Proyecto Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Planificación de movimiento de vehículos acuáticos de superficie basado en Optimización Bayesiana

Autor: Pedro José Díaz García
Tutor: Daniel Gutiérrez Reina

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Proyecto Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Planificación de movimiento de vehículos acuáticos de superficie basado en Optimización Bayesiana

Autor:
Pedro José Díaz García

Tutor:
Daniel Gutiérrez Reina
Profesor Contratado

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Proyecto Fin de Grado: Planificación de movimiento de vehículos acuáticos de superficie basado en Optimización Bayesiana

Autor: Pedro José Díaz García

Tutor: Daniel Gutiérrez Reina

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A todos los que me han acompañado
en este camino.

Agradecimientos

Debo empezar por el principio, y este trabajo comienza con todas aquellas personas que me han brindado su apoyo, a las que quiero expresar mi más sincero agradecimiento:

A mi tutor Daniel Gutiérrez y a Federico Peralta por toda la ayuda prestada durante el desarrollo de este trabajo.

A mi familia por ser un punto de apoyo indiscutible tanto en el ámbito académico como en el personal. Gracias por creer en mí siempre.

A todos mis amigos, tanto fuera como dentro de clase, por soportarme en mis malos ratos, animarme en los peores y acompañarme en los mejores. En especial a Raúl, junto a quien volvería a recorrer este camino mil veces más.

Una vez más, porque nunca es suficiente, gracias a todos.

Pedro José Díaz García

Sevilla, 2020

Resumen

Este trabajo forma parte del proyecto de colaboración entre la Universidad de Sevilla (US) y la Universidad Nacional de Asunción (UNA) para la creación de un sistema de monitorización basado en una flota de vehículos autónomos acuáticos para supervisar el estado del lago Ypacaraí, con el principal objetivo de hacer frente a la contaminación creciente en este. Se trata de un proyecto amplio, en el sentido de que se trabaja en campos diferentes: desde el diseño de los vehículos acuáticos hasta la red de comunicación entre ellos, pasando por las tácticas de exploración que deberán llevar a cabo los vehículos. Respecto a esto último, se han realizado diversos estudios acerca de metodologías para la planificación del movimiento que deben realizar los vehículos sobre el lago, entre los que se encuentra el presente trabajo.

En concreto, este trabajo lleva a cabo un estudio de la Optimización Bayesiana y su adaptación al problema de planificación de la exploración del lago. El método propuesto resulta ser, bajo una correcta parametrización, adecuado para lograr el objetivo perseguido, en tanto que permite obtener un modelo aproximado del lago, el cual resulta muy útil en tareas de monitorización.

Abstract

This work is part of the collaborative project between the University of Seville (US) and the National University of Asuncion (UNA) to create a monitoring system based on a fleet of autonomous water vehicles to monitor the state of Lake Ypacaraí, with the main objective of addressing the growing pollution in this. This is a broad project, in the sense that it works in different fields: from the design of the water vehicles to the communication network between them, including the exploration tactics that the vehicles will have to carry out. Regarding the latter, several studies have been carried out on methodologies for planning the movement that the vehicles should carry out on the lake, among which is this work.

Specifically, this work carries out a study of the Bayesian Optimization and its adaptation to the problem of planning the exploration of the lake. The proposed method is, under a correct parameterization, suitable for achieving the desired objective, while allowing to obtain an approximate model of the lake, which is very useful in monitoring tasks.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

km	Kilómetros
km^2	Kilómetros cuadrados
m	Metros
p. ej.	Por ejemplo
$\propto$	Relación de proporcionalidad
exp	Función Exponencial
π	Número Pi
GP	Proceso gaussiano (*Gaussian Process*)
<	Menor que
>	Mayor que
$\mathcal{N}$	Distribución Normal
argmax	Argumento del máximo
max	Máximo

1 INTRODUCCIÓN

1.1 Antecedentes

El lago Ypacaraí constituye una de las zonas geográficas más importantes de Paraguay. Ya sea por su atractivo turístico, o por el uso de sus aguas para actividades agrícola-ganaderas e industriales, el lago es una pieza clave en el motor económico del país. Dada la importancia que tiene, la calidad de sus aguas no desmerece una atención especial.

Esta masa de agua dulce se sitúa a unos 37 km al este de la capital Asunción. Con un área total de unos 90 km^2 y una longitud de 24 km de norte a sur y 5 a 6 km de este a oeste, se posiciona como el lago más grande de Paraguay. Se trata de un lago poco profundo, teniendo una profundidad media de 3 m [1].

Figura 1-1. Geografía del lago Ypacaraí. A la izquierda, mapa de Paraguay, donde se indican las localizaciones del lago Ypacaraí y de la capital Asunción. A la derecha, mapa físico del lago Ypacaraí. Fuentes: https://upload.wikimedia.org/wikipedia/commons/e/e9/Paraguay_rel_location_map.svg

El lago Ypacaraí cumple un papel clave para la sociedad de la zona. Por una parte, funciona como fuente de abastecimiento de agua para los municipios cercanos. Las distintas poblaciones de la zona aprovechan esta agua tanto para consumo directo como para llevar a cabo actividades ganaderas, agrícolas e industriales. Por otro lado, también constituye un importante enclave turístico con especial relevancia en la temporada estival, durante la cual sus aguas acogen tanto a bañistas como a practicantes de diferentes deportes acuáticos.

En las últimas décadas, la presencia del lago ha favorecido el desarrollo económico y demográfico de la región. Sin embargo, esto ha conllevado un incremento de los residuos vertidos al lago a través de sus arroyos: desechos domésticos, urbanos e industriales, así como otros aportes difusos, en su mayoría inorgánicos, procedentes de la actividad agrícola-ganadera. Como consecuencia, ha tenido lugar un fenómeno conocido como eutrofización [2], en el que el vertido incontrolado de estos desechos da lugar a una acumulación excesiva de nutrientes, principalmente fósforo y nitrógeno, impidiendo que se lleve a cabo un ciclo de asimilación natural de estos. Adicionalmente, se produce una caída drástica en los niveles de oxígeno, provocando una gran disminución de la calidad del agua, haciéndola no apta para algunos de sus usos actuales, como son su uso doméstico o como escenario para actividades acuáticas.

Adicionalmente, la presencia de estos nutrientes en altas concentraciones favorece la aparición de cianobacterias, dando a las aguas un color verde frente al azulado que tenían anteriormente. El principal peligro de la aparición de cianobacterias es que la combinación de las floraciones de este tipo de alga junto a las cianotoxinas que producen puede ser un problema tanto para el ser humano como para otros organismos que habiten el medio acuático.

Aunque los niveles detectados de cianobacterias están por debajo de los niveles mínimos para que sea considerado peligroso, estos niveles deben ser tratados ya que estas algas producen un fuerte y desagradable olor que perjudica la imagen del lago, resultando poco atractivo para el ocio de sus usuarios.

Figura 1-2. Floración de cianobacterias.

Fuente: https://www.hoy.com.py/nacionales/vuelve-la-espuma-verde-al-lago-ypacarai

Con la finalidad de solventar este problema, se han llevado a cabo diferentes proyectos para tomar medida de la calidad del agua. Sin embargo, estos resultaban ser poco eficientes, ya que las mediciones eran poco continuas y se realizaban en puntos específicos, además de que requerían de la presencia de personal técnico para tomar las muestras. Es en 2016 cuando arranca el presente proyecto, con la promesa de innovar en los aspectos en los que los anteriores proyectos fallaban.

Este trabajo se enmarca en la línea de cooperación entre la Universidad de Sevilla (US) y la Universidad Nacional de Asunción (UNA) para la monitorización de la calidad del agua en el lago Ypacaraí. En el caso de este proyecto, se pretende llevar a cabo la toma de datos empleando una flota de drones acuáticos de superficie que recorrerá la mayor superficie del lago posible recabando información en diferentes puntos de esta. De esta manera, no será necesario personal técnico presencial para tomar las muestras ni instalaciones estáticas.

En concreto, el proyecto de colaboración entre la UNA y la US persigue cuatro objetivos fundamentales:

- Construir una red de vehículos no tripulados provistos de sensores que les permitan captar información útil para medir la calidad del agua, y con capacidad de comunicación entre ellos mediante una red inalámbrica que les permita compartir la información capturada.

- Otorgar a la red de vehículos la capacidad de aprendizaje continuo sobre el entorno cambiante, actualizando los modelos a medida que se incorporan nuevos datos.

- Poner en marcha un sistema de telemetría y monitorización en tiempo real de las condiciones del lago que permita determinar cómo influye el entorno sobre las floraciones de cianobacterias en base a los datos capturados.

- Poner el sistema desarrollado a disposición de las entidades locales encargadas de la gestión del agua del lago. La monitorización continua del lago permitirá mejorar la información disponible para la población, la sensibilización sobre el problema y un posible desarrollo normativo que lo solucione o mitigue.

El proyecto en todo su conjunto resulta innovador en cuanto a la medición de contaminación en ríos y lagos, pero también ofrece la posibilidad de ser aplicado a otros problemas ambientales. No solo eso, sino que las investigaciones aquí realizadas serán extrapolables a otras aplicaciones relacionadas con el muestreo mediante vehículos autónomos no tripulados.

En el caso de este proyecto, se está trabajando con dos prototipos de vehículo autónomo de superficie (ASV, *Autonomous Surface Vehicle*): el Cormorán II y el Cormorán III. En el caso del primero, se trata de un vehículo acuático de 4x2 m, provisto de dos sistemas de detección de obstáculos (un LIDAR y un sistema de procesamiento de imágenes basado en redes neuronales convolucionales), y alimentado mediante dos baterías de 20 Ah y paneles fotovoltaicos que le permiten alargar su autonomía. En el caso del Cormorán III, se trata de un vehículo de menores dimensiones (1.3 x 0.95 m), cuyos sensores y electrónica de control son los mismos que se tenían en el Comorán II, aunque se pierden algunas prestaciones, como el uso de placas fotovoltaicas, lo que reduce su autonomía. En caso de que el lector quiera saber más acerca de los ASV desarrollados para este proyecto, puede consultar [3].

Figura 1-3. Prototipos de ASV. A la izquierda, Cormorán II. A la derecha, Cormorán III. Fuente: [3]

1.2 Objetivos

El proyecto de colaboración en el que trabajan la US y la UNA comprende varios módulos de desarrollo, como el diseño de los ASVs que formarán parte de la flota, el diseño de la red de comunicación entre los vehículos, o la planificación del movimiento de estos. Este trabajo se centra en esta última tarea, teniendo como objetivo final obtener un modelo de las variables del lago en el menor tiempo posible empleando la técnica de Optimización Bayesiana.

Como su nombre indica, esta técnica está enfocada a la optimización, es decir, a hallar los puntos donde una determianda función alcanza valores extremos. Sin embargo, la Optimización Bayesiana presenta algunas particularidades que la hacen una opción a considerar para llevar a cabo la monitorización del lago. Ya que el objetivo no es encontrar los puntos del lago donde ciertas variables tomen valores máximos o mínimos, sino conseguir un modelado espacial completo de esas variables, el uso que se hará de esta técnica no será aquel para el que fue concebida a priori.

Por tanto, el propósito general de este trabajo es llevar a cabo una evaluación de la Optimización Bayesiana como herramienta que permita describir el movimiento del ASV consiguiendo información del lago de manera eficiente. Para ello, cobra especial relevancia un estudio completo de la técnica, lo que permitirá su adaptación para ser utilizada con la finalidad alternativa anteriormente descrita.

Más específicamente, los objetivos a cumplir son:

1. Realizar un recorrido por el estado actual del uso de esta técnica, tanto en el campo de la planificación de trayectorias como en el de la optimización, con la finalidad de tener una visión completa de las virtudes y defectos de este método.

2. Estudio de la Optimización Bayesiana, conociendo sus fundamentos y las funciones matemáticas de las que hace uso. Esto permitirá llevar a cabo una correcta adaptación del método para la finalidad buscada.

3. Evaluación del rendimiento de la técnica bajo distintas condiciones y empleando varias configuraciones, de manera que no solo pueda verificarse que el uso de la Optimización Bayesiana es propicio para el proyecto, sino también la configuración bajo la cual tiene un mejor desempeño.

Cabe recalcar que este trabajo se enfoca en el uso de la Optimización Bayesiana para obtener la secuencia de coordenadas en las que los ASVs deberán tomar muestras para obtener el modelo buscado de las variables del lago. En ningún momento se realizará trabajo alguno sobre el algoritmo de desplazamiento de los vehículos hasta dichas coordenadas.

1.3 Motivación

La robótica cumple un papel fundamental en el panorama tecnológico actual. Su asombroso avance en la últimas décadas le ha permitido conquistar el sector industrial, llegando a sustituir la intervención manual en aquellos procesos más rutinarios de los sistemas de producción, provocando un aumento en la productividad, así como en la flexibilidad y seguridad con las que se llevan a cabo las tareas de producción. Sin embargo, el avance de este campo no se detiene aquí. Si bien el mundo conoció las bondades del uso de robots gracias a su aplicación en el sector industrial, no son menos sus aplicaciones fuera de este entorno. En los últimos años, ha aumentado el número de aplicaciones de la robótica en otros campos, tales como la medicina, la enseñanza, el hogar o la supervision y exploración. Y es en este último sector -la supervisión y la exploración- donde recae el motivo del presente trabajo.

Estas dos tareas, en esencia la misma, consisten en llevar a cabo un movimiento por todo un espacio -conocido en el caso de la supervisión, desconocido en el caso de la exploración- con el objetivo de recabar diferentes datos a lo largo de dicho movimiento. Sin embargo, para que estas tareas puedan ejecutarse de una manera eficiente, los robots involucrados deben disponer de cierta autonomía, que les permita percibir el entorno y escoger las acciones adecuadas para desenvolverse dentro de este. Es así como la robótica autónoma cobra la máxima relevancia en este campo.

Atendiendo al tipo de entorno en el que operan se distinguen diferentes tipos de vehículos autónomos: aéreos, de tierra, submarinos o de superficie. En concreto, el presente proyecto trabaja con vehículos de este último tipo, el cual puede encontrarse en aplicaciones como batimetría, monitorización medioambiental o tareas de búsqueda y rescate.

A pesar de la clasificación anterior, los distintos tipos de vehículos autónomos comparten importantes ventajas frente a otros robots o vehículos más convencionales que los convierten en la mejor opción para realizar tareas como la descrita anteriormente. Su reducido tamaño en comparación con otros vehículos clásicos conlleva no solo una reducción en los costes de *hardware*, sino también una mayor facilidad para operar en zonas de difícil acceso. Por otra parte, el hecho de que su uso no dependa de personal humano a bordo permite que trabajen en ambientes de riesgo, como zonas catastróficas. Además, la naturaleza de este tipo de vehículos favorece la formación de flotas, de manera que, ya sea trabajando de forma distribuida o centralizada, el conjunto logre completar la tarea con mayor eficiencia.

En contraparte, una alta autonomía conlleva un incremento en la complejidad del control del vehículo. Esta mayor autonomía pasa por dotar al robot de la capcidad de aprender sobre el entorno en el que se mueve, a menudo cambiante, de manera independiente y con la menor intervención humana posible. La búsqueda de esta autonomía ha llevado a la aplicación de técnicas de Machine Learning sobre estos vehículos. El Machine Learning comprende algoritmos con capacidad de auto-mejora, esto es, las acciones resultantes de estos algoritmos serán mejores conforme aumente el número de ejecuciones; este comportamiento se consigue a base de retroalimentar el algoritmo con los datos obtenidos tras cada ejecución. De esta forma, el robot conseguirá perfeccionar sus acciones reduciendo la necesidad de intervención humana.

Por otra parte, entra en juego otro de los requerimientos clásicos: la optimización de recursos. Más concretamente, en el campo de la supervisión y la exploración se pretende conseguir la mayor cantidad de información posible sobre el entorno recorrido en el menor tiempo posible y asumiendo costes de operación mínimos. Por lo general, ambas restricciones se traducen en hacer que el robot siga una trayectoria en la que recabe los resultados buscados recorriendo la menor distancia posible. Surge entonces la necesidad de implantar algoritmos de optimización que permitan al robot calcular dicha trayectoria óptima por la que llevar a cabo su movimiento.

Este proceso resulta más o menos complicado en función del conocimiento total que se tenga del entorno en el que se trabaja, entendiendo por conocimiento total el conjunto de espacio físico e información que puede obtenerse en dicho espacio. Por ejemplo, en el caso de un dron cuyo trabajo sea recorrer una determinada instalación controlando la temperatura en diferentes puntos de la misma, el proceso para obtener la trayectoria óptima se vuelve relativamente simple: conocida la posición de cada uno de los puntos a visitar pueden conocerse todas las posibles trayectorias que el dron podría recorrer pasando por dichos puntos, ya que estas serán independientes de los datos de temperatura obtenidos, de forma que únicamente habría que calcular qué trayectoria es la que mejor cumple el requerimiento pedido. Es decir, la optimización se convierte en un problema de selección. Sin embargo, la tarea de obtención de un modelo de variables de un lago en base a los datos que que se captan a medida que se recorre el mismo resulta radicalmente distinta: es imposible definir una trayectoria óptima a priori, ya que no puede conocerse la información que facilitará un movimeinto concreto. El hecho de que los datos obtenidos influyan directamente sobre la validez de la trayectoria seguida obliga a emplear un algoritmo que lleve a cabo el proceso de optimización de un modelo desconocido, en este caso, el modelo de las variables del lago que se quieren medir.

Por fortuna, la Optimización Bayesiana responde a estas necesidades. Este método de optimización se engloba dentro de los métodos bayesianos, que tienen su base en el teorema de Bayes y cuya principal utilidad es su capacidad para inferir información desconocida en base a información ya conocida, tal y como lo hacen los algoritmos de Machine Learning. No se necesita, por tanto, un número elevado de evaluaciones, ya que a partir de pocos datos puede inferirse un valor para el resto. Además, no es necesario conocer de antemano la función a optimizar, pues se irá construyendo un modelo de esta a medida que se adquieren nuevos datos. Consecuentemente, la Optimización Bayesiana resulta un método válido para realizar la tarea que este trabajo tiene por finalidad.

2 ESTADO DEL ARTE

Dado que el presente trabajo forma parte del proyecto de colaboración entre la US y la UNA para la monitorización del lago Ypacaraí, resulta interesante no solo realizar un recorrido por el estado actual del estudio y uso de la técnica implementada -en este caso, la Optimización Bayesiana- sino también exponer cuál es el estado actual del proyecto y las distintas soluciones que se han propuesto para resolver el problema hasta el momento.

2.1 Estado del proyecto

La mayor parte de los estudios realizados hasta el momento en torno al problema de monitorización del lago describen este como un problema de cobertura: se considera que una zona del lago está cubierta si el vehículo se desplaza por esa zona y toma muestras en ella. Más allá, el problema de cobertura puede modelarse como un problema de planificación de trayectoria en tanto que el área cubierta será directamente proporcional a la distancia recorrida por el vehículo, salvando la redundancia al pasar este por la misma zona más de una vez.

De hecho, este problema puede ser más concretamente modelado como un TSP (*travelling salesman problem)*, un clásico problema de combinatoria [4]. En su versión más simple, el problema consiste en encontrar el camino óptimo que pasa por *n* puntos y vuelve al punto de partida, siendo las distancias entre los puntos conocidas y simétricas. Si bien resulta simple, su interés radica en que otros muchos problemas más complejos pueden reducirse a un TSP. Siguiendo esta línea, se dispone alrededor de todo el lago una red inalámbrica formada por 60 balizas. La resolución del problema de monitorización pasa por encontrar la secuencia óptima que indique el orden en el que el ASV debe visitar las balizas.

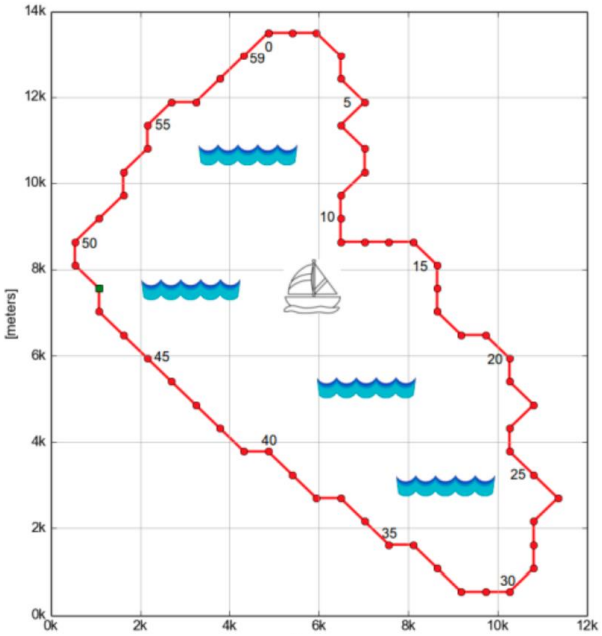

Figura 2-1. Representación de la posición de las balizas sobre el contorno del lago Ypacaraí. Fuente: [10]

Matemáticamente, estos estudios se han servido de la teoría de grafos de manera que la secuencia óptima buscada se corresponde con un circuito que recorra el grafo conformado por las balizas distribuidas por la orilla del lago. Por lo general, el tipo de recorrido buscando se corresponde con los denominados circuitos eulerianos, o bien con circuitos hamiltonianos. Según la teoría de grafos [5], un circuito euleriano es aquel que pasa por todas las aristas del grafo sin repetición, mientras que un circuito hamiltoniano es aquel que pasa por todos los vértices del grafo sin repetición.

En [6] se realiza un análisis del problema empleando tanto circuitos eulerianos como hamiltonianos, pretendiendo encontrar aquel cuya distancia total recorrida resulte menor; como conclusión, se determina que la utilización de los pimeros proporciona un mejor desempeño final. Otros trabajos, como [7] y [8], se centran directamente en el uso de algoritmos genéticos para encontrar el circuito euleriano óptimo, aunque en este caso el criterio de optimización pasa a perseguir la mayor cobertura del lago posible, lo cual se relaciona proporcionalmente con una distancia recorrida mayor.

Los algoritmos genéticos son métodos generalmente usados en problemas de optimización y búsqueda [9]. Tienen su fundamento en las leyes de la evolución biológica: partiendo de un conjunto inicial de posibles soluciones, se obtienen nuevas soluciones aplicando ciertos operadores biológicos, como la mutación, el cruce y la selección. Aquellas soluciones que sean consideradas mejores en base a la restricción del problema de optimización tendrán más posibilidades de participar en los operadores biológicos, produciendo mejores soluciones, hasta finalmente dar una solución óptima.

En [10] se exploran las mismas condiciones anteriores, pero se introduce un procedimiento de dos fases, cada una de ellas centrada en la exploración o en la explotación. De esta forma, el ASV se centra primero en localizar los focos de contaminación (fase de exploración), para después llevar a cabo un monitoreo más exhaustivo de dichas zonas (fase de explotación).

Otros trabajos más recientes innovan más respecto al planteamiento del problema de los anteriores, como es el caso de [11], que hace uso de técnicas de Deep Learning para resolver el problema, a la par que introduce una nueva restricción temporal mediante la cual la importancia de visitar una determinada zona del lago estará influida por la cantidad de tiempo que ha transcurrido desde la última vez que fue visitada.

2.2 Estado de la técnica

Este trabajo escapa de la línea de investigación seguida en el proyecto por los trabajos descritos anteriormente ya que, en lugar de buscar un circuito óptimo, lo que se pretende es obtener un modelado completo del lago lo más certero posible. Para ello, centra su esfuerzo en el estudio de una técnica de optimización: la Optimización Byaesiana.

En [12], [13] y [14] puede encontrarse un desarrollo detallado de toda la teoría que fundamenta esta técnica, así como algunas modificaciones que permiten aplicarla a casos más avanzados, como implementación en paralelo o bajo condiciones de entorno dinámicas. Más novedoso resulta [15] cuya propuesta es la utilización de redes neuranales como alternativa a los procesos gaussianos.

Dado que se trata de un método de optimización, la mayor parte de los trabajos que emplean dicha técnica versan sobre su aplicación a problemas clásicos de optimización; es el caso de [16] o [17] que enfocan la Optimización Bayesiana a tareas de diseño o de configuración de simulaciones. Algunos casos concretos de este tipo de aplicaciones pueden encontrarse en [18], cuya aplicación se lleva a cabo sobre la ciencia de materiales, más concretamente sobre la optimización de predicciones sobre estructuras moleculares orgánicas absorbidas por superficies metálicas; también puede usarse este método para optimizar la parametrización de otros algoritmos de Machine Learning, como se propone en [19]; y en [20], donde se emplea la Optimización Bayesiana para la configuración de parámetros de modelos de animación utilizados para recrear efectos visuales o en creación de videojuegos.

Sin embargo, aunque su aplicación en planificación de trayectorias es menos común, trabajos como [21] y [22] sí que emplean la Optimización Bayesiana en este campo. En [23] el método es utilizado para guiar el movimiento de exploración de un robot en un espacio parcialmente conocido, y cuyo modelado se irá mejorando conforme el robot se desplace por el entorno. En [24] puede encontrarse una aplicación directa a un caso análogo al de estudio, en el que se pretende aplicar la Optimización Bayesiana en la monitorización medioambiental de un entorno determinado mediante la obtención de un modelo espacio-temporal que aproxime lo mejor posible dicho entorno. En la misma línea que el anterior, [25] centra su esfuerzo en llevar a cabo una aproximación genérica para definir funciones de covarianza para el modelado mediante procesos gaussianos de entornos que vayan a ser monitorizados.

3 OPTIMIZACIÓN BAYESIANA

La Optimización Bayesiana es un método de optimización basado en Machine Learning cuya principal aplicación tiene lugar sobre funciones objetivo con alguna de las siguientes características:

- La función objetivo es a priori desconocida, usualmente denominada *black-box function*, lo que conlleva varios inconvenientes. Por una parte, imposibilita conocer las derivadas de la función, impidiendo el uso de métodos de optimización clásicos como el descenso del gradiente o el método de Newton. Por otro lado, limita las suposiciones que pueden hacerse sobre la función y que ayudarían a mejorar la eficiencia del proceso de optimización, como su linealidad o su convexidad. De hecho, al no conocer su convexidad, debe presuponerse la existencia de máximos o mínimos locales, forzando a utilizar un método de optimización global, con la consecuente delimitación del espacio a optimizar.
- La función objetivo es costosa de evaluar en el sentido de que el número de observaciones que se pueden realizar sobre esta no es ilimitado. Esta limitación puede venir dada porque el coste monetario de realizar la observación sea elevado (p.ej.: ensayos destructivos), porque el tiempo que conlleve realizar la observación sea demasiado elevado (p.ej.: simulaciones complejas), o porque las oportunidades de realizar observaciones sean limitadas.
- La función objetivo puede presentar ruido. Si bien es cierto que la Optimización Bayesiana es perfectamente aplicable a casos libres de ruido, esta técnica se presenta como una alternativa robusta en casos donde el ruido sí está presente.

Según esto, el objetivo de la Optimización Bayesiana es encontrar aquellos valores de la función que aporten la mayor cantidad de información posible sobre esta. Esto permitirá que el número de observaciones necesarias sea menor, de forma que se reduce el tiempo y coste de optimización.

Es importante tener en cuenta las condiciones bajo las que se debe hacer uso de esta técnica si se quieren obtener buenos resultados, ya que será en estos casos donde la Optimización Bayesiana alcance su máximo potencial. En comparación con otras técnicas de optimización, como pueden ser los algoritmos genéticos o la optimización por enjambre, la Optimización Bayesiana, por lo general, obtiene peores resultados. Sin embargo, en el caso de que la función objetivo cumpla alguna de las condiciones expuestas anteriormente, dichas técnicas alternativas resultan muy ineficientes o imposibles de aplicar, ya que basan su funcionamiento en evaluar muchas veces la función objetivo, siendo esto en ocasiones muy costoso de realizar.

3.1 Fundamento y algoritmo

Tal como su nombre indica, la Optimización Bayesiana tiene su base en el Teorema de Bayes (Thomas Bayes, 1763), que relaciona las probabilidades de que ocurra un evento aleatorio A condicionado a otro evento aleatorio B y la de que ocurra B condicionado a A. El teorema se define como sigue [26]:

Teorema 2-1 (Teorema de Bayes)

Sea $\{A_1, A_2, \dots, A_i, \dots, A_n\}$ *un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos es distinta de cero. Sea B un suceso cualquiera del que se conocen las probabilidades condicionales* $P(B|A_i)$*. Entonces, la probabilidad* $P(A_i|B)$ *viene dada por la expresión:*

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

donde:

$P(A_i)$ *y* $P(B)$ *son las probabilidades a priori de los sucesos* A_i *y* B*, respectivamente.*

$P(B|A_i)$ *es la probabilidad de B condicionada a* A_i*.*

$P(A_i|B)$ *es la probabilidad a posteriori.*

Aplicado el teorema anterior al caso de un problema de optimización, la expresión anterior toma la forma:

$$P(M|E) \propto P(E|M)P(M) \tag{3-1}$$

donde *M* denota al modelo y *E* a la evidencia, es decir, al conjunto de observaciones realizadas.

