Animación

Animación de función general $f(x,t)$.
Graficada en algun intervalo $[x_0, x_f] \times [t_0, t_f]$ con intervalos dx y dt.

Basado en trabajo de Francisco Nemiña.
Readaptado por Ionatan Perez.

Darío Mitnik

In [ ]:

```
# Acá hay distintas opciones, que pueden no funcionar

# Esta funciona en general, si no lo hace, probar
# de repetir la sentencia en cada ploteo.
%matplotlib notebook

# Para hacer las animaciones en la misma página.
# %matplotlib inline

# Para hacer las animaciones en ventanas separadas.
# (puede hacer algunos problemas si no está bien instalado)
# %matplotlib qt
```

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import animation
from IPython.display import HTML
```

Definimos la función animar.
Por defecto en el intervalo $[0, 1] \times [0, 1]$ con un paso de 0.01 en dx y dt.
$y_m = -2$ y $y_M = +2$ son los mínimos y máximos de la función a animar.
Estos valores se pueden cambiar cuando se llama a la funcion.

In [ ]:

```
def animar(f,x0=0,xf=1,dx=0.01,t0=0,tf=1,dt=0.01, ym=-2, yM= 2):
    """
    Funcion para realizar animaciones. Toma como entrada una funcion
        f(x,t)
    donde el primer argumento es la coordenada espacial y el segundo la tempora
l.

    Ademas pide como argumento de entrada
        x0 - la coordenada x inicial
        xf - la coordenada x final
        dx - el valor entre dos puntos sucesivos de x
        t0 - la coordenada t inicial
        tf - la coordenada t final
        dt - el valor entre dos puntos sucesivos de t
        ym - Valor minimo para y
        yM - Valor maximo para y


    Esta harcodeado el tiempo entre dos frames en 50 ms (20 frames por segundo).

    Se puede cambiar cambiando el valor de `interval` que aparece mas abajo.
    """

    # Definimos los intervalos para graficar
    nf = int( (xf-x0)/dx + 1)
    nt = int( (tf-t0)/dt + 1)
    x = np.linspace(x0,xf,nf)
    t = np.linspace(t0,tf,nt)


    # Ponemos los ejes para la figura
    fig, ax = plt.subplots()

    ax.set_xlim((x0, xf))
    ax.set_ylim((ym, yM))

    line, = ax.plot([], [], lw=2)

    # Definimos la funcion que crea un grafico vacio
    def init():
        line.set_data([], [])
        return (line,)

    # Funcion que crea cada frame
    def animate(i):
        y = f(x,i)
        line.set_data(x, y)
        return (line,)
    # Creamos el objeto animado
    anim = animation.FuncAnimation(fig, animate, init_func=init,
                                   frames=t, interval=20, blit=True)
    plt.show()
    return anim
```

Ejemplo: onda estacionaria

$f(x,t) = \sin(\frac{2\pi}{\lambda}x)\ \cos(2\pi f t) = \sin(kx) * \cos(\omega t)$

In [ ]:

```
# Definimos la funcion que queremos animar
def f(x,t):
    lamb=1
    frec=1
    knum =  2*np.pi/lamb
    omega = 2*np.pi*frec

    y = np.sin(knum*x)*np.cos(omega*t)
    return y
```

Hacemos la animacion

In [ ]:

```
lamb=1
frec=1
knum =  2*np.pi/lamb
omega = 2*np.pi*frec

anim = animar(f,x0=0,xf=lamb,dx=0.01,t0=0,tf=1/frec,dt=1./(50*frec), ym=-2, yM=
2)
plt.show()
```

In [ ]:

```
# Visualización Controlada

# Si ffmpeg está instalado se puede usar:
# HTML(anim.to_html5_video())

HTML(anim.to_jshtml())
```

Ejemplo: onda viajera

$f(x,t) = \sin(\frac{2\pi x}{\lambda} - 2\pi f t) = \sin(kx - \omega t) = \sin(x - vt)$

In [ ]:

```
# Definimos la funcion que queremos animar
def f(x,t):
    lamb=1
    frec=2
    knum =  2*np.pi/lamb
    omega = 2*np.pi*frec
    v = omega/knum

    y = np.sin(knum*(x - v*t))
    return y
```

Hacemos la animacion

In [ ]:

```
anim = animar(f,x0=0,xf=2,tf=1,dt=1./50);
```

In [ ]:

```
# Visualización Controlada

# Si ffmpeg está instalado se puede usar:
# HTML(anim.to_html5_video())

HTML(anim.to_jshtml())
```

Ejemplo: Gaussiana con centro que se mueve

$f(x,t) = e^{-(x-t)^2}$

In [ ]:

```
# Definimos la funcion que queremos animar
def f(x,t):
    return np.exp(-(x-t)**2)
```

Hacemos la animacion

In [ ]:

```
anim = animar(f,x0=-2,xf=2,t0=0,tf=1,ym=0,yM=1.1)
```

In [ ]:

```
# Visualización Controlada

# Si ffmpeg está instalado se puede usar:
# HTML(anim.to_html5_video())

HTML(anim.to_jshtml())
```

Guardado de animaciones

En el caso que se quieran guardar las animaciones es necesario tener instalada la libreria ffmpeg en el sistema operativo. En ubuntu se instala como

```
sudo apt-get install ffmpeg
```

Para guardar la animación hacemos

anim.save("nombre.mp4")

In [ ]:

```
anim.save("paquete.mp4")
```