Trabajo Fin de Grado
Grado en Ingeniería Electrónica Robótica y Mecatrónica

Predicción basada en datos de un sistema de tres tanques

Autor: Manuel López Pulido

Tutor: Daniel Limón Marruedo

Dpto. de Ingeniería de Sistemas y Automatización
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado
Grado en Ingeniería Electrónica Robótica y Mecatrónica

Predicción basada en datos de un sistema de tres tanques

Autor:
Manuel López Pulido

Tutor:
Daniel Limón Marruedo
Catedrático de Universidad

Dpto. de Ingeniería de Sistemas y Automatización
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado: Predicción basada en datos de un sistema de tres tanques

Autor: Manuel López Pulido
Tutor: Daniel Limón Marruedo

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

A mi entorno afectivo, tanto familia como amigos, por el apoyo recibido durante el grado. A mi tutor Daniel por los ánimos recibidos a pesar de las dificultades encontradas.

Resumen

El propósito de este estudio es desarrollar el control predictivo en bucle abierto de un sistema no lineal. Para ello será necesario primero un proceso de entrenamiento, a partir del cual sea posible conocer el comportamiento de dicho sistema y cómo se comportará frente a nuevas señales de control, siendo este el proceso de validación.

Primero se llevarán a cabo las predicciones frente a sistemas estáticos y finalmente se utilizarán sistemas dinámicos como es el caso del sistema de 3 tanques conectados en línea, de tal modo que al recibir una señal de control deseada, en este caso un caudal de entrada, se pueda llegar a predecir cómo será el comportamiento de la salida, es decir, la altura del último depósito. Todo esto se puede desarrollar mediante distintos procesos de Machine Learning como pueden ser mínimos cuadrados, redes neuronales, procesos gausiandos, etc. aunque en este trabajo se implementará usando el algoritmo de predicción de Kinky Inference, el cual se apoya en el conocimiento de los datos pasados para poder predecir los futuros, llegando a convertir el problema de identificación de parámetros en un problema de aprendizaje.

Abstract

The purpose of this study is to develop open-loop predictive control of a non-linear system. A training process will first be necessary, from which it is possible to know the behavior of this system and how it will behave in the face of new control signals, this being the validation process.

First, the predictions with static systems will be carried out and finally dynamic systems will be used, such as the 3-tank system connected in line, in such a way that when receiving a desired control signal, in this case an inflow, it is possible to predict how the output will behave, that is, the height of the last tank. All this can be developed through different Machine Learning processes such as least squares, neural networks, Gaussian processes, etc. Although in this work it will be implemented using the Kinky Inference prediction algorithm, which relies on the knowledge of past data to be able to predict future ones, turning the parameter identification problem into a learning problem.

Índice Abreviado

Índice

Notación

$\mathbb{R}$	Cuerpo de los números reales
$\mathbb{C}$	Cuerpo de los números complejos
$\|\mathbf{v}\|$	Norma del vector $\mathbf{v}$
$\langle \mathbf{v}, \mathbf{w} \rangle$	Producto escalar de los vectores $\mathbf{v}$ y $\mathbf{w}$
$\|\mathbf{A}\|$	Determinante de la matriz cuadrada $\mathbf{A}$
$\det(\mathbf{A})$	Determinante de la matriz (cuadrada) $\mathbf{A}$
$\mathbf{A}^\top$	Transpuesto de $\mathbf{A}$
$\mathbf{A}^{-1}$	Inversa de la matriz $\mathbf{A}$
$\mathbf{A}^\dagger$	Matriz pseudoinversa de la matriz $\mathbf{A}$
$\mathbf{A}^{\mathrm{H}}$	Transpuesto y conjugado de $\mathbf{A}$
$\mathbf{A}^*$	Conjugado
c.t.p.	En casi todos los puntos
c.q.d.	Como queríamos demostrar
■	Como queríamos demostrar
□	Fin de la solución
e.o.c.	En cualquier otro caso
e	número e
$\mathrm{e}^{\mathrm{j}x}$	Exponencial compleja
$\mathrm{e}^{\mathrm{j}2\pi x}$	Exponencial compleja con 2π
$\mathrm{e}^{-\mathrm{j}x}$	Exponencial compleja negativa
$\mathrm{e}^{-\mathrm{j}2\pi x}$	Exponencial compleja negativa con 2π
$\mathbb{R}\mathrm{e}$	Parte real
$\mathbb{I}\mathrm{m}$	Parte imaginaria
sen	Función seno
tg	Función tangente
arc tg	Función arco tangente
$\sin^y x$	Función seno de x elevado a y
$\cos^y x$	Función coseno de x elevado a y
Sa	Función sampling
sgn	Función signo
rect	Función rectángulo
Sinc	Función sinc
$\frac{\partial y}{\partial x}$	Derivada parcial de y respecto a x
x°	Notación de grado, x grados.
$\Pr(A)$	Probabilidad del suceso A
$E[X]$	Valor esperado de la variable aleatoria X

X	Varianza de la variable aleatoria X
$\sim f_X(x)$	Distribuido siguiendo la función densidad de probabilidad $f_X(x)$
$\mathcal{N}(m_X, \mathrm{X})$	Distribución gaussiana para la variable aleatoria X, de media m_X y varianza X
$\mathbf{I}_n$	Matriz identidad de dimensión n
$\mathrm{diag}(\mathbf{x})$	Matriz diagonal a partir del vector $\mathbf{x}$
$\mathrm{diag}(\mathbf{A})$	Vector diagonal de la matriz $\mathbf{A}$
SNR	Signal-to-noise ratio
MSE	Minimum square error
:	Tal que
$\stackrel{\text{def}}{=}$	Igual por definición
$\|\mathbf{x}\|$	Norma-2 del vector $\mathbf{x}$
$\|\mathbf{A}\|$	Cardinal, número de elementos del conjunto $\mathbf{A}$
$\mathbf{x}_i, i = 1,2,\ldots,n$	Elementos i, de 1 a n, del vector $\mathbf{x}$
$\mathrm{d}x$	Diferencial de x
$\leqslant$	Menor o igual
$\geqslant$	Mayor o igual
$\backslash$	Backslash
$\Leftrightarrow$	Si y sólo si
$x = a + 3 \underset{\substack{\uparrow \\ a=1}}{=} 4$	Igual con explicación
$\frac{a}{b}$	Fracción con estilo pequeño, a/b
Δ	Incremento
$b \cdot 10^a$	Formato científico
$\underset{x}{\rightarrow}$	Tiende, con x
O	Orden
™	Trade Mark
$\mathbb{E}[x]$	Esperanza matemática de x
$\mathbf{C}_\mathbf{x}$	Matriz de covarianza de $\mathbf{x}$
$\mathbf{R}_\mathbf{x}$	Matriz de correlación de $\mathbf{x}$
x	Varianza de x

1 Introducción Machine Learning

El Machine Learning es la rama de la inteligencia artificial que estudia como dotar a la máquinas de capacidad de aprendizaje, entendido este como la generalización del conocimiento a partir de un conjunto de experiencias siguiendo una serie de algoritmos (siendo un algoritmo un conjunto de pasos que se suceden para realizar una tarea).

Esta técnica de aprendizaje automático busca construir un modelo que se encargue de realizar una tarea, para ello necesita primero disponer de una gran cantidad de datos con los que realizar un buen entrenamiento, para posteriormente tener la habilidad de realizar predicciones.

