Trabajo de Fin de Grado

Grado en Ingeniería de Tecnologías Industriales

Implementación y pruebas de un algoritmo de predicción intervalar

Autor: Pablo José González Bizcocho

Tutor: Daniel Rodríguez Ramírez

Alfonso Daniel Carnerero Panduro

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

Sevilla, 2022

Trabajo de Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Implementación y pruebas de un algoritmo de predicción intervalar

Autor:
Pablo José González Bizcocho

Tutor:
Daniel Rodríguez Ramírez
Profesor titular
Alfonso Daniel Carnerero Panduro

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Proyecto Fin de Carrera: Implementación y pruebas de un algoritmo de predicción intervalar

Autor: Pablo José González Bizcocho

Tutor: Daniel Rodríguez Ramírez
Alfonso Daniel Carnerero Panduro

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Este trabajo pone la puntilla a una bonita etapa de mi vida donde he aprendido muchos conocimientos matemáticos, físicos, químicos… pero sobre todo he crecido mucho mentalmente: saber tranquilizarme en momentos duros, entender que no todo sale a la primera (de hecho, es sospechoso cuando sale a la primera), a saber encajar las críticas para aprender de ellas y a tomarme la vida como un continuo aprendizaje. Todo ello gracias a diversos profesores que siempre ayudaban, a mis amigos y sobre todo a mi familia que siempre ha sido el mayor pilar donde respaldarme.

De las cosas más importantes que me llevo de este grado es el haber aprendido a pensar por mí mismo para así encontrar una solución y buscarme los métodos necesarios para ejecutarla.

Por último, agradecer a Daniel y a Alfonso por toda la atención que me han prestado durante este trabajo y por todo el tiempo invertido en mí.

Resumen

El objetivo principal de este trabajo de fin de grado es desarrollar un algoritmo de predicción de intervalos eficiente. Se tendrá una entrada x_k y se pretende hallar un intervalo que contenga a la salida y_k con cierta probabilidad especificada. Además, para que dicho intervalo calculado sea útil deberá tener el menor tamaño posible.

Se introducirá el concepto de función de disimilitud, que simplemente mide la semejanza entre un punto/vector y un conjunto de datos. Para este TFG se presentará una familia de funciones de disimilitud específica que resuelven un problema de optimización convexa. Con el cálculo de la disimilitud se procederá a buscar una función de distribución capaz de proporcionar los intervalos de predicción buscados, usando una metodología que se definirá paso a paso. Se pretende implementar diferentes algoritmos que sean capaces de devolver intervalos de predicción precisos cumpliendo con la probabilidad requerida.

Otra idea principal de este trabajo es la implementación en Matlab, al ser la herramienta más empleada durante el grado y estar al alza durante los últimos años en muchas empresas que necesitan resolver simulaciones de diferentes tipos. Se pretende que el código implementado en Matlab sea eficiente ya que los algoritmos a resolver tienen un elevado coste computacional por ello se buscará en cualquier caso la mejor opción para ser empleado en Matlab.

También se comparará el método de predicción implementado con otro método con el que se puede realizar lo mismo, la Regresión Cuantílica. De este modo se pretenderá dar valor a la importancia y la no trivialidad de implementar un algoritmo de predicción de intervalos robusto y eficiente pese a tener un elevado coste computacional debido al tratamiento de multitud de datos simultáneamente.

Abstract

The main objective of this final project is to develop an efficient interval prediction algorithm. We will have an input x_k and we intend to find an interval that contains the output y_k with a certain specified probability. Furthermore, for such a calculated interval to be useful it should have the smallest possible size.

The concept of dissimilarity function, which simply measures the similarity between a point/vector and a data set, will be introduced. For this TFG, a family of specific dissimilarity functions that solve a convex optimization problem will be presented. With the calculation of the dissimilarity, we will proceed to search for a distribution function capable of providing the prediction intervals sought, using a methodology that will be defined step by step. It is intended to implement different algorithms that are able to return accurate prediction intervals complying with the required probability.

Another main idea of this project is the implementation in Matlab, being the most used tool during the studies and being on the rise during the last years in many companies that need to solve simulations of different types. It is intended that the code implemented in Matlab is efficient since the algorithms to be solved have a high computational cost, so the best option to be used in Matlab will be sought in any case.

We will also compare the prediction method implemented with another method with which the same thing can be done, the quantile regression. In this way, the importance and non-triviality of implementing a robust and efficient interval prediction algorithm will be emphasized, despite the high computational cost due to the simultaneous processing of a multitude of data.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

A^*	Conjugado
c.t.p.	En casi todos los puntos
c.q.d.	Como queríamos demostrar
■	Como queríamos demostrar
e.o.c.	En cualquier otro caso
e	número e
IRe	Parte real
IIm	Parte imaginaria
sen	Función seno
tg	Función tangente
arctg	Función arco tangente
sen	Función seno
$\sin^x y$	Función seno de x elevado a y
$\cos^x y$	Función coseno de x elevado a y
Sa	Función sampling
sgn	Función signo
rect	Función rectángulo
Sinc	Función sinc
∂y ∂x	Derivada parcial de y respecto
$x_\circ$	Notación de grado, x grados.
Pr(A)	Probabilidad del suceso A
SNR	Signal-to-noise ratio
MSE	Minimum square error
:	Tal que
<	Menor o igual
>	Mayor o igual
\	Backslash
⇔	Si y sólo si
TFG	Trabajo de Fin de Grado

1 INTRODUCCIÓN

Se ha vuelto terriblemente obvio que nuestra tecnología
ha superado nuestra humanidad

-Albert Einstein-

La predicción de datos se ha vuelto de vital importancia en estos últimos tiempos, puede verse aplicada en multitud de campos científico/tecnológicos de nuestra sociedad, tales como la predicción meteorológica, el control automático… Incluso también podemos observarla en el ámbito económico: analizar el comportamiento del consumidor, identificar el riesgo, hacer una predicción de la demanda, realización operaciones en bolsa… Se observa en esta herramienta que a principios de su descubrimiento solo se le veía una aplicación tecnológica, pero ha pasado a ser una herramienta fundamental en la toma de decisiones de nuestra sociedad. En el ámbito empresarial, tener un algoritmo de predicción de datos robusto y eficiente puede marcar una diferencia notable frente a otras empresas de la competencia. Si a esta necesidad de obtener buenas predicciones le sumamos la mejora computacional de la tecnología, hace que podamos tener bases de datos lo suficientemente grandes como para obtener predicciones realmente buenas.

En el desarrollo de este trabajo vamos a desarrollar un algoritmo de predicción que nos dará una "predicción intervalar", es decir, obtener un intervalo lo más pequeño posible que contenga nuestra salida futura. Más concretamente, dada una entrada x_k obtener con una probabilidad prestablecida el menor intervalo posible que contenga la salida y_k. Para ello vamos a plantear una metodología para obtener intervalos de predicción de un sistema dinámico mediante el uso de una familia de funciones de disimilitud propuesta por Alfonso Daniel Carnerero y Daniel Rodríguez Ramírez.

Todo el trabajo se desarrollará en la plataforma de programación y calculo numérico Matlab, la cual nos permite un fácil análisis de resultados e implementar problemas de cálculo numérico y optimización con grandes bases de datos fácilmente, haciendo uso de un ordenador adecuado.

1.1 Motivación

Las predicciones de intervalos han tomado bastante importancia en estos últimos años, desde estudios de previsión previos a la inversión de un proyecto (demanda, generación energética, valor económico…) hasta la gestión óptima de recursos en empresas energéticas.

Para la implementación de un algoritmo de predicción de intervalos es necesario la resolución de un problema de optimización convexa para hallar la disimilitud que hay entre el punto a predecir y el set de datos disponible. El mayor problema de estos algoritmos es el tiempo que tardan en converger a una solución debido a la gran

cantidad de datos utilizados. Debido a ello en este Trabajo de Fin de Grado vamos a analizar diferentes formas de obtener la disimilitud de modo que emplearemos la más eficiente para hallar una solución a nuestro problema de hallar una predicción de intervalos con cierta probabilidad predefinida.

1.2 Objetivos

El objetivo principal de este TFG es elaborar un algoritmo para obtener el intervalo más pequeño posible que contenga a la salida de nuestro sistema con cierta probabilidad preestablecida.

Previamente al objetivo principal analizaremos diferentes métodos para obtener la disimilitud entre un punto a predecir y un set de datos definido, comparándolos para así poder usar el más eficiente para operar con un número elevado de datos.

Posteriormente, aplicaremos el algoritmo que vamos a implementar en Matlab a 2 bases de datos diferentes y analizaremos los resultados.

Por último, vamos a comparar el método programado con el método de "quantile regression" para observar las diferencias de rendimiento obtenido.

1.3 Alcance

La elaboración del TFG se va a realizar buscando un punto de vista de eficiencia para su uso en Matlab.

Para ello primero se va a explicar y calcular la disimilitud a través de diferentes métodos comparando los tiempos de cálculo en cada caso, eligiendo el método más rápido posible y aplicándolos con un ejemplo sencillo de una elipse formada por una nube de puntos. Se presentará una familia de funciones de disimilitud para el cálculo de predicciones de intervalos y se probará su funcionamiento con los diferentes métodos.

Después se presentará el cálculo de las predicciones de intervalos con una metodología definida en la bibliografía. Usaremos las funciones de disimilitud en el contexto de la regresión para hallar funciones de densidad de probabilidad empírica de la salida y_k condicionada a nuestro regresor x_k. Presentaremos el algoritmo 1 del método que nos devolverá una predicción de intervalos en función de 2 parámetros: γ y c. Tras ello veremos el algoritmo 2 que sirve para afinar el valor del parámetro c de modo que el intervalo sea el menor posible para una probabilidad previamente definida. También veremos un método que nos devuelve el valor óptimo de γ para cierta c dada. De esta forma buscaremos la pareja de parámetros γ y c óptima para nuestro problema.

Por último, se probará el algoritmo implementado con 2 bases de datos diferentes buscando predecir la salida con diferentes probabilidades y hallar la pareja de parámetros γ y c óptima en cada caso. También se comparará el método estudiado con el método de "quantile regression" proporcionado por los tutores del TFG para así compararlos entre sí, comentado ventajas y desventajas.

Se comentan algunas posibles ampliaciones del proyecto las cuales no serán abordadas en este documento: implementar una aplicación de control predictivo e implementar un regresor adaptativo según el ejemplo a tratar.

1.4 Metodología

Durante todo el desarrollo del proyecto se usará la herramienta de cálculo y programación Matlab, la cual resulta idónea para este tipo de proyectos, es capaz de ejecutar algoritmos de un elevado cálculo numérico sin problemas con un ordenador adecuado. También nos aporta una manera sencilla de mostrar resultados y compararlos con otros métodos ejecutados.

Se partirá de un artículo científico elaborado por Alfonso Daniel Carnerero y proporcionado por Daniel Rodríguez Ramírez, a partir del cual se implementarán sus definiciones y métodos para diferentes ejemplos y bases de datos buscando siempre la eficiencia en Matlab. Las bases de datos usadas serán: base de datos a partir del atractor de Lorenz y otra del índice bursátil SP500, ambas proporcionadas por los tutores del Trabajo de Fin

de Grado.

2 FUNCIONES DE DISIMILITUD Y SIMILITUD

Cada día sabemos más y entendemos menos

-Albert Einstein-

Para el desarrollo del objetivo principal del TFG resulta indispensable explicar que son las funciones de disimilitud y su aplicación en el campo de la regresión. Presentaremos la familia de funciones de disimilitud que será usada en el desarrollo de este trabajo para obtener la predicción de intervalos que contienen la salida requerida.

Además, presentaremos diferentes métodos para calcular la similaridad o disimilaridad en Matlab, comparándolos entre sí para poder usar el más eficiente para tratar un número elevado de datos. Dichos métodos serán puestos a prueba para un ejemplo de una elipse que será presentado posteriormente.

2.1 Concepto de funciones de disimilitud y similitud

Dado un set de datos tal que: $D = \{z_i : i = 1, \dots, N\} \subset \mathbb{R}^n$, las funciones de disimilitud nos ofrecen la posibilidad de determinar si un vector o un escalar z puede ser considerado similar a otros vectores o escalares del set de datos D. La función de disimilitud vendrá definida por:

$$J_d(.,.): \quad \mathbb{R}^n \; x \; D \rightarrow [0, \infty] \,,$$

y nos devolverá la disimilitud entre un punto z y un set de datos D. Valores elevados de $J_d(z, D)$ representan grandes diferencias entre z y D, mientras que valores pequeños de $J_d(z, D)$ representan un alto grado de similitud entre z y D. Existen muchos operadores que trabajan como funciones de disimilitud, si tomamos el caso específico en el que el set de datos D está formado por un único vector: $D = \{z_D\}$ una función de disimilitud podría ser:

$$J_d(z, z_D) = \| \; z - z_D \; \| \,,$$

donde $\| \cdot \|$ se conoce como norma y sirve para cuantificar el espacio entre 2 vectores, de modo que cuanto menor sea su valor, más parecidos serán ambos vectores entre sí, y viceversa.

Partiendo de una función de disimilitud $J_d(x\,, D)$ se puede obtener una función de similitud $J_s(x, D)$ del

siguiente modo:

$$J_s(x, D) = e^{-\sigma J_{d(z,D)}}, \sigma > 0 \tag{2–1}$$

Donde si el valor de $J_s(x, D)$ es pequeño significará que x no es similar a D, y si es elevado entonces x será similar a D.

Funciones de disimilitud y similitud pueden ser usadas en el contexto de agrupación y organización de diferentes datos, ejemplos de ello podrían ser el clustering o la clasificación de imágenes en fábricas a la hora de clasificar productos con diferentes forma o características medibles.

2.2 Familia de funciones de disimilitud a emplear

Primeramente, es importante comentar que la familia que se va a proponer tiene la propiedad de ser invariante con respecto a transformaciones afines, de este modo se garantiza que el análisis empleado no se verá afectado por la elección de un sistema de coordenadas. Esta singularidad no es común en la mayoría de las familias de funciones de disimilitud, por ejemplo: cualquier función de disimilitud basada en la distancia entre un punto z y un set de datos D depende de la elección del sistema de coordenadas.

Según [1] se pretende resolver un problema de optimización convexa con el que obtener la medida de la disimilitud entre un punto z y una base de datos D. Dicho problema de optimización queda presentado como:

Dado un set de datos $D = \{z_i : i = 1, \dots, N\} \subset \mathbb{R}^n$ y un escalar $\gamma \geq 0$ definimos la función de disimilitud $J_\gamma(z, D)$ como:

$$J_\gamma(z, D) = \min_{\lambda_1, \dots, \lambda_N} \sum_{i=1}^{N} \lambda_i^2 + \gamma \sum_{i=1}^{N} |\lambda_i|$$

$$s.t \quad z = \sum_{i=1}^{N} \lambda_i z_i \tag{2–2}$$

$$1 = \sum_{i=1}^{N} \lambda_i$$

Se observa que el problema de optimización a resolver está sujeto a 2 restricciones de igualdad, buscando obtener los valores óptimos de $\lambda = \{\lambda_i : i = 1, \dots, N\}$ que minimicen la función de coste mostrada sujeta a ambas restricciones. El parámetro γ será optimizado en apartados posteriores.

2.3 Funciones de disimilitud en el contexto de regresión

Para desarrollar el algoritmo de este trabajo se usarán las funciones de disimilitud en el contexto de la regresión para así calcular funciones de densidad empíricas condicionadas.

Suponiendo que tenemos un par $\{x_i, y_i\}$, donde $i = 1, \dots, N$, y que queremos estimar y teniendo como dato x. Una posible estimación de y podría ser:

$$\hat{y} = \sum_{i=1}^{N} \lambda_i y_i$$

Donde los escalares λ_i serán calculados haciendo uso de una función de similitud $J_s(\cdot,\cdot)$, de modo que si el valor de λ_i es pequeño se debe a que el valor de la función de similitud para ese punto también es pequeño. Para el cálculo de λ_i se pueden implementar diferentes aproximaciones más o menos sofisticadas en función del uso que se le vaya a dar.

A continuación, se adapta la familia de funciones de disimilitud a emplear en el contexto de la regresión para elaborar el algoritmo de predicción de intervalos deseado.

Suponiendo que tenemos un set de datos $D = \left\{ z_i = \begin{bmatrix} y_i \\ x_i \end{bmatrix} : i = 1, \dots, N \right\} \subset \mathcal{Y}\, x\, \mathcal{X}$, formado por las entradas y salidas de un sistema. Dados x_k como un punto de entrada y un parámetro $\gamma \geq 0$, se podría obtener una estimación $\widehat{y_k}$ de la salida y_k minimizando la función de disimilitud entre el vector $\begin{bmatrix} y \\ x_i \end{bmatrix}$ y el set de datos D:

$$\widehat{y_k} = \arg\min_{y} J_\gamma\left(\begin{bmatrix} y \\ x_k \end{bmatrix}, D\right), \tag{2–3}$$

viendo esto podríamos obtener la estimación $\widehat{y_k}$ resolviendo el problema de optimización mostrado en el apartado anterior del siguiente modo:

$$\min_{y,\lambda_1,\dots,\lambda_N} \sum_{i=1}^{N} \lambda_i^2 + \gamma \sum_{i=1}^{N} |\lambda_i|$$

$$s.t \quad \begin{bmatrix} y \\ x_k \end{bmatrix} = \sum_{i=1}^{N} \lambda_i \begin{bmatrix} y_i \\ x_i \end{bmatrix} \tag{2–4}$$

$$1 = \sum_{i=1}^{N} \lambda_i$$

A partir de la primera restricción de la ecuación (2-4) se observa que la variable y solo aparece en dicha restricción en todo el problema, entonces se podría simplificar el problema ignorando dicho límite:

$$y = \sum_{i=1}^{N} \lambda_i y_i$$

De este modo el valor óptimo de y se podría obtener del siguiente modo:

$$\widehat{y_k} = y^* = \sum_{i=1}^{N} \lambda_i^* y_i \tag{2–5}$$

donde λ_i^*, i=1,…,N, son los valores obtenidos del problema de optimización:

$$\min_{\lambda_1,\dots,\lambda_N} \sum_{i=1}^{N} \lambda_i^2 + \gamma \sum_{i=1}^{N} |\lambda_i|$$

$$s.t \qquad x_k = \sum_{i=1}^{N} \lambda_i x_i \tag{2–6}$$

$$1 = \sum_{i=1}^{N} \lambda_i$$

De este modo, la estimación obtenida por este método es una combinación lineal de las salidas y_i. En apartados posteriores, veremos como implementaremos dicho problema y su papel para calcular funciones de densidad empíricas.

2.4 Métodos de implementación en Matlab.

Para el desarrollo de este TFG se usará solamente Matlab. En este apartado trataremos de ver diferentes métodos y funciones que podemos usar para resolver el problema de optimización convexa presentado anteriormente.

Matlab ofrece multitud de “toolboxes” para diferentes aplicaciones, para este TFG se usará “Optimization Toolbox” de Matlab que permite resolver problemas de optimización mediante funciones capaces de calcular minimizaciones y/o maximizaciones de objetivos sujetas a restricciones variadas. Incluye funciones para programación lineal, programación lineal entera mixta, programación cuadrática, programación cónica de segundo orden, programación no lineal, mínimos cuadrados lineales con restricciones, mínimos cuadrados no lineales y ecuaciones no lineales.

En este apartado se pretende presentar diferentes métodos para implementar el problema de optimización convexa previamente definido en (2-6), se van a analizar las características de 3 métodos distintos que nos resuelven el problema de optimización planteado.

2.4.1 Función fmincon

Función proporcionada por “Optimization Toolbox” de Matlab, capaz de encontrar el mínimo de una función multivariable no lineal restringida. Encuentra el mínimo de un problema especificado por:

$$\min_x f(x) \ sujeto\ a \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ Ax \leq b \\ Aeq\ x = beq \\ lb \ \leq x \leq \ ub \end{cases}$$

Donde b y beq son vectores, A y Aeq son matrices, $c(x)$y $ceq(x)$ son funciones que devuelven matrices, y $f(x)$ es una función que devuelve un escalar. $f(x), c(x)\ y\ ceq(x)$ pueden ser funciones no lineales, $x, lb\ y\ ub$ se pueden pasar como vectores o matrices. Sabiendo el tipo de problema que resuelve dicha función es fácil ver que es aplicable a nuestro problema de optimización, donde $f(x)$ sería la función a minimizar que deberá ser declarada antes de su uso en Matlab

La implementación en Matlab la veremos más adelante.

2.4.2 Función quadprog

Función proporcionada por "Optimization Toolbox" de Matlab, resuelve funciones objetivo-cuadráticas con límites lineales. Es capaz de encontrar el mínimo para un problema especificado por:

$$\min_x \frac{1}{2} x^T Hx + \ f^T x \ \ sujeto\ a \begin{cases} Aeq\ x = beq \\ Ax \leq b \\ lb \ \leq x \leq \ ub \end{cases}$$

Donde $H, A, y\ Aeq$ son matrices, y x, $f, b, beq, \ lb\ y\ ub$ son vectores. Vemos que en este caso se plantea la función a minimizar desde el inicio, de modo que para aplicarlo al caso en cuestión debemos crear una matriz H y un vector f adecuados para que el problema requerido no difiera del resuelto con *quadprog*. Otro problema para resolver de este método sería el cómo tratar el valor absoluto de λ_i en (2-2) con la función propuesta en *quadprog*.

