COLEGIO DE POSTGRADUADOS

INSTITUCIÓN DE ENSEÑANZA E INVESTIGACIÓN EN CIENCIAS AGRÍCOLAS

CAMPUS MONTECILLO

POSGRADO EN SOCIOECONOMÍA, ESTADÍSTICA, E INFORMÁTICA

DISEÑO EXPERIMENTAL PARA UNA MEZCLA DE q COMPONENTES Y n VARIABLES DE PROCESO PARA INVESTIGAR LAS PROPIEDADES REOLÓGICAS DE GELES DE PECTINA

CHRISTIAN ISRAEL PONCE GUERRERO

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS

MONTECILLO, TEXCOCO, EDO. DE MÉXICO
2016

La presente tesis titulada: "**Diseño experimental para una mezcla de *q* componentes y *n* variables de proceso para investigar las propiedades reológicas de geles de pectina**", realizada por el alumno: "**Christian Israel Ponce Guerrero**", bajo la dirección del Consejo Particular indicado, ha sido aprobada por el mismo y aceptada como requisito parcial para obtener el grado de:

MAESTRO EN CIENCIAS
ESTADÍSTICA

CONSEJO PARTICULAR

CONSEJERO: _____
Dr. Ciro Velasco Cruz

ASESOR: _____
Dr. Luís Medina Torres

ASESOR: _____
Dr. Gerardo H. Terrazas González

Montecillo, Texcoco, Edo. de México, 3 de abril de 2016.

DISEÑO EXPERIMENTAL PARA UNA MEZCLA DE q COMPONENTES Y n VARIABLES DE PROCESO PARA INVESTIGAR LAS PROPIEDADES REOLÓGICAS DE GELES DE PECTINA

Christian Israel Ponce Guerrero, Mtro.
Colegio de Postgraduados, 2016.

El problema de modelar una o varias respuestas dentro de un espacio con alta dimensionalidad requiere diseñar un plan de muestreo que logre captar eficazmente las irregularidades en el espacio. Recientemente, el proceso Gaussiano arbolado ha sido utilizado en experimentos por computadora para ajustar modelos estadísticos cuya estructura de correlación varía dentro de un hipercubo en el que todos los factores son independientes. En este trabajo, se utilizó una variación del algoritmo NTLBG para generar diseños experimentales uniformes dentro del hipercubo $\bar{I}^4$ y dentro de tres *Símplices*. Se utilizó, además, una transformación para mapear los puntos dentro de cada *Símplex* hacia el espacio euclidiano. De este modo, el espacio experimental es una combinación de un hípercubo de cuatro dimensiones y tres *Símplices*, i.e., $\bar{I}^4 \times T_4 \times T_2 \times T_2$, que se mapeó al hipercubo $\bar{I}^9$ para el análisis.

Por otro lado, se comparó un enfoque *inteligente* basado en *Active Learning Cohn* (ALC) para hacer crecer el diseño experimental inicial con base en los resultados, en un proceso iterativo de experimentación/simulación-análisis, contra un enfoque *estático* con un tamaño inicial que igualó al del enfoque *inteligente* en cada iteración.

Los resultados obtenidos sugieren que es conveniente comenzar con un diseño muy pequeño, i.e. $n_0 = 20$, y añadir nuevas corridas experimentales en las regiones de mayor incertidumbre, en lugar de comenzar con un diseño experimental de mayor tamaño $n_0 >> 20$. En la práctica, un diseño experimental de tamaño 150 podría considerarse excesivamente grande, pero si se considera que el espacio experimental tiene 9 dimensiones independientes (12 dimensiones en total), esta metodología es mucho más eficiente que la utilizada en diseños factoriales fraccionales, pues explora cada variable en un número mayor de niveles con un número reducido de experimentos.

Finalmente, dada la naturaleza estocástica de la exploración de la distribución *a posteriori* de los parámetros, es necesario permitir una exploración exhaustiva en busca del árbol con máxima densidad *a posteriori*.

Keywords: Proceso Gaussiano arbolado, Diseño de experimentos, Experimentos con mezclas, Diseño uniforme, NTLBG, ALC

EXPERIMENTAL DESIGN FOR A MIXTURE OF q COMPONENTS AND n PROCESS VARIABLES IN ORDER TO MODEL THE RHEOLOGICAL PROPERTIES OF PECTIN GELS

Christian Israel Ponce Guerrero, Mtro.
Colegio de Postgraduados, 2016.

The problem of fitting a statistical model in a high-dimensional space requires a sampling plan that effectively captures the irregularities in the space.

Recently, the treed Gaussian Process has been succesfully used in computer experiments where the asoumption of stationarity in the correlation structure between independent factors is inadequate.

In this work, we used a variation of the NTLBG algorithm in order to create uniform designs in the hypercube $\bar{I}^4$ and in three *Simplices*. Moreover, we used a transformation to map the vectors from each *Simplex* onto the Euclidian space. Thus, the experimental space is a combination of a four-dimension hypercube and three *Simplices*, i.e., $\bar{I}^4 \times T_4 \times T_2 \times T_2$, mapped onto the hypercube $\bar{I}^9$ for analysis.

On the other hand, we compared an *intelligent* approach based on *Active Learning Cohn* (ALC) to grow the initial experimental design based on the results, in an iterative process of simulation-analysis, against an *static* approach where the initial size of the experiment matches that of the *intelligent* approach in each iteration.

Results suggest that it is preferable to start with a design of a smaller size of $n_0 = 20$, and to grow it intelligently by adding additional runs in the regions where uncertainty is greater, rather than start with a uniform design of a bigger size $n_0 >> 20$.

In practice, an experimental design of size ≥ 150 could be deemed excesively big but, considering that the experimental space has 9 independent dimensions (12 dimensions in total), this methodology is way more efficient than fractional factorial, because it explores each variable in a greater number of levels with a reduced number of experiments.

Finally, given the stochastic nature of the exploration of the *a posteriori* distribution, it is necessary to allow an exhaustive exploration to search for the maximum *a posteriori* (MAP) tree.

Keywords: Treed Gaussian Process, Design of experiments, Experiments with mixtures, Uniform design, NTLBG, ALC

Para Ana Sofía Blancas Gallardo, cuya existencia me ha motivado a continuar hacia adelante y me ha enseñado a sacrificar sin extrañar. Para todo hay un momento en la vida, y el tuyo llegó a mi vida para sincronizar todos los demás.

En memoria de mi perro Bobby, quien me regaló 16 años de su vida a mi lado y por todas las noches que durmió junto a mí mientras leía, escribía, estudiaba, trabajaba y dormía.
(Sept. 1998 – 31 de julio de 2014).

Agradecimientos

Desde antes de comenzar a escribir este trabajo, sentí que debía agradecer en primer lugar a todas las personas que lo hicieron posible y que jamás conoceré. Muchas gracias a todos los mexicanos que sí pagan su impuestos y que a través de CONACyT hacen posible que personas como yo estudien hasta niveles incontratables.

En segundo lugar, quiero agradecer a mi madre: Verónica Guerrero, por quererme y dejarme de querer en los momentos que así lo requirieron: ¡gracias!.

Quiero agradecer a mi familia (de sangre y por elección) por todo el apoyo económico y emocional, así como por las múltiples ocasiones que me recordaron que tenía que terminar de escribir esta tesis. En verdad no lo iba a olvidar jamás, pero gracias por recordármelo una y otra y otra vez.

Quiero hacer una mención especial al Dr. Ciro Velasco Cruz, por su apoyo y orientación para llevar a cabo este trabajo. Estoy seguro que sabes que pusiste la vara muy arriba, pero sabías que lo íbamos a lograr.

Finalmente, quiero agradecer al Dr. Robert Brandon Gramacy, por haber desarrollado y mantener la librería `tgp` de R y por sus comentarios, los cuales hicieron posible el análisis de este trabajo.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En el contexto de este trabajo, una mezcla se define como la combinación de dos o más ingredientes (en lo sucesivo, componentes) en proporciones que simbólicamente se dan por

$$\mathbf{x} = \{(x_1, x_2, \ldots, x_q);\ 0 \leq l_i \leq x_i \leq u_i \leq 1,\ x_i \in \mathbb{R},\ i = 1,2,...,q;\ i,q \in \mathbb{N}\}$$

donde q es el número total de componentes sujetos a por lo menos una restricción lineal, tal como $\sum_{i=1}^{q} x_i = 1$, que define un espacio conocido como *Símplex*; l_i y u_i son los límites inferior y superior del i-ésimo componente. Dado que las proporciones de los q componentes siempre suman una constante, la proporción de todos los componentes no se puede definir arbitrariamente, ya que cada una de ellas depende de la proporción de los demás componentes.

En la industria alimenticia y farmacéutica, entre otras, el desarrollo de un nuevo producto, o su reformulación, parte de una *receta* o mezcla inicial (existente o teórica). Los componentes de la mezcla pueden o no tener una función tecnológica, como los aditivos, coadyuvantes, etc., que producen características deseables en el producto final. Es común que dichos componentes estén sujetos a otras restricciones adicionales, como por ejemplo límites de aplicación que pueden ser establecidos ya sea por la normatividad vigente, por la aplicación tecnológica o por la toxicidad.

Por ejemplo, la Norma CODEX STAN 296-2009 para los confituras, jaleas y mermeladas establece que el contenido de fruta utilizada como ingrediente en el producto terminado no debe ser menor a 45 % en general a excepción de algunas frutas. Además, los sólidos solubles para confituras, jaleas y mermeladas de agrios, como productos terminados, debe estar en todos los casos entre 60 y 65 %, o superior. Asimismo, para las mermeladas de frutos cítricos, la cantidad de fruta utilizada como ingrediente en el producto terminado no debe ser menor al 30 %, y el contenido de sólidos solubles debe estar entre 40 y 65 %, o menos. Por otro lado, la Norma CODEX STAN 192-1995 para los aditivos alimentarios establece que la pectina no tiene un límite de aplicación salvo un par de excepciones. Sin embargo, por sus propiedades espesantes y gelificantes, es preferible

utilizarla hasta un máximo de $2\%p/p$ en confituras, jaleas y mermeladas.

Crear un producto con características deseables como resultado de una mezcla se reduce a mezclar los componentes en proporciones adecuadas; sin embargo, en ocasiones también es necesario estudiar el efecto de otros factores que no necesariamente forman parte del producto pero que participan durante el proceso de su fabricación, como pueden ser la temperatura, presión, tiempo, pH, etc. En este contexto, a estos factores se les denomina "variables de proceso" y definen el "espacio factorial". Por lo que ahora se identifican dos espacios, uno definido por el *Símplex* y otro por el factorial.

La exploración conjunta de los espacios *Símplex* y factorial, con el objetivo de identificar una región donde se obtendrá un producto con características óptimas para su uso o aplicación, se podría realizar mediante uno de los siguientes enfoques:

Ensayo y error: este enfoque es quizás el más común y el menos recomendable. Este consiste en confiar en la experiencia del *formulador* para que, con base en dicha experiencia, se vaya ajustando iterativamente las proporciones de las mezclas y las variables de proceso hasta encontrar las condiciones óptimas. Este enfoque es poco recomendable porque es meramente subjetivo, no garantiza que se encuentre una región óptima y casi nunca explora todo el espacio, por lo que en caso de encontrar una región óptima, esta podría no ser la única ni la mejor.

Basada en teoría estadística: este enfoque usa la 'Metodología de superficie de respuesta' como un medio más objetivo, ya que utiliza secuencialmente diseños de experimentos para ajustar modelos lineales y, con base en los resultados obtenidos en cada ensayo, se determina en qué dirección se debe ajustar las mezclas y las variables de proceso para alcanzar una región óptima. Debido que estos experimentos son un tipo particular de experimentos factoriales, es claro que conforme el número de componentes y de variables de proceso se incrementa, el número de experimentos en cada ensayo también se incrementa rápidamente. Además, dado que la exploración del espacio es secuencial y depende del punto de partida, la región óptima encontrada podría corresponder a un óptimo local y no necesariamente sería la mejor[1].

Debido a que ninguno de estos enfoques ha dado respuesta objetiva y general a los problemas donde se manejan mezclas, es necesario plantear una metodología experimental adecuada que permita ajustar modelos estadísticos que describan la(s) respuesta(s) de interés dentro del *Símplex* y el espacio factorial, de un modo eficiente y exhaustivo con la finalidad de encontrar regiones de interés dentro de dicho espacio.

El presente trabajo consiste en proponer una metodología experimental para explorar exhaustiva y eficientemente el espacio definido por la proporción de los componentes de un gel de pectina,

[1] Mejor en el sentido de variabilidad y tamaño de la región

así como la fuerza iónica y el pH como variables de proceso, con la finalidad de determinar uno o más modelos estadísticos que relacionen los componentes de la mezcla y las variables de proceso con las respuestas de interés.

Capítulo 2

Antecedentes

En la literatura *clásica* (Fang et al., 2005; Santner et al., 2003) sobre diseño de experimentos por computadora, el modelo estadístico preferido como *modelo sustituto* o *meta-modelo* de los simuladores más complejos es el proceso Gaussiano. Este modelo tiene las siguientes ventajas:

1. Es un modelo conceptualmente sencillo.

2. Permite incluir fácilmente conocimiento *a priori* a través de la estructura de covarianzas.

3. Permite estimar intervalos de confianza con menor incertidumbre.

Todas estas son propiedades deseables en un modelo; sin embargo, Gramacy and Lee (2008a) mencionan que el Proceso Gaussiano tiene las siguientes tres desventajas:

1. La inferencia sobre el proceso Gaussiano escala deficientemente con el número de puntos N en el diseño experimental $\mathcal{P}$, requiriendo típicamente tiempos de $O(N^3)$ para calcular la matriz inversa de la matriz de covarianzas de dimensión $N \times N$.

2. La suposición de estacionalidad del proceso Gaussiano dentro de todo el espacio es una suposición muy fuerte; en muchas aplicaciones reales no es posible modelar uniformemente la incertidumbre, debido a que algunas regiones tenderán a mostrar mayor variabilidad que otras.

3. El error de predicción estimado de un modelo estacionario no depende directamente de la respuesta observada localmente; el error de predicción en un punto depende de la distancia entre observaciones, dando más importancia a las observaciones cercanas; también de una medida global del error que utiliza todas las discrepancias entre las observaciones y las predicciones sin considerar la distancia desde el punto de interés.

Todas estas deficiencias se pueden abordar creando particiones del espacio $\mathcal{B}$ de las variables de entrada y ajustando procesos Gaussianos estacionarios distintos dentro de cada región (Kim et al., 2005). Al crear particiones, se tiene un mecanismo sencillo para ajustar procesos Gaussianos no estacionarios y así aliviar cargas computacionales debido a que se ajustan modelos con menos datos. En la Sección 2.4.2 se trata este tema con más detalle.

Para diseñar un experimento con el objetivo de ajustar una superficie de respuesta o estimar $I = \int_{\mathcal{B}} f(\mathbf{u}) d\mathbf{u}$ mediante algún método numérico, ciertas propiedades son deseables en el conjunto de puntos $\mathcal{P}$. Por ejemplo, siempre es deseable que el experimento sea fácil de construir, que N (el tamaño del conjunto $\mathcal{P}$) pueda crecer, que se pueda proyectar en $k < s$ dimensiones sin que haya repetición de puntos, y que el error de la estimación sea lo más pequeño posible, entre otros. Prácticamente, existe una infinidad de formas de seleccionar un conjunto de puntos dentro de la región de interés, porque conceptualmente los puntos se definen como:

$$\mathcal{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n : \mathbf{x_i} \in \mathcal{B} \subseteq \mathbb{R}^s,\ 1 \leq s \in \mathbb{N},\ i = 1, \ldots, n\}$$

El conjunto $\mathcal{P}$ corresponde a las condiciones en que se ensayarían los experimentos.

Supongamos que se desea desarrollar un nuevo producto o que es necesario optimizar la formulación del mismo, así como las condiciones del proceso, sin restricciones en el número de corridas experimentales que se pueden realizar. Por un lado, si se considera sólo las variables correspondientes al proceso de fabricación, desde un enfoque clásico del diseño experimental y de la metodología de superficie de respuesta, probablemente se seguiría un enfoque secuencial, partiendo de un diseño factorial ortogonal de tamaño fijo y ajustando en cada iteración las condiciones (niveles de los factores) en busca de las condiciones *óptimas*. Por otro lado, si se considera sólo los componentes de la formulación, Cornell (2002) proporciona un compendio de diseños experimentales *estándar* con mezclas que podrían ser usados.

Cuando es necesario considerar las variables de proceso junto con los componentes de la formulación, se puede utilizar una combinación de diseños factoriales y diseños para mezclas. Al combinar ambos tipos de diseños el número de corridas experimentales crece de manera exponencial con el número de variables y resulta prácticamente imposible correr un experimento factorial. Además, en caso de que alguna de las variables del proceso resultara no significativa, al colapsar el diseño en un menor número de dimensiones, debido a la naturaleza del diseño factorial, se tendría repeticiones en los niveles de las variables restantes, y por lo tanto el diseño resultaría ineficiente. (Santner et al., 2003, Pág. 125)

Es por lo anterior que es necesario buscar alternativas a los diseños experimentales clásicos para explorar de manera eficiente el espacio experimental $\mathcal{B}$, definido por las variables de proceso y los componentes de la formulación.

A continuación se introducen las bases necesarias en que se fundamenta una alternativa que será utilizada para la exploración de espacios ortogonales y no ortogonales. Se presentan los métodos numéricos conocidos como métodos Monte Carlo y métodos Cuasi-Monte Carlo.

2.1. Métodos Monte Carlo

La aplicación de métodos Monte Carlo a los diseños experimentales depende principalmente de la generación y la selección de los puntos en donde se simularán las muestras. Las ideas para la selección de un conjunto de puntos en el diseño experimental

$$\mathcal{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n;\ \mathbf{x}_i \in \bar{I}^s, 1 \leq s \in \mathbb{N}\}$$

donde $\bar{I}^s = [0,1]^s$ es el hipercubo unitario cerrado de dimensión s, se toman del método Monte Carlo que fue desarrollado por Metropolis and Ulam (1949) para la aproximación de integrales de alta dimensionalidad en un dominio $\mathcal{B} \subseteq \mathbb{R}^s$, y que satisface $0 < \lambda_s(\mathcal{B}) < \infty$, donde λ_s denota la medida de Lebesgue en s dimensiones:

$$\int_{\mathcal{B}} f(\mathbf{u}) d\mathbf{u} = \lambda_s(\mathcal{B}) \int_{\mathcal{B}} f d\mu = \lambda_s(\mathcal{B}) E(f), \tag{2.1}$$

donde $E(f)$ es la esperanza matemática de la variable aleatoria f; el espacio $\mathcal{B}$ se convierte aquí en un espacio probabilístico con medida de probabilidad $d\mu = d\mathbf{u}/\lambda_s(\mathcal{B})$.

Típicamente, cuando se dispone de realizaciones de la variable aleatoria f, el problema de integración numérica se reduce al problema de aproximar una esperanza matemática. Se utiliza la media muestral como una aproximación de la esperanza, que es insesgada y presumiblemente de mínima varianza. Sea f una variable aleatoria en un espacio probabilístico arbitrario $(A, \mathcal{A}, \lambda)$. La estimación Monte Carlo de la esperanza $E(f)$ se obtiene tomando una muestra de N puntos independientes $a_1, \ldots, a_N \in A$ con distribución $d\lambda$ y tomando

$$E(f) \approx \frac{1}{N} \sum_{n=1}^{N} f(a_n). \tag{2.2}$$

La ley fuerte de los grandes números garantiza que este procedimiento converge casi en todas partes (c.t.p).

$$\lim_{N \to \infty} \frac{1}{N} \sum_{n=1}^{N} f(a_n) = E(f)\ \ \lambda^{\infty}\text{c.t.p,.}$$

Al sustituir 2.2 en 2.1, la aproximación Monte Carlo resulta en

$$\int_{\mathcal{B}} f(\mathbf{u})d\mathbf{u} \approx \frac{\lambda_s(\mathcal{B})}{N} \sum_{\substack{n=1 \\ \mathbf{x}_n \in \mathcal{B}}}^{N} f(\mathbf{x}_n), \tag{2.3}$$

donde $\mathbf{x}_1, \ldots, \mathbf{x}_N$ es una muestra aleatoria con distribución $d\mu$ en el espacio $\mathcal{B}$. Si se considera que $\mathcal{B} = \bar{I}^s = [0,1]^s$, entonces el estimador Monte Carlo es

$$\int_{\bar{I}^s} f(\mathbf{u})d\mathbf{u} \approx \frac{1}{N} \sum_{\substack{n=1 \\ \mathbf{x}_n \in \bar{I}^s}}^{N} f(\mathbf{x}_n), \tag{2.4}$$

donde $\mathbf{x}_1, \ldots, \mathbf{x}_N$ son una muestra aleatoria con distribución uniforme en $\bar{I}^s = [0,1]^s$ (Niederreiter, 1992).

En la actualidad, este método es una herramienta muy general que no está limitado sólo a la aproximación de integrales; por este motivo también se les llama "métodos Monte Carlo" en plural. En términos simples, el método Monte Carlo se puede describir como "un método numérico basado en muestreo aleatorio" (Niederreiter, 1992), por lo que pertenece al área de *simulación estocástica*.

Una de las características importantes de los métodos Monte Carlo es que *prometen* errores de integración cuyo orden de magnitud depende del tamaño del conjunto de puntos $\mathcal{P}$ pero no depende del número de dimensiones s. Sin embargo, estos métodos no garantizan cotas para el error, por este motivo Niederreiter (1992) utiliza la palabra *prometen*.

A pesar de ser útiles en múltiples aplicaciones, Niederreiter (1992) resume las siguientes deficiencias de los métodos Monte Carlo:

- sólo tienen cotas probabilísticas para el error;
- no se reflejan la regularidad del integrando;
- es difícil generar muestras aleatorias.

Por este motivo es preferible utilizar otros métodos con características superiores a los métodos Monte Carlo, como son los métodos Cuasi-Monte Carlo.

2.2. Métodos Cuasi-Monte Carlo

Para cada método Monte Carlo es posible construir un método Cuasi-Monte Carlo remplazando las muestras aleatorias por un conjunto de puntos P generados de manera determinística que logre un mejor desempeño al aproximar la integral en 2.1. Por lo tanto, los métodos Cuasi-Monte Carlo son la versión determinística de los métodos Monte Carlo (Niederreiter, 1992).

A diferencia de los métodos Monte Carlo, la naturaleza determinística de los métodos Cuasi-Monte Carlo garantiza que se alcance la "cota para los errores". Además, para un mismo número de evaluaciones de la función en 2.3, los métodos Cuasi-Monte Carlo son más precisos que los métodos Monte Carlo. Entonces, con base en estas dos características, los métodos Cuasi-Monte Carlo son superiores a los métodos Monte Carlo. Adicionalmente, el problema de generar muestras realmente aleatorias mediante el método Monte Carlo desaparece en su versión determinística. (Niederreiter, 1992)

Para la integración numérica, el criterio para la selección de un conjunto de puntos es fácil de hallar y conduce a los conceptos de *sucesión distribuida uniformemente* y *discrepancia*. A continuación se describe brevemente los conceptos de uniformidad y discrepancia.