1.1 Tipos de aprendizaje

El Machine Learning puede dividirse en tres grupos diferentes:

- Aprendizaje supervisado.
- Aprendizaje no supervisado.
- Aprendizaje reforzado

1.1.1 Aprendizaje supervisado

El algoritmo trabaja con datos etiquetados, de modo que a partir de estas, se encarga de asignar la etiqueta de salida correspondiente a cada uno de los datos. Para que se realice de manera correcta, previamente dicho algoritmo debe entrenarse con un histórico de datos con el cual llegar a aprender y poder predecir el valor de salida.

Generalmente estos algoritmos engloban problemas de clasificación y regresión.

Los algoritmos más comunes para este método de aprendizaje son:

- Árboles de decisión.
- Clasificación de Naïve Bayes.
- Regresión por mínimos cuadrados.
- Regresión Logística.
- Support Vector Machines (SVM).
- Métodos “Ensemble” (Conjuntos de clasificadores).

1.1.2 Aprendizaje no supervisado

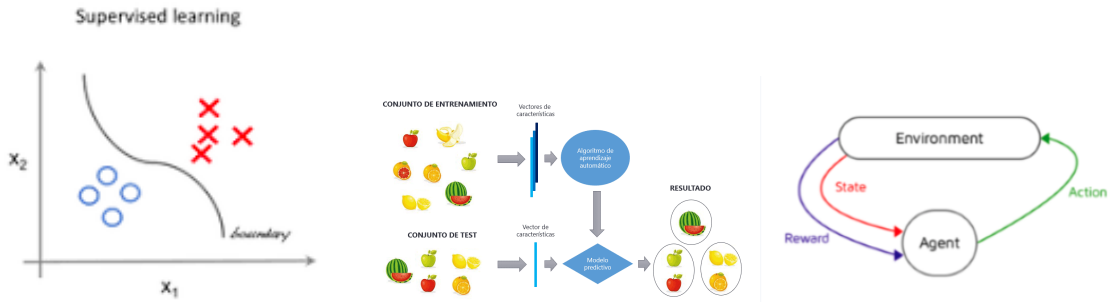
En este tipo de aprendizaje sólo se puede describir la estructura de los datos ya que no se dispone de etiquedas para los datos de salida, de modo que se intenta encontrar algún método de organización para poder simplificar el análisis, es decir, buscan agrupar basándose en las similitudes.

Los algoritmos más comunes para este método de aprendizaje son:

- Algoritmos de clustering.
- Análisis de componentes principales.
- Descomposición en valores singulares (singular value decomposition).
- Análisis de componentes principales (Independent Component Analysis)

1.1.3 Aprendizaje reforzado

Este aprendizaje está caracterizado por perfeccionar la respuesta del modelo usando un proceso de realimentación. De este modo, se puede considerar que el sistema aprende basándose en el método de **ensayo-error**.



(a) Apredizaje supervisado. **(b)** Apredizaje no supervisado. **(c)** Apredizaje reforzado.

Figura 1.1 Clasificación Machine Learning.

2 Kinky Inference

Kinky Inference es una técnica de Machine Learning que se basa en el aprendizaje supervisado, el cual es capaz de predecir la trayectoría de un sistema no lineal. Este modelo de control predictivo (MPC) se basa en la interpolación de Lipschitz.

Para implementar dicho algoritmo es necesario que se cumpla la siguiente condición:

$$\|f(w_1) - f(w_2))\| \leq L \|w_1 - w_2\| \tag{2.1}$$

Resultando:

$$\tilde{f}_j(q;\theta,D) = \frac{1}{2} min_{i=1,\ldots,N_D}(\tilde{f_{i,j}} + L_D \|q - w_i\|^{\alpha}) + \frac{1}{2} max_{i=1,\ldots,N_D}(\tilde{f_{i,j}} - L_D \|q - w_i\|^{\alpha}) \tag{2.2}$$

Donde w_i hace referencia a los regresores del sistema.

2.1 Constante de Lipschitz

Para calcular la constante L basta con determinar la mínima pendiente resultante de unir todos los puntos obtenidos en el entrenamiento.

$$L_D = max_{(w,\tilde{f})\varepsilon D} \left\{ \frac{\|f_1 - f_2)\| - 2\bar{e}}{\|w_1 - w_2\|} \right\} \tag{2.3}$$

Una vez determinada la L a partir de la ecuación 2.3, se aplica dicha pendiente a cada punto que se desea validar originando una serie de áreas con formas similares a rombos en las cuales se irán situando los puntos de las futuras predicciones, por lo que se estaría hablando de un error acotado por dichas rectas. Esto se puede observar en la siguiente figura 2.1

Tras ser determinados los puntos en los que se evaluará la función, estos se tomarán como query points y comenzará el cálculo de los puntos predichos.

Una vez calculados dichos puntos por la función KKI, se irán acumulando en un vector llamado SOL_v, para posteriormente poder comparar dichos resultados con los resultados reales.

Se adjunta el código KKI realizado para la siguiente función ejemplo 2.2:

$$y = \frac{5}{50^2} \cdot \|x\|^2 + 3 \cdot sin(0.1 \cdot min(x) + 5) \tag{2.4}$$

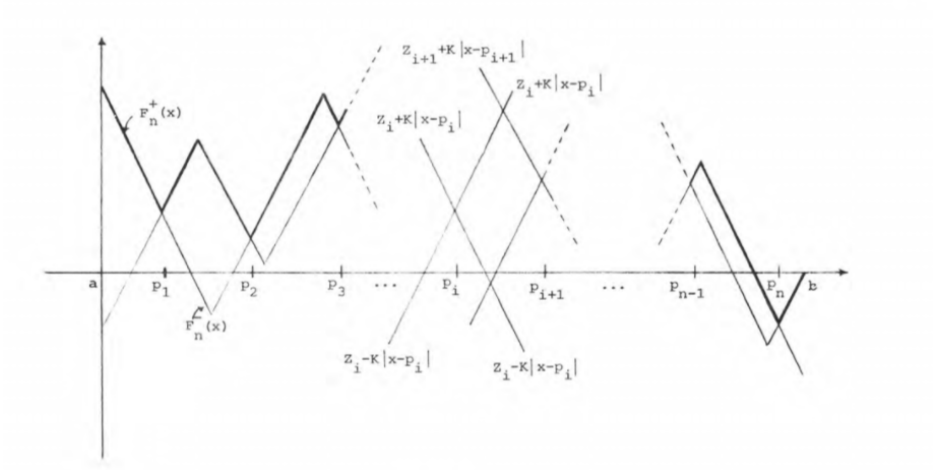


Figura 2.1 Áreas confinadas por las pendientes entre los puntos.