El problema del valor absoluto se resuelve mediante la técnica expuesta en [2], concluyendo con la obtención de 2 nuevas restricciones. El problema de optimización a resolver resultaría:

$$J_\gamma(z, D) = \min_{\lambda_1,\dots,\lambda_N} \sum_{i=1}^{N} \lambda_i^2 + \gamma \sum_{i=1}^{N} |\lambda_i| = \min_{\lambda_1,\dots,\lambda_N} \sum_{i=1}^{N} \lambda_i^2 + \gamma \sum_{i=1}^{N} \lambda_i abs$$

$$s.t \quad z = \sum_{i=1}^{N} \lambda_i z_i \tag{2–7}$$

$$1 = \sum_{i=1}^{N} \lambda_i$$

$$-\lambda \leq \lambda abs$$

$$\lambda \leq \lambda abs$$

Añadiendo 2 restricciones de desigualdad queda resuelto el problema del valor absoluto. Ahora se crea la matriz H y el vector f para adaptarlo a nuestro problema de optimización:

- Matriz H: correspondiente a los términos cuadráticos del problema, λ_i. Al añadir una variable nueva λabs, el tamaño de la matriz cuadrada será el doble de los componentes del conjunto de datos donde la primera mitad de la diagonal será entera de 2 y la segunda mitad 0. A continuación se muestra un ejemplo de la forma de H para un set de datos de 3 componentes

$$H = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Vector f: correspondiente a los términos lineales del problema, $\lambda_i abs$. Su tamaño también será el doble de los componentes del set de datos, pero en este caso la primera mitad del vector valdrá 0 y la segunda mitad el valor del parámetro γ. A continuación, se muestra un ejemplo de la forma del vector f para un set de datos de 3 componentes:

$$f = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \gamma \\ \gamma \\ \gamma \end{bmatrix}$$

Con todo esto ya se podría implementar la función quadprog para el problema de optimización convexa planteado en este TFG en Matlab.

2.4.3 Algoritmo FISTA

El método FISTA (*Fast Ineractive Shrinkage-Thresholding Algorithm*) es una extensión del algoritmo gradencial clásico, donde en cada iteración nos acercamos a la solución en la dirección del gradiente de la función objetivo, la elección del punto a evaluar el gradiente en cada iteración es la diferencia entre los distintos métodos gradenciales. El algoritmo FISTA es muy rápido y resulta idóneo para resolver problemas de optimización con una carga elevada de datos.

El método FISTA no está incluido en Matlab por lo que deberá ser programado como función para ser usado. En este TFG no nos centraremos en su implementación, lo usaremos directamente ya que ha sido proporcionado por los tutores del trabajo Daniel Rodríguez Ramírez y Alfonso Daniel Carnerero como una función de Matlab.

La implementación en Matlab de este algoritmo será mostrada en apartados posteriores.

2.5 Ejemplo práctico

Para darle más sentido al cálculo de la disimilitud vamos a implementar un ejemplo sencillo de una elipse un 2D en Matlab, es decir, vamos a generar un set de datos con los puntos que formarían una elipse. Luego, vamos a calcular la disimilitud por los métodos vistos y compararemos los resultados obtenidos para diferentes puntos. De este modo, obtendremos una idea de cómo varía la solución obtenida en función del punto elegido para el cálculo del problema y podremos comparar la eficiencia de los 3 métodos presentados, así como el tiempo empleado en realizar el cálculo con una base de datos considerable.

Para implementar la elipse en Matlab necesitamos la ecuación típica de la elipse y un vector x con diferentes valores, de este modo hallaremos las coordenadas y para cada punto x dado. La ecuación de la elipse a implementar para nuestro ejemplo será:

$$\frac{x^2}{25} + \frac{y^2}{16} = 1$$

Tomaremos como dato un vector x que vaya del valor 5 al -5 con un paso de 0.02, es decir nuestro set de datos estará compuesto de 1002 componentes. Con esto ya podemos obtener los valores de y de la elipse para completar el set de datos, además de poder representar gráficamente todo el set de dato que conforma la elipse:

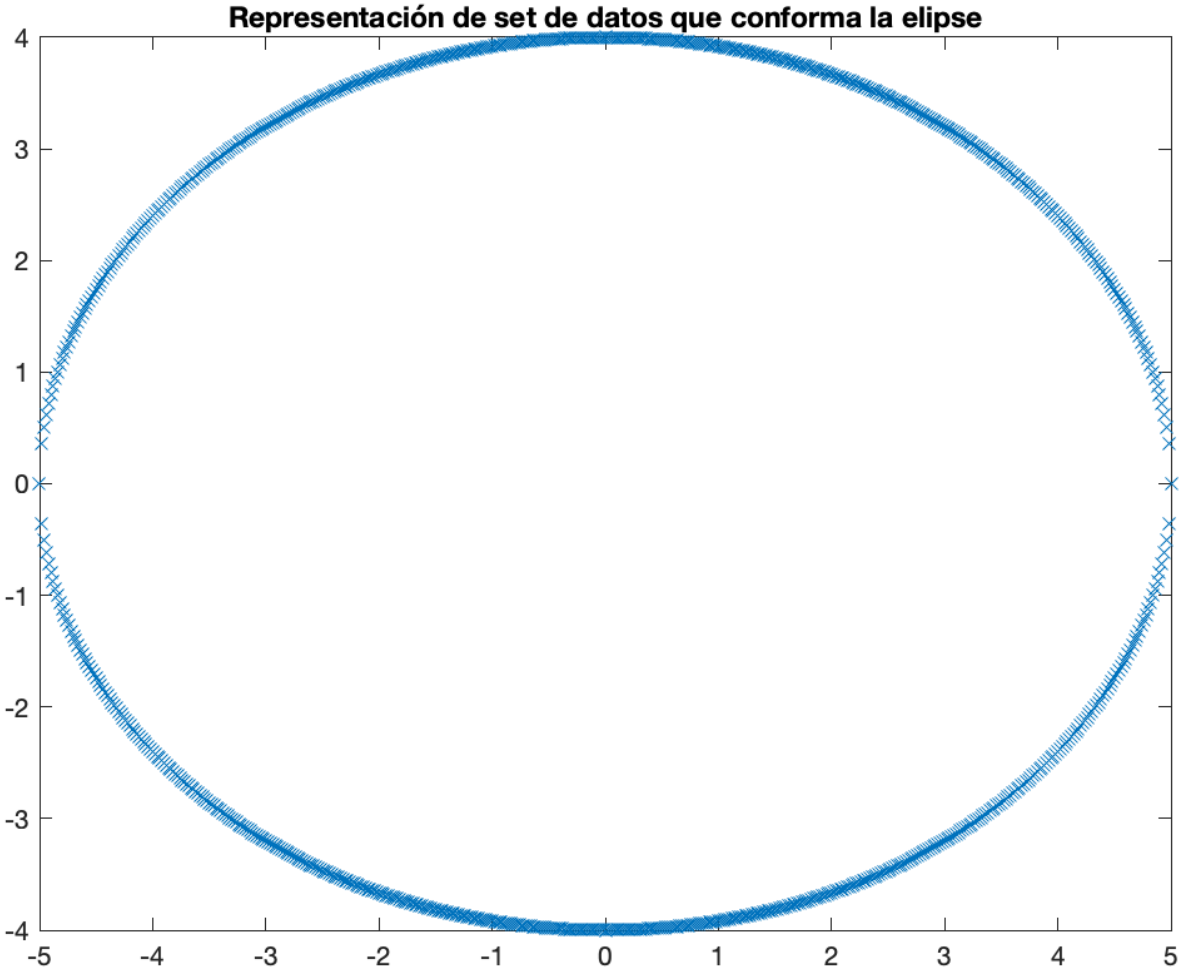

Ilustración 2-1. Elipse

La primera tarea por realizar en este ejemplo será comparar los resultados obtenidos por los diferentes métodos explicados en este capítulo para diferentes puntos elegidos, es decir, hallaremos la disimilitud de diferentes

puntos respecto al set de datos de la elipse. Tomaremos puntos que estén dentro de la elipse, fuera de ella, contenidos en ella y muy alejados de la misma. De este modo:

Punto\Método	**fmincon**	**quadprog**	**FISTA**
(0,0)	0.5010	0.5010	0.5005
(1,2)	0.5015	0.5015	0.5007
(4,3)	0.7886	0.5662	0.5595
(10,20)	3.1825	2.8370	2.7824
(0,-4)	0.6181	0.5126	0.5079
(-4,-5)	0.9417	0.7638	0.7557

Tabla 2-1. Valores disimilitud para diferentes puntos

En Tabla 2-1 se observa que cuanto más se aleja el punto elegido de la elipse, mayores valores de disimilitud se obtienen en cualquiera de los tres métodos, véase el punto (10,20) en comparación con el resto, esto es totalmente coherente con lo explicado durante todo este apartado. El centro de la elipse es el punto que mejores aproximaciones obtiene para todos los métodos, cuanto más nos alejamos de dicho punto los resultados obtenidos para *fmincon* difieren mucho de *quadprog* y *FISTA*, esto se debe a limitaciones internas de la función *fmincon* relacionadas con tolerancias de cálculo.

Otro análisis para realizar es la comparativa de tiempo de ejecución para cada método, esto resultará de vital importancia a la hora de implementar el algoritmo de predicción de intervalos de este TFG ya que necesitaremos ejecutar este problema de optimización repetidas veces. Para realizar este experimento vamos a usar la herramienta *tic…toc* de Matlab que devuelve esta variable, lo realizaremos para los mismos puntos que en el análisis anterior:

Punto\Método	**fmincon**	**quadprog**	**FISTA**
(0,0)	0.1796s	13.0063s	0.0184s
(1,2)	0.5168s	16.4494s	0.0166s
(4,3)	0.5920s	19.7728s	0.0400s
(10,20)	0.7222s	21.1738s	0.0226s
(0,-4)	0.6941s	18.3886s	0.0602s
(-4,-5)	0.5636s	18.4900s	0.0319s

Tabla 2-2. Valores de tiempo de ejecución en segundos

En la Tabla 2-2 se observa que hay una diferencia considerable entre los diferentes métodos de cálculo del problema de optimización a implementar. La función *quadprog* es la que emplea un tiempo más elevado debido a que para resolver el problema del valor absoluto emplea el doble de datos que los otros 2 para realizar los mismos cálculos, de este modo en *quadprog* para este ejemplo emplea 2004 datos en lugar de 1002 de los otros dos métodos. Luego la función *fmincon* reduce mucho el tiempo respecto a *quadprog* pero sin dejar de ser elevado para una aplicación donde se tendrá que repetir la resolución del problema multitud de veces. La que obtiene mejores resultados es *FISTA* siendo 10 veces más rápido que *fmincon* y 1000 veces más rápido que *quadprog*.

Viendo estos 2 experimentos se puede concluir que el método más efectivo en Matlab es el FISTA ya que emplea un tiempo de ejecución bajo a la vez que proporciona resultados fiables usando un número elevado de datos. De este modo concluimos que para la resolución de nuestro algoritmo el método más eficiente para resolver el problema de optimización convexa planteado es el FISTA.

3 ALGORITMO DE PREDICCIÓN DE INTERVALOS

La ciencia nunca resuelve un problema sin crear otros 10

-George Bernard Shaw-

En este capítulo vamos a aplicar los conocimientos y conclusiones tomadas en el capítulo anterior para elaborar un algoritmo que nos proporcione predicciones de intervalos de la salida y_k con cierta probabilidad, es decir, establecer una probabilidad para la que la salida y_k estará contenida en el intervalo calculado.

Usaremos el cálculo de las funciones de disimilitud mediante el método FISTA para hallar funciones de densidad de probabilidad empíricas condicionadas de cada salida y_k a calcular. Luego se podrá calcular la función de distribución empírica condicionada con la que se podrá estimar que la salida este contenida en el intervalo hallado, normalmente se usará una probabilidad del 90 o 95%

3.1 Definición De Función De Densidad De Probabilidad Empírica

La disimilitud de un vector $z \in \mathcal{Z} \subseteq \mathbb{R}^n$ con respecto a los elementos de un set de datos D puede ser usada para definir la función de densidad de probabilidad empírica. Para ello usaremos la definición de función de disimilitud (2-2)

La función de densidad de probabilidad empírica se define como la división de la similitud de un vector z entre la similitud de todo el dominio $\mathcal{Z}$. Más específicamente:

Dado un conjunto de medidas $D = \{z_1, \dots, z_N\} \subset \mathcal{Z}, \gamma \geq 0\ y\ c \geq 0$, la función de densidad de probabilidad empírica $ep_{\gamma,c}(z, D)$ es definida para cada $z \in \mathcal{Z}$ como

$$ep_{\gamma,c}(z, D) = \frac{\exp\left(-cJ_\gamma(z, D)\right)}{\int_Z \exp\left(-cJ_\gamma(\hat{z}, D)\right) d\hat{z}} \tag{3-1}$$

Si observamos la ecuación (3-1) se concluye que devuelve una familia de funciones de densidad de probabilidad parametrizadas por las constantes γ y c, según ambos valores la distribución puede variar

notablemente. Además, por construcción se observa:

$$\int_{Z} \exp\left(-cJ_{\gamma}(\hat{z}, D)\right) d\hat{z} = 1$$

Se remarca que si el conjunto $\mathcal{Z}$ es compacto, una elección de $c = 0$ nos aporta una función de densidad de probabilidad empírica uniforme.

3.2 Definición De Función De Densidad De Probabilidad Empírica Condicionada

Ahora se plantea la definición de la función de densidad de probabilidad empírica condicionada ya que se conoce la entrada x_k y se quiere estimar la salida $\widehat{y_k}$. D este modo, se puede definir como:

Conocidas la entrada $x_k \in \mathcal{X}$, un conjunto de datos $D = \left\{\begin{bmatrix} y_i \\ x_i \end{bmatrix} : i = 1, \dots, N\right\} \subset \mathcal{Y}\, x\, \mathcal{X}$ y los escalares no negativos c y γ, se puede definir la función de densidad de probabilidad empírica condicionada como

$$ep_{\gamma,c}(y, x_k, D) = \frac{\exp\left(-cJ_{\gamma}\left(\begin{bmatrix} y \\ x_k \end{bmatrix}, D\right)\right)}{\int_{Y} \exp\left(-cJ_{\gamma}\left(\begin{bmatrix} \hat{y} \\ x_k \end{bmatrix}, D\right)\right) d\hat{y}}, \forall y \in \mathcal{Y} \tag{3-2}$$

La definición (3-2) permite calcular la probabilidad de obtener la salida y dada la entrada x_k , condicionada por los parámetros c y γ. Conocidas estas definiciones podemos empezar a plantear como calcular el intervalo de estimación de y_k, a partir de la entrada x_k.

3.2.1 Simplificación para el conjunto de salidas Y

Teniendo en mente la ejecución de un algoritmo de predicción de intervalos con una gran cantidad de datos, es conveniente simplificar los cálculos.

Para poder calcular adecuadamente la integral de (3-2) es conveniente simplificar su integración numérica mediante un sumatorio. Idealmente los límites de dicha integral son $[-\infty, +\infty]$ para contener todas las salidas posibles, esto hace que computacionalmente el cálculo sea muy costoso. Debido a ello se va a aproximar el conjunto de datos de salida $\mathcal{Y}$ por el nuevo conjunto de datos $\bar{\mathcal{Y}}$, donde $\bar{\mathcal{Y}} = \{\overline{y_1}, \dots, \overline{y_M}\}$ y cuyos valores máximos y mínimos coinciden con los valores máximos y mínimos del conjunto de datos D. Formalmente consideramos:

$$\bar{\mathcal{Y}} = \{\overline{y_1}, \dots, \overline{y_M}\} \; donde \; \overline{y_j} < \overline{y_{j+1}}, j = 1, \dots, M-1$$

Donde los valores extremos de $\bar{\mathcal{Y}}$ son elegidos para garantizar que y_k esté contenido en $[\bar{y}_1, \bar{y}_M]$ con una alta probabilidad. Un posible procedimiento para construir el conjunto de datos $\bar{\mathcal{Y}}$ podría ser:

$$\overline{y_1} = \min_{i=1,\dots,N} y_i$$
$$\overline{y_M} = \max_{i=1,\dots,N} y_i$$

$$\overline{y_j} = \overline{y_1} + \left(\frac{\overline{y_M} - \overline{y_1}}{M-1}\right)(j-1), j = 2, \dots, M-1 \tag{3-3}$$

Se observa fácilmente que a mayores valores de M se obtendrán aproximaciones más precisas a cambio de incrementar el coste computacional.

3.2.2 Función de distribución empírica condicionada discreta

Una vez ha sido explicada la necesidad de acotar el conjunto de salidas $\overline{\mathcal{Y}}$, es posible aplicarlo al concepto de función de densidad de probabilidad empírica condicionada. De este modo se define la función de distribución empírica condicionada discreta como:

$$\overline{ecp}_{\gamma,c}(y, x_k, D) = \frac{\exp\left(-cJ_\gamma\left(\begin{bmatrix} y \\ x_k \end{bmatrix}, D\right)\right)}{\sum_{j=1}^{M} \exp\left(-cJ_\gamma\left(\begin{bmatrix} \overline{y_j} \\ x_k \end{bmatrix}, D\right)\right)} \tag{3-4}$$

Donde por construcción:

$$\sum_{j=1}^{M} \overline{ecp}_{\gamma,c}(\overline{y_l}, x_k, D) = 1 \tag{3-5}$$

La función de densidad de probabilidad empírica condicionada discreta define una función de distribución condicionada en $\overline{\mathcal{Y}}$, que denotaremos como $Prob_{\overline{\mathcal{Y}}|x_k}$. Con respecto a esta distribución, se denota $\overline{y_l}$ como el elemento l-ésimo de $\overline{\mathcal{Y}}$, entonces la probabilidad de que la salida y_k condicionada a x_k sea menor o igual que $\overline{y_l}$ queda expresada como:

$$Prob_{\overline{\mathcal{Y}}|x_k}\{y \le \overline{y_l}\} = \sum_{j=1}^{l} \overline{ecp}_{\gamma,c}\left(\overline{y_j}, x_k, D\right) \tag{3-6}$$

Del mismo modo, la probabilidad de que la salida y_k condicionada a x_k sea mayor o igual que $\overline{y_l}$ queda expresada como:

$$Prob_{\bar{y}|x_k}\{y \geq \overline{y_l}\} = \sum_{j=1}^{l} \overline{ecp}_{\gamma,c}(\overline{y_J}, x_k, D) \tag{3-7}$$

3.2.3 Cuantiles empíricos condicionados

Tomando como referencia las expresiones [ec de arriba] y [ec de arriba], se define:

Dada una entrada x_k, los parámetros $\gamma \geq 0$ y $c \geq 0$ y $\tau \in (0,1)$. Se define y_τ^+ como el τ-cuantil condicionado superior refiriéndose al menor valor del conjunto de salidas $\bar{y}$ que satisface:

$$Prob_{\bar{y}|x_k}\{y \leq y_\tau^+\} \geq 1 - \tau \tag{3-8}$$

De manera análoga, se define y_τ^- como el τ-cuantil condicionado inferior refiriéndose al mayor valor del conjunto de salidas $\bar{y}$ que satisface:

$$Prob_{\bar{y}|x_k}\{y \geq y_\tau^-\} \geq 1 - \tau \tag{3-9}$$

Concluimos que dados una entrada x_k y $\tau \in (0,1)$, el intervalo de predicción obtenido para y_k será $[y_\tau^- , y_\tau^+]$, pudiendo asumir que la probabilidad de que la salida y_k condicionada x_k esté contenida en el intervalo $[y_\tau^- , y_\tau^+]$ es:

$$Prob_{\bar{y}|x_k}\{y \in [y_\tau^- , y_\tau^+]\} \geq 1 - 2\tau \tag{3-10}$$

El tamaño de los intervalos dependerá en cualquier caso del valor de τ para el cálculo de los cuantiles superior e inferior. A medida que se disminuye el valor de τ es más probable que la salida esté contenida en el intervalo calculado, sin embargo, cuando aumentamos el valor de τ es más probable que la salida este fuera de dicho intervalo forzando obtener un intervalo con un tamaño más preciso.

3.3 Algoritmo 1. Predicción de intervalos.

Una vez se ha explicado la necesidad de discretizar el conjunto de salidas como $\bar{y}$ y la manera de obtener una salida y_k que este contenida en un intervalo $[y_\tau^- , y_\tau^+]$ cierta probabilidad preestablecida, se puede explicar el algoritmo planteado para hallar las predicciones de intervalos que contengan a la salida y_k.

Este apartado se dividirá en la explicación del algoritmo y en cómo será adaptado en Matlab.

3.3.1 Explicación del Algoritmo 1

Para la ejecución de este algoritmo partiremos de una entrada x_k proveniente de un conjunto de entradas $\mathcal{X}$, un valor para el parámetro del cuantil τ contenido entre 0 y 1, los valores de los parámetros γ y c que serán optimizados en apartados posteriores, un set de datos D y conjunto de salidas discretizadas $\bar{\mathcal{Y}} = \{\overline{y_1}, \dots, \overline{y_M}\}$ computadas según la aplicación (se recuerda un ejemplo en (3-3) de una posible aplicación), con todo esto se puede proceder a la explicación del algoritmo poco a poco teniendo en mente que queremos obtener el intervalo $[y_\tau^-, y_\tau^+]$.