Nótese que, en comparación con la definición formal del teorema, ha desaparecido el término del denominador, por lo que deben entenderse las nuevas probabilidades como "probabilidades normalizadas" por dicho valor. Por ello, se pierde la condición de igualdad, pasando a reconocerse como una relación de proporcionalidad; aunque será igualmente válida para el objetivo perseguido. La expresión (3-1) puede interpretarse como:

La probabilidad a posteriori del modelo M dada una evidencia E es proporcional al producto de la probabilidad de obtener dicha evidencia E dado M y la probabilidad a priori del modelo M.

Para llevar a cabo un muestreo eficiente de la función de evaluación aplicando el teorema de Bayes, se sustituye la función objetivo por un modelo más sencillo (modelo subrogado), denominado *prior*. Este modelo otorgará información sobre la probabilidad de los datos, y se irá mejorando en base a las evidencias que se obtengan del modelo real en el proceso de muestreo, entendiéndose como mejora un mayor acercamiento a la función real que se está modelando. De esta forma, se reescribe la expresión anterior como:

$$P(f|D_{1:t}) \propto P(D_{1:t}|f)P(f) \tag{3-2}$$

donde *f* es la función objetivo real y $D_{1:t} = \{\boldsymbol{x}_{1:t}, f(\boldsymbol{x}_{1:t})\}$ es el conjunto de muestras tomadas sobre dicha función.

Como se ha especificado anteriormente, esta técnica se aplica cuando la función objetivo es costosa de evaluar. Por ello, para llevar a cabo la optimización, se utiliza otra función, denominada función de adquisición. Esta función estará basada en el modelo subrogado anterior, pero será más simple, conllevando un coste de evaluación mucho menor en comparación con la función objetivo real.

De esta forma, el proceso de optimización se realizará sobre la función de adquisición en lugar de sobre la función objetivo, y se tomará la siguiente muestra del modelo real en el punto donde la función de adquisición sea óptima.

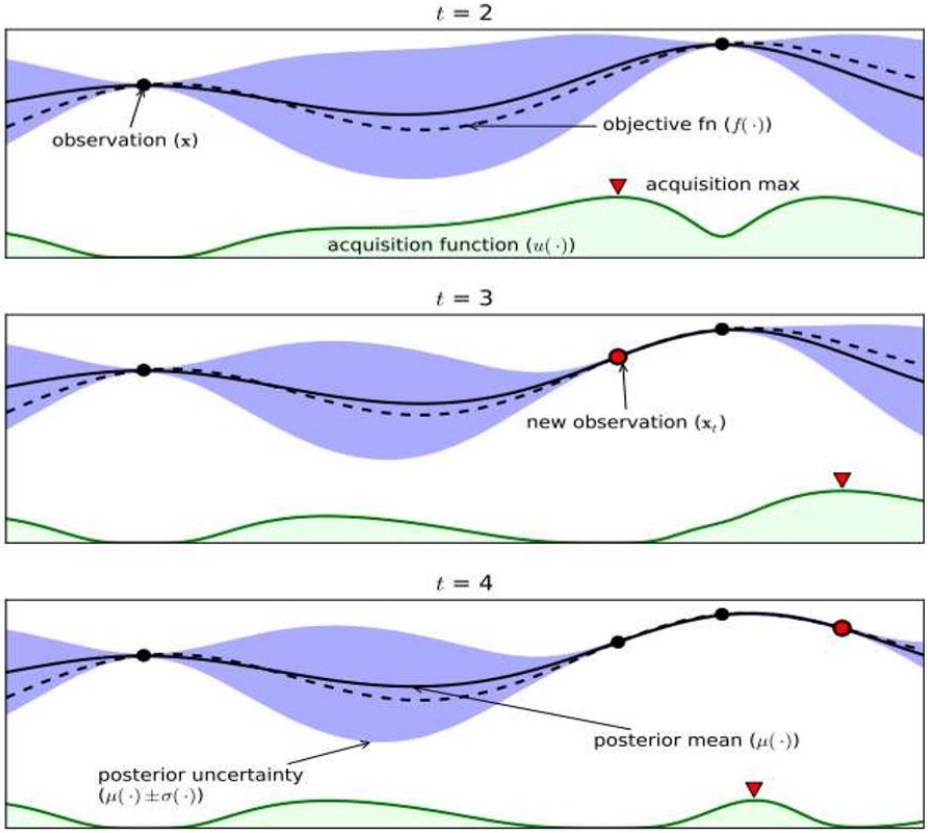

Figura 3-1. Representación gráfica para el caso de una dimensión del funcionamiento de la Optimización Bayesiana. Fuente: [14]

Con lo expuesto hasta ahora, puede desarrollarse el siguiente algoritmo para aplicar la técnica de Optimización Bayesiana:

Algoritmo 2-1 (Optimización Bayesiana)

for $t = 1,2, \ldots$ ***do***

1. *Encontrar el siguiente punto a muestrear* $\boldsymbol{x}_t$ *optimizando la función de adquisición* u*:*

$$\boldsymbol{x}_t = argmax_x\, u(\boldsymbol{x}|D_{1:t-1})$$

2. *Muestrear la función objetivo en dicho punto:* $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$
3. *Actualizar el conjunto de datos* $D_{1:t} = \{D_{1:t-1}; (\boldsymbol{x}_t, y_t)\}$ *con las nuevas muestras.*
4. *Actualizar el modelo con los nuevos datos.*

end for

Puede apreciarse que el algoritmo anterior se divide en dos fases: en primer lugar, se llevan a cabo la optimización y el muestreo y, posteriormente, se realiza la actualización del modelo con los nuevos datos. De esta forma, al final de cada iteración, se tiene el modelo actualizado. A estas fases se les denomina, respectivamente, fase de decisión y fase de aprendizaje.

Empleando este algoritmo, en cada iteración el modelo subrogado irá mejorando, por lo que los valores óptimos obtenidos al optimizar la función de adquisición serán más parecidos a los valores óptimos de la función real. Obviamente, los resultados serán buenos en tanto que el modelo subrogado que se este utilizando sea adecuado para el problema. Al utilizar el modelo para guiar el proceso de optimización, la Optimización Bayesiana obtiene mejores resultados que otros métodos, como la búsqueda en rejilla o la búsqueda aleatoria.

A modo de resumen, se han presentado dos herramientas clave en la implementación de la Optimización Bayesiana: el *prior*, que generalmente será modelado mediante un proceso gaussiano, y la función de adquisición; la obtención de buenos resultados pasará por una correcta elección de ambas herramientas. Adicionalmente, es típico encontrar hiperparámetros en la definición del *prior* por lo que será conveniente llevar a cabo un proceso previo de optimización para encontrar los valores óptimos de dichos hiperparámetros.

3.2 Procesos Gaussianos

Como se introdujo anteriormente, la función objetivo real se sustituirá por un modelo más sencillo. Esta simplificación conlleva que la obtención de buenos resultados estará condicionada a la fidelidad con la que dicho modelo represente el comportamiento de la función real. Esta fidelidad sería mayor en tanto mayor fuera el número de propiedades asumibles de la función real, con el inconveniente de que esta puede ser desconocida.

De cara a afrontar este problema de modelado, se dan dos puntos de vista comunes. Un primer acercamiento pasaría por restringir el tipo de funciones que se debe considerar (p.ej.: funciones lineales), descartando todos los demás casos. Esta metodología tiene un claro problema: en caso de escoger un tipo de función que no se asemeje a la función subyacente, las predicciones realizadas serán erróneas. El segundo acercamiento pretendería mitigar este inconveniente, siendo mucho más flexible: en lugar de restringir el tipo de función, se otorgaría una probabilidad a cada función, siendo esta mayor cuanto más semejante a la función subyacente se considere (p.ej.: otorgar mayor probabilidad a funciones con mayor suavidad). Sin embargo, esta solución también presenta un serio problema: cómo tener en consideración las infinitas funciones posibles y, más allá, cómo computarlas en un tiempo finito.

A efectos de salvar estos inconvenientes, los procesos gaussianos constituyen una herramienta perfecta. Un proceso gaussiano es una extensión multivariante de la distribución gaussiana, definiéndose como una función de probabilidad sobre funciones aleatorias. Es decir, el proceso gaussiano puede interpretarse como una función que devuelve en cada punto una media y una varianza de una distribución normal, adquiriendo una gran flexibilidad a la hora de ajustarse a cualquier tipo de función. También solventa el problema de la tratabilidad computacional, ya que no es necesario operar con los infinitos puntos de la función para observar las propiedades de un determinado grupo de ellos.

Además, al no tratarse de un modelo paramétrico, no hay que preocuparse por si el modelo puede ajustarse o no a los nuevos datos; aunque se añadan más datos al modelo, este será suficientemente flexible para adaptarse a ellos.

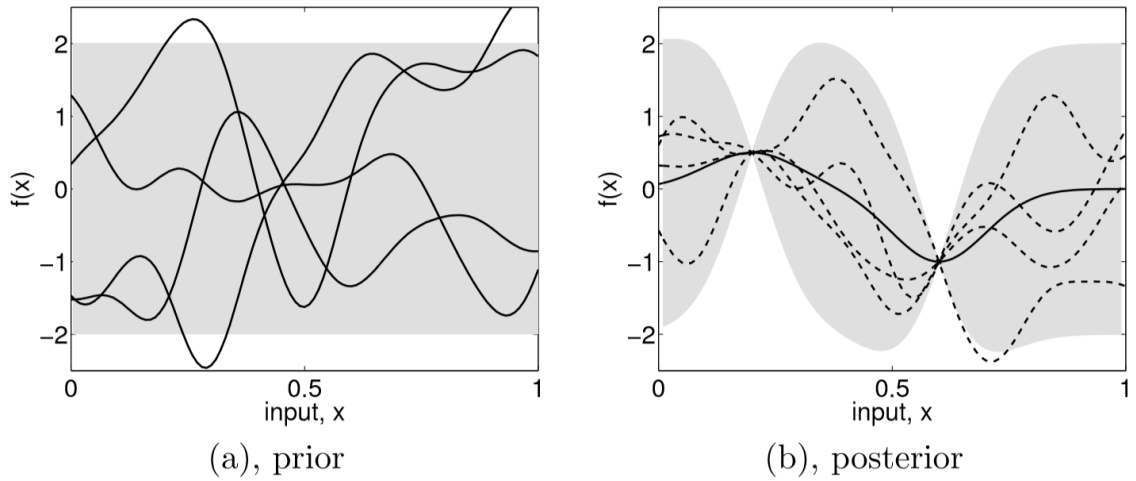

Figura 3-2. Modelado mediante procesos gaussianos. Fuente: [32]

En la Figura 2-2a puede verse cómo ante la falta de datos sobre la función real son muchas las funciones que pueden corresponderse con esta. Sin embargo, en la Figura 2-2b, tras el muestreo de algunos puntos de la función real, el número de funciones que pueden modelarla será menor, ya que solo serán válidas aquellas funciones que se ajusten a los puntos observados.

Matemáticamente, al igual que la distribución gaussiana o normal queda definida en base a su media y su desviación típica tal y como indica la expresión (3-3), un proceso gaussiano se define en base a una función de media $m(\boldsymbol{x})$ y una función de covarianza $k(\boldsymbol{x}, \boldsymbol{x}')$, como indica la expresión (3-4).

$$N(x{:}\,\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right] \tag{3-3}$$

$$f(x) \sim GP(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')) \tag{3-4}$$

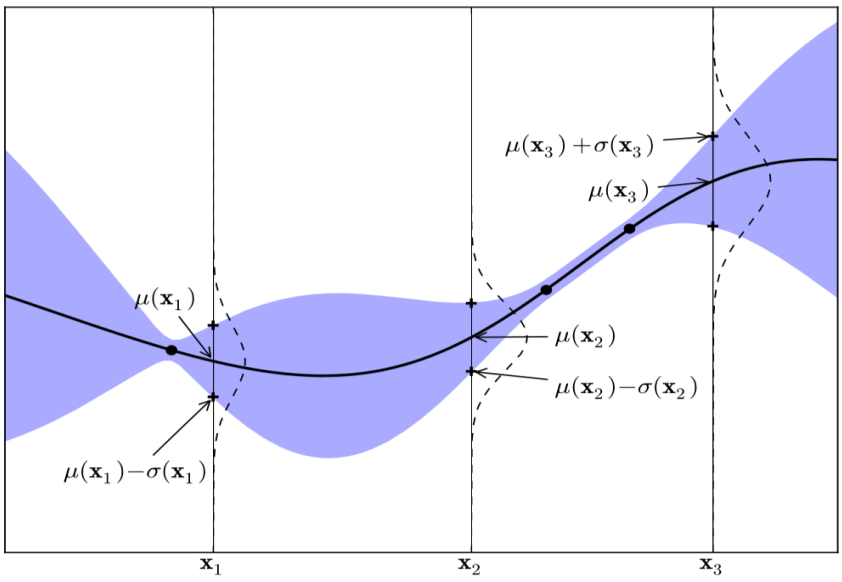

Figura 3-3. Representación de proceso gaussiano. En cada punto, el proceso gaussiano devuelve los valores de media y desviación típica que definen la distribución normal que siguen los valores de la función en dicho punto. Fuente: [14]

Por lo general, la función de media se considera nula, por lo que el proceso gaussiano queda enteramente definido a partir de la elección de la función de covarianza, comúnmente denominada *kernel*. Será el *kernel* el que determine las propiedades del proceso gaussiano, pudiendo controlar a partir de dicha función cómo variará el prior a partir de los puntos muestreados.

De esta forma, los valores de un proceso gaussiano con estas características seguirán una distribución normal multivariante:

$$\mathcal{N}(0, \boldsymbol{K}), siendo\ \boldsymbol{K} = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_t) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_t, \boldsymbol{x}_1) & \ldots & k(\boldsymbol{x}_t, \boldsymbol{x}_t) \end{bmatrix} \tag{3-5}$$

donde debe tenerse en cuanta que $k(\boldsymbol{x}_i, \boldsymbol{x}_i) = 1$.

Típicamente, los *kernels* tienen la propiedad de que puntos cercanos en el espacio de entrada estarán más correlacionados, es decir:

$$Si\ \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| < \|\boldsymbol{x}_1 - \boldsymbol{x}_3\|, \qquad entonces\ k(\boldsymbol{x}_1, \boldsymbol{x}_2) > k(\boldsymbol{x}_1, \boldsymbol{x}_2)$$

Dado que el *prior* debe permanecer actualizado, debe obtenerse una expresión que permita definir el proceso gaussiano en base al proceso gaussiano de un estado anterior y los nuevos datos obtenidos en cada iteración. Partiendo de la expresión (3-5), el proceso gaussiano puede actulizarse según:

$$\begin{bmatrix} \boldsymbol{f}_t \\ f_{t+1} \end{bmatrix} = \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{k} \\ \boldsymbol{k^T} & k(x_{t+1}, x_{t+1}) \end{bmatrix}\right) \tag{3-6}$$

donde la matriz **K** ya se definió anteriormente y el vector **k** se construye como:

$$\boldsymbol{k} = [k(x_{t+1}, x_1) \quad \ldots \quad k(x_{t+1}, x_t)]$$

Bajo estas definiciones, se tienen las siguientes expresiones mediante las cuales se puede obtener la distribución del siguiente punto a muestrear:

$$P(f_{t+1}|D_{1:t}, \boldsymbol{x}_{t+1}) = \mathcal{N}\left(\mu_t(\boldsymbol{x}_{t+1}), \sigma_t^2(\boldsymbol{x}_{t+1})\right) \begin{cases} \mu_t(\boldsymbol{x}_{t+1}) = \boldsymbol{k}^T \boldsymbol{K}^{-1} \boldsymbol{f}_{1:t} \\ \sigma_t^2(\boldsymbol{x}_{t+1}) = k(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+1}) - \boldsymbol{k}^T \boldsymbol{K}^{-1} \boldsymbol{k} \end{cases} \tag{3-7}$$

En la expresión anterior, resulta destacable que el cálculo queda completamente definido en base a la función de covarianza empleada, por lo que la elección de esta resulta clave para definir el proceso gaussiano. A continuación, se presentan dos de los *kernels* más utilizados.

Radial-basis function kernel (RBF)

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\left(-\frac{1}{2l^2}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right) \qquad (l > 0) \tag{3-8}$$

El *kernel* RBF, también conocido como *squared exponential kernel*, es probablemente el kernel más usado en este campo, pese a que su alta suavidad resulta poco realista para modelar ciertos fenómenos. Esta función de covarianza queda parametrizada por l, denominado *length-scale*, que controla la flexibilidad del *kernel* a la hora de adaptarse a los datos: a menor l, mayor flexibilidad.

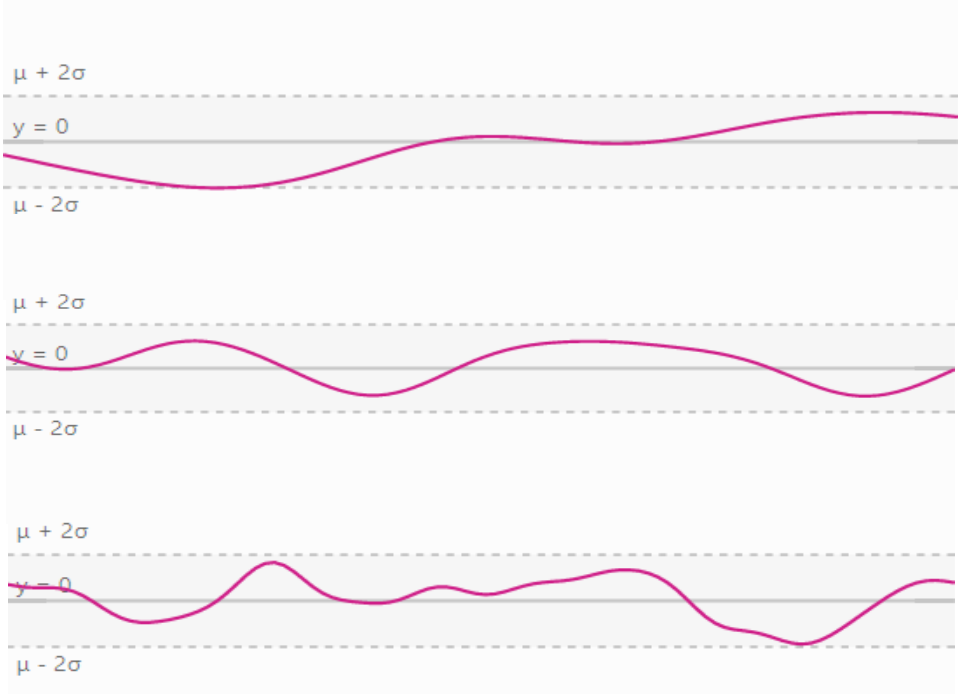

Figura 3-4. Variación del kernel RBF en función del valor del *lenght-scale.* Arriba *l=1.5*, centro *l=1*, abajo *l=0.5*. *Fuente: https://distill.pub/2019/visual-exploration-gaussian-processes/*

Màtern kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{2^{\nu-1}\Gamma(\nu)}\left(\frac{\sqrt{2\nu}}{l}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{l}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\right) \tag{3-9}$$

El grupo de los *Màtern kernels* consiste en una generalización del kernel RBF descrito anteriormente. En la expresión (3-9), $\Gamma(\cdot)$ es la función gamma y $K_\nu(\cdot)$ es una modificación de la función de Bessel. Por otra parte, este kernel cuenta con un parámetro adicional ν que permite controlar la suavidad de la función de covarianza resultante, siendo esta menos suave para valores de ν más pequeños.

De hecho, cuando $\nu \to \infty$ el kernel resultante es equivalente al kernel RBF, y cuando $\nu = 0.5$ se convierte en el kernel exponencial absoluto. Además, existen otros valores típicos, como $\nu = 1.5$ y $\nu = 2.5$, para los que el costo computacional se ve reducido debido a que no es necesario reevaluar la función de Bessel para dichos valores.

Los kernels anteriores, así como tantos otros que no han sido expuestos en este trabajo, cuentan con hiperparámetros (como l) cuyo valor deberá escogerse adecuadamente para un buen desempeño del algoritmo. Si bien siempre existe la posibilidad de escoger estos valores manualmente, también pueden aplicarse otros métodos que permitan determinar los valores óptimos para estos hiperparámetros y obtener así resultados mejores.

Una característica excepcional de los procesos gaussianos es que, a diferencia de otros modelos de Machine Learning, cuentan con una expresión analítica computacionalmente tratable para la integral del modelo sobre los parámetros, lo que permite el cálculo del valor óptimo del hiperparámetro que permitirá el mejor ajuste del modelo.

Esta expresión de denomina verosimilitud marginal, y viene dada por:

$$p(\boldsymbol{y}|X) = \int p(\boldsymbol{y}|\boldsymbol{f}, X)p(\boldsymbol{f}|X)d\boldsymbol{f} \tag{3-10}$$

En su forma logarítmica:

$$\log p(\boldsymbol{y}|X) = -\frac{1}{2}\boldsymbol{y}^{\boldsymbol{T}}(K + \sigma_n^2 I)^{-1}\boldsymbol{y} - \frac{1}{2}log|K + \sigma_n^2 I| - \frac{n}{2}\log(2\pi) \tag{3-11}$$

Teniendo en cuenta que el kernel varía en base al hiperparámetro θ, puede escribirse la siguiente expresión que permite llevar a cabo la optimización para obtener el valor óptimo del hiperparámetro:

$$\frac{\partial}{\partial\theta}\log p(\boldsymbol{y}|X, \theta) = \frac{1}{2}\boldsymbol{y}^{\boldsymbol{T}}K^{-1}\frac{\partial K}{\partial\theta}K^{-1}\boldsymbol{y} - \frac{1}{2}tr\left(K^{-1}\frac{\partial K}{\partial\theta}\right) \tag{3-12}$$

3.3 Funciones de adquisición

Una vez se tiene el *prior* que modela la función objetivo real, se debe seleccionar una función de adquisición adecuada. Como se dijo anteriormente, será esta la función sobre la que se aplique el método de optimización, para hallar así el siguiente punto a muestrear. En efecto, la función de adquisición debe estar estrechamente relacionada con el *prior*, de manera que el punto óptimo encontrado se corresponda con un punto de alto interés en la función objetivo, ya sea por encontrarse cerca del óptimo real o por encontrarse en zonas inexploradas de la función objetivo. Es en este punto donde entra en juego el dilema *exploración vs explotación.*

Se denomina exploración a la obtención de nuevas muestras en zonas de alta incertidumbre, es decir, en puntos no cercanos a muestras anteriores. Por su parte, se denomina explotación a la obtención de muestras en zonas donde se conoce que la función objetivo alcanza valores altos, es decir, en puntos cercanos a los puntos óptimos. Será la función de adquisición la que determine el tipo de tendencia que seguirá el algoritmo, por lo que la correcta elección de esta resulta de gran importancia para la obtención de los resultados perseguidos.

Con todo esto, la finalidad de la función de adquisición es doble: por una parte, debe permitir discernir cuáles son los mejores puntos a muestrear en el siguiente paso que permitan la mayor mejora del modelo y, por otro lado, debe estar definida por una expresión sencilla en cuanto a complejidad de cálculo, lo que permitirá el uso de métodos de optimización clásicos sobre las mismas. A continuación, se presentan algunas de las funciones de adquisición más utilizadas.

Probability of Improvement (PI)

Esta función mide la probabilidad de que la siguiente muestra sea mejor que el mejor valor obtenido hasta el momento.

$$PI(\boldsymbol{x}) = P\big(f(\boldsymbol{x}) \geq f(\boldsymbol{x}^+)\big) = \Phi\left(\frac{\mu(\boldsymbol{x}) - f(\boldsymbol{x}^+)}{\sigma(\boldsymbol{x})}\right), con\ \boldsymbol{x}^+ = argmax_{x_i \in x_{1:t}} f(\boldsymbol{x}_i) \qquad (3\text{-}13)$$

donde $\boldsymbol{x}^+$ denota el punto donde se ha obtenido la mejor muestra hasta el momento, y $\Phi(\cdot)$ es la función de distribución acumulada normal.

El mayor inconveniente de esta función es que tiende puramente a la explotación, es decir, una vez se encuentra un punto con un buen resultado, las siguientes muestras se tomarán en las cercanías de ese punto.

En el caso de que se quiera que la mejora buscada supere un cierto umbral, se puede añadir un hiperparámetro que haga las veces de valor umbral:

$$PI(\boldsymbol{x}) = P(f(\boldsymbol{x}) \geq f(\boldsymbol{x}^+) + \xi) = \Phi\left(\frac{\mu(\boldsymbol{x}) - f(\boldsymbol{x}^+) - \xi}{\sigma(\boldsymbol{x})}\right) \qquad (3\text{-}14)$$

Expected Improvement (EI)

En este caso, en lugar de utilizar la probabilidad de mejora, se utiliza la magnitud de la mejora esperada.

$$EI(\boldsymbol{x}) = \mathrm{E}\, max(f(\boldsymbol{x}) - f(\boldsymbol{x}^+), 0) \qquad (3\text{-}15)$$

Analíticamente, la mejora esperada puede ser evaluada como:

$$EI(\boldsymbol{x}) = \begin{cases} (\mu(\boldsymbol{x}) - f(\boldsymbol{x}^+) - \xi)\Phi(\mathrm{Z}) + \sigma(\boldsymbol{x})\phi(Z) \;\; si\ \sigma(\boldsymbol{x}) > 0 \\ 0 \;\; si\ \sigma(\boldsymbol{x}) = 0 \end{cases} \qquad (3\text{-}16)$$

Donde $\Phi(\cdot)$ y $\phi(\cdot)$ son la función de distribución acumulada normal (CDF, *Cumulative Distribution Function*) y la función de distribución de probabilidad normal (PDF, *Probability Distribution Function*), respectivamente. Por su parte, la distribución Z queda definida como:

$$Z = \begin{cases} \dfrac{\mu(\boldsymbol{x}) - f(\boldsymbol{x}^+) - \xi}{\sigma(\boldsymbol{x})} \;\; si\ \sigma(\boldsymbol{x}) > 0 \\ 0 \;\; si\ \sigma(\boldsymbol{x}) = 0 \end{cases}$$

En la expresión de la mejora esperada anterior pueden diferenciarse dos términos sumados. El primero se denomina término de explotación, y el segundo término de exploración. De hecho, será el valor del hiperparámetro ξ el que determine la proporción de cada una de estas tendencias.

Confidence Bound Criteria (UCB (Upper Confidence Bound) or LCB (Lower Confidence Bound))

En este caso se persiguen aquellos puntos que pudieran tener los valores más altos.

$$UCB(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \kappa\sigma(\boldsymbol{x}), con\ \kappa \geq 0 \tag{3-17}$$

En el caso de escoger un valor $\kappa \leq 0$, se estaría utilizando el LCB, equivalente a UCB para el caso de que se quiera minimizar la función objetivo.

4 CASO REAL. RESULTADOS

Como es sabido, la Optimización Bayesiana tiene por objetivo natural encontrar el valor óptimo de la función objetivo que se está analizando; si bien es cierto que durante el proceso puede obtenerse un modelo que se aproxime a la función objetivo desconocida, esta no es su finalidad principal. Sin embargo, para el caso de estudio, el objetivo primordial sí que será obtener un modelo que represente correctamente las variables del lago Ypacaraí, relegando encontrar los valores máximos o mínimos de dichas variables a un segundo plano. Para ello se deberán realizar las adaptaciones y ajustes necesarios del algoritmo presentado anteriormente de forma que su uso pueda realizarse con tal fin.

En esta sección se presentan los resultados de una serie de simulaciones experimentales cuyo fin es poner a prueba el funcionamiento del algoritmo bajo diferentes configuraciones. En primer lugar, se hace una descripción de los diferentes elementos involucrados en estos experimentos: hipótesis de trabajo sobre las que se llevan a cabo, datos con los que se trabaja, los criterios con los que medir la validez del modelo obtenido frente al modelo objetivo real con el que se estaba trabajando, y el código implementado en Python que aúna todos los elementos anteriores. Seguidamente se exponen los resultados obtenidos para cada una de las distintas configuraciones consideradas para, en último lugar, realizar una comparación entre todas ellas.

4.1 Planteamiento de los experimentos

Hipótesis de trabajo

Los valores de las variables del lago que quieren analizarse seguirán una distribución superficial, es decir, la función objetivo será una función bidimiensional cuyas dimensiones se correspondan directamente con las coordenadas de cada punto de la superficie del lago. Formalmente, esto no conlleva ningún cambio respecto al desarrollo teórico de la sección 3. Por otro lado, se buscará un modelo que represente la distribución de una única variable medible.

También es necesario hacer un apunte sobre el papel que juega el vehículo acuático dentro del funcionamiento del algoritmo: el vehículo únicamente tomará muestras en los puntos de destino indicados por el algoritmo de optimización, es decir, no tomará muestras durante el desplazamiento entre un punto y el siguiente. Además, se considera el caso de un único vehículo operando, por lo que no se tendrán en cuenta metodologías de coordinación de flotas.

