Clase IX
Reconstruyendo el sistema dinámico a partir de datos.

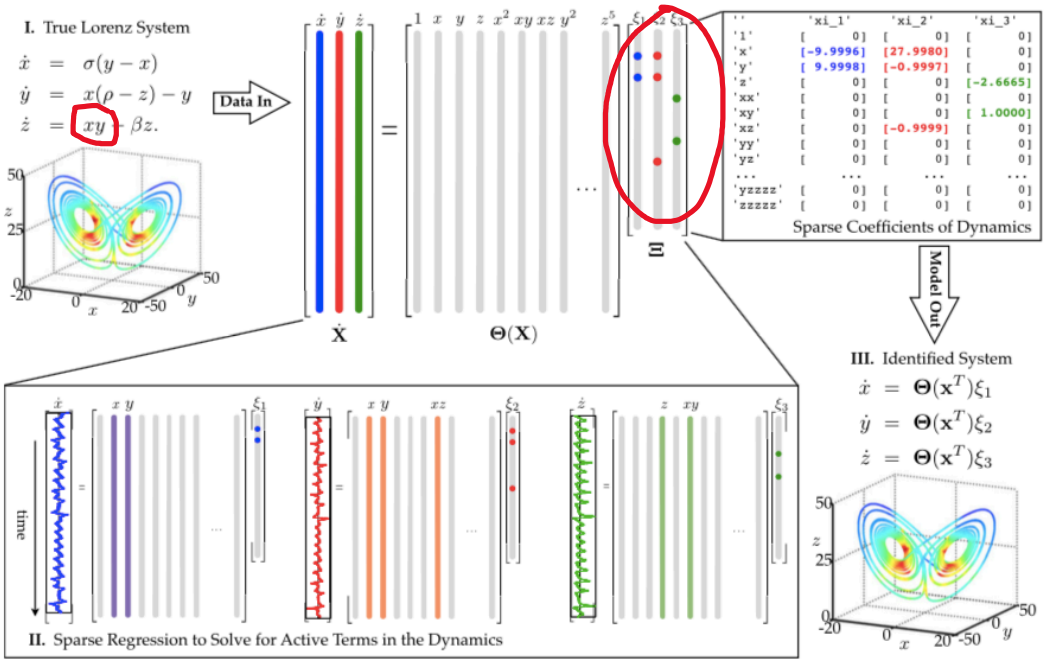
Hasta ahora vimos cómo leer propiedades cualitativas de un flujo a través del análisis del campo vector. Sin embargo, no hemos discutido cómo se deriva el conjunto de ecuaciones de interés. En muchos casos, la derivación sigue un estudio a partir de primeros principios. En otros, se proponen ecuaciones fenomenológicas en las cuales se intenta capturar algún aspecto clave de la dinámica. Un ejemplo de este tipo de modelado es el de las ecuaciones que describen la dinámica de una epidemia, en las cuales la tasa de variación de susceptibles ($\dot{S}$) de la población se asume depende de $-SI$, el producto entre la población susceptible y la población infectada, donde el producto modela el encuentro. Sin embargo, es deseable contar con herramientas que nos permitan dilucidar el sistema dinámico que eventualmente sea responsable de los datos, directamente, y en lo posible, automáticamente a partir de éstos. Es pertinente preguntarse qué gana uno escribiendo un sistema de ecuaciones a partir de los datos. Después de todo, predicción puede realizarse sin tener un modelo dinámico: **cuando vimos redes neuronales vimos una estrategia basada en datos, que no requirió modelado alguno.** Y aún en la historia de la física, modelos "data driven" permitieron espectaculares predicciones, como la de Kepler y las trayectorias elípticas de los planetas orbitando una estrella. La respuesta surge también de la historia: el tipo de predicción que logramos con Newton es más útil: no sólo permite predecir el comportamiento de un planeta cuyos datos sirvieron para ajustar el modelo, sino que uno puede intentar aplicar un modelo en un rango paramétrico lejano al recorrido durante la toma de datos. Por ejemplo, con Newton no sólo explicamos trayectorias elípticas, si no que podemos predecir las parabólicas o hiperbólicas.