Primero se va a calcular la hallar la disimilitud de cada conjunto de datos $\begin{bmatrix}\bar{y}_j \\ x\end{bmatrix}$ con respecto al set de datos D, donde $j = 1, \dots, M$, y se guardará en un vector de tamaño M denotado como d_j, más específicamente se calculará el problema de optimización convexa presentado en (2-2):

$$d_j = J_\gamma\left(\begin{bmatrix}\bar{y}_j \\ x\end{bmatrix}, D\right)$$

Luego se computa la similitud de cada conjunto $\begin{bmatrix}\bar{y}_j \\ x\end{bmatrix}$ con respecto al set de datos D, gracias a la disimilitud hallada anteriormente:

$$\exp(-cd_j) \tag{3-11}$$

Una vez conocemos como hallar la similitud del conjunto de datos respecto al set de datos es posible hallar las probabilidades condicionadas siguiendo la ecuación (3-4), almacenando los resultados para cada par de datos $\begin{bmatrix}\bar{y}_j \\ x\end{bmatrix}$ en otro vector que se denominará como p_j, de tamaño M:

$$p_j = \overline{ecp}_{\gamma,c}(\bar{y}_j, x, D) = \frac{\exp(-cd_j)}{\sum_{l=1}^{M}\exp(-cd_l)}$$

Gracias al vector de probabilidades p_j se podrán realizar dos sumatorios, uno desde 1 hasta M y otro desde M hasta 1, para hallar de este modo los índices l_τ^+ y l_τ^- correspondientes a los τ-cuantiles superior e inferior condicionados. Ambos sumatorios se detendrán cuando la probabilidad acumulada tenga un valor mayor o igual a $1-\tau$:

$$l_\tau^+ = el\ menor\ índice\ l\ que\ satisface \sum_{j=1}^{l} p_j \geq 1-\tau$$

$$l_\tau^- = el\ mayor\ índice\ l\ que\ satisface \sum_{j=l}^{M} p_j \geq 1-\tau$$

Una vez se han obtenido ambos índices solo queda evaluarlos en el conjunto de salidas $\bar{y}$ para obtener el intervalo deseado:

$$y_\tau^- = \bar{y}_{l_\tau^-}$$

$$y_\tau^+ = \bar{y}_{l_\tau^+}$$

Resumiendo, dada una entrada x_k este algoritmo calcula un intervalo $[y_\tau^- , y_\tau^+]$ en el que se encontrará la salida y_k con una probabilidad de $1 - 2\tau$. Los resultados obtenidos dependerán de los valores de los parámetros γ y c, cuyos valores óptimos serán necesarios para obtener mejores predicciones de intervalos.

A continuación, se mostrará una ilustración del significado del algoritmo 1, el ejemplo proviene de un extracto de ejecución de código realizado con una base de datos proporcionada por los tutores de este TFG para realizar pruebas llamado *Dataset de Lorenz* (se profundizará en dicho ejemplo más adelante)

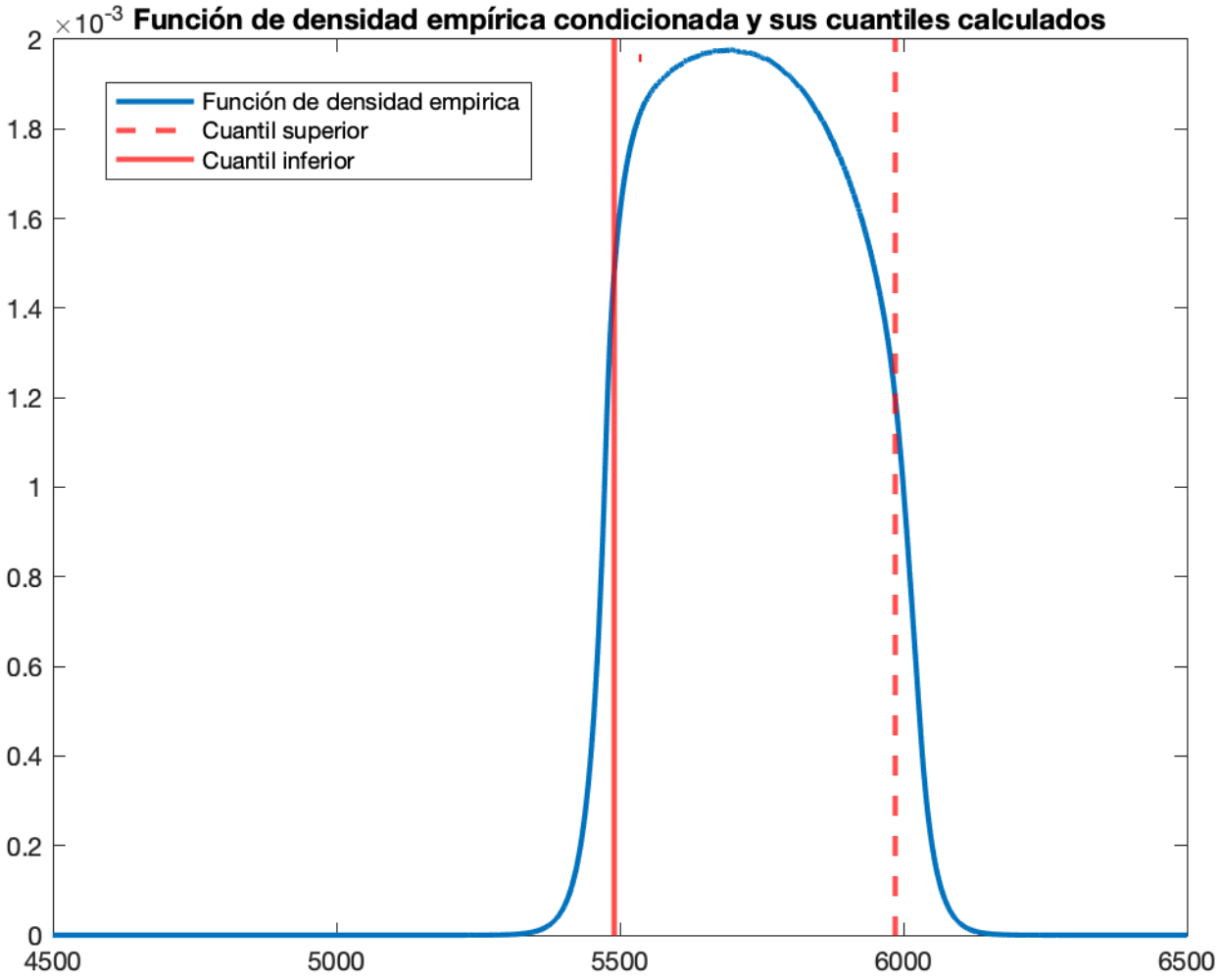

Ilustración 3-1. Cuantiles hallados en el algoritmo 1

En la Ilustración 3-1 se observa un ejemplo de función de densidad empírica condicionada calculada en el algoritmo 1 gracias al conjunto $\bar{y}$ y a una entrada x_k, la forma de dicha función se debe a los parámetros γ y c. En la ilustración se observa los valores l_τ^+ y l_τ^- hallados en el algoritmo 1, dichos valores nos proporcionarán el intervalo de predicción al evaluarlos en el conjunto $\bar{y}$. La idea es que la salida y_k este contenida entre dicho intervalo calculado con cierta probabilidad como se explicó anteriormente.

3.3.2 Implementación en Matlab del Algoritmo 1

En este apartado se explicará cómo se ha implementado el algoritmo 1 en Matlab, así como algunos detalles y trucos para mejorar la eficiencia del código.

Primero se expondrá paso a paso como se ha implementado el algoritmo 1 en Matlab:

Algoritmo 1

Se requiere: una entrada $x_k,\ c \geq 0,\ \gamma \geq 0,\ \tau \in [0,1],\ D$ y $\bar{\mathcal{Y}} = \{\overline{y_1}, \dots, \overline{y_M}\}$

Se obtiene: y_τ^-, y_τ^+

1) Se normaliza los datos a emplear, se inicializan variables: Aeq (para restricciones de tipo Ax=b) y vector T necesario para ejecutar FISTA Se calcula la disimilitud para todo el conjunto de salidas discretizado $\bar{\mathcal{Y}}$ usando el algoritmo FISTA. Para hallar la disimilitud en todo el conjunto $\bar{\mathcal{Y}}$ se realizará con un bucle for de 1 a M que vaya cambiando la restricción de igualdad *beq*.

$$FOR\ j = 1\ hasta\ M$$
$$beq = \left[\bar{y}_j, x, 1\right]$$
$$[J] = FISTA(T, \gamma, Aeq, beq, x_{ini})$$
$$d_j = J$$
$$END\ FOR$$

Para acelerar el cálculo en Matlab se le impondrá un punto inicial de búsqueda x_{ini} al algoritmo FISTA de valor 0. El valor de la disimilitud obtenido se almacena en un vector d

2) Cálculo de las probabilidades condicionadas según ecuación (3-4), almacenando resultados en un vector p. Se emplea la función *sum* de Matlab para resolver el sumatorio fácilmente.

$$p_j = \frac{\exp\left(-cd_j\right)}{\sum_{l=1}^{M} \exp\left(-cd_l\right)}$$

3) Cálculo de índices l_τ^- y l_τ^+ correspondientes a los cuantiles condicionados superior e inferior. Se empleará la función *cumsum* de Matlab para hacer sumatorios acumulados y la función *find* que nos permitirá encontrar los índices buscados dentro de la suma acumulada de probabilidades.
4) Se computan los índices l_τ^- y l_τ^+ en el conjunto de salidas $\bar{\mathcal{Y}}$ para obtener y_τ^-, y_τ^+

Se ha usado el método FISTA ya que es el más eficiente de los 3 que se han explicado para aplicar a Matlab, además si se pretende acelerar el cálculo de dicho algoritmo se puede usar la función *parfor* de Matlab que es capaz de ejecutar bucles for en paralelo haciendo uso de varios procesadores del ordenador para así agilizar el cálculo.

3.4 Algoritmo 2. Optimización del parámetro c.

Una vez ya se conoce como obtener el intervalo de predicción de la salida y_k, ahora se prestará atención a la

optimización de dicho intervalo. La forma de obtener un intervalo de predicción óptimo es con unos valores de los parámetros c y γ óptimos a los que llamaremos c_τ^* y γ_τ^*. En este apartado nos centraremos en obtener el valor c_τ^* mediante un algoritmo, dado un cierto valor de γ.

El parámetro c resulta de vital importancia a la hora de obtener intervalos de un tamaño adecuado, a medida que el valor de c aumenta el tamaño del intervalo obtenido se hace más pequeño, pero esto afecta negativamente a la probabilidad de que la salida y_k esté contenida en dicho intervalo. Del mismo modo, disminuir el valor de c hace que el tamaño del intervalo aumente y por lo tanto será más probable que la salida y_k esté contenida en dicho intervalo. Entonces, debemos encontrar un valor de c que no nos devuelva el menor tamaño posible del intervalo y que cumpla con la probabilidad especificada previamente para que y_k esté contenida en el intervalo hallado.

3.4.1 Explicación del Algoritmo 2

El algoritmo 2 pretende obtener el valor máximo de c que garantice que las salidas están contenidas en el intervalo calculado mediante el algoritmo 1 con la probabilidad preestablecida mediante τ.

Previamente se analiza el papel que toma c en el cálculo del algoritmo 1, se emplea en la ecuación (3-4) donde si se toma un valor de $c = 0$ se obtiene una distribución plana donde cada elemento de $\bar{\mathcal{Y}}$ está condicionado con una probabilidad de $\frac{1}{M}$. Unos valores grandes de c se traducen en distribuciones muy estrechas en torno al punto del conjunto $\bar{\mathcal{Y}}$ que la minimiza provocando que sea más probable que un mayor número de salidas y_k se queden fuera del intervalo.

El algoritmo 2 que se va a explicar va a utilizar un conjunto de datos de validación denotado como

$$\mathcal{V} = \left\{ \begin{bmatrix} \widetilde{y}_s \\ \widetilde{x}_s \end{bmatrix} : s = 1, \dots, N_{\mathcal{V}} \right\} \subset \mathcal{Y} \times \mathcal{X} ,$$

un valor fijo del parámetro $\gamma \geq 0,\ \tau \geq 0, c \geq 0$. Para la realización de este se va a recurrir al ya explicado algoritmo 1 usando sus predicciones de intervalos, para ejecutarlo se necesitará el conjunto de salidas $\overline{\mathcal{Y}}$ y el set de datos D. Además, como se pretende hallar un valor de c óptimo, se partirá de unos valores máximo y mínimo iniciales que se irán modificando a lo largo del desarrollo del algoritmo, el valor mínimo de c siempre se tomará como 0 y el valor máximo dependerá del ejemplo al que se quiera aplicar el algoritmo. En función de una tolerancia preestablecida entre los valores máximo y mínimo de c se podrá decir que se ha llegado al valor óptimo de c.

El algoritmo 2 recalcula en cada iteración el valor de c que va a emplear, luego computa los $N_{\mathcal{V}}$ intervalos de predicción gracias al algoritmo 1 ya implementado. Con los intervalos de predicción calculados va revisando si el valor de la salida está contenido en dicho intervalo, de ese modo calcula el número de violaciones que se comete para el valor de c proporcionado. Si el cociente de ese número de violaciones entre el valor del tamaño del conjunto de validación $N_{\mathcal{V}}$ es superior al valor de τ, se tendrá que reducir el valor de c puesto que el número de violaciones cometido es superior al requerido. En caso contrario, se aumentará el valor de c hasta que la diferencia entre la c máxima y la mínima sea inferior a una tolerancia denotada como ε.

3.4.2 Implementación en Matlab del Algoritmo 2

Se expone el paso a paso para poder implementar el algoritmo 2 en Matlab.

Algoritmo 2

Se requiere: $c_{max} \geq 0,\ \gamma \geq 0,\ \tau \geq 0,\ \varepsilon \geq 0,\ D$, $\bar{\mathcal{Y}} = \{\overline{y_1}, \dots, \overline{y_M}\}$ y el conjunto de validación $\mathcal{V} = \left\{\begin{bmatrix}\widetilde{y_s}\\ \widetilde{x_s}\end{bmatrix} : s = 1, \dots, N_{\mathcal{V}}\right\}$

Se obtiene: c_γ

1) Se inicializa el valor del parámetro $c_{min} = 0$
2) Se inicia el bucle **while** de búsqueda del valor óptimo de c

$$while\ c_{max} - c_{min} > \varepsilon$$

3) Se recalcula el valor de c a utilizar en cada iteración

$$c = \frac{1}{2}(c_{max} + c_{min})$$

4) Se ejecuta el algoritmo 1 para obtener los $N_{\mathcal{V}}$ intervalos de predicción, luego para cada intervalo hallado se compara si su salida está dentro o fuera del mismo, aumentando el número de violaciones en caso afirmativo.

$$[y_\tau^-, y_\tau^+] = Algoritmo1(\widetilde{x_s}, \tau, \gamma, c, D, \bar{\mathcal{Y}})$$

$$\boldsymbol{SI}(\ y_\tau^- > \widetilde{y_s}) \rightarrow n_{viol}^- = n_{viol}^- + 1$$

$$\boldsymbol{SI}(\ y_\tau^+ < \widetilde{y_s}) \rightarrow n_{viol}^+ = n_{viol}^+ + 1$$

5) Analizamos el número de violaciones cometido para el valor de c propuesto, recalculando valores de c_{min} y c_{max}.

$$\boldsymbol{SI}\left(\frac{\max(n_{viol}^-, n_{viol}^+)}{N_{\mathcal{V}}}\right) < \tau \rightarrow\ c_{min} = c$$

$$\boldsymbol{SI}\left(\frac{\max(n_{viol}^-, n_{viol}^+)}{N_{\mathcal{V}}}\right) > \tau \rightarrow\ c_{max} = c$$

6) Se finaliza bucle **while** propuesto.
7) Se obtiene valor óptimo de c

$$c_\gamma = c_{min}$$

Se puede dar el caso en que no se encuentre ninguna c_{min}, en ese caso no se podría obtener el valor óptimo del parámetro c y habría que modificar algunos parámetros iniciales.

Se concluye que el algoritmo 2 proporciona el valor de c que garantiza que en el conjunto de validación $\mathcal{V}$ los intervalos calculados contienen a las salidas con la probabilidad definida, mayor que $1 - 2\tau$ y que además tienen el menor tamaño posible.

Para el cálculo del algoritmo 1 se ha vuelto a usar el método FISTA ya que como se estudió en apartados anteriores es el más eficiente para Matlab de los 3 planteados en este proyecto.

Una forma de agilizar el algoritmo 2 es ir evaluando los intervalos hallados y el número de violaciones a la vez que son calculados, de modo que pueda darse el caso que antes de evaluar los $N_\mathcal{V}$ intervalos de predicción ya se cumpla la condición para que se evalúe otra c diferente ahorrando ejecuciones del algoritmo 1. Esto ocurrirá cuando:

$$\left(\frac{\max\left(n_{viol}^{-}, n_{viol}^{+}\right)}{N_\mathcal{V}} \right) > \tau \rightarrow \; c_{max} = c \tag{3-12}$$

Si se supera el número de violaciones antes de llegar a evaluar los $N_\mathcal{V}$ intervalos se podrá tomar un nuevo valor para c_{max} y reevaluar todo el algoritmo de nuevo.

3.5 Optimización del parámetro γ

A partir del análisis del algoritmo 2 se concluye que el valor óptimo de c corresponde a un predefinido parámetro $\gamma \geq 0$. En este apartado se va a explicar una forma de obtener el valor óptimo del parámetro γ, esta vez fijando el valor de c_γ haciendo uso del algoritmo 2.

El valor del parámetro γ puede ser optimizado maximizando el valor del ratio probabilístico siguiente, para un valor especificado de γ y su correspondiente c_γ:

$$L_\gamma = \sum_{s=1}^{N_\mathcal{V}} \log\left(ecp_{\gamma, c_\gamma}\left(\widetilde{y_J}, \widetilde{x_J}, D\right)\right) \tag{3-13}$$

Usando el conjunto de salidas parametrizado $\overline{\mathcal{Y}} = \{\overline{y_1}, \ldots, \overline{y_M}\}$, una posible aproximación numérica al valor óptimo de γ es definida como:

$$\gamma_\tau^* \approx \arg\max_{\gamma\in\Gamma} \sum_{s=1}^{N_\mathcal{V}} \log\left(\frac{\exp\left(-c_\gamma J_\gamma\left(\begin{bmatrix}\widetilde{y_s}\\ \widetilde{x_s}\end{bmatrix}, D\right)\right)}{\sum_{j=1}^{M}\exp\left(-c_\gamma J_\gamma\left(\begin{bmatrix}\overline{y_J}\\ \widetilde{x_s}\end{bmatrix}, D\right)\right)}\right) \tag{3-14}$$

Donde Γ es el conjunto de valores que puede tomar γ a la hora de resolver el problema de maximización.

Otro criterio para obtener el valor γ_τ^* es minimizando una función de coste Q_γ que penalice la longitud media de los intervalos obtenidos.

De este modo, se consigue obtener la pareja de parámetros $(\gamma_\tau^*, c_\gamma)$ que consigue minimizar la longitud del intervalo hallado en el algoritmo 1 cumpliendo con la especificación de probabilidad para que dichos intervalos contengan a las salidas.

3.5.1 Implementación en Matlab del Algoritmo de Optimización de γ

Se implementará en Matlab la ecuación (3-14) para obtener el valor óptimo del parámetro γ, se expone el paso a paso de su desarrollo.

Algoritmo para γ óptimo

Se requiere: Valor de c_γ, conjunto de salidas $\overline{\mathcal{Y}} = \{\overline{y_1}, \dots, \overline{y_M}\}$, conjunto de validación $\mathcal{V} = \left\{\begin{bmatrix}\widetilde{y_s}\\ \widetilde{x_s}\end{bmatrix} : s = 1, \dots, N_\mathcal{V}\right\}$, conjunto de valores Γ y D

Se obtiene: γ_τ^*

1) Inicio de bucle **for** que recorre todos los valores que toma γ del conjunto Γ
2) Con el valor de γ elegido se calcula la disimilitud entre $\begin{bmatrix}\widetilde{y_s}\\ \widetilde{x_s}\end{bmatrix}$ y D, y la disimilitud entre $\begin{bmatrix}\overline{y_J}\\ \widetilde{x_s}\end{bmatrix}$ y D. Se hará uso del algoritmo FISTA para resolver el problema de optimización convexa en ambos casos. Para dicho calculo se usarán 2 bucles **for**, uno para los $N_\mathcal{V}$ valores del conjunto de validación y otro para los M valores del conjunto $\overline{\mathcal{Y}}$. Ambos valores se guardan en vectores para su posterior uso.

   Serán bucle **for** anidados, de modo que para cada valor de disimilitud del conjunto $N_\mathcal{V}$ se calculan todos los valores de disimilitud para las salidas del conjunto $\overline{\mathcal{Y}}$ y entrada $\widetilde{x_s}$, realizando posteriormente su sumatorio y obteniendo $N_\mathcal{V}$ sumatorios.

$$FOR\ s = 1\ hasta\ N_\mathcal{V}$$

$$beq = [\widetilde{y_s}, \widetilde{x_s}, 1]$$

$$[J] = FISTA(T, \gamma, Aeq, beq, x_{ini})$$

$$d_s = J$$

$$FOR\ j = 1\ hasta\ M$$

$$beq = [\overline{y_J}, \widetilde{x_s}, 1]$$

$$[J] = FISTA(T, \gamma, Aeq, beq, x_{ini})$$

$$d1_j = J$$

$$END\ FOR$$

$$SUMATORIO(d1)$$

$$END\ FOR$$

3) Se realiza el sumatorio que se desea maximizar según ecuación (3-14) para el valor de γ elegido inicialmente y se compara para ver si es el valor máximo hallado hasta el momento. El γ que devuelva el valor máximo de este sumatorio será el valor óptimo deseado, γ_τ^*

$$\sum_{s=1}^{N_V} \log\left(\frac{\exp(-c_\gamma d_s)}{SUMATORIO(d1)}\right)$$

4) Fin de bucle **for** inicial, una vez se hallan ejecutado todos los valores de γ del conjunto Γ.