Resumidamente, las condiciones de los experimentos que se han llevado a cabo pueden condensarse en:

Aplicación del método de Optimización Bayesiana sobre una función $v = f(\boldsymbol{x}) = f(x_1, x_2)$, siendo v la variable del lago a modelar y el par (x_1, x_2) las coordenadas espaciales sobre la superficie del lago, obteniendo una nueva muestra en cada iteración.

Datos de trabajo

Dada la imposibilidad de realizar experimentos reales sobre la superficie de un lago para posteriormente comprobar la adecuación del modelo obtenido respecto al modelo real del lago, se realizará esta misma tarea empleando *test functions*. En este caso, se han seleccionado diferentes funciones de dos dimensiones con el objetivo de evaluar el desempeño del algoritmo frente a modelos con diferentes características.

Las tres funciones con las que se han realizado los experimentos han sido la función Branin, la función Schwefel de dimensión 2 y la función Styblinski-Tang de dimensión 2, cuyas representaciones matemáticas y gráficas se presentan a continuación [27].

- Función Branin

$$f(\boldsymbol{x}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10 \tag{4-1}$$

Evaluada en el cuadrado $x_1 \in [-5, 10], x_2 \in [0, 15]$

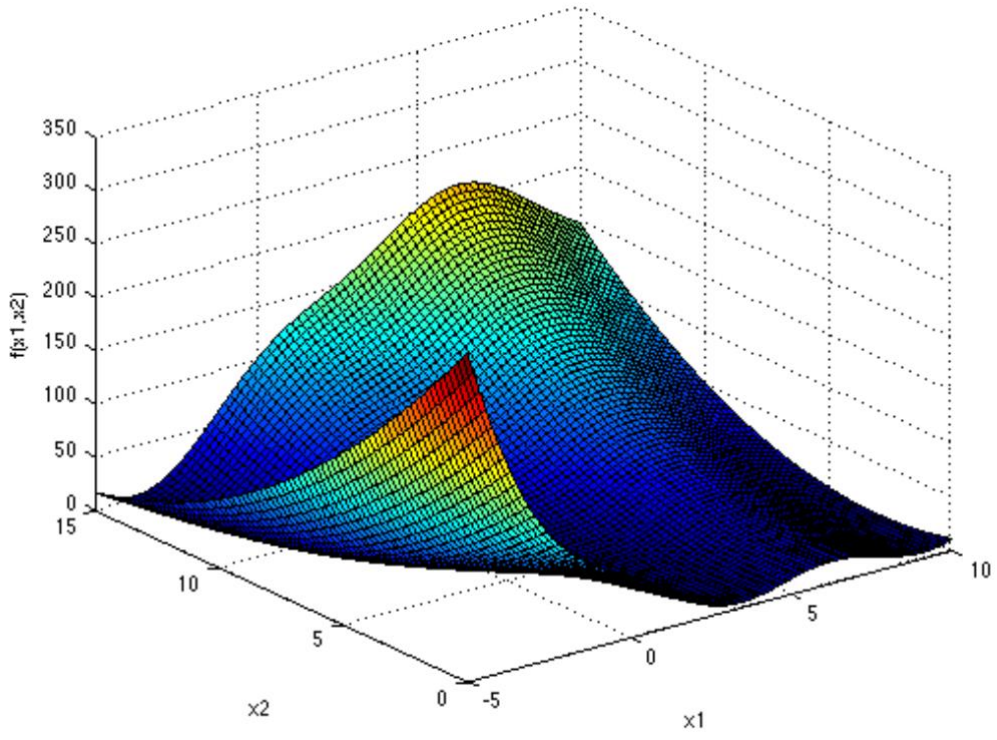

Figura 4-1. Representación tridimensional de la función de Branin. Fuente: [27]

- Función Schwefel (d = 2)

$$f(\boldsymbol{x}) = 418.9892 \cdot 2 - \left(x_1 sen\left(\sqrt{|x_1|}\right) + x_2 sen\left(\sqrt{|x_2|}\right)\right) \tag{4-2}$$

Evaluada en el cuadrado $x_1 \in [-500, 500], x_2 \in [-500, 500]$

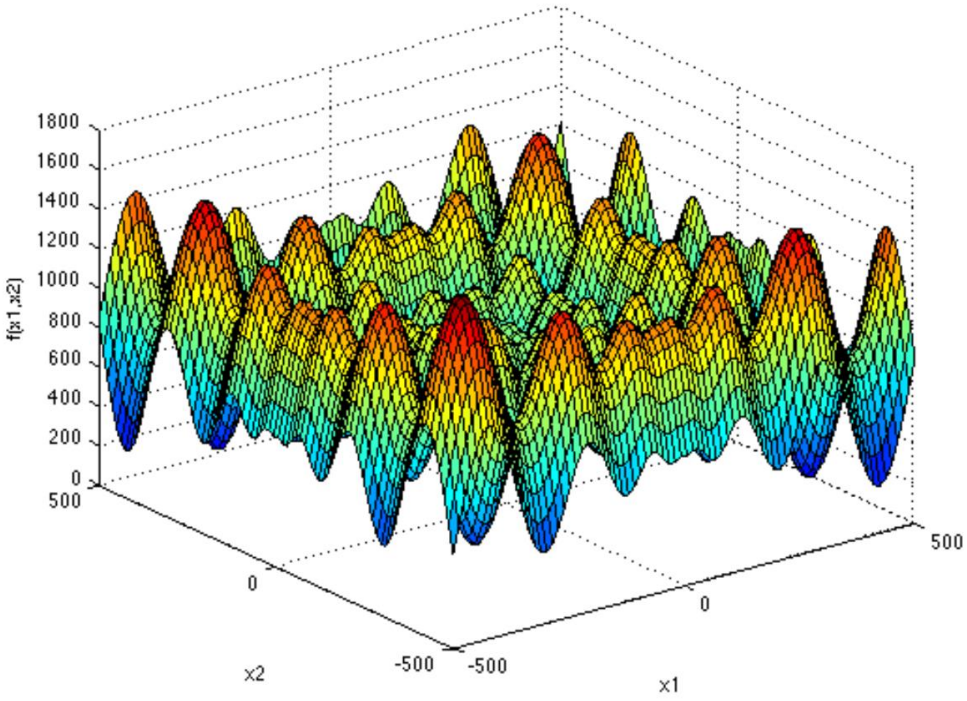

Figura 4-2. Representación tridimensional de la función de Schwefel (para *d=2*). Fuente: [27]

- Función Styblinski-Tang (d = 2)

$$f(\boldsymbol{x}) = \frac{1}{2}\left((x_1^4 - 16x_1^2 + 5x_1) + (x_2^4 - 16x_2^2 + 5x_2)\right) \tag{4-3}$$

Evaluada en el cuadrado $x_1 \in [-5, 5], x_2 \in [-5, 5]$

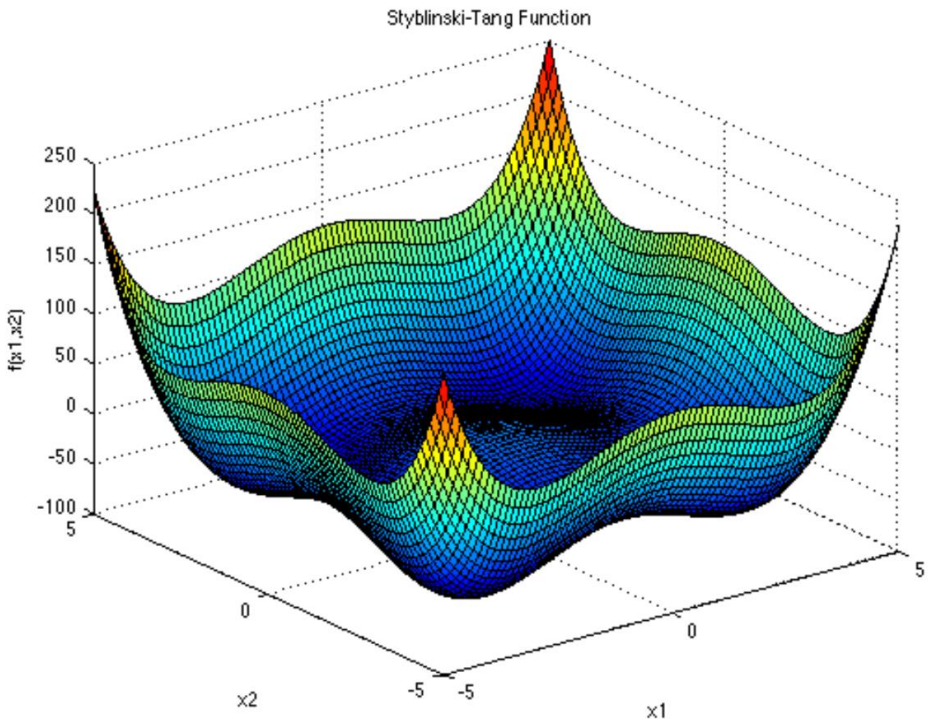

Figura 4-3. Representación tridimensional de la función de Styblinski-Tang (para *d=2*). Fuente: [27]

Indicadores de validez

Como resultado de la aplicación del algoritmo de Optimización Bayesiana sobre las funciones anteriores se obtiene un modelo de dicha función en base a un proceso gaussiano. Por tanto, es necesario definir unos indicadores con la finalidad de cuantificar la fiabilidad de los modelos obtenidos. Son dos los aspectos a considerar: la exactitud y la precisión del modelo.

La exactitud hace referencia a la proximidad entre los valores del modelo y los valores verdaderos. Para ello, se modela una *función de error* que en cada punto toma el valor del error absoluto cometido al estimar la función objetivo real (en este caso, la *test function*) mediante la función de media del proceso gaussiano que representa el modelo obtenido mediante el uso del algoritmo. Además, dicha función permitirá graficar un mapa de error con el que comprobar qué puntos presentan más error.

$$f_{error}(\boldsymbol{x}) = \left| f_{obj}\ (\boldsymbol{x}) - \mu_{modelo}\ (\boldsymbol{x}) \right| \tag{4-4}$$

A partir de dicha función, se define el *error cuadrático medio* como:

$$e_{cm} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} f_{error}^2(\boldsymbol{x_i})} \tag{4-5}$$

Siendo N el número total de puntos en el que se ha dividido el espacio evaluable de la función objetivo.

Por otra parte, la precisión hace referencia a la repetibilidad de obtener una determinada medida gracias al modelo. La cuantificación de la precisión puede modelarse mediante la función de desviación típica del proceso gaussiano que representa el modelo obtenido mediante el uso del algoritmo. De igual forma que en el caso anterior, esta función permitirá graficar un mapa de precisión, así como obtener un indicador *variación cuadrática media* definido como:

$$\sigma_{cm} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \sigma_{modelo}^2(\boldsymbol{x_i})} \tag{4-7}$$

Siendo N el número total de puntos en el que se ha dividido el espacio evaluable de la función objetivo.

4.2 Código de simulación

En este apartado se detalla el código desarrollado en Python para llevar a cabo los distintos experimentos. En líneas generales, se ha construido una función denominada "experimento" que engloba todo el código, de manera que para cada ejecución únicamente es necesario realizar una llamada a la función con la serie de parámetros que definirán la configuración de la simulación que se quiere realizar.

Para ello se ha hecho uso de varias librerías fundamentales: Numpy para el cálculo matemático, Scikit-Learn para la implementación del proceso gaussiano y los *kernels*, Scipy para la implementación de los algoritmos de optimización, Matplotlib para graficar los mapas de error y precisión, y Pandas para la exportación de todos los datos obtenidos a un archivo Excel.

La siguiente imagen representa el diagrama de flujo que sigue la ejecución del código implementado:

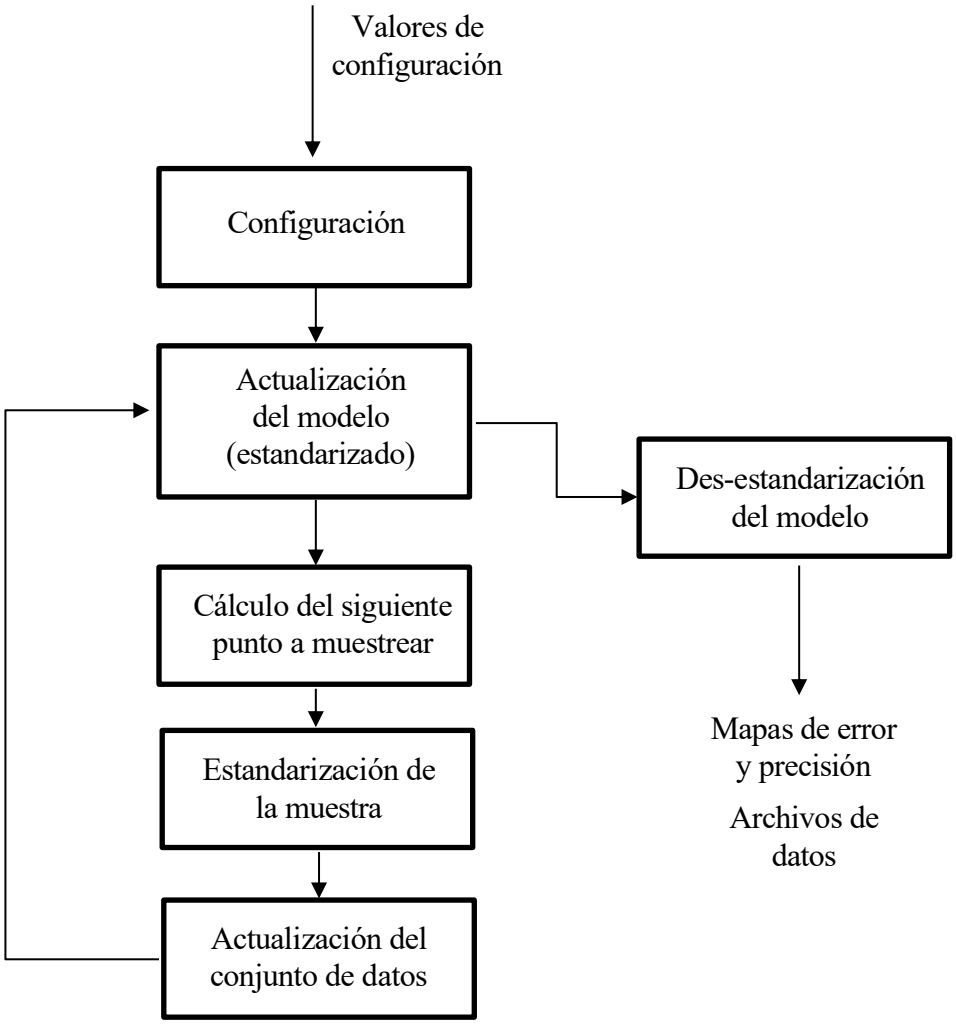

Figura 4-4. Diagrama de flujo del código implementado.

La ejecución de cada experimento se divide en dos etapas. En la primera se lleva a cabo la configuración del mismo en base a los valores con los que se ha llamado a la función; de esta forma se definen la función de test, la función de adquisición, el *kernel* y el valor del parámetro de *trade-off* que se van a utilizar, así como las funciones de error y de precisión, el punto inicial y los límites de definición de la función de test. En la segunda etapa se lleva a cabo la implementación de la Optimización Bayesiana tal y como se indica en la Figura 4-4. En esta segunda etapa destacan dos aspectos importantes.

El primero de ellos hace referencia a la estandarización de los datos obtenidos durante el muestreo. Este procedimiento es un requerimiento común de muchos algoritmos de machine learning, que basan su funcionamiento en que los datos sigan una distribución normal estándar, es decir, una distribución gaussiana con media cero y varianza unitaria. Esto no es más que realizar la siguiente operación sobre el valor de la muestra:

$$y_{standard} = \frac{y_{muestra} - \mu}{s}$$

donde μ y s son la media y la desviación estándar del conjunto de datos.

Dado que el modelo se irá formando en base a los valores estandarizados, será necesario llevar a cabo la transformación inversa para calcular los indicadores de validez, de forma que dicho cálculo se haga con los valores reales equivalentes y no con los valores estandarizados.

En segundo lugar, a la hora de definir el proceso gaussiano, se ha incluido un parámetro denominado *alpha* que modela el nivel de ruido en las muestras; matemáticamente, se trata de un valor constante añadido a la diagonal del *kernel*. Como es lógico, a mayor nivel de ruido, peor serán los resultados obtenidos. Sin embargo, ha sido necesario utilizar dicho parámetro por una cuestión de robustez computacional.

En la expression (3-7) se vio que para actualizar los valores de media y varianza del proceso gaussiano es necesario llevar a cabo la inversión de la matriz $\boldsymbol{K}$, proceso que puede dar lugar a inestabilidad numérica. Esta inestabilidad resulta mayor cuanto mayor sea el número de condición de la matriz, definido como la relación entre el mayor y el menor de sus autovalores. Teniendo en cuenta que los valores extremos de dichos autovalores aparecen cuando los valores de la función están muy correlacionados, resulta apropiado añadir el citado parámetro para prevenir problemas de inestabilidad numérica. Esto da lugar a un nuevo *trade-off* entre la robustez computacional y la precisión del modelo obtenido.

Tanto el Código implementado en Python como los resultados presentados posteriormente pueden encontrarse en el repositorio de *GitHub* https://github.com/pjdg/TFG-Opt.-Bayesiana.

4.3 Resultados

En esta sección se exponen los resultados de los experimentos realizados bajo las condiciones descritas anteriormente. Empleando las tres funciones de test anteriores y los kernels y funciones de adquisición que se presentaron en la sección 3 se han definido un total de 18 combinaciones posibles para las que obtener los indicadores de validez anteriores. Adicionalmente, cada una de estas configuraciones se ha ejecutado para varios valores del parámetro que controla el *trade-off* exploración/explotación en cada una de las funciones de adquisición. Más concretamente, el valor del parámetro de *trade-off* toma los valores 0.01, 0.1 y 0.2.

Por otra parte, para los valores de los hiperparámetros que definen los kernels empleados se ha escogido $\nu = 1.5$ y un valor de l igual al 10% de rango del mapa sobre el que se estudia la *test function*. Cabe destacar que en el caso de l este únicamente será su valor inicial, ya que su valor será optimizado en cada iteración para mejorar el ajuste del kernel.

Por otro lado, con el objetivo de obtener una estadística de los resultados obtenidos, se han realizado 30 réplicas de cada simulación. Esto se debe a que el algoritmo de optimización utilizado para optimizar la función de adquisición y obtener el siguiente punto a muestrear tiene una componente aleatoria, por lo que el conjunto de puntos muesteados varía de una ejecución a otra, y con él el valor de los indicadores finales.

Para cada una de las diferentes configuraciones se ha construido una tabla resumen con los valores de error y desviación evaluados cada 10 iteraciones hasta un total de 50, siendo estos valores la media de los datos correspondientes a cada una de las réplicas simuladas. Además, la distribución del conjunto de estos datos podrá verse en las gráficas *de caja y bigotes* que acompañan a cada una de las tablas.

También se expone un conjunto de gráficas que muestran la evolución de los mapas de error y de desviación conforme aumenta el número de iteraciones, así como los puntos que han sido muestreados sobre dichos mapas. Carece de sentido presenter el evolutivo para cada una de las réplicas simuladas, por lo que para cada experimento se muestra un único conjunto de mapas que ha sido escogido bajo criterio de mejor ajuste a la distribución de datos de dicho experimento.

Experimento 1: Branin - RBF - Probability of Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	44,07955	146,2179
	20	27,76165	91,21267
	30	22,93526	58,90926
	40	18,78967	45,89285
	50	16,07038	34,56775
0.1	10	34,55251	109,9385
	20	19,92341	45,85731
	30	10,27441	6,427224
	40	8,155726	4,993284
	50	7,134203	4,230362
0.2	10	37,88151	74,97493
	20	16,95405	20,14319
	30	9,648402	5,946918
	40	8,156001	4,543914
	50	6,351028	4,525501

Tabla 4-1. Resumen de resultados para el experimento 1

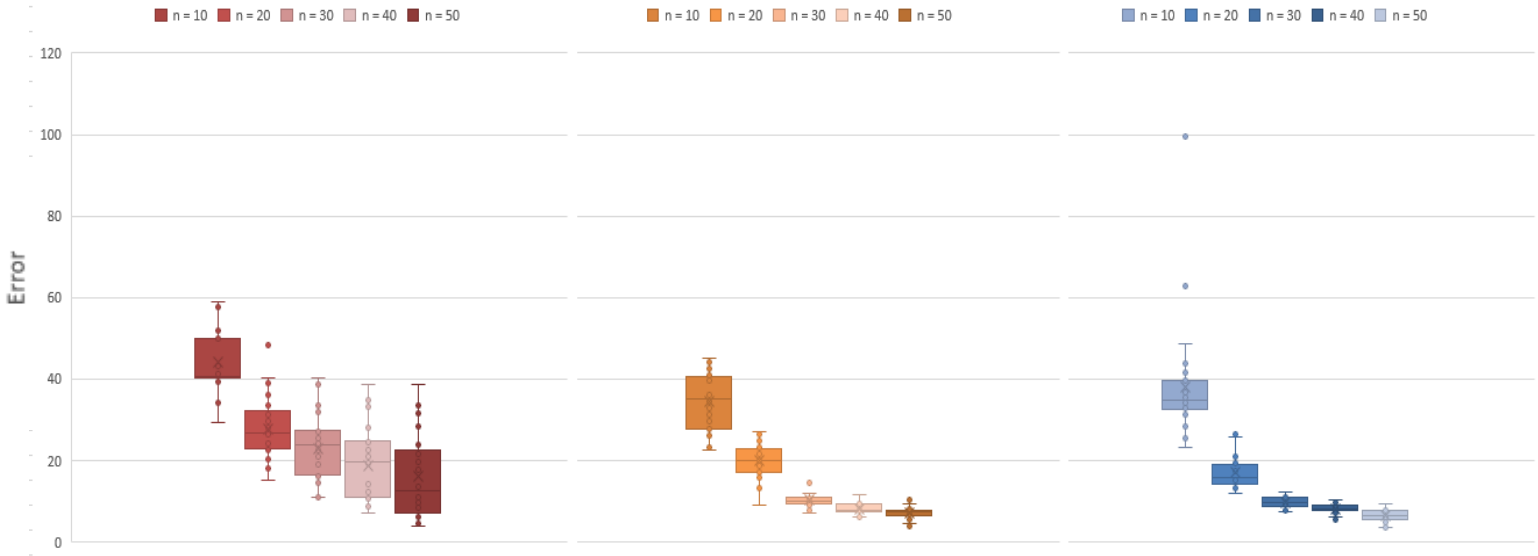

Figura 4-5. Resultados obtenidos para el experimento 1 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

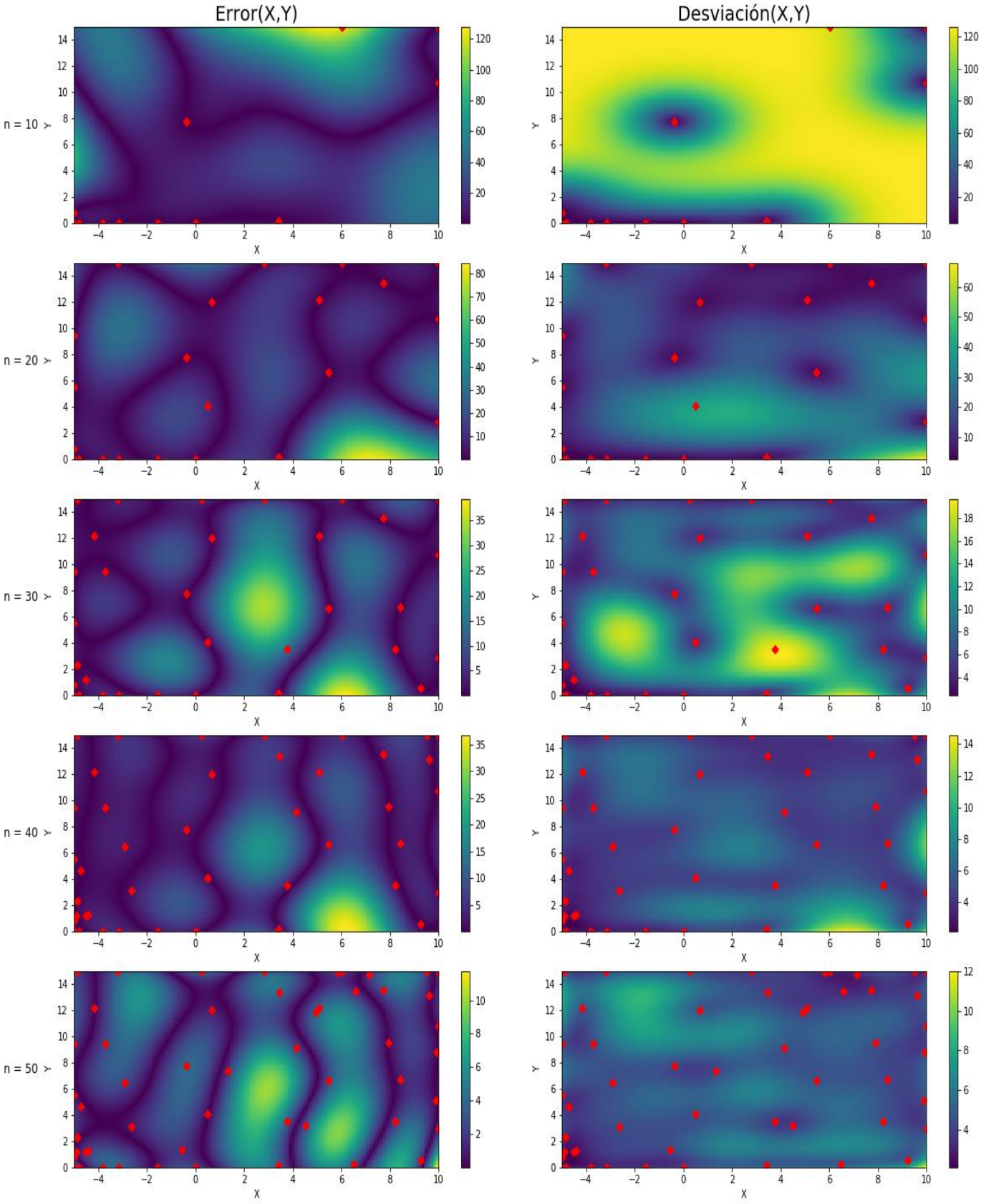

Figura 4-6. Evolución de los mapas de error y precisión durante el experimento 1

Experimento 2: Branin - RBF – Expected Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	29,95152	76,61173
	20	16,00498	25,29415
	30	9,953684	5,367825
	40	8,124547	4,639944
	50	6,540404	4,396046
0.1	10	31,40159	80,24767
	20	13,99115	15,37785
	30	9,207279	5,182346
	40	7,478193	4,394321
	50	6,373788	4,342641
0.2	10	30,8151	57,56132
	20	13,33601	10,7587
	30	8,783677	5,416258
	40	7,4899	4,286748
	50	6,32555	4,329187

Tabla 4-2. Resumen de resultados para el experimento 2

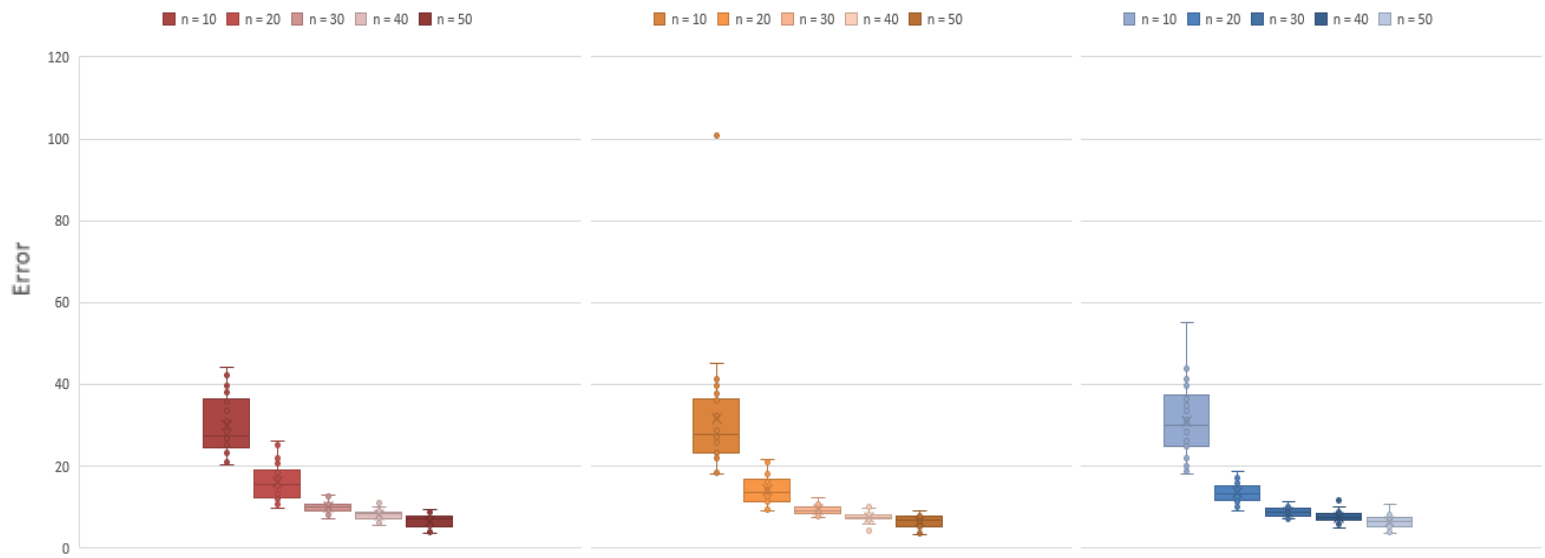

Figura 4-7. Resultados obtenidos para el experimento 2 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