Si tenemos las variables del problema, todas ellas, medidas en un conjunto de instancias temporales, podemos emplear la regresión para calcular las ecuaciones, bajo un conjunto de hipótesis razonables. Una es que el sistema original pueda en efecto escribirse en términos de monomios. La segunda hipótesis que permite una razonable eficacia en la recuperación de ecuaciones se fundamenta en la observación que los problemas en las ciencias naturales tiende a provenir de sistemas dinámicos con un conjunto ralo de monomios. Esto es,

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}), \qquad con\ \boldsymbol{x}\ \epsilon\ R^n,$$

y $\boldsymbol{f}$ una función con pocos términos lineales, cuadráticos, cúbicos, etc. Deseamos aproximar a ese campo vector entonces por:

$$\boldsymbol{f}(\boldsymbol{x}) \approx \sum_{k=1}^{p} \theta_k(\boldsymbol{x})\, \boldsymbol{\xi}_k = \boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\xi}$$



Así, para recuperar un campo vectorial a partir de datos, uno comienza tomando un conjunto de series temporales $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))^T$, y calculamos o medimos las derivadas de las variables también:

$$\begin{bmatrix} \boldsymbol{x^T}(t_1) \\ \boldsymbol{x^T}(t_2) \\ \ldots \\ \boldsymbol{x^T}(t_m) \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) \ldots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) \ldots & x_n(t_2) \\ \ldots & \ldots \ldots & \ldots \\ x_1(t_m) & x_2(t_m) & x_n(t_m) \end{bmatrix}$$

$$\begin{bmatrix} \dot{\boldsymbol{x}}^{\boldsymbol{T}}(t_1) \\ \dot{\boldsymbol{x}}^{\boldsymbol{T}}(t_2) \\ \ldots \\ \dot{\boldsymbol{x}}^{\boldsymbol{T}}(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) \ldots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) \ldots & \dot{x}_n(t_2) \\ \ldots & \ldots \ldots & \ldots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \dot{x}_n(t_m) \end{bmatrix}$$

Luego, construimos una "librería" con todos los términos de funciones no lineales que se nos ocurra pertinente incluir en el modelo (por ejemplo, todas las funciones polinomiales hasta cierto grado, funciones periódicas…):

$$\Theta(\boldsymbol{X}) = [\boldsymbol{I} \quad \boldsymbol{X} \quad \boldsymbol{X^{P_2}} \quad \boldsymbol{X^3} \ldots \quad \boldsymbol{cos}(\boldsymbol{X}) \ldots],$$

donde

$$\boldsymbol{X} = [\boldsymbol{x}(t_1), \boldsymbol{x}(t_2), \ldots \boldsymbol{x}(t_m)]^T$$

y planteamos un algoritmo de regresión "rala", tipo LASSO, para calcular los vectores de coeficientes del ajuste:

$$\Xi = [\boldsymbol{\xi_1} \quad \boldsymbol{\xi_2} \dots \quad \boldsymbol{\xi_n}],$$

Donde cada coeficiente dará cómo entras los términos no lineales, en cada una de las componentes del campo vector. De este modo, cada vector $\boldsymbol{\xi_1}$ tiene tantas componentes como columnas tenga la librería $\Theta(\boldsymbol{X})$.

Así, el sistema dinámico original, evaluado en los tiempos $\{t_1, t_2, \dots, t_m\}$ da lugar al siguiente sistema (sobre determinado!) de ecuaciones algebraicas:

$$\dot{\boldsymbol{X}} = \boldsymbol{\Theta}(\mathbf{X})\, \boldsymbol{\Xi},$$

donde cada columna $\boldsymbol{\xi_k}$ en $\boldsymbol{\Xi}$ es un vector cuyos coeficientes indican cuales son los términos presentes en el campo vector en la k-ésima fila del campo vector original:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}), \qquad con\ \boldsymbol{x}\ \epsilon\ R^n,$$

o sea, la de la componente:

$$\frac{dx_k}{dt} = f_k(x_{\mathbf{1}}, x_{\mathbf{2}}, \dots x_{\boldsymbol{n}}).$$

Con esta notación, entonces, un modelo para ajustar estos coeficientes (que según nuestra hipótesis de modelos "ralos" deberían ser pocos por componente) se logra estimando esos coeficientes mediante un algoritmo de regresión rala (l_1-regularizada):

$$\boldsymbol{\xi_k} = argmin_{\xi_{k'}}\left(\left\|\dot{\boldsymbol{X}}_{\boldsymbol{k}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\xi_{k'}}\right\|\right) + \lambda\|\boldsymbol{\xi_{k'}}\|),$$

donde $\dot{\boldsymbol{X}}_{\boldsymbol{k}}$ es la k-ésima columna de

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \dot{x_1}(t_1) & \dot{x_2}(t_1) & \dot{x_n}(t_1) \\ \dot{x_1}(t_2) & \dot{x_2}(t_2) & \dot{x_n}(t_2) \\ \dots & \dots \quad \dots & \dots \\ \dot{x_1}(t_m) & \dot{x_2}(t_m) & \dot{x_x}(t_m) \end{bmatrix}$$

El parámetro λ es el responsable de penalizar la presencia de términos en el ajuste. Es decir, uno busca el conjunto de coeficientes que constituyen $\boldsymbol{\xi_k}$ que minimizan la diferencia entre las derivadas y el campo vector, pero poniéndole un precio a cada término que agrego. El precio es precisamente λ.