Este algoritmo es el que supone un mayor coste computacional de los 3 presentados en este TFG, de modo que resulta de vital importancia emplear las técnicas y simplificaciones comentadas en apartados posteriores para acelerar el proceso.

4 APLICACIÓN 1. LORENZ

La perfección se alcanza, no cuando no hay nada más que añadir, sino cuando no queda nada que quitar.

- Antoine de Saint-Exupery-

En este capítulo se van a aplicar todos los algoritmos presentados y descritos a una base de datos real. Ya se han explicado todos los algoritmos y sus detalles para implementarlos en Matlab y ahora se va a obtener la pareja de parámetros $(\gamma_\tau^*, c_\gamma)$ con la que obtener predicciones de intervalos óptimas haciendo uso del algoritmo 1 empleado para un gran número de entradas reales, de este modo se obtendrán resultados gráficos apreciables. Además, se pretende mostrar gráficamente el comportamiento del parámetro c y la gráfica que se puede obtener gracias al algoritmo de γ óptimo.

4.1 Introducción de la base de datos

Se utilizará el ejemplo del atractor de Lorenz, el cual es un sistema dinámico determinístico tridimensional no lineal derivado de las ecuaciones simplificadas de rollos de convección que se producen en las ecuaciones dinámicas de la atmósfera terrestre. Dicho sistema es exhibe un comportamiento caótico para ciertos valores de los parámetros a, b y c. Este sistema aparece en láseres, en generadores eléctricos y en determinadas ruedas de agua.

Las ecuaciones del sistema son las siguientes:

$$\frac{dx}{dt} = a(y - x)$$

$$\frac{dy}{dt} = x(b - z) - y \tag{4-1}$$

$$\frac{dz}{dt} = xy - cz$$

Para el ejemplo que se va a plantear los parámetros reales tienen los siguientes valores: $a = 10, b = 28\ y\ c = 8/3$. Para obtener el conjunto de datos a raíz del sistema presentado se ha tomado un tiempo de muestreo de 0.1s y unas condiciones iniciales: $a(0) = 1, b(0) = 1\ y\ c(0) = 1$.

Se pretende predecir un paso hacia delante el valor de x del sistema, denotado como y_k para aplicar los algoritmos, haciendo uso de los 2 valores previos de x, que será el regresor $x_k = [y_{k-1}, y_{k-2}]$ a emplear como

entrada.

Se normalizarán todos los datos a emplear en un rango de 0 a 1. Se realizarán pruebas para diferentes tamaños de D: 200 y 350 puntos. Se emplearán 1000 puntos para el conjunto $\mathcal{V}$ y otros 1000 puntos para el conjunto de test $\mathcal{S}$. El conjunto Γ será tomado como diferentes valores entre los límites [0,3], con un paso de 0.2. Y el conjunto de salidas $\bar{\mathcal{Y}}$ será implementado como el conjunto de valores que existen entre los límites [-0.1893, 1.2298] con un paso de $1.4191x10^{-4}$.

Una vez se ha explicado la naturaleza del ejemplo y los parámetros y consideraciones elegidas, se obtienen los diferentes conjuntos de datos necesarios y es posible aplicar los algoritmos explicados en el transcurso de este trabajo.

4.2 Predicción de intervalos

Se van a realizar diferentes experimentos con la base de datos presentada, con diferentes valores de τ y número de puntos dentro del conjunto D. Antes será necesario hallar la pareja $(\gamma_\tau^*, c_\gamma)$ para los diferentes casos.

Primeramente, se realizarán varias pruebas para un valor de $\tau = 0.05$ variando el número de datos en el conjunto D. De este modo, se deben obtener intervalos que contengan a las salidas con una probabilidad mayor que 0.9 según (3-10). El parámetro γ se obtiene maximizando el ratio probabilístico de la ecuación (3-13)

A continuación, se muestran los resultados obtenidos para un conjunto de datos D que contiene 200 puntos, hallando previamente la pareja de parámetros óptima para dicho caso.

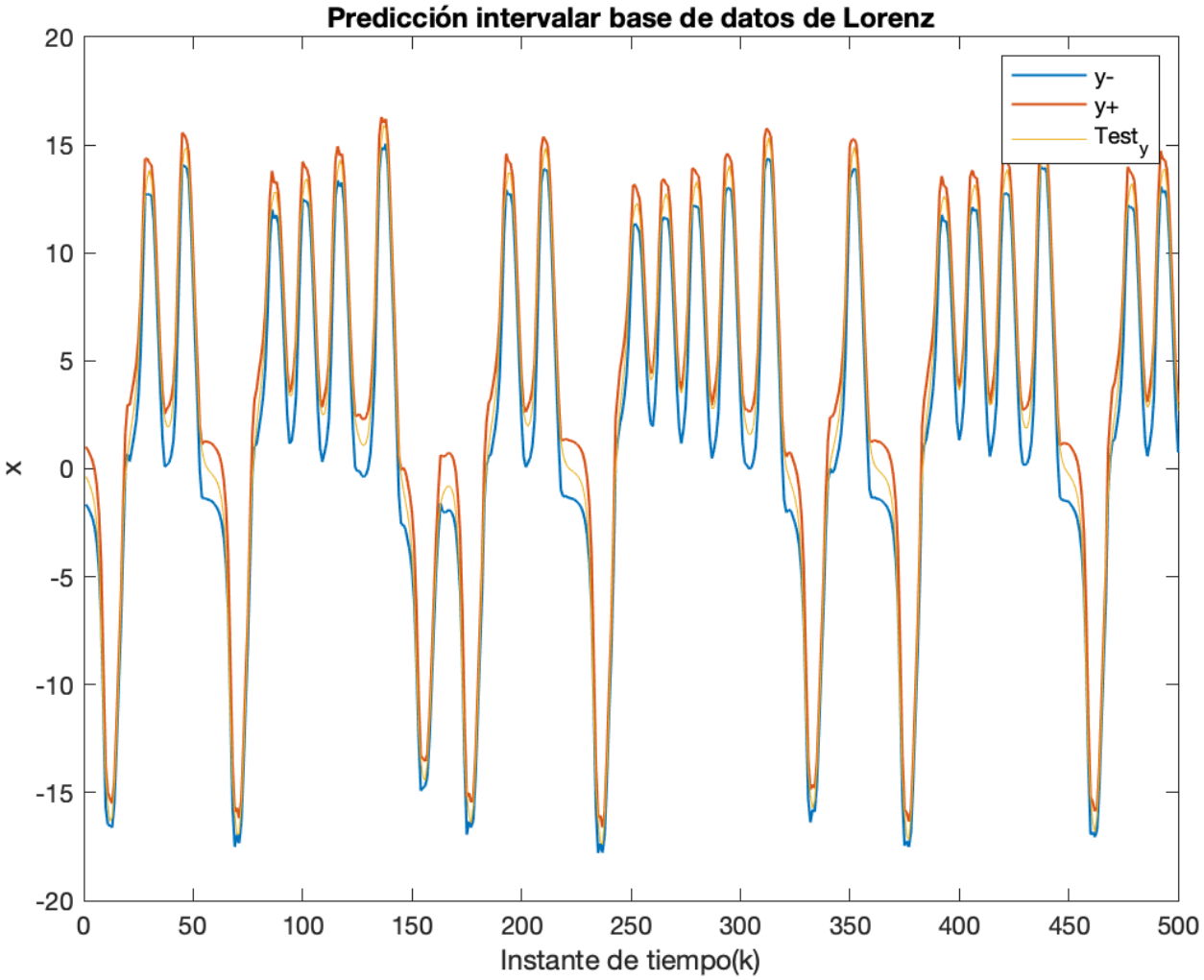

Ilustración 4-1. Predicción intervalar para γ=2.2, c=5.5, τ=0.05 y 200 en D

Para una interpretación más clara se puede exponer los resultados para un menor número de instantes de tiempo, de este modo se podrá apreciar como la salida suele estar contenida en el intervalo calculado gracias al algoritmo 1 y a una adecuada pareja de parámetros.

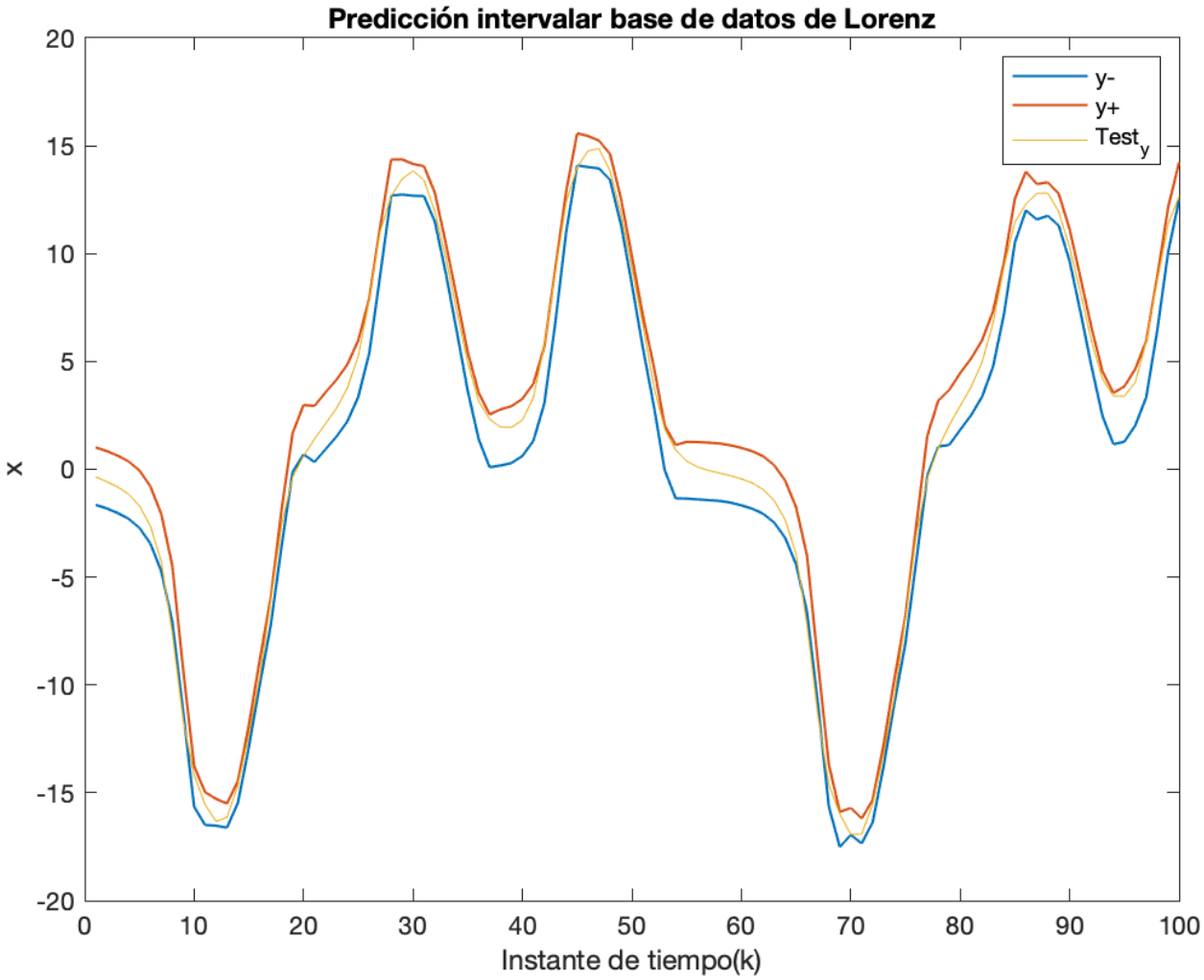

Ilustración 4-2. Predicción intervalar para γ=2.2, c=5.5, τ=0.05 y 200 en D. 100 puntos

También se puede mostrar cómo se ha elegido el valor óptimo de γ de 2.2 se ha hallado haciendo uso de la ecuación (3-14) y eligiendo el índice del conjunto Γ que proporciona el argumento máximo.

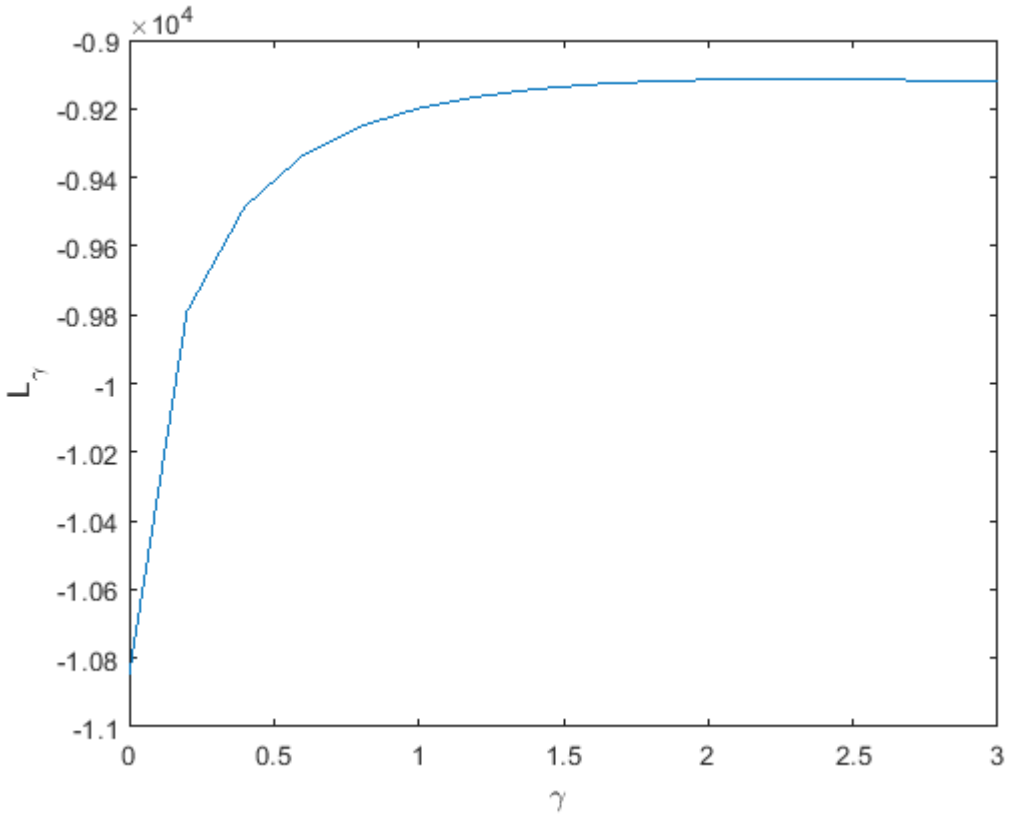

Ilustración 4-3. Maximización de ratio L_γ en función de γ. τ=0.05

Ahora se pretender aumentar el conjunto de datos D a 350 puntos y ver qué resultados se obtienen y como varía la pareja de parámetros obtenida.

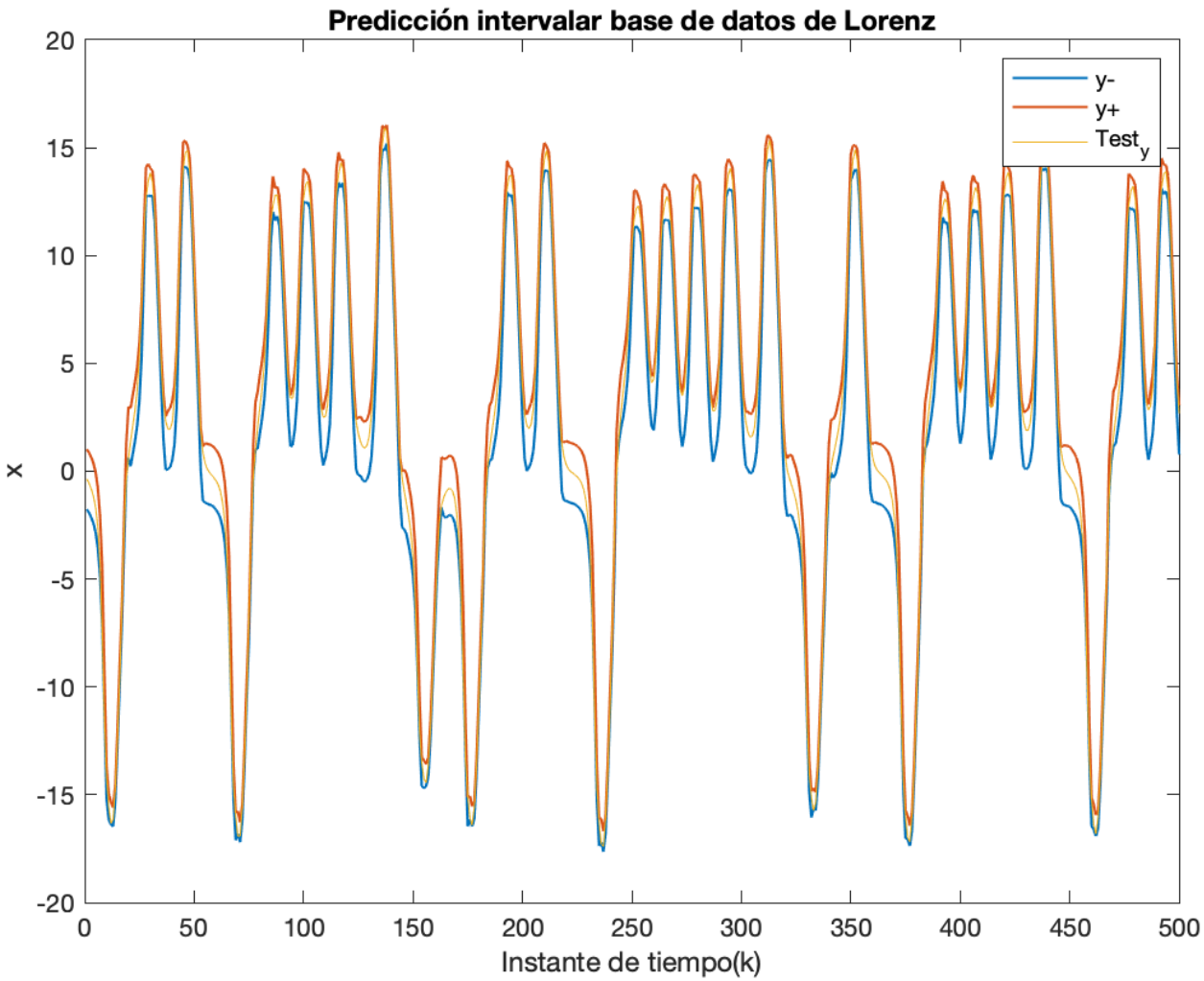

Ilustración 4-4. Predicción intervalar para γ=2, c=11, τ=0.05 y 350 en D

Se observa que, al aumentar el tamaño del set de datos, el parámetro c aumenta. Además, se puede realizar un análisis numérico de los resultados obtenidos calculando el tamaño medio de los intervalos calculados y la probabilidad empírica resultante para ambos tamaños de set de datos.

Tamaño set de datos	Tamaño medio	Probabilidad empírica
200 puntos	2.0105	0.9180
350 puntos	1.9540	0.9020

Tabla 4-1. Predicciones para $\tau = 0.05$

Se observa que el tamaño medio del intervalo no es demasiado grande en comparación con los datos de la base y que la probabilidad empírica hallada cumple con la especificación propuesta en ambos casos.

Ahora se van a repetir los cálculos para un valor de $\tau = 0.10$, de modo que se quieren obtener intervalos que contengan a las salidas con una probabilidad mayor que 0.8. Al disminuir la probabilidad especificada se podrán encontrar tamaños medios de intervalos menores. Se analiza primero para 200 puntos en el conjunto de datos D.

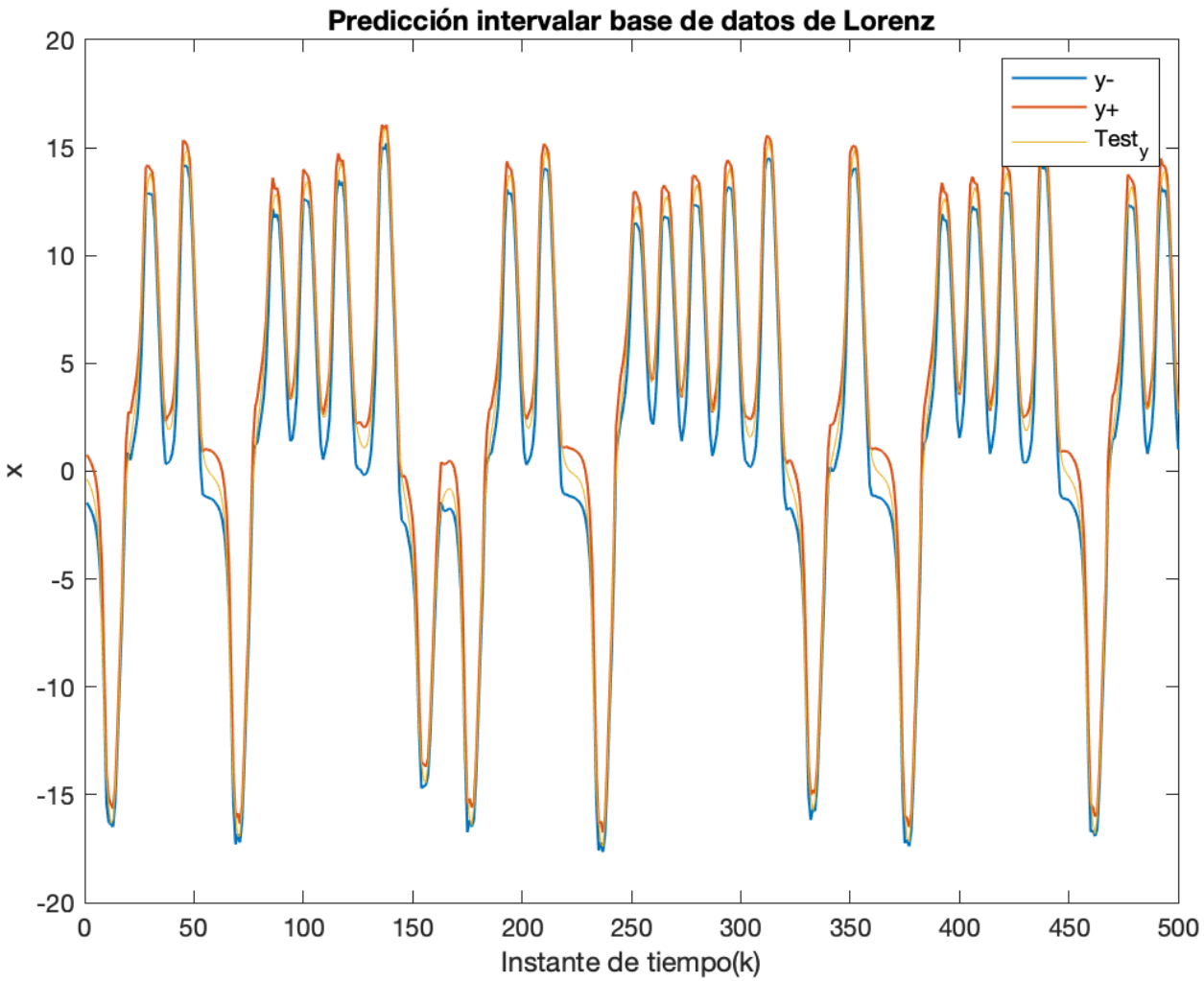

Ilustración 4-5. Predicción intervalar para γ=1.5, c=7.97, τ=0.1 y 200 en D

La gráfica obtenida para el cálculo del γ óptimo es análoga que para el caso de τ=0.05.