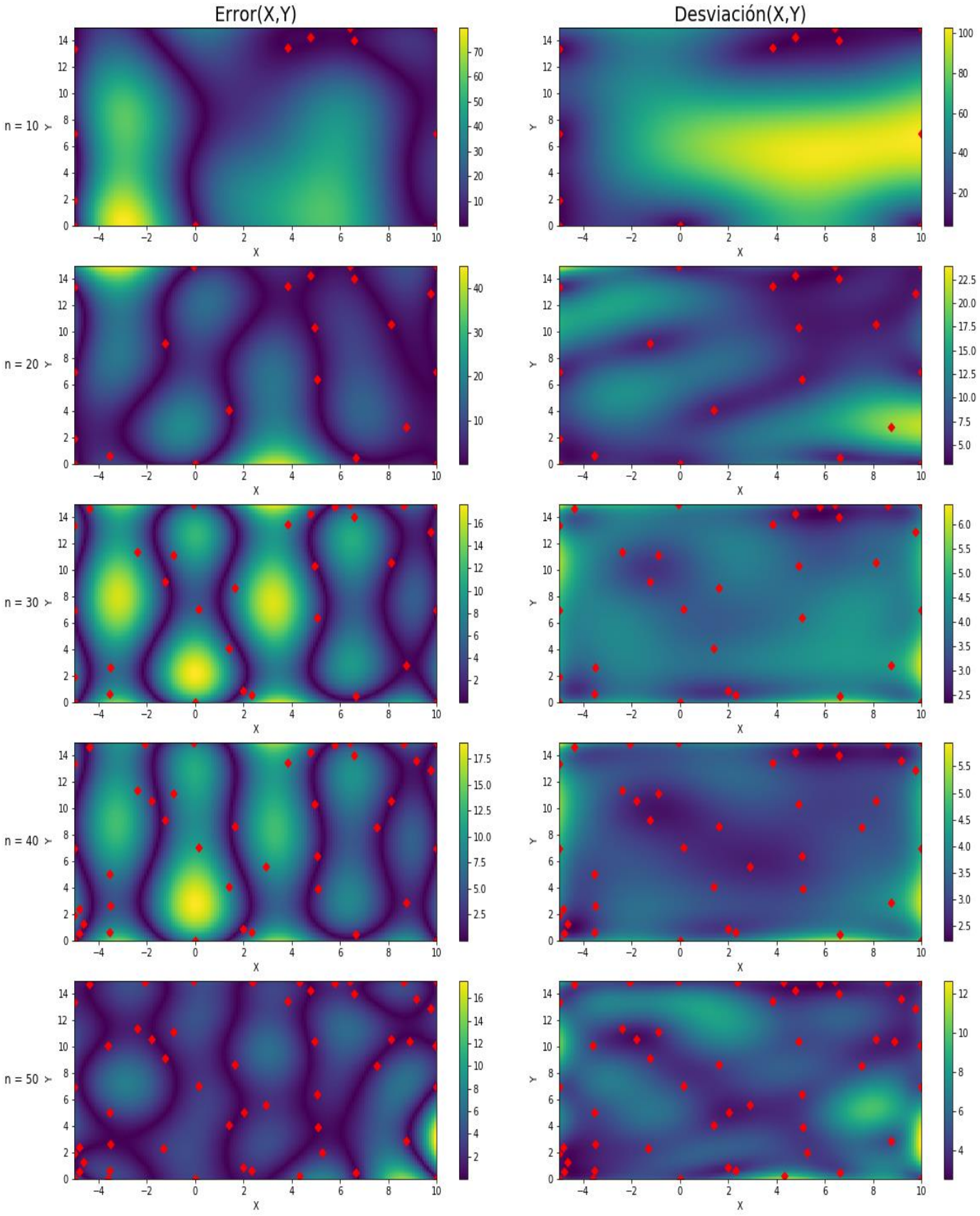

Figura 4-8. Evolución de los mapas de error y precisión durante el experimento 2

Experimento 3: Branin - RBF - Upper Confidence Bound

ξ	n	e_{cm}	σ_{cm}
0.01	10	45,36464	94,01446
	20	50,63141	122,5875
	30	55,3247	139,4896
	40	55,07298	143,3605
	50	55,07749	143,4042
0.1	10	47,68209	121,3829
	20	60,73948	155,552
	30	60,766	155,843
	40	60,76506	155,7414
	50	60,71696	155,7442
0.2	10	53,97435	132,7817
	20	60,95523	148,2527
	30	60,75523	150,3981
	40	60,76647	150,5017
	50	60,61323	150,2205

Tabla 4-3. Resumen de resultados para el experimento 3

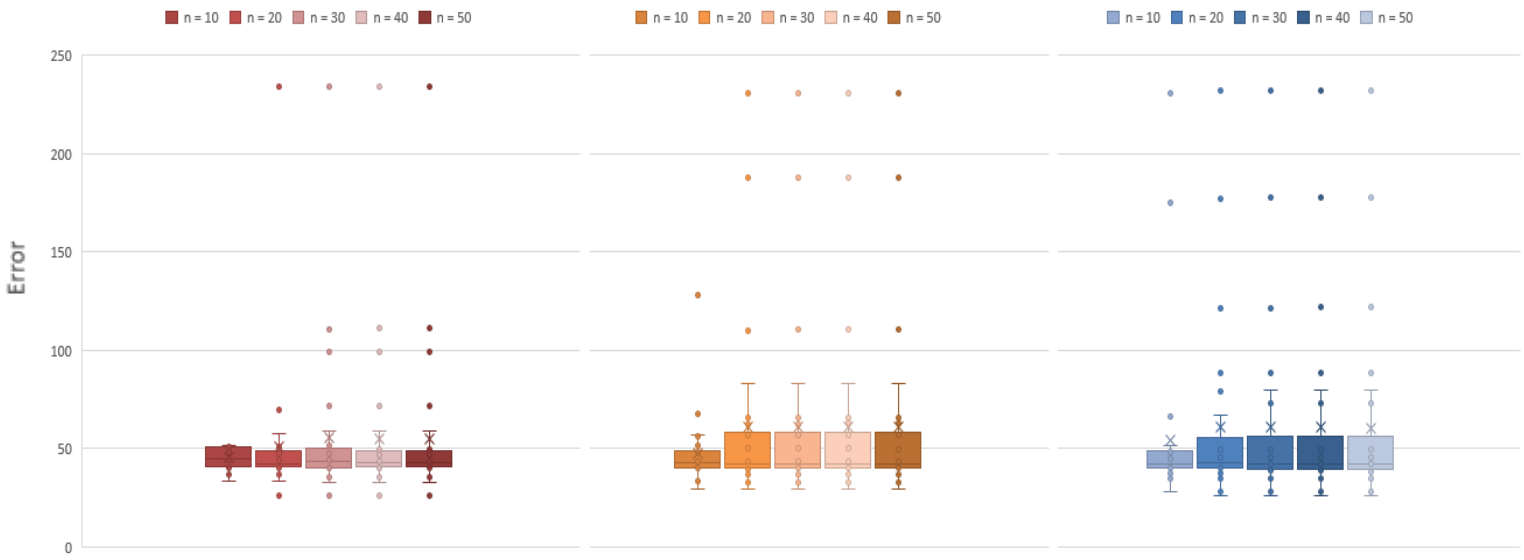

Figura 4-9. Resultados obtenidos para el experimento 3 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

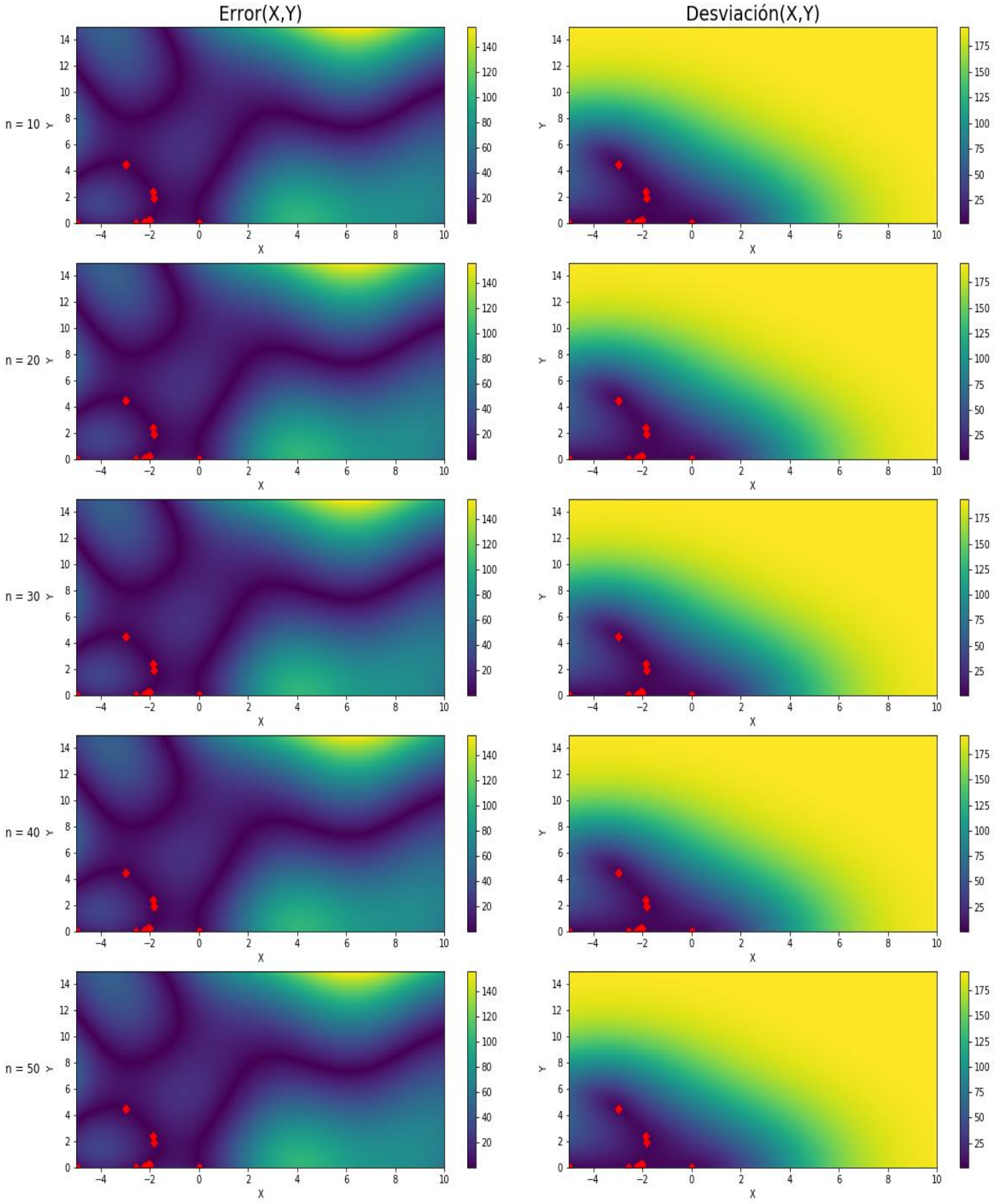

Figura 4-10. Evolución de los mapas de error y precisión durante el experimento 3

Experimento 4: Branin - Màtern kernel - Probability of Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	65,68003	114,9414
	20	48,1067	88,85316
	30	32,06024	49,85593
	40	25,84069	38,19654
	50	23,22447	31,03117
0.1	10	39,35278	95,39939
	20	16,79825	33,95207
	30	8,929918	13,91978
	40	6,330839	9,551916
	50	4,740604	7,698958
0.2	10	45,72681	69,4166
	20	12,82491	25,8767
	30	8,440863	11,97793
	40	5,880425	8,977626
	50	4,377671	7,370899

Tabla 4-4. Resumen de resultados para el experimento 4

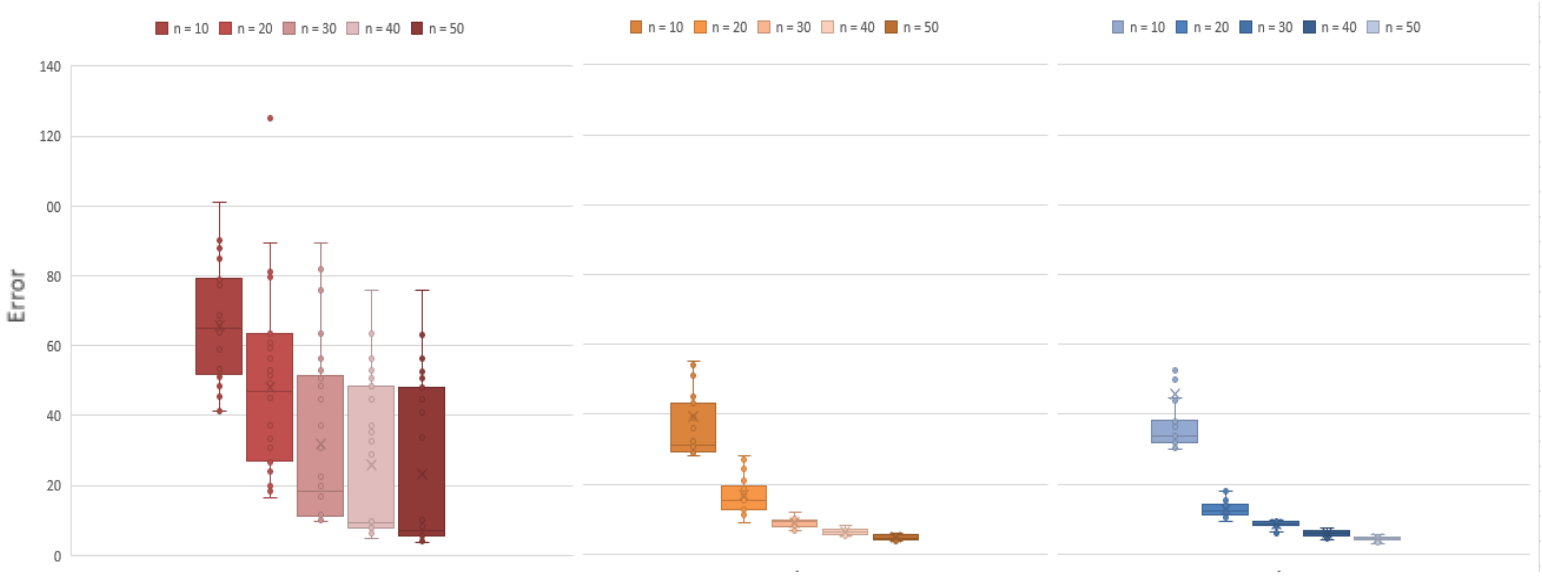

Figura 4-11. Resultados obtenidos para el experimento 4 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

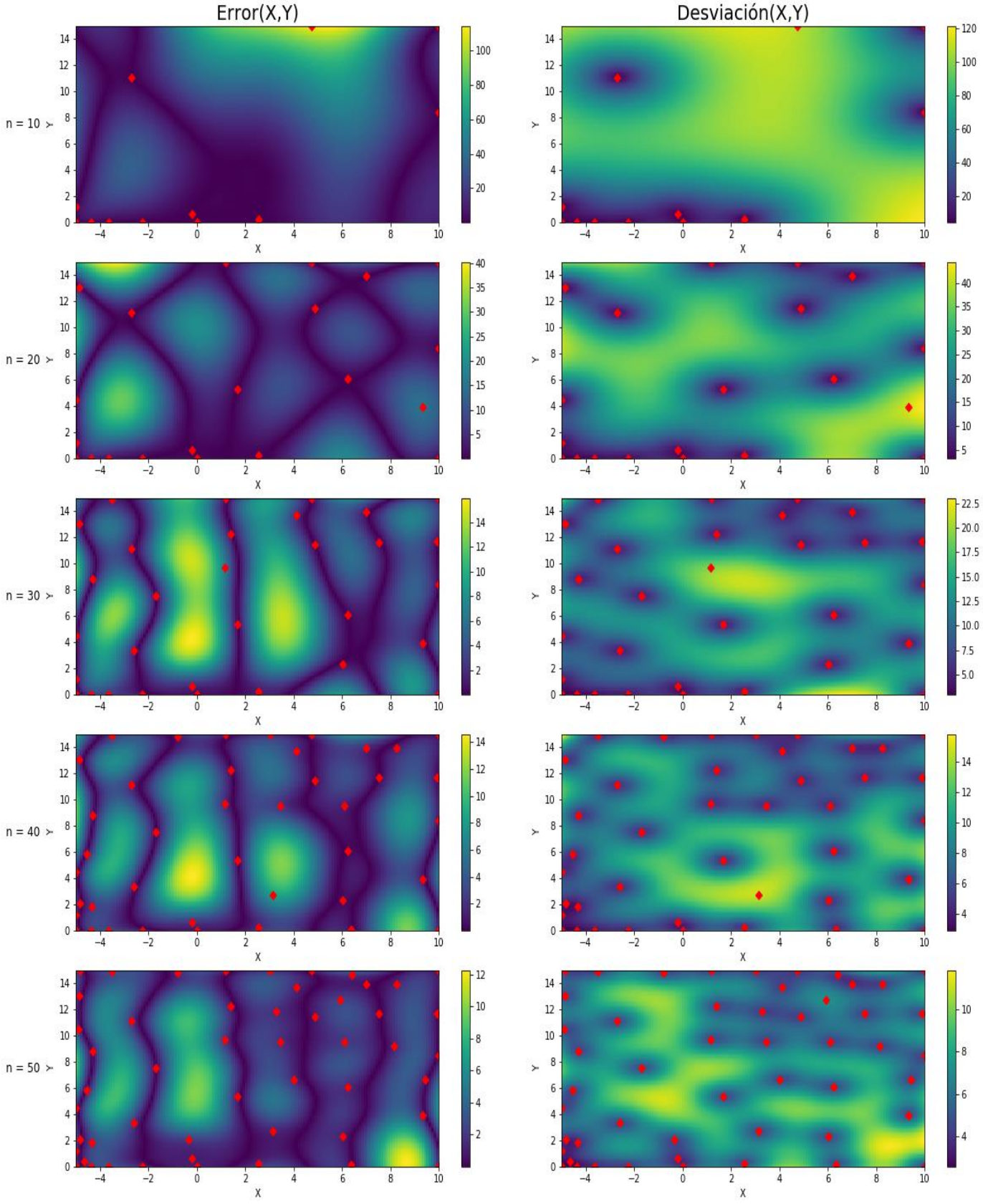

Figura 4-12. Evolución de los mapas de error y precisión durante el experimento 4

Experimento 5: Branin - Màtern kernel - Expected Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	34,30103	68,59154
	20	14,51102	24,21988
	30	9,379668	12,97234
	40	7,163106	9,881139
	50	5,605673	8,39426
0.1	10	33,89855	75,44512
	20	11,61629	19,82972
	30	8,090199	11,13779
	40	5,947346	8,513646
	50	4,472125	7,258064
0.2	10	31,84267	58,89502
	20	10,95404	19,50552
	30	8,093116	10,89064
	40	5,617009	8,401158
	50	4,244055	7,128649

Tabla 4-5. Resumen de resultados para el experimento 5

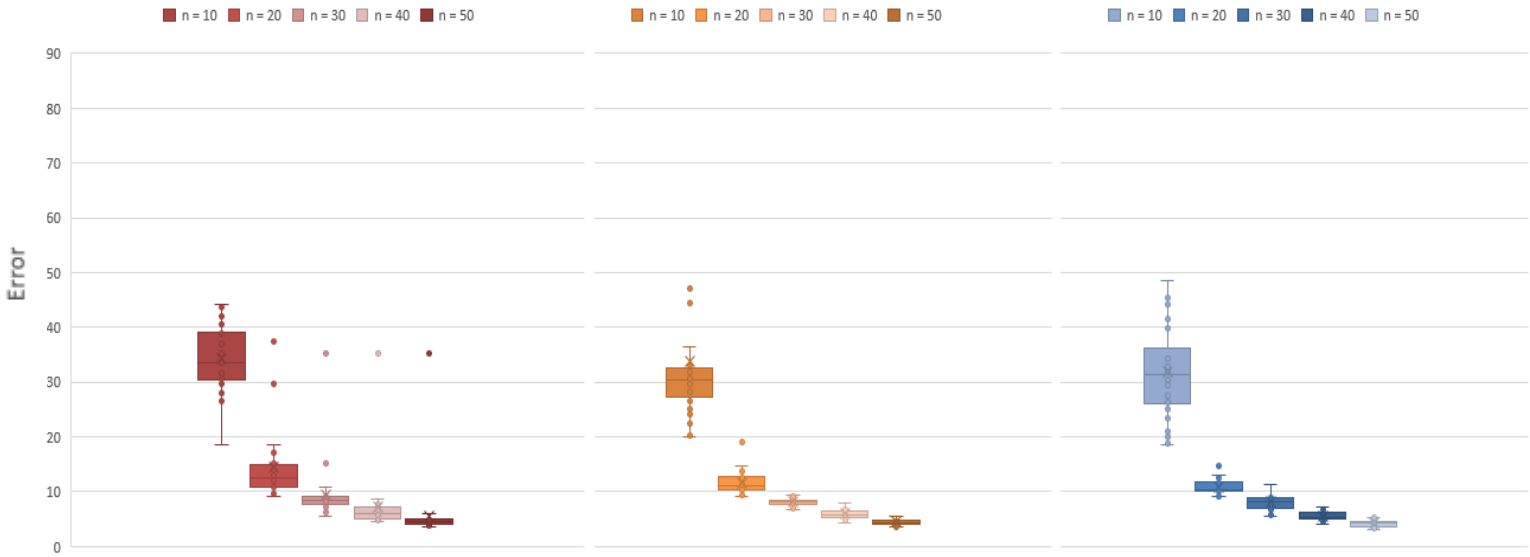

Figura 4-13. Resultados obtenidos para el experimento 5 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

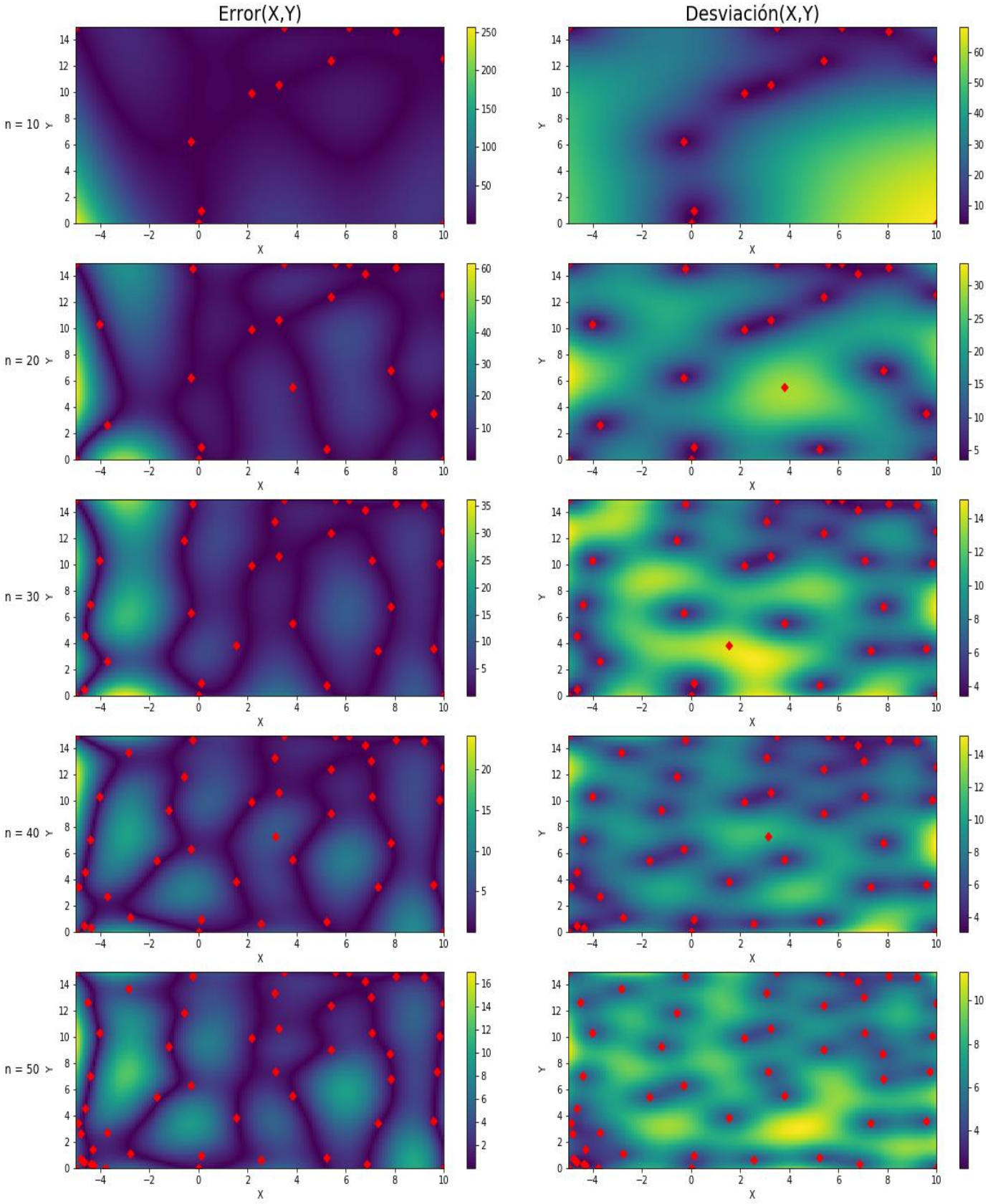

Figura 4-14. Evolución de los mapas de error y precisión durante el experimento 5

Experimento 6: Branin - Màtern kernel - Upper Confidence Bound

ξ	n	e_{cm}	σ_{cm}
0.01	10	56,5816	82,08413
	20	58,14318	87,06355
	30	58,10882	87,23676
	40	58,1014	94,16946
	50	59,05679	99,87241
0.1	10	56,75197	103,2499
	20	67,6573	135,9287
	30	83,71597	150,0557
	40	85,39836	150,3818
	50	85,40811	150,396
0.2	10	61,22607	127,4201
	20	84,18145	147,7641
	30	85,36536	148,0076
	40	85,34896	148,2518
	50	85,25059	148,1951

Tabla 4-6. Resumen de resultados para el experimento 6

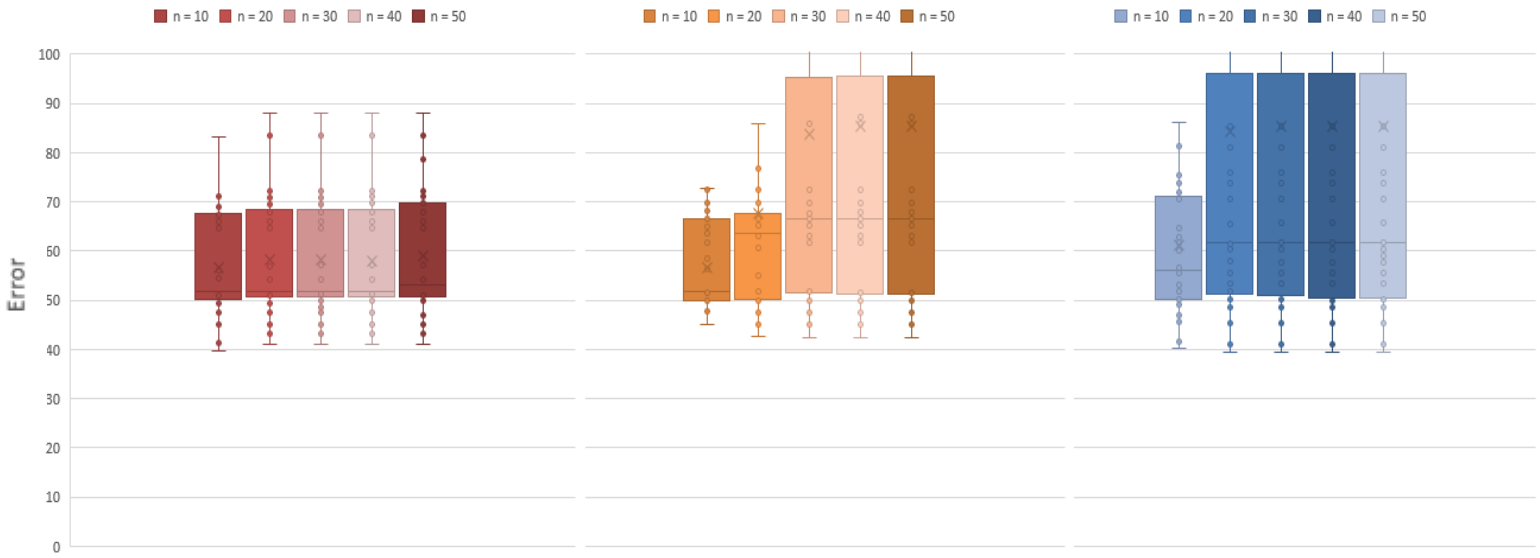

Figura 4-15. Resultados obtenidos para el experimento 6 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