2.2.1. Uniformidad

En lo sucesivo se considerará al dominio de integración como el hipercubo unitario cerrado en s-dimensiones $\bar{I}^s = [0,1]^s$. Para la aproximación Monte-Carlo en la expresión 2.3, se remplaza el conjunto de puntos $\mathbf{x}_1, \ldots, \mathbf{x}_n$ por la sucesión infinita $\mathbf{x}_1, \mathbf{x}_2, \ldots$ de puntos. Un requisito mínimo para esta sucesión es que se obtenga un método convergente a partir de la expresión 2.3, es decir, se desea la sucesión $\mathbf{x}_1, \mathbf{x}_2, \ldots$ tal que

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x}_n) = \int_{\bar{I}^s} f(\mathbf{u}) d\mathbf{u} \tag{2.5}$$

se mantenga para una clase razonable de integrandos, como por ejemplo, para todas las funciones f continuas en $\bar{I}^s$. Equivalentemente, si para la sucesión $\mathbf{x}_1, \mathbf{x}_2, \ldots$ dentro de $\bar{I}^s$ se cumple que

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} c_J(\mathbf{x}_n) = \lambda_s(J) \tag{2.6}$$

para todos los subintervalos J de $\bar{I}^s$, donde c_J es la función característica de J y λ_s denota la medida de Lebesgue en s dimensiones, entonces la sucesión $\mathbf{x}_1, \mathbf{x}_2, \ldots$ debe estar *uniformemente dispersa* sobre $\bar{I}^s$ (Niederreiter, 1992).

Fang and Wang (1993) proporcionan la siguiente definición más formal para *uniformidad*:

> Sea $F(\mathbf{x})$ una función de densidad acumulada continua en s dimensiones, N un subconjunto infinito de números naturales y $\{P_n, n \in N\}$ una sucesión de conjuntos de $\mathbb{R}^s$, donde P_n tiene cierta estructura. Si
>
> $$D^*(P_n) = o(n^{-1/2}), \text{ conforme } n \to \infty, \tag{2.7}$$
>
> entonces $\{P_n\}$ es un conjunto de *puntos representativos* de $F(\mathbf{x})$. Cuando $F(\mathbf{x})$ es la función

de densidad acumulada $U(D)$, donde D es un dominio cerrado y acotado y $U(\cdot)$ es la función de densidad uniforme, entonces se dice que el conjunto $\{P_n\}$ está *uniformemente disperso* sobre D. Si para cualquier $\varepsilon > 0$ se tiene

$$D^*(P_n) = O(n^{-1+\varepsilon}) \tag{2.8}$$

entonces se dice que $\{P_n\}$ está *uniformemente bien disperso* sobre D.

Regresando a los métodos Monte Carlo, se tiene que el conjunto de puntos $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \sim U(\bar{I}^s)$

$$D_N^*(P) = O(n^{-1/2}(\log\log n)^{1/2}) \tag{2.9}$$

con probabilidad uno (Chung (1949); Kiefer (1961) en Fang and Wang (1993)), donde $D^*(P_n) = D_N^*(P)$ como se define en la expresión 2.11 de la Sección 2.2.2 a continuación. Este hecho indica que el conjunto P_n generado por el método Monte Carlo no está *uniformemente disperso* en $\bar{I}^s$ con probabilidad uno.

Lo anterior sugiere que un conjunto de puntos deseable es aquel para el cual la distribución empírica se asemeja mucho a la distribución uniforme sobre $\bar{I}^s$, es decir, el conjunto de puntos $\mathbf{x}_1, \ldots, \mathbf{x}_N$ debe estar disperso uniformemente sobre $\bar{I}^s$.

2.2.2. Discrepancia

Es posible generar un conjunto de puntos de manera determinística para que la cota del error de aproximación en la expresión 2.4 sea lo más pequeña posible, en particular, que los errores sean menores que la cota probabilística de los métodos Monte Carlo. Las distintas definiciones de discrepancia se consideran medidas cuantitativas para las desviaciones de la distribución uniforme o, equivalentemente, irregularidades de la distribución. Para un subconjunto $B \in \bar{I}^s$, se define

$$A(B;P) = \sum_{i=1}^{N} c_B(\mathbf{x}_i)$$

donde c_B es la función característica de B, por lo tanto $A(B;P)$ es la función contadora que indica el número n, $1 \leq n \leq N$, para el cual $\mathbf{x}_i \in B$. Si $\mathcal{B}$ es una familia no vacía de subconjuntos de $\bar{I}^s$ que sean Lebesgue-medibles, entonces una noción general de discrepancia del conjunto P está dado por

$$D_N(\mathcal{B};P) = \sup_{B\in\mathcal{B}} \left| \frac{A(B;P)}{N} - \lambda_s(B) \right|. \tag{2.10}$$

Note que $0 \leq D_N(\mathcal{B};P) \leq 1$, siempre. Considerando ciertos $\mathcal{B}$ específicos y tomando $I^s = [0,1)^s$, se obtienen los dos conceptos de discrepancia más importantes:

D_N^***:** La *discrepancia estrella*, $D_N^*(P) = D_N^*(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ del conjunto de puntos P, se define como $D_N^*(P) = D_N(\mathcal{J}^*; P)$, donde $\mathcal{J}^*$ es la familia de todos los subintervalos de la forma $[\mathbf{0}, \mathbf{u})$, $\mathbf{u} = \{(u_1, \ldots, u_s); \mathbf{u} \in I^s\}$, con volumen $\prod_{i=1}^s u_i = u_1 \cdots u_s$.

Por lo tanto, la $D_N^*(P)$ se puede escribir como

$$D_N^*(P) = \sup_{\mathbf{u} \in I^s} \left| \frac{A(\mathbf{u}, P)}{N} - \prod_{i=1}^{s} u_i \right| \tag{2.11}$$

D_N**:** La *discrepancia* (extrema), $D_N(P) = D_N(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ del conjunto de puntos P, se define como $D_N(P) = D_N(\mathcal{J}; P)$, donde $\mathcal{J}$ es la familia de todos los subintervalos de I^s de la forma $\prod_{i=1}^s [u_i, v_i)$.

Para cualquier conjunto de puntos $P \in \bar{I}^s$ se tiene que

$$D_N^*(P) \leq D_N(P) \leq 2^s D_N^*(P)$$

Se puede consultar la demostración en Niederreiter (1992, Pág. 15).

Niederreiter (1992) establece además que los conjuntos de puntos con baja $D_N^*(P)$ garantizan errores de aproximación acotados por la desigualdad de Koksma-Hlawka, dada por

$$\left| \int_{I^s} f(\mathbf{x}) d\mathbf{x} - \frac{1}{n} \sum_{k=1}^{n} f(\mathbf{x}_k) \right| \leq D_N^*(P) V(f), \tag{2.12}$$

con $D_N^*(P)$ definida en la expresión 2.11 y $V(f)$ es la variación total de f en el sentido de Hardy y Krause (Niederreiter, 1992). Es decir, para la aproximación en la expresión 2.3, lo que importa no es la aleatoreidad, sino la dispersión uniforme del conjunto de puntos P de la muestra sobre el intervalo de integración.

Hasta aquí se tiene que para lograr una mejor aproximación numérica de integrales, es posible utilizar método Cuasi-Monte Carlo. A continuación se presentan los diseños experimentales que se utilizan con frecuencia y los que usan los métodos Cuasi-Monte Carlo.

2.3. Diseño experimental

Actualmente, existe literatura basta dedicada a diseños estándar para factores independientes, i.e., Diseño completamente al azar, Diseño factorial, etc. (Hinkelmann and Kempthorne, 2007; Montgomery, 2008; Myers et al., 2011) y para mezclas, i.e., *Simplex-Centroid*, *Simplex-Lattice*,

etc. (Cornell, 2002). Estos diseños son bastante útiles cuando el número de factores, bloques y covariables es pequeño, o cuando el número de niveles de los factores también es pequeño (por ejemplo 2 o 3). Los diseños experimentales mencionados para mezclas también son útiles siempre y cuando el número de componentes de la mezcla sea pequeño (menor a 12) y no existan restricciones lineales en las proporciones de los componentes.

Sin embargo, cuando el número de factores es muy grande, y/o cada factor tiene un número grande de niveles, resulta prácticamente imposible llevar a cabo un diseño factorial completo debido a que el número de corridas experimentales aumenta exponencialmente. Por ejemplo, el diseño para un experimento factorial completo con 7 factores y 2 niveles produce $2^7 = 128$ combinaciones de los niveles de los factores o tratamientos; asimismo, el diseño para un experimento factorial con 5 factores y 3 niveles produce $3^5 = 243$ tratamientos.

Una alternativa es un Diseño factorial fraccional que utiliza sólo una fracción del diseño factorial completo y tiene como resultado que algunos efectos estén confundidos con otros y no se pueda diferenciar entre ellos; por este motivo, se procura que las interacciones de mayor orden estén confundidas con los efectos principales, o con las interacciones entre dos factores (Montgomery, 2008). Existen también los llamados diseños óptimos, los cuales son dependientes de una selección predeterminada del modelo. La dependencia de los diseños óptimos con la selección del modelo no siempre es adecuada, especialmente cuando no se conoce el modelo. (Gramacy and Lee, 2009)

Otra alternativa son los diseños Diseño *space filling*, entre los cuales se ha desarrollado varios métodos basados en métodos Monte Carlo y métodos Cuasi-Monte Carlo. Koehler and Owen (1996); Santner et al. (2003); Fang et al. (2005) proporcionan un listado exhaustivo de los diseños *space filling* usados más comúnmente en experimentos por computadora y que también se pueden utilizar en experimentos físicos, entre los que se puede mencionar el diseño hipercubo latino generado mediante diversos métodos (McKay et al., 1979; Tang, 1993), el conjunto *good lattice point*, las redes-(t,m,s) y las sucesiones-(t,s) (Niederreiter, 1988, 2005), el algoritmo LBG de Linde et al. (1980) y su versión determinística NTLBG de Ning et al. (2011b), entre otros. En las secciones 2.3.2, 2.3.3 y 2.3.5 se explica algunos de estos métodos.

2.3.1. Diseños basados en conjuntos de puntos con baja discrepancia

A un conjunto de puntos P con N elementos (o cardinalidad N) en $\bar{I}^s$, se le llama informalmente conjunto de puntos con baja discrepancia si su D_N^* o D_N es pequeña. Owen (1995), Niederreiter (1992) y Fang and Wang (1993) proporcionan métodos Cuasi-Monte Carlo para generar conjuntos de puntos P con baja discrepancia. Además, Fang and Wang (1993) proporcionan dos métodos para la generación de conjuntos de puntos de baja discrepancia en los siguientes dominios:

$$A_s = \{(x_1,\ldots,x_s) : 0 \le x_1 \le \cdots \le x_s \le 1\}$$

$$B_s = \{(x_1,\ldots,x_s) : x_1^2 + \cdots + x_s^2 \le 1\}$$

$$U_s = \{(x_1,\ldots,x_s) : x_1^2 + \cdots + x_s^2 = 1\}$$

$$V_s = \{(x_1,\ldots,x_s) \in \mathbb{R}_+^s : x_1 + \cdots + x_s \le 1\}$$

$$T_s = \{(x_1,\ldots,x_s) \in \mathbb{R}_+^s : x_1 + \cdots + x_s = 1\}$$

El primer método se conoce como *método de transformación inversa*, y se basa en el siguiente teorema:

Teorema 2.3.1 *Sea D un dominio cerrado y acotado en $\mathbb{R}^s$, y $\mathbf{x} \sim U(D)$ una variable aleatoria con distribución uniforme $U(\cdot)$ en D. Suponga además que $\mathbf{x}$ tiene una representación estocástica*

$$\mathbf{x} = \mathbf{x}(\phi) \tag{2.13}$$

donde ϕ es un vector aleatorio t-dimensional, con función de densidad independiente $p_i(\phi_i)$ y función de densidad acumulada $F_i(\phi_i)$.

Para cada $\mathbf{r} \in \bar{I}^t$, $t < s$, sea

$$G_{\mathbf{r}} = \{\mathbf{x} : \mathbf{x} = \mathbf{x}(\phi), \phi \le \mathbf{r}\} \tag{2.14}$$

un dominio. Como $\mathbf{x} \sim U(D)$, entonces $P(\mathbf{x} \in G_{\mathbf{r}}) = v(G_{\mathbf{r}})/v(D)$, donde $v(D)$ y $v(G_{\mathbf{r}})$ representan el volumen de D y $G_{\mathbf{r}}$, respectivamente.

Sea $N(P_F, G_{\mathbf{r}})$ el número de puntos $\mathbf{x}_k$ que caen dentro de $G_{\mathbf{r}}$, entonces

$$GD(P_F, D) = \sup_{\mathbf{r}\in\bar{I}^t} \left| \frac{N(P_F, G_{\mathbf{r}})}{n} - \frac{v(G_{\mathbf{r}})}{v(D)} \right| \tag{2.15}$$

proporciona una medida de uniformidad para P_F sobre D.

Sea $P = \{\mathbf{c}_k, k = 1,\ldots,n\}$ un conjunto en $\bar{I}^t$ con discrepancia d. Entonces el conjunto de puntos $P_F = \{\mathbf{x}_k, k = 1,\ldots,n\}$ definido por

$$\mathbf{x}_k = \mathbf{x}(F_1^{-1}(c_{k1}), \cdots, F_t^{-1}(c_{kt})), k = 1,\ldots,n \tag{2.16}$$

tiene una cuasi discrepancia d con respecto a F (F-discrepancia) como se define en 2.15.

De este modo, basta con generar un conjunto de puntos $P \in \bar{I}^t$ con baja discrepancia y aplicar el método de transformación inversa para generar un conjunto de puntos P_F, también con baja discrepancia, en los dominios mencionados al principio de esta sección.

El segundo método se emplea cuando no es posible utilizar el método de transformación inversa y no se dará más detalles aquí. Para más información se puede consultar el algoritmo en Fang and Wang (1993, Pág. 53).

2.3.2. El conjunto *Good lattice point*

Un método determinístico para generar un conjunto $\mathcal{P}$ en el dominio $\bar{I}^s$ con baja discrepancia es el método *good lattice point*. Al conjunto de puntos obtenido mediante un llamado *good lattice point* módulo n se le conoce como conjunto de puntos *good lattice point*. La siguiente definición por Fang and Wang (1993) proporciona una forma de construir un conjunto de puntos *good lattice point*.

Conjunto *good lattice point*: sea $(n; h_1, \cdots, h_s)$ un vector con componentes enteros que satisfacen $1 \leq h_i < n$, $h_i \neq h_j$ $(i \neq j)$, $s < n$ y máximo común divisor $(n, h_i) = 1$, $i = 1, \cdots, s$.

Sea:

$$q_{ki} = kh_i (\mathrm{mod}\ n);\ k = 1, \ldots, n,\ i = 1, \ldots, s \tag{2.17}$$

$$x_{ki} = (2q_{ki} - 1)/2n, \tag{2.18}$$

donde se utiliza el operador mod n de modo que q_{ki} esté confinado dentro de $1 \leq q_{ki} \leq n$[1]. Entonces, el conjunto $\mathcal{P}_n = \{\mathbf{x}_k = (x_{k1}, \cdots, x_{ks}),\ k = 1, \ldots n\}$ se conoce como conjunto *lattice point* del vector generador $(n; h_1, \ldots, h_s)$. Si el conjunto $\mathcal{P}_n$ tiene la menor discrepancia entre todos los posibles vectores generadores, entonces el conjunto $\mathcal{P}_n$ se conoce como conjunto *good lattice point*.

Encontrar el vector generador $(n; h_1, \ldots, h_s)$ que genere un conjunto *good lattice point* es una tarea complicada y se han establecido ciertas características que deben cumplir dichos vectores para que generen conjuntos *good lattice point*. Fang and Wang (1993, Apéndice A) proporciona tablas con vectores generadores para obtener conjuntos con baja discrepancia para algunos valores de n. Por ejemplo, el vector (101;1,40,85) genera el conjunto de puntos que se muestra en la Figura 2.1.

2.3.3. Algoritmo LBG para generar diseños uniformes con factores independientes

En el campo de procesamiento de señales y compresión de datos existe un proceso llamado cuantificación (*quantization* en inglés). Este consiste en dos funciones: una codificadora y una

[1] Si $q_{ki} = 0$, entonces se sustituye por $q_{ki} = k$

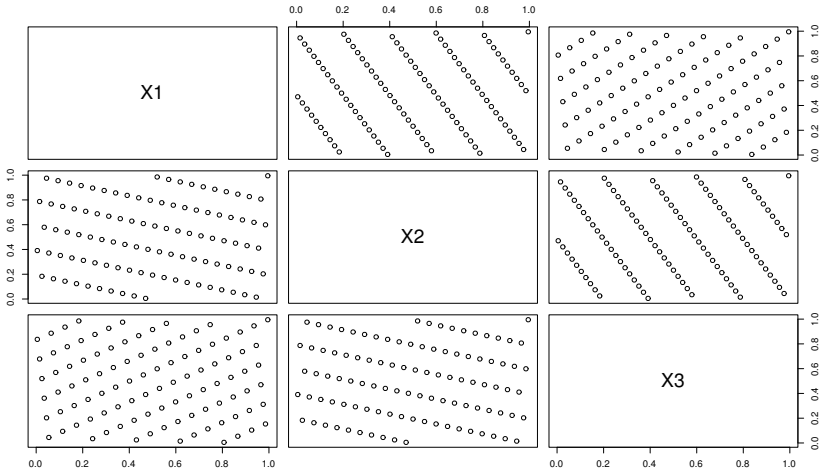

Figura 2.1: Conjunto *good lattice point* generado con el vector$(101;\ 1,40,85)$.

decodificadora. La función codificadora divide el intervalo de valores de entrada en un número finito de intervalos. Cada intervalo representa a todos los valores que se encuentran dentro de él. Como es posible que exista un número demasiado grande de valores dentro de cada intervalo, la función de codificación es irreversible (Sayood, 2012). Un dispositivo o algoritmo para llevar a cabo la cuantificación se conoce como cuantificador (*quantizer* en inglés).

Las condiciones necesarias para un cuantificador óptimo son las siguientes:

- asociado con cada *quantizer* de salida $\mathbf{y}_i$ existe una partición S_i, $i = 1,\ldots,n$, que satisface;

$$S_i = \{\mathbf{x} : ||\mathbf{x}-\mathbf{y}_i|| < ||\mathbf{x}-\mathbf{y}_j||,\ j \neq i\}, \tag{2.19}$$

- para cada partición S_i, los $\mathbf{y}_i$ deben ser iguales a la media condicional

$$\mathbf{y}_i = E[\mathbf{x}|\mathbf{x} \in S_i]. \tag{2.20}$$

Cuando el número de vectores cuantificadores n no es muy grande, Linde et al. (1980) propusieron un algoritmo iterativo para encontrar vectores cuantificadores (algoritmo LBG) que se basa en una secuencia de entrenamiento. Las condiciones necesarias en las expresiones 2.19 y 2.20 proporcionan la base para el algoritmo LBG a continuación (tomado de Fang and Wang (1993)):

1. Iniciar con $t = 0$; proporcionar un conjunto inicial de vectores de salida $\mathbf{Y}_t = \{\mathbf{y}_{ti},\ldots,\mathbf{y}_{tn}\}$ que define la particiones $\mathcal{S}_t = (S_{t1},\ldots,S_{tn})$ mediante la expresión 2.19.

2. Generar una sucesión de vectores de entrenamiento $\mathbf{x}_1,\ldots,\mathbf{x}_N$, $N >> n$, mediante el método Monte Carlo que represente la distribución del vector aleatorio $\mathbf{x}$.

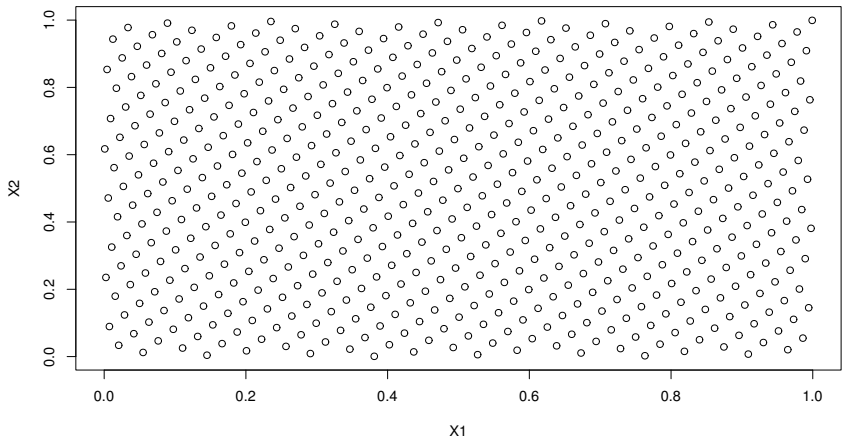

Figura 2.2: Conjunto *good lattice point* de tamaño $N = 610$ en $s = 2$ dimensiones.

3. Asignar cada $\mathbf{x}_i$ al vector más cercano de la partición $\mathcal{S}_t$.

4. Calcular la media muestral $\mathbf{y}_{i+1}$ de $\mathbf{x}_i$ dentro de cada partición $S_{t,j}$, $j = 1,\ldots,n$. Se define $\mathbf{Y}_{t+1} = \{\mathbf{y}_{t+1,1},\ldots,\mathbf{y}_{t+1,n}\}$; si $\mathbf{Y}_{t+1} = \mathbf{Y}_t$, entonces $\mathbf{Y}_t$ es el vector de salida final, de otro modo

5. Hacer $t = t+1$, y volver al paso 3.

De acuerdo con Fang and Wang (1993), el algoritmo LBG tiene la ventaja de ser un algoritmo general que puede generar vectores cuantificadores, pero también tiene las siguientes desventajas:

1. Los vectores cuantificadores de salida sólo son óptimos locales y dependen del conjunto inicial de vectores de salida.

2. La sucesión de entrenamiento se basa en el método Monte Carlo. Por lo tanto, la velocidad de convergencia del método Monte Carlo es $O(n^{-1/2})$ en probabilidad y es lento.

Actualmente se han creado versiones modificadas de este algoritmo, como el NTLBG de Ning et al. (2011a), que se basa en método Cuasi-Monte Carlo, para generar vectores cuantificadores con una distribución uniforme en el *Símplex* T_q. En la siguiente sección se explica la variante del algoritmo LBG.

Por ejemplo, en la Figura 2.4 se muestra el resultado del algoritmo en el espacio Euclidiano generado a partir del conjunto *good lattice point* que se muestra en la Figura 2.2 con $h = (610; 1,377)$, y el conjunto de vectores con distribución $\mathbf{y}_i \sim U(\hat{I}^2)$ que se muestra en la Figura 2.3. Se puede observar cómo los puntos (vectores) que inicialmente aparecen juntos, i.e. 10 y 19; 8 y 14; 7 y 1 en

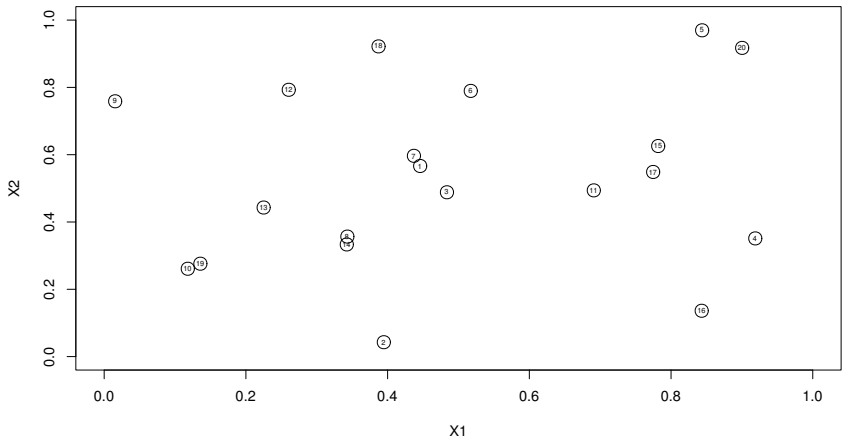

Figura 2.3: Vectores de entrenamiento. Distribución inicial de los puntos en $s = 2$ dimensiones seleccionados aleatoriamente.

la Figura 2.3, se separan y redistribuyen más uniformemente después ser sometidos al algoritmo LBG en la Figura 2.4.