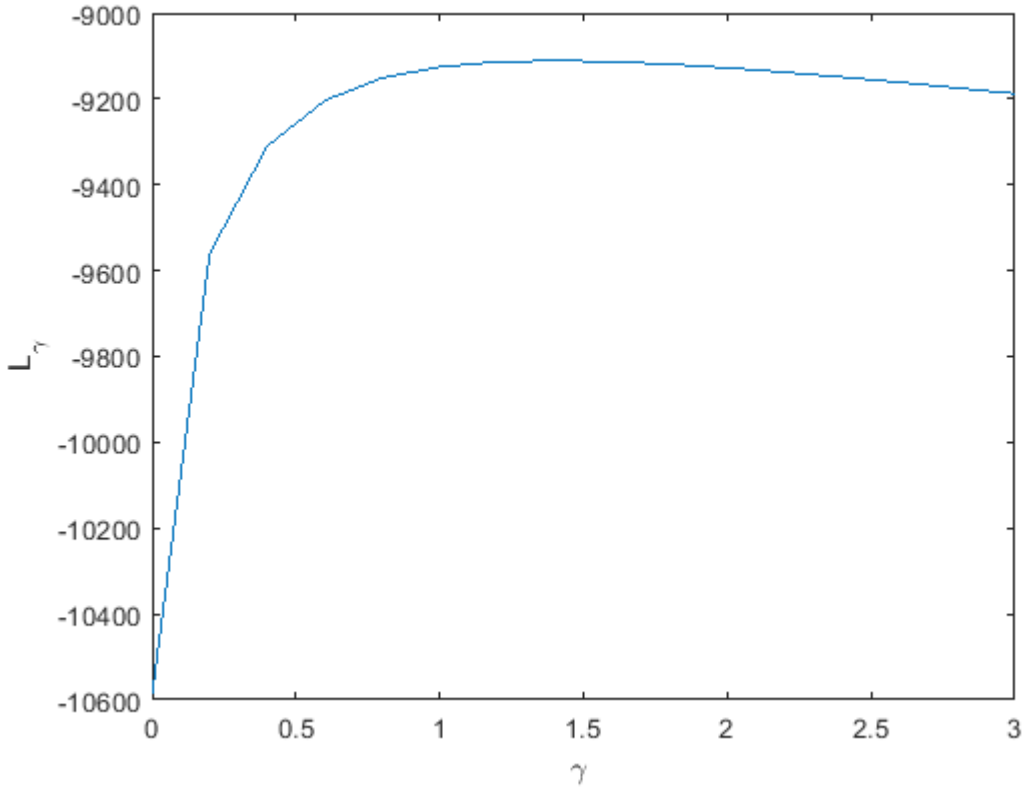

Ilustración 4-6. Maximización de ratio L_γ en función de γ. τ=0.1

Luego se prueba a aumentar el tamaño del conjunto de datos D a 350 puntos y ya se podría comparar el efecto de variar τ y el tamaño de D con los resultados anteriores.

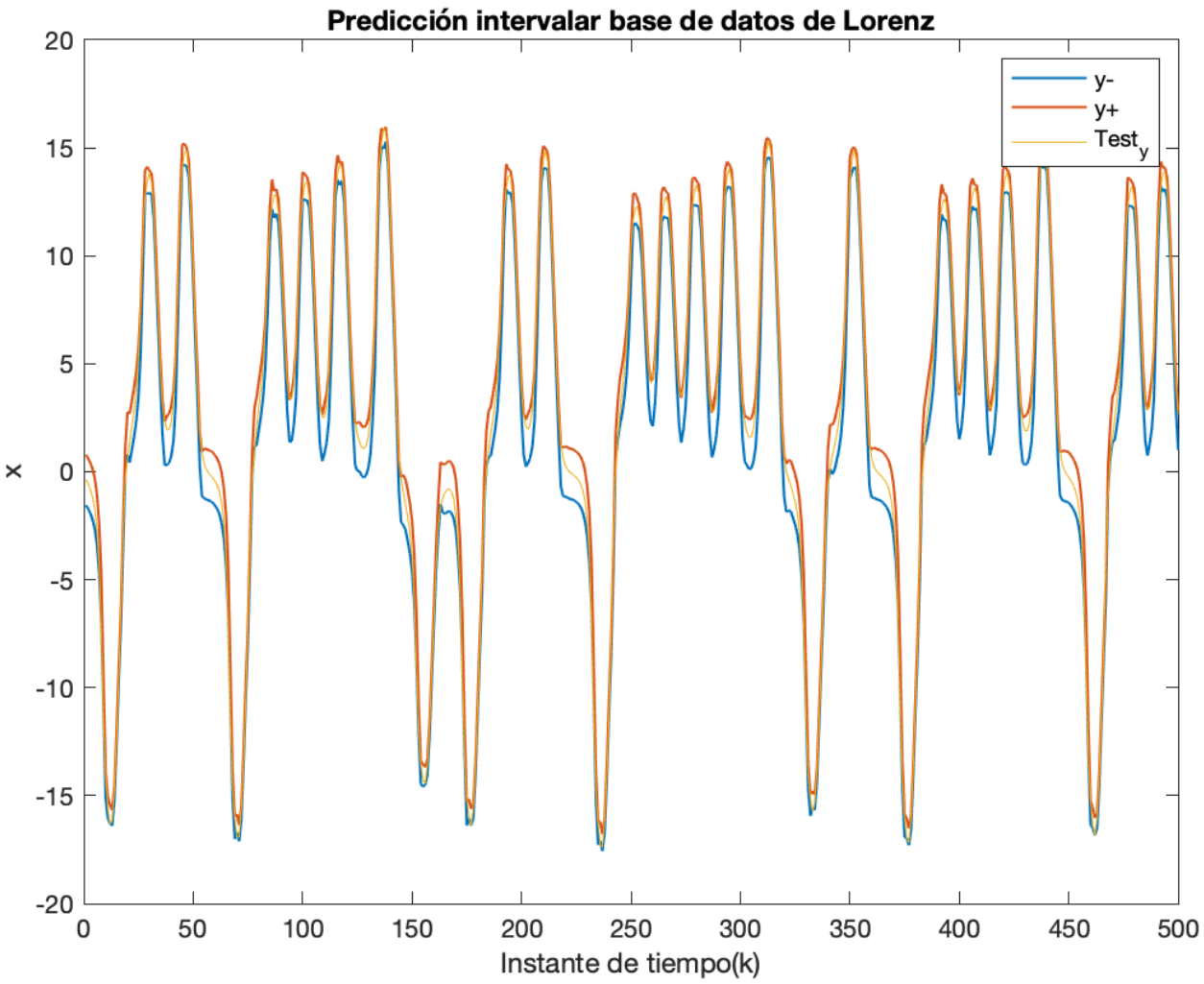

Ilustración 4-7. Predicción intervalar para γ=1.4, c=13.5, τ=0.1 y 350 en D

Se vuelve a realizar el análisis del tamaño medio de los intervalos obtenidos y la probabilidad empírica hallada.

Tamaño set de datos	Tamaño medio	Probabilidad empírica
200 puntos	1.6151	0.807
350 puntos	1.5865	0.800

Tabla 4-2. Predicciones para $\tau = 0.1$

Con todas las pruebas realizadas, se comprueba comparando las Tabla 4-1. Predicciones para $\tau = 0.05$ y Tabla 4-2. Predicciones para $\tau = 0.1$ que tal y como se esperaba al disminuir τ y forzar que la probabilidad del intervalo que pretende contener a la salida disminuya, el tamaño medio de los intervalos decrece. Para un mismo valor de τ, incrementar el número de puntos en el conjunto de datos D hace que se puedan obtener intervalos más finos, disminuyendo el tamaño medio de los mismos y afinando la probabilidad empírica, a cambio de incrementar el coste computacional.

Una interesante prueba sería analizar como varía el intervalo de predicción hallado aumentando el horizonte de predicción, por ejemplo, dado un regresor $x_k = [y_{k-1}, y_{k-2}]$ predecir el intervalo de predicción que contenga a y_{k+1} o y_{k+2} de este modo se estaría intentado predecir el valor a la salida 2 o 3 pasos hacia delante, en las

pruebas anteriores siempre se calculaba el intervalo para y_k, es decir, se predecía un paso hacia delante. De este modo, se realizan dichas pruebas para un conjunto D de tamaño 200 y un valor de $\tau = 0.05$ e intentando predecir 2 y 3 pasos hacia delante.

Primero se va a intentar calcular las predicciones 2 pasos hacia delante, una forma sencilla de implementarlo es modificando el regresor para la ejecución del algoritmo 1 con la siguiente estructura $x_k = [y_{k-2}, y_{k-3}]$ e intentando hallar el intervalo que contenga a y_k con cierta probabilidad. Aplicando a la base de datos de Lorenz se obtiene:

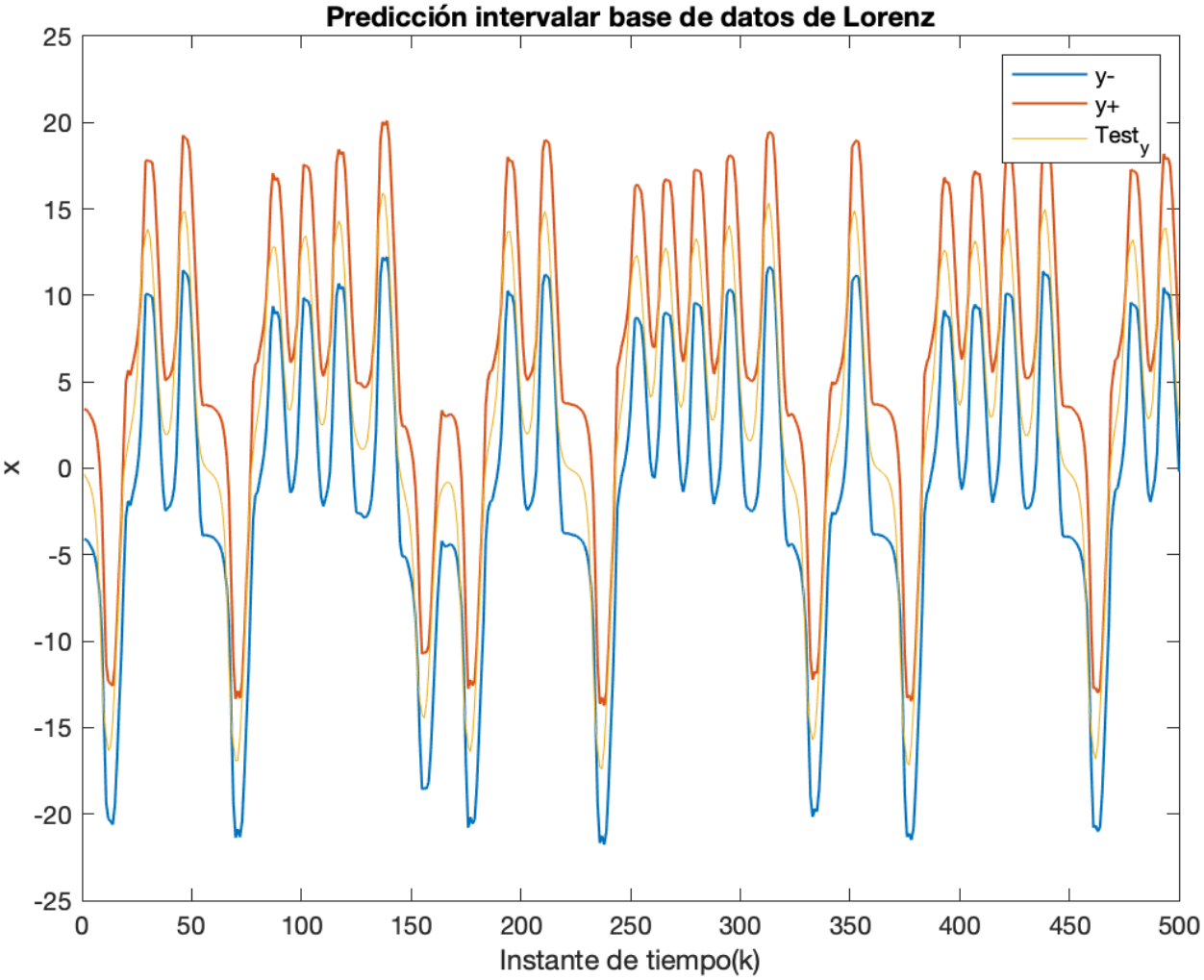

Ilustración 4-8. Predicción intervalar para γ=1.8, c=0.47, τ=0.05 y 200 en D. 2 pasos hacia delante.

Para obtener la gráfica de arriba se ha tenido que repetir la búsqueda de los parámetros óptimos c y γ de nuevo haciendo uso de los algoritmos correspondientes. Se observa que al aumentar el horizonte de predicción el tamaño medio del intervalo aumenta considerablemente y por consecuente el valor de c debe disminuir.

Si se repite el proceso intentando predecir 3 pasos hacia delante, el regresor será $x_k = [y_{k-3}, y_{k-4}]$ para predecir el intervalo que contenga a y_k. Se obtiene la siguiente gráfica:

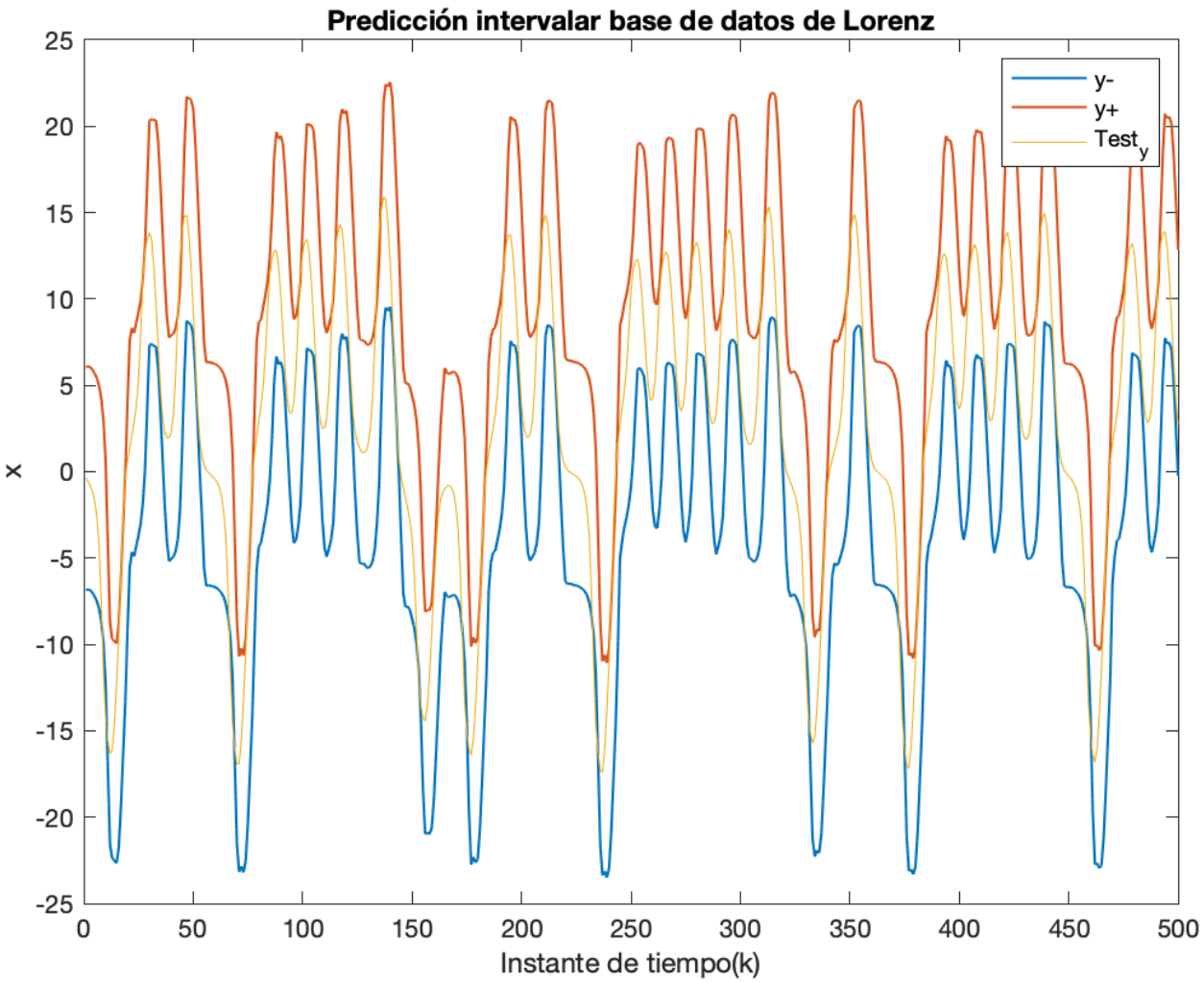

Ilustración 4-9. Predicción intervalar para γ=1.8, c=0.235, τ=0.05 y 200 en D. 3 pasos hacia delante

Se pueden analizar los resultados numéricos obtenidos para ambas pruebas con respecto al tamaño medio de los intervalos y la probabilidad empírica.

Horizonte de predicción	Tamaño medio	Probabilidad empírica
1 paso hacia delante	2.0105	0.918
2 pasos hacia delante	7.6656	0.902
3 pasos hacia delante	12.9946	0.902

Tabla 4-3. Predicciones para $\tau = 0.05$, 200 en D

Comparando los resultados de la Tabla 4-3, se concluye que si se quiere mantener la probabilidad con la que el intervalo hallado contiene a la salida y se aumenta a la vez el horizonte de predicción, el tamaño medio del intervalo calculado aumentará, cuanto más se aumente dicho horizonte de mayor tamaño será el intervalo obtenido.

4.3 Análisis del parámetro c y γ

Una vez se ha mostrado cómo funciona el algoritmo 1 cuando se ha obtenido la pareja de parámetros óptima $(\gamma_\tau^*, c_\gamma)$. Es sencillo mostrar gráfica y numéricamente que pasaría si los parámetros c y γ varían de sus valores óptimos, de esto modo se demostrará la importancia de obtener los valores adecuados para ejecutar predicciones de intervalos precisas.

Para dicho análisis de utilizará la base de datos de Lorenz, un conjunto de datos D de tamaño 200 y un valor de $\tau = 0.05$. Para dicho ejemplo ya se conocen los valores óptimos (véase Ilustración 4-1. Predicción intervalar para γ=2.2, c=5.5, τ=0.05 y 200 en D por lo que se irán variando ambos parámetros y analizando los resultados obtenidos.

Para el análisis de este apartado se va a ir variando los valores de c y γ y observando el efecto que tienen en el tamaño medio de los intervalos y en la probabilidad empírica.

c	**γ**	**Tamaño medio**	**Probabilidad empírica**
5.5 (óptimo)	2.2 (óptimo)	2.0105	0.918
25	2.2 (óptimo)	1.4526	0.712
0.5	2.2 (óptimo)	3.2516	0.984
5.5 (óptimo)	3	1.9487	0.888
5.5 (óptimo)	0.4	3.3701	0.988

Tabla 4-4. Análisis de resultados τ=0.05 y 200 en D

Se observa que a medida que aumenta el valor de c respecto del óptimo hallado mediante el algoritmo 2, la probabilidad empírica disminuye respecto a el valor requerido mediante τ, dejando muchas más salidas fuera del intervalo de predicción calculado de lo esperado. Con respecto a la variación de γ, según se observa en Ilustración 4-6. Maximización de ratio L_γ en función de γ. τ=0.1 o en Ilustración 4-3. Maximización de ratio L_γ en función de γ. τ=0.05 al tomar un valor de γ=3 cuyo ratio no se diferencia mucho del óptimo no se observan muchas diferencias, pero al tomar un valor de 0.4 cuyo ratio difiere bastante del óptimo si se observa que el tamaño medio y la probabilidad empírica aumentan respectivamente.