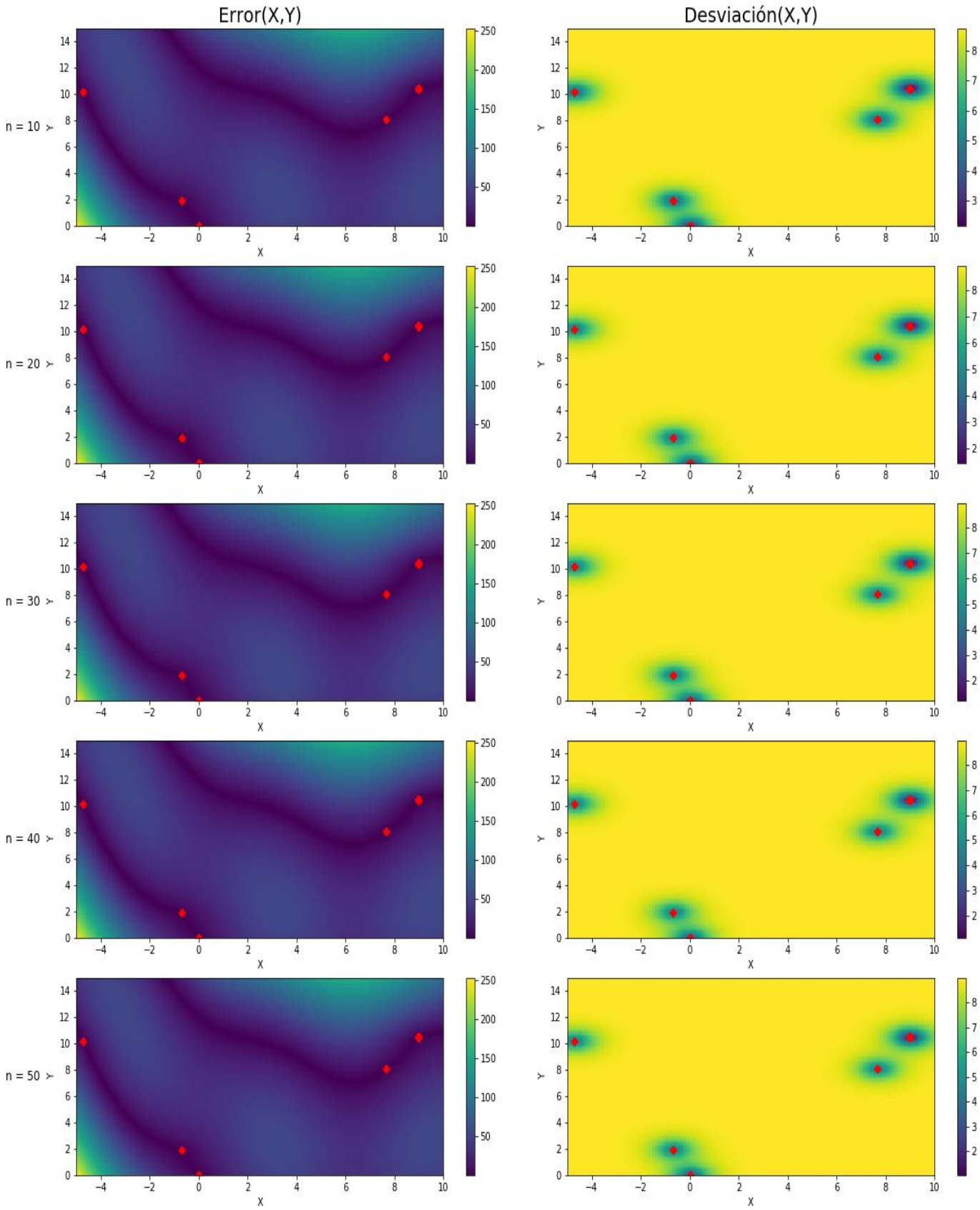

Figura 4-16. Evolución de los mapas de error y precisión durante el experimento 6

Experimento 7: Schwefel - RBF - Probability of Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	275,1924	274,5377
	20	271,0361	280,6564
	30	269,9369	274,0069
	40	266,9901	269,2102
	50	264,0339	261,2116
0.1	10	271,9169	282,1527
	20	262,0595	270,907
	30	256,427	259,5464
	40	251,5191	247,7878
	50	245,5649	238,781
0.2	10	274,2515	258,0262
	20	260,6021	263,0362
	30	253,3126	254,7738
	40	246,9116	247,4898
	50	238,8641	231,6963

Tabla 4-7. Resumen de resultados para el experimento 7

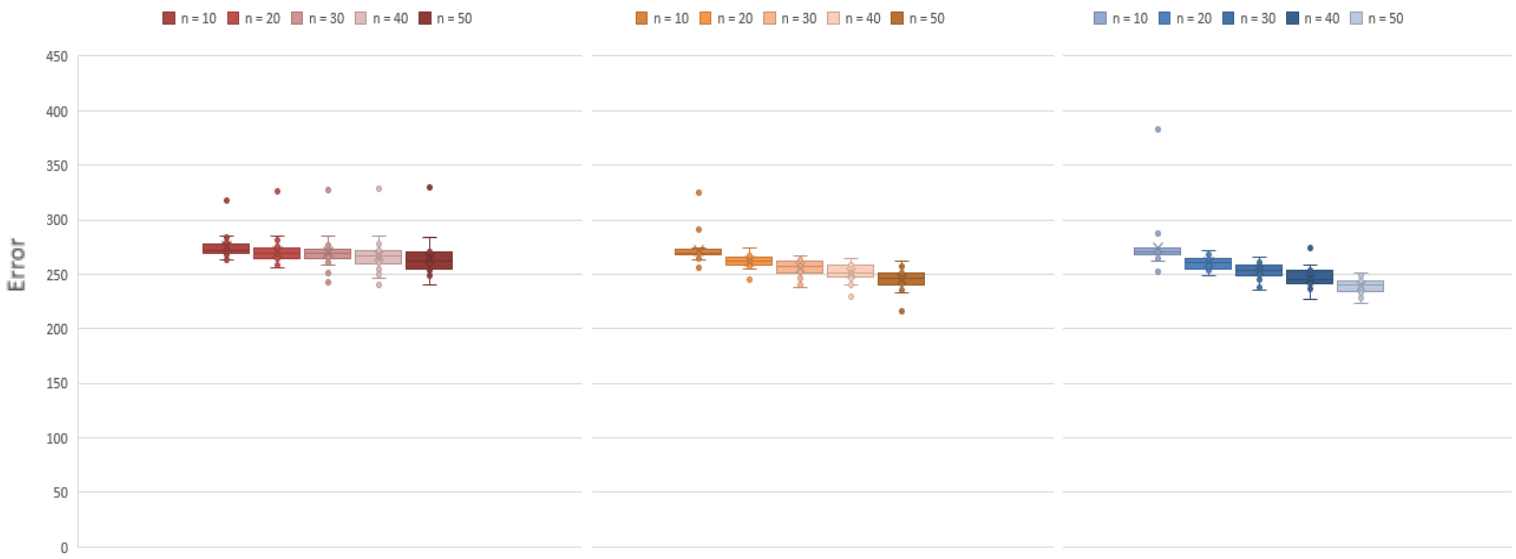

Figura 4-17. Resultados obtenidos para el experimento 7 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

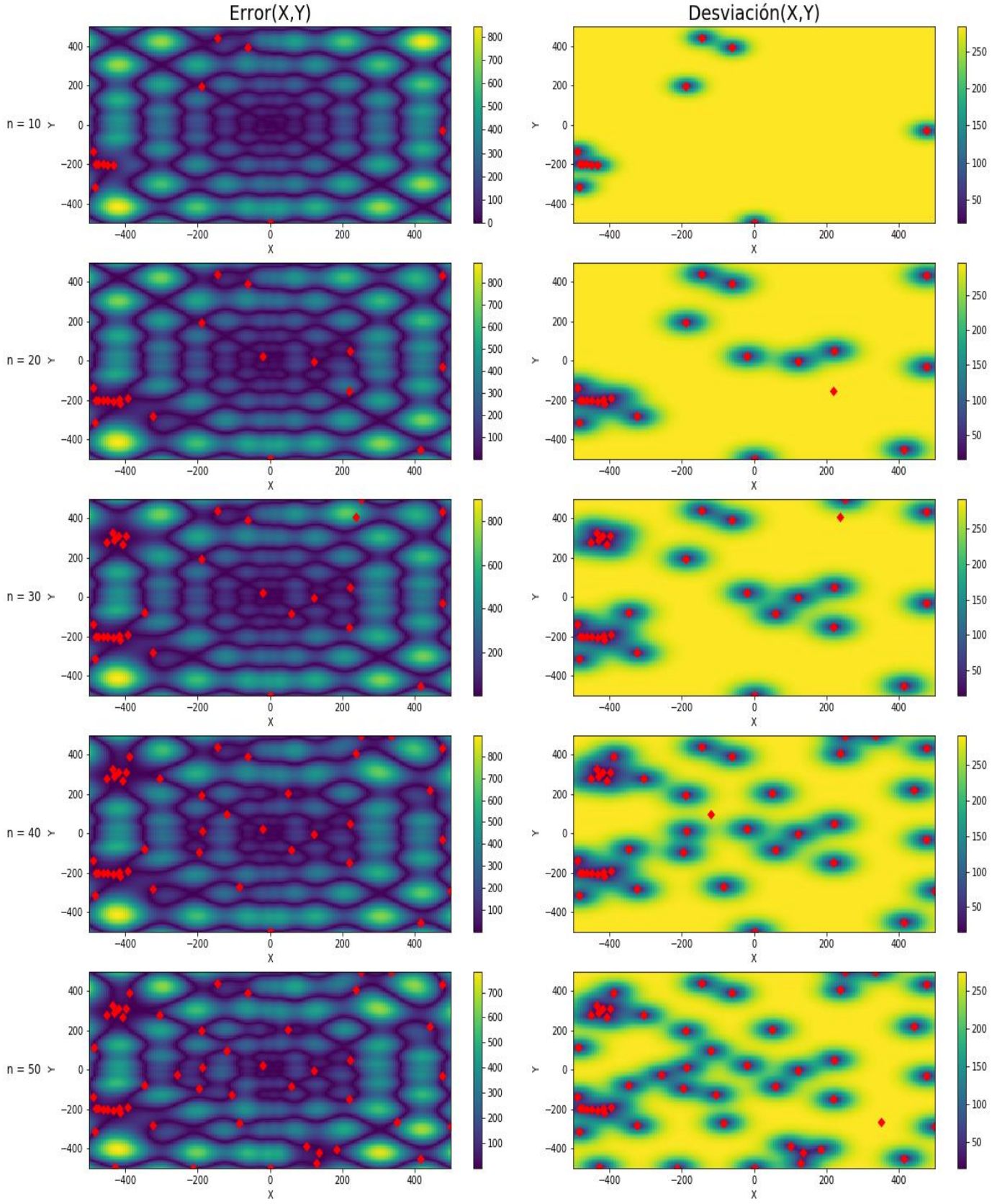

Figura 4-18. Evolución de los mapas de error y precisión durante el experimento 7

Experimento 8: Schwefel - RBF – Expected Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	277,9958	257,0411
	20	262,367	260,6042
	30	255,068	245,71
	40	249,168	238,0689
	50	243,0912	229,3364
0.1	10	278,0818	263,9036
	20	261,3414	259,8486
	30	252,9058	247,1717
	40	245,5073	235,312
	50	236,5413	226,061
0.2	10	276,8218	264,2663
	20	259,7398	265,7426
	30	252,9667	253,0179
	40	243,573	238,8563
	50	236,4514	225,0281

Tabla 4-8. Resumen de resultados para el experimento 8

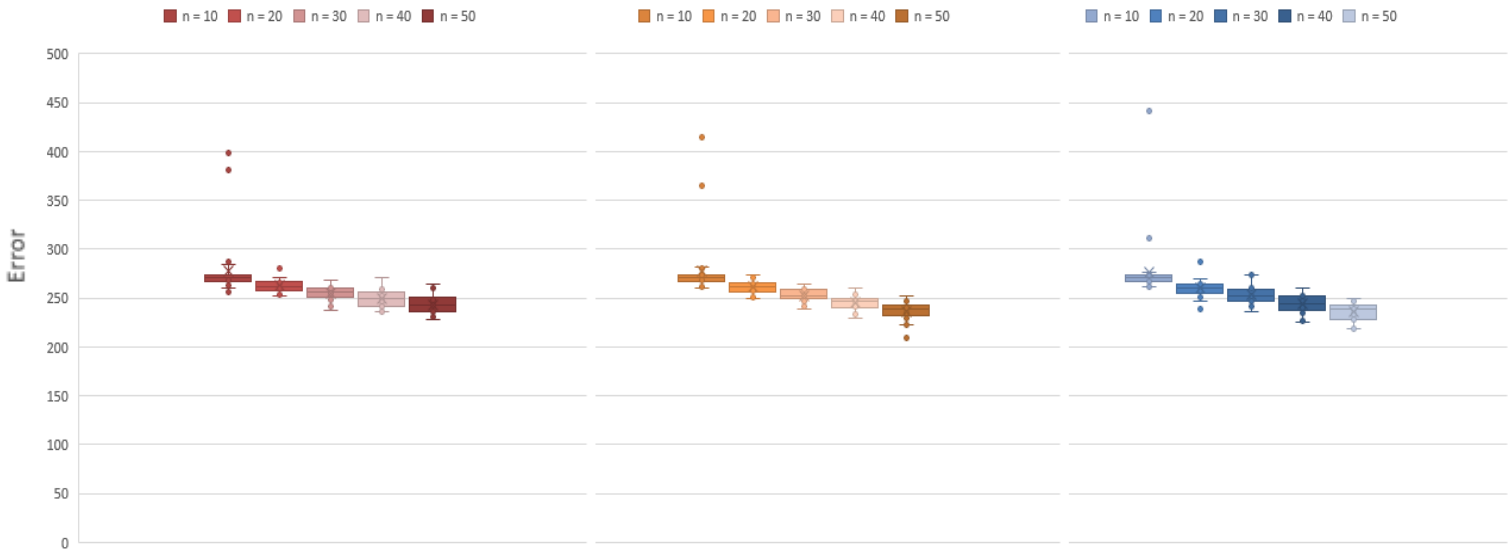

Figura 4-19. Resultados obtenidos para el experimento 8 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

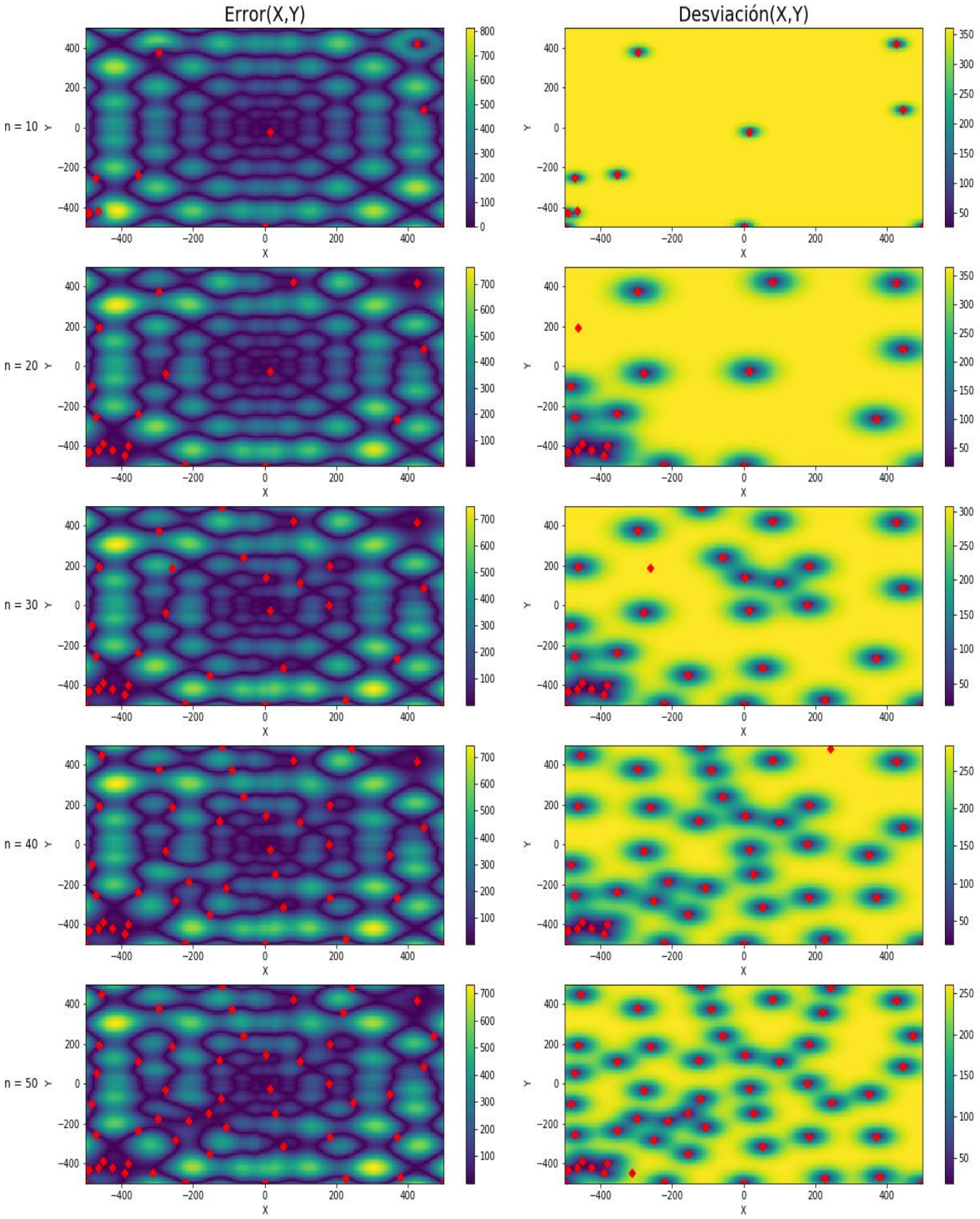

Figura 4-20. Evolución de los mapas de error y precisión durante el experimento 8

Experimento 9: Schwefel - RBF - Upper Confidence Bound

ξ	n	e_{cm}	σ_{cm}
0.01	10	286,8533	262,0136
	20	287,0928	279,7611
	30	285,0387	278,9623
	40	285,6591	286,4022
	50	284,0568	287,5619
0.1	10	276,9246	272,9275
	20	274,8248	290,7182
	30	274,703	287,4644
	40	274,7673	284,4386
	50	274,7472	282,7443
0.2	10	270,3407	281,9114
	20	268,2504	288,2165
	30	268,189	288,8565
	40	267,4405	288,5387
	50	266,5523	292,4716

Tabla 4-9. Resumen de resultados para el experimento 9

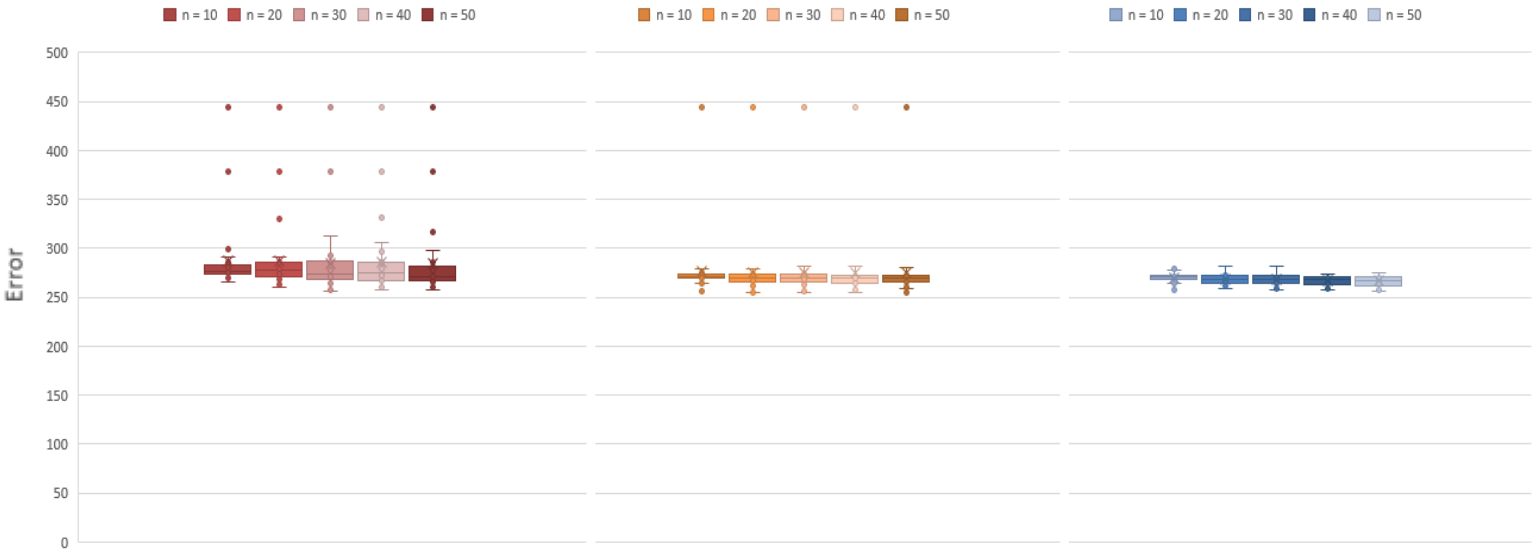

Figura 4-21. Resultados obtenidos para el experimento 9 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

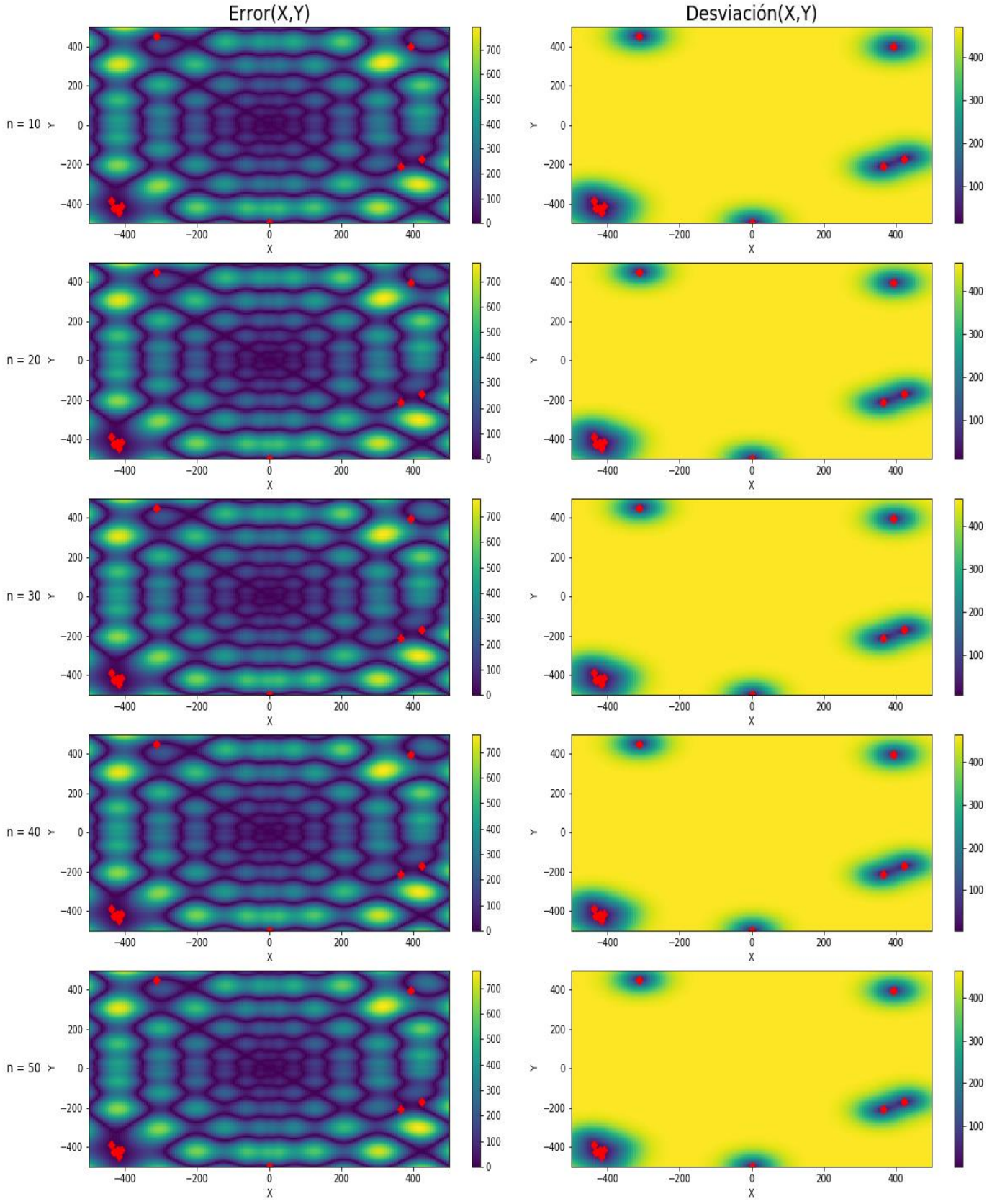

Figura 4-22. Evolución de los mapas de error y precisión durante el experimento 9

Experimento 10: Schwefel - Màtern kernel - Probability of Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	296,283	279,5525
	20	291,5839	294,0639
	30	288,4264	296,199
	40	288,0077	293,9555
	50	287,7507	289,9383
0.1	10	284,406	296,2679
	20	273,4523	272,5003
	30	270,2511	257,4923
	40	267,7147	245,4775
	50	268,9378	233,7598
0.2	10	275,6236	267,2042
	20	270,4353	255,7418
	30	268,5634	242,6891
	40	264,9959	237,2975
	50	263,6105	229,2345

Tabla 4-10. Resumen de resultados para el experimento 10

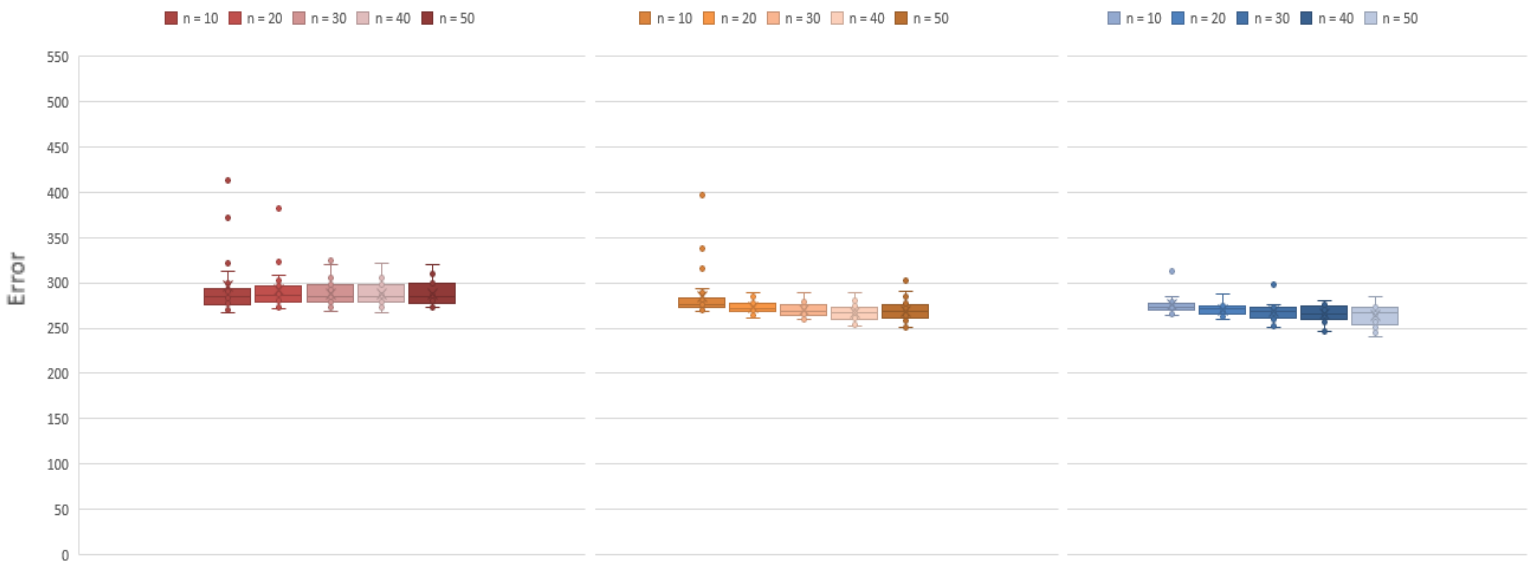

Figura 4-23. Resultados obtenidos para el experimento 10 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

