    3.6875   3.75     3.8125   3.875    3.9375
  4.       4.0625   4.125    4.1875   4.25     4.3125   4.375    4.4375
  4.5      4.5625   4.625    4.6875   4.75     4.8125   4.875    4.9375
  5.       5.0625   5.125    5.1875   5.25     5.3125   5.375    5.4375
  5.5      5.5625   5.625    5.6875   5.75     5.8125   5.875    5.9375
  6.       6.0625   6.125    6.1875   6.25     6.3125   6.375    6.4375
  6.5      6.5625   6.625    6.6875   6.75     6.8125   6.875    6.9375
  7.       7.0625   7.125    7.1875   7.25     7.3125   7.375    7.4375
  7.5      7.5625   7.625    7.6875   7.75     7.8125   7.875    7.9375
  8.       8.0625   8.125    8.1875   8.25     8.3125   8.375    8.4375
  8.5      8.5625   8.625    8.6875   8.75     8.8125   8.875    8.9375
  9.       9.0625   9.125    9.1875   9.25     9.3125   9.375    9.4375
  9.5      9.5625   9.625    9.6875   9.75     9.8125   9.875    9.9375
 10.    ]

Malla para h[4]=0.031
[-10.       -9.96875  -9.9375   -9.90625  -9.875    -9.84375  -9.8125
  -9.78125  -9.75     -9.71875  -9.6875   -9.65625  -9.625    -9.59375
  -9.5625   -9.53125  -9.5      -9.46875  -9.4375   -9.40625  -9.375
  -9.34375  -9.3125   -9.28125  -9.25     -9.21875  -9.1875   -9.15625
  -9.125    -9.09375  -9.0625   -9.03125  -9.       -8.96875  -8.9375
  -8.90625  -8.875    -8.84375  -8.8125   -8.78125  -8.75     -8.71875
  -8.6875   -8.65625  -8.625    -8.59375  -8.5625   -8.53125  -8.5
  -8.46875  -8.4375   -8.40625  -8.375    -8.34375  -8.3125   -8.28125
  -8.25     -8.21875  -8.1875   -8.15625  -8.125    -8.09375  -8.0625
  -8.03125  -8.       -7.96875  -7.9375   -7.90625  -7.875    -7.84375
  -7.8125   -7.78125  -7.75     -7.71875  -7.6875   -7.65625  -7.625
  -7.59375  -7.5625   -7.53125  -7.5      -7.46875  -7.4375   -7.40625
  -7.375    -7.34375  -7.3125   -7.28125  -7.25     -7.21875  -7.1875
  -7.15625  -7.125    -7.09375  -7.0625   -7.03125  -7.       -6.96875
  -6.9375   -6.90625  -6.875    -6.84375  -6.8125   -6.78125  -6.75
  -6.71875  -6.6875   -6.65625  -6.625    -6.59375  -6.5625   -6.53125
  -6.5      -6.46875  -6.4375   -6.40625  -6.375    -6.34375  -6.3125
  -6.28125  -6.25     -6.21875  -6.1875   -6.15625  -6.125    -6.09375
  -6.0625   -6.03125  -6.       -5.96875  -5.9375   -5.90625  -5.875
  -5.84375  -5.8125   -5.78125  -5.75     -5.71875  -5.6875   -5.65625
  -5.625    -5.59375  -5.5625   -5.53125  -5.5      -5.46875  -5.4375
  -5.40625  -5.375    -5.34375  -5.3125   -5.28125  -5.25     -5.21875
  -5.1875   -5.15625  -5.125    -5.09375  -5.0625   -5.03125  -5.
  -4.96875  -4.9375   -4.90625  -4.875    -4.84375  -4.8125   -4.78125
  -4.75     -4.71875  -4.6875   -4.65625  -4.625    -4.59375  -4.5625
  -4.53125  -4.5      -4.46875  -4.4375   -4.40625  -4.375    -4.34375
  -4.3125   -4.28125  -4.25     -4.21875  -4.1875   -4.15625  -4.125
  -4.09375  -4.0625   -4.03125  -4.       -3.96875  -3.9375   -3.90625
  -3.875    -3.84375  -3.8125   -3.78125  -3.75     -3.71875  -3.6875
  -3.65625  -3.625    -3.59375  -3.5625   -3.53125  -3.5      -3.46875
  -3.4375   -3.40625  -3.375    -3.34375  -3.3125   -3.28125  -3.25
  -3.21875  -3.1875   -3.15625  -3.125    -3.09375  -3.0625   -3.03125
  -3.       -2.96875  -2.9375   -2.90625  -2.875    -2.84375  -2.8125
```