Este análisis resalta la importancia de los algoritmos 2 y del cálculo del γ óptimo a la hora de implementar el algoritmo 1 para predecir una cantidad indefinida de puntos. La elección de los parámetros c y γ no es trivial y por esto ha sido necesaria la implementación de 2 algoritmos para calcularlos. La correcta ejecución de los Algoritmos 2 y del γ óptimo proporcionará predicciones de intervalos del menor tamaño posible y que contengan a las salidas con la probabilidad preestablecida, es decir, predicciones óptimas.

5 APLICACIÓN 2. SP500

En algún lugar, algo increíble está esperando ser conocido.

- Carl Sagan-

En este capítulo se van a repetir algunos de los experimentos más importantes del apartado anterior pero con una base de datos diferente, demostrando que los algoritmos implementados en este TFG pueden ser aplicados en ejemplos más cotidianos que un sistema de ecuaciones como se ha visto hasta ahora.

Para este apartado se expondrán directamente los resultados del algoritmo 1 para la pareja de parámetros óptima hallada, ya que el análisis de c y γ ya se realizó con la base de datos anterior y en ambos casos los resultados son análogos.

5.1 Introducción a la base de datos

Para este apartado se usará una base de datos proporcionada por los tutores de este TFG del índice bursátil SP500. Este índice bursátil es el principal de EE. UU, se creó en 1923 para un listado de 233 empresas, pero hasta el año 1957 no alcanzó las 500 empresas, Dichas empresas son las más importantes de la economía americana: Apple, Microsoft, Tesla, Amazon, NVIDIA... El SP500 es el índice bursátil más representativo del mundo ya que sus movimientos afectan de manera directa al resto de mercados y sus activos relacionados. Se puede decir que este índice es un medidor del pulso de la economía global.

Se pretende predecir un paso hacia delante el valor del SP500, denotándolo como y_k para aplicar los algoritmos, haciendo uso de los 2 valores previos del mismo, que será el regresor $x_k = [y_{k-1}, y_{k-2}]$ a emplear como entrada.

La base de datos se compone del valor de índice bursátil SP500 a lo largo de los años, se elegirá un tramo de dicha base para crear un set de datos D de 200 puntos, un set de validación V de 1000 puntos y conjunto para test T de otros 1000 puntos. El conjunto Γ será tomado como diferentes valores entre los límites [0,3], con un paso de 0.2. Y el conjunto de salidas $\bar{Y}$ será implementado como 1000 puntos existentes entre el límite $[0.75 x Valor\ mínimo\ datos, 1.25 x\ Valor\ máximo\ datos]$

5.2 Predicción de intervalos

Se buscará predecir los intervalos que contengan a las salidas con una probabilidad mayor que 0.9, es decir, se usará un valor de $\tau = 0.05$. Antes se buscará la pareja $(\gamma_\tau^*, c_\gamma)$ de parámetros óptimos, cuya importancia y análisis se hizo con el ejemplo anterior.

Se muestra una ilustración de los intervalos obtenidos con un conjunto D de tamaño 200 tras encontrar la pareja de parámetros haciendo uso del algoritmo 2 y de γ óptimo.

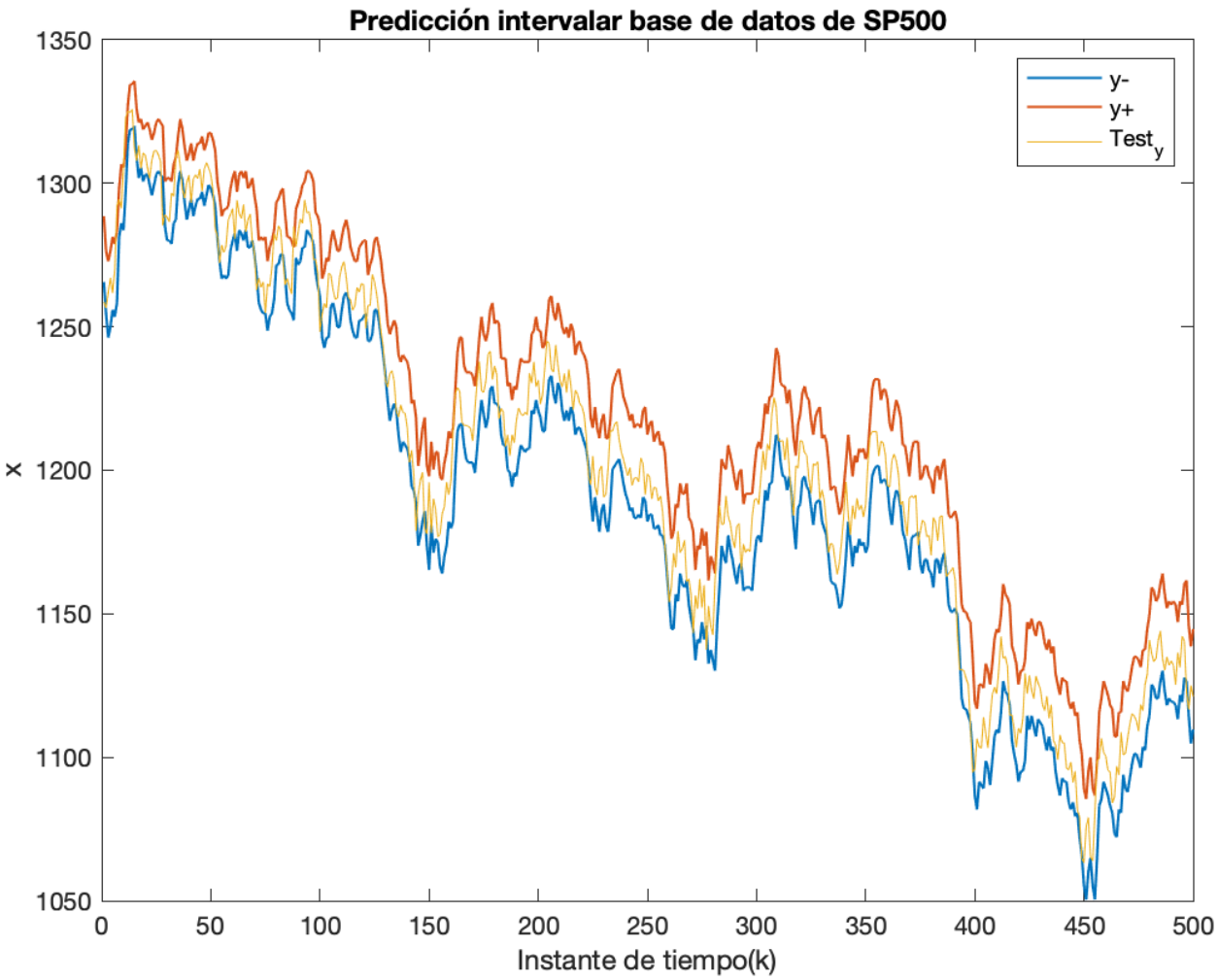

Ilustración 5-1. Predicción intervalar para γ=1.2, c=581.3, τ=0.05 y 200 en D

Una observación obvia es que el orden del intervalo hallado ha aumentado respecto del ejemplo anterior debido a que las variaciones del índice bursátil SP500 son mucho mayores que las variaciones dentro de la base de datos de Lorenz. Por ejemplo, el día 30 de agosto a las 14:00 el valor del índice era de 3976.71 y el día 29 de agosto a la misma hora era de 4053.39. De este modo, se observa gráficamente que las predicciones se adaptan medianamente bien a las salidas reales. En la siguiente ilustración se observa un zoom de la ilustración anterior para observar mejor los resultados.

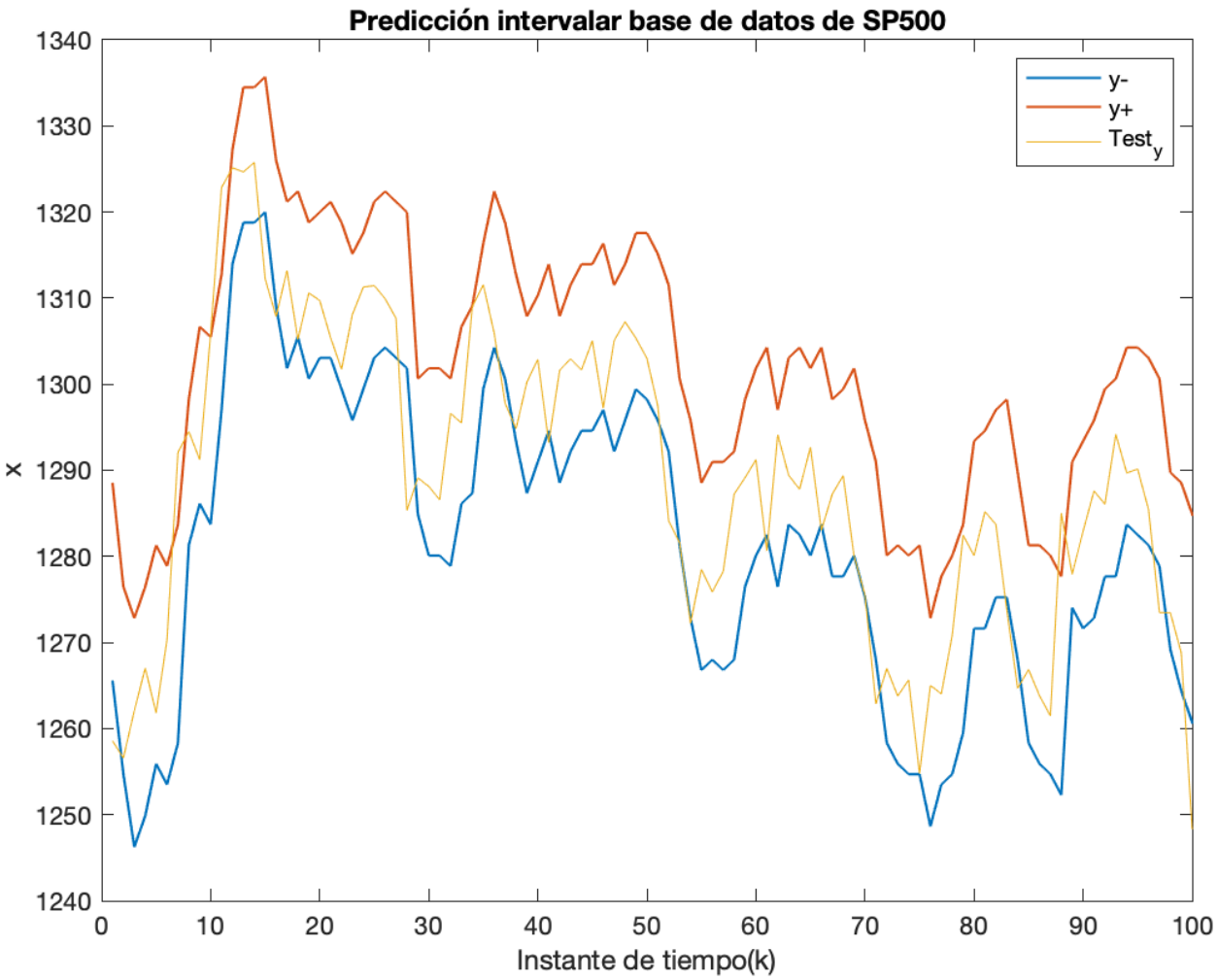

Ilustración 5-2. Predicción intervalar para γ=1.2, c=581.3, τ=0.05 y 200 en D. 100 puntos

A continuación, se repetirá el ejemplo forzando que los intervalos contengan a las salidas con una probabilidad mayor que 0.8, es decir, se cambiará el valor de τ a 0.1, manteniendo el tamaño del conjunto D.

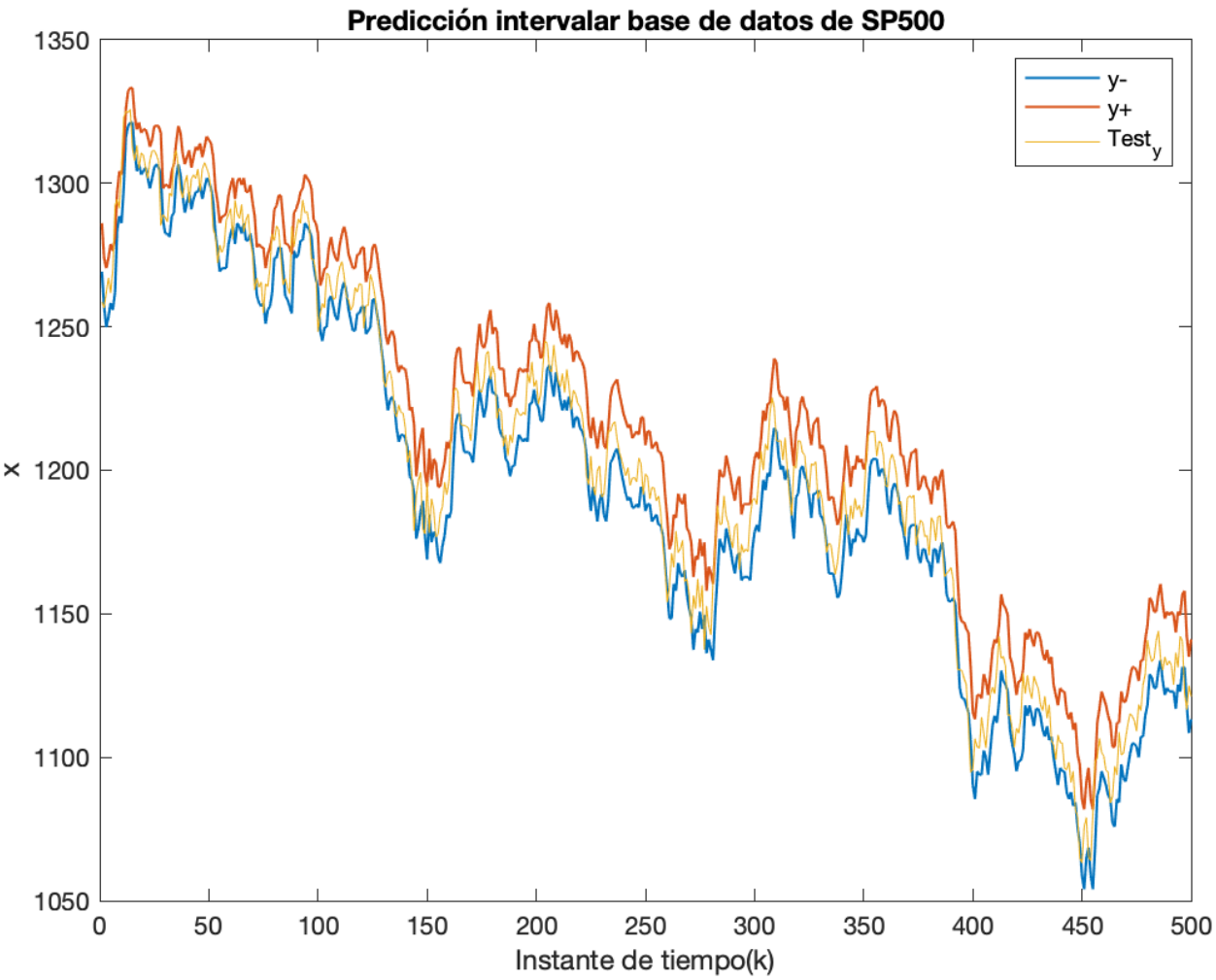

Ilustración 5-3. Predicción intervalar para γ=1.2, c=603.2, τ=0.1 y 200 en D

Por último, se compararán los resultados numéricos obtenidos en ambos ejemplos para ver que cumplen con las especificaciones y que son estimaciones coherentes.

τ	Tamaño medio	Probabilidad empírica
0.05	29.3346	0.906
0.1	23.0025	0.816

Tabla 5-1. Análisis de resultados para conjunto D de tamaño 200

Se observa que en ambos casos se cumple con la condición de probabilidad impuesta inicialmente y que el tamaño medio de los intervalos resulta coherente dadas las variaciones del SP500 y la reducción de τ, que provoca una reducción del tamaño medio de los intervalos calculados.

Al igual que como se hizo con la base de datos de Lorenz se puede probar a calcular los intervalos de predicción 2 o más pasos hacia delante en lugar de 1 como se venía haciendo, es decir, aumentar el horizonte de predicción. Para predecir 2 pasos hacia delante se va a tomar el regresor como $x_k = [y_{k-2}, y_{k-3}]$ intentando predecir el intervalo que contenga a y_k, realizando esta prueba para un conjunto de datos D de tamaño 200 y un valor de $\tau = 0.05$ se obtiene:

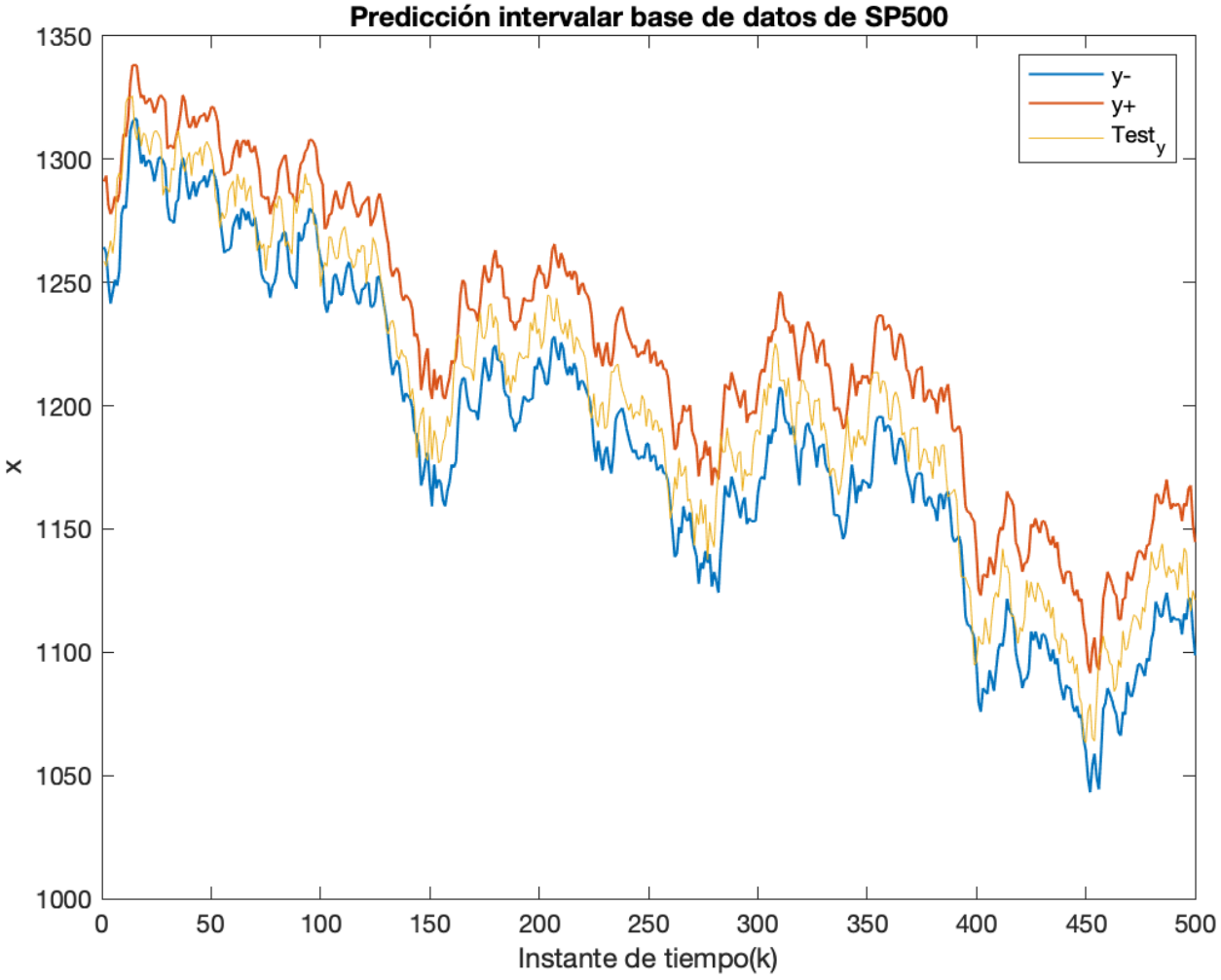

Ilustración 5-4. Predicción intervalar para γ=1.2 y c=148.4. 2 pasos hacia delante.

Para obtener la gráfica de la Ilustración 5-4 se ha tenido que volver a calcular los parámetros óptimos c y γ, de modo que se ha obtenido el menor tamaño de los intervalos que contienen a las salidas con una probabilidad mayor que 0.9. Se observa que el tamaño medio de los intervalos aumenta respecto a los obtenidos en la Ilustración 5-1 donde el horizonte de predicción era de solo 1 paso hacia delante con el regresor $x_k = [y_{k-1}, y_{k-2}]$.