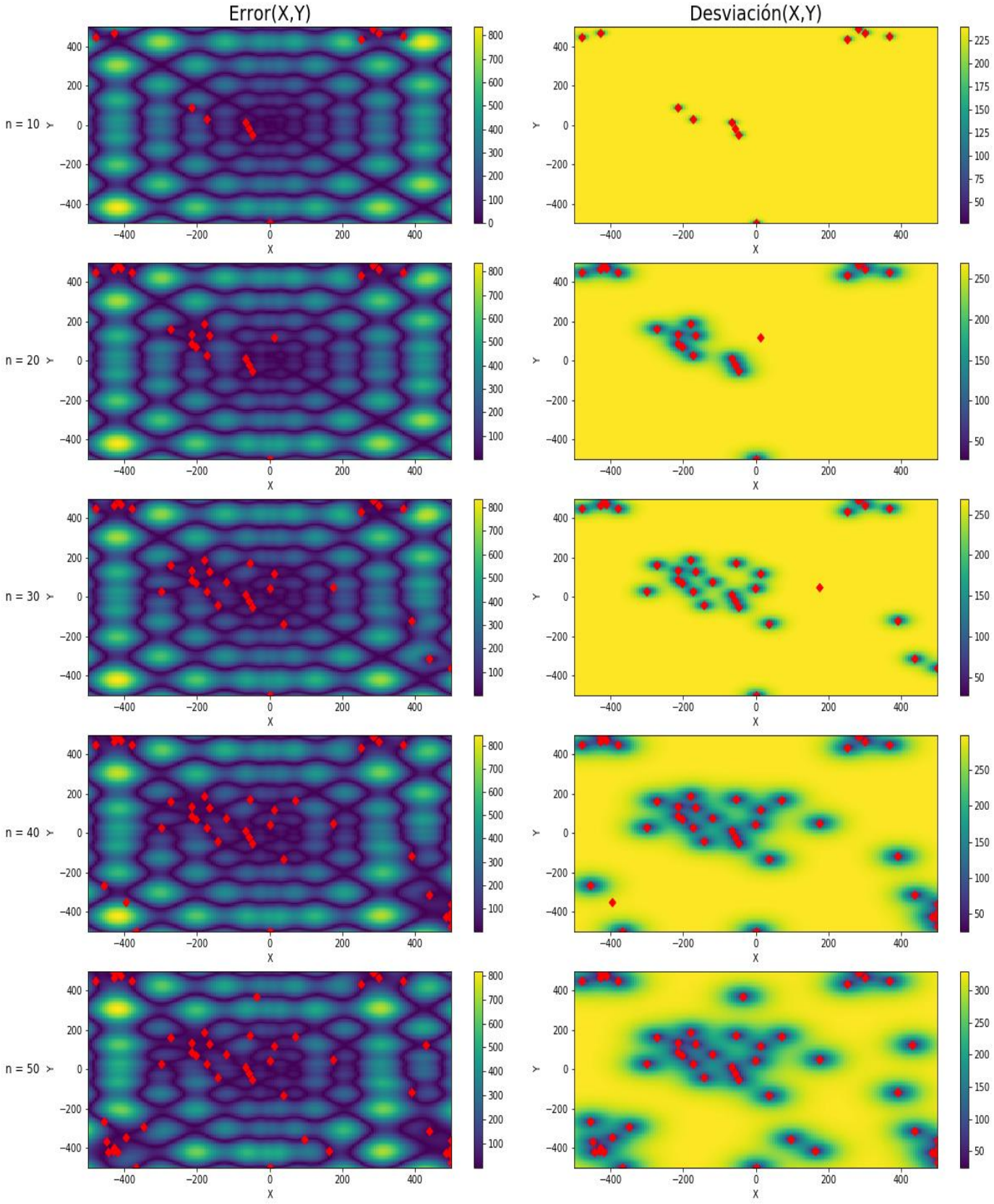

Figura 4-24. Evolución de los mapas de error y precisión durante el experimento 10

Experimento 11: Schwefel - Màtern kernel - Expected Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	274,161	264,5666
	20	267,2623	262,5879
	30	266,3991	247,5796
	40	265,9628	235,6715
	50	260,0571	226,1262
0.1	10	276,6291	262,1144
	20	265,9611	261,6339
	30	261,9644	251,4827
	40	257,8072	241,1697
	50	254,3258	230,2749
0.2	10	276,532	268,8768
	20	269,6792	257,957
	30	265,4016	249,3654
	40	259,3011	237,1734
	50	251,9166	230,4637

Tabla 4-11. Resumen de resultados para el experimento 11

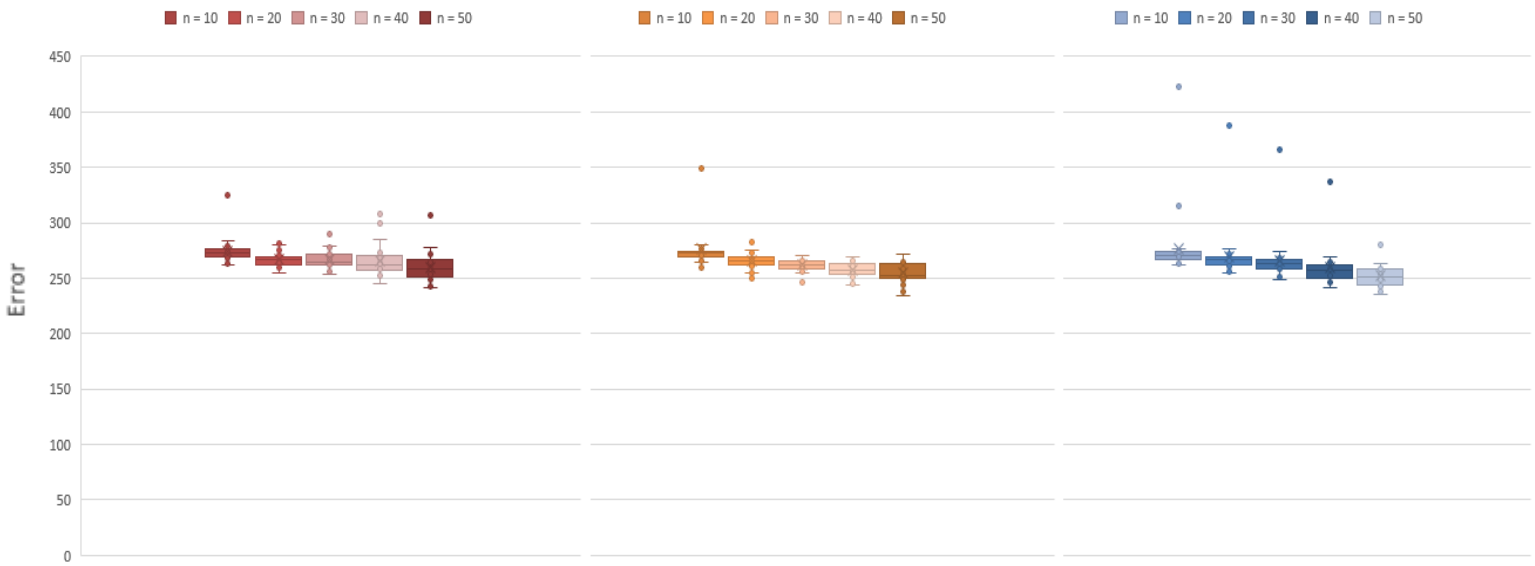

Figura 4-25. Resultados obtenidos para el experimento 11 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

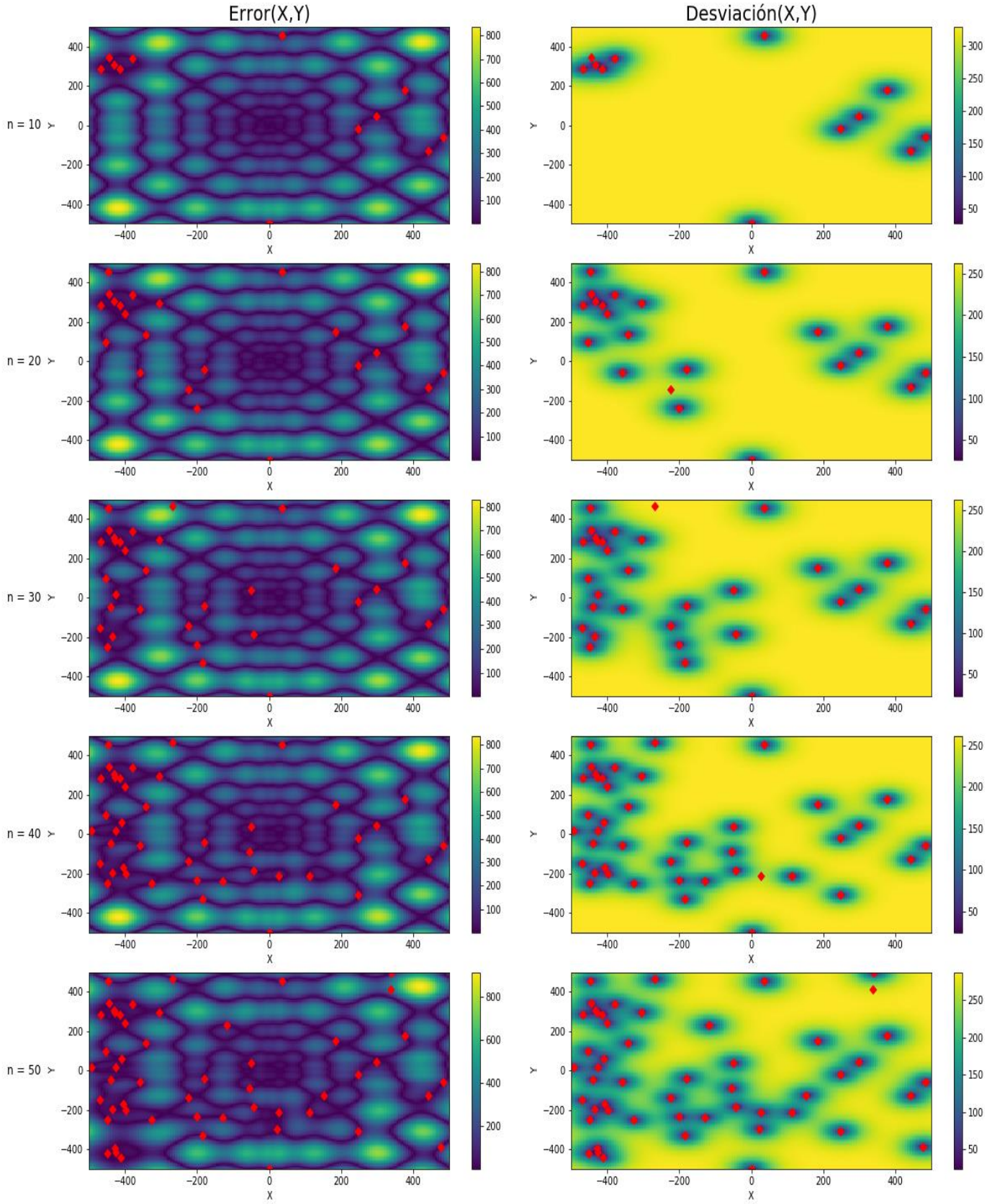

Figura 4-26. Evolución de los mapas de error y precisión durante el experimento 11

Experimento 12: Schwefel - Màtern kernel - Upper Confidence Bound

ξ	n	e_{cm}	σ_{cm}
0.01	10	301,1965	211,8157
	20	301,854	220,423
	30	301,9836	236,0428
	40	303,6296	240,4566
	50	304,1494	240,683
0.1	10	303,6195	238,3356
	20	305,3599	245,2556
	30	305,518	243,0707
	40	305,9632	242,4683
	50	306,2622	242,415
0.2	10	294,5107	248,3997
	20	295,6405	254,9291
	30	296,7841	262,1055
	40	296,9488	262,1932
	50	297,0962	262,3435

Tabla 4-12. Resumen de resultados para el experimento 12

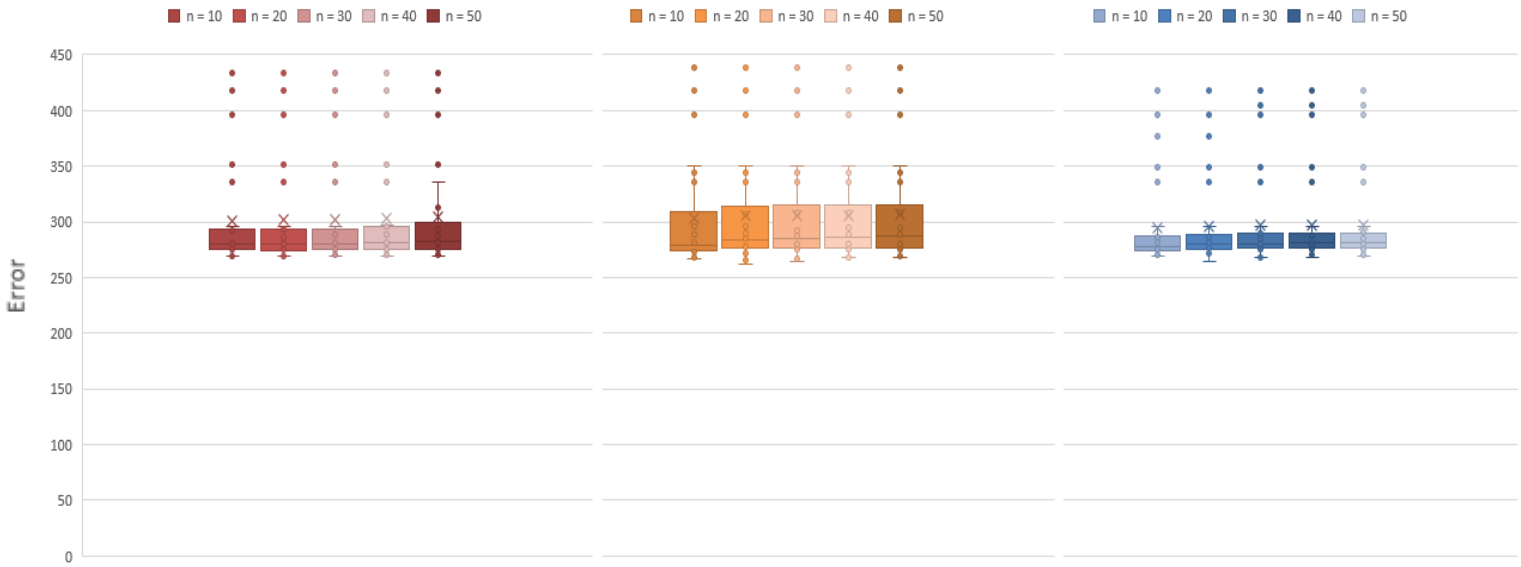

Figura 4-27. Resultados obtenidos para el experimento 12 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

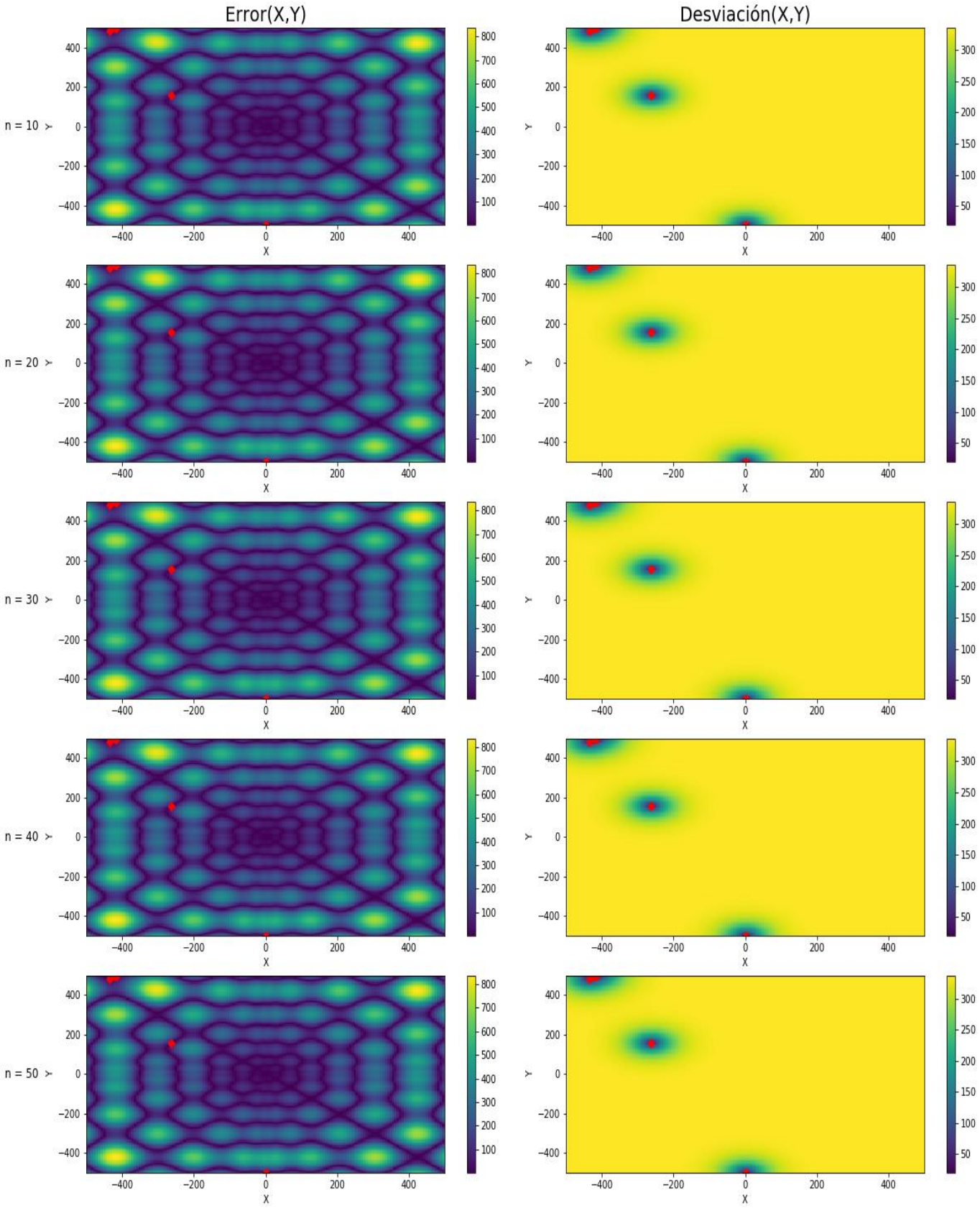

Figura 4-28. Evolución de los mapas de error y precisión durante el experimento 12

Experimento 13: Styblinski-Tang - RBF - Probability of Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	163,8562	48,2266
	20	47,41523	122,7773
	30	57,59379	103,128
	40	47,83466	99,11988
	50	47,63339	78,4238
0.1	10	57,17951	128,1725
	20	48,04974	92,83741
	30	33,11327	63,77079
	40	7,107989	14,28465
	50	2,372481	3,646713
0.2	10	58,60751	119,4962
	20	43,22886	87,58109
	30	22,55457	53,18484
	40	2,805073	4,042866
	50	2,220176	3,342814

Tabla 4-13. Resumen de resultados para el experimento 13

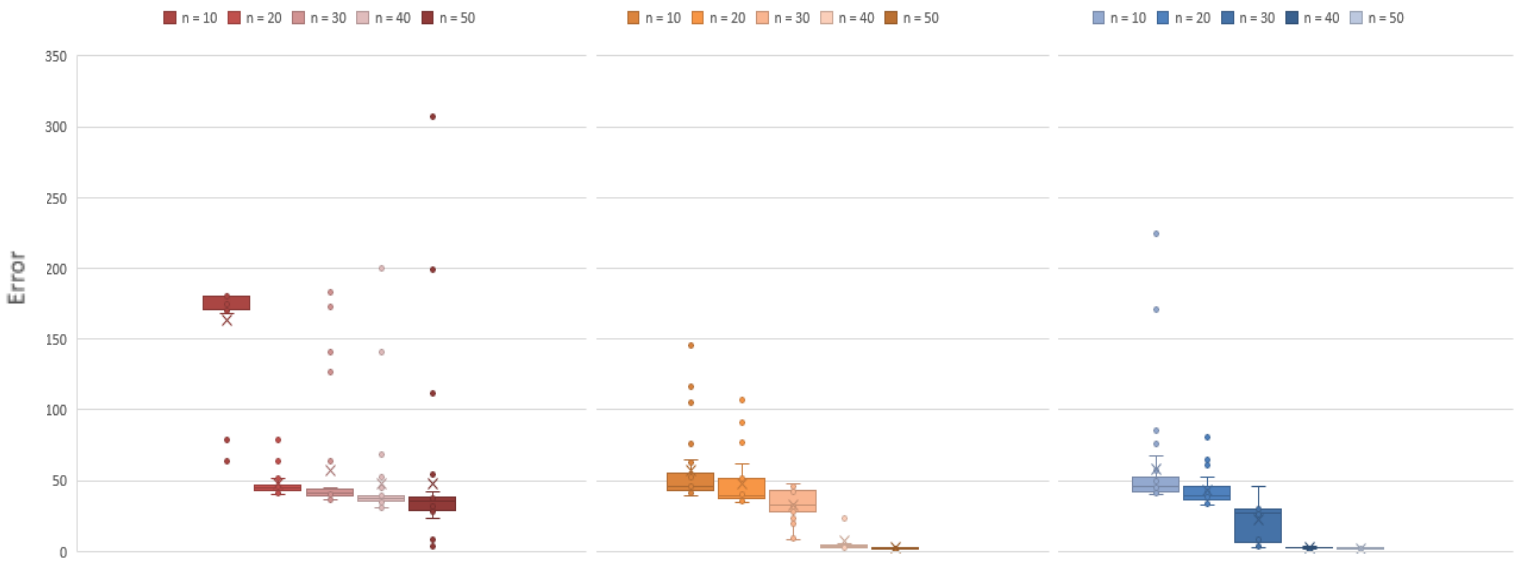

Figura 4-29. Resultados obtenidos para el experimento 13 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

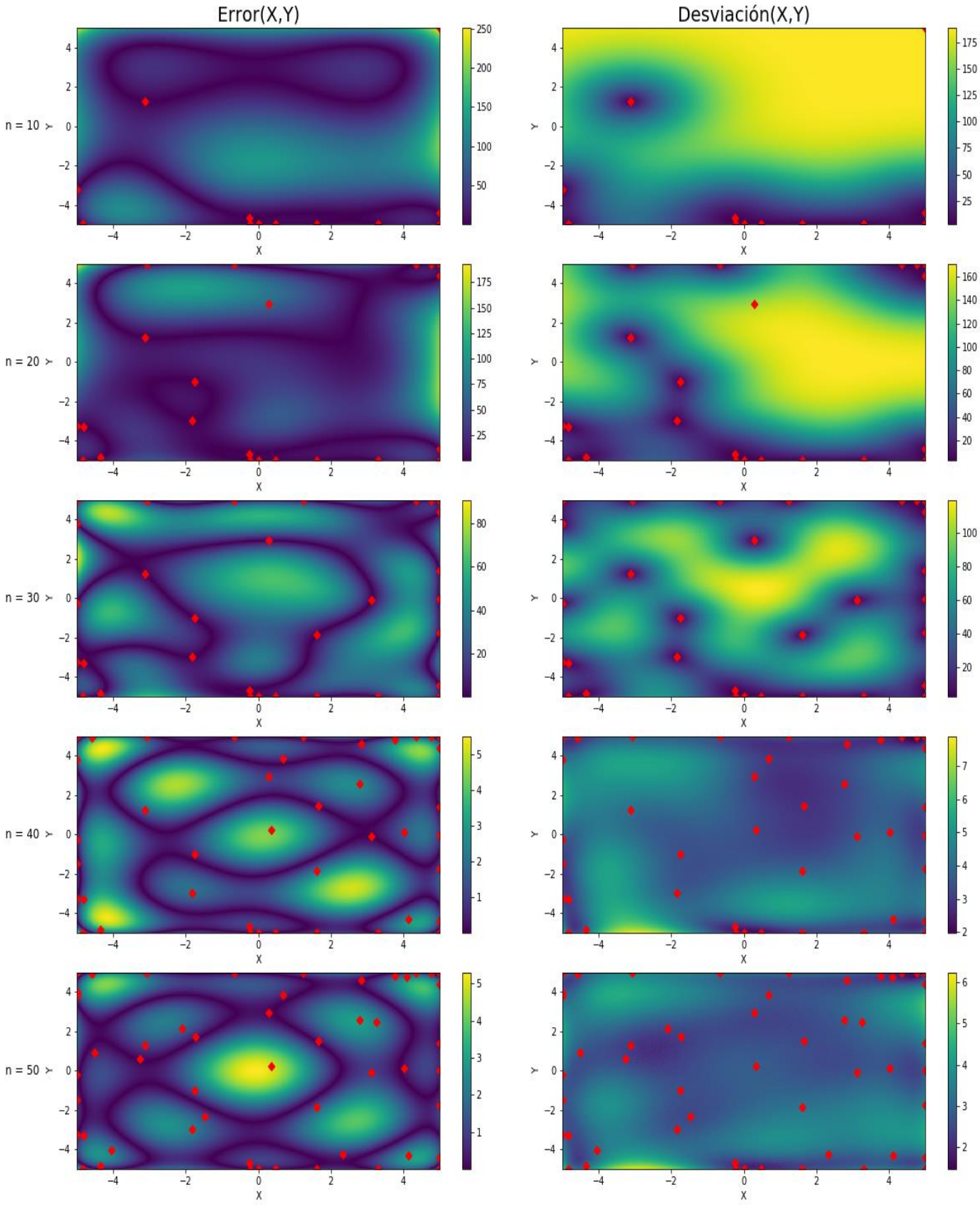

Figura 4-30. Evolución de los mapas de error y precisión durante el experimento 13

Experimento 14: Styblinski-Tang - RBF – Expected Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	60,58986	95,64628
	20	45,99541	84,55547
	30	24,8968	47,048
	40	5,581468	11,53205
	50	2,263652	3,398309
0.1	10	62,91841	93,52237
	20	44,42688	85,8582
	30	24,32549	57,51368
	40	5,708312	11,37686
	50	2,071959	3,299818
0.2	10	58,74697	95,02164
	20	45,23443	74,76332
	30	19,50313	37,83791
	40	3,511051	5,482893
	50	2,238897	3,285646

Tabla 4-14. Resumen de resultados para el experimento 14

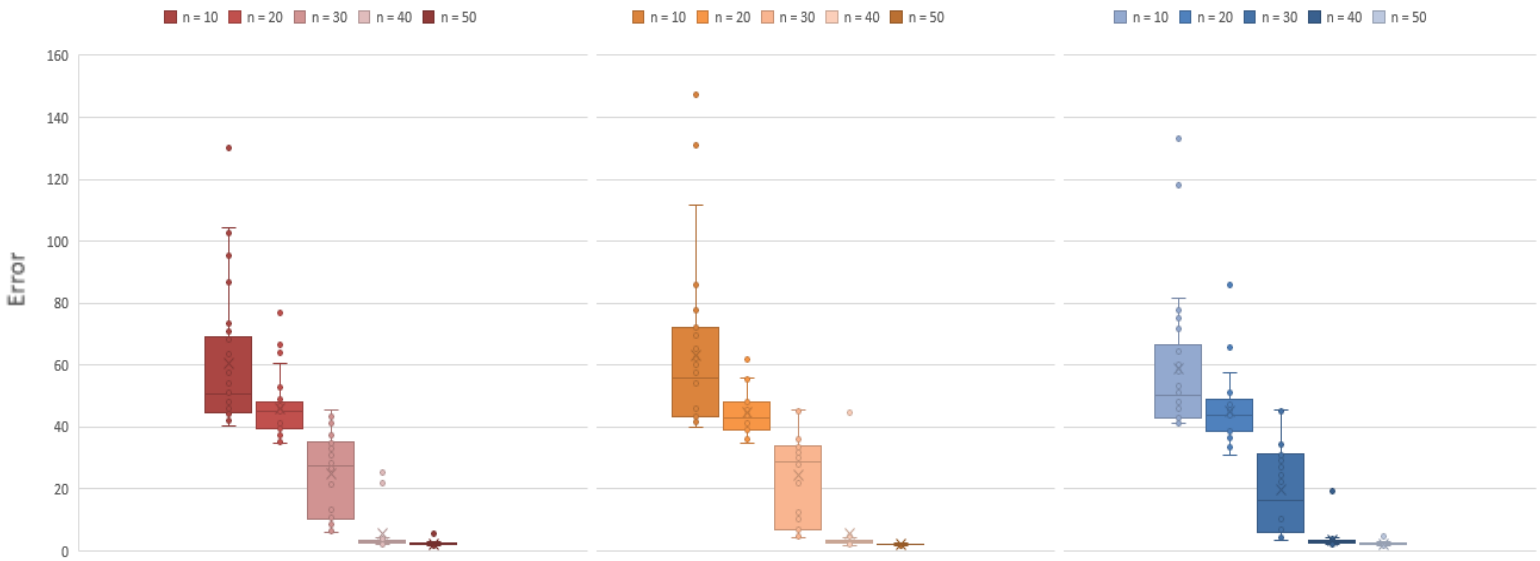

Figura 4-31. Resultados obtenidos para el experimento 14 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