Código 2.1 Función Kinky Inference.

```
N=10;
W=[];
Y=[];

for i=1:N
    x=10*i;
    y=(5/50^2*norm(x)^2+3*sin(0.1*min(x))+5)';
    W=[W;x];
    Y=[Y;y];
end

Wc=[];
Yc=[];

for j=1:1000
    x=j/10;
    y=(5/50^2*norm(x)^2+3*sin(0.1*min(x))+5)';
    Wc=[Wc;x];
    Yc=[Yc;y];
end


plot(Wc,Yc);
hold on
plot(W,Y,'bx')
grid on


%CALCULO DE L

for i=1:N-1
 for j=(i+1):N
  L_candidata(i,j)=norm(Y(i,:)-Y(j,:))/norm(W(i,:)-W(j,:));
 end
end
```

```
L=max(max(L_candidata(:,:)));



Q_v=[];
SOL_v=[];
N_v=1000;
ny=1;
for j=1:N_v
    q=j/N_v*100;
    max1=-10000*ones(ny,1);
    min1=10000*ones(ny,1);
    L_v=L*ones(ny,1);
    for i=1:N
        max1=max(max1,Y(i,:)'-L_v*norm(q-W(i,:)'));
        min1=min(min1,Y(i,:)'+L_v*norm(q-W(i,:)'));
    end
    f_gorro=0.5*max1+0.5*min1;
    Q_v=[Q_v;q];
    SOL_v=[SOL_v;f_gorro];
end
```

El resultado obtenido con dicho procedimiento se adjunta en 2.2:

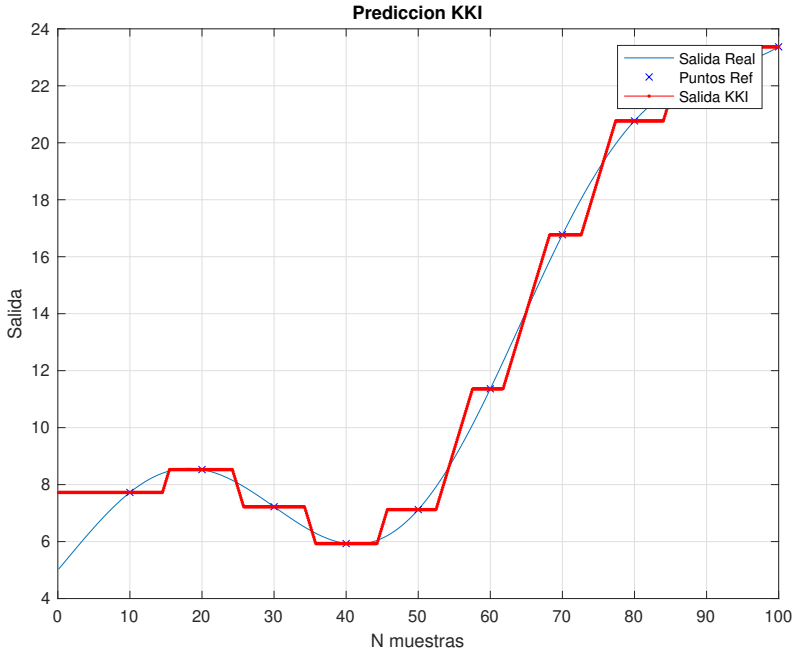


Figura 2.2 KKI aplicado a función .

Si en lugar de tener una función estática SISO, como es el caso anterior, pasamos a tener una función estática MISO como es la función Peaks 2.3:

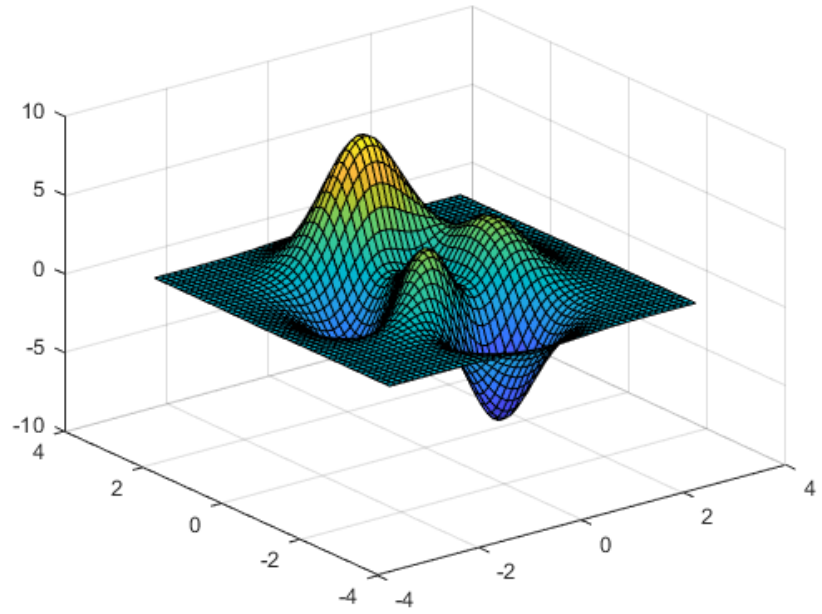


Figura 2.3 Función Peaks.

Se obtiene el siguiente resultado 2.4:

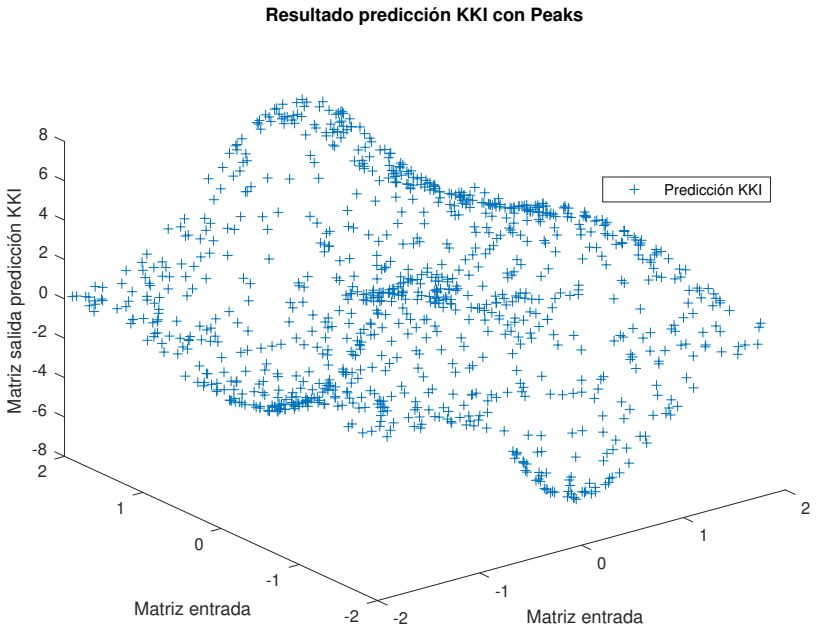
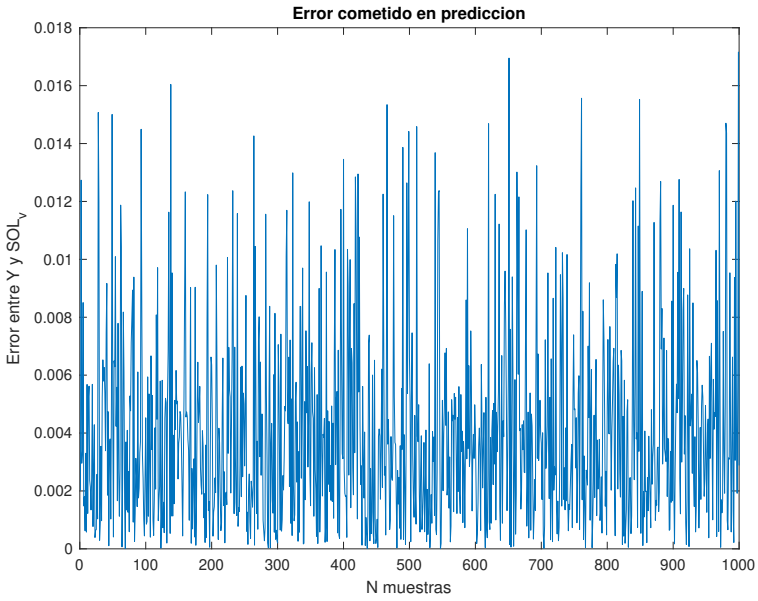