2.3.4. Diseños experimentales clásicos para experimentos con mezclas

En los diseños experimentales para mezclas se asume que la respuesta depende de la proporción de los componentes de la mezcla pero no así de la cantidad total de la mezcla. Sin embargo, también existen experimentos donde la cantidad de mezcla es de interés. Por ejemplo, durante el desarrollo de un nuevo fertilizante, donde no sólo la proporción N-P-K es de interés, sino también la cantidad o la concentración del fertilizante. Los diseños experimentales para mezclas se caracterizan por imponer algunas restricciones lineales sobre las proporciones de los componentes de las mezclas. Suponiendo una mezcla de q componentes en proporciones x_i, $i = 1,2,\ldots,q$, la principal restricción es que las proporciones de los q componentes deben sumar a una constante k, generalmente, $k = 1$ o $k = 100\,\%$; como por ejemplo

$$\mathbf{j}^\top \mathbf{x} = \sum_{i=1}^{q} x_i = 1, \tag{2.21}$$

como en T_q en la Subsección 2.3.1. $\mathbf{j}^\top = (1,\ldots,1)$ es un vector de unos de dimensión q. Al dominio T_q dado en la Sección 2.3.1 se le conoce como *Símplex*.

Adicionales a la restricción lineal en 2.21, pueden existir otras restricciones lineales en un solo componente

$$0 \leq a_i \leq x_i \leq b_i \leq 1,\ i = 1,2,\ldots,q \tag{2.22}$$

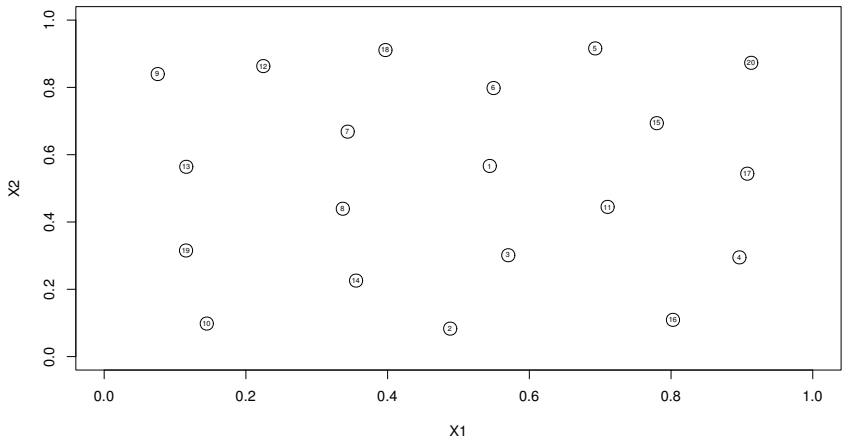

Figura 2.4: Vectores *quantizers*. Distribución final de los puntos en $s = 2$ dimensiones después de redistribuir los puntos mediante el algoritmo LBG.

o en múltiples componentes

$$L_v \leq \sum_{i=1}^{q} C_{vi} x_i \leq U_v,\ v = 1,2,\ldots,V \tag{2.23}$$

donde V es el número de restricciones lineales en múltiples componentes (Cornell, 2002; Ning et al., 2011a). El caso particular en el que $a_i = 0$ y $b_i = 1$ para todos los componentes, corresponde al *Símplex* T_q.

Los diseños clásicos para diseños con mezclas sin restricciones son *Simplex-Centroid* y *Simplex-Lattice*, este último se representa mediante $\{q,m\}$ donde q es el número de componentes de la mezcla y m es el número de niveles de cada componente. La cardinalidad de los diseños *Simplex-Centroid* y *Simplex-Lattice* es $n = 2^q - 1$ y $n = \binom{q+m-1}{m}$, respectivamente.

Los algoritmos *clásicos* para generar diseños experimentales para mezclas, por ejemplo, *Simplex-Centroid*, *Simplex-Lattice*, y con base en algún criterio de optimización, por ejemplo, *D-optimal*, de acuerdo con Ning et al. (2011b) tiene al menos dos desventajas. En primer lugar tienden a seleccionar mezclas en los límites de la región experimental, y en segundo, el diseño óptimo no es robusto ante la selección del modelo. Por ejemplo, en el diseño *Simplex-Centroid* en la Figura 2.5, los $n = 2^3 - 1 = 7$ puntos están distribuidos en los vértices $\{(1,0,0),(0,1,0),(0,0,1)\}$, en el centro de los lados $\{(1/2,1/2,0),(1/2,0,1/2),(0,1/2,1/2)$ y en el centroide $(1/3,1/3,1/3)$, es decir, sólo 1 de 7 puntos está dentro del *Simplex*. En el diseño *Simplex-Lattice* en la Figura 2.6 $\{3,5\}$, con $n = \dfrac{(3+5-1)!}{5!(3-1)!} = 21$ puntos, tiene 15 puntos de un total de 21 puntos en los extremos de la región experimental, y sólo 6 puntos en el interior del *Simplex*.

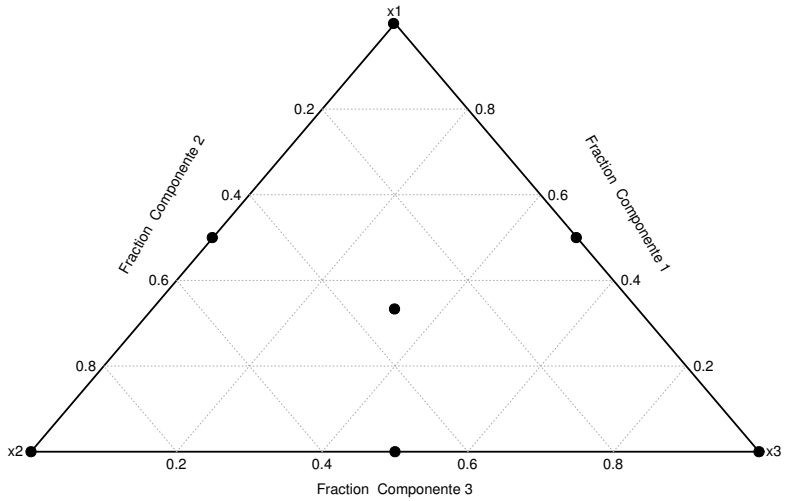

Figura 2.5: Diseño *Simplex-Centroid* con $q = 3$ componentes y 2 niveles.

2.3.5. Algoritmo NTLBG para generar diseños uniformes con mezclas

Se han propuesto métodos para crear diseños uniformes para experimentos con mezclas (también llamados diseños uniformes de mezclas). Por ejemplo, el método de transformación inversa mencionado en la Sección 2.3.1 para generar puntos en T_q a partir del hipercubo $\bar{I}^{q-1}$. Mediante estos métodos se crean diseños que son robustos ante la selección del modelo al *repartir* los puntos uniformemente dentro del *Símplex*.

Se debe notar que el método de transformación inversa no garantiza la uniformidad dentro del *Símplex* aún cuando los puntos sean uniformes dentro del hipercubo del que provienen (Ning et al., 2011a), por lo que se ha desarrollado algoritmos para redistribuir los puntos de un diseño uniforme y mejorar así su uniformidad, i.e., algoritmos para ajustar un diseño experimental con base en algún criterio o medida de discrepancia.

Uno de los algoritmos para mejorar la uniformidad de los puntos dentro del *Símplex* es el algoritmo propuesto por Ning et al. (2011a) conocido como algoritmo NTLBG (*Number-Theoretic LBG*), llamado así porque es la versión determinística del algoritmo LBG propuesto por Linde et al. (1980).

El algoritmo NTLBG propuesto por Ning et al. (2011a) es el siguiente:

1. Generar un conjunto *good lattice point* de tamaño $N >> n$, $\mathbf{M} = (m_{ij})_{N\times q-1}$, en el hipercubo $\bar{I}^{q-1}$ como se indica en la Sección 2.3.2.

2. Mapear el conjunto *good lattice point* al *Símplex* T_q, $\mathbf{T} = (t_{ij})_{N\times q}$, utilizando el método de

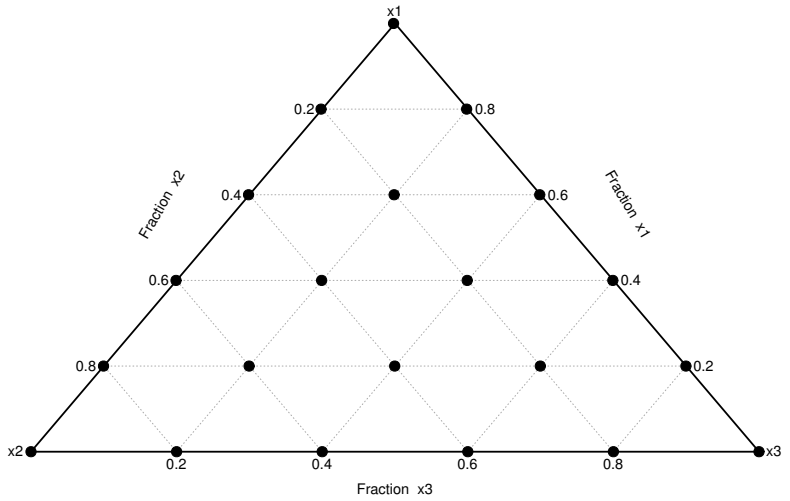

Figura 2.6: Diseño *Simplex-Lattice* con $q = 3$ componentes y $m = 5$ niveles.

transformación inversa[2] (Fang and Wang, 1993):

$$t_{ki} = \left(1 - m_{ki}^{1/(q-i)}\right) \qquad i = 1, \tag{2.24}$$

$$t_{ki} = \left(1 - m_{ki}^{1/(q-i)}\right) \prod_{j=1}^{i-1} m_{kj}^{1/(q-j)} \quad i = 2, \ldots, q-1, \tag{2.25}$$

$$t_{kq} = \prod_{j=1}^{q-1} m_{kj}^{1/(q-j)} \qquad k = 1, \ldots, N. \tag{2.26}$$

3. Generar aleatoriamente un diseño experimental de tamaño n en la región experimental T_q, que se denomina diseño experimental original D_0 para calcular el criterio *rmsd(D)* dado por:

$$rmsd(D) = \sqrt{\frac{1}{N} \sum_{k=1}^{N} \min_{1 \le j \le n} d_s^2(\mathbf{x}_j, \mathbf{t}_k)}, \tag{2.27}$$

donde $d(\cdot, \cdot)$ es la distancia euclidiana.

4. Crear n particiones de $\mathbf{T}$ de la siguiente manera:

Sea $\mathbf{T} = \{\mathbf{t}_k,\ k = 1, \ldots, N\}$ y el diseño actual $\mathbf{x} \in D_0 = (\mathbf{x}_1, \ldots, \mathbf{x}_n)'$, entonces

$$P_{x_j} = \{\mathbf{t}_k : d(\mathbf{t}_k, \mathbf{x}) = \min_{\mathbf{x}_{j'} \in D_0} d(\mathbf{t}_k, \mathbf{x}_{j'})\},\ j = 1, \ldots, n. \tag{2.28}$$

La partición $\{P_{x_j},\ j = 1, \ldots, n\}$ tiene dos propiedades: 1) $\cup_{\mathbf{x} \in D_0} P_{\mathbf{x}} = \mathbf{T}$ y 2) $P_{x_{j_1}} \cap P_{x_{j_2}} = \emptyset$

[2]También podría se necesario utilizar otro método cuando existan otras restricciones lineales.

para todo $j_1 \neq j_2$.

5. Calcular la media muestral de $P_{\mathbf{x}_j}$ mediante

$$\bar{\mathbf{x}}_j = \frac{1}{N_j} \sum_{\mathbf{t}_k \in P_{\mathbf{x}_j}} \mathbf{t}_k,$$

donde N_j es la cardinalidad de $P_{\mathbf{x}_j}$. Tomar $D = (\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_n)$ como el nuevo diseño y calcular el criterio *rmsd(D)*.

6. Verificar si $rmsd(D_0) - rmsd(D) > \alpha > 0$ (α establecido previamente), entonces $D_0 = D$, repetir pasos 4 a 6. De otro modo detener el algoritmo y tomar D_0 como el diseño final.

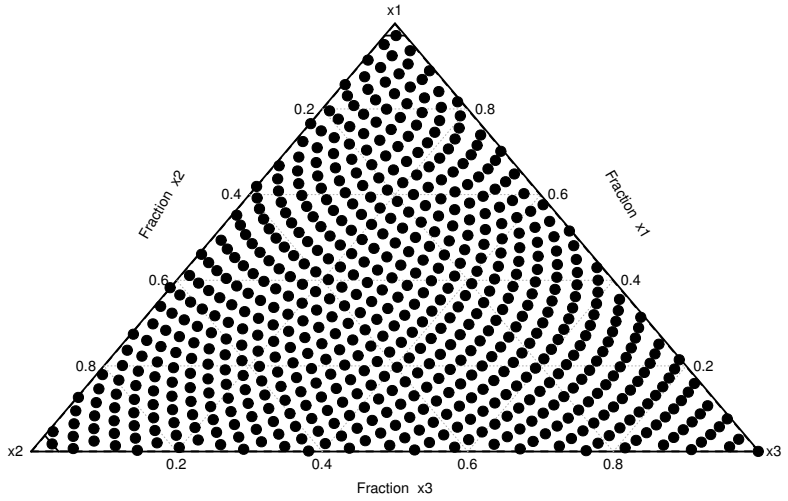

Figura 2.7: Conjunto *good lattice point* mapeado al símplex mediante el método de transformación inversa.

En la Figura 2.2 se muestra un conjunto *good lattice point* de tamaño 610 en $q-1=2$ dimensiones generado mediante el vector (610; 1, 377) (Fang and Wang, 1993, Tabla A.2), cuyo mapeo al *Símplex* T_3 mediante el método de transformación inversa se muestra en la Figura 2.7. A partir del conjunto de $n = 20$ vectores de entrenamiento de dimensión $q-1=2$, generados de manera aleatoria en el plano $\bar{I}^2$ y mapeados al *Símplex* T_3 que se muestra en la Figura 2.8 (Diseño inicial D_0), se genera el diseño final que se muestra en la Figura 2.9 mediante el algoritmo NTLBG.

Para la generación del diseño en la Figura 2.9, se utilizó el siguiente criterio que difiere del criterio original dado en el paso 6 del algoritmo:

$$\sum_{j=1}^{n} d_s^2(\mathbf{x}_j, \mathbf{x}'_j) > 0.00001,$$

donde $\mathbf{x}_j$ representa el j-ésimo punto en el diseño actual y $\mathbf{x}'_j$ el j-ésimo punto en el nuevo diseño.

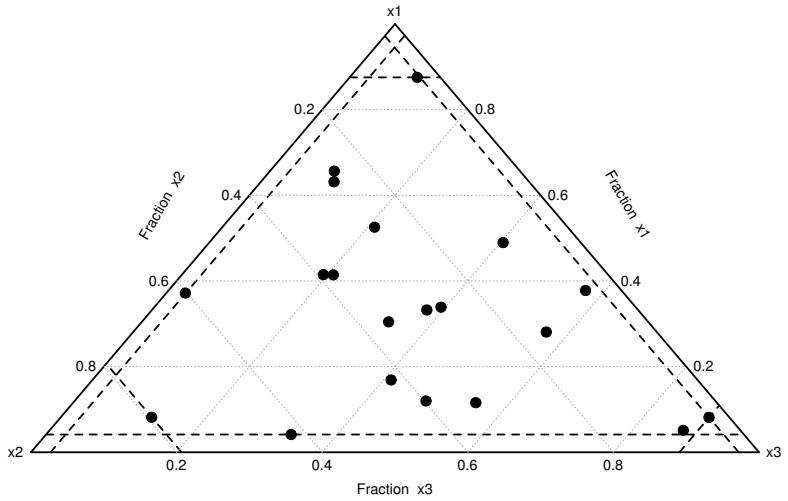

Figura 2.8: Vectores de entrenamiento. Diseño inicial D_0 generado de manera aleatoria en el plano $\bar{I}^2$ y mapeado al *Simplex* T_3 mediante el método de transformación inversa.

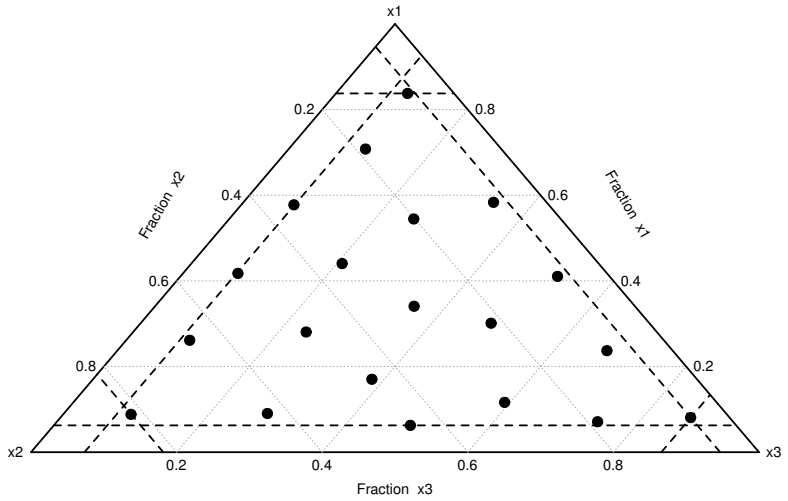

Figura 2.9: Vectores *quantizers*. Diseño final dentro del *Simplex* T_3 generado mediante el algoritmo NTLBG.

Se puede observar que el diseño generado mediante el algoritmo NTLBG (Figura 2.9) posee puntos más uniformemente distribuidos dentro de la región experimental T_3 que los puntos en el diseño aleatorio inicial D_0 (Figura 2.8). Además, a diferencia del diseño *Simplex-Lattice* en la Figura 2.6, todas las mezclas seleccionadas se encuentran dentro del *Símplex*.

Debido a la restricción lineal en la expresión 2.21, la matriz diseño de un modelo lineal con intercepto no es de rango completo. Cornell (2002) presenta varios métodos para lidiar con dicha

deficiencia del rango. Por ejemplo, el método *Slack-variable* elimina una variable de la matriz diseño; esta variable generalmente es el componente que se encuentra en mayor proporción ≥ 0.8 o el más inerte o inactivo, por ejemplo H_2O. Por otro lado, existe también el método de Claringbold (1955) que transforma los niveles de las q variables dependientes a $q-1$ variables independientes mediante una matriz de transformación ortogonal; este método consiste en dos pasos:

1. Cambiar el origen del sistema hacia el centroide del simplex, i.e., el punto en el que las proporciones de los componentes son $1/q$. Mediante esta simple transformación se cambia la escala para que los vértices no tengan coordenadas fraccionarias en el nuevo sistema $\bar{X}$:

$$\bar{\mathbf{X}}_\mathbf{i} = q\left(\mathbf{X}_i - \frac{1}{q}\right) = q\mathbf{X}_i - \mathbf{j}, \tag{2.29}$$

   donde $\mathbf{j}$ es un vector de unos de dimensión q.

2. Girar los ejes para que uno de los componentes, por ejemplo, el q-ésimo componente, sea ortogonal al *Símplex*. Esto se logra mediante una matriz Θ de transformación ortogonal.

$$\bar{\bar{\mathbf{X}}}_i = q'\bar{\mathbf{X}}_\mathbf{i}\Theta, \tag{2.30}$$

   con $q' = \frac{q(q-1)}{q}$ y Θ como se define en Claringbold (1955).

Siguiendo la idea de Claringbold (1955), el mapeo utilizado en el método de transformación inversa en el algoritmo NTLBG es fácilmente invertible para mapear del *Símplex* T_q al espacio ortogonal $\bar{I}^{q-1}$, invirtiendo las expresiones 2.24 a 2.26:

$$m_{ki} = (1-t_{ki})^{q-i} \qquad i = 1, \tag{2.31}$$

$$m_{ki} = \left(1 - \frac{t_{ki}}{1 - \sum_{j=1}^{i-1} t_{kj}}\right)^{q-i} \qquad i = 2,\ldots,q. \tag{2.32}$$

Mediante esta transformación, es posible trabajar con variables independientes de un modo similar al propuesto por Claringbold (1955).

En la Figura 2.10 se muestra el mapeo de los puntos en el *Simplex* T_q de la Figura 2.9 al espacio ortogonal $\bar{I}^{q-1}$.

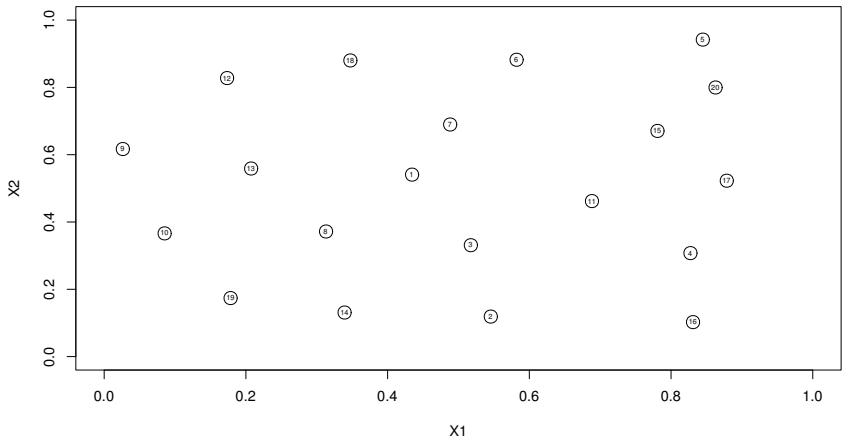

Figura 2.10: Mapeo del *Símplex* T_3 al espacio ortogonal $\bar{I}^{3-1}$

2.4. Modelo estadístico

2.4.1. Proceso Gaussiano

El modelo estadístico más usado para modelar la salida de experimentos por computadora es el Proceso Gaussiano (Santner et al., 2003; Fang et al., 2005). Este modelo es conceptulamente sencillo y permite incorporar información *a priori* de la correlación espacial a través de la matriz de correlación.

Formalmente, un proceso Gaussiano es una colección de variables aleatorias, $Z(\mathbf{x})$, indexadas por $\mathbf{x}$, que tienen una distribución conjunta normal multivariada para cualquier subconjunto finito de índices (Stein, 1999). Este proceso está especificado por una función de media $\mu(\mathbf{x})$ y una función de correlación, $K(\mathbf{x}_j, \mathbf{x}_k)$, dada por:

$$\mu(\mathbf{x}) = E(Z(\mathbf{x})) \tag{2.33}$$

$$K(\mathbf{x}_j, \mathbf{x}_k) = \frac{1}{\sigma^2} E\left(\left[Z(\mathbf{x}_j) - \mu(\mathbf{x}_j) \right] \left[Z(\mathbf{x}_k) - \mu(\mathbf{x}_k) \right]^\top \right) \tag{2.34}$$

La función $\mu(\mathbf{x})$ generalmente se especifica como $\mu(\mathbf{x}) = \beta^\top f(\mathbf{x})$ pero se puede especificar de manera más general como $\xi(\mathbf{x}, \beta)$.

Típicamente, la función de correlación se puede descomponer en

$$K(\mathbf{x}_j, \mathbf{x}_k | g) = K^*(\mathbf{x}_j, \mathbf{x}_k) + g\delta_{j,k}, \tag{2.35}$$

donde $\delta_{\cdot,\cdot}$ es la función delta de Kronecker

$$\delta_{j,k} = \begin{cases} 1 \text{ si } j = k \\ 0 \text{ si } j \neq k \end{cases} \tag{2.36}$$

y $K^*(\cdot,\cdot)$ es una función de correlación generalmente paramétrica adecuada, i.e., simétrica, positiva semidefinida y contínua en $K^*(\mathbf{x}_j, \mathbf{x}_k)$. Además debe cumplir con $K^*(\mathbf{x}_j, \mathbf{x}_j) = 1$ (Santner et al., 2003, Pág. 33-34). El término de error puro g (también conocido como *nugget*) siempre debe ser positivo ($g > 0$) y tiene dos propósitos. En primer lugar proporciona un mecanismo para introducir un término de error puro en el proceso estocástico

$$Z(\mathbf{x}) = \xi(\mathbf{x}, \beta) + w(\mathbf{x}) + \eta(\mathbf{x}) \tag{2.37}$$

donde $w(\cdot)$ es un proceso con correlaciones gobernadas por la función $K^*(\cdot,\cdot)$ y $\eta(\cdot)$ es simplemente ruido blanco (ruido Gaussiano) que introduce el término g en el modelo. En segundo lugar, previene que la matriz de correlación $K(\mathbf{x}_j, \mathbf{x}_k)$ sea singular.