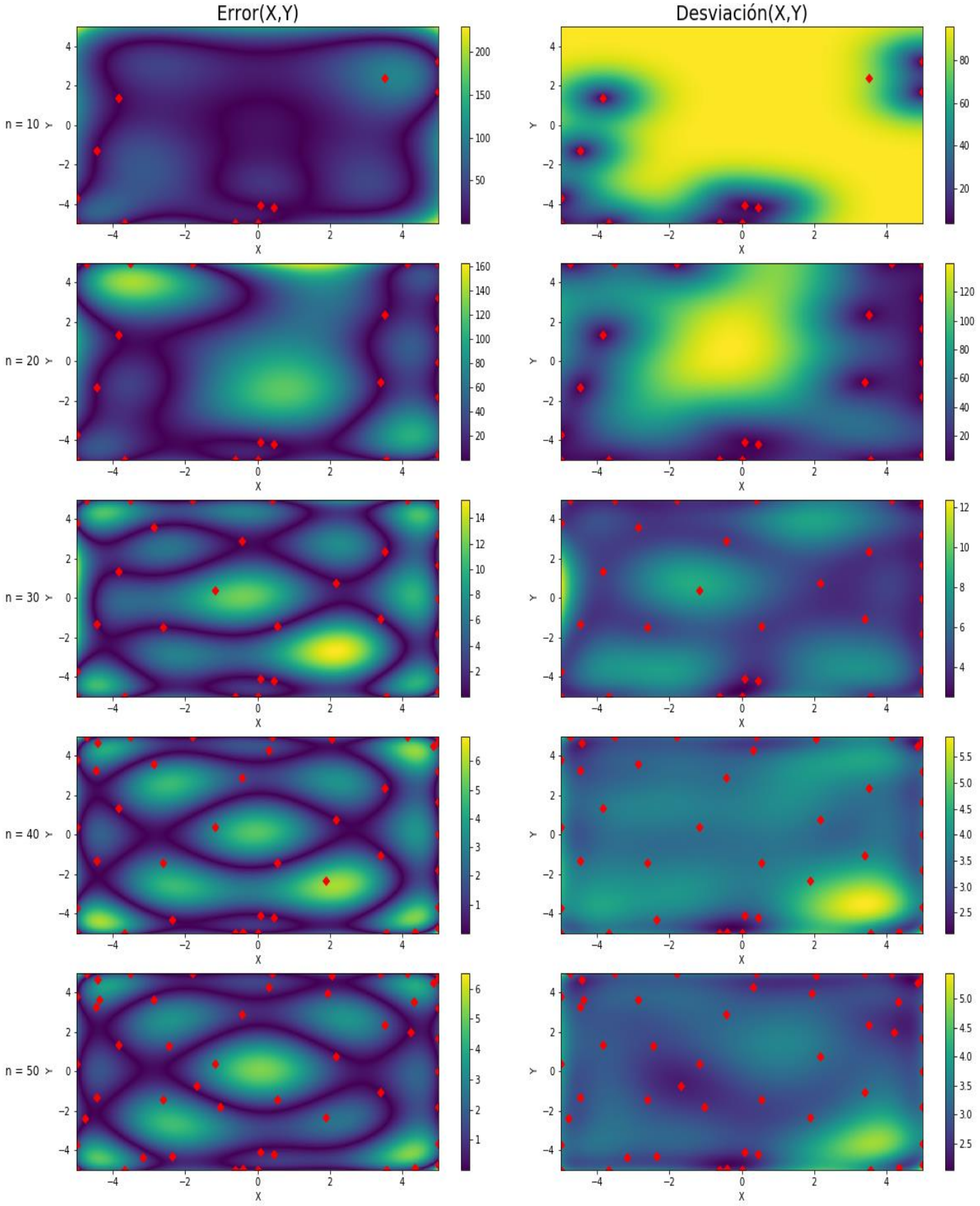

Figura 4-32. Evolución de los mapas de error y precisión durante el experimento 14

Experimento 15: Styblinski-Tang - RBF - Upper Confidence Bound

ξ	n	e_{cm}	σ_{cm}
0.01	10	46,36384	106,9993
	20	98,08343	25,60979
	30	61,58963	87,74274
	40	61,50054	88,09244
	50	61,33036	89,05384
0.1	10	58,68632	99,34411
	20	58,38843	100,7454
	30	58,59393	102,5536
	40	58,58331	102,5749
	50	58,23373	102,893
0.2	10	49,83124	92,82863
	20	48,13505	86,64746
	30	47,87486	90,69775
	40	59,53432	93,8802
	50	59,15811	93,2365

Tabla 4-15. Resumen de resultados para el experimento 15

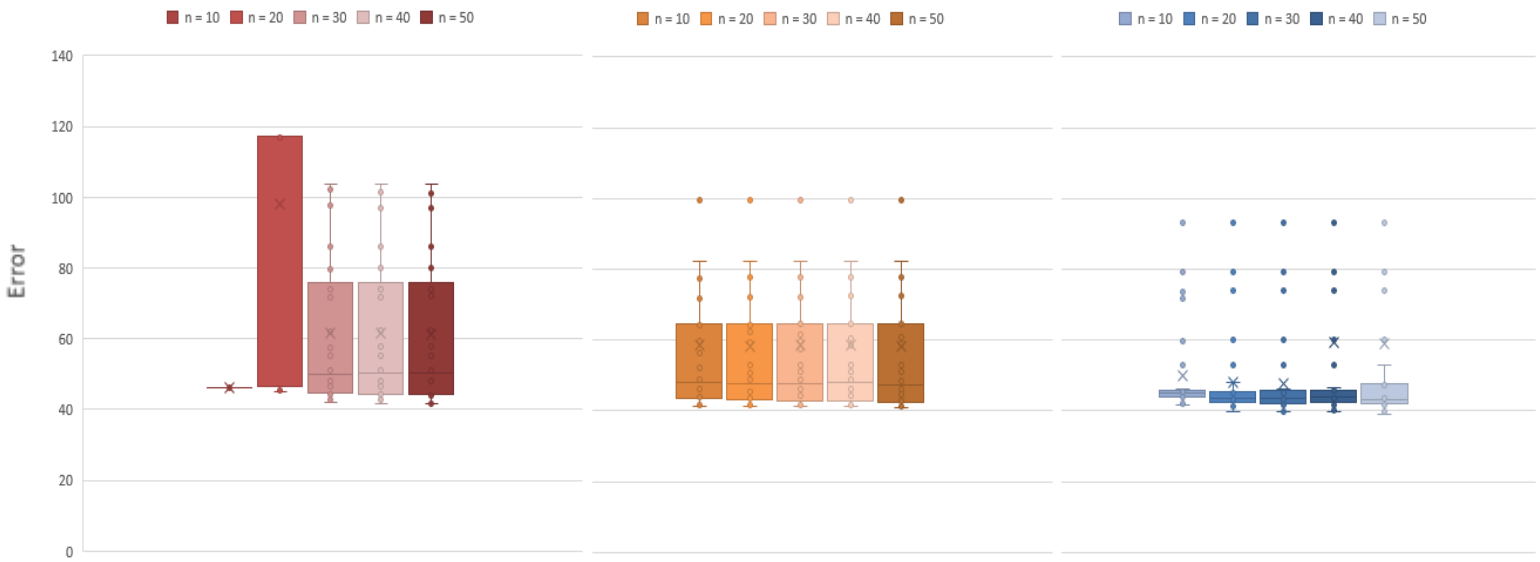

Figura 4-33. Resultados obtenidos para el experimento 15 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

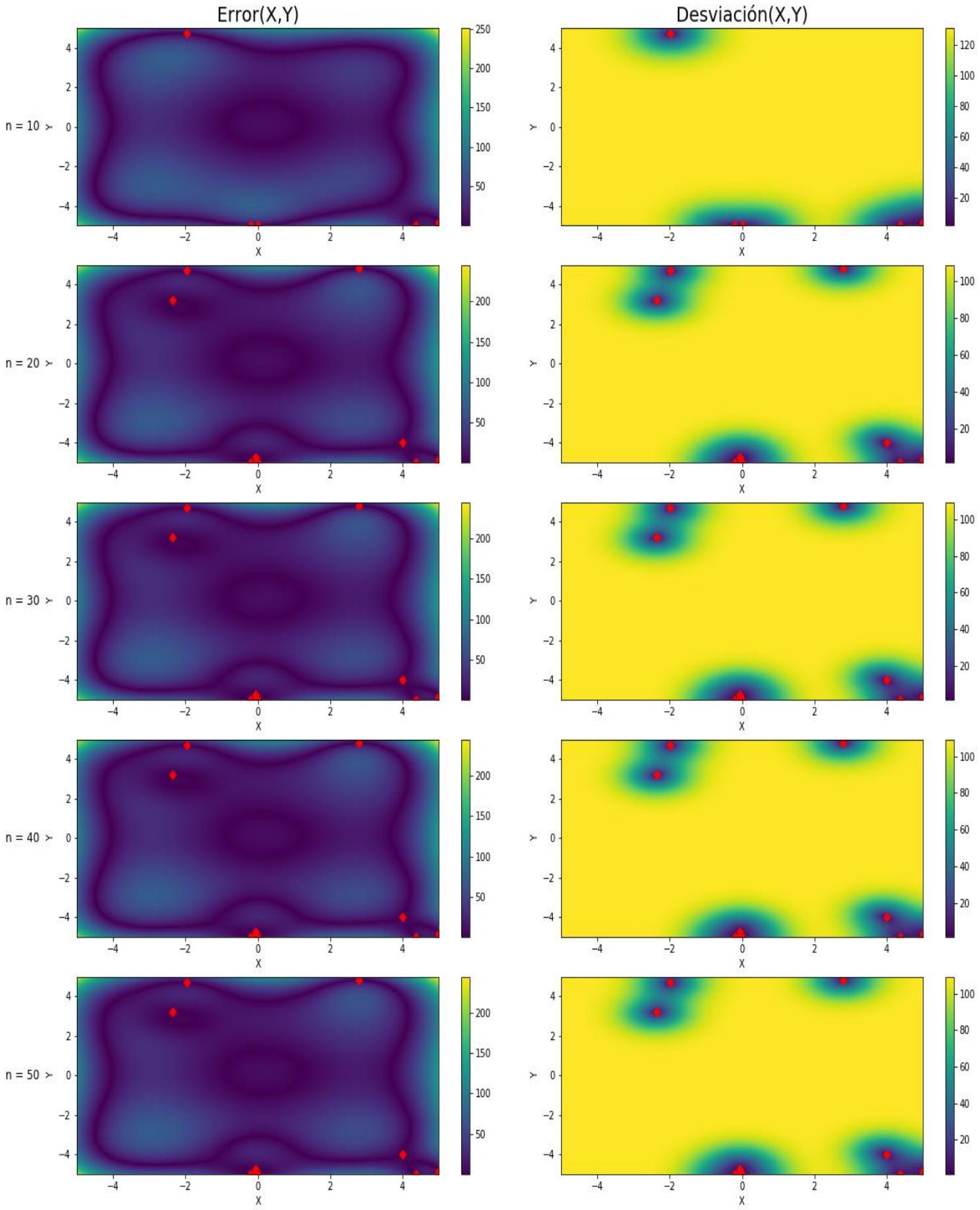

Figura 4-34. Evolución de los mapas de error y precisión durante el experimento 15

Experimento 16: Styblinski-Tang - Màtern kernel - Probability of Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	56,97751	148,7337
	20	47,73169	139,7696
	30	45,25832	131,4879
	40	44,73806	126,6398
	50	44,01831	124,1064
0.1	10	61,90941	134,459
	20	60,58832	95,04539
	30	35,30998	46,06423
	40	12,03501	20,92078
	50	7,438777	14,97738
0.2	10	87,03103	156,5593
	20	45,63212	99,26978
	30	18,66724	31,37681
	40	9,802387	17,47103
	50	6,119003	13,07478

Tabla 4-16. Resumen de resultados para el experimento 16

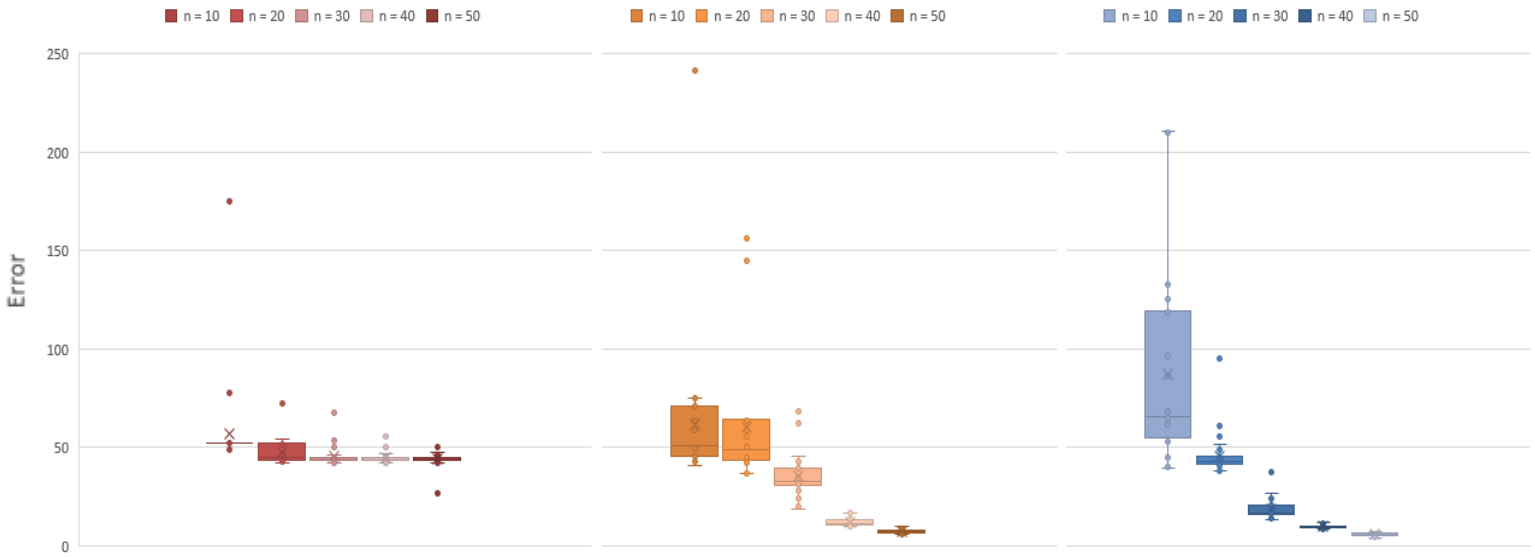

Figura 4-35. Resultados obtenidos para el experimento 16 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

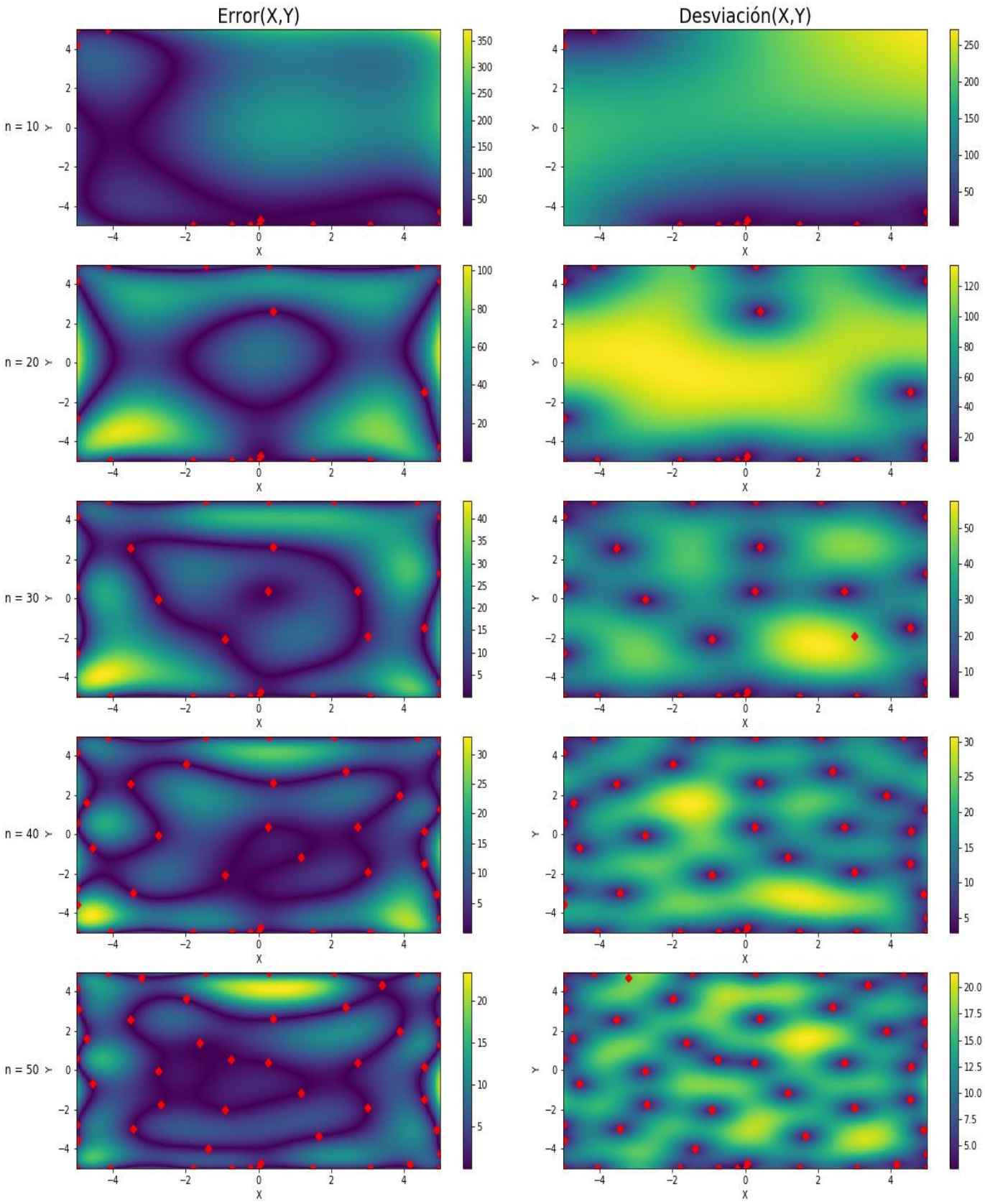

Figura 4-36. Evolución de los mapas de error y precisión durante el experimento 16

Experimento 17: Styblinski-Tang - Màtern kernel - Expected Improvement

ξ	n	e_{cm}	σ_{cm}
0.01	10	63,97136	105,2279
	20	50,32912	80,36717
	30	19,98814	29,80382
	40	9,719155	18,17172
	50	5,669146	13,45697
0.1	10	60,43697	106,3266
	20	49,20846	79,8255
	30	17,36313	27,2876
	40	9,11913	17,22617
	50	5,672913	12,92843
0.2	10	60,24561	108,7522
	20	48,9526	80,16455
	30	16,42493	26,31698
	40	9,058595	17,01159
	50	5,512996	12,74784

Tabla 4-17. Resumen de resultados para el experimento 17

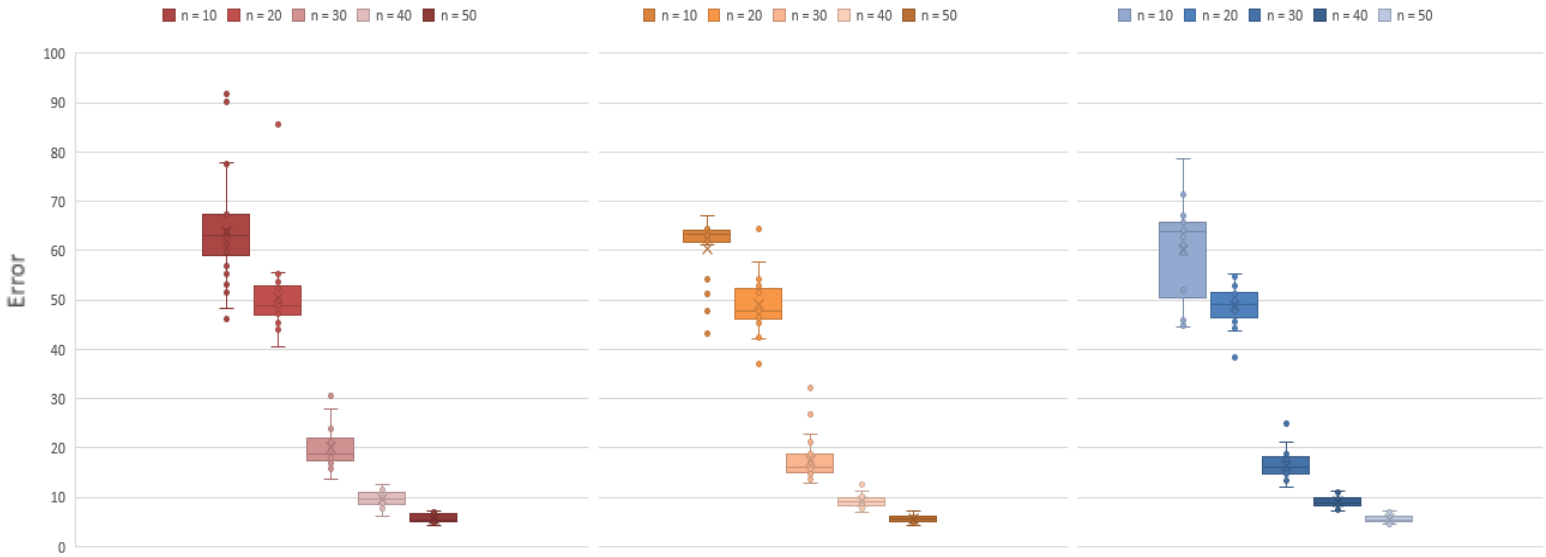

Figura 4-37. Resultados obtenidos para el experimento 17 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

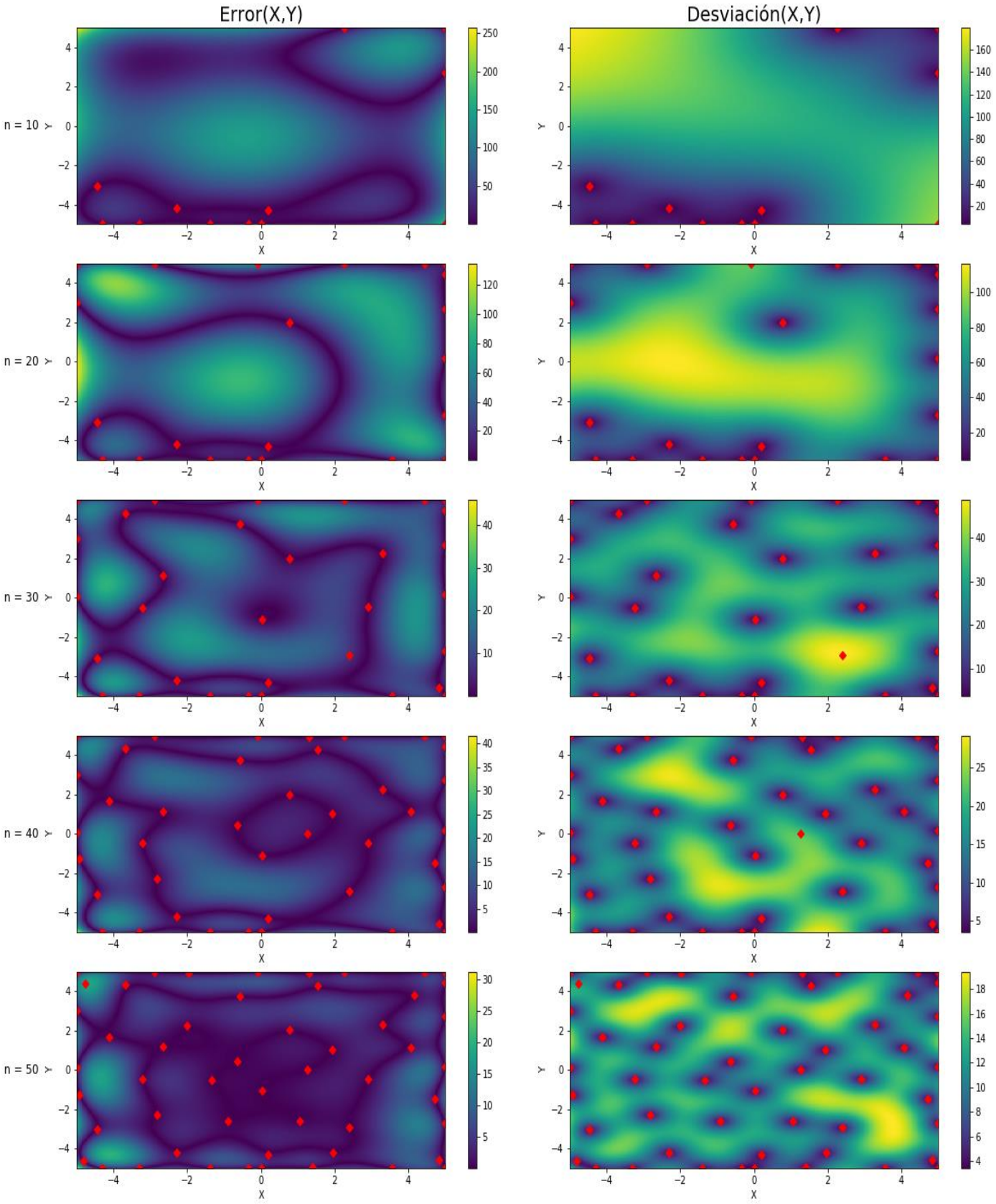

Figura 4-38. Evolución de los mapas de error y precisión durante el experimento 17

Experimento 18: Styblinski-Tang - Màtern kernel - Upper Confidence Bound

ξ	n	e_{cm}	σ_{cm}
0.01	10	46,3315	107,2135
	20	105,2183	23,47403
	30	75,16282	63,87796
	40	75,30469	63,42633
	50	75,35785	63,32431
0.1	10	130,0195	67,23454
	20	130,0185	67,03853
	30	130,0227	66,96965
	40	130,0276	66,9196
	50	130,0319	66,88169
0.2	10	80,25931	86,54647
	20	80,0187	86,53393
	30	79,81513	86,53393
	40	79,81609	86,55375
	50	79,8162	86,56698

Tabla 4-18. Resumen de resultados para el experimento 18

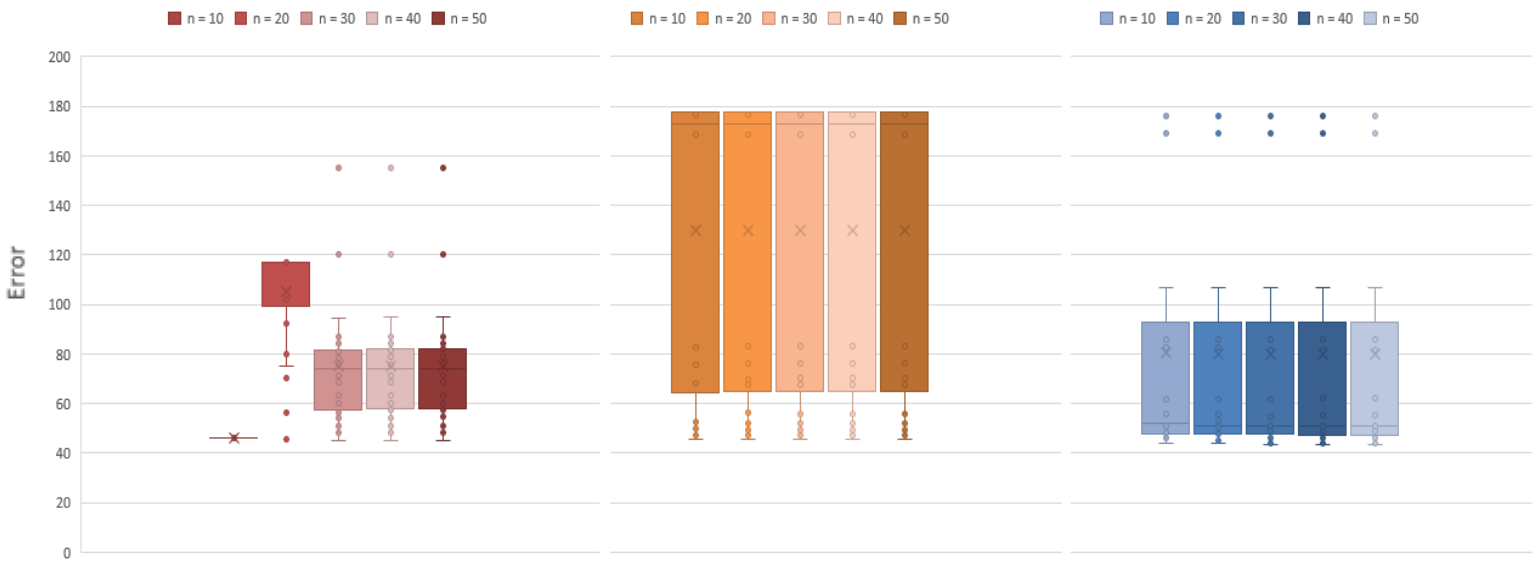

Figura 4-39. Resultados obtenidos para el experimento 18 a las *n* iteraciones. De izquierda a derecha: *ξ=0.01, ξ=0.1, ξ=0.2*

