TESIS CARRERA DE DOCTORADO EN FÍSICA

CRITERIOS BÁSICOS PARA LA PRESENTACIÓN DE LA TESIS EN EL INSTITUTO BALSEIRO TANTO DE DOCTORADO COMO DE MAESTRÍA

J. Autor
Doctorando

Dr. J. Director
Director

Dr. J. Otro más
Co-director

Miembros del Jurado
Dr. J. J. Jurado (Instituto Balseiro)
Dr. Segundo Jurado (Universidad Nacional de Cuyo)
Dr. J. Otro Jurado (Univ. Nac. de LaCalle)
Dr. J. López Jurado (Univ. Nac. de Mar del Plata)
Dr. U. Amigo (Instituto Balseiro, Centro Atómico Bariloche)

12 de Diciembre de 2022

Colisiones Atómicas – Centro Atómico Bariloche

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

A mi familia
A mis amigos
A todos los que me conocen
A toda esa otra gente que no

Índice de símbolos

Índice de contenidos

Resumen

Este es el resumen en castellano.
La tesis debe reflejar el trabajo desarrollado, mostrando la metodología utilizada, los resultados obtenidos y las conclusiones que pueden inferirse de dichos resultados.

Palabras clave: FORMATO DE TESIS, LINEAMIENTOS DE ESCRITURA, INSTITUTO BALSEIRO

Abstract

This is the title in English:
The thesis must reflect the work of the student, including the chosen methodology, the results and the conclusions that those results allow us to draw.

Keywords: THESIS FORMAT, TEMPLATES, INSTITUTO BALSEIRO

Capítulo 1

Marco Teórico

1.1. Bases Reducidas

1.2. Optimización Bayesiana

Capítulo 2

Metodología Utilizada

2.1. Datos utilizados

Se utilizaron ondas gravitacionales generadas a partir del modelo híbrido *NRHybSur3dq8*[1] de relativdad numérica y aproximaciones post Newtonianas para colisiones de agujeros negros binarios.

Cada onda h generada se representa por una serie temporal compleja de la forma:

$$h = h_+ + ih_\times$$

Recordando que h está parametrizada por λ

$$h = h(t;\lambda) = h_\lambda(t) = h_\lambda$$

En este caso λ tendrá 3 dimensiones, $(q, \chi_{1_Z}, \chi_{2_Z})$, y estará acotado de la siguiente manera:

- Relación entre masas q: $1 \leq q \leq 8$
- Espín del agujero negro más pesado (liviano) $\chi_{1_Z}(\chi_{1_Z})$: $|\chi_{1_Z}|, |\chi_{2_Z}| < 0{,}8$

Se representa un conjunto $\mathcal{K}$ de N muestras de λ de la siguiente forma:

$$\mathcal{K} = \{h_{\lambda_i}\}, \quad i = 1, ..., N$$

y debido a que cada h_{λ_i} es una serie temporal, se puede representar $\mathcal{K}$ en forma de la matriz $H \in \mathbb{C}^{N\times L}$:

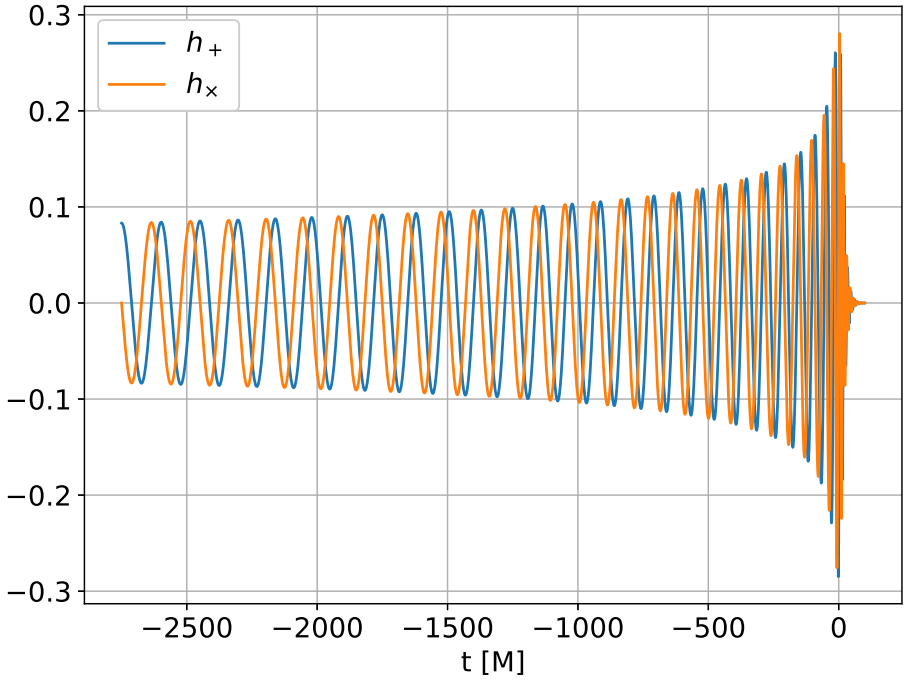

Figura 2.1: polarizaciones h_+ y $h_\times$ para $q = 3$, $\chi_{1z} = \chi_{2z} = 0$, en el modo $l = 2$, $m = 2$.

$$H = \begin{bmatrix} h_{\lambda_1} \\ h_{\lambda_2} \\ \vdots \\ h_{\lambda_N} \end{bmatrix} = \begin{bmatrix} h_{\lambda_1}(t_1) & h_{\lambda_1}(t_2) & \cdots & h_{\lambda_1}(t_L) \\ h_{\lambda_2}(t_1) & h_{\lambda_2}(t_2) & \cdots & h_{\lambda_2}(t_L) \\ \vdots & \vdots & \ddots & \vdots \\ h_{\lambda_N}(t_1) & h_{\lambda_N}(t_2) & \cdots & h_{\lambda_N}(t_L) \end{bmatrix}$$

Siendo L la longitud de la serie temporal. De forma que cada fila de H es una onda gravitacional.

2.2. Bases Reducidas hp Greedy

El nombre del método hp greedy viene de la combinación del *"refinamiento p"* y del *"refinamiento h"*. El *refinamiento p* proviene de los métodos espectrales con bases polinomiales [2] y se refiere a la propiedad de que el error de representación disminuye al aumentar el grado del polinomio (en el caso de las bases reducidas aumenta el número de elementos en la base). Por otro lado el término de *refinamiento h* se toma prestado de los métodos de diferencias finitas, donde el tamaño de cada celda de la grilla es representado por h. En este caso el refinamiento ocurre en el espacio de los parámetros (y no en el dominio físico).

2.2.1. Refinamiento h

Partiendo de la siguiente notación:

- V: espacio de parámetros para un dado subdominio D.
- V_1, V_2 : particiones de V.
- Λ_V : parámetros *greedy* para V.
- $\hat{\Lambda}_V$: punto de anclaje para V.

El refinamiento en el dominio de los parámetros ocurre a partir de la división recursiva de cada subdominio V del dominio total D en dos subdominios V_1 y V_2. De forma que se obtiene una estructura de árbol binario.

Esta descomposición binaria del dominio está descrita en forma de pseudocódigo en el algoritmo 1.

Al algoritmo ingresan tres objetos:

- λ_V: conjunto de parámetros resultado de un muestreo de V.
- $\hat{\Lambda}_{V_1}, \hat{\Lambda}_{V_2}$: puntos de anclaje (son los primeros dos elementos de Λ_V).