La función de correlación $K^*(\cdot,\cdot)$ generalmente se especifica a través de una estructura paramétrica de baja dimension, que garantiza que sea simétrica y positiva definida. Por ejemplo, la familia Matérn que incluye a la exponencial, Gaussiana, etc.

La función de correlación de Matérn entre dos puntos $\mathbf{x}_j$ y $\mathbf{x}_k$ separadas a una distancia $d_{(\mathbf{x}_j,\mathbf{x}_k)}$ es:

$$K^*(\mathbf{x}_j, \mathbf{x}_k | \nu, \theta) = \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(\sqrt{2\nu} \frac{d_{(\mathbf{x}_j,\mathbf{x}_k)}}{\theta} \right)^{\nu} K_\nu \left(\sqrt{2\nu} \frac{d_{(\mathbf{x}_j,\mathbf{x}_k)}}{\theta} \right) \tag{2.38}$$

donde $\Gamma(\cdot)$ es la función gama, $K_\nu(\cdot)$ es la función de Bessel modificada de segundo tipo, θ y ν son parámetros no negativos de la correlación. Si la distancia $d_{(\mathbf{x}_j,\mathbf{x}_k)}$ es Euclideana, entonces la función de correlación de Matérn también es isotrópica. En general, los modelos isotrópicos no son útiles cuando los factores (variables de entrada) se encuentran en diferentes escalas (Santner et al., 2003).

Cuando $\nu \to \infty$ la función de correlación de Matérn converge a la función de covarianza exponencial cuadrática (función de correlación Gaussiana)

$$K^*(\mathbf{x}_j, \mathbf{x}_k | \theta) = \exp\left(-\frac{d^2_{(\mathbf{x}_j,\mathbf{x}_k)}}{2\theta^2} \right). \tag{2.39}$$

Cuando $\nu = 1/2$ la función de correlación de Matérn es idéntica a la función de correlación exponencial

$$K^*(\mathbf{x}_j, \mathbf{x}_k | \theta) = \exp\left(-\frac{d_{(\mathbf{x}_j,\mathbf{x}_k)}}{\theta} \right). \tag{2.40}$$

Siguiendo el desarrollo de Gramacy and Lee (2008a, 2009), si los vectores $\mathbf{x}$ de variables explicativas tienen dimensión m_x, se especifica la función de correlación exponencial separable:

$$K^*(\mathbf{x}_j, \mathbf{x}_k|\theta) = \exp\left(-\sum_{i=1}^{m_x} \frac{|x_{ij} - x_{ik}|^{p_0}}{\theta_i}\right) \tag{2.41}$$

con $0 < p_0 \leq 2$ y $\theta_i > 0$ para $i = 1, \ldots, m_x$, que se utiliza comúnmente para modelar experimentos por computadora. En la literatura, la potencia p_0 se le conoce como parámetro de forma y θ se conoce como parámetro de escala (Santner et al., 2003).

En la práctica, los procesos Gaussianos son *no singulares*, i.e., para cualquier elección de factores, la matriz de correlación de la distribución normal multivariada de $Z(\mathbf{x})$ es *no singular*.

Un caso particular del proceso Gaussiano es el proceso con $\mu(\mathbf{x}) = f(\mathbf{x})^\top \beta$ y función de correlación Gaussiana (2.39) y que se representa como

$$Z(\mathbf{x}) = f(\mathbf{x})^\top \beta + w(\mathbf{x}) + \eta(\mathbf{x}), \tag{2.42}$$

donde $f(\mathbf{x})$ es un vector definido en función de las coordenadas de $\mathbf{x}$, $w(\mathbf{x}) \sim N(\mathbf{0}, K^*(\mathbf{x}_j, \mathbf{x}_k))$ y $\eta(\mathbf{x}) \sim N(\mathbf{0}, g\mathbf{I})$.

2.4.2. Proceso Gaussiano arbolado

En aplicaciones, el proceso Gaussiano se asume estacionario, i.e., una misma estructura de covarianza dentro de todo el espacio $\mathcal{B}$ de los factores se considera válida (Gramacy and Lee, 2008a, 2009). Debido a esta suposición todos los puntos de respuesta contribuyen a la estimación del error local a través de la función de correlación $K(\cdot, \cdot)$. Sin embargo, esta es una suposición muy fuerte que facilita la estimación debido a la estacionalidad de la matriz de correlación, pero no permite incluir matrices de covarianza distintas dentro del espacio factorial.

Por esta razón, Gramacy and Lee (2008a) y Konomi et al. (2014a) proporcionan una forma de *lidiar* con procesos Gaussianos no estacionarios a través de la creación de particiones del espacio factorial utilizando un árbol de particiones similar al modelo de Árbol de clasificación y regresión (CART) (Breiman et al., 1984; Chipman et al., 1998) y al modelo arbolado Bayesiano (Chipman et al., 2002) para determinar particiones; típicamente, en cada partición se ajusta un proceso Gaussiano estacionario independiente con una cantidad mínima de datos. Este modelo tiene dos componentes: un árbol $\mathcal{T}$ con $R \geq 1$ nodos terminales, y un parámetro $\Theta = \{\theta_v\}_{v=1}^R$ que asocia el parámetro θ_v con el v-ésimo nodo terminal. Si $\mathbf{x}$ se encuentra dentro de la región correspondiente al v-ésimo nodo, entonces $y|\mathbf{x}$ tiene distribución $f(y|\theta_v)$ donde f representa la familia paramétrica indexada por θ_v (Chipman et al., 1998, 2002). Este enfoque además reduce la carga computacional

(en relación a un solo proceso Gaussiano para todos los datos) porque en cada partición se invierte una matriz con un menor número de puntos (Gramacy and Lee, 2008a).

Un árbol binario $\mathcal{T}$ subdivide el espacio de la siguiente manera: cada nodo interno (representado en la Figura 2.12 como $\{u, s_u\}$) tiene asociada una regla de partición que utiliza un factor (la u-ésima dimensión) para asignar las observaciones a sus nodos hijos. De este modo, los nodos terminales son particiones del espacio de las observaciones de acuerdo con las subdivisiones definidas por las reglas de partición. Para factores cuantitativos, la regla de partición es la siguiente:

- si $x_u \leq s_u$, entonces asigna las observaciones que cumplen con la condición a su nodo hijo izquierdo;
- las observaciones que no cumplen con la condición, i.e., $x_u > s_u$, a su nodo hijo derecho.

De este modo las particiones resultantes son paralelas a los ejes de los factores y recursivas, es decir, cada nueva partición es una subpartición de otra existente. Esta forma es suficientemente general como para manejar cualquier función de partición de la forma $h(\mathbf{x}) \leq s_u$ y $h(\mathbf{x}) > s_u$ al tratar $h(\mathbf{x})$ como un factor. De modo similar, para factores categóricos, la regla de partición es la siguiente:

- si $x_u \in C$, entonces asigna las observaciones que cumplen con la condición a su nodo hijo izquierdo;
- las observaciones que cumplen con la condición, i.e., $x_u \notin C$, a su nodo hijo derecho.

(Breiman et al., 1984; Chipman et al., 1998, 2002).

Por ejemplo, la Figura 2.11 muestra gráficamente un árbol $\mathcal{T}$ con $R = 3$ particiones binarias en el espacio $\bar{I}^2$. Los puntos $\mathbf{x}_i = (x_{i1}, x_{i2})^\top$ corresponden al diseño de las Figuras 2.9 y 2.10.

Para crear las primeras dos particiones se aplica la regla $x_1 \leq 0.2$, y se obtiene las particiones $a = \{\mathbf{x} : 0 \leq x_1 \leq 0.2,\ 0 \leq x_2 \leq 1\}$ (partición a en la Figura 2.11) y $d = \{\mathbf{x} : 0.2 < x_1 \leq 1,\ 0 \leq x_2 \leq 1\}$ (no se muestra). Para la segunda y tercera partición se aplica la regla $x_2 \leq 0.5$ dentro de la partición d y se obtiene las particiones $b = \{\mathbf{x} : 0.2 < x_1 \leq 1,\ 0 \leq x_2 \leq 0.5\}$ (partición b en la Figura 2.11) y $c = \{\mathbf{x} : 0.2 < x_1 \leq 1,\ 0.5 < x_2 \leq 1\}$ (partición c en la Figura 2.11). La representación gráfica del árbol de particiones binarias correspondiente a la Figura 2.11 se muestra en la Figura 2.12.

Las particiones binarias son preferidas sobre otro tipo de particiones múltiples porque pueden ser obtenidas a partir particiones binarias recursivas (Breiman et al., 1984). Aunque el método es bastante bueno para ajustar modelos en los que la no estacionalidad es paralela a los ejes, Gramacy and Lee (2008a,b) mostraron que también es adecuado en casos donde la no estacionalidad es más general.

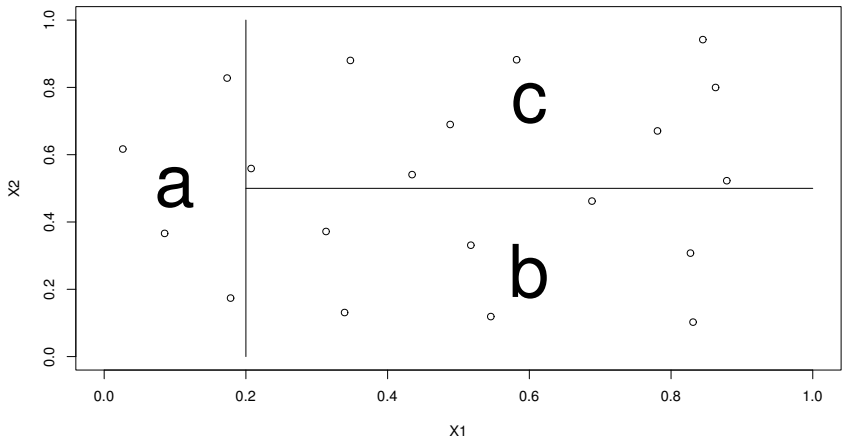

Figura 2.11: Espacio creado por los factores $\mathbf{x} = (x_1, x_2)^\top \in \bar{I}^2$ con tres particiones binarias.

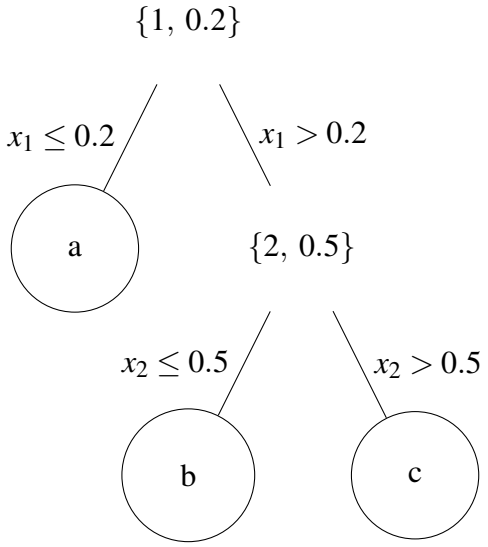

Figura 2.12: Árbol de particiones binarias en el espacio creado por $\mathbf{x} = (x_1, x_2)^\top \in \bar{I}^2$. Los nodos internos $\{u, s_u\}$ corresponden a la regla de partición $x_u \leq s_u$ en la u-ésima dimensión $u = \{1, 2\}$.

Modelo jerárquico

Para el proceso Gaussiano arbolado con $\{r_v\}_{v=1}^R$ particiones (Gramacy and Lee, 2008a, 2009), $\{y_{vj}\}_{v=1}^R$ son las observaciones dentro cada una de las v particiones $j = 1, \ldots, n_v$, entonces se define

$$\mathbf{Y} \equiv (Y_1, \ldots, Y_R)^\top, \text{ donde } Y_v \equiv (y_{v1}, \ldots, y_{vn_v})^\top, \tag{2.43}$$

y $\mathbf{X}$ y X_v de manera análoga. Dado $(\Theta, \mathcal{T})$, los valores de y dentro de cada nodo terminal son independientes e idénticamente distribuidos y los valores de y entre nodos son independientes. De tal forma que

$$p(\mathbf{Y}|X,\Theta,\mathcal{T}) = \prod_{v=1}^{R} f(Y_v|\theta_v) = \prod_{v=1}^{R}\prod_{j=1}^{n_v} f(y_{vj}|\theta_v) \tag{2.44}$$

El modelo condicionado en 2.44 queda completamente especificado si se asume una distribución *a priori* para los parámetros Θ y $\mathcal{T}$, $p(\Theta,\mathcal{T})$ (Gramacy and Lee, 2008a; Chipman et al., 1998). Siguiendo Chipman et al. (1998) la distribución conjunta de Θ y $\mathcal{T}$, está dada por

$$p(\Theta,\mathcal{T}) = p(\Theta|\mathcal{T})p(\mathcal{T}), \tag{2.45}$$

para especificar $p(\mathcal{T})$ y $p(\Theta|\mathcal{T})$ por separado. Una característica de este enfoque es que es posible especificar una distribución *a priori* para Θ que permita formas analíticas y que facilite la estimación de la distribución *a posteriori* (Chipman et al., 1998; Gramacy and Lee, 2008a).

El proceso Gaussiano se especifica con media $\mu(\mathbf{x}) = \mathbf{F}_v\beta_v$, $\mathbf{F}_v = (\mathbf{1};\mathbf{X}_v)$, y matriz de covarianzas $\sigma^2\mathbf{K}_v$ independientes entre las particiones, con $\mathbf{K}_v = K(\mathbf{x}_j,\mathbf{x}_k|g)$ como en la expresión 2.35 y $K^*(\mathbf{x}_j,\mathbf{x}_k)$ en la familia exponencial separable como en la expresión 2.41.

Las distribuciones se especifican de la siguiente manera:

$$Y_v|\beta_v,\sigma_v^2,\mathbf{K}_v \sim N_{n_v}(\mathbf{F}_v\beta_v,\sigma_v^2\mathbf{K}_v), \tag{2.46}$$

$$\beta_0 \sim N_m(\mu,\mathbf{B}),\ \sigma_v^2 \sim IG(\alpha_\sigma/2, q_\sigma/2), \tag{2.47}$$

$$\beta_v|\sigma_v^2,\tau_v^2,\mathbf{W},\beta_0 \sim N_m(\beta_0,\sigma_v^2\tau_v^2\mathbf{W}),\ \text{y} \tag{2.48}$$

$$\mathbf{W}^{-1} \sim W((\rho\mathbf{V})^{-1},\rho),\ \tau_v^2 \sim IG(\alpha_\tau/2, q_\tau/2) \tag{2.49}$$

donde $\mathbf{W}$ es una matriz de dimensión $m\times m$; $m = m_x + 1$; N, IG y W son las distribuciones Normal (multivariada), Gamma Inversa y Wishart, respectivamente. Los datos $\{\mathbf{X},Y\}_v$ en la región r_v se utilizan para estimar los parámetros $\theta_v = \{\beta,\sigma,\mathbf{K},\tau\}_v$ del modelos dentro de cada región; los parámetros en las distribuciones *a priori* sólo dependen de $\{\theta_v\}_{v=1}^R$.(Gramacy and Lee, 2009)

Con las distribuciones *a priori* especificadas, la predicción de la respuesta en una nueva configuración de los factores $\mathbf{x} \in r_v$, sigue una distribución normal, con media y varianza:

$$\hat{y}(\mathbf{x}) = \mathrm{E}[Y(\mathbf{x})|Y_v,\mathbf{x}\in r_v] = \mathbf{f}(\mathbf{x})^\top\tilde{\beta}_v + \mathbf{k}_v^\top(\mathbf{x})\mathbf{K}_v^{-1}(Y_v - \mathbf{F}_v\tilde{\beta}_v), \tag{2.50}$$

$$\hat{\sigma}^2(\mathbf{x}) = \mathrm{Var}[Y(\mathbf{x})|Y_v,\mathbf{x}\in r_v] = \sigma_v^2[\kappa(\mathbf{x},\mathbf{x}) - \mathbf{q}_v^\top(\mathbf{x})\mathbf{C}_v^{-1}\mathbf{q}_v(\mathbf{x})]. \tag{2.51}$$

Donde

$$\mathbf{C}^{-1} = (\mathbf{K}_v + \tau_v^2\mathbf{F}_v\mathbf{W}\mathbf{F}_v^\top)^{-1}, \tag{2.52}$$

$$\mathbf{q}_v(\mathbf{x}) = \mathbf{k}_v(\mathbf{x}) + \tau_v^2 \mathbf{F}_v \mathbf{W}_v \mathbf{f}(\mathbf{x}),\ \text{y} \tag{2.53}$$

$$\kappa(\mathbf{x},\mathbf{y}) = K_v(\mathbf{x},\mathbf{y}) + \tau_v^2 \mathbf{f}^\top(\mathbf{x}) \mathbf{W} \mathbf{f}(\mathbf{y}). \tag{2.54}$$

Con $\mathbf{f}(\mathbf{x})^\top = (1, \mathbf{x}^\top)$ y $\mathbf{k}_v(\mathbf{x})$ un vector de dimensión n_v con $k_{vj}(\mathbf{x}) = K_v(\mathbf{x}, \mathbf{x}_j)$ para toda $\mathbf{x}_j \in X_v$ y $\tilde{\beta}_v$ como la estimación promedio de la distribución *a posteriori* de β_v. El proceso global no es estacionario debido al árbol $\mathcal{T}$, por lo tanto $\hat{\sigma}^2(\mathbf{x})$ es específica para cada región (Gramacy and Lee, 2008a, 2009).

Proceso para generar el árbol

El enfoque de Chipman et al. (1998, 2002) requiere la especificación de una distribución *a priori* sobre la distribución condicional en los nodos terminales del árbol (expresión 2.45) que, junto con la verosimilitud del modelo arbolado (expresión 2.44), produce una distribución *a posteriori* sobre el conjunto de modelos arbolados:

$$p(\Theta, \mathcal{T}|Y,X) \propto p(Y|X,\Theta,\mathcal{T}) p(\Theta|\mathcal{T}) p(\mathcal{T}). \tag{2.55}$$

Aunque es posible que dicha distribución *a posteriori* no se pueda calcular en problemas no triviales, se puede utilizar el algoritmo de Metropolis-Hastings para explorar la distribución *a posteriori* y guiar la búsqueda estocástica hacia modelos prometedores, i.e., modelos con alta probabilidad *a posteriori*.

Especificación de la distribución de $p(\mathcal{T})$.

En lo sucesivo se sigue el desarrollo de Chipman et al. (1998, 2002). En lugar de especificar una expresión en forma cerrada para $p(\mathcal{T})$, esta se especifica implícitamente mediante un proceso estocástico para generar el árbol. Cada realización de dicho proceso es una muestra de esta distribución *a priori*.

Para muestrear de $p(\mathcal{T})$, se inicia con un árbol sin particiones. El árbol crece partiendo aleatoriamente los nodos terminales mediante reglas de partición. Por lo tanto, el proceso de crecimiento está especificado por dos funciones: una función que asigna probabilidades a los factores que se pueden partir $p_{SPLIT}(\eta, \mathcal{T})$, y una función que asigna probabilidades a los valores disponibles dado el factor que se partirá, $p_{RULE}(\rho|\eta, \mathcal{T})$. En otras palabras, para un árbol intermedio $\mathcal{T}$, $p_{SPLIT}(\eta, \mathcal{T})$ es la probabilidad de que el nodo terminal η sea dividido, y $p_{RULE}(\rho|\eta, \mathcal{T})$ es la probabilidad de asignar la regla de partición ρ a η, si el nodo se divide.

El proceso estocástico para muestrear un árbol a partir de esta distribución *a priori* se describe de la siguiente manera:

1. Comenzar con un árbol $\mathcal{T}$ con un solo nodo, η.

2. Proponer dividir el nodo terminal η con probabilidad $p_{SPLIT}(\eta, \mathcal{T})$.

3. Asignar una regla de partición ρ de acuerdo con la distribución $p_{RULE}(\rho|\eta, \mathcal{T})$ para crear los nodos hijos izquierdo y derecho. Se acepta la división de η con probabilidad α. Denotar como $\mathcal{T}$ al nuevo árbol y aplicar los pasos 2 y 3 con η igual a los nuevos nodos hijos izquierdo y derecho.

Mediante la función $p_{SPLIT}(\eta, \mathcal{T})$ es posible controlar el tamaño específico del árbol. Por ejemplo, Chipman et al. (1998) propone la distribución *a priori* conjunta para η y $\mathcal{T}$ dada por:

$$p_{SPLIT}(\eta, \mathcal{T}) = \alpha(1+q_\eta)^{-\beta}, \tag{2.56}$$

donde $q_\eta \in \mathbb{N}_0$ denota la profundidad de $\eta \in \mathcal{T}$, $0 < \alpha \leq 1$ y $\beta \geq 0$ son parámetros seleccionados para dar un tamaño adecuado y propagación a la distribución de los árboles. El parámetro α se interpreta como una probabilidad "base" de crecer un árbol mediante la partición de un nodo terminal, y β como la velocidad con la que decrece conforme crece el árbol.

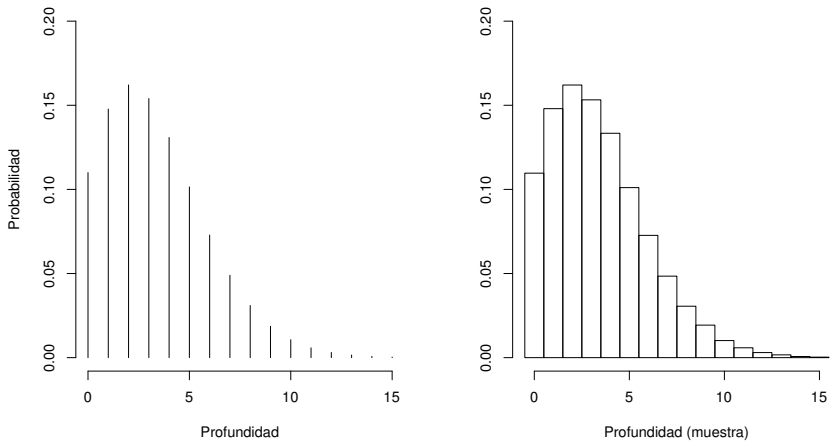

Figura 2.13: Distribución *a priori* $p(\mathcal{T})$ con $\alpha = 0.95$, $\beta = 0.5$. Izquierda: diagrama de densidad discreta; Derecha: histograma de una muestra tamaño $100,000$.

Bajo esta especificación, p_{SPLIT} es una función decreciente en q_η y β, que asigna menos densidad a los nodos más profundos, tal como se muestra en la Figura 2.13, que representa la densidad $p_{SPLIT}(\eta, \mathcal{T}) = 0.95(1+q_\eta)^{-0.5}$, donde la media de la distribución es 3.41 nodos terminales.

Como se describió al inicio de esta sección, una regla de corte ρ se determina mediante la selección de una dimensión x_u y un valor de corte s_u ($\{u, s_u\}$) o un subconjunto C de una categoría

($\{u, C\}$), si la dimensión x_u es cuantitativa o categórica, respectivamente. Por lo tanto, la función $p_{RULE}(\rho|\eta, \mathcal{T})$ se puede describir por una distribución sobre el conjunto de dimensiones disponibles $\{x_i\}_{i=1}^{s}$ y, condicional en la dimensión x_u, una distribución para el conjunto de valores de corte s_u, o subcategorías, disponibles. Por "disponibles" se entiende aquellos valores s_u o subconjuntos C que no produzcan nodos terminales vacíos.