Si se muestra una menor cantidad de puntos se puede apreciar de mejor modo como aumenta el tamaño medio de los intervalos (véase Ilustración 5-5)

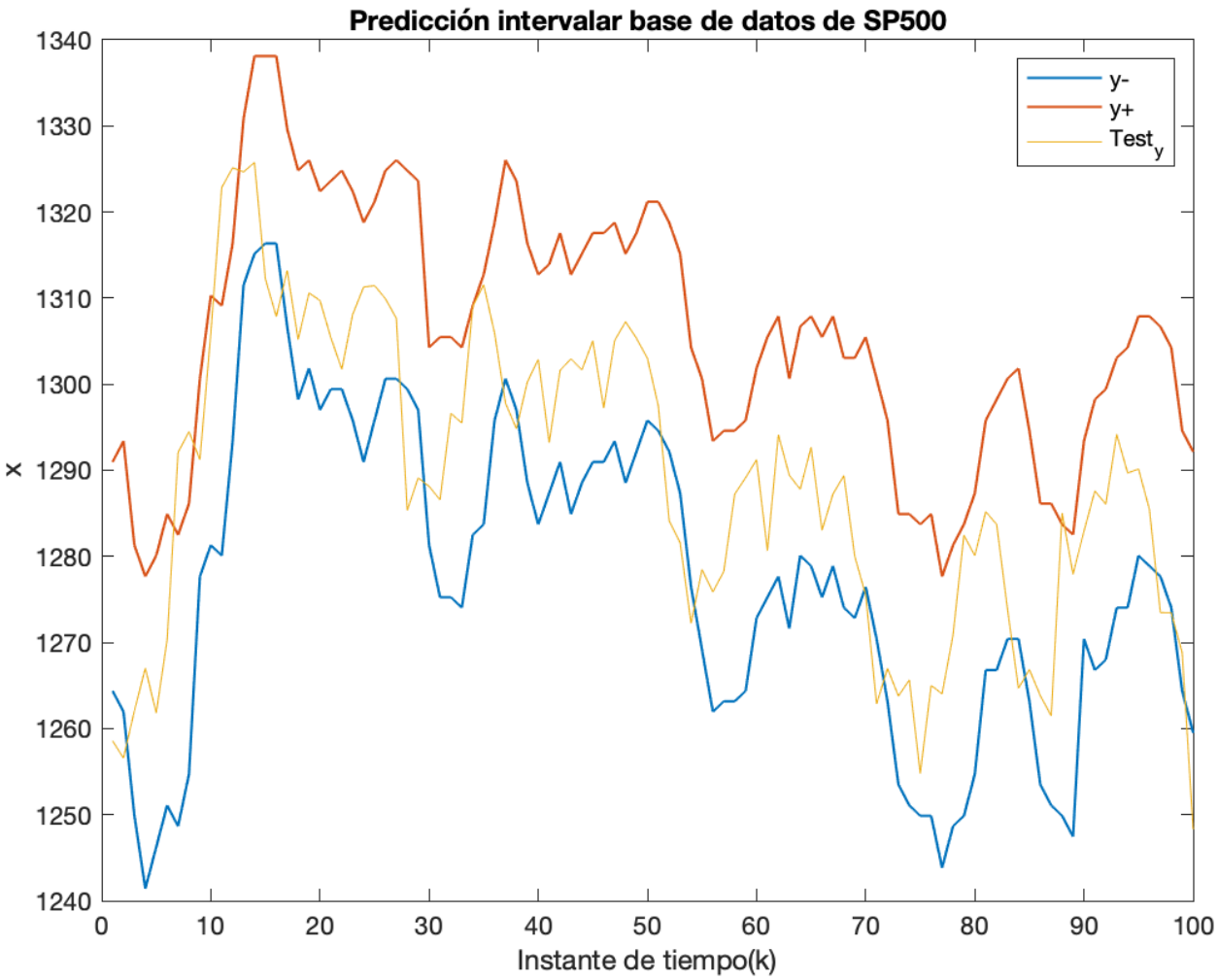

Ilustración 5-5. Predicción intervalar para γ=1.2 y c=148.4. 2 pasos hacia delante.100 puntos.

Ahora se va a intentar aumentar todavía más el horizonte de predicción intentado predecir 3 pasos hacia delante, para ello se tomará un regresor $x_k = [y_{k-3}, y_{k-4}]$ para intentar predecir un intervalo que contenga a y_k con cierta probabilidad, obteniendo los siguientes resultados:

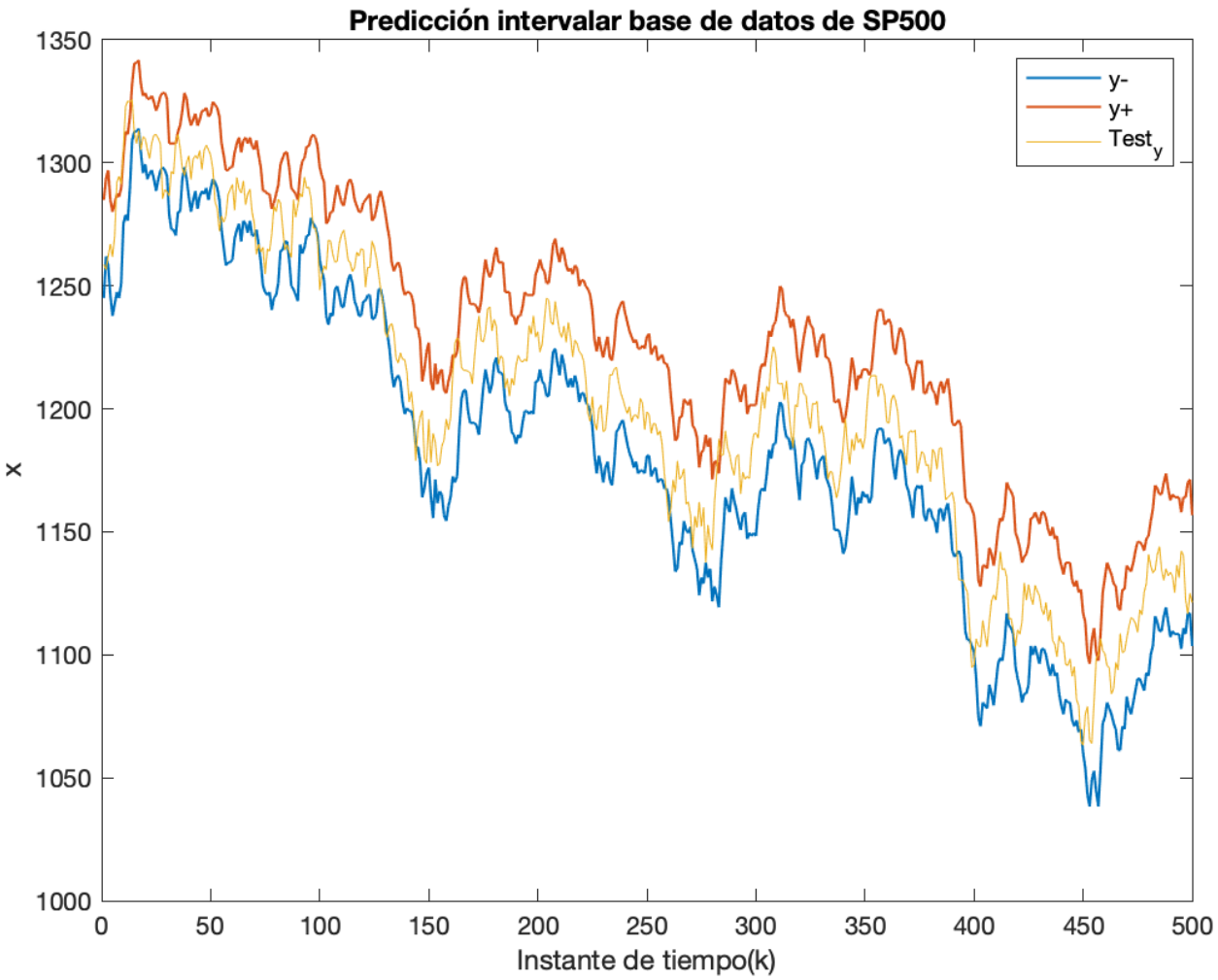

Ilustración 5-6. Predicción intervalar para γ=1.4 y c=85.6. 3 pasos hacia delante.

Se pueden analizar los resultados numéricos obtenidos para ambas pruebas con respecto al tamaño medio de los intervalos y la probabilidad empírica.

Horizonte de predicción	Tamaño medio	Probabilidad empírica
1 paso hacia delante	29.3346	0.906
2 pasos hacia delante	39.3610	0.906
3 pasos hacia delante	47.1776	0.906

Tabla 5-2. Predicciones para $\tau = 0.05, 200$ en D. SP500

Del mismo modo que se hizo con la base de datos de Lorenz, si se comparan los resultados de la Tabla 5-2, se observa que cuanto más se aumente el horizonte de predicción y si se quiere mantener la probabilidad con la que el intervalo calculado contiene a la salida, de mayor tamaño será el intervalo obtenido, es decir, la precisión disminuirá considerablemente.

6 REGRESIÓN CUANTÍLICA

En principio la investigación necesita más cabezas que medios.

- Severo Ochoa-

En este apartado se va a ejecutar otro método útil para generar intervalos de predicción como es la Regresión Cuantílica. Dicho método modela la relación entre un conjunto de variables predictoras y percentiles específico (o cuantiles) de una variable de destino, es una extensión de la regresión lineal por mínimos cuadrados. Este método permite obtener un determinado cuantil de una variable especificada, de este modo si se usa con los cuantiles adecuados se pueden obtener intervalos de predicción que contengan a las salidas con cierta probabilidad predefinida, tal y como si ha hecho con el algoritmo 1 explicado en el transcurso de este TFG.

La implementación de dicho método no se abordará en este trabajo, los tutores de este han proporcionado una función que implementa dicho método para poder realizar pruebas y comparar los resultados obtenidos con el método explicado anteriormente.

6.1 Predicción de intervalos y comparativa

Se buscará aplicar dicho método para las bases de datos de Lorenz y del SP500, con diferentes tamaños del conjunto D y probabilidades τ, analizando los resultados obtenidos.

Se comenzará aplicándolo para la base de datos de Lorenz, usando diferentes tamaños para el set de datos D. Si se quiere hallar el intervalo que pretenda contener a las salidas con una probabilidad mayor que 0.9, es decir, un valor de $\tau = 0.05$, se tendrán que calcular los cuartiles 0.05 para el límite inferior del intervalo y 0.95 para el límite superior. De este modo se obtienen los siguientes resultados prediciendo 500 puntos como en el método anterior.

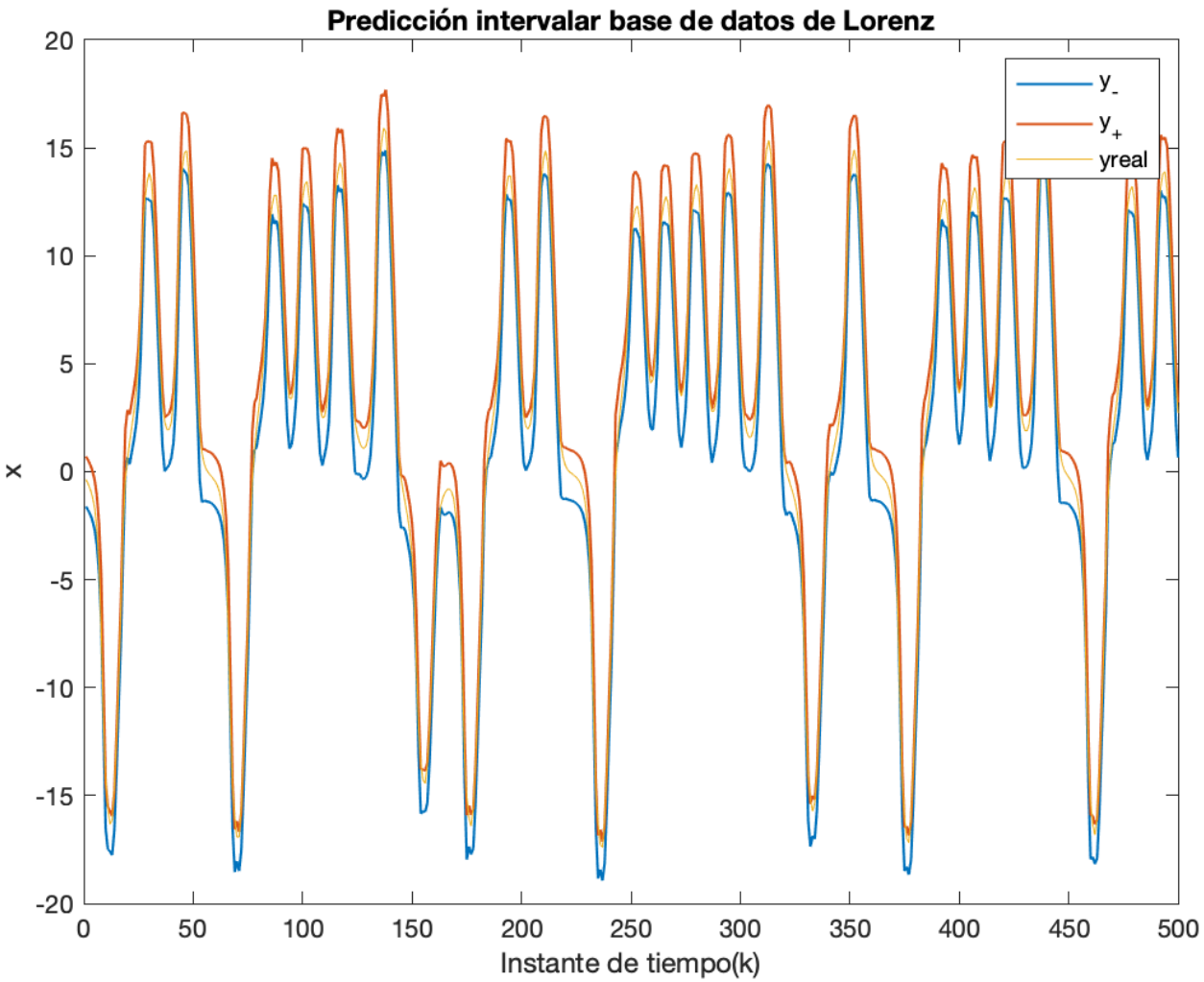

Ilustración 6-1. Predicción intervalar para Quantile Regression, $\tau = 0.05$ y 200 en D

Si se muestran solo 100 puntos (véase Ilustración 6-2) se puede ver más claramente las diferencias respecto a la Ilustración 4-2 calculada con el método desarrollado en este TFG, donde se observa que tanto el tamaño medio de los intervalos como la probabilidad empírica empeoran.

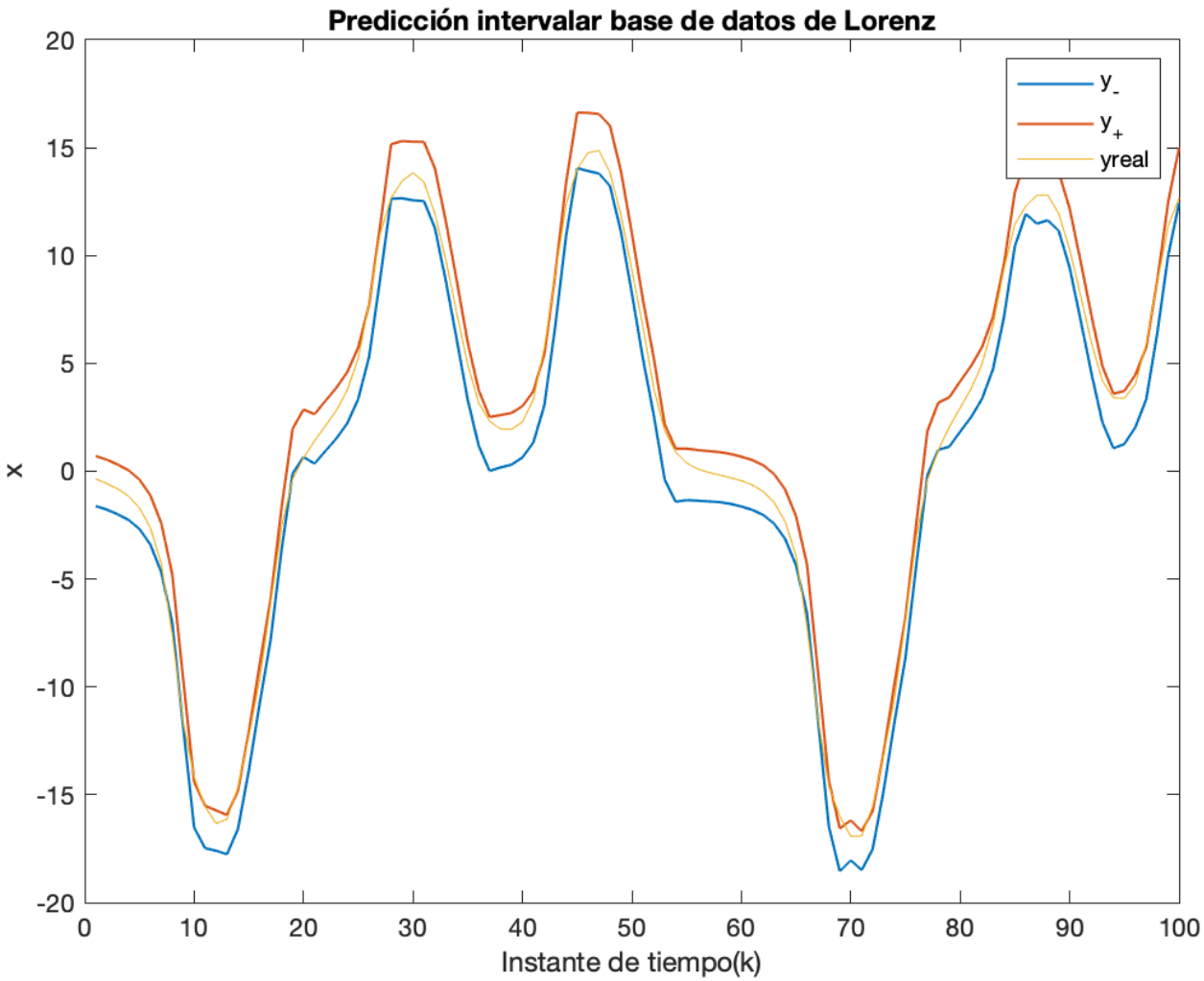

Ilustración 6-2. Predicción intervalar para Quantile Regression, $\tau = 0.05$ y 200 en D. 100 puntos.

Aumentando el tamaño del conjunto de datos D a 350 puntos se obtienen resultados muy parecidos a los anteriores usando un tamaño de 200.

Ahora si se quiere que los intervalos contengan a las salidas con una probabilidad mayor que 0.8, se tendrá que calcular los cuartiles 0.1 y 0.9 para los límites inferior y superior, respectivamente. Se obtienen gráficas muy parecidas a las anteriores.

Para un mejor análisis se expondrán los resultados numéricos del tamaño medio de los intervalos y la probabilidad empírica para todos los casos ejecutados.

Tamaño de D	τ	**Tamaño medio**	**Probabilidad empírica**
200	0.05	2.4046	0.8720
200	0.1	1.9557	0.7860
350	0.05	2.3282	0.8720
350	0.1	1.7691	0.7700

Tabla 6-1. Análisis de base de datos de Lorenz con método Quantile Regression

Se puede comparar los resultados de la Tabla 6-1con los obtenidos en las tablas Tabla 4-1 y Tabla 4-2, (véase Tabla 6-2) se observa que en todos los casos el tamaño medio de los intervalos es mayor cuando se usa el algoritmo de la regresión cuantílica. Además, la condición de la probabilidad empírica especificada para cada caso no se cumple en ninguna aplicación para la Regresión Cuantílica, cuando con los algoritmos 1, 2 y del γ óptimo se cumplía de manera óptima, es decir, minimizando el tamaño del intervalo hallado y conteniendo a las salidas con la probabilidad deseada. Además, si se observan solamente los resultados obtenidos para la Regresión Cuantílica se observa lo mismo que con el anterior método, al disminuir el valor de τ y/o aumentar el número de puntos del conjunto D, el tamaño medio de los intervalos disminuye y la probabilidad empírica se acerca a la teórica.

Método	**τ**	**Tamaño de D**	**Tamaño medio**	**Probabilidad empírica**
Quantile Regression	0.05	200	2.4046	0.872
Algoritmo 1	0.05	200	2.0105	0.918
Quantile Regression	0.05	350	2.3282	0.872
Algoritmo 1	0.05	350	1.9540	0.902
Quantile Regression	0.1	200	1.9557	0.786
Algoritmo 1	0.1	200	1.6151	0.807
Quantile Regression	0.1	350	1.7691	0.770
Algoritmo 1	0.1	350	1.5865	0.800

Tabla 6-2. Comparativa Quantile Regression y Algoritmo 1 con base de Lorenz

A continuación, se repetirán los cálculos usando la base de datos del índice bursátil SP500 para ver cómo se comporta ante el método de la regresión cuantílica. Primero se analizará la gráfica obtenida cuando se ejecuta el algoritmo de predicción para un tamaño de 200 en el conjunto D y un valor de $\tau = 0.05$.