Luego, para cada parámetro del conjunto λ_V se evalúa su distancia a los puntos de anclaje a partir de la *función de proximidad* $d : d(\lambda_1, \lambda_2)$:

$$d(\lambda_1, \lambda_2) = ||\lambda_1 - \lambda_2||_2,$$

de forma que se obtengan dos conjuntos; λ_{V_1} con los λ_i más proximos a $\hat{\Lambda}_{V_1}$, y λ_{V_2} con los λ_i más proximos a $\hat{\Lambda}_{V_2}$, tal que $\lambda_V = \lambda_{V_1} \cup \lambda_{V_2}$. Este resultado es la división del espacio de parámetros a partir de los puntos de anclaje.

Algoritmo 1 `Partition`$(\lambda_V, \hat{\Lambda}_{V_1}, \hat{\Lambda}_{V_2})$

Input: $\lambda_V, \hat{\Lambda}_{V_1}, \hat{\Lambda}_{V_2}$

$\lambda_{V_1} = \lambda_{V_2} = \emptyset$
for each $\lambda_i \in \lambda_V$ **do**
    if $d(\lambda_i, \hat{\Lambda}_{V_1}) < d(\lambda_i, \hat{\Lambda}_{V_2})$ **then**
        $\lambda_{V_1} = \lambda_{V_1} \cup \lambda_i$
    else if $d(\lambda_i, \hat{\Lambda}_{V_1}) > d(\lambda_i, \hat{\Lambda}_{V_2})$ **then**
        $\lambda_{V_2} = \lambda_{V_2} \cup \lambda_i$
    else
        $\lambda_{V'} = \text{random choice}([\lambda_{V_1}, \lambda_{V_2}])$
        $\lambda_{V'} = \lambda_{V'} \cup \lambda_i$
    end if
end for

Output: $\lambda_{V_1}, \lambda_{V_2}$

2.2.2. Refinamiento hp-greedy

El refinamiento hp-greedy es un método que combina el algoritmo greedy para la construcción de bases reducidas con la partición del dominio de parámetros.

Esta partición recursiva del dominio de parámetros da lugar a una estructura de árbol binario, la cual tendrá diferentes niveles l de profundidad, con un l_{max} establecido por el usuario, de forma que $l : 0 \leq l \leq l_{max}$, donde $l = 0$ es el nodo raíz. Cada nodo del árbol estará etiquetado por un conjunto de índices B_l, que parte de:

$$B_0 = (0,),$$

luego sus dos hijos ($l = 1$) tendrán las etiquetas:

$$B_1 = (0, 0,) \;\; o \;\; (0, 1,),$$

y en general:

$$B_l = (0, i_1, \cdots, i_l), \quad con \;\; i_j = \{0, 1\},$$

donde cada nivel l tendrá un máximo de 2^l nodos. Los nodos que no tengan hijos se llamarán nodos *hojas*.

El método está explicado en el algoritmo 2; partiendo de un dado dominio de parámetros V se construye una base reducida a partir de un conjunto de entrenamiento $\mathcal{T}_V = \{h_{\lambda_{V_i}}\}_{i=1}^{N}$, una *tolerancia greedy* ε y un n_{max} (para esto se utiliza el algoritmo [**VERIFICAR REFERENCIA**]). Si el error de representación σ es mayor que la tolerancia ε, y si la profundidad del nivel l es menor a l_{max}, entonces se realizará una partición del dominio V utilizando como puntos de anclaje a los dos primeros parámetros *greedy*. En cada dominio se realizará el mismo procedimiento hasta que se

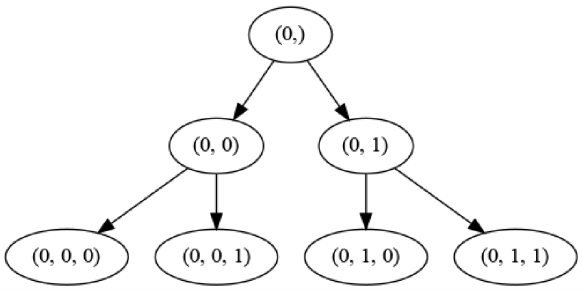

Figura 2.2: Representación de los nodos de un árbol con $l_{max} = 2$.

cumpla que $l = l_{max}$ o hasta que $\sigma \leq \varepsilon$.

Algoritmo 2 `hpGreedy`$(\mathcal{T}_V, \lambda_V, \varepsilon, n_{max}, l, l_{max}, B_l)$

Input: $\mathcal{T}_V, \lambda_V, \varepsilon, n_{max}, l, l_{max}, B_l$

1: rb, Λ_V, σ = `GreedyRB`$(\mathcal{T}_V, \lambda_V, \varepsilon, n_{max})$
2: **if** $\sigma > \varepsilon$ **and** $l < l_{max}$ **then**
3: $\quad \hat{\Lambda}_{V_1} = \Lambda_V[1]$
4: $\quad \hat{\Lambda}_{V_2} = \Lambda_V[2]$
5: $\quad \lambda_{V_1}, \lambda_{V_2}$ = `Partition`$(\lambda_V, \hat{\Lambda}_{V_1}, \hat{\Lambda}_{V_2})$
6: $\quad out_1$ = `hpGreedy`$(\mathcal{T}_{V_1}, \lambda_{V_1}, \varepsilon, n_{max}, l+1, l_{max}, (B_l, 0))$
7: $\quad out_2$ = `hpGreedy`$(\mathcal{T}_{V_2}, \lambda_{V_2}, \varepsilon, n_{max}, l+1, l_{max}, (B_l, 1))$
8: $\quad out = out_1 \cup out_2$
9: **else**
10: $\quad out = \{(rb, \Lambda_V, B_l)\}$
11: **end if**

Output: out

El resultado del algoritmo 2 es una estructura arbórea, donde cada nodo contiene la información de sus puntos de anclaje, por lo que en el caso de querer proyectar un conjunto de validación, cada onda gravitacional se proyectará a la base reducida del nodo hoja con el punto de anclaje más cercano al parámetro de la onda.

2.2.3. Aplicación a Ondas Gravitacionales

Se trabaja a partir de un conjunto de ondas gravitacionales con parámetro bidimensional, donde $\chi_{1_z} = \chi_{2_z} = \chi_z$, es decir que $\lambda = (q, \chi_z)$. De esta forma se puede graficar fácilmente el dominio de parámetros.

En la figura 2.3 se puede observar una representación de la partición del dominio de parámetros. En la primera imagen se pueden ver los dos puntos de anclaje, que son los primeros dos elementos de la base global construida inicialmente. En cada nueva división se construye una nueva base global con la cual se realiza la siguiente partición de cada subdominio.

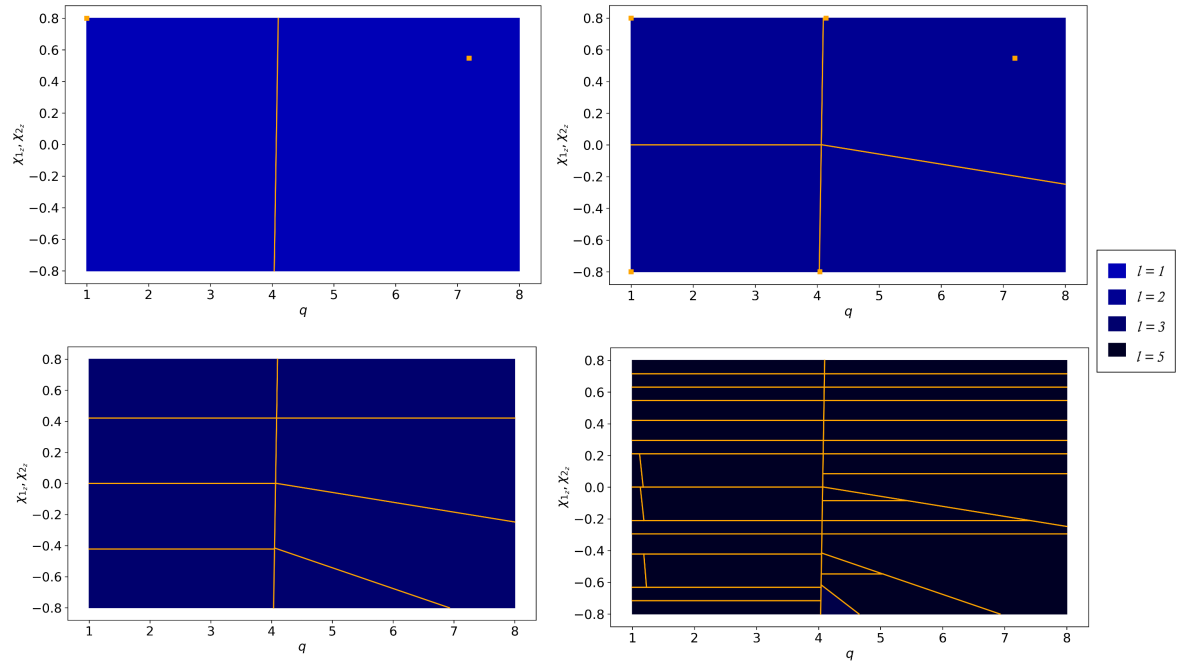

Figura 2.3: Ejemplo de partición del espacio de parámetros bidimensional para $l_{max} = 1, 2, 3$ y 5. En los primeros dos casos se muestran los puntos de anclaje.

En la figure 2.4 se compara el máximo error de representación obtenido para un conjunto de validación con una base global, es decir, con $l_{max} = 0$, y con $l_{max} = 4$. La velocidad de convergencia es claramente mayor en el segundo caso.

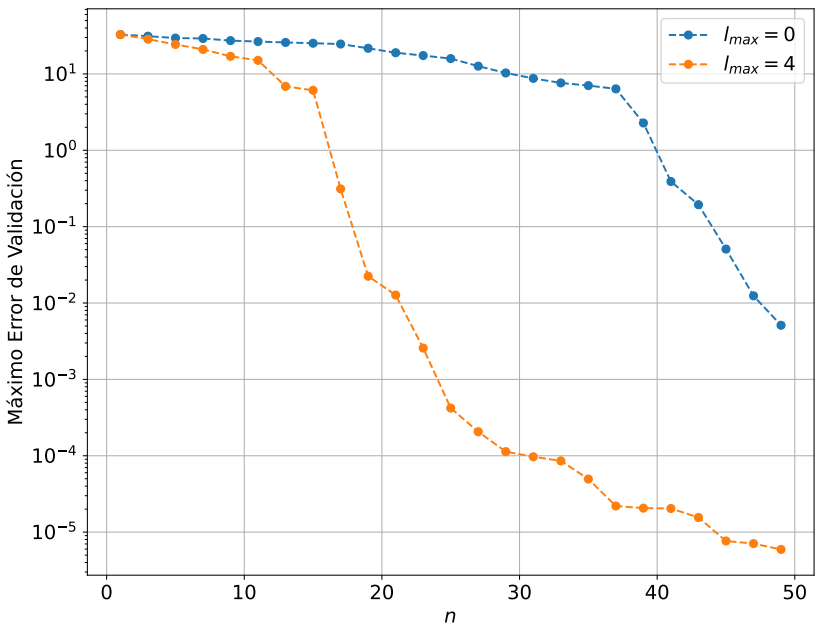

Figura 2.4: Base global ($l_{max} = 0$) versus base con $l_{max} = 4$ para distintos valores de n.

El aspecto más importante de este método es que permite disminuir la complejidad

temporal del algoritmo a la hora de proyectar la base, a cambio de aumentar la complejidad espacial, pues si bien cada subdominio tendrá un máximo de n_{max} elementos en su base, habrá un máximo de $2^{l_{max}}$ subdominios.

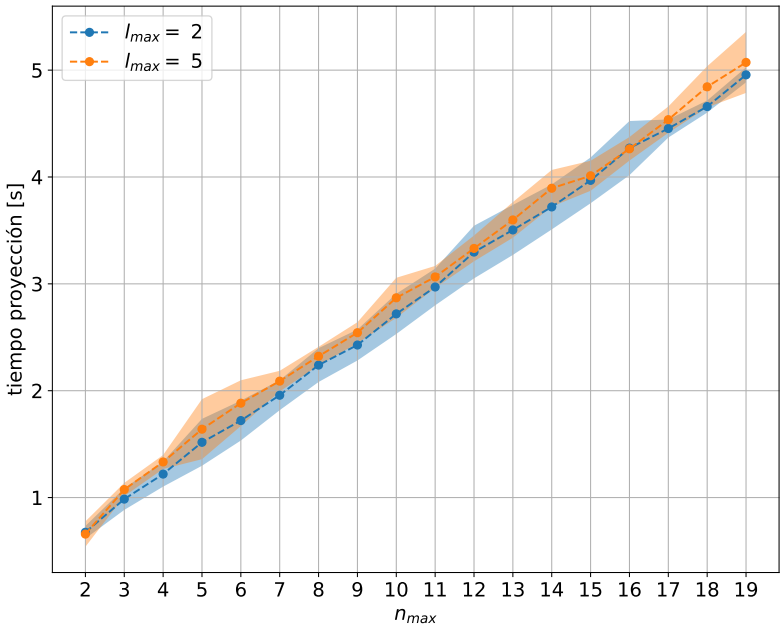

Figura 2.5: tiempos de proyección de un conjunto de validación a dos bases con distinto l_{max} en función del n_{max}. En cada caso la linea de trazo representa el valor medio, y el área de color indica una desviación estandar desde el valor medio, para cada medición.

En la figura 2.5 se graficó el tiempo de proyección de un conjunto de validación a dos bases *hp-greedy* con distinto valor de l_{max}. Se observa que el tiempo es bastante lineal en relación al n (elementos de las bases locales), y no parece ser afectado por l_{max}.

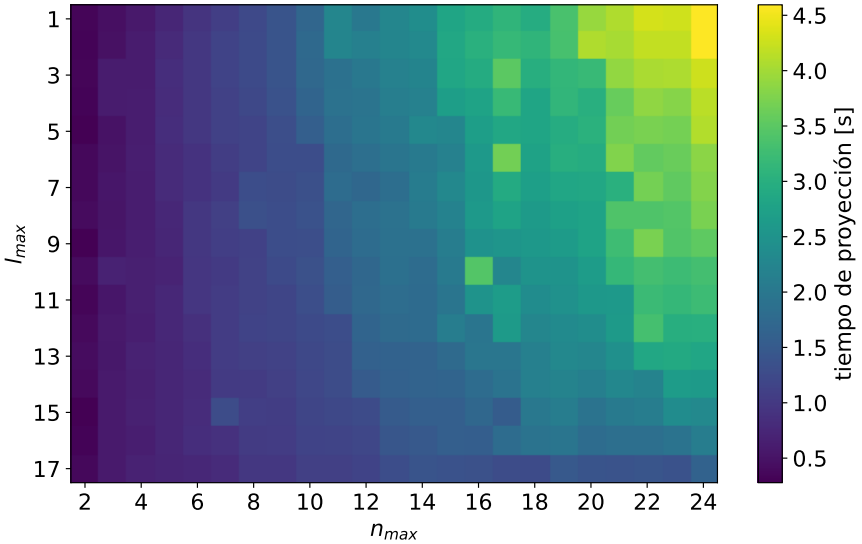

Figura 2.6: Tiempo de proyección de un conjunto de validación para diferentes valores de n_{max} y l_{max}

En la figura 2.6 se puede ver el tiempo de proyección para más valores de n_{max} y l_max. En los primeros valores de l_{max} se observa un comportamiento similar al descrito anteriormente, donde el tiempo depende casi únicamente de n_{max}. Sin embargo al aumentar el l_{max} se observa que el tiempo disminuye. Esto se puede entender en dos partes:

- **Independencia aparente entre el tiempo de proyección y** l_{max}: para realizar la proyección de cada onda del conjunto de validación en la base *hp-greedy* primero se debe buscar el subdominio (la hoja) correspondiente utilizando los puntos de anclaje de la estructura arbórea de la base. Luego se proyectará la onda en la base local del subdominio en cuestión. Si bien la búsqueda en el árbol tiene una complejidad temporal $O(l_{max})$, el trabajo de cómputo más importante es el que se realizará al momento de proyectar la base, que es independiente de l_{max}, con una complejidad temporal $O(n_{max})$. Pero esto solo se cumple hasta ciertos valores de l_{max}.

- **Disminución del tiempo de proyección al aumentar** l_{max}: ya se mencionó en más de una ocasión que por cada nivel l hay un máximo de 2^l subdomínios. Es decir que si se quiere obtener el número de elementos de todas las bases en las hojas del árbol, suponiendo un árbol denso, este número será $n_{max} \times 2^{l_{max}}$. En la figura 2.6 se utilizó un conjunto de entrenamiento con 1400 ondas, por lo que al llegar a unos valores de $l_{max} = 6$ y $n_{max} = 24$ en total debería haber 1536 elementos de base en total. Es decir, más elementos de base que ondas en el conjunto de entrenamiento. Por lo tanto al aumentar el l_{max} rápidamente se aumenta el número de subdominios, reduciendo su tamaño como resultado y reduciendo el número de elementos de las bases locales (cada subdominio tendrá una cantidad de elementos menor a n_{max}). De esta forma se explica la disminución del tiempo de proyección para valores grandes de l_{max}, consecuencia del tamaño limitado del conjunto de entrenamiento.

2.2.4. Hiperparámetros

Al momento de construir una base *hp-greedy* entran en juego cuatro hiperparámetros. Los primeros tres son los parámetros de parada;

- $\boldsymbol{n_{max}}$: determina la cantidad máxima de elementos para cada base local. A mayor cantidad de elementos el error de representación será menor, pero el tiempo requerido para proyectar un conjunto de validación a la base depende casi exclusivamente de este hiperparámetro.

- $\boldsymbol{l_{max}}$: determina la máxima profundidad de las hojas del árbol. En general al aumentar l_{max} disminuye el error de representación, pero valores muy elevados

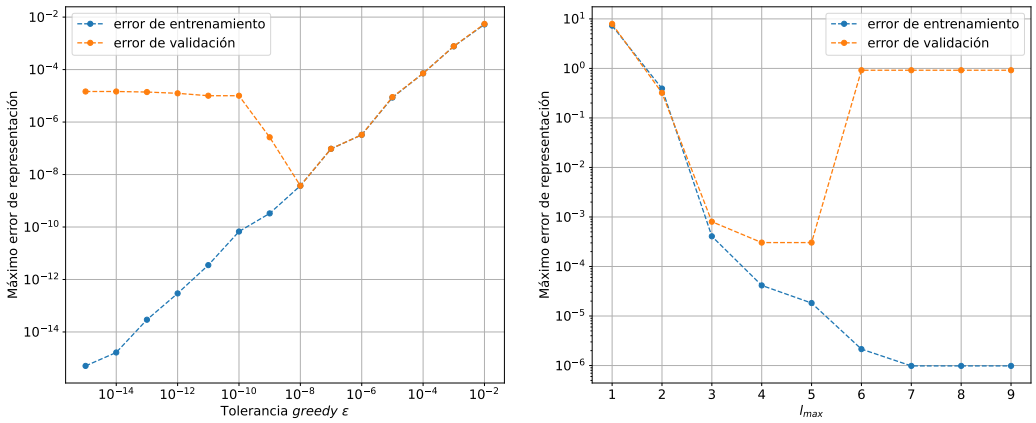

Figura 2.7: Ejemplos de *sobreajuste*. A la izquierda variando ε con $(n_{max}, l_{max}) = (25, 19)$ en un conjunto de parámetro unidimensional $(\lambda_i = q_i)$. A la derecha variando l_{max} con $(n_{max}, \varepsilon) = (20, 1 \times 10^{-6})$ en un conjunto de parámetro bidimensional$(\lambda_{ij} = (q_i, \chi_{z_j}))$.

junto a cierta combinación de hiperparámetros pueden dar lugar a sobreajustes en el modelo, un ejemplo de esto se puede ver en la figura 2.7. Este es un comportamiento típico de las estructuras arbóreas.

- $\boldsymbol{\varepsilon}$: la tolerancia *greedy* interviene tanto en el tamaño de las bases locales como en la profundidad de las hojas del árbol. Un valor de ε demasiado bajo también puede dar lugar a sobreajuste, sobre todo con valores muy altos de l_{max}. Un valor de $\varepsilon = 0$ implica que se obtiene un árbol totalmente denso, determinado únicamente por n_{max} y l_{max}, y al aumentar el valor de ε se puede pensar en la analogía de podar un árbol, de forma que se previene el sobreajuste.

Al cuarto hiperparámetro se le da el nombre de ***semilla*** y se la denota con $\hat{\boldsymbol{\Lambda}}_{\mathbf{0}}$;

- $\hat{\boldsymbol{\Lambda}}_{\mathbf{0}}$: la semnilla no es más que el primer parámetro *greedy* de la base global. En cada base local, el primer parámetro *greedy* no es relevante, pero en el caso de las bases *hp-greedy* cada semilla dará lugar a una división diferente del dominio de parámetros. En la figura 2.8 se puede ver como cuatro semillas diferentes dan lugar a cuatro curvas de error con distinta convergencia. En la figura 2.9, por otro lado, se observa el resultado de la partición del dominio para tres semillas diferentes. En general las semillas que mejor funcionan (con el conjunto de datos utilizado) son las que logran una partición regular del dominio de parámetros.

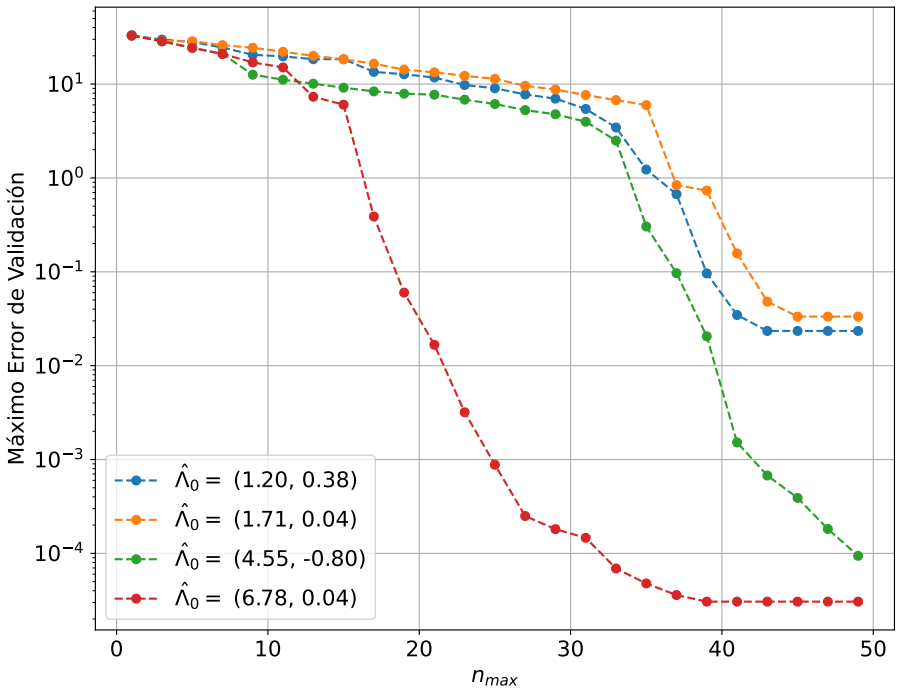

Figura 2.8: Error de validación para diferentes semillas.

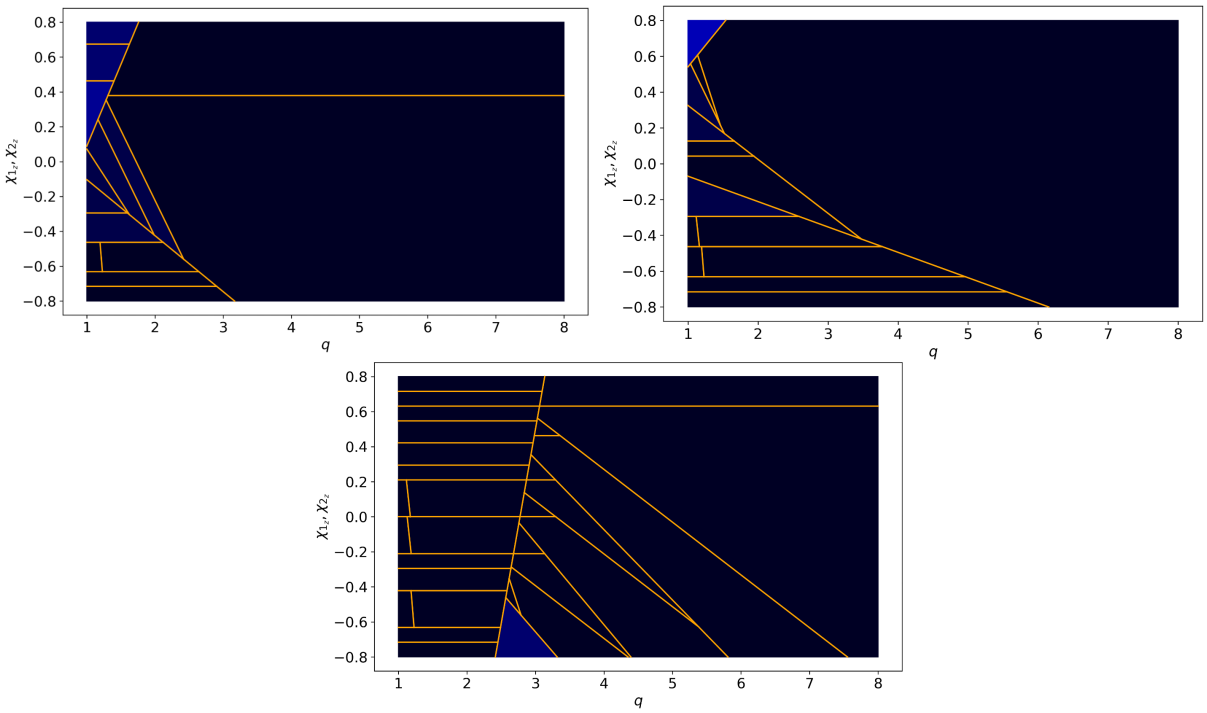

Figura 2.9: Partición del espacio de parámetros para tres semillas diferentes; a la izquierda $\hat{\Lambda}_0 = (1{,}2 \;\; 0{,}38)$, a la derecha $\hat{\Lambda}_0 = (1{,}71 \;\; 0{,}04)$ y al centro $\hat{\Lambda}_0 = (4{,}55 \;\; -0{,}8)$.

2.3. Optimización de Hiperparámetros

Sea $f : X \to \mathbb{R}$ una función que devuelve el máximo error de validación de un modelo entrenado a partir de una combinación de hiperparámetros $\mathbf{x} \in X$, se desea encontrar $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}} = arg\min_{\mathbf{x} \in X} f(\mathbf{x})$$

Es decir, se busca encontrar la combinación óptima de hiperparámetros dentro de un dominio X para obtener el mínimo error de representación en un dado conjunto de validación.En el caso de la construcción de una base *hp-greedy* óptima:

$$\mathbf{x} = (n_{max}, l_{max}, \varepsilon, \hat{\Lambda}_0)$$

El hiperparámetro de mayor interés es sin duda la *semilla*, pues a primera vista no hay una forma obvia de elegir su valor óptimo. Además una semilla óptima para una combinación de (n_{max_1}, l_{max_1}) no lo será necesariamente para otra combinación (n_{max_2}, l_{max_2}).

También se observó que en ciertas situaciones un valor muy elevado de l_{max} o muy pequeño de la *tolerancia greedy* pueden conducir a un modelo *sobreajustado*. Por esto es de interés realizar una optimización que involucre a la totalidad de los hiperparámetros dentro del dominio X.

El problema al momento de realizar esta optimización es que la función f no es una función analítica, sino es que es el resultado de entrenar el modelo y evaluar el error de representación con un conjunto de validación, lo que la hace costosa de evaluar (computacionalmente hablando). Esta sección se centrará en la **optimización Bayesiana** [3, 4], un método que intenta reducir al mínimo el número de evaluaciones de f para encontrar $\hat{\mathbf{x}}$ y se puede colocar dentro de una categoría llamada optimización secuencial basada en modelos, o **SMBO**[5, 6] (*Secuential Model-Based Optimization*).

Además existen dos métodos muy utilizados que no utilizan modelos, los cuales son la busqueda exaustiva (o *grid search*) y la búsqueda aleatoria. Antes de hablar de la optimización Bayesiana se introducirá brevemente dl método de la búsqueda exhaustiva para luego realizar una comparación entre ambos métodos en un problema de optimización relativamente sencillo.

2.3.1. Búsqueda Exhaustiva

La búsqueda exhaustiva o *grid search* consiste en probar todas las combinaciones posibles dentro de un espacio de hiperparámetros para seleccionar la solución óptima. Es decir que si se quiere buscar la combinación óptima de (n_{max}, l_{max}) para un rango

de valores $n_{max} \in N$, $l_{max} \in L$ se deberán probar todas las combinaciones posibles del producto cartesiano $N \times L = \{(n_{max}, l_{max}) | n_{max} \in N, l_{max} \in L\}$.

La ventaja de este método está en que el resultado óptimo está garantizado, pues se pueden comparar todos los resultados entre sí y elegir el mejor. El problema es que, como se mencionó, la función f es costosa de evaluar, y por otro lado el número de combinaciones posibles escala exponencialmente con cada hiperparámetro extra a optimizar (además se debe tener en cuenta que la semilla $\hat{\Lambda}_0$ tendrá generalmente más de una dimension).

Por estas razones la búsqueda exhaustiva no será un método viable en la mayoría de los casos que son de interés para este trabajo. Sin embargo se puede poner a prueba con casos simplificados para luego comparar los resultados con otros métodos más eficaces.

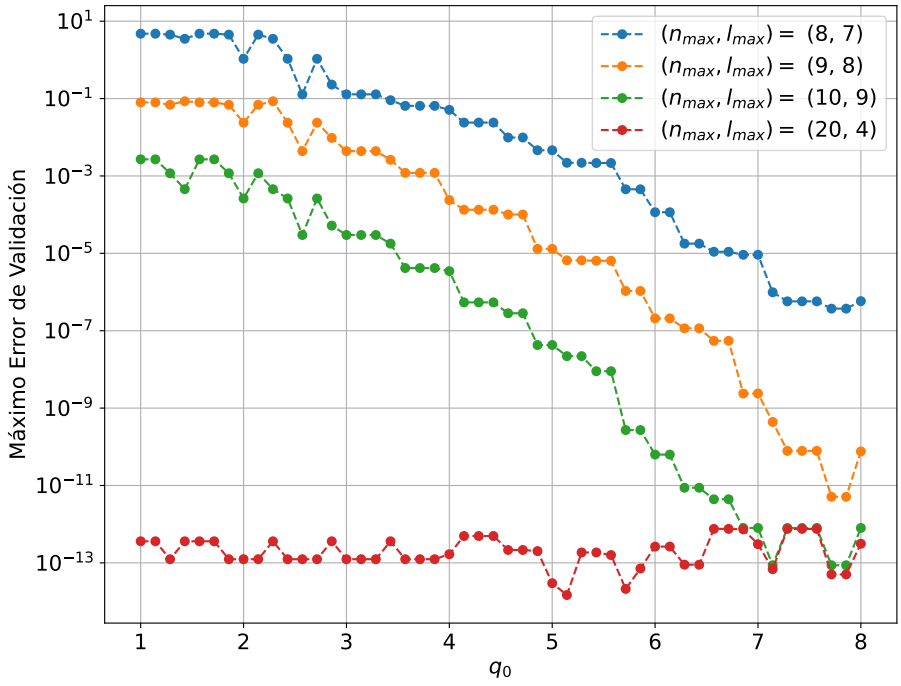

Figura 2.10: Variación de la semilla q_0 para distintas combinaciones de (n_{max}, l_{max})

Por ejemplo, para un conjunto de entrenamiento con cincuenta ondas equidistantes en el espacio del parámetro unidimensional $q : 1 < q < 8$, se quiere optimizar el error de representación para un conjunto de validación con mil ondas. Los hiperparámetros a optimizar son $\mathbf{x} = (n_{max}, l_{max}, \hat{\Lambda}_0)$, dejando ε fijo en 1×10^{-12} para simplificar la búsqueda, que se realiza en los siguientes intervalos:

$$
\begin{aligned}
n_{max} &\in [5, 20], \\
l_{max} &\in [1, 10], \\
\hat{\Lambda}_0 &\in \{q_0 \mid q_0 = 1 + i\Delta q,\ i \in \mathbb{N} : 0 \leq i \leq 49,\ \Delta q = 7/49\}.
\end{aligned}
$$

Son 16 valores de n_{max}, 10 valores de l_{max} y 50 para q_0 ($\hat{\Lambda}_0 = q_0$). Lo que hace un total de 8000 combinaciones posibles.

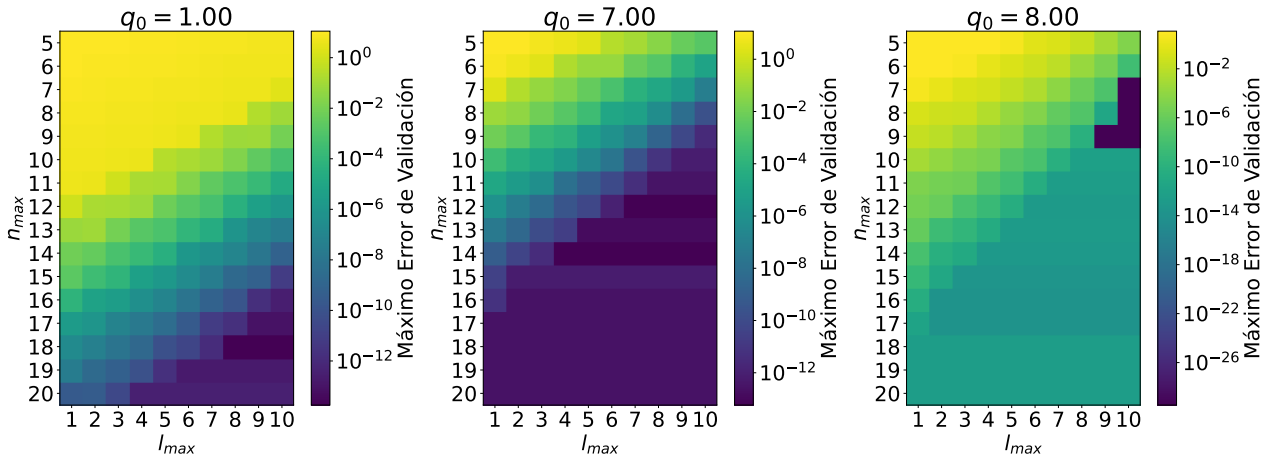

Figura 2.11: Máximo error de validación en función de n_{max} y l_{max} para tres diferentes semillas q_0.

Si bien no se puede graficar el error en función de los tres hiperparámetros a la vez, se puede obtener bastante información al dejar fijo uno o dos hiperparámetros. Por ejemplo en la figura 2.10 se ven los resultados de variar únicamente la semilla para diferentes combinaciones de n_{max} y l_{max}. Se observa que en este conjunto de datos la semilla suele ser óptima a valores cercanos a $q_0 = 8$, pero a la vez al aumentar el n_{max} la influencia de la semilla es menor; con $(n_{max}, l_{max}) = (20, 4)$ hay una diferencia de un orden de magnitud entre la mejor y la peor semilla, mientras que con $(n_{max}, l_{max}) = (10, 9)$, por ejemplo, hay una diferencia de diez ordenes de magnitud.

Luego, en la figura 2.11 se observa el error de validación en función de las combinaciones posibles de n_{max} y l_{max} para tres diferentes semillas. A la derecha, con $q_0 = 8$ se observa el mejor error de representación obtenido en la búsqueda exhaustiva, con un valor de 4×10^{-30} (el cual se obtuvo para 16 combinaciones de hiperparámetros), casi 15 ordenes de magnitud menor que el siguiente mejor error.

2.3.2. Optimización Secuencial Basada en Modelos (SMBO)

La optimización secuencial basada en modelos es un formalismo que es útil para explicar de manera general el funcionamiento de la optimización bayesiana.

La ventaja de este método es que cada evaluación de la función f se decide en base al conjunto D de observaciones realizadas previamente. Esto se logra entrenando un modelo sustituto $\mathcal{M}$ en base al conjunto D, y luego utilizando una función S llamada función de *adquisición*, que decidirá el mejor punto a evaluar en la función real.

Se parte de un conjunto D generado a partir de un muestreo de observaciones de la función f de la forma $(\mathbf{x}_j, f(\mathbf{x}_j))$, a partir del cual se ajusta el modelo sustituto $\mathcal{M}$. Luego, maximizando la función de adquisición S se elije el siguiente conjunto de

hiperparámetros $\mathbf{x}_i$ para evaluar la función f y se agrega el par $(\mathbf{x}_i, f(\mathbf{x}_i))$ al conjunto de observaciones D. Una vez hecho esto se vuelve a ajustar el modelo $\mathcal{M}$ y se repite el proceso, que está explicado en forma de pseudocódigo en el algoritmo 3.

Algoritmo 3 `SMBO`

Input: $f, X, S, \mathcal{M}$

1: $D = \text{InicializarMuestras}(f, X)$
2: **for** $i = 1, 2, ...$ **do**
3: $\quad \mathcal{M} = \text{AjustarModelo}(D)$
4: $\quad \mathbf{x}_i = arg\,\text{máx}_{\mathbf{x} \in X}\, \mathcal{S}(\mathbf{x}, \mathcal{M})$.
5: $\quad y_i = f(\mathbf{x}_i)$ ▷ Paso costoso
6: $\quad D = D \cup \{(\mathbf{x}_i, y_i)\}$
7: **end for**

2.3.3. Optimización Bayesiana

Lo que caracteriza a la optimización bayesiana dentro del formalismo de la optimización secuencial basada en modelos, es justamente la creación del modelo. En la optimización bayesiana se construye un modelo estadístico, donde se representa con $P(y|\mathbf{x})$ la predicción del modelo, siendo y el resultado de una evaluación $f(\mathbf{x})$. El nombre del método se debe a que para la construcción del modelo se utiliza el teorema de Bayes:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)\ P(y)}{P(\mathbf{x})}$$

En la terminología bayesiana, se conoce a $P(y|\mathbf{x})$ como probabilidad a posteriorí o *posterior*, que es proporcional a la probabilidad a priori o *prior* $P(y)$ por la función de verosimilitud o *likelihood* $P(\mathbf{x}|y)$.

TPE [6]
graph [7]
Optimización 1D
Comparación

2.3.4. Semilla 1D

2.3.5. Semilla 2D

2D optimización semilla grid search
2D optimización semilla optune
comparación
Full optimización 2D

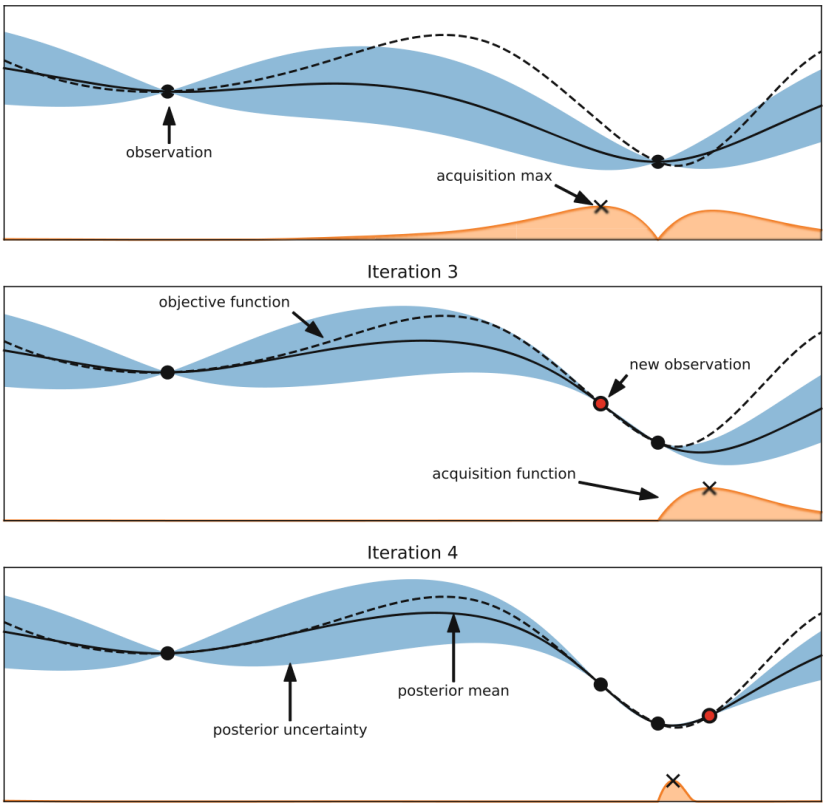

Figura 2.12: Máximo error de validación en función de n_{max} y l_{max} para tres diferentes semillas q_0. [7]

optimización con tiempo

2.3.6. Importancia de los Hiperparámetros

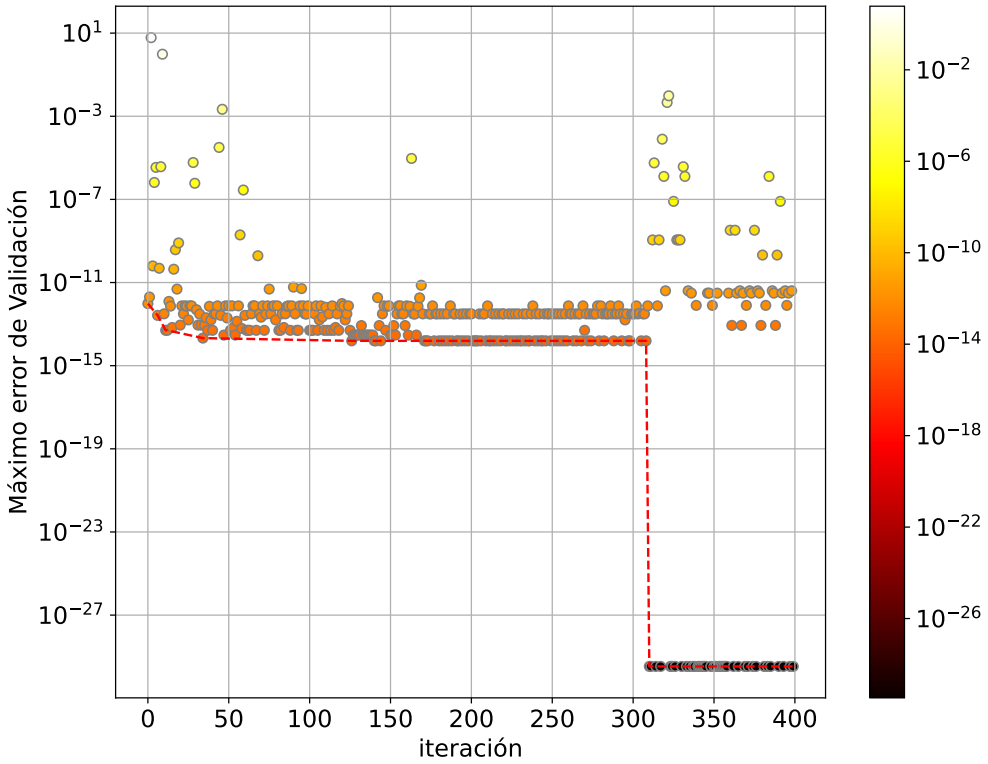

Figura 2.13: Máximo error de validación en función de n_{max} y l_{max} para tres diferentes semillas q_0.

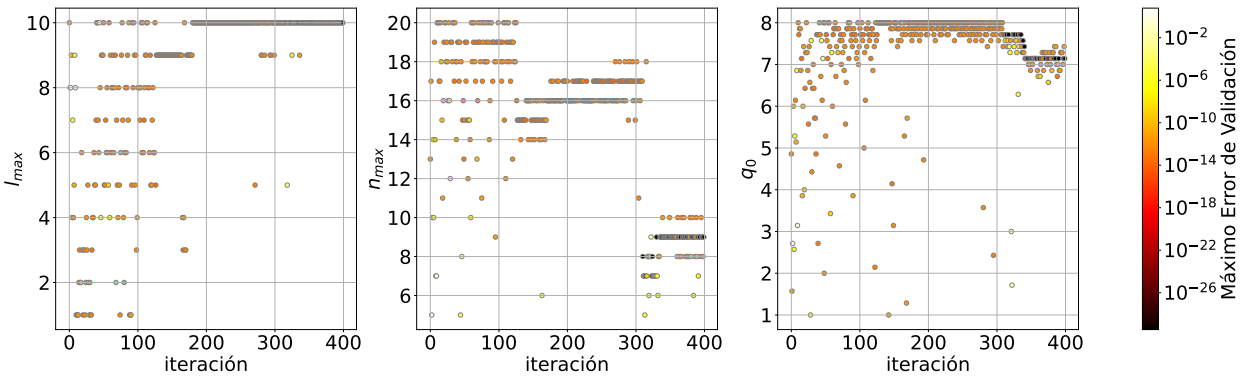

Figura 2.14: Máximo error de validación en función de n_{max} y l_{max} para tres diferentes semillas q_0.

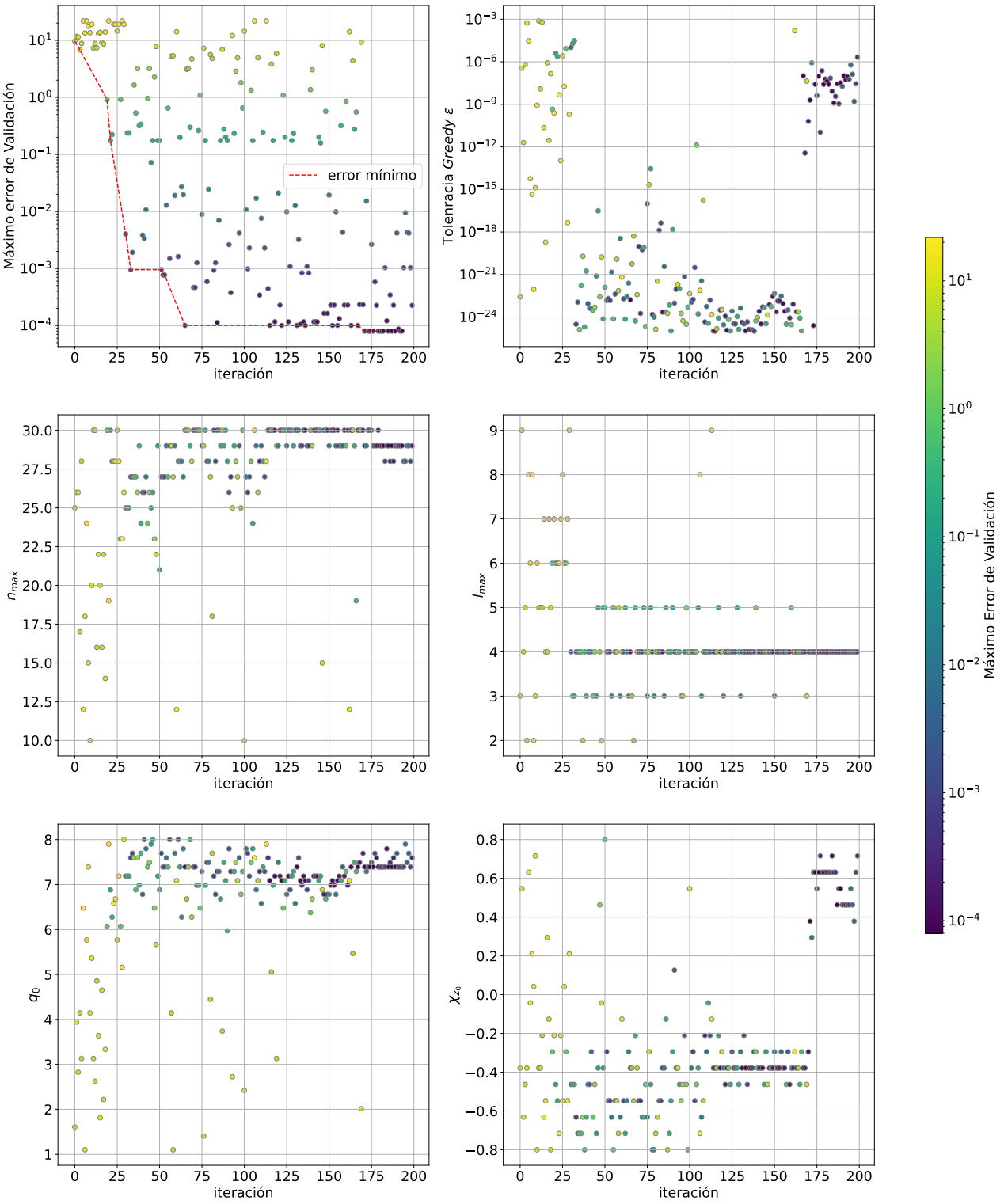

Figura 2.15: Máximo error de validación en función de n_{max} y l_{max} para tres diferentes semillas q_0.

Bibliografía

[1] Varma, V., Field, S. E., Scheel, M. A., Blackman, J., Kidder, L. E., Pfeiffer, H. P. Surrogate model of hybridized numerical relativity binary black hole waveforms. *Physical Review D*, **99** (6), mar 2019. 3

[2] Hesthaven, J. S., Gottlieb, S., Gottlieb, D. Spectral Methods for Time-Dependent Problems. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2007. 5

[3] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., de Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, **104** (1), 148–175, 2016. 13

[4] Brochu, E., Cora, V. M., de Freitas, N. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010. URL https://arxiv.org/abs/1012.2599. 13

[5] Dewancker, I., McCourt, M., Clark, S. Bayesian optimization primer, 2015. URL https://app.sigopt.com/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf. 13

[6] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B. Algorithms for hyper-parameter optimization. En: J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger (eds.) Advances in Neural Information Processing Systems, tomo 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf. 13, 16

[7] Feurer, M., Hutter, F. Hyperparameter Optimization, págs. 3–33. Cham: Springer International Publishing, 2019. URL https://doi.org/10.1007/978-3-030-05318-5_1. 16, 17

Publicaciones asociadas

1. Mi primer aviso en la revista **ABC**, 1996
2. Mi segunda publicación en la revista **ABC**, 1997

Agradecimientos

A todos los que se lo merecen, por merecerlo