```
-2.78125  -2.75     -2.71875  -2.6875   -2.65625  -2.625    -2.59375
-2.5625   -2.53125  -2.5      -2.46875  -2.4375   -2.40625  -2.375
-2.34375  -2.3125   -2.28125  -2.25     -2.21875  -2.1875   -2.15625
-2.125    -2.09375  -2.0625   -2.03125  -2.       -1.96875  -1.9375
-1.90625  -1.875    -1.84375  -1.8125   -1.78125  -1.75     -1.71875
-1.6875   -1.65625  -1.625    -1.59375  -1.5625   -1.53125  -1.5
-1.46875  -1.4375   -1.40625  -1.375    -1.34375  -1.3125   -1.28125
-1.25     -1.21875  -1.1875   -1.15625  -1.125    -1.09375  -1.0625
-1.03125  -1.       -0.96875  -0.9375   -0.90625  -0.875    -0.84375
-0.8125   -0.78125  -0.75     -0.71875  -0.6875   -0.65625  -0.625
-0.59375  -0.5625   -0.53125  -0.5      -0.46875  -0.4375   -0.40625
-0.375    -0.34375  -0.3125   -0.28125  -0.25     -0.21875  -0.1875
-0.15625  -0.125    -0.09375  -0.0625   -0.03125   0.        0.03125
 0.0625    0.09375   0.125     0.15625   0.1875    0.21875   0.25
 0.28125   0.3125    0.34375   0.375     0.40625   0.4375    0.46875
 0.5       0.53125   0.5625    0.59375   0.625     0.65625   0.6875
 0.71875   0.75      0.78125   0.8125    0.84375   0.875     0.90625
 0.9375    0.96875   1.        1.03125   1.0625    1.09375   1.125
 1.15625   1.1875    1.21875   1.25      1.28125   1.3125    1.34375
 1.375     1.40625   1.4375    1.46875   1.5       1.53125   1.5625
 1.59375   1.625     1.65625   1.6875    1.71875   1.75      1.78125
 1.8125    1.84375   1.875     1.90625   1.9375    1.96875   2.
 2.03125   2.0625    2.09375   2.125     2.15625   2.1875    2.21875
 2.25      2.28125   2.3125    2.34375   2.375     2.40625   2.4375
 2.46875   2.5       2.53125   2.5625    2.59375   2.625     2.65625
 2.6875    2.71875   2.75      2.78125   2.8125    2.84375   2.875
 2.90625   2.9375    2.96875   3.        3.03125   3.0625    3.09375
 3.125     3.15625   3.1875    3.21875   3.25      3.28125   3.3125
 3.34375   3.375     3.40625   3.4375    3.46875   3.5       3.53125
 3.5625    3.59375   3.625     3.65625   3.6875    3.71875   3.75
 3.78125   3.8125    3.84375   3.875     3.90625   3.9375    3.96875
 4.        4.03125   4.0625    4.09375   4.125     4.15625   4.1875
 4.21875   4.25      4.28125   4.3125    4.34375   4.375     4.40625
 4.4375    4.46875   4.5       4.53125   4.5625    4.59375   4.625
 4.65625   4.6875    4.71875   4.75      4.78125   4.8125    4.84375
 4.875     4.90625   4.9375    4.96875   5.        5.03125   5.0625
 5.09375   5.125     5.15625   5.1875    5.21875   5.25      5.28125
 5.3125    5.34375   5.375     5.40625   5.4375    5.46875   5.5
 5.53125   5.5625    5.59375   5.625     5.65625   5.6875    5.71875
 5.75      5.78125   5.8125    5.84375   5.875     5.90625   5.9375
 5.96875   6.        6.03125   6.0625    6.09375   6.125     6.15625
 6.1875    6.21875   6.25      6.28125   6.3125    6.34375   6.375
 6.40625   6.4375    6.46875   6.5       6.53125   6.5625    6.59375
 6.625     6.65625   6.6875    6.71875   6.75      6.78125   6.8125
 6.84375   6.875     6.90625   6.9375    6.96875   7.        7.03125
 7.0625    7.09375   7.125     7.15625   7.1875    7.21875   7.25
 7.28125   7.3125    7.34375   7.375     7.40625   7.4375    7.46875
 7.5       7.53125   7.5625    7.59375   7.625     7.65625   7.6875
 7.71875   7.75      7.78125   7.8125    7.84375   7.875     7.90625
 7.9375    7.96875   8.        8.03125   8.0625    8.09375   8.125
 8.15625   8.1875    8.21875   8.25      8.28125   8.3125    8.34375
 8.375     8.40625   8.4375    8.46875   8.5       8.53125   8.5625
 8.59375   8.625     8.65625   8.6875    8.71875   8.75      8.78125
 8.8125    8.84375   8.875     8.90625   8.9375    8.96875   9.
 9.03125   9.0625    9.09375   9.125     9.15625   9.1875    9.21875
 9.25      9.28125   9.3125    9.34375   9.375     9.40625   9.4375
 9.46875   9.5       9.53125   9.5625    9.59375   9.625     9.65625
```