Ahora, si $\boldsymbol{\Theta}(\boldsymbol{x})$ es un vector fila de funciones simbólicas de $\mathbf{x}$ (ojo, no confundir con $\boldsymbol{\Theta}(\mathbf{X})$, *con argumento* $\boldsymbol{X}$, que es una matriz!), el sistema dinámico es:

$$\frac{dx_k}{dt} = \boldsymbol{\Theta}(\boldsymbol{x})\boldsymbol{\xi_k}.$$

Veamos un ejemplo. Supongamos tenemos un sistema regido por las ecuaciones de Lorenz, y supongamos, además que las tres variables del sistema dinámico pueden ser medidas. Así, el problema está regido por estas ecuaciones:

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = x(\rho - z)$$
$$\frac{dz}{dt} = xy - \beta z.$$

Medimos las tres variables, calculamos las derivadas, y tenemos:

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \dot{x}(t_1) & \dot{y}(t_1) & z(t_1) \\ \dot{x}(t_2) & \dot{y}(t_2) & \dot{z}(t_2) \\ \cdots & \cdots & \cdots \\ \dot{x}(t_m) & \dot{y}(t_m) & \dot{z}(t_m) \end{bmatrix}.$$

Luego, calculamos la librería sobre las mediciones, esto es:

$$\Theta(\boldsymbol{X}) =$$
$$\begin{bmatrix} 1 & x(t_1) & y(t_1) & z(t_1) & x^2(t_1) & xy(t_1) & xz(t_1) & yz(t_1) & y^2(t_1) & z^2(t_1) \\ 1 & x(t_2) & y(t_2) & z(t_2) & x^2(t_2) & xy(t_2) & xz(t_2) & yz(t_2) & y^2(t_2) & z^2(t_2) \\ 1 & x(t_m) & y(t_m) & z(t_m) & x^2(t_m) & xy(t_m) & xz(t_m) & yz(t_m) & y^2(t_m) & z^2(t_m) \end{bmatrix}$$

y lo que necesitamos calcular son los coeficientes de los vectores ($\boldsymbol{\xi_1}, \boldsymbol{\xi_2}, \boldsymbol{\xi_3}$), cada uno de los cuales tiene 10 componentes. Así, la primera ecuación del sistema que debemos resolver lucirá:

$$\begin{bmatrix} \dot{x}(t_1) \\ \dot{x}(t_2) \\ \cdots \\ \dot{x}(t_m) \end{bmatrix} = \begin{bmatrix} 1 & x(t_1) & y(t_1) & z(t_1) & xy(t_1) & \ldots \\ 1 & x(t_2) & y(t_2) & z(t_2) & xy(t_2) & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \\ 1 & x(t_m) & y(t_m) & z(t_m) & xy(t_m) & \ldots \end{bmatrix} \begin{bmatrix} {\xi_1}^1 \\ {\xi_1}^2 \\ \ldots \\ {\xi_1}^{10} \end{bmatrix}$$

la segunda ecuación a resolver es:

$$\begin{bmatrix} \dot{y}(t_1) \\ \dot{y}(t_2) \\ \cdots \\ \dot{y}(t_m) \end{bmatrix} = \begin{bmatrix} 1 & x(t_1) & y(t_1) & z(t_1) & xy(t_1) & \ldots \\ 1 & x(t_2) & y(t_2) & z(t_2) & xy(t_2) & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \\ 1 & x(t_m) & y(t_m) & z(t_m) & xy(t_m) & \ldots \end{bmatrix} \begin{bmatrix} {\xi_2}^1 \\ {\xi_2}^2 \\ \ldots \\ {\xi_2}^{10} \end{bmatrix}$$

y la tercera es:

$$\begin{bmatrix} \dot{z}(t_1) \\ \dot{z}(t_2) \\ \dots \\ \dot{z}(t_m) \end{bmatrix} = \begin{bmatrix} 1\ x(t_1)\ y(t_1)\ z(t_1)\ xy(t_1)\ \dots \\ 1\ x(t_2)\ y(t_2)\ z(t_2)\ xy(t_2)\ \dots \\ \dots \quad \dots \quad \dots \quad \dots \quad \dots \\ 1\ x(t_m) y(t_m)\ z(t_m) xy(t_m)\ \dots \end{bmatrix} \begin{bmatrix} {\xi_3}^1 \\ {\xi_3}^2 \\ \dots \\ {\xi_3}^{10} \end{bmatrix}$$