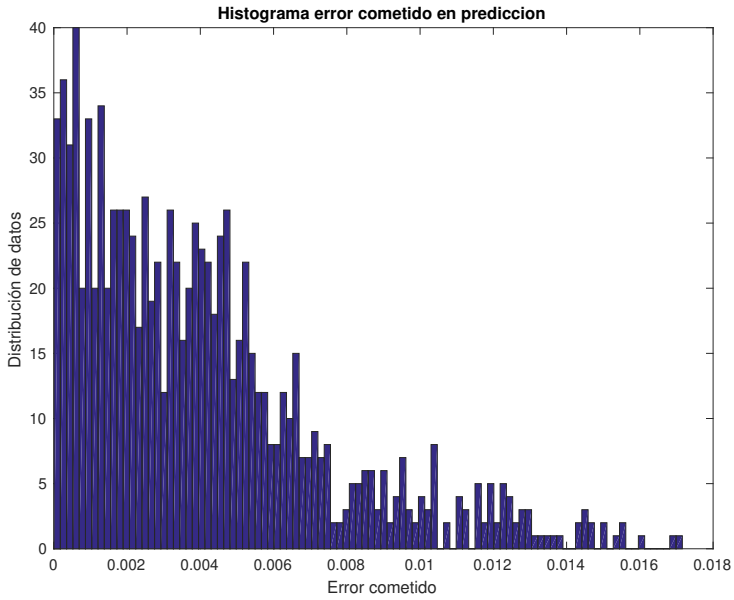
Figura 2.4 Predicción función Peaks.

Para ello se siguió el mismo procedimiento que para la función anterior, aunque para comprobar que el algoritmo se estaba implementando y prediciendo de manera correcta, en lugar de utilizar los propios regresores de entrada de la base de datos, se le añadió una pequeña desviación, por lo que el error cometido en predicción debía ser del mismo orden que dicha desviación. Los resultados

obtenidos fueron los siguientes:



(a) Predicción - Valor real.



(b) Histograma Error cometido.

Figura 2.5 Error cometido en predicción.

Hay que añadir que al aumentar el número de puntos tomados durante el proceso de entrenamiento, el proceso de validación será mucho más preciso, aunque esto implicaría un problema por la gran carga computacional que supondría.

2.2 Compilación de KKI mediante mex

Una de las alternativas que se planteó fue la de optimitizar la función de Kinky Inference mediante la compilación de funciones .c en MATLAB usando un mex file.

Para ello primero hay que explicar cómo se genera un fichero .c compilable en mex. Consta de tres partes identificables:

- Librerías y variables tipo define:Aquí deben incluirse las librerías *"mex.h"* y *<math.h>*. Además de las demás variables predefinidas en el algoritmo de Kinky Inference, como son la constante de Lipschitz, número de iteraciones, tamaño de las matrices, etc.

Código 2.2 Librerías y define del mex.

```
#include "mex.h"
#include <math.h>
#define nsol 1000
#define N_v 1000
#define N 10
#define L 0.540920629075315
#define nfil 1000
#define ncol 1
```

- Función que se desea realizar, en este caso la función KKI para calcular los puntos predichos de la función. Aquí se deben definir como entradas a la función los regresores, las salidas del sistema procedentes del proceso de entrenamiento, y los valores que se tomarán como query points. Los índices o demás matrices auxiliares que sean necesarios para el propio cálculo se definirán aquí dentro. Esta función será llamada y ejecutada una vez se haya compilado usando el comando *mex 'NombreFuncion.c'* generando el fichero compilado con extensión .mexw64 o .mexmaci64, dependiendo del Sistema Operativo del ordenador.

Código 2.3 Funcion void mexKKI.

```
    void MexKKI( double *A, double *f, double *h,double *f_v)
{
    mwIndex i,j,k;

    double min_auxmin,max_auxmax,q,f_gorro;
    double auxpot;
    double auxmin[nsol],auxmax[nsol];

    for(j=0;j<N_v;j++){
        min_auxmin=1000;
        max_auxmax=-1000;
        for(i=0;i<N;i++){
            auxpot=0;
            for(k=0;k<ncol;k++){
                auxpot=auxpot+pow(f_v[j+k*nfil]-f[i+k*nfil],2);
            }
            auxmax[i]=h[i]-L*sqrt(auxpot);
            auxmin[i]=h[i]+L*sqrt(auxpot);

            if(max_auxmax < auxmax[i]) {
                max_auxmax = auxmax[i];
            }
```

```
            if(min_auxmin > auxmin[i]) {
                min_auxmin = auxmin[i];
            }
        }
        f_gorro=0.5*max_auxmax+0.5*min_auxmin;
        A[j]=f_gorro;
    }
}
```

- Función con parámetros y declaraciones propias a la función mex, aquí se definen los punteros de las matrices de entrada, los tamaños de estas y se declara la matriz de salida, reservando el tamaño necesario.

  Los parámetros de entrada a dicha función mex se clasifican en:

  - Nrhs : Número de argumentos de entrada o del tamaño de la matriz prhs.
  - Prhs : Matriz de argumentos de entrada.
  - Nlhs : Número de argumentos de salida o del tamaño de la matriz plhs.
  - Plhs : Matriz de argumentos de entrada.

Código 2.4 Función mexFunction.

```
        void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *
            prhs[])
{
    mwSize mrows1, ncols1; /* declaracion de variables de tamano*/
    mwSize mrows2, ncols2;
    mwSize mrows3, ncols3;

    double *f; /* puntero de las matrices f h f_v */
    double *h;
    double *f_v;

    //w
    f = mxGetPr(prhs[0]);    /* Asigna el puntero al array 1D de la
        primera entrada*/
    mrows1 = mxGetM(prhs[0]); /* toma el tamano */
    ncols1 = mxGetN(prhs[0]); /* numero de columnas de la primera matriz
        */
    //y
    h = mxGetPr(prhs[1]);   /* Asigna el puntero al array 1D de la
        segunda entrada*/
    mrows2 = mxGetM(prhs[1]);
    ncols2 = mxGetN(prhs[1]);

    //Wv
    f_v = mxGetPr(prhs[2]);   /* Asigna el puntero al array 1D de la
        segunda entrada*/
    mrows3 = mxGetM(prhs[2]);
    ncols3 = mxGetN(prhs[2]);



    /* se reserva un array de tamano mrowsxncols */
    double *A = mxCalloc(nsol, sizeof(double));
```

```
30
    /* Funcion con el orden en el que enviare los parametros */
    MexKKI(A,f,h,f_v);

    /* inicializa la salida */
35  plhs[0]=mxCreateDoubleMatrix(0,0,mxREAL);


    mxSetPr(plhs[0], A);
    mxSetM(plhs[0], nsol);//tamano matriz salida
40  mxSetN(plhs[0], ncols2);

    /* No liberar A, ya que la salida plhs[0] apunta a A */

    return;
45 }
```

Una vez compilada la función, habrá que defeninir previamente los parámetros de entrada, en este caso la matriz de regresores de entramiento y la salida del sistema correspondiente a este proceso, junto con una tercera matriz correspondiente a los puntos en los que se evaluará la función KKI o query points.