```
 9.6875    9.71875   9.75      9.78125   9.8125    9.84375   9.875
 9.90625   9.9375    9.96875  10.      ]

Malla para h[5]=0.016
[-10.        -9.984375  -9.96875  ...   9.96875    9.984375  10.      ]
```

Y ahora calculemos la función $\arctan(x)$ en cada uno de esos puntos. La guardaremos en el arreglo `F`, que tiene la misma estructura que `X`.

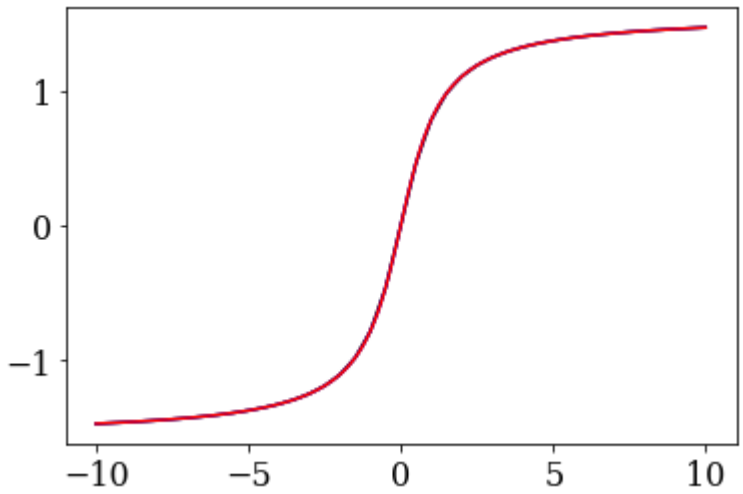
In [12]:
```
F = np.empty_like(X)
for i in range(len(h)):
    F[i] = np.arctan(X[i])
```

Out[12]: 6

¿Cómo se ve eso? Acá graficamos `F[0]`, pero intenten con `F[1]`, `F[2]`, `F[3]` y `F[4]` si quieren asegurarse.

In [10]:
```
pyplot.plot(X[0], F[0], c='k')
pyplot.plot(X[1], F[1], c='b')
pyplot.plot(X[2], F[2], c='r')
```

Out[10]: [<matplotlib.lines.Line2D at 0xe3d8ec8>]



Acá trabajan ustedes

A continuación, deberán generar funciones que calculen la aproximación de la derivada en cada punto de la malla usando diferencia adelantada, atrasada y centrada, y además calcular la aproximación de la segunda derivada. Para evitar problemas, no calculen la derivada en los extremos ($x = -10$ y $x = 10$), ya que necesitarían un valor de $f(x)$ que está fuera del dominio (arreglo) que definimos.

La función debe aproximar $\partial f/\partial x$ para un valor de h, y sus datos de entrada son $f(x)$ y h

In [37]:
```
### ALUMNO
#def diferencia_adelantada(f,h):
#def diferencia_atrasada(f,h):
#def diferencia_centrada(f,h):
#def segunda_derivada(f,h):
# Hacer funciones que aproximen la derivada con diferencia
# adelantada, atrasada, centrada y para la segunda derivada
###


def diferencia_adelantada(f,h):
    df = np.zeros(len(f)-2)
    for j in range(1,len(f)-1):
        df[j-1] = (f[j+1]-f[j])/h