El resultado del algoritmo de ajuste por regresión lineal, ralo, debería dar:

$$\begin{bmatrix} {\xi_1}^1 \\ {\xi_1}^2 \\ {\xi_1}^3 \\ {\xi_1}^4 \\ {\xi_1}^5 \\ {\xi_1}^6 \\ {\xi_1}^7 \\ {\xi_1}^8 \\ {\xi_1}^9 \\ {\xi_1}^{10} \end{bmatrix} = \begin{bmatrix} 0 \\ -\sigma \\ \sigma \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} {\xi_2}^1 \\ {\xi_2}^2 \\ {\xi_2}^3 \\ {\xi_2}^4 \\ {\xi_2}^5 \\ {\xi_2}^6 \\ {\xi_2}^7 \\ {\xi_2}^8 \\ {\xi_2}^9 \\ {\xi_2}^{10} \end{bmatrix} = \begin{bmatrix} 0 \\ \rho \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} {\xi_3}^1 \\ {\xi_3}^2 \\ {\xi_3}^3 \\ {\xi_3}^4 \\ {\xi_3}^5 \\ {\xi_3}^6 \\ {\xi_3}^7 \\ {\xi_3}^8 \\ {\xi_3}^9 \\ {\xi_3}^{10} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\beta \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

En qué consiste un ajuste de regresión “ralo”, tipo LASSO? Simplemente en buscar minimizar la función diferencia entre lo que esté del lado derecho e izquierdo de nuestra ecuación, pero sumándole un término por cada coeficiente distinto de cero que se introduzca en el ajuste. De ese modo, buscar la minimización metiendo tantos términos como sea posible queda excluido por la penalización.

Estos algoritmos están eficientemente implementados en las librerías estadísticas de cualquier paquete de programación de alto nivel. En Python, por ejemplo, el proceso es tan sencillo como armar con los datos la librería y aplicar una función de ajuste via Lasso. Supongamos nuestro problema viene descripto por tres variables (x, y, p), a partir de cuya medicion calculamos las derivadas temporales $(\dot{x}, \dot{y}, \dot{p})$. suponemos por simplicidad que consideraremos sólo términos lineales. Entonces las instrucciones:

```
theta[:,0]=np.ones_like(x)
theta[:,1]=x
theta[:,2]=y
theta[:,3]=p
```

cargan la matriz $\Theta(\boldsymbol{X})$, y asignan un uno al primer término, que se encarga de los términos constantes en el campo vector. Ahora, el ajuste es tan sencillo como correr las dos instrucciones que escribimos a continuación,

```
clf=linear_model.Lasso(alpha=0.01,max_iter=10000000
0,fit_intercept=False,normalize=False)
clf.fit(-theta,dxdt)
```

Es pertinente preguntarse por qué es necesario, o conveniente, contar con un sistema dinámico capaz de generar soluciones similares a los datos. Después de todo, si el interés fuera predecir, esto es, seguir integrando al sistema dinámico para obtener una nueva señal a partir de datos ya tomados, existen métodos alternativos, como el empleo de redes neuronales entrenadas con segmentos de señales como entrada, y los valores posteriores de los segmentos como objetivos.

Para entender la diferencia esencial entre "Data driven science" y dinámica es ilustrativo revisitar una de las primeras veces en que esta diferencia de aproximaciones se empleó para entender un problema. Las trayectorias planetarias. La aproximación de Kepler fue definitivamente "data driven". Y la conclusión en términos de resumen, no podría haber sido más espectacular. Sin embargo, lo que no puede Kepler predecir es que para ciertas condiciones iniciales, en lugar de elipses las trayectorias podrán ser parábolas o hipérbolas. Para llevar a cabo esa predicción necesitamos dinámica, y formas funcionales con parámetros capaces de no sólo ajustar datos, si no de predecir regímenes. No predecir la continuidad de la evolución de un flujo, si no los dramáticos cambios cualitativos que éstos pueden presentar para diferentes valores de los parámetros.