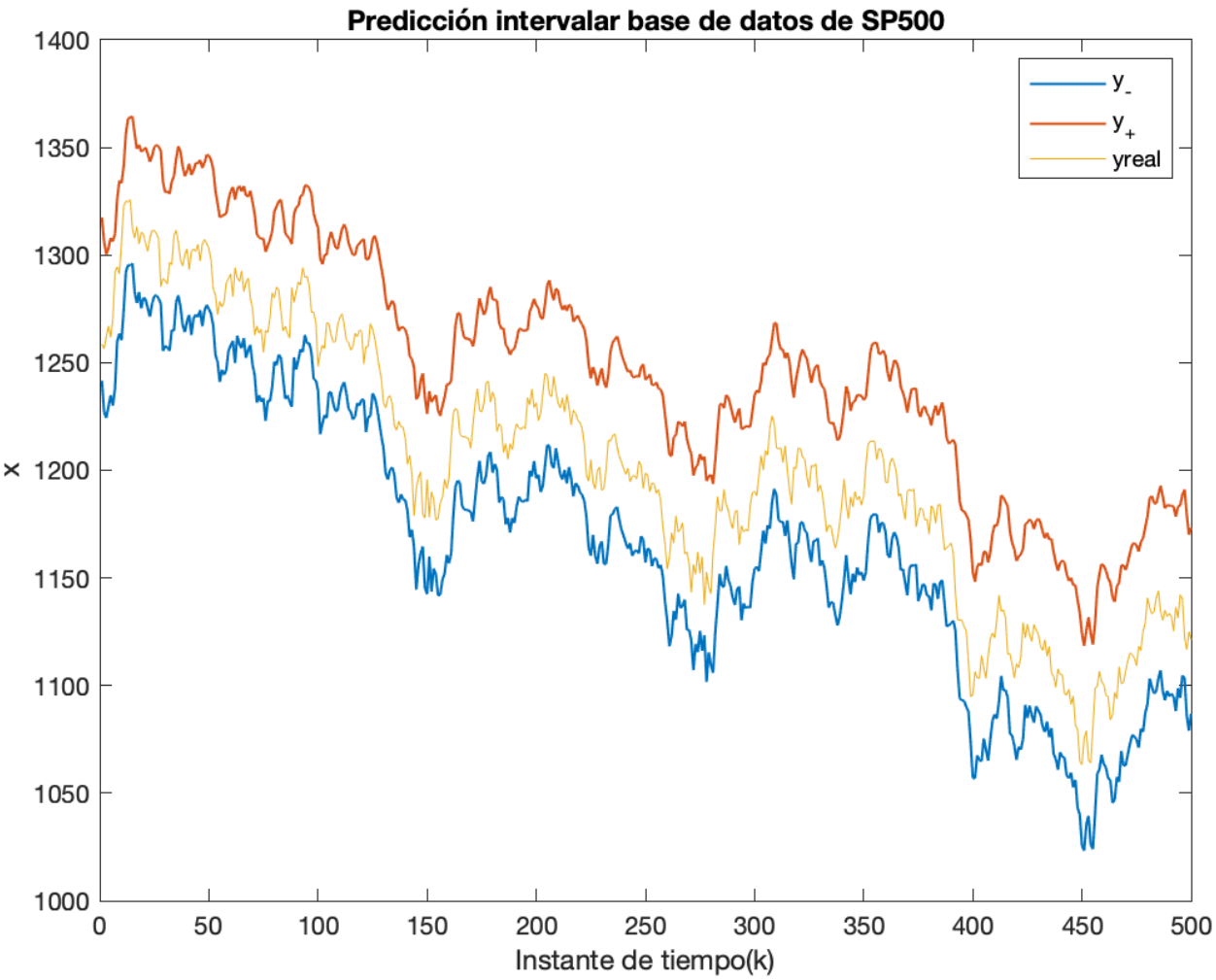

Ilustración 6-3. Predicción intervalar para Quantile Regression, $\tau = 0.05$ y 200 en D. SP500

Si comparamos la gráfica de la Ilustración 6-3 con la gráfica de la Ilustración 5-1 se observan que los intervalos hallados son de mucho mayor tamaño que los obtenidos con los algoritmos nuevos implementados en este TFG. Además, al tener un tamaño tan elevado la probabilidad empírica toma un valor de 1, asegurando que los intervalos calculados contienen a las salidas, en cualquier caso.

Ahora se prueba a variar el valor de τ.

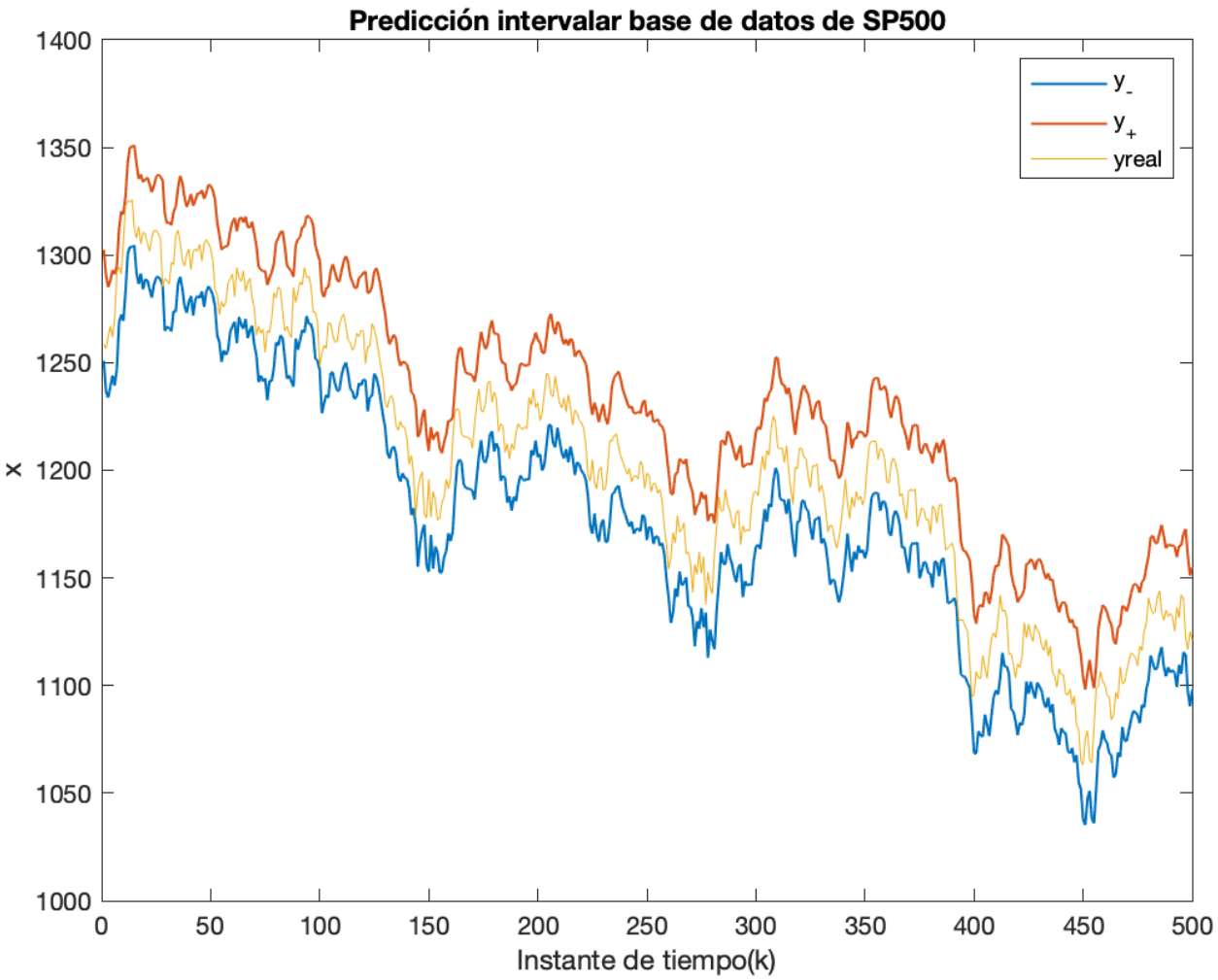

Ilustración 6-4. Predicción intervalar para Quantile Regression, $\tau = 0.1$ y 200 en D. SP500

Se observa que al tomar un valor de $\tau = 0.1$ los resultados mejoran, pero siguen siendo demasiado pobres con respecto a lo visto con el anterior método de predicción de intervalos con los mismos parámetros de ejecución.

Se realizarán más pruebas para un mayor número de puntos en el conjunto D y se analizarán los resultados numéricos tal y como se hizo con la base de datos de Loren anteriormente.

Tamaño de D	τ	Tamaño medio	Probabilidad empírica
200	0.05	81.0412	1
200	0.1	54.2205	0.998
350	0.05	81.2650	1
350	0.1	58.9191	1

Tabla 6-3. Análisis de base de datos del SP500 con método Quantile Regression

Se comparan la Tabla 6-1 con la Tabla 5-1y se observa que las predicciones halladas mediante el método de la regresión cuantílica son demasiada grandes y se encuentran muy lejos de los valores óptimos (véase Tabla 6-4). Se destaca que en este caso las condiciones de probabilidad especificadas se cumplen en ambos casos, pero es a

posta de tener un tamaño medio de los intervalos demasiado grande, asegurando en cualquier caso su cumplimiento, pero sin llegar a ser resultados óptimos.

Método	**τ**	**Tamaño de D**	**Tamaño medio**	**Probabilidad empírica**
Quantile Regression	0.05	200	81.0412	1
Algoritmo 1	0.05	200	29.3346	0.906
Quantile Regression	0.1	200	54.2205	0.998
Algoritmo 1	0.1	200	23.0025	0.816

Tabla 6-4. Comparativa Quantile Regression y Algoritmo 1 con base de SP500

Se concluye que la Regresión Cuantílica usada para comparar los algoritmos implementados en el desarrollo de este trabajo no es útil para hallar predicciones de intervalos cuando se necesita precisión en los resultados, a cambio este método es muy rápido y tiene un bajo coste computacional por lo que podría servir para obtener una rápida pero no precisa estimación inicial, según la base de datos empleada. Esto hace notar que la obtención de intervalos de predicción que se adapten y contengan de manera adecuada a las salidas no es un problema trivial, y hace valer los algoritmos desarrollados en este TFG.

También hay que destacar que el comportamiento del índice bursátil SP500 es muy impredecible y variable por lo que es una tarea muy complicada obtener un intervalo de predicción preciso que contenga a la salida para cierta probabilidad predefinida, por lo que es normal tener un tamaño medio de los intervalos tan elevado con respecto a los ejemplos realizados con la base de datos de Lorenz, donde las salidas variaban en todo momento entre un intervalo fijo de [12,-15].

7 CONCLUSIONES

No entiendes realmente algo a menos que seas capaz de explicárselo a tu abuela.

- Albert Einstein-

El objetivo principal de este trabajo de fin de grado es implementar un algoritmo que calcule intervalos de predicción para un conjunto de salidas con cierta probabilidad. Este tipo de cálculos requieren un elevado coste computacional de modo que se pretenderá reducir lo máximo para su implementación en Matlab.

Al principio, se introdujo la familia de funciones de disimilitud a emplear que no es más que presentar un problema de optimización convexa dependiente de parámetro γ. Para su implementación en Matlab se ha probado 3 métodos distintos como son: las funciones propias de Matlab **fmincon** y **quadprog**, y el algoritmo **FISTA** (proporcionado por los tutores de este TFG), comprobando que el método más eficiente para resolver un problema de elevado coste computacional en Matlab es el algoritmo **FISTA**. Esto junto a una correcta elección del conjunto $\bar{y}$ reducirán el coste computacional lo suficiente.

Tras ello, se procedió a explicar el algoritmo 1 que realiza los cálculos a raíz del valor de disimilitud obtenido para crear una función de densidad empírica con la que calcular los intervalos de predicción en función de la pareja de parámetros c y γ. El tamaño del conjunto datos D con el que se calcula la disimilitud resulta de vital importancia para los resultados finales ya que, si se aumenta dicho tamaño, el tamaño medio de los intervalos y la probabilidad empírica disminuirán, pero esto solo ocurrirá si la base de datos es lo suficientemente representativa. Por ejemplo, en el ejemplo de la base de datos de Lorenz aumentar el tamaño de D si aporta resultados óptimos debido a que dicha base de datos resulta lo suficientemente representativa.

Se analizó también la importancia de obtener una pareja de parámetros c y γ óptima, haciendo uso de los algoritmos 2 y del γ óptimo. Esta pareja proporcionará el menor tamaño del intervalo de predicción que se ajuste a las salidas cumpliendo cierta probabilidad, es decir, que la mayoría de las salidas estén contenidas en el intervalo calculado. Se observó que la elección de γ óptimo es menos crítica que la de c, ya que la maximización del ratio para hallar su valor óptimo se estabiliza a partir de cierto valor de γ (véase Ilustración 4-3).

Aplicando los algoritmos explicados a diferentes bases de datos como la de Lorenz y el SP500, se ha demostrado que en función de la base de datos a predecir la precisión de los intervalos hallados variará. Cuando más impredecible y variable sea la base de datos menos precisa será la predicción.

Para comprobar que los algoritmos implementados son verdaderamente eficientes y que el cálculo de los parámetros c y γ no es un asunto trivial, se han aplicado ambas bases de datos presentadas a otro famoso método capaz de devolver intervalos de predicción como la Regresión Cuantílica. Analizando los resultados conseguidos con este último método se ha comprobado que no es capaz de cumplir con las restricciones de probabilidad requerida de manera precisa para ningún conjunto de datos, además el tamaño medio de los intervalos hallados es más preciso con los algoritmos implementados que con la Regresión Cuantílica.

Por último es importante comentar que siempre se ha buscado obtener cierta eficiencia y reducir el coste computacional en Matlab, para ello se han usado diferentes funciones y estrategias como: uso de bucles **parfor** en Matlab con los que es posible realizar diferentes ejecuciones de un bucle **for** en paralelo, aproximaciones de puntos iniciales para algoritmo FISTA lo que reduce considerablemente su coste computacional y elegir un

conjunto de salidas $\bar{y}$ que se adapte bien al orden de las salidas a obtener y que no sea exageradamente grande.

8 CÓDIGO DE MATLAB

La ciencia de hoy es la tecnología del mañana.

- Edward Teller -

En este último apartado se expondrán los diferentes archivos de Matlab implementados para la ejecución de todas las pruebas y la elaboración de gráficas. Matlab es de los programas matemáticos más usados en el transcurso del Grado en Ingeniería de Tecnologías Industriales y por eso se ha elegido para la realización de este TFG. Además, Matlab es una herramienta tremendamente versátil con la que se puede realizar simulaciones respecto a diferentes campos de la tecnología, así como resolver problemas numéricos variados.

8.1 Creación de base de datos

Se expondrá el código que es capaz de crear los conjuntos de datos D, de validación $\mathcal{V}$ y de test $\mathcal{S}$, a partir de las bases de datos proporcionadas mediante un archivo con extensión ".mat". El tamaño de los conjuntos de datos y del regresor puede ser modificado según el ejemplo, para variar de una base de datos a otro solamente habría que modificar la línea 3.

```
%% Data Set
clear all; clc;
load('Dataset_Lorenz.mat')

tam_regr = 2;
x_data = x_data';
[N,~] = size(x_data);

cnt=1;
for j=tam_regr+1:1:N-tam_regr
    if(x_data(j-tam_regr)~=0 && j+tam_regr<=N)
        Dataset_x(cnt,:) = x_data(j:-1:j-tam_regr+1);
        Dataset_y(cnt,:) = x_data(j+1:j+tam_regr);
        cnt=cnt+1;
    end
end

Validation_x = Dataset_x(501:2000,:);
Validation_y = Dataset_y(501:2000,:);

Test_x = Dataset_x(2001:3500,:);
Test_y = Dataset_y(2001:3500,:);

Dataset_x = Dataset_x(1:200,:);
Dataset_y = Dataset_y(1:200,:);
```

8.2 Algoritmo 1

Se expone el algoritmo 1 explicado como función de Matlab.

```
function [y_mas,y_menos]=algoritmo1(x_pto,tau,gamma,c,D,y_gorro)

    tam=size(y_gorro);
    M=tam(:,2);
    vector_disi=zeros(1,M);
    Aeq=[D; ones(1,200)];
    T=0.5*ones(200,1);

    % PASO 1: Obtener el valor de la disimilitud para cada elemento de D
    x=zeros(4,1);
    for i=1:1:M
        beq=[y_gorro(i);x_pto;1];
        [x,u,J,iter]=Fista_Method(T,gamma,Aeq,beq,x);
        vector_disi(i)=J;
    end

    % PASO 2: Probabilidades condicionadas
    sum_exp=sum(exp(-c*vector_disi));
    vector_proba= exp(-c*vector_disi)/sum_exp;

    % PASO 3: indices mayor y menor de cuantiles condicionados que satisfacen 1- tau
    suma=cumsum(vector_proba);
    l_mas=find(suma>=(1-tau),1,'first');

    suma2=cumsum(vector_proba,2,'reverse');
    l_menos=find(suma2>=(1-tau),1,'last');

    % PASO 4: evaluar indices para hallar el intervalo pedido

    y_menos=y_gorro(l_menos);
    y_mas=y_gorro(l_mas);

end
```

8.3 Algoritmo 2

Se expone el algoritmo 2 explicado como función de Matlab.

```
function [c_optimo]=Algoritmo2_fun(cmax,cmin,regresor_V,tau,gamma,epsilon,y_gorro,NV,D,y_real)

    while(cmax-cmin >= epsilon)
    c=0.5*(cmax+cmin);
    nviol_mas=0;
    nviol_menos=0;

        parfor i=1:1:NV

        [yl_mas,yl_menos]=algoritmo1(regresor_V(i,:)',tau,gamma,c,D,y_gorro);
            if (y_real(i) > yl_mas)
            nviol_mas= nviol_mas + 1;
            elseif (y_real(i) < yl_menos)
            nviol_menos= nviol_menos + 1;
            end

        end

        if ((max(nviol_mas,nviol_menos)/NV)<tau)
        cmin=c;
        else
        cmax=c;
        end

    end
    c_optimo=cmin;

end
```

8.4 Algoritmo γ óptimo

Se expone el algoritmo para el cálculo del γ óptimo explicado como función de Matlab.

```
function [gamma_optimo]=Algoritmo_gamma(gamma,y_gorro,D,regresor_V,tau,c_optimo,y_vali)
    tam=size(gamma);
    N=tam(:,2);
    tam=size(y_gorro);
    M=tam(:,2);
    vector_disi=zeros(1,NV);
    vector_disi1=zeros(1,M);
    sumatorio_ygorro=zeros(1,NV);

    for j=1:1:N
        suma_max=-1e4;
        gamma0=gamma(j);
        Aeq=[D; ones(1,200)];
        T=0.5*ones(200,1);
        x=zeros(4,1);
        for i=1:1:NV
            beq1=[y_vali(i);punto1(i);punto2(i);1];
            [x,u,J,iter]=Fista_Method(T,gamma0,Aeq,beq1,x);
            vector_disi(i)=J;

            x=zeros(4,1);
            for k=1:1:M
                beq2=[y_gorro(k);regresor_V(i,:)';1];
                [x,u,J,iter]=Fista_Method(T,gamma0,Aeq,beq2,x);
                vector_disi1(k)=J;
            end
            sumatorio_ygorro(i)=sum(exp(-c_opt*vector_disi1));
        end
        suma_total=0;
        for w=1:1:NV
            suma_total = suma_total + log((exp(-c_opt*vector_disi(w)))/(sumatorio_ygorro(w)));
        end
        vector_maximos(j)=suma_total;

    end
[maximo,Indice]=max(vector_maximos);
gamma_optimo=gamma(Indice);
end
```

8.5 Método FISTA

Se expone el algoritmo FISTA proporcionado por los tutores de este TFG para resolver el problema de optimización convexa para hallar la disimilitud.

```
function [x,u,J,iter]=Fista_Method(T,alpha,A,b,x_inic)
    n=size(A,1);
    if (nargin==4)
        x=zeros(n,1);
    else
        x=x_inic;
    end
    B_T_sqrt=repmat(sqrt(T)',n,1);
    ATsqrt=A.*B_T_sqrt;
    H=ATsqrt*ATsqrt';
    R=chol(H)';
    S=R\eye(n);
    As=S*A;
    bs=S*b;
    iter=0;
    GoOn=1;
    t0=1;
    x_old=x;
    y=x;
    while (GoOn)

        t1=0.5*(1+sqrt(1+4*t0^2));

        u=c_u_Dual(As'*y,T,alpha);
        x_new=y-(As*u-bs);
        y=x_new+((t0-1)/t1)*(x_new-x_old);
        t0=t1;
        x_old=x_new;
        iter=iter+1;
        if (norm(A*u-b)<1e-4)
            GoOn=0;
        end
    end
    x=x_new;
    J=(1/2)*u'*(u./T) + alpha*sum(abs(u));
```

8.6 Regresión cuantílica

Se expone el método de la regresión cuantílica para obtener intervalos de predicción de una forma diferente a la explicada en este TFG.

```
function theta=c_QuartilRegressor(R,Y,rho,alpha)

n_theta=size(R,2);
N=length(Y);

I_theta=eye(n_theta);
I_tau=eye(N);
Z_tau=zeros(N);

H=[rho*I_theta zeros(n_theta,2*N); zeros(2*N,n_theta+2*N)];
f=[zeros(n_theta,1);alpha*ones(N,1);(1-alpha)*ones(N,1)];

A=[zeros(2*N,n_theta) -eye(2*N)];
b=zeros(2*N,1);
Ae=[R I_tau -I_tau];
be=Y;

z=quadprog(H,f,A,b,Ae,be);
theta=z(1:n_theta);
end
```

REFERENCIAS

[1] Carnerero, A. D., Rodriguez Ramirez, D., & Alamo, T. (2020). Probabilistic interval predictor based on dissimilarity function.

[2] Cornell University Computational Optimization Open Book. (s.f.). *Optimization with absolute values*. Recuperado el 04 de 2022, de https://optimization.cbe.cornell.edu/index.php?title=Optimization_with_absolute_values#Absolute_Values_in_Objective_Functions

[3] IBM. (s.f.). *Regresión Cuantil*. Obtenido de https://www.ibm.com/docs/es/spss-statistics/saas?topic=regression-quantile

[4] MathWork. (s.f.). *Find minimum of constrained nonlinear multivariable function*. Recuperado el 05 de 2022, de https://es.mathworks.com/help/optim/ug/fmincon.html?lang=en

[5] MathWork. (s.f.). *Quadratic programming*. Recuperado el 05 de 2022, de https://es.mathworks.com/help/optim/ug/quadprog.html