Se adjunta una tabla donde se comparan los tiempos de ejecución de Kinky Inference desarrollado en un .m y en un .mex variando el número de iteraciones calculadas durante el proceso:

Tabla 2.1 Relación tiempos de ejecución Mex-MATLAB.

Relación tiempos Mex-MATLAB			
Nº de puntos/Iteraciones	Mex (seconds)	MATLAB (seconds)	ratio matlab/mex
1000	0.0093	0.0241	2.5914
5000	0.0112	0.1222	10.9107
10000	0.0154	0.3144	20.4156
20000	0.0163	0.7054	43.2761
50000	0.0178	4.9505	278.1180
100000	0.0211	15.6567	742.0237

Destacar que para la medición de los tiempos se ha usado la función tic toc de Matalb con la media de 10 muestras ya que los tiempos de acceso a la memoria por parte de matlab disminuyen tras acceder a ella las primeras veces.

En la figura 2.6 puede apreciarse como al aumentar el número de iteraciones o puntos de operación, el programa compilado en MATLAB tiende a ralentizarse en comparación al compilado en mex, el cual se mantiene en los mismos tiempos de ejecución, convirtiéndolo en más eficiente de cara al proceso.

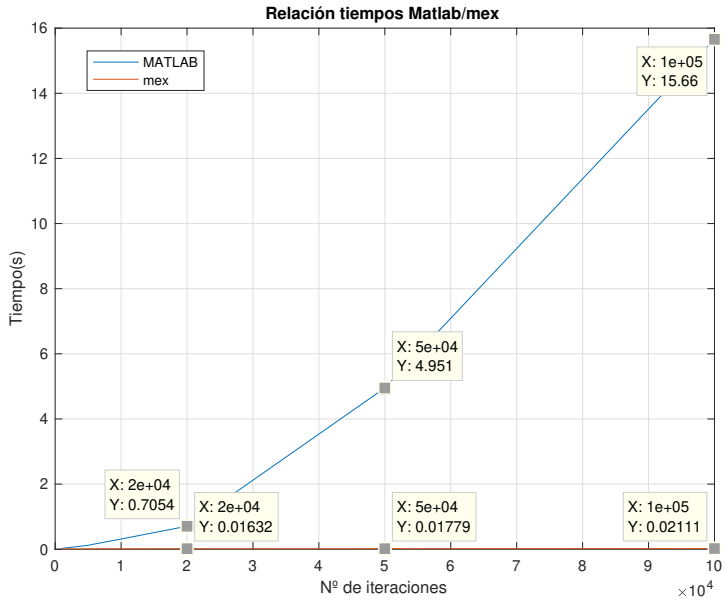


Figura 2.6 Relación tiempo de ejecución mex-matlab.

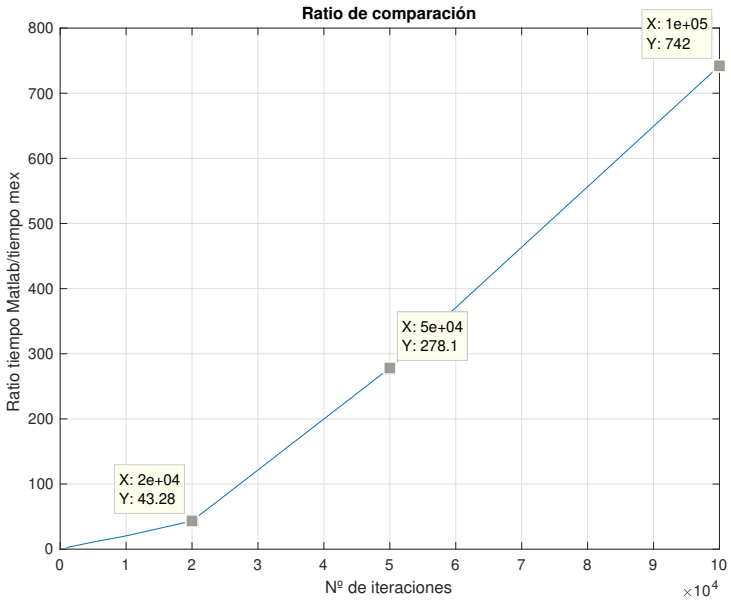


Figura 2.7 Ratio tiempo mex-matlab.

En la figura anterior 2.7 puede apreciarse los órdenes de magnitud que se obtiene al comparar el tiempo de procesado de un fichero mex frente a un fichero Matlab.

3 Identificación de sistemas no lineales

Cualquier sistema puede modelarse como un modelo NARX (*Non lineal AutoRegressive with eXternal input*). Consiste en una funcion de términos de valores pasados, una función desconocida. De tal modo que sabiendo los valores pasados que hacen falta, hay que tratar de descubrir la relación entre los valores pasados y el valor medido, y de esto se encarga el problema de aprendizaje.

Dicho problema de aprendizaje puede plantearse como un proceso gaussiano, de mínimos cuadrados o mediante el uso de regresores con Kinky Inference.

La función que gobierna el sistema es la siguiente:

$$y(k+1) = f(x_c(k), u(k)) \tag{3.1}$$

Donde x_c simboliza el estado canónico, en este caso representa cada una de las alturas de los depósitos (H1, H2 y H3). Mientras que y simboliza la salida medible del sistema, siendo lo único accesible por parte del controlador.

Como no se tienen medidas de los estados, ya que serían necesarias la función de transición y la función de salida, pero al no tener medidas para ello, hay que buscar la relación que existe entre ellas en un modelo entrada-salida.

De esto se encargan los regresores:

$$w(k) = (y(k), ..., y(k-n_a), u(k), u(k-1), u(k-n_b)) \tag{3.2}$$

Estos regresores están formados por los valores pasados de las salidas y de las entradas, los cuales deben conocerse para poder predecir los valores futuros.

3.1 Cálculo de regresores

El número de datos pasados conocidos serán el número de horizontes, generalmente llamado como *na* y *nb*. Donde na simboliza el número de datos de la salida conocidos, y nb el correspondiente a los valores de entrada.