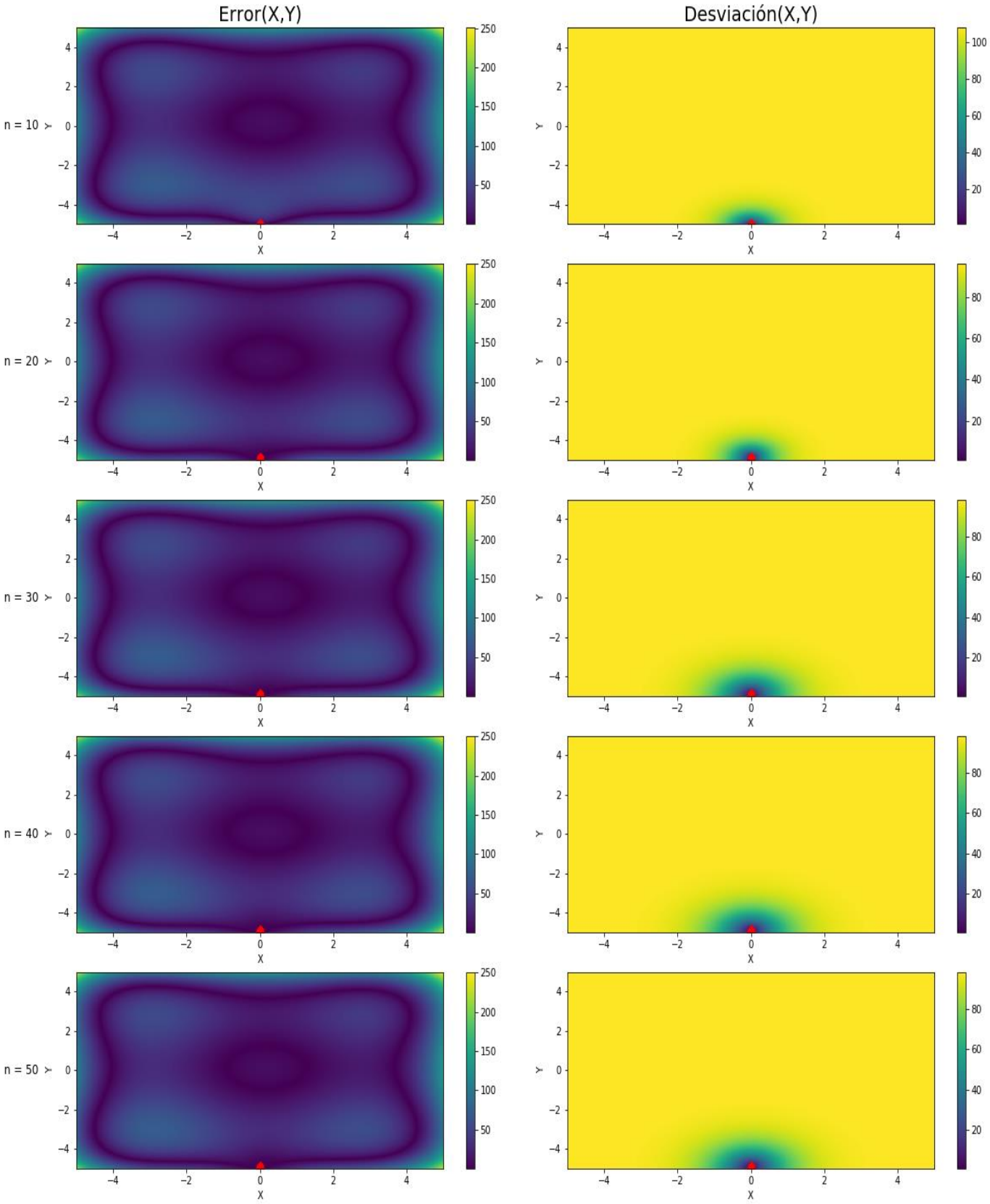

Figura 4-40. Evolución de los mapas de error y precisión durante el experimento 18

4.4 Discusión de los resultados

Con el objetivo de obtener conclusiones adecuadas respecto a las diferentes configuraciones para las que se han presentado los resultados anteriores, se realizará un análisis de estos en base a los diferentes elementos configurables que se han empleado -*test function*, *kernel*, función de adquisición y parámetro de *trade-off*- a modo de descenso desde el elemento más general al más concreto. En este recorrido será de gran ayuda la figura 4-41, que reúne los resultados genéricos de cada experimento a modo de resumen.

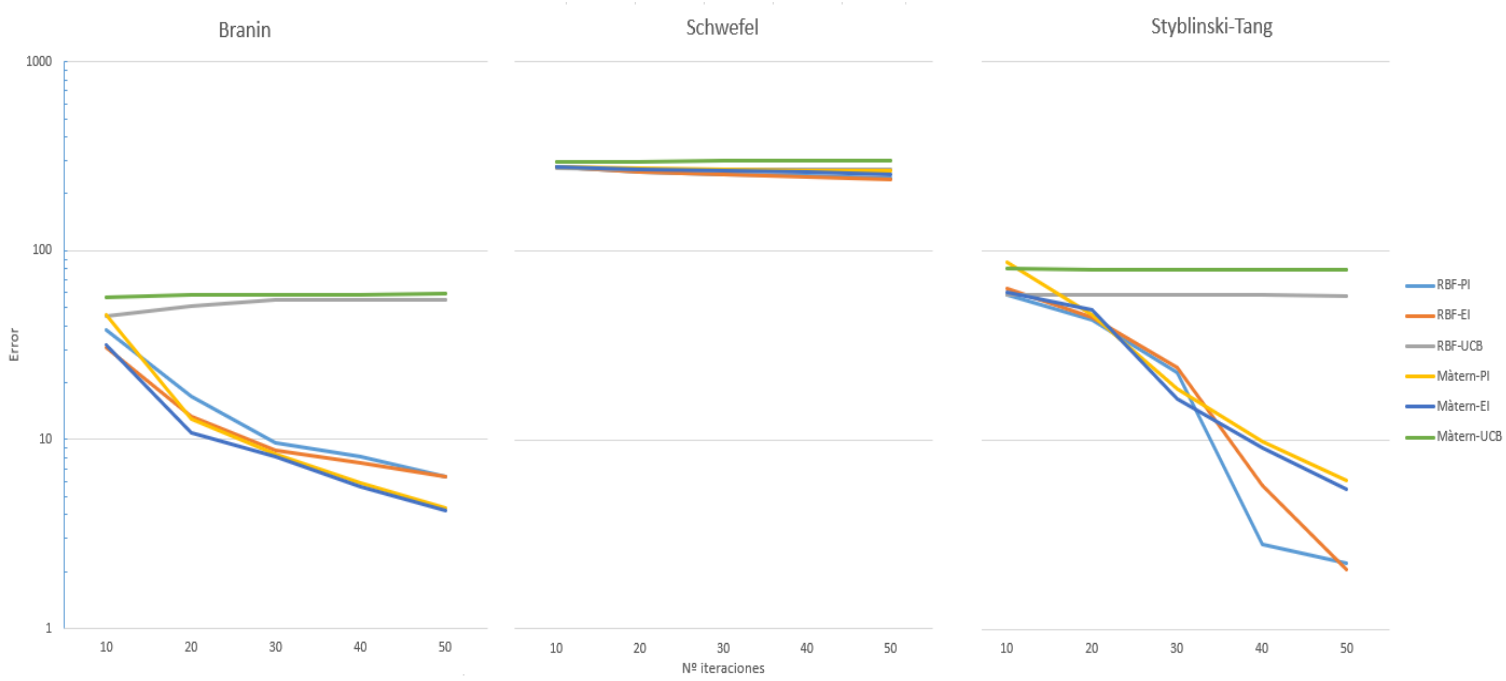

Figura 4-41. Resumen de los resultados de error para n iteraciones.

En primer lugar, los modelos obtenidos cuando se intenta aproximar la función de Schwefel no resultan acertados en ninguno de los casos propuestos: en todos ellos el error alcanza valores entre 200 y 300, lo que hace que los modelos resulten inadmisibles para la finalidad perseguida. Aún más, el incremento en el número de iteraciones no conlleva una mejora sustancial en el modelo (llegando incluso a empeorar el error en el caso del experimento 11) con una disminución del error de un 15% (medido entre 10 y 50 muestras) en el caso más favorable. Además, a medida que se toman más muestras también se aprecia un aumento en la incertidumbre del modelo, por lo que el conjunto de las dos tendencias descritas lleva a pensar que la calidad del modelo empeorará a medida que aumente el número de iteraciones.

Sin embargo, el panorama cambia cuando se pretenden aproximar las dos funciones de test restantes. En ambos casos se llega a obtener buenos resultados con todas las configuraciones: salvando aquellos casos en los que se emplea UCB como función de adquisición, en los que se incidirá más adelante, el error final obtenido más alto ronda un valor de 6, mientras que en los casos más favorables baja a 4 para la función de Branin y a 2 para la función de Styblinski-Tang. Además, en estos casos la evolución a medida que aumenta el número de muestras tomadas resulta mucho más favorable tanto para el error como para la incertidumbre del modelo, logrando disminuciones de error del 91% y del 96% en los casos para Branin y para Styblinski-Tang, respectivamente; en definitiva, el aumento en el número de iteraciones hace que que el modelo se vuelva más exacto y preciso.

La polarización de resultados entre la pareja formada por estas dos últimas funciones y la función de Schwefel se debe fundamentalmente a la elección del *kernel*; como ya se explicó en la sección 3, las consecuencias de esta decisión predominan en el resultado del algoritmo. Los *kernels* Mátern y RBF resultan ser más suaves de lo que lo es la función de Schwefel, lo que dificulta enormemente que el modelo pueda adaptarse a una función de este tipo. Además, el hecho de que dicha función presente tal número de máximos y mínimos (Figura 4-2) dificulta enormemente que un algoritmo cuyo próposito original es la optimización lleve a cabo un proceso de exploración más exahustivo, tendiendo a tomar muestras en zonas más concretas; esto impide la recopilación de información que le permita mejorar la predicción de la función. Por su parte, las funciones de Branin y de Styblinski-Tang resultan ser mucho más suaves y semejantes entre sí, lo que resulta coherente con la mejora y la similitud presentes en los resultados obtenidos en estos casos.

Cabe destacar que todos estos resultados no deben llevar a la conclusión de que la Optimización Bayesiana no es apta para el modelado de funciones como la de Schwefel. Podrían conseguirse buenos resultados en tanto que la elección del *kernel* empleado resulte más acorde con esta función. Sin embargo, en este trabajo no se ha querido seguir este camino dado que hacerlo entraría en conflicto con la realidad subyacente del caso estudiado: en la monitorización del lago Ypacaraí el modelo del lago se considera completaente desconocido, por lo que resulta de interés medir qué tan buenos resultados pueden obtenerse enfrentando un *kernel* predeterminado a diferentes entornos, y no a la inversa.

Sin ánimo de incidir más sobre la importancia del *kernel* en el desempeño del algoritmo, puede apreciarse que en el caso de la función de Schwefel esta elección no juega un papel muy relevante, mientras que los dos casos restantes si presentan ciertas diferencias. Si bien en el caso de la función de Branin se obtienen mejores resultados cuando se emplea el *kernel Màtern* (con un valor del error de 4 frente a 6 si se emplea el RBF), al intentar modelar la función de Styblinski-Tang es preferible usar el *kernel* RBF (obteniendo un error de 2 frente al 6 del *Màtern*). En todos los demás casos se comprueba que el uso del RBF resulta favorable frente al uso del *Màtern*.

Esta pequeña contraposición de resultados entre los experimentos con las funciones de Styblinski-Tang y de Branin se debe, una vez más, al grado de adecuación del *kernel* a la función de test. En el primer caso, la función presenta un comportamiento más similar al exponencial, principalmente en zonas cercanas a la frontera definida, por lo que es lógico que el RBF obtenga mejores resultados. Sin embargo, la función de Branin se aleja ligeramente de este comportamiento, lo que hace que este *kernel* empeore su desempeño siendo superado por el *Màtern* (más flexible).

Independientemente del *kernel* utilizado, la gráfica resumen anterior permite comprobar que, en líneas generales, los resultados obtenidos al emplear las funciones de adquisición PI y EI no presentan una diferencia sustancial entre ellos, mientras que sí difieren con respecto a los datos obtenidos al utilizar UCB, siendo estos resultados significativamente peores y apenas variantes con el número de muestras tomadas. Esto se debe a que la función de adquisición UCB tiene un enfoque de explotación pura, haciendo que el muestreo se realice de manera más focalizada (como puede comprobarse en la evolución de los mapas para estos casos), con la consecuente falta de información debido a una exploración deficiente.

En el caso de las funciones de Branin y de Styblinski-Tang, que resultan más favorables para la exploración, el error final alcanzado aumenta un orden de magnitud cuando se pasa de usar PI o EI a usar UCB como funciones de adquisición. El caso de la función de Schwefel resulta algo más particular, ya que su abundancia de máximos y mínimos difculta enormemente la exploración. Si bien puede comprobarse en las gráficas presentadas anteriormente cómo se mantiene la tendencia de mayor exploración al usar PI o EI, las muestras suelen tomarse en los puntos donde la función toma valores extremos, por lo que el error cometido por el modelo no se reduce notablemente con respecto al alcanzado al usar UCB.

Por tanto, queda demostrada la importancia de una correcta exploración para conseguir buenos resultados; de ahí la relevancia del balance entre la tendencia innata del algoritmo hacia la explotación y la búsqueda necesaria de la exploración, en el que el parámetro de *trade-off* ξ toma relevancia. La Figura 4-40 se ha construido tomando los resultados más favorables de cada caso, omitiendo la información relativa a la variación del parámetro ξ pero que sí está presente en las tablas de resultados expuestas anteriormente.

Por una parte, puede comprobarse que, aunque se ha dicho que las funciones de adquisición PI y EI no presentan una diferencia sustancial en cuanto a los resultados obtenidos, sí que lo hacen para el valor más bajo del parámetro de *trade-off*, es decir, cuando este apenas tiene efecto sobre las propias funciones. En estos casos se aprecia la mejora proporcionada por la función EI frente a PI, consiguiendo una diminución del error del final de hasta el 78% en el caso de la función de Branin y de un 95% para la función de Styblinski-Tang. Por otro lado, el efecto de este parámetro sobre la función UCB puede apreciarse en los mapas de evolución, teniendo lugar una dispersión de puntos de muestreo mayor a medida que el valor del parámetro aumenta: valores mayores favorecen la exploración, contrarrestando en cierta medida la fuerte tendencia de explotación de la función UCB.

5 IMPLEMENTACIÓN REAL

Hasta el momento se ha llevado a cabo un estudio y análisis de la Optimización Bayesiana como técnica de modelado de un entorno desconocido. Es importante conocer el método y su alcance, pero también lo es conocer su implementación en el caso práctico real, por lo que resulta importante llevar a cabo una breve descripción de cuál es concretamente el papel que juega la Optimización Bayesiana en el sistema del ASV [28].

El sistema que gobierna el movimiento del vehículo se trata de un GNC (*Guidance, Navigation and Control*), cuyo esquema queda representado en la Figura 5-1. Dentro de este conjunto, la Optimización Bayesiana pertenece al bloque de orientación, encargado de decidir el punto de destino y la ruta a seguir hasta este. A grandes rasgos, el sistema presenta un comportamiento cíclico en el que el sistema de navegación y los sensores aportarán al sistema de orientación la información necesaria para decidir el movimiento que será ejecutado por el sistema de control del vehículo.

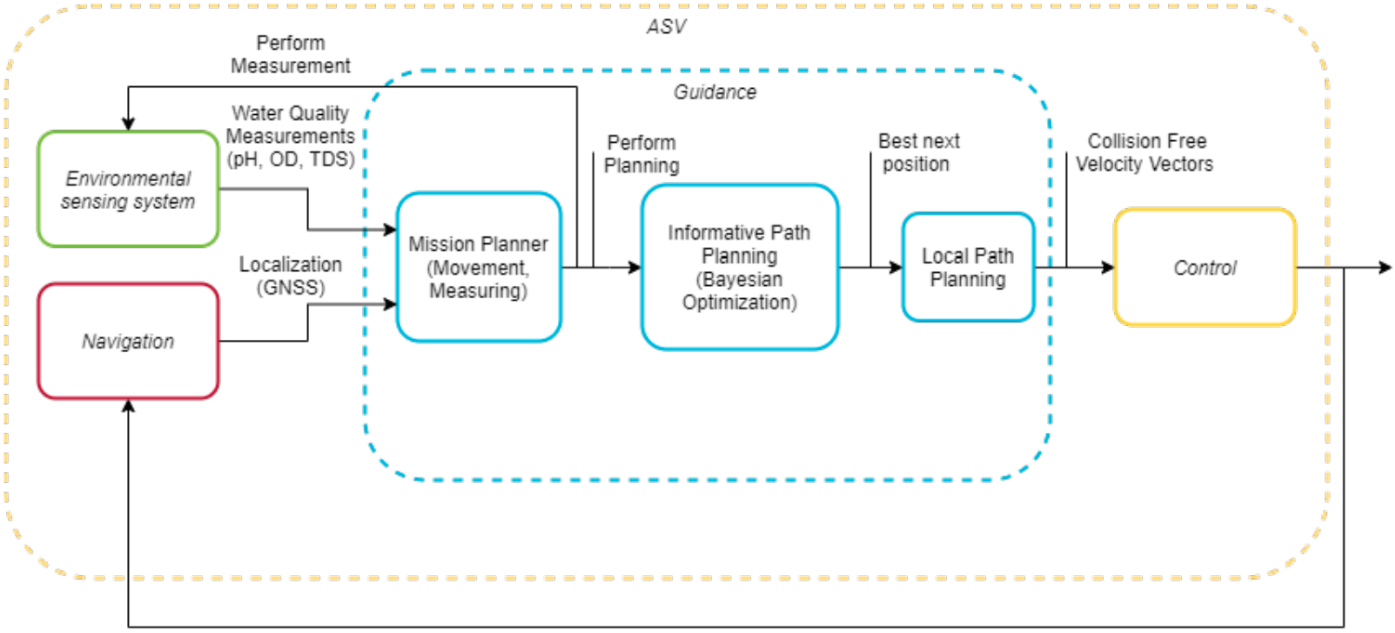

Figura 5-1. Esquema del sistema GNC del ASV. Fuente: [28]

El sistema de posicionamiento deberá tener una precisión aproximadamente igual a la resolución del mapa que se haya definido para el entorno, de manera que en todo momento el vehículo quede localizado en uno de los puntos que conforman dicho mapa. Por otra parte, el sistema de orientación debe incluir un algoritmo de planificación de trayectoria local que que lleve a cabo la evasión de obstáculos mientras el vehículo recorre el camino descrito por el algoritmo de planificación global, en este caso la Optimización Bayesiana.

6 CONCLUSIONES Y TRABAJO FUTURO

6.1 Conclusiones

Durante el desarrollo de este trabajo se han llevado a cabo varias tareas en torno a la Optimización Bayesiana y su aplicación al proyecto de monitorización de aguas del lago Ypacaraí.

En primer lugar, se presentó la problemática existente que da lugar al proyecto. Tras un análisis del problema, se concluyó que la Optimización Bayesiana resultaba un método viable para alcanzar una solución al problema propuesto. Seguidamente se hizo una revisión tanto del estado del propio proyecto -llegando a la conclusión de que el uso de esta técnica resulta innovador dentro del mismo- como del uso de la Optimización Bayesiana en otros trabajos, tras lo que se comprobó que, si bien la aplicación principal de esta técnica no es la monitorización medioambiental, sí que existen trabajos que la emplean con este fin, reforzando la idea de que sea un método admisible de resolución del problema.

Tras esto, se llevó a cabo el estudio teórico de la técnica con la idea de realizar un análisis de su eficacia posteriormente. Este estudio permitió adaptar la Optimización Bayesiana al caso real de trabajo mediante su particularización bajo diferentes condiciones, y realizar diferentes simulaciones con las que se obtuvieron un amplio conjunto de datos que permitiron discernir los puntos fuertes y débiles del método.

Como conclusión principal se determinó que la obtención de unos resultados buenos pasa por una correcta elección del *kernel*. Si bien en el caso real el modelo del lago es desconocido, sí que se podrían tener en cuenta ciertas hipótesis sobre el mismo que permitan conseguir mayor éxito en la elección del *kernel*. Bajo esta premisa, la Optimización Bayesiana resulta ser un método eficaz para lograr el objetivo perseguido, como demuestran los buenos resultados obtenidos.

6.2 Trabajo futuro

Si bien los resultados obtenidos han sido buenos, no son pocas las líneas de posible desarrollo para mejorar el funcionamiento de la técnica sobre el problema real estudiado.

Durante el trabajo se ha considerado que únicamente se quería modelar una variable del lago. Sin embargo, dado que el ASV tiene la capacidad de tomar medida de diferentes variables de manera simultánea, es plausible que se quiera obtener un modelado de cada una de esas variables. Por ello sería conveniente trabajar en la extensión del algoritmo a una forma multivariable, en la que se calculen varios modelos simultáneamente y se tengan en cuenta todos ellos a la hora de elegir el siguiente punto a muestrear.

Otras de las hipótesis consideradas durante el desarrollo del trabajo ha sido que el ASV únicamente tomará muestras en el punto de destino. En la realidad, esto carece completamente de sentido, puesto que lo más recomendable de cara a la eficiencia sería aprovechar el trayecto entre puntos para obtener más muestras, consiguiendo así más información haciendo el mismo gasto de recursos. Es decir, deberían estudiarse las posibilidades de un muestreo "cuasi-continuo" durante el desplazamiento. Esto abriría nuevas posibilidades, como la planificación en tiempo real de la trayectoria a seguir -y no al final de cada trayecto-, o la posibilidad de definir una función de adquisición que tenga en cuenta no solo la información que se obtendría en el punto de destino, sino el cómputo completo de todo el recorrido que se realizaría hasta este.

Por otro lado, se ha considerado el caso de un único vehículo realizando la tarea. Sin embargo, el uso de una flota de n vehículos conllevaría grandes mejoras en el rendimiento de la técnica, ya que permitiría cubrir una mayor superficie del lago en el mismo tiempo. Para ello, se debería estudiar no solo la adaptación del algoritmo para obtener n puntos a muestrear en la siguiente iteración, sino también la coordinación de movilidad entre los diferentes vehículos que forman la flota.

Por último, resulta interesante contemplar nuevas restricciones que acerquen la aplicación de la Optimización Bayesiana aún más al caso real, como imponer restricciones de movilidad debido al contorno del lago o diferentes obstáculos presente en este, o tener en cuenta el tiempo transcurrido desde la última visita a los puntos muestreados a la hora de determinar el siguiente punto al que dirigir el vehículo.

BIBLIOGRAFÍA

[1] Centro Internacional de Hidroinformática (CIH), Sistema de Monitoreo, Control y Estudios de la Cuenca del Lago Ypacaraí, 2016.

[En línea] Disponible: http://hidroinformatica.itaipu.gov.py/gestiondecuenca/py/ypacarai/

[Último acceso: 4 de Diciembre de 2020]

[2] M. García, "Eutrofización: una visión general", 26 de Septiembre de 2016.

[En línea] Disponible: http://www.cienciacierta.uadec.mx/2016/09/26/eutrofizacion-una-vision-general/

[Último acceso: 4 de Diciembre de 2020]

[3] S. Toral, D. Gutiérrez, "Formación en Ingeniería y Cooperación Internacional: Diseño de drones acuáticos para monitorización de variables medioambientales", Mayo 2020.

[4] S. Chatterjee, C. Carrera, L. Lynch, "Genetic algorithms and traveling salesman problems", Octubre 1994.

[5] R. Masià, J. Pujol, J. Rifà, M. Villanueva, "Grafos eulerianos y grafos hamiltonianos".

[6] M. Arzamendia, “Reactive Evolutionary Path Planning For Autonomous Surface Vehicles in Lake Environments,” December, 2018.

[7] D. Gutiérrez, M. Arzamendia and D. Gregor, "Evolutionary Computation for Path Planning of Autonomous Surface Vehicles using Eulearian Graphs", July, 2018.

[8] D. Gutiérrez, M. Arzamendia and D. Gregor, "An Evolutionary Approach to Constrained Path Planning of an Autonomous Surface Vehicle for Maximizing the Covered Area of Ypacarai Lake", October, 2017.

[9] M. Gestal, D. Rivero, J. Rabuñal, J. Dorado, A. Pazos, "Introducción a los Algoritmos Genéticos y la Programación Genética", Universidad da Coruña, 2010.

[10] D. Gutiérrez, M. Arzamendia, D. Gregor, S. Toral, E. Asimakopoulou and N. Bessis, "Intelligent Online Learning Strategy for an Autonomous Surface Vehicle in Lake Environments using Evolutionary Computation".

[11] S. Yanes, "Aplicaciones de técnicas de Deep Reinforcement Learning a misiones de exploración para vehículos autónomos en escenarios lacustres", 2020.

[12] P. Frazier, "A Tutorial on Bayesian Optimization", July, 2018.

[13] B. Shahriari, K. Swersky, Z. Wang, R. Adams and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization".

[14] E. Brochu, V. Cora and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning", December, 2010.

[15] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, Md. Patwary, Prabhat, R. Adams, "Scalable Bayesian Optimization Using Deep Neural Networks".

[16] D. Fernández, "Creación de una Herramienta de Optimización Bayesiana en Python", Junio, 2019.

[17] J. García, "Optimización Bayesiana aplicada a la simulación de fluidos", Septiembre, 2015.

[18] D. Packwood, "Bayesian Optimization for Materials Science", Agosto 2017.

[19] J. Snoek, H. Larochelle, R. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms".

[20] E. Brochu, T. Brochu, N. de Freitas, "A Bayesian Interactive Optimization Approach to Procedural Animation Design", 2010.

[21] R. Marchant and F. Ramos, "Bayesian Optimization for Informative Continuous Path Planning", Junio, 2014.

[22] S. Bai, J. Wang, F. Chen and B. Englot, "Information-Theoretic Exploration with Bayesian Optimization", October, 2016.

[23] R. Martínez, N. de Freitas, A. Doucet, J. Castellanos, "Active Policy Learning for Robot Planning and Exploration under Uncertainty".

[24] R. Marchant, F. Ramos, "Bayesian Optimization for Intelligent Environmental Monitoring", Octubre 2012

[25] A. Singh, F. Ramos, H. Durrant, W. Kaiser, "Modeling and Decision Making in Spatio-Temporal Processes for Environmental Surveillance", Mayo 2010.

[26] Wikipedia, "Teorema de Bayes", 5 de Octubre de 2020. [En línea] Disponible: https://es.wikipedia.org/wiki/Teorema_de_Bayes [Último acceso: 4 de Diciembre de 2020]

[27] "Virtual Library of Simulation Experiments: Test Functions and Datasets". [En línea] Disponible: https://www.sfu.ca/~ssurjano/index.html [Último acceso: 4 de Diciembre de 2020].

[28] F. Peralta, D. Gutiérrez, S. Toral, M. Arzamendia, D. Gregor, "A Bayesian Optimization Approach for Water Resources Monitoring through an Autonomous Surface Vehicle: The Ypacarai Lake Case Study", 2020. [Unpublished]

[29] Distill, "A Visual Exploration of Gaussian Processes", 2019. [En línea] Disponible: https://distill.pub/2019/visual-exploration-gaussian-processes/ [Último acceso: 4 de Diciembre de 2020]

[30] Distill, "Exploring Bayesian Optimization", 2020. [En línea] Disponible: https://distill.pub/2020/bayesian-optimization/ [Último acceso: 4 de Diciembre de 2020]

[31] Scikit-Learn, User Guide. [En línea] Disponible: https://scikit-learn.org/stable/user_guide.html [Último acceso: 4 de Diciembre de 2020]

[32] C. Rasmussen, C. Williams, "Gaussian Processes for Machine Learning", 2006.

[33] T. Mitchell, "Machine Learning", 1997.

[34] F. Archetti, A. Candelieri, "Bayesian Optimization and Data Science".

[35] Numpy, Documentation. [En línea] Disponible: https://numpy.org/doc/stable/ [Último acceso: 4 de Diciembre de 2020]

[36] Matplotlib, Documentation. [En línea] Disponible: https://matplotlib.org/ [Último acceso: 4 de Diciembre de 2020]