Así, supongamos que tenemos un sistema que depende de parámetros. Podemos incorporarlos en nuestro paradigma de trabajo si escribimos:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x};\mu)$$
$$\frac{d\mu}{dt} = 0$$

lo cual refleja justamente que μ es un parámetro, esto es, no evoluciona al menos en la misma escala temporal que el resto de las variables para las cuales modelamos la evolución mediante nuestro sistema dinámico. Nos proponemos entonces incluir a μ como si fuera una variable más, y buscamos un campo vector que sea una combinación rala de variables y parámetro de bifurcación. Está claro que dicha estrategia requiere que uno busque varios segmentos de señal, tomados para distintos valores de los parámetros.

Una estrategia similar puede emplearse si uno sospecha de una forma funcional para un forzante temporal. Si un forzante viene representado por una función $\boldsymbol{u} = \boldsymbol{u}(t)$, es posible incorporarla en el paradigma escribiendo al sistema como:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{u}(t), t)$$

$$\frac{dt}{dt} = 1$$

Por supuesto, será muy infrecuente que uno tenga mediciones de todas las variables que entran en un proceso dinámico particular. Si bien en el problema de una partícula moviéndose en una dimensión, las variables dinámicas dadas por la ley de Newton permiten reconstruir el espacio de fases a partir de la posición solamente, en principio será frecuente que uno tenga acceso a una o un subconjunto de las variables que determinan la dinámica del sistema. Por ejemplo, en un laser tendremos acceso a la intensidad, pero seguramente no a la inversión de población. Por eso uno debe estar preparado para una primera etapa de análisis que consistirá en la reconstrucción de un espacio de fases, típicamente recurriendo al teorema de Takens. Si el problema es extendido, la primera etapa consistirá en identificar una variedad de dimensión menor a la que colapsen los infinitos modos del problema, probablemente vía "Proper ortogonal decomposition" (POD), o la codificación minimal vía auto codificadores.

Estas herramientas probablemente nos ayuden a encarar problemas en diversas disciplinas para los cuales los datos comienzan a acumularse sin descanso, ya sea en el área de la neurociencia como el de la ecología, la dinámica de fluidos o el clima. Todos esos campos comparten la existencia de enormes bases de datos, y lo elusivo de los modelos que necesitaríamos para comprender la dinámica subyacente.

Estos métodos, sin embargo, tienen problemas serios

1. La hipótesis de que los campos vectores son ralos, lleva a proponer una regularización, y los resultados son muy inestables con los parámetros.
2. El ajuste de las derivadas es, por mas que se intente trabajarlo, problemático.

Vamos a explorar otro método de ajuste, mas convencional, para un problema sencillo. Y luego compararemos la solución con lo obtenido con Sindy.

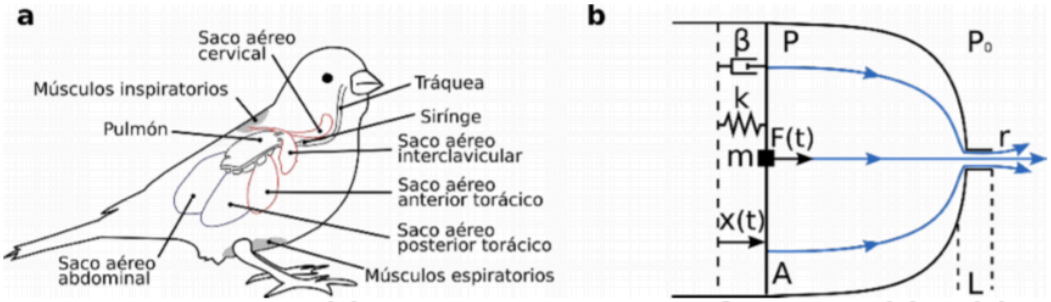


Figura 1: ***a*** *Esquema del sistema respiratorio aviar.* ***b*** *Esquema del modelo propuesto para el sistema respiratorio aviar.*