Vease con el siguiente ejemplo de dos tanques 3.1 cuyas alturas iniciales son 1 y 4 metros, evolucionando las alturas de la siguiente manera:

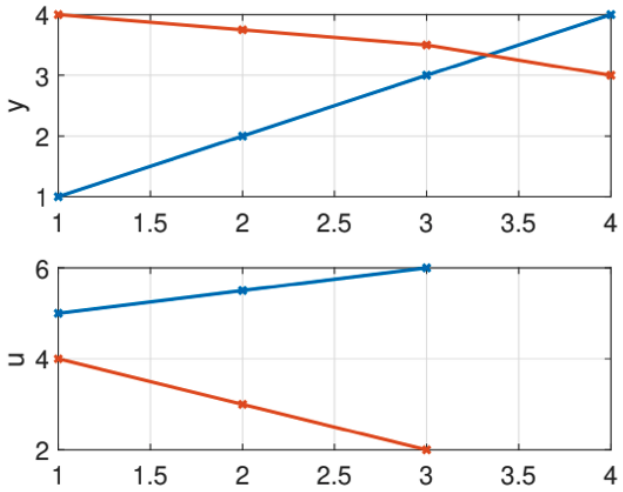


Figura 3.1 Evolución temporal del sistema.

$$y = \begin{pmatrix} 1 & 4 \\ 2 & 3.75 \\ 3 & 3.5 \\ 4 & 3 \end{pmatrix} \quad u = \begin{pmatrix} 5 & 4 \\ 5.5 & 3 \\ 6 & 2 \end{pmatrix} \tag{3.3}$$

Si los horizontes elegidos son $na = 1$ y $nb = 0$ se transformaría en:

$$w = \begin{pmatrix} 2 & 3.75 & 1 & 4 & 5.5 & 3 \\ 3 & 3.5 & 2 & 3.75 & 6 & 2 \end{pmatrix} \tag{3.4}$$

Este cálculo de la matriz de regresores se ha implementado de la siguiente manera:

Código 3.1 Función halla w.

```
function [w,z]=halla_wk(y_k,u_k,na,nb)
k=0;
na=na+1;
nb=nb+1;
for i=max(na,nb)+1:length(y_k)
    k=k+1;
    aux_w=[];
    for j=1:na
        aux_w=[aux_w,y_k(i-j+1,:)];
    end
    for j=1:nb
        aux_w=[aux_w,u_k(i-j+1,:)];
    end
    w(k,:)=aux_w;
    z(k,:)=y_k(i,:);
end
end
```

3.2 Descripción del sistema de 3 tanques

Debido a la emergencia sanitaria, el sistema de tanques del laboratorio de la facultad se tuvo que ver reemplazado por un sistema implementado en Simulink desarrollado por Manuel Gil Ortega Linares.

Este sistema consiste en 3 tanques de agua interconectados, donde (*Qe*) es el caudal de entrada expresado en l/min y la salida la altura del tercero, expresada en centímetros.

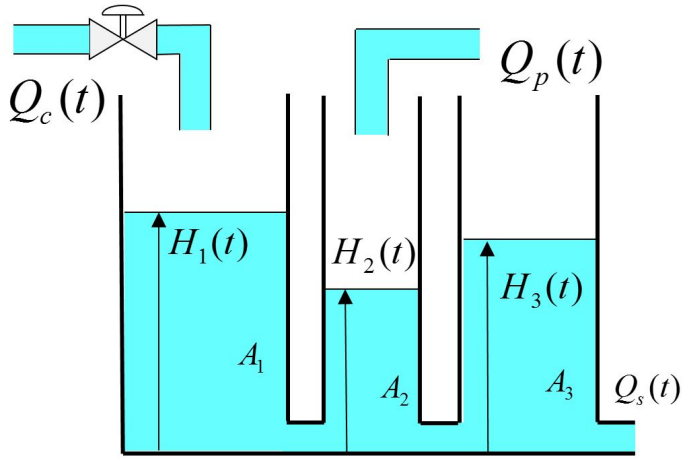


Figura 3.2 Sistema Depósitos 3 tanques.

El modelo matemático que rige el sistema está formado por las siguientes ecuaciones:

$$A_1 \cdot \frac{dH_1}{dt} = Q_e - signo(H_1 - H_2) \cdot k_1 \sqrt{|H_1 - H_2|} \tag{3.5}$$

$$A_2 \cdot \frac{dH_2}{dt} = Q_p - signo(H_1 - H_2) \cdot k_1 \sqrt{|H_1 - H_2|} - signo(H_2 - H_3) \cdot k_2 \sqrt{|H_2 - H_3|} \tag{3.6}$$

$$A_3 \cdot \frac{dH_3}{dt} = signo(H_2 - H_3) \cdot k_2 \sqrt{|H_2 - H_3|} - k_3 \sqrt{|H_3|} \tag{3.7}$$

Donde H_i referencia las alturas de cada depósito, A_i el área, k_i las ganancias, y Q_e y Q_p los caudales de entrada y perturbación.

Los valores empleados en el sistema por defecto son los siguientes:

Tabla 3.1 Valor de las variables originales del sistema.

Variable	Valor
k_1	$11.78 \frac{m^3}{s \cdot \sqrt{Pa}}$
k_2	$10.10 \frac{m^3}{s \cdot \sqrt{Pa}}$
k_3	$10 \frac{m^3}{s \cdot \sqrt{Pa}}$
A_1	$40\ cm^2$
A_2	$10\ cm^2$
A_3	$30\ cm^2$

A su vez la característica estática del sistema cuando aplicamos señales de control constantes es la siguiente 3.3:

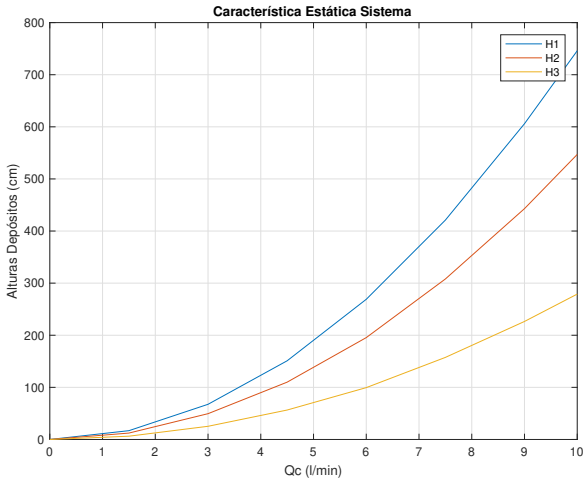


Figura 3.3 Característica Estática del Sistema.

Por todo lo anterior puede apreciarse que se trata de un sistema no lineal.

3.3 KKI aplicado al sistema de tres tanques

A continuación se explica cómo se puede adaptar Kinky Inference al estudio del sistema de tres tanques, ya que cualquier modelo dinámico se puede poner como una función no lineal de las salidas siguientes en función de los valores pasados de entradas y salidas.

3.3.1 Proceso de entrenamiento

Primero es necesario realizar un entrenamiento con el sistema para poder tener un dataset bastante completo dentro del rango de entradas permitido, en este caso, entre 0 y 10 litros por minuto. Para tener la nube de puntos más dispersa y que cubra la mayor parte de las zonas posibles, se ha utilizado una concatenación de señales chirp cuyas frecuencias pueden ser calculadas mediante ensayos de relés o incluso de manera experimental observando que la señal chirp se va atenuando a la vez que se va desarrollando, obteniendo el siguiente dataset de entrenamiento 3.4:

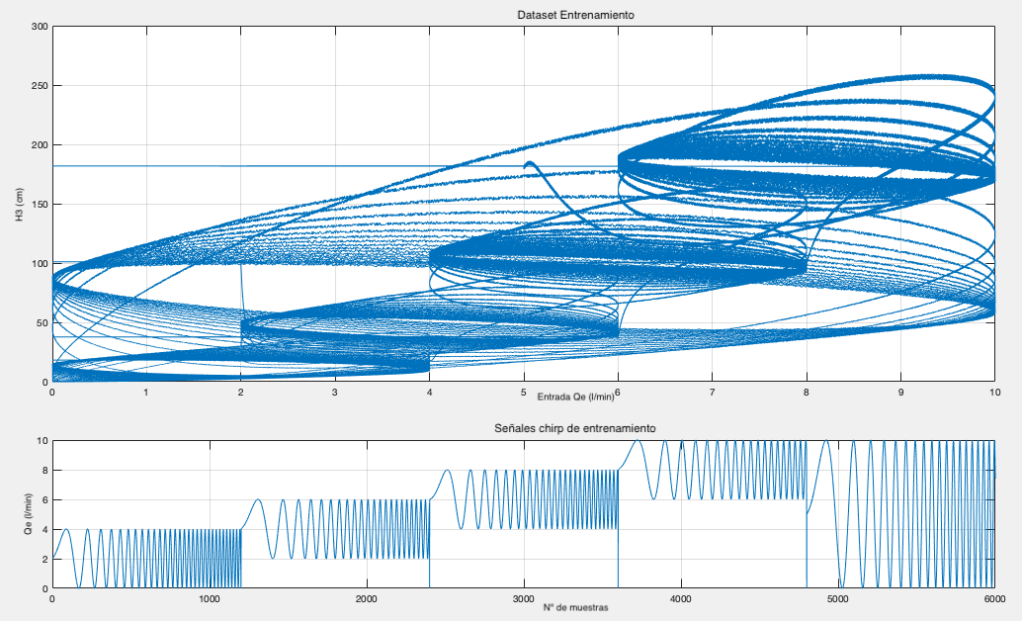


Figura 3.4 BBDD de entrenamiento.

Mientras mayor sea la zona ocupada en el dataset de entrenamiento, será mejor el resultado de la predicción con los datos de validación.

3.3.2 Proceso de validación

Una vez realizado el proceso de entrenamiento, se pasa al de validación o test, en este paso el sistema recibe una señal distinta a las recibidas durante el entrenamiento, en este caso se ha elegido una señal chirp con una frecuencia y un punto de operación que no había sido usado por ninguna de las chirps de entrenamiento.

La Base de Datos de validación corresponde a la figura 3.5:

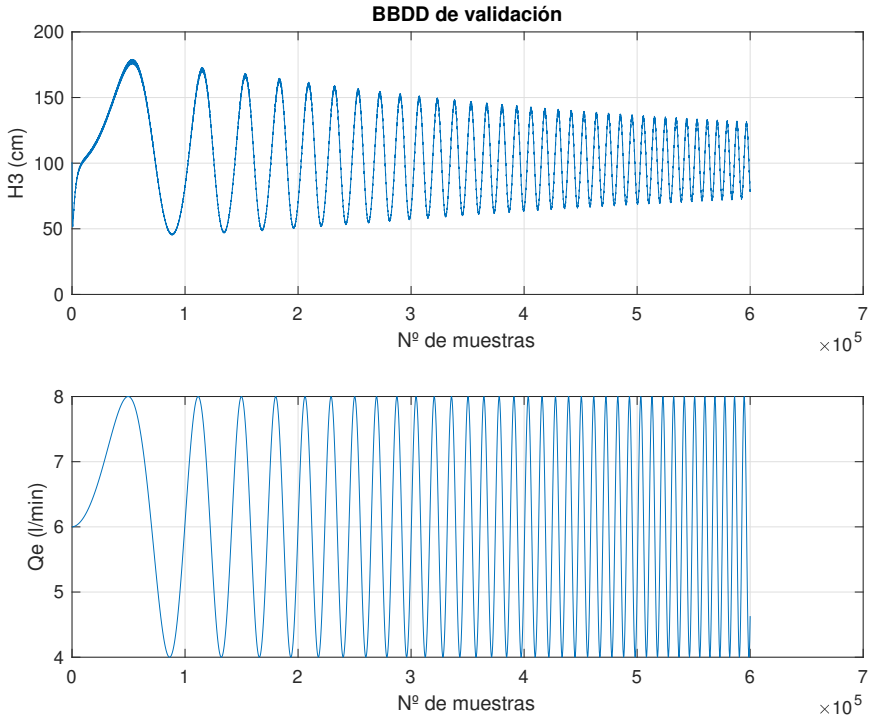


Figura 3.5 BD de validación.

El tiempo de muestreo en ambos procesos, tanto en entrenamiento como en validación, es de 0.001s, generando un total de 600.000 muestras. Este tiempo de muestreo no se ajusta correctamente a la velocidad a la que evoluciona el sistema y genera una cantidad excesiva de datos llegando a meter ruido, por ello, se decide aumentar el tiempo de muestreo a 0.5s, obteniendo un total de 1200 muestras para ambos procesos.

Como los regresores dependen del número de horizontes seleccionados, esto puede hacer que, dependiendo del número seleccionado, las predicciones tengan un error mayor o menor en comparación con los valores reales. Es por ello que deberán calcularse dichos regresores con varios horizontes (generalmente basta con elegir un número entre 0 y 3 horizontes)

El cálculo de la constante de Lipschitz será igual aunque en este caso se caracteriza por tomar el percentil correspondiente al 99 % para poder evitar posibles ruidos o puntos dispersos que afecten negativamente a la predicción. Por otro lado, los query points con los que se irá evaluando la función son los regresores de validación.

De tal modo que el cálculo de Kinky Inference se realizará de la siguiente manera:

Código 3.2 KKI de validación.

```
Q_v=[];
SOL_v=[];
v_dist=[];
N_v=N;
ny=1;
for j=1:N_v
    q=Wv(j,:)'; %query points: regresores de validacion
    max1=-10000*ones(ny,1);
    min1=10000*ones(ny,1);
    dist_min=10000;
    if(i~=j)
    for i=1:N
        max1=max(max1,Y(i,:)'-L*norm(q-W(i,:)'));
        min1=min(min1,Y(i,:)'+L*norm(q-W(i,:)'));
        dist_min=min(dist_min,norm(q-W(i,:)'));
    end
    end
    v_dist=[v_dist;dist_min];
    f_gorro=0.5*max1+0.5*min1;
    Q_v=[Q_v;q];
    SOL_v=[SOL_v;f_gorro];
end
```

Tras calcular el error absoluto para cada combinación de horizontes, se escoge la combinación cuyo percentil de 99 % error es menor.

En este caso se han obtenido los siguientes errores:

Tabla 3.2 Errores cometidos según número de horizontes.

na	nb	L	error abs (cm)
0	0	4.4351	5.5273
0	1	3.6204	5.1179
0	2	3.4038	5.5394
0	3	3.2271	6.1790
1	0	2.3276	8.8684
1	1	2.2645	8.5397
1	2	2.2417	8.6719
1	3	2.2058	8.6488
2	0	1.8529	12.1467
2	1	1.8216	11.6128
2	2	1.8154	11.6240
2	3	1.8048	11.5742
3	0	1.5669	13.4677
3	1	1.5482	13.1296
3	2	1.5433	13.2703
3	3	1.5409	13.1352

Dichos errores absolutos son respecto a valores máximos de salida de 178.9876 cm. La combinación de horizontes con menor error absoluto sería: $na = 0$, $nb = 1$ (lo que sería un error relativo de 2.8594 %).

En la figura 3.6 se muestra el histograma con los errores cometidos durante el proceso de validación con $na = 0$ y $nb = 1$.