    return df

def diferencia_atrasada(f,h):
    df = np.zeros(len(f)-2)
    for j in range(1,len(f)-1):
        df[j-1] = (f[j]-f[j-1])/h
    return df

def diferencia_centrada(f,h):
    df = np.zeros(len(f)-2)
    for j in range(1,len(f)-1):
        df[j-1] = (f[j+1]-f[j-1])/(2*h)
    return df

def segunda_derivada(f,h):
    df = np.zeros(len(f)-2)
    for j in range(1,len(f)-1):
        df[j-1] = (f[j+1]-2*f[j]+f[j-1])/(h**2)
    return df
```

Ahora, usemos sus funciones para aproximar las derivadas con cada método. Las guardaremos en un arreglo 3D: para cada valor de h, habrá un arreglo 2D con la información de la siguiente forma:

- Fila 0: diferencia adelantada
- Fila 1: diferencia atrasada
- Fila 2: diferencia centrada
- Fila 3: segunda derivada

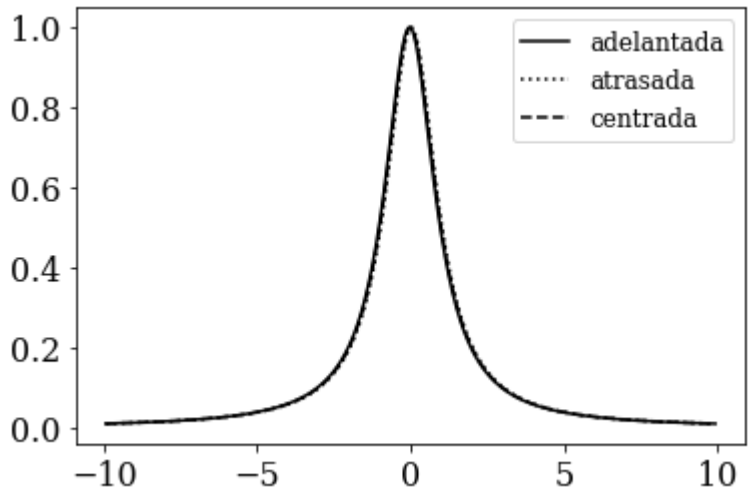
Para esto, debemos guardar `f0`, `f1`, `f2` y `f3` en un arreglo 2D también, que llamaremos `F`.

In [53]:
```
# Inicializamos el arreglo donde guardaremos los resultados
aprox = numpy.empty(len(h), dtype=object)
for i in range(len(h)):
    aprox[i] = numpy.zeros((4,len(F[i])-2))