En la práctica se considera sólo la especificación de una distribución *a priori* para la cual el conjunto de posibles valores de corte es finito. Por lo tanto, cada $p_{RULE}(\rho|\eta, \mathcal{T})$ siempre es una distribución discreta. Por consiguiente, el soporte de $p(\mathcal{T})$ siempre tendrá un número finito de árboles. Nótese que como las reglas de corte dependen de X, la distribución *a priori* también dependerá de X.

La distribución $p_{RULE}(\rho|\eta, \mathcal{T})$ utilizada por Chipman et al. (1998, 2002) y Gramacy and Lee (2008a), es una distribución uniforme para seleccionar x_u, y una distribución uniforme para seleccionar s_u, o C si es cuantitativa, sobre los valores observados de la dimensión x_u seleccionada previamente.

Búsqueda estocástica de la distribución *a posteriori*

Con el desarrollo de los modelos arbolados no es posible obtener formas cerradas para la distribución *a posteriori* en 2.55. Sin embargo es posible integrar analíticamente con respecto al parámetro Θ, i.e., $p(Y|X, \mathcal{T}) = \int p(Y|X, \Theta, \mathcal{T}) p(\Theta|\mathcal{T}) d\Theta$, y obtener un forma cerrada (Chipman et al., 2002; Gramacy and Lee, 2008a). Una vez marginalizada, $p(Y|X, \mathcal{T})$ se puede combinar con una distribución *a priori* $p(\mathcal{T})$ para explorar la distribución *a posteriori* utilizando el algoritmo de Metropolis-Hastings; dicho algoritmo simula una cadena de Markov para los árboles

$$\mathcal{T}^0, \mathcal{T}^1, \mathcal{T}^2, \ldots, \tag{2.57}$$

que convergen a la distribución *a posteriori* $p(\mathcal{T}|Y, X)$ (Chipman et al., 1998). Como dicha simulación tiende a gravitar hacia regiones con alta probabilidad *a posteriori*, se puede utilizar la simulación para hacer una búsqueda estocástica de los árboles con alta probabilidad *a posteriori*.

Chipman et al. (1998) presentan el siguiente algoritmo para simular la cadena de Markov $\mathcal{T}^0, \mathcal{T}^1, \mathcal{T}^2, \ldots$. Iniciar con un árbol $\mathcal{T}^0$, simular iterativamente la transición de $\mathcal{T}^i$ a $\mathcal{T}^{i+1}$ mediante los dos pasos siguientes:

1. Generar un candidato $\mathcal{T}^*$ con distribución de probabilidad $q(\mathcal{T}^i, \mathcal{T}^*)$.

2. Establecer $\mathcal{T}^{i+1} = \mathcal{T}^*$ con probabilidad

$$\alpha(T^i, T^*) = min \left\{ \frac{q(T^*, T^i)}{q(T^i, T^*)} \frac{p(Y|X, \mathcal{T}^*) p(\mathcal{T}^*)}{p(Y|X, \mathcal{T}^i) p(\mathcal{T}^i)} \right\} \tag{2.58}$$

de otro modo, establecer $\mathcal{T}^{i+1} = \mathcal{T}^i$.

Bajo condiciones débiles (Tierney (1994) en Chipman et al. (1998)), la sucesión en 2.57 obtenida mediante este algoritmo converge en distribución a $p(\mathcal{T}|Y,X)$. En este algoritmo, la constante de normalización no es necesaria para calcular 2.58.

Para implementar el algoritmo, Chipman et al. (1998) consideraron kernels $q(\mathcal{T},\mathcal{T}^*)$ que permiten generar $\mathcal{T}^*$ a partir de $\mathcal{T}$ seleccionando de manera aleatoria entre cinco[3] movimientos posibles que consisten en:

GROW: Seleccionar aleatoriamente un nodo terminal. Dividirlo en dos nuevos nodos asignándole aleatoriamente una regla de partición de acuerdo con la regla de partición p_{RULE} utilizado en la distribución *a priori*.

PRUNE: Seleccionar aleatoriamente un padre de dos nodos terminales y convertirlo en un nodo terminal colapsando los nodos hijos.

CHANGE: Seleccionar aleatoriamente un nodo interno y reasignar aleatoriamente una regla de partición de acuerdo con p_{RULE} utilizado en la *a priori*.

SWAP: Seleccionar aleatoriamente un par de nodos internos padre-hijo. Intercambiar las reglas de partición a menos que el otro hijo tenga una regla idéntica, en cuyo caso, intercambiar la regla de partición del padre con la de ambos hijos.

ROTATE: Seleccionar aleatoriamente dos nodos padre-hijo internos y rotar la regla de partición. Gramacy and Lee (2008b) propuso este movimiento como alternativa a SWAP cuando este último se aplica a pares padre-hijo que tienen reglas de partición en la misma dimensión x_u y que generan, por tanto, nodos vacíos. En este movimiento las particiones en los nodos terminales no se modifican, y por lo tanto la razón de verosimilitudes de una partición propuesta siempre es 1 en la expresión 2.58. La única parte "activa" del algoritmo de Metropolis-Hastings en este movimiento es el cociente de distribuciones *a priori* sobre $\mathcal{T}$, que favorece particiones con menor profundidad.

Los movimientos GROW, CHANGE y SWAP se restringen a reglas de partición que no producen árboles con nodos terminales vacíos y que tienen un número mínimo de observaciones en cada nodo terminal. (Chipman et al., 1998, 2002; Gramacy and Lee, 2008a)

El kernel de transición produce una cadena de Markov reversible, lo cual es una característica deseable, ya que se interpreta como que la cadena consiste de movimientos antagónicos. De hecho, el movimiento *GROW* tiene su contraparte *PRUNE*, y *CHANGE* tiene su contraparte *SWAP*.

[3]Gramacy and Lee (2008b) propuso el quinto movimiento, ROTATE

(Chipman et al., 1998) El movimiento *ROTATE* propuesto por (Gramacy and Lee, 2008b) no tiene contraparte. Otra característica deseable es que $\alpha(\cdot,\cdot)$ del algoritmo de Metropolis-Hastings es fácil de calcular, i.e., al utilizar p_{RULE} en el movimiento *GROW* se cancelan varios términos entre $p(\mathcal{T}^*)$ y $q(\mathcal{T},\mathcal{T}^*)$ en la expresión 2.58. Asimismo, para el movimiento *PRUNE*, se cancelan términos entre $p(\mathcal{T})$ y $q(\mathcal{T}^*,\mathcal{T})$. En los movimientos *CHANGE* y *SWAP* el cociente que involucra a q siempre es 1 (Chipman et al., 1998).

La dependencia en la estructura del árbol, $\mathcal{T}$, se puede integrar sobre todos los árboles posibles mediante *reversible–jump* MCMC (RJ-MCMC) (Gramacy and Lee, 2008a; Green, 1995). La predicción se condiciona en la estructura del árbol y se promedia con las distribuciones *a posteriori* para considerar completamente la incertidumbre de la selección del modelo (Gramacy and Lee, 2008a; Hoeting et al., 1999).

2.4.3. Diseños adaptativos

Gramacy and Lee (2009) propusieron un enfoque novedoso que permite "crecer" los diseños experimentales. Esta característica además permite considerar los casos en los que los datos provienen de un proceso no estacionario. Dicho enfoque "aprende activamente" de los datos y explora más intensamente las regiones del espacio que son más complicadas o interesantes, i.e., regiones donde existe mayor incertidumbre o estructuras más complejas.

El Aprendizaje activo incluye dos componentes: un diseño experimental inicial y el ajuste de un modelo para los datos obtenidos en el diseño. Adicionalmente, se requiere una lista de condiciones experimentales $\tilde{\mathbf{X}}$ en las que se predice la respuesta usando el modelo ajustado y de los cuales se debe añadir al diseño si cumplen con ciertas características. Las condiciones experimentales que se añadirán al diseño experimental $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ se seleccionan mediante uno de los siguientes criterios (Gramacy and Lee, 2008b, 2009):

ALM *Active Learning MacKay*: trata de maximizar la información obtenida sobre los parámetros del modelo seleccionando las condiciones experimentales $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ que tiene la mayor desviación estándar de predicción (MacKay, 1992). Las muestras MCMC de la distribución *a posteriori* proporcionan una estimación adecuada de la varianza en puntos específicos, a través del rango de los cuantiles de predicción.

ALC *Active Learning Cohn*: selecciona las condiciones experimentales $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ que maximizan la reducción esperada del error cuadrado promediado sobre el espacio de las variables de entrada (Cohn et al., 1996). Gramacy and Lee (2009) proporcionan el desarrollo para obtener las expresiones para calcular el estadístico ALC: 1) bajo el modelo especificado en 2.46 a 2.49,

y 2) bajo un modelo lineal. Estas se muestra en las expresiones 2.59 y 2.60, respectivamente.

$$\Delta\hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) = \frac{\sigma^2[\mathbf{q}_N^\top(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{x}) - \kappa(\mathbf{x},\mathbf{y})]^2}{\kappa(\mathbf{x},\mathbf{x}) - \mathbf{q}_N^\top(\mathbf{x})\mathbf{C}_N^{-1}\mathbf{q}_N(x)}, \tag{2.59}$$

$$\Delta\hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) = \frac{\sigma^2[\mathbf{f}^\top(\mathbf{y})\mathbf{V}_{\tilde{\beta}_N}\mathbf{f}(\mathbf{x})]^2}{1+g+\mathbf{f}^\top(\mathbf{x})\mathbf{V}_{\tilde{\beta}_N}\mathbf{f}(\mathbf{x})}, \tag{2.60}$$

donde $\mathbf{C}^{-1}$, $\mathbf{q}_N$ y $\kappa(\cdot,\cdot)$ son como se define en las expresiones 2.52, 2.53 y 2.54, dentro de cada partición; y $\mathbf{V}_{\tilde{\beta}_n} = (\mathbf{F}_v^\top\mathbf{K}_v^{-1}\mathbf{F}_v + \mathbf{W}^{-1}/\tau_v^2)^{-1}$ como en Gramacy and Lee (2008b).

Se debe notar que el desempeño del proceso de aprendizaje activo depende de la calidad de la lista de candidatos $\tilde{\mathbf{X}}$. Esta puede ser generada utilizando cualquier método, como por ejemplo diseño hipercubo latino, *good lattice point*, máxima entropía, etc. Además, la lista de candidatos no debe ser muy densa porque tenderá a seleccionar candidatos muy cercanos entre sí debido a que los criterios arriba mencionados generarán resultados muy similares (Gramacy and Lee, 2009).

Finalmente, el diseño inicial debe tener suficientes observaciones para ajustar un modelo dentro de cada partición. Konomi et al. (2014b) sugieren que dentro de cada partición exista un mínimo de $n_v = m+k$ observaciones, donde $m = m_x+1$ es la dimensión del vector de variables explicativas $\mathbf{x}$ más el intercepto y k es el número de respuestas.

Capítulo 3

Objetivos e hipótesis

3.1. Objetivo general.

1. Proponer una metodología que permita explorar el espacio experimental (ortogonal y *Símplex*) de manera inteligente utilizando el menor número de experimentos en el laboratorio.

2. Ajustar modelos estadísticos para las respuestas mecánicas de geles de pectina en función de los componentes de la mezcla y las variables de proceso.

3.2. Objetivo específico.

1. Generar un diseño experimental para los componentes de la mezcla y las variables de proceso que permita ajustar modelos estadísticos para la respuesta mecánica de geles de pectina.

2. Simular la respuesta mecánica de geles de pectina en el espacio $\mathbb{R}^4_+ \times T_4 \times T_2 \times T_2$.

3. Ajustar un Proceso Gaussiano Arbolado para predecir la respuesta del sistema en una nueva configuración de las condiciones experimentales.

Capítulo 4

Materiales y métodos

Para la formulación de geles de pectina se propone estudiar una mezcla base de componentes en proporciones y condiciones fisicoquímicas establecidas a fin de modelar la(s) respuesta(s) de interés y poder diseñar productos con características específicas. Se asume que los diversos productos saborizantes no influyen sobre la respuesta y no se consideran dentro del diseño experimental.

Para evaluar la metodología propuesta se simulará el modelo ficticio en las expresiones 4.1 a 4.6 que considera los componentes de tres mezclas distintas (Cationes divalentes + monovalentes, Pectina de alto + bajo metoxilo, sacarosa + glucosa), y las condiciones fisicoquímicas que producen geles de pectina (pH, fuerza iónica, concentración de pectina y concentración de sólidos solubles).

La concentración de sólidos solubles (*SS*) y pectina (*Pect*), la fuerza iónica (*I*) y el pH (*pH*) se tratarán como factores independientes y definen un espacio ortogonal $\mathbb{R}^4_+ = [0.25, 0.75] \times [0,1] \times [0,100] \times [2.5, 4.5]$. La proporción de los cationes $[Ca^{++}] : [Mg^{++}] : [Na^+] : [K^+]$ con los que se ajusta la fuerza iónica, la proporción de pectina de bajo y alto metoxilo $[LM] : [HM]$, y la proporción de sacarosa y glucosa $[Sac] : [Glu]$, forman mezclas que definen tres *Símplices* distintos, T_4, T_2, T_2. La combinación de estos cuatro espacios definen el espacio experimental $\mathbb{R}^4_+ \times T_4 \times T_2 \times T_2$.

Se genera un diseño experimental en el hípercubo $\bar{I}^9$. Cuatro columnas de la matriz diseño corresponden al espacio ortogonal $\mathbb{R}^4_+$, y cinco columnas se utilizan en los tres mapeos $\bar{I}^{q-1} \rightarrow T_q$ de cada una de las tres mezclas. Se hace una transformación de $\bar{I}^4 \rightarrow \mathbb{R}^4_+$ para definir los niveles de los factores independientes. Asímismo, se utiliza un mapeo de $\bar{I}^3 \rightarrow T_4$ y dos mapeos de $\bar{I}^1 \rightarrow T_2$ para definir las proporciones de los componentes de cada una de las tres mezclas. Sin embargo, el modelo se ajusta dentro del hípercubo $\bar{I}^9$ debido a que es posible y necesario trabajar dentro de un espacio ortogonal.

4.1. Componentes de la mezcla base.

El Cuadro 4.1 muestra los factores y los componentes que definen el espacio experimental. Los factores independientes se muestran en la columna izquierda, y los componentes de las distintas mezclas se muestran en la columna derecha. El agua también es un componente de la mezcla pero se asume que tiene un efecto de dilución, por lo tanto se considera como variable *Slack*.

Variable independiente	**Componentes**
Sólidos solubles ·SS·	(T_2) Sacarosa y Glucosa $[Sac] : [Glu]$
Polisacáridos ·Pect·	(T_2) Pectina de alto metoxilo y bajo metoxilo $[HM] : [LM]$
Fuerza iónica ·I·	(T_4) $NaCl$, KCl, $CaCl_2$ y $MgCl_2$ $[Na^+] : [K^+] : [Ca^{++}] : [Mg^{++}]$
pH	–

Cuadro 4.1: Factores y componentes que definen el espacio experimental. La concentración de Sólidos solubles (*SS*) y Polisacáridos (*Pect*), la fuerza iónica (*I*) y el pH (*pH*) son factores independientes. T_q indica a qué *Símplex* pertenecen los componentes de las mezclas.

4.1.1. Restricciones lineales.

De acuerdo con su función para la formación de geles, los Sólidos solubles (*SS*) y Polisacáridos (*Pect*) (como componentes de una mezcla acuosa) están sometidos a las restricciones que se muestran en el Cuadro 4.2. Sin embargo, como forman parte de una solución acuosa, donde el agua es una variable *Slack*, estas restricciones sólo sirven para definir los límites superior e inferior de cada factor.

	Inferior	Superior
Agua	–	–
Sólidos solubles	25.0	75.0
Polisacáridos	0.0	1.0

Cuadro 4.2: Límites superiores e inferiores para los Sólidos solubles (*SS*) y los Polisacáridos (*Pect*) en $\%p/p$. La proporción de agua es libre y suficiente para completar el 100 %.

4.1.2. Restricciones lineales para los componentes de cada *Símplex*.

La única restricción para los componentes de cada *Símplex* es que sumen 100 % de su concentración, o fuerza iónica, en las proporciones establecidas en el diseño experimental.

4.2. Variables de proceso.

La fuerza iónica I se define como

$$I = \frac{1}{2}\sum_{i=1}^{n} c_i z_i^2;\ i = 1,2,3,\ldots,n$$

donde n es el número total de iones en solución, c_i y z_i representan la concentración molar ($mol \cdot dm^{-3}$) y la carga, respectivamente, del i-ésimo ión. Para facilitar el estudio, sólo se consideran sales en forma de cloruros (Cl^-). De la definición de fuerza iónica, resulta claro que los iones forman una mezcla que corresponde al *Simplex* T_4 en el diseño experimental.

La concentración de los cationes es importante porque:

1. Se espera que los cationes divalentes (con carga 2+) produzcan geles con las pectinas de bajo metoxilo.

2. Los cationes monovalentes (con carga 1+) compiten con los cationes divalentes por las cargas negativas de la cadena de pectina. Esto modifica las características de los geles.

Dado que la pectina es un polisacárido polianiónico con una constante de acidez $pK_a \approx 3.4$, se espera una interacción entre la pectina, el pH y la fuerza iónica. Se asume que el pH se ajustará con una solución amortiguadora (*buffer*) de ácido cítrico/citrato de sodio en una concentración fija.

En el Cuadro 4.2 y 4.3 se muestra los niveles mínimo y máximo para los factores independientes.

Variables de proceso	Inferior	Superior
pH (adimensional)	2.4	4.4
Fuerza iónica (mM)	0.0	100.0

Cuadro 4.3: Intervalo de valores que pueden tomar los factores independientes pH y Fuerza iónica.

4.3. Modelo simulado

El árbol que se simula se presenta en la Figura 4.3. Las particiones representan las condiciones reales en las cuales se espera obtener geles con características muy distintas en cuanto a la respuesta. Dicho árbol tiene un total de 8 particiones en el espacio $\mathbb{R}^4_+ \times T_4 \times T_2 \times T_2$, pero en realidad sólo se esperan tres regímenes distintos, a saber;

1. Cuando no forma geles.

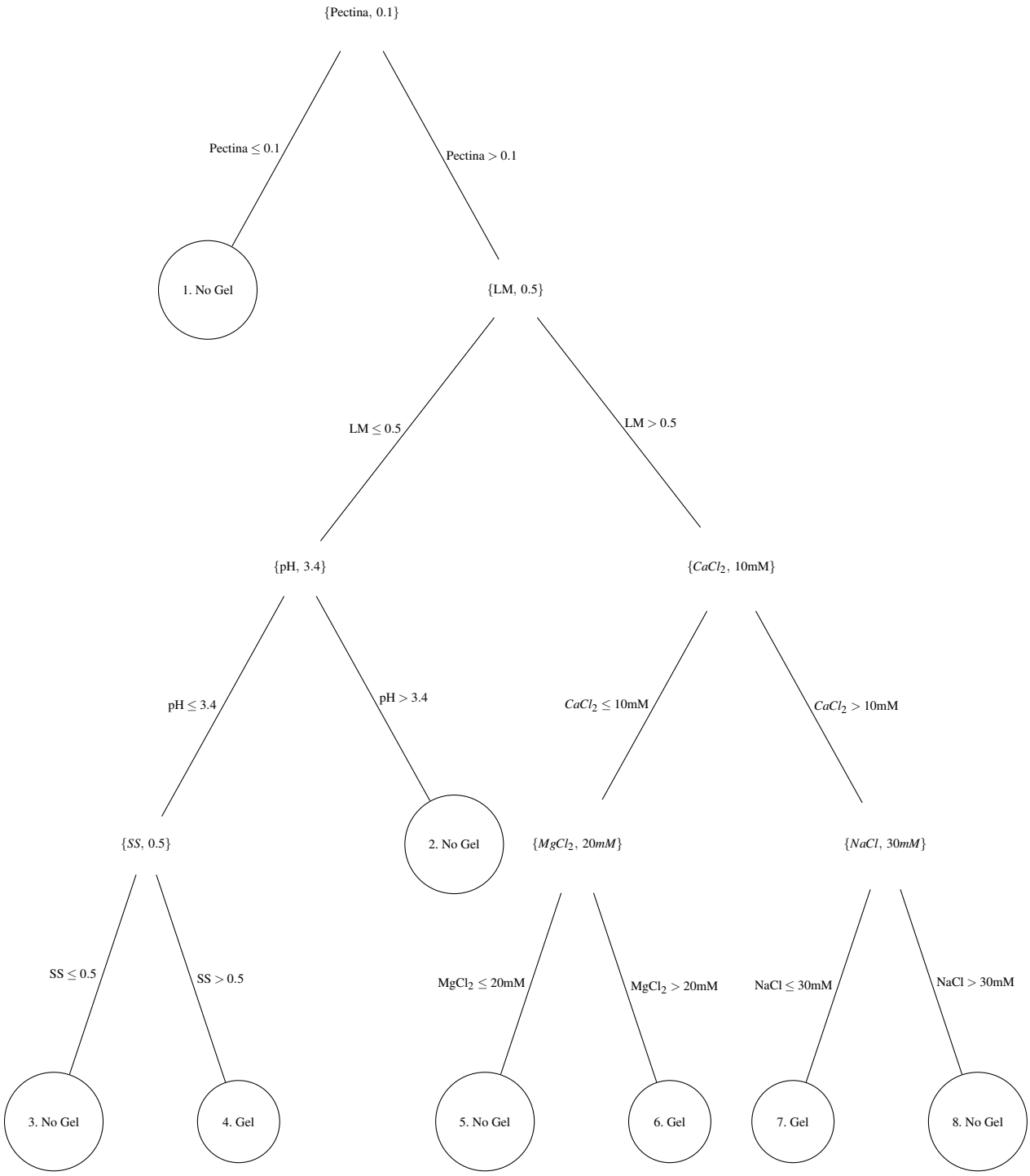

Figura 4.1: Árbol simulado en el espacio experimental. Los nodos marcados con “Gel” y “No Gel” indican cuando existen las condiciones de gelificación.

2. Cuando forma geles donde domina la pectina de alto metoxilo “HM”.

3. Cuando forma geles donde domina la pectina de bajo metoxilo “LM”.

En el diseño inicial creado mediante el algoritmo NTLBG, en número de puntos dentro de cada una de las 8 particiones con 20 puntos en total, es: $n_1 = 0$, $n_2 = 5$, $n_3 = 2$, $n_4 = 1$, $n_5 = 1$, $n_6 = 2$, $n_7 = 6$, $n_8 = 3$.

Además, cada una de las pectinas gelifican bajo condiciones fisicoquímicas distintas:

LM: Sólo requiere Ca^{++} para gelificar. El pH ácido ($pH < pK_a$) neutraliza las cargas negativas de la cadena y reduce el efecto gelificante del Ca^{++}; se espera que los cationes Na^+ y K^+ tengan un efecto similar al H^+ cuando el pH es ácido.

HM: Requiere una concentración alta de sólidos solubles y $pH < pK_a = 3.4$. El Ca^{++} tiene un ligero efecto gelificante en concentraciones muy altas $[Ca^{++}] > 50mM$. A diferencia de la pectina LM, se espera que los cationes Na^+ y K^+ tengan un efecto en favor de la gelificación, al neutralizar las cargas de la cadena de pectina similar al efecto de $[H^+]$ en pH ácido.

Con esta información, el modelo que se simula es una mezcla de ambos regímenes ponderado por la proporción LM:HM y una región lineal cuando la concentración de pectina no es suficiente para producir geles $[Pect] \leq 0.1$. Además, se incluirá interacciones entre los componentes del *Simplex* y las variables independientes, i.e., $pH * [LM]$, $I * [LM]$ y $[Ca^{++}] * [LM]$ a fin de añadir una curvatura en la superficie de respuesta. Finalmente, dado que las concentraciones de pectina y de sólidos solubles son las que más afectan la respuesta mecánica, se añade un coeficiente global “$\sqrt{Pect + SS} \in [0, \sqrt{2}]$” para considerar un efecto de dilución.