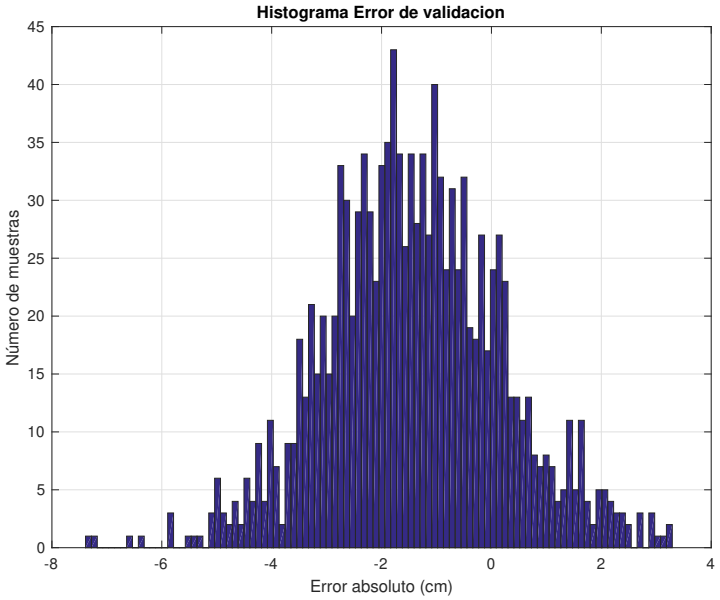


Figura 3.6 Histograma de error de validación para $na = 0$ y $nb = 1$.

4 Predictor

Una vez se dispone de los procesos de cálculo de regresores y de Kinky Inference para el cálculo de valores futuros, se desea integrar todo en busca de un algoritmo capaz de calcular las predicciones del sistema contando únicamente con unas muestras iniciales previamente conocidas y con una señal de control también conocida elegida por el usuario.

4.1 Predictor a N pasos

La idea del algoritmo de este predictor es que, a partir de esas muestras conocidas tanto de salida como de entrada, se puedan ir construyendo los correspondientes regresores y comenzar a predecir, de tal modo que las muestras conocidas más antiguas se vayan reemplazando por los nuevos valores predichos, generando nuevos regresores a raiz de estas predicciones. De este modo se pueden realizar predicciones a N pasos.

El código capaz de implementar tal algoritmo sería el siguiente:

Código 4.1 Función predictor.

```
clear
load('kki_test_na0nb1.mat') %variables guardadas de proceso de KKI
na=0;
nb=1;

Muestra_conocida_ini=3;
k_ini=Muestra_conocida_ini-max(na+1,nb+1)+1;
k_fin=Muestra_conocida_ini+1;

%datos conocidos
salidas_conocidas=H3r(k_ini:k_fin,:);
u_conocidas=Qcr(k_ini:k_fin);
w_conocidas=halla_wk(salidas_conocidas,u_conocidas,na,nb); %regresores
    conocidos
Uf=Qcr(k_ini:end);
L=5;
pred=[salidas_conocidas]; %lista donde acumulare las predicciones de la salida

N=size(Y,1);
```

```
20
for j=1:length(W)-max((nb+1),(na+1))
        min1=10000;
        max1=-10000;
        q= w_conocidas(1,:);
25      for i=1:N
            max1=max(max1,Y(i,:)'-L*norm(q-W(i,:)));
            min1=min(min1,Y(i,:)'+L*norm(q-W(i,:)));
        end

30      f_gorropred=0.5*max1+0.5*min1;

    pred=[pred;f_gorropred];
    k_ini_j=j+1;
    k_fin_j=j+max(na+1,nb+1);
35
    y_j=[pred(k_ini_j:k_fin_j);f_gorropred];
    u_j=Uf(k_ini_j:k_fin_j+1);
    w_conocidas=halla_wk(y_j,u_j,na,nb);
end
40 plot(H3r([2:end]))
hold on
plot(pred)
```

Como se puede apreciar en el ejemplo, el proceso de predicción parte de la muestra inicial 3, por lo que, al tener un horizonte 0 de salidas y 1 de entradas (realmente esto sería 1 y 2 correspondiente, al haber sido programado con el incremento de una unidad en el código de hallar los regresores), es por ello que el valor del dato número 4, depende de la y(3) y de la u(2),u(3).

Por tanto los resultados obtenidos con dicho predictor y la señal de test utilizada anteriormente (gráfica test: 3.5) se pueden apreciar en la siguiente figura 4.1:

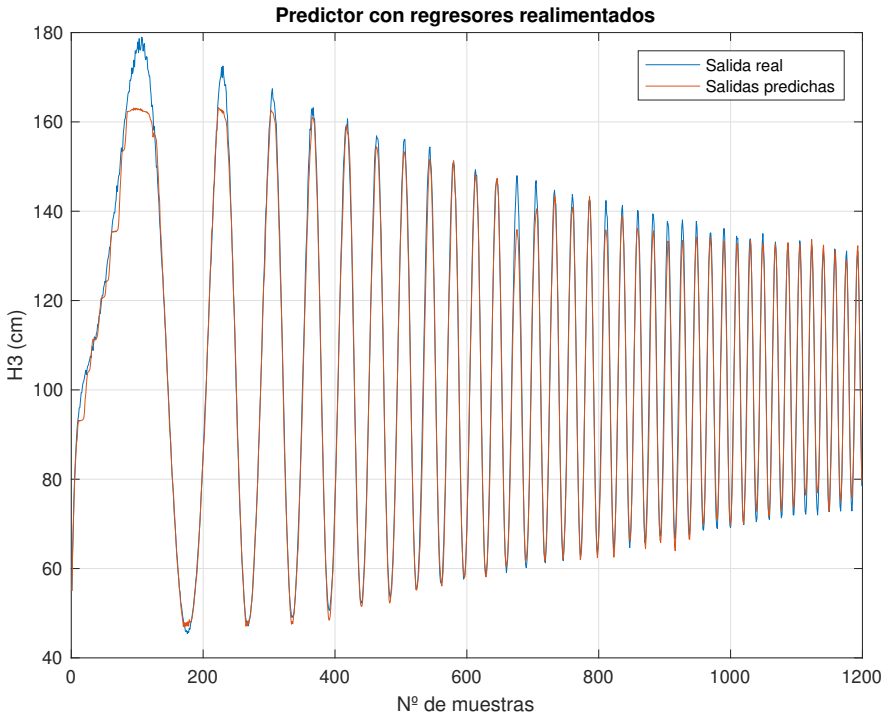


Figura 4.1 Predicción de la señal test.

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] Paloma Recuero de los Santos. Tipos de aprendizaje en machine learning: supervisado y no supervisado, November 2017. URL *https://empresas.blogthinkbig.com/que-algoritmo-elegir-en-ml-aprendizaje/*.

[2] D. Muñoz de la Peña y J. Callies J.M. Manzano, D. Limon. *Output Feedback MPC based on Smoothed Projected Kinky Inference*. The Institution of Engineering and Technology, 1ª edition, 2019.

[3] Dot CSV. ¿qué es el machine learning?¿y deep learning? un mapa conceptual | dotcsv, November 2017. URL *https://www.youtube.com/watch?v=KytW151dpqU*.

[4] Alberto López Estrada. How to create c source mex file, November 2020. URL *https://stackoverflow.com/questions/64977290/how-to-create-c-source-mex-file?noredirect=1#comment114934819_64977290*.

Índice alfabético