for i in range(len(h)):
    aprox[i][0,:] = diferencia_adelantada(F[i],h[i])
    aprox[i][1,:] = diferencia_atrasada(F[i],h[i])
    aprox[i][2,:] = diferencia_centrada(F[i],h[i])
    aprox[i][3,:] = segunda_derivada(F[i],h[i])
```

¡Listo! ¿Cómo se ve esta aproximación? Veamos el caso $h = 0.125$

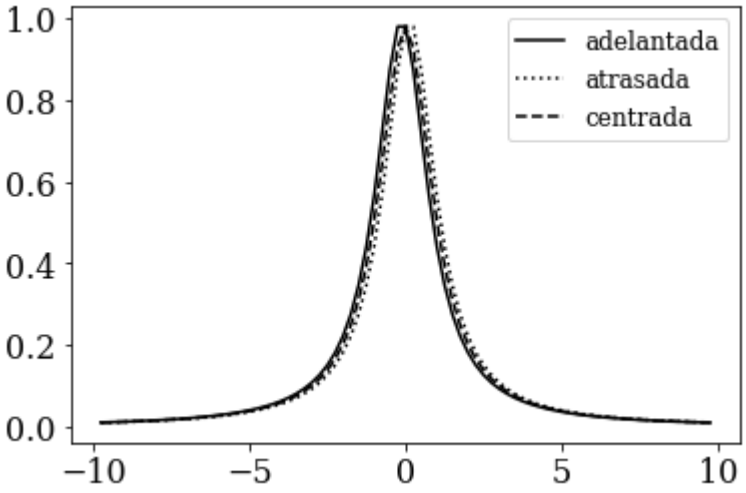
In [54]:
```
pyplot.plot(X[3][1:-1], aprox[3][0,:], c='k',ls='-',label='adelantada')
pyplot.plot(X[3][1:-1], aprox[3][1,:], c='k',ls=':',label='atrasada')
pyplot.plot(X[3][1:-1], aprox[3][2,:], c='k',ls='--',label='centrada')
pyplot.legend(loc='best',prop={'size':12})
```

Out[54]:
```
<matplotlib.legend.Legend at 0x108cab08>
```



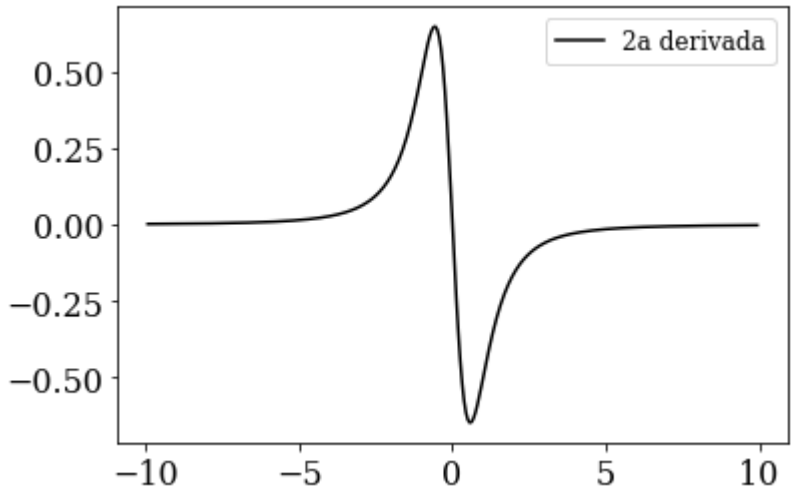
In [55]:
```
pyplot.plot(X[1][1:-1], aprox[1][0,:], c='k',ls='-',label='adelantada')
pyplot.plot(X[1][1:-1], aprox[1][1,:], c='k',ls=':',label='atrasada')
pyplot.plot(X[1][1:-1], aprox[1][2,:], c='k',ls='--',label='centrada')
pyplot.legend(loc='best',prop={'size':12})
```

Out[55]: <matplotlib.legend.Legend at 0x107ea6a0>



In [56]:
```
pyplot.plot(X[3][1:-1], aprox[3][3,:], c='k',ls='-',label='2a derivada')
pyplot.legend(loc='best',prop={'size':12})
```

Out[56]: <matplotlib.legend.Legend at 0x108cec70>



Cálculo del error

Para saber que tan buenas son estas aproximaciones, necesitamos calcular el error. Afortunadamente, podemos calcular las derivadas de $\arctan(x)$. A continuación, escriban una función para calcular $\partial \arctan(x)/\partial x$ en los mismos puntos donde calcularon la aproximación con diferencias finitas. Acuérdense que no calcularon la derivada en los extremos ($x = -10$ y $x = 10$).

In [57]:
```
###ALUMNO
#def dfdx_analitico(x):
#def ddfdxx_analitico(x):
# Hacer funciones para encontrar la solución analítica
# para cada caso
###

def dfdx_analitico(x):
    x = x[1:-1]
    x2 = 1/(x**2+1)
    return x2

def ddfdxx_analitico(x):
    x = x[1:-1]
    x2 = -2*x/((x**2+1)**2)
    return x2

ddfdxx_analitico(X[0])
```

Out[57]:
```
array([ 0.00228185,  0.00267698,  0.00316835,  0.00378698,  0.00457657,
        0.0056    ,  0.00694978,  0.00876552,  0.011264  ,  0.0147929 ,
        0.0199308 ,  0.02768166,  0.03987184,  0.06      ,  0.09512485,
        0.16      ,  0.28402367,  0.5       ,  0.64      , -0.        ,
       -0.64      , -0.5       , -0.28402367, -0.16      , -0.09512485,
       -0.06      , -0.03987184, -0.02768166, -0.0199308 , -0.0147929 ,
       -0.011264  , -0.00876552, -0.00694978, -0.0056    , -0.00457657,
       -0.00378698, -0.00316835, -0.00267698, -0.00228185])
```

Usando las funciones recién creadas podemos calcular el error en cada punto x_i. Un buen promedio de este error es su norma L_2 relativa:

$$||e||_2 = \sqrt{\frac{\sum_i (f_{i,aprox} - f_{i,an})^2}{\sum_i {f_{i,an}}^2}}$$

Generen una función para calcular la norma L_2 relativa

In [58]:
```
###ALUMNO
#def L2_error(f_ap, f_an):
# Hacer funcion para calcular norma L2 del error
###