La magnitud y el signo de los coeficientes del modelo se proponen subjetivamente con base en el efecto esperado en cada régimen. Los modelos a simular son:

$$\begin{aligned} Y_{LM} =& 2pH + 5I + 2SS + 0Pect + 10[Ca^{++}] + 9[Mg^{++}] - 4[Na^+] - 2[K^+] + 1[Sac] \\ &+ 1[Glu] + 0[HM] + 0[LM] + 2(pH * [LM]) + 7([I] * [LM]) + 7([Ca^{++}] * [LM]) \end{aligned} \tag{4.1}$$

$$\begin{aligned} Y_{HM} =& -4pH + 5I + 9SS + 0Pect + 3[Ca^{++}] + 1[Mg^{++}] + 4[Na^+] + 3[K^+] + 10[Sac] \\ &+ 10[Glu] + 0[HM] + 0[LM] + 1(pH * [LM]) + 1([I] * [LM]) + 1([Ca^{++}] * [LM]) \end{aligned} \tag{4.2}$$

En cada partición, se simula uno de los tres regímenes distintos. Los subíndices del vector Y indican en qué nodos del árbol en la Figura 4.3 aplica dicho modelo.

$$Y_{\{1\}} = \sqrt{SS + Pect} * (\mu_{NoGel} + I + SS + Pect) + \varepsilon \tag{4.3}$$

$$Y_{\{2,3,5,8\}} = \sqrt{SS + Pect} * (\mu_{NoGel} + [LM] * Y_{LM} + [HM] * Y_{HM}) + \varepsilon \tag{4.4}$$

$$Y_{\{6,7\}} = \sqrt{SS + Pect} * (\mu_{Gel_{LM}} + [LM] * Y_{LM} + [HM] * Y_{HM}) + \varepsilon \tag{4.5}$$

$$Y_{\{4\}} = \sqrt{SS + Pect} * (\mu_{Gel_{HM}} + [LM] * Y_{LM} + [HM] * Y_{HM}) + \varepsilon \tag{4.6}$$

Con $\mu_{NoGel} = 10$, $\mu_{Gel_{LM}} = 100$ y $\mu_{Gel_{HM}} = 300$ y $\varepsilon \sim N(0,1)$.

4.4. Diseño experimental.

Se comparan dos enfoques:

Inteligente: comenzar con un diseño experimental pequeño $n_0 = 20$ y utilizar la estadística *ALC* para crecer el diseño en cada iteración. En cada iteración se selecciona los 3 o 4 puntos con mayor *ALC*.

Estático: comenzar con un diseño experimental igual al tamaño que toma n_i en la i-ésima iteración del enfoque *inteligente*. Por ejemplo, $n_0 = \{20, 23, 26, ...\}$.

Para el diseño inicial tamaño n_0 de ambos enfoques, primero se genera un conjunto *glp* con el vector generador $h = (3997; 1, 3888, 3564, 3034, 2311, 1417, 375, 3211, 1962)$ (matriz de dimensión 3997×9) como se explica en la Sección 2.3.2. Las columnas $1, 3, 4, 5$ del conjunto *glp* se utilizan para las variables de proceso en el espacio $\bar{I}^4$, las columnas $6, 7, 8$ para el *Simplex* T_4 de $[Ca^{++}] : [Mg^{++}] : [Na^{+}] : [K^{+}]$; la columna 2 para el *Simplex* T_2 de $[Sac] : [Glu]$; y la columna 9 para el *Simplex* T_2 de $[HM] : [LM]$. El conjunto *glp* generado de este modo se utiliza para generar los diseños experimentales iniciales en el espacio $\bar{I}^4 \times T_4 \times T_2 \times T_2 \equiv \bar{I}^{12}$ mediante el algoritmo NTLBG.

Para evaluar la conveniencia de iniciar con un diseño experimental pequeño y hacerlo crecer *inteligentemente*, se compara el error cuadrado medio de predicción del modelo obtenido en cada iteración del enfoque inteligente contra el de un diseño nuevo generado mediante el algoritmo NTLBG en cada iteración del enfoque estático. En ambos casos, se simula la respuesta $\mathbf{Y}$ mediante el modelo en las expresiones 4.1 a 4.6 y el árbol en la Figura 4.1 en cada una de las condiciones experimentales $\mathbf{X} \in \bar{I}^9$ que se alimentan al paquete.

4.4.1. Diseño experimental adaptativo.

Para el análisis del diseño *inteligente*, se utilizan las siguientes condiciones:

- diseño inicial: $n_0 = 20$, generado mediante el algoritmo NTLBG;
- *Burn-in*: 1000; tamaño de la cadena: 4000; muestreo: cada 3 eslabones para adelgazar la muestra y disminuir la autocorrelación de la misma;
- correlación Gaussiana isotrópica: $p_0 = 2$, $\theta_i = \theta$;
- $p_{SPLIT} = 0.95(1 - q_\eta)^2$;
- vector $\mathbf{Y}$ estandarizado con media 0 y varianza 1;
- densidades *a priori* impropias (uniformes);
- reiniciar la cadena 20 veces con las mismas condiciones para garantizar la convergencia de las mismas;
- diseño hipercubo latino: $n_{lhd} = 600$; este diseño proporciona una lista de puntos candidatos de los que se preselecciona un subconjunto de tamaño n_{D-opt} en el siguiente paso;
- diseño D-óptimo: $n_{D-opt} = 400$; este diseño representa la lista de puntos para las que se hará predicción de la respuesta y se calculará el estadístico ALC;
- calcular *ALC* y seleccionar los 3 o 4 puntos con *ALC* más grande para la siguiente iteración;
- 40 iteraciones en total.

En cada iteración, dado los datos $\{\mathbf{X}, \mathbf{Y}\}$, el paquete hace una exploración estocástica de la distribución *a posteriori*. Tras dicha exploración inicial, es necesario generar una lista de puntos candidatos de la siguiente forma:

1. Generar un diseño hipercubo latino de tamaño $n_{lhd} = 600$ en $\bar{I}^9$.
2. Dada la estructura del árbol y el modelo ajustado en cada partición durante la exploración inicial del espacio, generar un diseño D-óptimo de tamaño $n_{D-opt}400$ con los puntos del diseño hipercubo latino.

Se calcula *ALC* para todos los puntos del diseño D-óptimo a fin de seleccionar los 3 o 4 puntos con mayor *ALC* y añadirlos al diseño experimental de la siguiente iteración. Este método busca sacar provecho del *active learning* para crecer el experimento de manera *inteligente*.

4.4.2. Diseño experimental mediante el algoritmo NTLBG.

Para el análisis de los diseños generados mediante el algoritmo NTLBG, se utiliza las condiciones siguientes:

- diseño inicial de tamaño igual al del diseño aumentado en cada iteración del enfoque *inteligente*;
- *Burn-in*: 1000; tamaño de la cadena: 4000; muestreo: cada 3 eslabones para adelgazar la muestra y disminuir la autocorrelación de la misma;
- correlación Gaussiana isotrópica: $p_0 = 2$, $\theta_i = \theta$;
- $p_{SPLIT} = 0.95(1 - q_\eta)^2$;
- vector $\mathbf{Y}$ estandarizado con media 0 y varianza 1;
- densidades *a priori* impropias (uniformes);
- reiniciar la cadena 20 veces con las mismas condiciones para garantizar la convergencia de las mismas;
- 40 iteraciones en total.

El diseño en cada iteración se genera mediante el algoritmo NTLBG igualando el tamaño del diseño en cada iteración del enfoque *inteligente*, y se considera un ajuste fijo.

4.5. Ajuste del modelo

Para el ajuste del modelo, se utilizará la biblioteca `tgp` (Gramacy, 2007) de R. Esta biblioteca tiene todas las funciones para explorar el espacio del árbol, ajustar un proceso Gaussiano en cada partición y calcular los estadísticos *ALM* y *ALC*. Además, cuenta con las funciones `lhs()`, para generar diseños hipercubo latino, y `tgp.design()`, para generar diseños D-óptimos.

Capítulo 5

Resultados y discusión

La matriz diseño inicial con $n = 20$ y con los elementos de cada *Símplex* T_q mapeados al hipercubo $\bar{I}^{q-1}$ se muestra en el Cuadro 5.1. Cada vector (fila) de la matriz $\mathbf{X} \in \bar{I}^9$ se generó utilizando algoritmo NTLBG en el hipercubo $\bar{I}^4$ y en los *Símplex*, T_4, T_2 y T_2. Una vez generada la matriz diseño en $\bar{I}^{12} \equiv \bar{I}^4 \times T_4 \times T_2 \times T_2$, se hizo una transformación lineal para obtener la misma matriz diseño inicial $n = 20$, en escala real $\mathbb{R}^4_+ \times T_4 \times T_2 \times T_2$. Esta matriz se muestra en el Cuadro 5.2.

	pH	I	SS	Pectina	f(I).1	f(I).2	f(I).3	f(SS)	f(Pectina)
1	0.230	0.791	0.738	0.768	0.350	0.720	0.845	0.982	0.636
2	0.763	0.248	0.226	0.784	0.732	0.457	0.164	0.327	0.682
3	0.425	0.752	0.243	0.831	0.125	0.639	0.759	0.391	0.879
4	0.230	0.157	0.316	0.229	0.737	0.197	0.751	0.188	0.726
5	0.757	0.248	0.716	0.764	0.304	0.407	0.231	0.689	0.165
6	0.808	0.488	0.764	0.356	0.348	0.319	0.717	0.259	0.949
7	0.230	0.212	0.807	0.229	0.705	0.212	0.246	0.604	0.533
8	0.827	0.725	0.230	0.619	0.739	0.046	0.495	0.114	0.477
9	0.213	0.819	0.260	0.436	0.127	0.154	0.493	0.802	0.843
10	0.740	0.159	0.735	0.250	0.711	0.451	0.496	0.838	0.983
11	0.759	0.237	0.244	0.237	0.108	0.672	0.253	0.873	0.358
12	0.262	0.223	0.148	0.742	0.339	0.760	0.508	0.557	0.586
13	0.692	0.751	0.287	0.193	0.735	0.794	0.380	0.945	0.419
14	0.232	0.739	0.764	0.198	0.344	0.081	0.467	0.038	0.806
15	0.267	0.505	0.181	0.223	0.022	0.419	0.521	0.450	0.295
16	0.756	0.758	0.746	0.788	0.753	0.807	0.886	0.648	0.231
17	0.727	0.846	0.779	0.229	0.368	0.773	0.160	0.765	0.767
18	0.264	0.278	0.865	0.751	0.743	0.786	0.633	0.506	0.914
19	0.382	0.451	0.596	0.449	0.733	0.445	0.837	0.728	0.033
20	0.230	0.270	0.499	0.768	0.762	0.816	0.119	0.909	0.100

Cuadro 5.1: Matriz diseño $\mathbf{X} \in \bar{I}^9$. Las columnas $f(\cdot)$ representa el mapeo inverso del Símplex T_q al hipercubo $\bar{I}^{q-1}$.

	pH	I	SS	Pect	[Ca++]	[Mg++]	[Na+]	[K+]	[Sac]	[Glu]	[HM]	[LM]
1	2.76	87.00	63.10	0.77	29.50	10.67	9.27	50.57	1.83	98.17	36.43	63.57
2	3.93	27.30	34.90	0.78	9.87	29.20	50.97	9.96	67.30	32.70	31.77	68.23
3	3.19	82.70	35.90	0.83	50.05	10.01	9.63	30.30	60.91	39.09	12.08	87.92
4	2.76	17.20	39.90	0.23	9.68	50.24	9.97	30.10	81.22	18.78	27.41	72.59
5	3.92	27.30	61.90	0.76	32.74	24.34	33.02	9.90	31.11	68.89	83.46	16.54
6	4.03	53.60	64.50	0.36	29.68	30.59	11.25	28.48	74.09	25.91	5.12	94.88
7	2.76	23.30	66.90	0.23	11.00	48.04	30.89	10.07	39.58	60.42	46.69	53.31
8	4.07	79.80	35.20	0.62	9.58	71.00	9.80	9.62	88.62	11.38	52.28	47.72
9	2.72	90.00	36.80	0.44	49.68	30.59	10.00	9.73	19.83	80.17	15.69	84.31
10	3.88	17.50	62.90	0.25	10.74	29.33	30.22	29.70	16.24	83.76	1.70	98.30
11	3.92	26.10	35.90	0.24	52.44	8.59	29.10	9.87	12.67	87.33	64.22	35.78
12	2.83	24.60	30.60	0.74	30.28	8.93	29.88	30.90	44.31	55.69	41.39	58.61
13	3.77	82.60	38.30	0.19	9.77	9.81	49.83	30.60	5.47	94.53	58.13	41.87
14	2.76	81.30	64.50	0.20	29.95	50.10	10.64	9.32	96.18	3.82	19.41	80.59
15	2.84	55.50	32.40	0.22	71.97	9.88	8.69	9.46	54.95	45.05	70.50	29.50
16	3.91	83.30	63.50	0.79	9.01	9.26	9.32	72.41	35.20	64.80	76.93	23.07
17	3.85	93.00	65.30	0.23	28.31	8.66	52.94	10.08	23.48	76.52	23.30	76.70
18	2.83	30.60	70.10	0.75	9.44	10.29	29.48	50.79	49.42	50.58	8.57	91.43
19	3.09	49.60	55.30	0.45	9.84	30.02	9.81	50.33	27.22	72.78	96.67	3.33
20	2.76	29.70	49.90	0.77	8.65	8.84	72.65	9.85	9.08	90.92	90.04	9.96

Cuadro 5.2: Matriz diseño $\mathbf{X} \in \mathbb{R}^4_+ \times T_4 \times T_2 \times T_2$. Las columnas se encuentran en escala real. Las columnas $[\cdot]$ (5 a 12) son $\%p/p$ dentro de cada *Símplex*.

5.1. Error cuadrado medio.

Utilizando el modelo ajustado en cada iteración, se interpola la respuesta de un diseño hipercubo latino de tamaño 100 y se calculó la raíz del error cuadrado medio mediante:

$$RMSE_j = \sqrt{\frac{1}{100}\sum_{i=1}^{100}(\hat{Y}_{ij} - Y_i)^2}. \tag{5.1}$$

Donde $\hat{Y}_{ij}$ es la respuesta interpolada utilizando el modelo ajustado en la j-ésima iteración y Y_i es la respuesta real simulada.

En la Figura 5.1 se muestra el cociente $RMSE_j^{ALC}/RMSE_j^{NTLBG}$ para cada iteración en función del tamaño del diseño. En $9/40$ $(22.5\,\%)$ ocasiones no se pudo generar un diseño experimental utilizando el algoritmo NTLBG debido a que las funciones programadas, las semillas y el algoritmo no lo permitieron ($j = \{13,14,17,18,34,35,36,38,39\}$). En todo momento, fue posible utilizar una semilla distinta para poder generar el diseño, sin embargo, se decidió utilizar la misma semilla para generar todos los diseños de este estudio. En $6/40$ $(15\,\%)$ ocasiones, $RMSE_j^{ALC} \leq RMSE_j^{NTLBG}$ ($j = \{3,4,6,21,22,24\}$). En el resto de las iteraciones $25/40$ $(62.5\,\%)$, $RMSE_j^{ALC} > RMSE_j^{NTLBG}$ ($j = \{1,2,5,7,8,9,10,11,12,15,16,19,20,23,25,26,27,28,29,30,31,32,33,37,40\}$).

A pesar de que un diseño de tamaño 20 es demasiado pequeño para modelar la respuesta en un

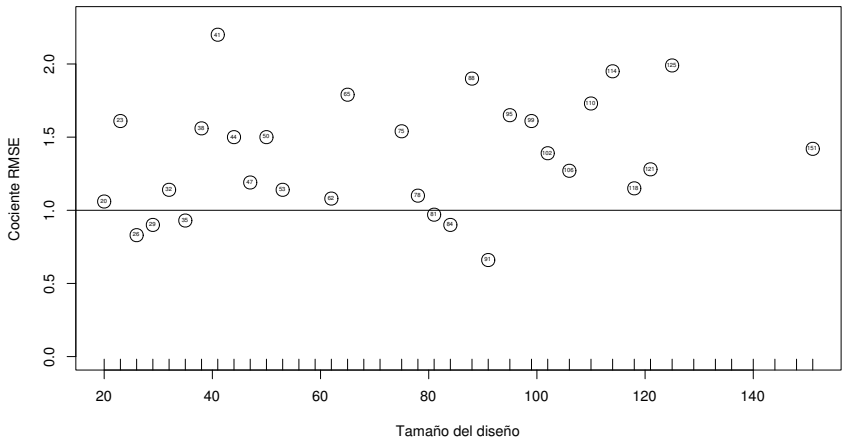

Figura 5.1: Cociente de la raíz del error cuadrado medio (*RMSE*) de predicción: ALC/NTLBG.

espacio con 9 dimensiones, estos resultados sugieren que es mejor comenzar con un diseño muy pequeño y dejar que los resultados vayan guiando la exploración del espacio experimental como en el enfoque inteligente.

5.2. Cobertura del espacio experimental.

	pH	I	SS	Pectina	f(I).1	f(I).2	f(I).3	f(SS)	f(Pectina)
Min.	0.213	0.157	0.148	0.193	0.022	0.046	0.119	0.038	0.033
1st Qu.	0.232	0.245	0.244	0.229	0.330	0.292	0.251	0.375	0.342
Median	0.404	0.469	0.547	0.443	0.537	0.454	0.495	0.626	0.611
Mean	0.490	0.483	0.507	0.492	0.489	0.498	0.498	0.581	0.569
3rd Qu.	0.756	0.751	0.750	0.765	0.735	0.763	0.725	0.811	0.815
Max.	0.827	0.846	0.865	0.831	0.762	0.816	0.886	0.982	0.983

Cuadro 5.3: Resumen del diseño inicial por columna, $n = 20$.

	pH	I	SS	Pectina	f(I).1	f(I).2	f(I).3	f(SS)	f(Pectina)
Min.	0.002	0.002	0.001	0.004	0.006	0.004	0.002	0.004	0.009
1st Qu.	0.137	0.222	0.464	0.132	0.180	0.232	0.179	0.153	0.154
Median	0.382	0.531	0.675	0.391	0.514	0.623	0.508	0.452	0.413
Mean	0.401	0.525	0.614	0.441	0.498	0.557	0.500	0.463	0.421
3rd Qu.	0.593	0.820	0.849	0.760	0.763	0.873	0.788	0.731	0.627
Max.	0.999	0.998	0.999	0.994	0.995	0.990	0.994	0.995	0.996

Cuadro 5.4: Resumen del diseño final generado utilizando ALC, n= 151

En las Tablas 5.3 a 5.5 se muestran algunas estadísticas del diseño experimental incial y los diseños finales de ambos enfoques. Se puede observar que el diseño inicial no alcanza los límites

	pH	I	SS	Pectina	f(I).1	f(I).2	f(I).3	f(SS)	f(Pectina)
Min.	0.062	0.008	0.015	0.030	0.005	0.010	0.045	0.007	0.002
1st Qu.	0.197	0.248	0.223	0.265	0.261	0.266	0.245	0.224	0.287
Median	0.497	0.505	0.491	0.536	0.523	0.501	0.490	0.486	0.523
Mean	0.498	0.497	0.487	0.510	0.510	0.497	0.506	0.485	0.509
3rd Qu.	0.782	0.779	0.777	0.770	0.812	0.734	0.772	0.725	0.731
Max.	0.991	0.955	0.958	0.966	0.915	0.967	0.966	0.996	0.999

Cuadro 5.5: Resumen del diseño final generado utilizando el algoritmo NTLBG, n= 151

del hipercubo $\bar{I}^9$, lo cual es un efecto del algoritmo NTLBG. Sin embargo, conforme el diseño crece *inteligentemente* utilizando ALC, la cobertura del diseño experimental tiende a extenderse hacia dichos límites. Esto era de esperarse porque el error de predicción es mayor en los límites de la región experimental y de las particiones.

Al comparar los diseños finales, se puede observar que el diseño que se generó utilizando ALC tiene una mayor cobertura del espacio experimental, i.e., salvo en la columna de $f(Pectina)$, los intervalos entre el mínimo y el máximo son más amplios en el enfoque *inteligente*.

5.3. Árbol.

Como no fue posible generar diseños mediante el algoritmo NTLBG para algunas iteraciones, sólo se compara los resultados de ambos enfoques en las últimas 6 iteraciones en las que sí fue posible generar diseños, i.e., $j = \{30, 31, 32, 33, 37, 40\}$. En cada iteración, se muestra el árbol que corresponde a la muestra con máxima probabilidad *a posteriori* (*MAP*) encontrado después de reiniciar la cadena 20 veces. Los nodos "$\{u, s_u\}$" corresponden a los nodos en el árbol simulado. Se debe notar que los valores de corte $s_u \in [0,1]$, esto es debido a que la función `btgpllm()` requiere que la matriz diseño $\mathbf{X} \in \bar{I}^9$ esté escalada para el análisis.

Iteración 30.

Para los diseños con tamaño $n = 114$, mediante el enfoque *inteligente*, el árbol con *MAP* logró identificar 6 particiones, mientras que el enfoque *estático* sólo logró identificar 4 particiones. Estos árboles se muestran en las Figuras A.1 y A.2.

En esta iteración, el cociente de los *RMSE* fue de 1.95.

Iteración 30, *ALC* (RMSE=34.106). En el árbol con *MAP* en la Figura A.1 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;

- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$;
- $X_4 \equiv Pectina \leq\approx 0.5$; no corresponde a ningún nodo, pero podría corresponder a un cambio en el régimen de gelificación cuando domina una pectina sobre la otra;
- $X_4 \equiv Pectina \leq\approx 0.3$; no corresponde a ningún nodo.

En esta iteración se logró identificar más nodos que los existentes en el árbol simulado. Es posible que el nodo $\{X_4, 0.3\}$ sea producto de la exploración estocástica de la distribución *a posteriori* y no tenga mejor explicación que esa.

Iteración 30, NTLBG (RMSE=66.468). En el árbol con *MAP* en la Figura A.2 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$.

En esta iteración, sólo se logró identificar algunos de los nodos existentes en el árbol simulado. Sin embargo, todos los nodos encontrados se encuentran en el árbol simulado.

Iteración 31.

Para los diseños con tamaño $n = 118$, mediante el enfoque *inteligente*, el árbol con *MAP* logró identificar 5 particiones, mientras que el enfoque *estático* sólo logró identificar 4 particiones. Estos árboles se muestran en las Figuras A.3 y A.4.

En esta iteración, el cociente de los *RMSE* fue de 1.15.

Iteración 31, *ALC* (RMSE=53.655). En el árbol con *MAP* en la Figura A.3 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_4 \equiv Pectina \leq\approx 0.1$; correspondiente al nodo $\{Pectina, 0.1\}$;
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$;

En esta iteración el enfoque *inteligente* logró identificar menos nodos que en la iteración previa. Además, todos los nodos encontrados se pueden localizar en el árbol simulado.

Iteración 31, NTLBG (RMSE=61.967). En el árbol con *MAP* en la Figura A.4 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$.
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;

Por su parte, el enfoque *estático* identificó los mismos 4 nodos que en la iteración anterior. Sin embargo, a diferencia de la iteración anterior, debe notar que los nodos tienen la misma estructura que en el árbol simulado.

Iteración 32.

Para los diseños con tamaño $n = 121$, mediante el enfoque *inteligente*, el árbol con *MAP* logró identificar 5 particiones, mientras que el enfoque *estático* sólo logró identificar 4 particiones. Estos árboles se muestran en las Figuras A.5 y A.6.

En esta iteración, el cociente de los *RMSE* fue de 1.28.

Iteración 32, *ALC* (RMSE=54.128). En el árbol con *MAP* en la Figura A.5 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_4 \equiv Pectina \leq\approx 0.1$; correspondiente al nodo $\{Pectina, 0.1\}$;
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$;

En esta iteración, se logró identificar los mismos nodos que en la iteración previa.

Iteración 32, NTLBG (RMSE=69.132). En el árbol con *MAP* en la Figura A.6 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$.
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;

En esta iteración se logró identificar los mismos nodos y la misma estructura que en la iteración previa. Nuevamente, se debe notar que los nodos tienen la misma estructura que en el árbol simulado.

Iteración 33.

Para los diseños con tamaño $n = 125$, mediante el enfoque *inteligente*, el árbol con *MAP* logró identificar 6 particiones, mientras que el enfoque *estático* sólo logró identificar 4 particiones. Estos árboles se muestran en las Figuras A.7 y A.8.

En esta iteración, el cociente de los *RMSE* fue de 1.99.

Iteración 33, *ALC* (RMSE=38.661). En el árbol con *MAP* en la Figura A.7 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$;
- $X_4 \equiv Pectina \leq\approx 0.2$; correspondiente al nodo $\{Pectina, 0.1\}$;
- $X_4 \equiv Pectina \leq\approx 0.5$; no corresponde a ningún nodo, pero sí a un cambio en el régimen de gelificación cuando domina una pectina sobre la otra.

En esta iteración, se logró identificar más nodos que en la iteración previa. Además, la estructura del árbol *MAP* no corresponde a la del árbol simulado.

Iteración 33, NTLBG (RMSE=76.863). En el árbol con *MAP* en la Figura A.8 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$.
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;

En esta iteración se logró identificar los mismos nodos y la misma estructura que en las dos iteraciones previas. Nuevamente, se debe notar que los nodos tienen la misma estructura que en el árbol simulado.

Iteración 37.

Para los diseños con tamaño $n = 140$, ambos enfoques lograron identificar 5 particiones con estructuras distintas. Estos árboles se muestran en las Figuras A.9 y A.10.

En esta iteración, el cociente de los *RMSE* fue de 3.53.

Iteración 37, *ALC* (RMSE=40.475). En el árbol con *MAP* en la Figura A.9 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$;
- $X_4 \equiv Pectina \leq\approx 0.25$; correspondiente al nodo $\{Pectina, 0.1\}$.

En esta iteración, se logró identificar un nodo menos que en la iteración 33. Además, la estructura del árbol con *MAP* no corresponde a la del árbol simulado.

Iteración 37, NTLBG (RMSE=143.073). El árbol con *MAP* en la Figura A.10 identifica los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$.
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;
- $X_2 \equiv I \leq\approx 0.5$; este nodo no se encuentra dentro de la estructura del árbol simulado;

En esta iteración se logró identificar un nodo más que en la iteración 33. El nodo $\{I, 0.5\}$ no se encuentra dentro del árbol simulado y tampoco ha sido identificado mediante el enfoque *inteligente*. Este nodo no tiene mucho sentido de acuerdo con el modelo y la estructura del árbol que se simuló. Es posible que este nodo sea una consecuencia de la exploración estocástica de la distribución *a posteriori* del modelo.

Iteración 40.

Para los diseños con tamaño $n = 151$, mediante el enfoque *inteligente*, el árbol con *MAP* logró identificar 6 particiones, mientras que el enfoque *estático* sólo logró identificar 4 particiones. Estos árboles se muestran en las Figuras A.11 y A.12.

En esta iteración, el cociente de los *RMSE* fue de 1.42.

Iteración 40, *ALC* (RMSE=43.914). En el árbol con *MAP* en la Figura A.11 se identifican los nodos siguientes:

- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;
- dos particiones distintas cercanas a $X_9 \equiv f(Pectina) \leq\approx 0.5$; alguna de ellas corresponde al nodo $\{LM, 0.5\}$;
- $X_4 \equiv Pectina \leq\approx 0.2$; correspondiente al nodo $\{Pectina, 0.1\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$;

En esta iteración, se logró identificar un nodo más que en la iteración 37. Un segundo nodo $\{f(Pectina), 0.5\}$ no tiene mucho sentido de acuerdo con el modelo y la estructura del árbol que se simuló. Este nodo adicional puede ser una consecuencia de la exploración estocástica de la distribución *a posteriori* del modelo. Claramente, la estructura del árbol con *MAP* no corresponde a la del árbol simulado.

Iteración 40, NTLBG (RMSE=62.178). En el árbol con *MAP* en la Figura A.12 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$ correspondiente al nodo $\{LM, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$.
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;

Nuevamente, en esta iteración se logró identificar los mismos nodos y la misma estructura del árbol con *MAP* que en las iteraciones 32, 32, 33 y 37.

5.4. Otros árboles con *MAP*; iteraciones 34, 35, 36, 38, 39.

En esta sección se analiza los árboles encontrados en las iteraciones 34, 35, 36, 38, 39 para compararlos entre sí, ya que no es posible compararlos contra los diseños generados mediante el algoritmo NTLBG.

Iteración 34 (RMSE=81.605).

Para el diseño con tamaño $n = 129$, mediante el enfoque *inteligente*, el árbol con *MAP* logró identificar 6 particiones. Este árbol se muestran en la Figura A.13.

En el árbol con *MAP* en la Figura A.13 se identifican los nodos siguientes:

- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;
- dos particiones distintas cercanas a $X_9 \equiv f(Pectina) \leq\approx 0.5$; alguna de ellas corresponde al nodo $\{LM, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$;
- $X_4 \equiv Pectina \leq\approx 0.1$; correspondiente al nodo $\{Pectina, 0.1\}$;

En esta iteración, se logró identificar dos nodos cercanos a $f(Pectina), 0.5$. Es posible que una corresponda al nodo $\{LM, 0.5\}$, sin embargo, el otro no tiene sentido, pues no existe en el árbol simulado. También es posible que este nodo sea producto de la exploración estocástica de la distribución *a posteriori* y no tenga mejor explicación que esa. Además, esta partición adicional no ha sido encontrada en ninguna otra iteración. Claramente, la estructura del árbol *MAP* no corresponde a la del árbol simulado.

Iteración 35 (RMSE=40.379).

Para el diseño con tamaño $n = 133$, mediante el enfoque *inteligente*, el árbol con *MAP* logró identificar 5 particiones. Este árbol se muestran en la Figura A.14.

En el árbol con *MAP* en la Figura A.14 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$; correspondiente al nodo $\{LM, 0.5\}$;
- $X_4 \equiv Pectina \leq\approx 0.1$; correspondiente al nodo $\{Pectina, 0.1\}$;
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$;

Nuevamente, la estructura del árbol *MAP* no corresponde a la del árbol simulado.

Iteración 36 (RMSE=54.407).

Para el diseño con tamaño $n = 136$, mediante el enfoque *inteligente*, el árbol con *MAP* logró identificar 6 particiones. Este árbol se muestran en la Figura A.15.

En el árbol con *MAP* en la Figura A.15 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$; correspondiente al nodo $\{LM, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$; correspondiente al nodo $\{pH, 3.4\}$;
- $X_4 \equiv Pectina \leq\approx 0.1$; correspondiente al nodo $\{Pectina, 0.1\}$;

- $X_3 \equiv SS \leq \approx 0.5$; correspondiente al nodo $\{SS, 0.5\}$;
- $X_6 \equiv f(I).2 \leq \approx 0.7$.

En esta iteración se logró identificar por primera y única ocasión un nodo en $X_6 \equiv f(I).2$. Como se puede observar en la Expresión 2.31, $f(I).2$ sólo es una función de la proporción de los cationes $[Ca^{++}]$ y $[Mg^{++}]$ del *Símplex* T_4, y por lo tanto es posible que corresponda a la partición 5 en la Figura 4.3, i.e., $\{Ca, Cl_2\ 10mM\}$ y $\{MgCl_2,\ 20mM\}$. Alternativamente, es posible que este nodo sea producto de la exploración estocástica de la distribución *a posteriori* y no tenga mejor explicación que esa.

La particiones paralelas a los ejes dentro del *Símplex* se distorsionan al mapearlas hacia el espacio ortogonal de una manera dependiente del componente en el que se encuentre la regla de partición. Por ejemplo, en una mezcla de tres componentes $\mathbf{X} = \{X_1, X_2, X_3\} \in T_3$, donde existe una partición en $X_1 \leq 0.5$, o $X_2 \leq 0.5$, o $X_3 \leq 0.5$, cada una de estas particiones mapeadas al espacio ortogonal $\bar{I}^2$ se observan como se muestra en la Figura 5.2.

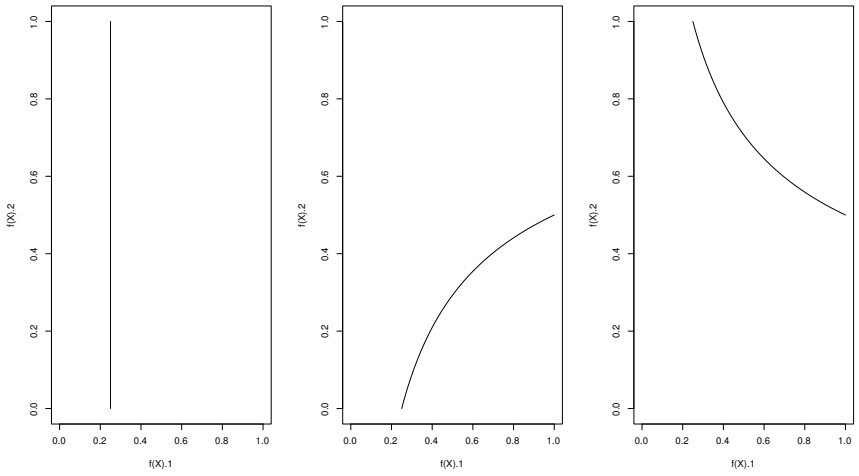

Figura 5.2: Particiones del simplex T_3 mapeadas al espacio ortogonal $\bar{I}^2$. Izquierda: Partición en $X_1 \leq 0.5$; Centro: Partición en $X_2 \leq 0.5$; Derecha: Partición en $X_3 \leq 0.5$. Los ejes corresponden a las transformaciones $f(X)$.

Por lo tanto, en caso de que la exploración del espacio experimental identifique particiones en $f(I)$, dichas particiones no necesariamente existen dentro del *Sïmplex* T_4, ni corresponden a las particiones simuladas.

Iteración 38 (RMSE=50.414).

Para el diseño con tamaño $n = 144$, mediante el enfoque *inteligente*, el árbol con *MAP* logró identificar 5 particiones. Este árbol se muestran en la Figura A.16.

En el árbol con *MAP* en la Figura A.16 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$; correspondiente al nodo $\{LM, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$;
- $X_4 \equiv Pectina \leq\approx 0.1$; correspondiente al nodo $\{Pectina, 0.1\}$;
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;

Nuevamente, la estructura del árbol con *MAP* no corresponde a la del árbol simulado.

Iteración 39 (RMSE=45.382).

Para el diseño con tamaño $n = 148$, mediante el enfoque *inteligente*, el árbol con *MAP* logró identificar 5 particiones. Este árbol se muestran en la Figura A.17.

En el árbol con *MAP* en la Figura A.17 se identifican los nodos siguientes:

- $X_9 \equiv f(Pectina) \leq\approx 0.5$; correspondiente al nodo $\{LM, 0.5\}$;
- $X_4 \equiv Pectina \leq\approx 0.1$; correspondiente al nodo $\{Pectina, 0.1\}$;
- $X_3 \equiv SS \leq\approx 0.5$ correspondiente al nodo $\{SS, 0.5\}$;
- $X_1 \equiv pH \leq\approx 0.5$ correspondiente al nodo $\{pH, 3.4\}$;

Nuevamente, la estructura del árbol *MAP* no corresponde a la del árbol simulado.

5.5. Exploración exhaustiva de la distribución *a posteriori*.

Para hacer una búsqueda más exhaustiva del árbol con *MAP*, se volvió a correr el análisis con el diseño experimental de tamaño $n = 151$ de la iteración 40 del enfoque *inteligente* y del enfoque *estático*. De los 151 puntos en el diseño experimental en el enfoque inteligente en la 40^a iteración, el número de puntos dentro de cada una de las particiones simuladas es: $n_1 = 32$, $n_2 = 21$, $n_3 = 14$, $n_4 = 40$, $n_5 = 6$, $n_6 = 6$, $n_7 = 23$ y $n_8 = 9$. En esta ocasión, se permitió que la cadena reiniciara un total de 50 veces. Los resultados se muestran en las Figuras A.18 y A.19.

El *RMSE* para el enfoque *inteligente* y *estático* fueron 54.282 y 81.945, respectivamente. En esta ocasión, el enfoque *inteligente* resultó ser 1.51 veces mejor que el enfoque *estático*.

Estructura del árbol

De los 151 puntos en el diseño experimental en el enfoque estático en la 40^a iteración, el número de puntos dentro de cada una de las particiones simuladas es: $n_1 = 14$, $n_2 = 32$, $n_3 = 19$, $n_4 = 14$, $n_5 = 8$, $n_6 = 10$, $n_7 = 43$ y $n_8 = 11$. En cuanto a la estructura del árbol, el enfoque *estático* nuevamente identificó los mismos nodos y la misma estructura que había identificado previamente en las iteraciones 31, 32, 33, 37 y 40. Estos nodos junto con esta estructura se logran identificar por primera vez a partir de la iteración 31, $n = 118$. Este diseño tiene una mejor distribución de los puntos en la particiones simuladas que el diseño bajo el enfoque *inteligente*. A pesar de esto, no fue posible identificar las particiones correctamente utilizando este diseño. Aparentemente, este enfoque tiene limitaciones para poder identificar correctamente todas las particiones.

Por su parte, el enfoque *inteligente* logró identificar correctamente hasta 5 particiones con la estructura correcta del árbol, i.e., nodo hijo izquierdo de $\{LM, 0.5\}$, $([LM] \leq 0.5)$. En ningún caso fue posible identificar correctamente las particiones del nodo hijo derecho de $\{LM, 0.5\}$, $([LM] > 0.5)$, las cuales se encuentran dentro del *Símplex* T_4 de los cationes.

En cuanto a la eficacia de ambos enfoques para identificar las particiones, el enfoque *inteligente* comenzó a identificar 4 particiones (con nodos correctos pero una estructura errónea) a partir de la iteración 16, $n = 65$. En contraste, el enfoque *estático*, comenzó a identificar 4 particiones a partir de la iteración 20, $n = 78$ (con nodos incorrectos $\{X_7, \approx 0.5\}$ en la iteración 20 y $\{X_6, \approx 0.5\}$ en la iteración 21, además de una estructura errónea).

Se debe notar que el diseño experimental en la 40^a iteración, bajo el enfoque *estático*, tiene una mejor distribución de los puntos dentro del espacio experimental que el diseño bajo el enfoque *inteligente*. Esta característica es deseable cuando se espera que las irregularidades sean homogéneas dentro de todo el espacio experimental. Sin embargo, en este problema, esta característica resulta fútil porque se explora excesiva e inútilmente las regiones poco interesantes del espacio.

Al analizar la cantidad de puntos dentro de cada partición bajo el enfoque *inteligente*, se puede concluir que la simulación finalizó antes de poder detectar todas las particiones y, si se hubiera corrido durante un número mayor de iteraciones, es posible que se hubieran identificado las particiones faltantes. Es de notar que el diseño *inteligente* tiene un número muy reducido de puntos en las particiones 3, 5, 6 y 9, lo cual es posible que sea insuficiente para identificar dichas particiones. Sin embargo, no hay forma de garantizar que dichas particiones serán identificadas con un número razonable de corridas experimentales.

Finalmente, aunque un diseño experimental de tamaño 151 es un diseño de gran tamaño que en ocasiones podría no ser práctico o posible. Lo importante en este ejercicio de simulación no sólo es identificar las particiones, sino ajustar un modelo de predicción adecuado.

Capítulo 6

Conclusiones

En esta investigación se evaluó la aplicación del método propuesto por Gramacy and Lee (2008a, 2009) en experimentos con mezclas y variables de proceso. Para este fin, fue necesario mapear los componentes de las mezclas de q componentes al hipercubo unitario de $q-1$ dimensiones mediante la inversión de las expresiones del método de transformación inversa propuesto por Fang and Wang (1993). Se evaluó también la conveniencia de iniciar la experimentación con un número muy reducido de experimentos y posteriormente hacer crecer el diseño contra un enfoque estático que utiliza diseño experimental inicial de diferentes tamaños.

Si se utiliza un enfoque *estático*, es preferible extender los límites del diseño experimental para que dichos límites sean mayores a los límites de la región de interés. Por otro lado, si se utiliza un enfoque *inteligente*, los límites del diseño experimental pueden ser iguales a los límites de la región de interés, pues conforme crezca el diseño experimental en cada iteración, se añadirán más puntos (condiciones experimentales) en las regiones de mayor incertidumbre, como por ejemplo, los límites de la región experimental y los límites de las particiones.

El enfoque *inteligente* que utiliza *ALC* logró identificar más rápidamente las particiones durante el análisis, i.e., puede comenzar a identificar particiones con un diseño de menor tamaño.

Por su parte, el enfoque *estático* logró identificar en varias ocasiones, para los nodos identificados, la misma estructura presente en el árbol simulado. Aunque esta consistencia es deseable, este enfoque parece estar limitado para identificar las particiones, aún con un número relativamente grande de experimentos.

En la metodología propuesta con un enfoque *inteligente*, $n_0 = 20$ y *ALC*, parecería inadecuado pretender modelar un espacio con alta dimensionalidad utilizando un número tan pequeño de puntos en el espacio. Sin embargo, al seleccionar las corridas experimentales subsecuentes a partir de los resultados ya obtenidos, se hace más eficiente el diseño experimental, pues sólo se exploran las regiones más interesantes del espacio.

No se encontró una sola referencia bibliográfica que utilice las expresiones invertidas del mé-

todo de transformación inversa propuesto por Fang and Wang (1993) para mapear del *Símplex* al espacio ortogonal, $T_q \rightarrow \bar{I}^{q-1}$. En este caso, estas expresiones fueron utilizadas para romper la dependencia lineal entre los componentes de cada *Símplex* y así poder utilizar el proceso Gaussiano arbolado y modelar una respuesta en un espacio distinto al espacio euclidiano.

La transformación $T_q \rightarrow \bar{I}^{q-1}$ permite romper la dependencia lineal entre los componentes de la mezcla pero, al igual que la transformación propuesta por Fang and Wang (1993), tiene la desventaja de ser dependiente de orden de los componentes de la mezcla en la matriz diseño. Por este motivo, su uso se debe limitar a situaciones en las que esta característica no representa un problema.

Una dificultad que resulta de utilizar la transformación $T_q \rightarrow \bar{I}^{q-1}$ en la metodología del presente estudio, es que no es posible identificar fácilmente en el espacio ortogonal aquellas particiones dentro de *Símplices* con dimensión mayor o igual a tres. Sin embargo, la metodología sí permite modelar satisfactoriamente una respuesta dentro de espacios distintos al espacio euclidiano, como el espacio $\bar{I}^4 \times T_4 \times T_2 \times T_2$ en el presente estudio.

Finalmente, este estudio proporciona una metodología adecuada y eficiente para experimentos que involucran mezclas y factores independientes. Como siguiente paso, es necesario llevar a cabo un análisis de sensibilidad para probar la significancia de los términos en el modelo. Adicionalmente, para los estudios que requieran maximizar, minimizar, u optimizar la respuesta dentro de un intervalo predeterminado, es necesario desarrollar un *profiler* con el modelo obtenido.

Apéndice A

Árboles con máxima probabilidad *a posteriori* en cada iteración

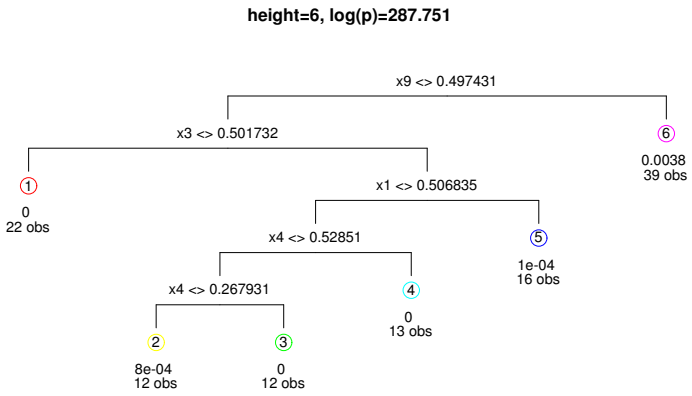

Figura A.1: Árbol con *MAP* en la iteración 30, encontrado mediante un enfoque *inteligente* (ALC).

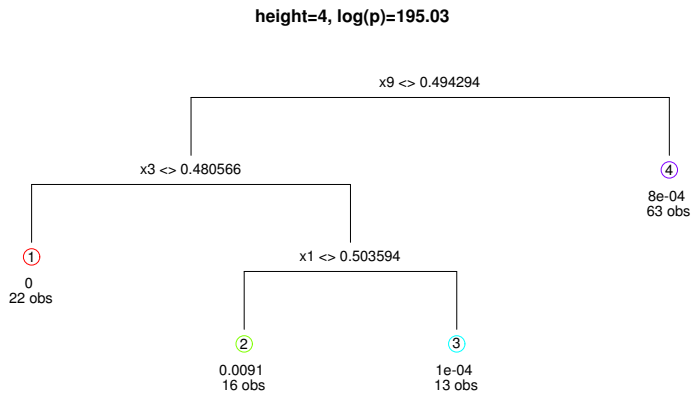

Figura A.2: Árbol con *MAP* en la iteración 30, encontrado mediante un enfoque *estático* (NTLBG).

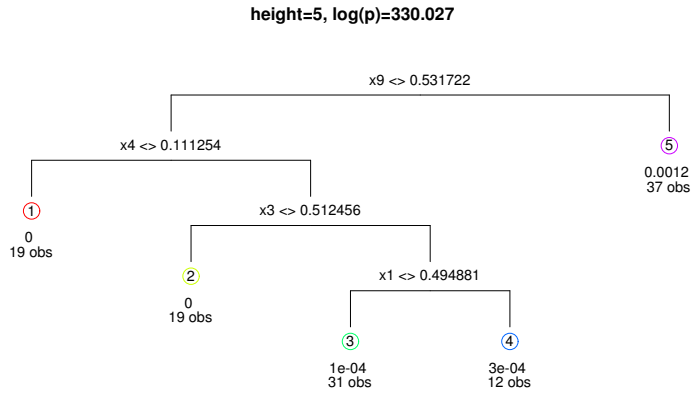

Figura A.3: Árbol con *MAP* en la iteración 31, encontrado mediante un enfoque *inteligente* (ALC).

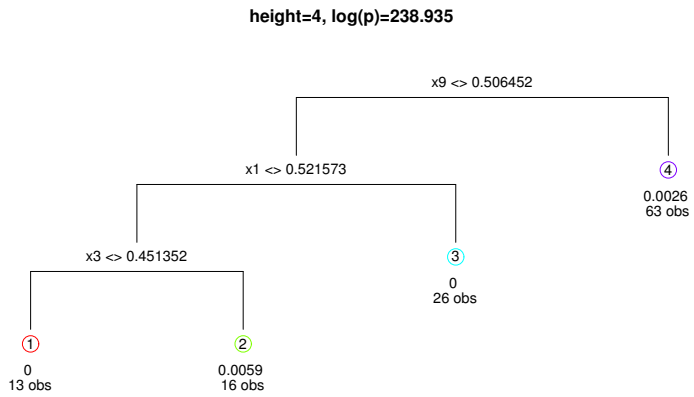

Figura A.4: Árbol con *MAP* en la iteración 31, encontrado mediante un enfoque *estático* (NTLBG).

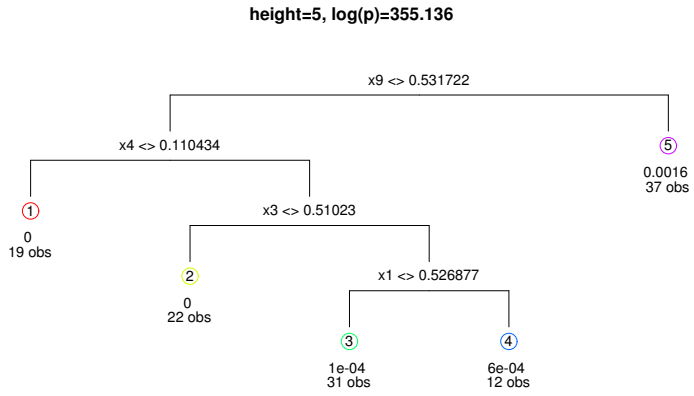

Figura A.5: Árbol con *MAP* en la iteración 32, encontrado mediante un enfoque *inteligente*.

Figura A.6: Árbol con *MAP* en la iteración 32, encontrado mediante un enfoque *estático* (NTLBG).

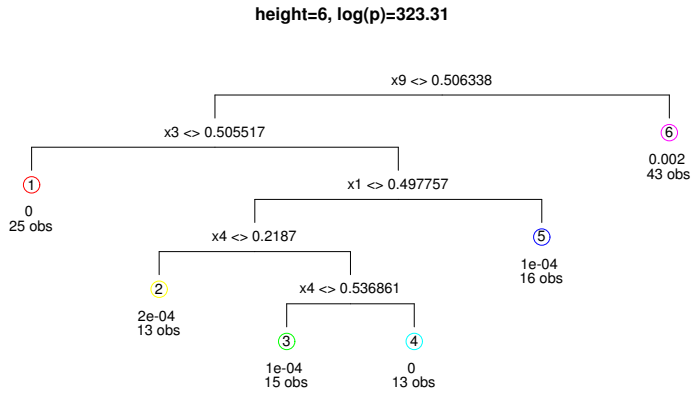

Figura A.7: Árbol con *MAP* en la iteración 33, encontrado mediante un enfoque *inteligente*.

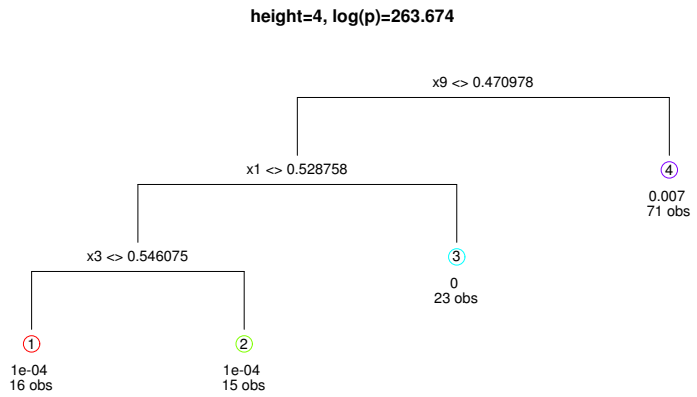

Figura A.8: Árbol con *MAP* en la iteración 33, encontrado mediante un enfoque *estático* (NTLBG).

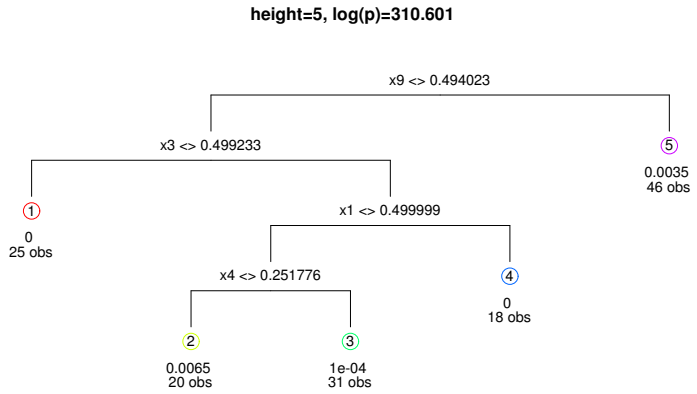

Figura A.9: Árbol con *MAP* en la iteración 37, encontrado mediante un enfoque *inteligente*.

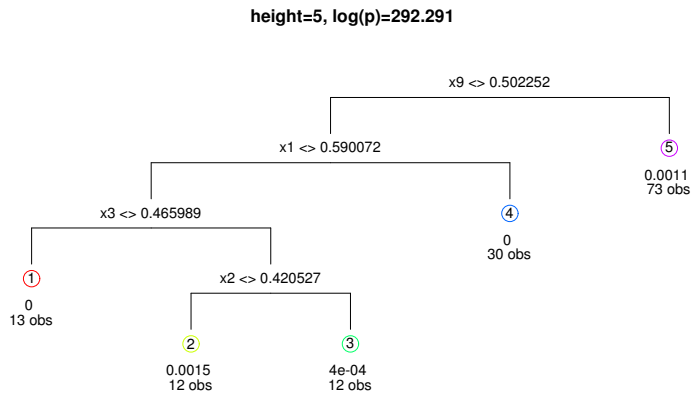

Figura A.10: Árbol con *MAP* en la iteración 37, encontrado mediante un enfoque *estático* (NTLBG).

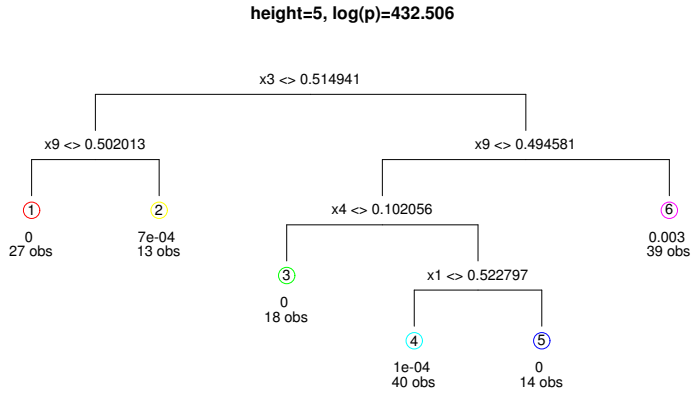

Figura A.11: Árbol con *MAP* en la iteración 40, encontrado mediante un enfoque *inteligente*.

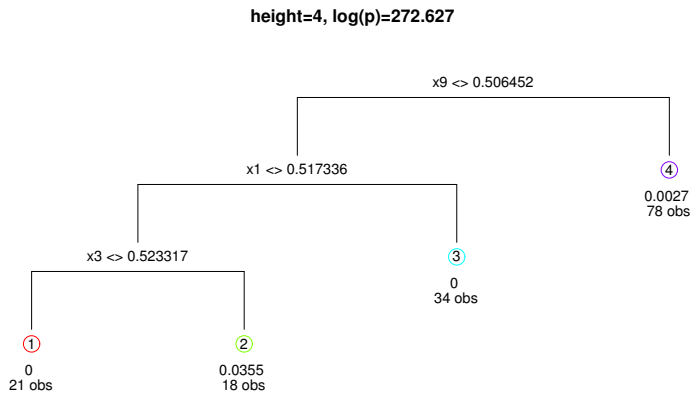

Figura A.12: Árbol con *MAP* en la iteración 40, encontrado mediante un enfoque *estático* (NTLBG).

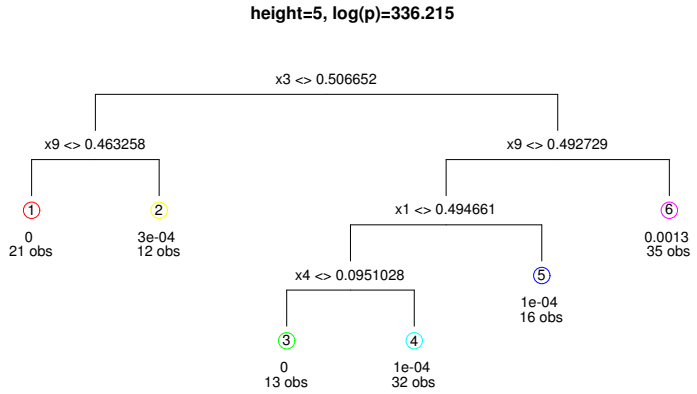

Figura A.13: Árbol con *MAP* en la iteración 34, encontrado mediante un enfoque *inteligente*.

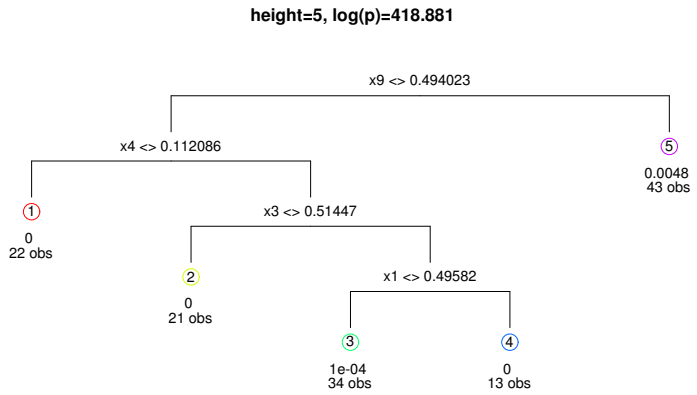

Figura A.14: Árbol con *MAP* en la iteración 35, encontrado mediante un enfoque *inteligente*.

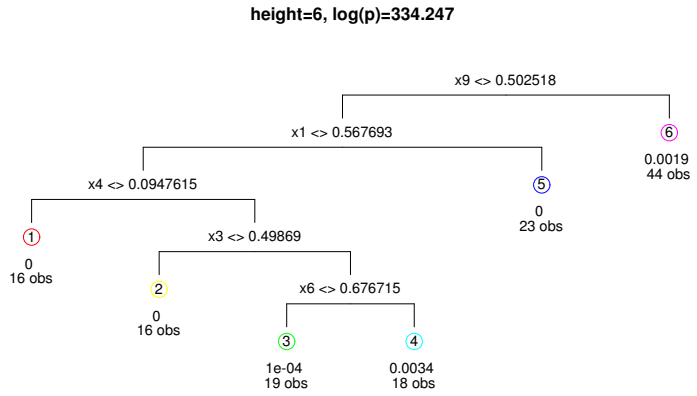

Figura A.15: Árbol con *MAP* en la iteración 36, encontrado mediante un enfoque *inteligente*.

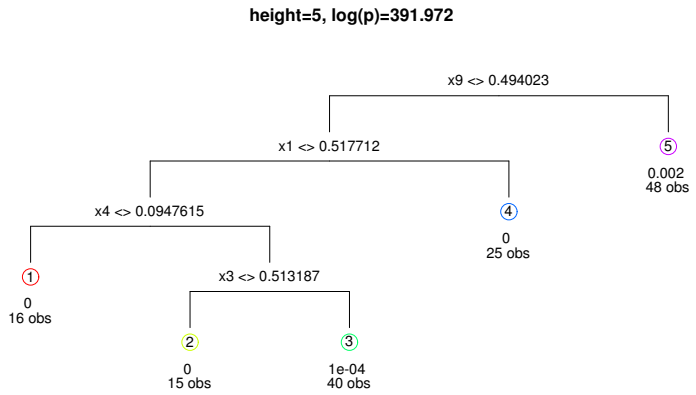

Figura A.16: Árbol con *MAP* en la iteración 38, encontrado mediante un enfoque *inteligente*.

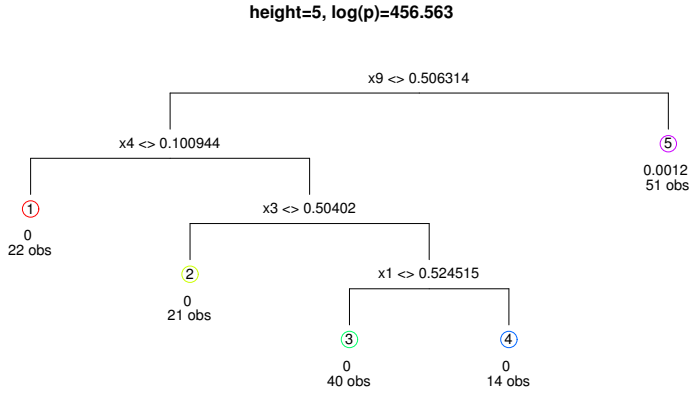

Figura A.17: Árbol con *MAP* en la iteración 39, encontrado mediante un enfoque *inteligente*.

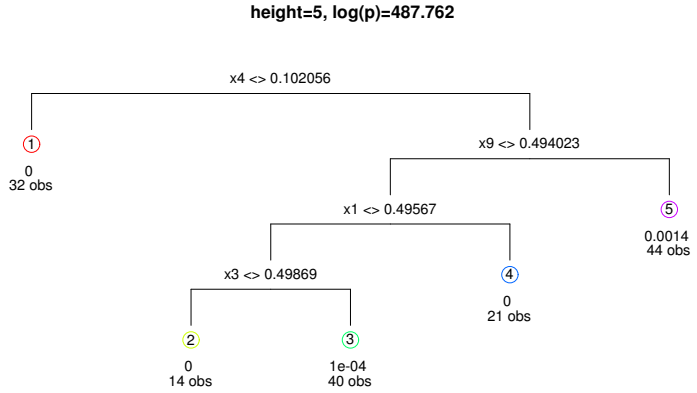

Figura A.18: Búsqueda exhaustiva del árbol con *MAP* después de reiniciar la cadena 50 veces. El diseño experimental corresponde a la iteración 40 del enfoque *inteligente* (ALC).

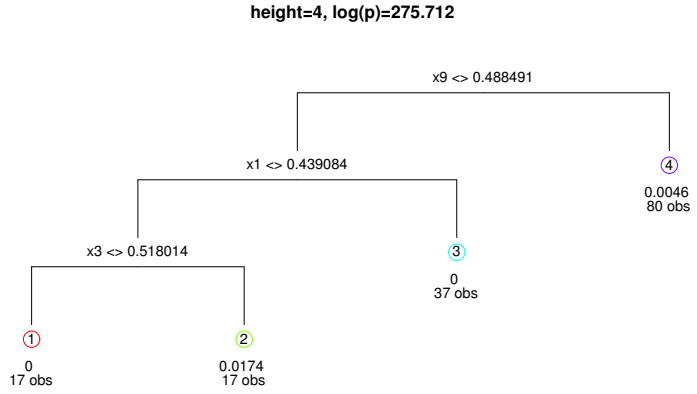

Figura A.19: Búsqueda exhaustiva del árbol con *MAP* después de reiniciar la cadena 50 veces. El diseño experimental corresponde a la iteración 40 del enfoque *estático* (NTLBG).

Bibliografía

Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984. ISBN 0412048418. URL http://books.google.com/books?id=JwQx-WOmSyQC{&}pgis=1.

Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayesian CART Model Search. *Journal of the American Statistical Association*, 93(443):935 —- 948, 1998. doi: 10.2307/2669832. URL http://www.jstor.org/stable/2669832.

Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayesian treed models. *Machine Learning*, 48(1-3):299 —- 320, 2002. doi: 10.1023/A:1013916107446. URL http://link.springer.com/article/10.1023{%}2FA{%}3A1013916107446.

Kai-lai Chung. An EstimateConcerning The Kolmogoroff Limit Distribution. *Transactions of the American Mathematical Society*, 67(1):36–50, 1949. doi: 10.2307/1990415. URL http://www.jstor.org/discover/10.2307/1990415?uid=3738664{&}uid=2{&}uid=4{&}sid=21105332832923http://www.ams.org/journals/tran/1949-067-01/S0002-9947-1949-0034552-5/S0002-9947-1949-0034552-5.pdf.

P. J. Claringbold. Use of the Simplex Design in the Study of Joint Action of Related Hormones. *Biometrics*, 11(2):174–185, 1955. doi: 10.2307/3001794. URL http://www.jstor.org.access.biblio.colpos.mx/stable/3001794.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, mar 1996. URL http://arxiv.org/abs/cs/9603104.

John A. Cornell. *Experiments with mixtures: designs, models, and the analysis of mixture data*. Wiley, 2002. ISBN 0471393673. URL https://books.google.com/books?id=oYQ{_}AQAAIAAJ{&}pgis=1.

Kai-Tai Fang and Yuan Wang. *Number-Theoretic Methods in Statistics*. CRC Press, 1993. ISBN 0412465205. URL http://books.google.com/books?id=Umb-zE{_}Gcr4C{&}pgis=1.

Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and Modeling for Computer Experiments*. CRC Press, 2005. ISBN 1420034898. URL http://books.google.com/books?id=VGqbyIUVZw8C{&}pgis=1.

Robert B. Gramacy. tgp: An R Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models. *The Journal of Statistical Software*, 19(9):1 —- 46, 2007. URL http://www.jstatsoft.org/v19/i09.

Robert B. Gramacy and Herbert K. H. Lee. Bayesian Treed Gaussian Process Models With an Application to Computer Modeling. *Journal of the American Statistical Association*, 103 (483):1119 —- 1130, sep 2008a. ISSN 0162-1459. doi: 10.1198/016214508000000689. URL http://www.tandfonline.com/doi/abs/10.1198/016214508000000689http://www.jstor.org/discover/10.2307/27640148?uid=3738664{&}uid=2134{&}uid=2{&}uid=70{&}uid=4{&}sid=21104671635043.

Robert B. Gramacy and Herbert K. H. Lee. Gaussian Processes and Limiting Linear Models. page 31, apr 2008b. URL http://arxiv.org/abs/0804.4685.

Robert B. Gramacy and Herbert K. H. Lee. Adaptive Design and Analysis of Supercomputer Experiments. *Technometrics*, 51(2):130–145, may 2009. ISSN 0040-1706. doi: 10.1198/TECH.2009.0015. URL http://www.tandfonline.com/doi/abs/10.1198/TECH.2009.0015.

Peter J. Green. Reversible jump Markov chain Monte Carlo computation Bayesian model determination. *Biometrika*, 82(4):711–732, 1995. doi: 10.1093/biomet/82.4.711. URL http://biomet.oxfordjournals.org/content/82/4/711.

Klaus Hinkelmann and Oscar Kempthorne. *Design and Analysis of Experiments, Introduction to Experimental Design*. John Wiley & Sons, 2007. ISBN 0470191740. URL http://books.google.com/books?id=T3wWj2kVYZgC{&}pgis=1.

Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: a tutorial (with comments by M. Clyde, David Draper and E. I. George, and a rejoinder by the authors. *Statistical Science*, 14(4):382–417, nov 1999. ISSN 2168-8745. URL http://projecteuclid.org/euclid.ss/1009212519.

Jack Karl Kiefer. On large deviations of the empiric d.f. of vector chance variables and a law of the iterated logarithm. *Pacific Journal of Mathematics*, 11(2):649–660, 1961. URL http://msp.org/pjm/1961/11-2/p17.xhtml.

Hyoung-Moon Kim, Bani K. Mallick, and C. C. Holmes. Analyzing Nonstationary Spatial Data Using Piecewise Gaussian Processes. 100(470):653 —- 668, jan 2005. doi: 10.1198/

016214504000002014. URL http://www.tandfonline.com.access.biblio.colpos.mx/doi/abs/10.1198/016214504000002014{#}.VCzO-nWx15Q.

J. R. Koehler and Art B. Owen. Computer Experiments 1 Introduction 2 Goals in computer experiments, 1996. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.5706.

Bledar A. Konomi, Georgios Karagiannis, Avik Sarkar, Xin Sun, and Guang Lin. Bayesian Treed Multivariate Gaussian Process With Adaptive Design: Application to a Carbon Capture Unit. *Technometrics*, 56(2):145 —- 158, may 2014a. ISSN 0040-1706. doi: 10.1080/00401706.2013.879078. URL http://www.tandfonline.com/doi/abs/10.1080/00401706.2013.879078.

Bledar A. Konomi, Huiyan Sang, and Bani K. Mallick. Adaptive Bayesian Nonstationary Modeling for Large Spatial Datasets Using Covariance Approximations. *Journal of Computational and Graphical Statistics*, 23(3):802 —- 829, jun 2014b. ISSN 1061-8600. doi: 10.1080/10618600.2013.812872. URL http://www.tandfonline.com/doi/abs/10.1080/10618600.2013.812872.

Yoseph Linde, Andrés Buzo, and Rober M. Gray. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications*, 28(1):84–95, 1980. doi: 10.1109/TCOM.1980.1094577. URL http://ayesha.lti.cs.cmu.edu/mlsp/courses/fall2010/class14/LBG.pdf.

David J. C. MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 4(4):590–604, jul 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.4.590. URL http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6795562.

M. D. McKay, R. J. Beckman, and W. J. Conover. Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21(2):239 —- 245, 1979. doi: 10.1080/00401706.1979.10489755. URL http://www.tandfonline.com/doi/abs/10.1080/00401706.1979.10489755{#}.VA{_}jqWCx15Q.

Nicholas Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341, sep 1949. ISSN 0162-1459. doi: 10.1080/01621459.1949.10483310. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1949.10483310.

Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2008. ISBN 0470128666. URL http://books.google.com/books?id=kMMJAm5bD34C{&}pgis=1.

Raymond H. Myers, Douglas C. Montgomery, and Christine M. Anderson-Cook. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John

Wiley & Sons, 2011. ISBN 1118210476. URL http://books.google.com/books?id=F6MJeRe2POUC{&}pgis=1.

Harald Niederreiter. Low-discrepancy and low-dispersion sequences. *Journal of Number Theory*, 30(1):51–70, sep 1988. ISSN 0022314X. doi: 10.1016/0022-314X(88)90025-X. URL http://www.sciencedirect.com/science/article/pii/0022314X8890025X.

Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, 1992. ISBN 0898712955. URL http://books.google.com/books?id=jrPZAGe0iHUC{&}pgis=1.

Harald Niederreiter. Constructions of (t,m,s)-nets and (t,s)-sequences. *Finite Fields and Their Applications*, 11(3):578 —- 600, aug 2005. ISSN 10715797. doi: 10.1016/j.ffa.2005.01.001. URL http://www.sciencedirect.com/science/article/pii/S1071579705000043.

Jian-Hui Ning, Kai-Tai Fang, and Yong-Dao Zhou. Uniform design for experiments with mixtures. *Communications in Statistics - Theory and Methods*, 40(10):1734 —- 1742, mar 2011a. ISSN 0361-0926. doi: 10.1080/03610921003637470. URL http://www.tandfonline.com/doi/abs/10.1080/.VA{_}S4WCx15Q{#}.VA{_}S-2Cx15Q.

Jian-Hui Ning, Yong-Dao Zhou, and Kai-Tai Fang. Discrepancy for uniform design of experiments with mixtures. *Journal of Statistical Planning and Inference*, 141(4):1487–1496, apr 2011b. ISSN 03783758. doi: 10.1016/j.jspi.2010.10.015. URL http://linkinghub.elsevier.com/retrieve/pii/S0378375810004842.

Art B. Owen. Randomly Permuted (t,m,s)-Nets and (t,s)-Sequences. In Harald Niederreiter and Peter Jau-Shyong Shiue, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 299 —- 317. Springer New York, vol. 106 edition, 1995. doi: 10.1007/978-1-4612-2552-2{_}19. URL http://www.springer.com/mathematics/probability/book/978-0-387-94577-4.

Thomas J. Santner, Brian J. Williams, and William Notz. *The Design and Analysis of Computer Experiments*. Springer Science & Business Media, 2003. ISBN 0387954201. URL http://books.google.com/books?id=O1itua2QzFkC{&}pgis=1.

Khalid Sayood. *Introduction to data compression, The Morgan Kaufmann Series in Multimedia Information and Systems*. Newnes, 2012.

Michael L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media, 1999. ISBN 0387986294. URL http://books.google.com/books?id=5n{_}XuL2Wx1EC{&}pgis=1.

Boxin Tang. Orthogonal Array-Based Latin Hypercubes. *Journal of the American Statistical Association*, 88(424):1392 —- 1397, 1993. URL http://www.jstor.org.access.biblio.colpos.mx/stable/2291282.

Luke Tierney. Markov chains for exploring posterior distributions. 22(4):1701–1728, 1994. doi: 10.1214/aos/1176325750. URL http://projecteuclid.org/euclid.aos/1176325750.