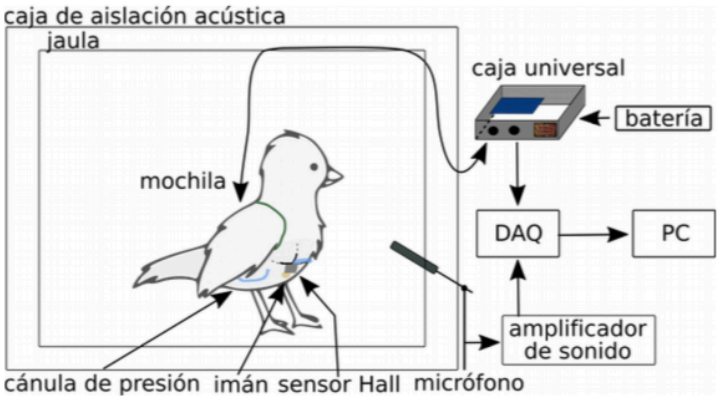
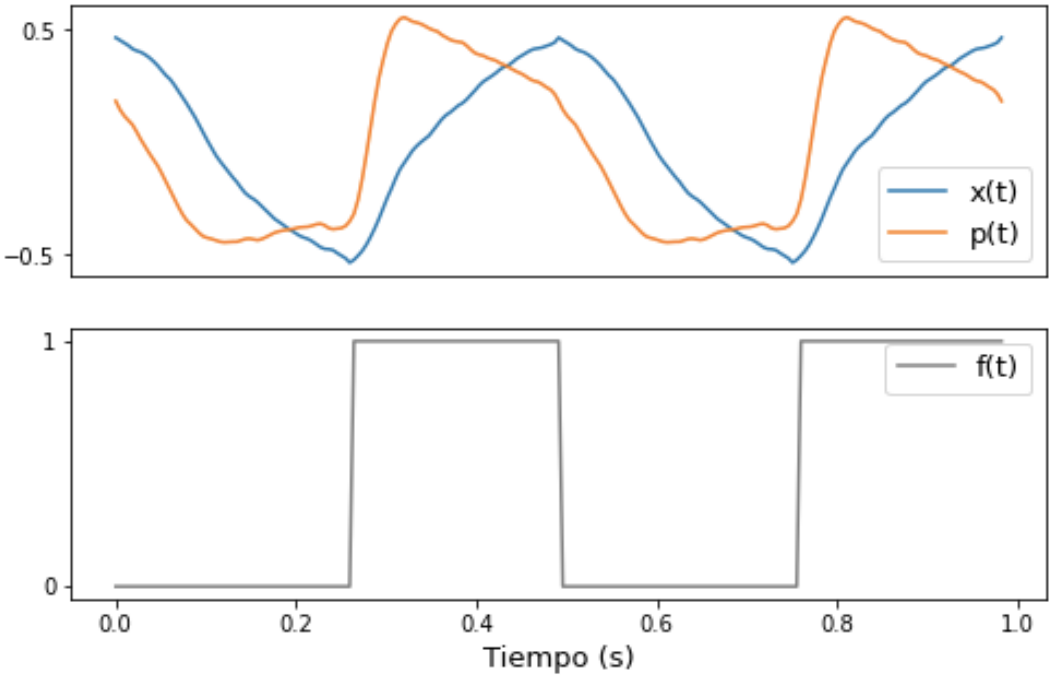


Figura 2: Esquema del armado experimental. Medimos presión de sacos aéreos, desplazamiento torácico y sonido.



En este caso, conjeturamos una forma funcional para el campo vector

$$\begin{cases} \dot{x} = y \\ \dot{y} = A + Bx + Cy + Dp + Ef(t) \\ \dot{p} = Hx + Fy + Gp \end{cases}$$

Para el siguiente conjunto de datos. Tanto x como p, medidos, el conjunto de tiempos en los que se registro la medición (t), como el forzante conjeturado están guardados.

¿De donde sale esta propuesta?

$$m\frac{d^2X(t)}{dt^2} = -kX(t) - \beta\frac{dX(t)}{dt} - A(P(t) - P_0) + F(t) \tag{3.1}$$

Con $P(t)$ y P_0 la presión interior y la presión exterior respectivamente, m la masa del pistón, k la constante de restitución, β la constante de disipación lineal, A el área del pistón y $F(t)$ la fuerza externa. Usamos $X(t)$ para el desplazamiento.

(II) Una ecuación para el flujo, que relacione la presión y el desplazamiento:

Buscamos una ecuación que dicte la dinámica de la presión. Usamos la ecuación de estado de un gas ideal:

$$P = \frac{Nk_BT}{V} = \frac{Nk_BT}{V_0 - AX} \tag{3.2}$$

Donde N es el número de moléculas en el interior, k_B la constante de Boltzmann, T la temperatura, y V_0 el volumen cuando el pistón está en la posición de equilibrio. Si diferenciamos:

$$dP = \frac{Nk_BT}{(V_0 - AX)^2}AdX + \frac{k_BT}{V_0 - AX}dN \tag{3.3}$$

Por otro lado, la variación del número de moléculas si estamos en una espiración es:

$$dN = -\rho u_{sal}dtA_{sal} = -\rho U_{sal}dt \approx -\rho_0 U_{sal}dt \tag{3.4}$$

Para un flujo laminar ($Re \approx 500$), incompresible ($M \approx 10^{-3}$)[1], tenemos la Ley de Hagen-Poiseuille que relaciona linealmente el flujo volumétrico con la diferencia de presión entre los extremos del tubo:

$$U = \frac{\pi R^4}{8\mu_f L}\Delta P \tag{3.5}$$

$$dN = -\rho_0\frac{\pi R^4}{8\mu_f L}\Delta P dt \tag{3.6}$$

Y entonces, si volvemos a la ecuación diferencial del gas ideal:

$$dP = \frac{Nk_BT}{(V_0 - AX)^2}AdX - \rho_0\frac{k_BT}{V_0 - AX}\frac{\pi R^4}{8\mu_f L}\Delta P dt \tag{3.7}$$

Si dividimos por el diferencial de tiempo, despreciamos todos los términos no lineales y usamos que la presión atmosférica exterior es constante en el tiempo, tenemos que:

$$\frac{d\Delta P(t)}{dt} = \frac{P_0}{V_0}A\frac{dX(t)}{dt} - \frac{P_0}{V_0}\frac{\pi R^4}{8\mu_f L}\Delta P(t) \tag{3.8}$$

Y tal como ocurre cuando uno tiene que ajustar los parámetros de una red neuronal, siempre termina en un problema de **optimización**.

Algorítmicamente, paquetes estadísticos como **lmfit**, de Python, tienen funciones como **minimize,** que toman:

1. Objetivo
2. Un objeto Params (que pueden variarse durante el ajuste)

Y realizan una búsqueda mediante métodos a elección, que incluyen "brute force" (posta), Nelder-Mead, etc…

La instrucción clave es entonces:

```
trial = minimize(residual,params=params_inicial,args=(t, data), method=metodo, calc_covar=False)
```

Donde el "residual" es la función por minimizar. La función minimize tiene como argumentos, además, el objeto "params", los argumentos ("args", que son los datos, necesarios para llevar a cabo el ajuste, y el método).

"params", se introducen aclarando el rango

```
params_inicial = Parameters()
params_inicial.add('A', value=inicial[:,i][0],min=-bound,max=0)
params_inicial.add('B', value=inicial[:,i][1],min=-bound, max=0)
params_inicial.add('C', value=inicial[:,i][2],min=-bound,max=0)
if p_on:
    params_inicial.add('D', value=inicial[:,i][3], min=-bound_p,max=0)
else:
    params_inicial.add('D', value=0, vary=False)
params_inicial.add('E', value=inicial[:,i][4], min=0,max=bound)
params_inicial.add('F', value=inicial[:,i][5],min=0,max=bound)
params_inicial.add('G', value=inicial[:,i][6],min=-bound,max=0)
if x_on:
```

"residual", se define como la diferencia entro los datos y lo que prevé el flujo sobre los mismos tiempos en que se midió:

```
def g(t, x0, paras):   #Solucion de la ODE con condicion inicial 'x0' y parametros 'paras'

    x = odeint(f, x0, t, args=(paras,))
    return x

def residual(paras, t, data):   #Función de costo a minimizar

    model = g(t, z0, paras)
    model = np.concatenate((model[:, 0],model[:, 2]))

    return model-data
```

(nota, en este ejemplo, los datos a contrastar son x y p, las componentes 0 y 2 del modelo).

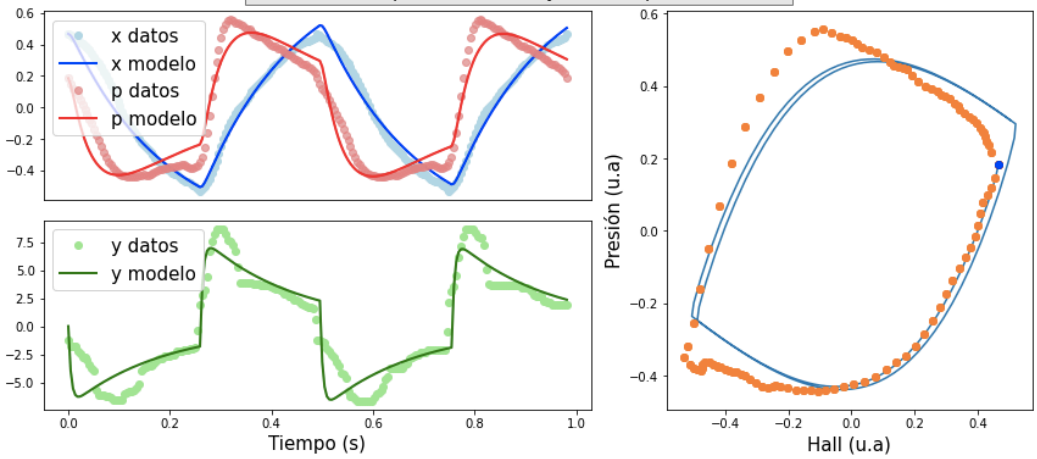
El campo vector necesario para calcular el flujo g, se declara previamente:

```
z0 = [x[0],0,p[0]] # Condicion inicial
data = np.concatenate((x,p))  #Ajusto un solo vector que contiene las señales de x y p

def f(z,tiempo,paras):    # z = (x,y,p)
    A = paras['A'].value
    B = paras['B'].value
    C = paras['C'].value
    D = paras['D'].value
    E = paras['E'].value
    F = paras['F'].value
    G = paras['G'].value
    H = paras['H'].value

    x = z[0]
    y = z[1]
    p = z[2]

    #    En principio, permito que  el campo vector tenga todos los términos,
    # pero después voy puedo elegir desactivar E y H.
    dxdt = y
    dydt = A*1 + B*x + C*y + D*p + E*funcion_forzante(tiempo)
    dpdt = H*x + F*y + G*p
    dzdt = [dxdt,dydt,dpdt]
    return dzdt
```

Mejora mucho, en este paradigma, hacer una búsqueda sobre una grilla de parámetros iniciales

El tipo de ajuste obtenido:

```
dy/dt =  - 830.  - 980. x  - 190. y  + 1800. f(t)

dp/dt =  + 2.3 y  - 23. p
```

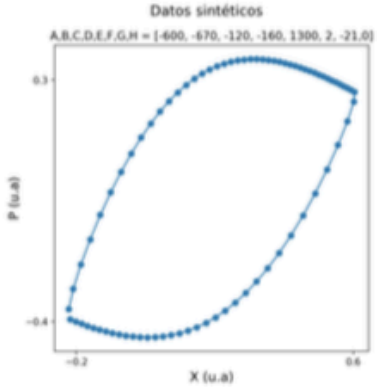


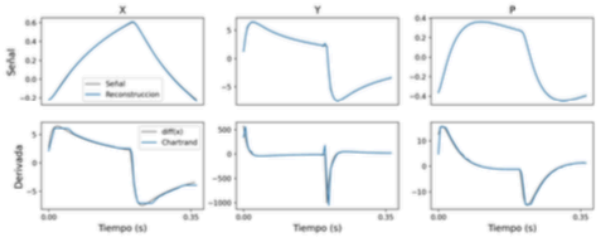
Ajuste SINDy de datos sintéticos (sin ruido)

Genero datos sintéticos con:

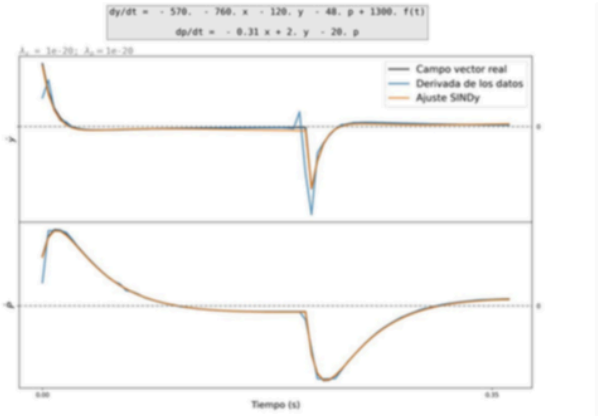
$$\begin{cases} \dot{x} = y \\ \dot{y} = A + Bx + Cy + Dp + E.f(t) \\ \dot{p} = Fy + Gp + Hx \end{cases}$$

```
A,B,C,D,E,F,G = [-600, -670, -120, -160, 1300, 2.1, -21, 0]
```





Y hago el ajuste SINDy:



	[A	B	C	D	E	F	G	H]	$\chi^2_\nu(y)$
λ_{reales}	[-600	-670	-120	-160	1300	2	-21	0]	8889
λ_{SINDy}	[-567	-763	-122	-48	1256	2	-20	-0.3]	8550

SCORE = 0.272

No solo aparece el termino con x en la segunda ecuación: el chi2 de la solución con el termino espurio da mejor que forzando a que el mismo desaparezca.

Guarda con los métodos automáticos.