def L2_error(f_ap,f_an):
    df_an_app = (f_ap-f_an)**2
    f_an2 = f_an**2
    return np.sqrt(np.sum(df_an_app)/np.sum(f_an2))
```

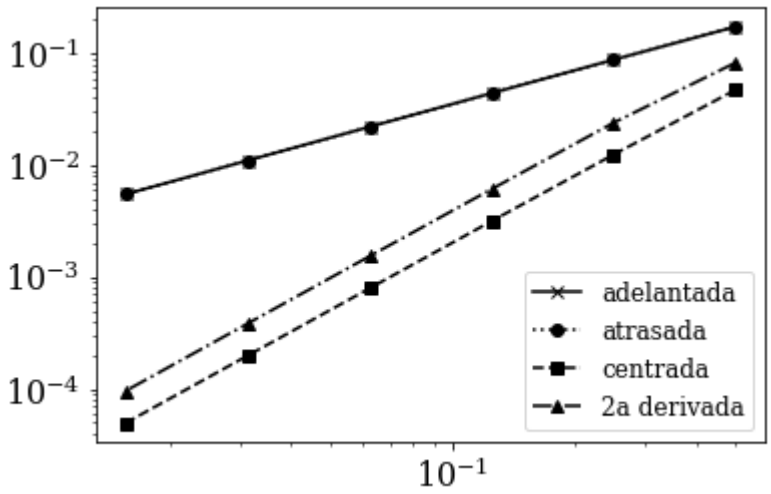
Usemos estas funciones para ver como se comporta el error con h

In [59]:
```
error_adelantada = numpy.zeros(len(h))
error_atrasada   = numpy.zeros(len(h))
error_centrada   = numpy.zeros(len(h))
error_2aderivada = numpy.zeros(len(h))

for i in range(len(h)):

    dfdx   = dfdx_analitico(X[i])
    ddfdxx = ddfdxx_analitico(X[i])

    error_adelantada[i] = L2_error(aprox[i][0,:],dfdx)
    error_atrasada[i]   = L2_error(aprox[i][1,:],dfdx)
    error_centrada[i]   = L2_error(aprox[i][2,:],dfdx)
    error_2aderivada[i] = L2_error(aprox[i][3,:],ddfdxx)
```

Grafiquemos los resultados

In [60]:
```
pyplot.loglog(h, error_adelantada, c='k', ls='-', marker='x', label='adelantada')
pyplot.loglog(h, error_atrasada, c='k', ls=':', marker='o', label='atrasada')
pyplot.loglog(h, error_centrada, c='k', ls='--', marker='s', label='centrada')
pyplot.loglog(h, error_2aderivada, c='k', ls='-.', marker='^', label='2a derivada')

pyplot.legend(loc='best',prop={'size':12})
```

Out[60]: <matplotlib.legend.Legend at 0x10ad2f10>



Conclusiones

Respondan las siguientes preguntas:

1. En una frase, comenten como se relaciona lo que ven en el último gráfico con los términos $O(h)$ y $O(h^2)$ de las aproximaciones.
2. Vimos que al afinar la malla, el resultado mejora. Sin embargo, ¿Existe alguna desventaja?
3. Si usamos diferencias centradas, y refinamos la malla en 4 veces, ¿Cuanto debiese disminuir el error de la aproximación?

Pregunta 1:

Se nota claramente como el error cae de distinta manera para los métodos centrados $O(h^2)$ y los métodos adelantados y atrasados $O(h)$. Tomando los primeros y los penúltimos valores de h, se nota que cuando los centrados caen en 2 ordenes de magintud, el adelantado y atrasado cae solo 1, reflejando lo anterior.

Pregunta 2

La gran ventaja que pueda existir es la gran capacidad computacional que se necesita para almacenar todos los valores de una malla e iterar sobre ella. Este es un ejemplo con un simple array, pero trabajando en la modelación de sistemas ingenieriles, el tiempo de ejecución y la memoria necesaria pueden ser determinantes.

Pregunta 3

Asumiendo que se pregunta sobre usar un $h_2 = \frac{h}{2^4}$, el error deberia caer en $2^8 = 256$ veces

In [ ]:

In [ ]: