
**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

# ESTABILIDAD EN SISTEMAS DISTRIBUIDOS DE CONSENSO DE LARGA VIDA

**T E S I S**

**QUE PARA OBTENER EL GRADO DE:**

**MAESTRO EN CIENCIAS (COMPUTACIÓN)**

**P R E S E N T A :**

**JORGE FIGUEROA CANALES**

**DIRECTOR DE TESIS:**

**DR. SERGIO RAJSBAUM**

**México, D.F.** **2007.**


# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA
DE LA COMPUTACIÓN

TESIS DE MAESTRÍA

---

## ESTABILIDAD EN SISTEMAS DISTRIBUIDOS DE CONSENSO DE LARGA VIDA

---

Presentada por:
**Jorge Figueroa Canales**
CUENTA NO. 506005022

ASESOR:
**Dr. Sergio Rajsbaum**

**Ciudad Universitaria. Septiembre del 2007.**

*A mis padres: Alonso y Luz María,*
*a quienes quiero dar muchas satisfacciones.*

*A mi hermano Héctor,*
*porque siempre quisiste ver este trabajo concluido.*

# Agradecimientos

Al posgrado de la UNAM por darme la oportunidad de realizar mis estudios de maestría. A todos mis profesores y compañeros de materias por sus enseñanzas y ayuda, en particular al Dr. Sergio Rajsbaum y al M. en C. Evzy Oscar García por sus valiosos comentarios y consejos en la realización de este trabajo. Y a Conacyt porque su apoyo es fundamental.

# Índice general

# Índice de figuras

# Índice de cuadros

# Resumen

Los sistemas distribuidos de consenso de larga vida son aquellos sistemas que llaman al algoritmo de consenso de manera continua. Un aspecto fundamental en este tipo de sistemas es el número de cambios en el valor de decisión de salida; llamado inestabilidad. Los investigadores han desarrollado modelos que permiten explicar la inestabilidad de dichos sistemas, han demostrado tendencias en su comportamiento y han encontrado condiciones para minimizarla. Además, se han planteado nuevas preguntas acerca de la implicación que tendría modificar algunos aspectos de los modelos originales.

Precisamente una modificación a uno de los primeros modelos es lo que permite hacer más robusta y a la vez menos limitada la ejecución de los sistemas de consenso de larga vida, obteniendo también una adecuada representación matemática.

Otro aspecto importante que trata esta tesis es simular el comportamiento de los modelos teóricos. El objetivo es obtener información adicional a la que nos proporcionan las ecuaciones de los artículos previos. Por ejemplo: los investigadores prueban que la inestabilidad en estos sistemas converge hacia un valor esperado. Sin embargo, las ecuaciones no nos dicen qué tan rápido convergen a dicho valor; por lo tanto, también introducimos el estudio de lo que llamamos alcance del estado estable.

Así pues, la aportación principal de este trabajo es presentar un estudio de la velocidad de convergencia en los modelos hacia el valor esperado. Para ello, empleamos técnicas estadísticas utilizadas por los investigadores de simulaciones y modelado computacional.

Concluimos que la aplicación de los conceptos de alcance del estado estable, periodo de calentamiento y punto de fricción, serán útiles para la comparación de qué tan rápido converge un mismo modelo, cuando el valor en su parámetro $n$ de entrada cambia.

De igual modo, a lo largo de su desarrollo, nos hemos encontrado con una serie de problemas a resolver; dando como resultado una serie de ejercicios aplicables para un curso de temas selectos de sistemas distribuidos.

# Capítulo 1

# Introducción

## 1.1. El problema del consenso en los sistemas distribuidos

A lo largo del desarrollo de las ciencias de la computación, los investigadores se han encontrado con una serie de problemas a tratar; en muchas ocasiones han sido resueltos, en otras no; y por ello se han estudiado escenarios o modelos teóricos donde se busca entender el comportamiento de dichos problemas bajo ciertas condiciones, o bien encontrar límites o cotas para las soluciones posibles.

Dentro de los problemas más encontrados y tratados en la teoría del cómputo distribuido está el problema del consenso. Dicho problema es de los más estudiados debido a su interés teórico y a sus importantes aplicaciones. En su formulación básica e informal podemos decir que cada proceso comienza proponiendo su valor de entrada y después de algunas comunicaciones unos con otros, tienen que acordar en uno de esos valores. Imaginemos el ejemplo de una base de datos distribuida, donde todas tienen que convenir si una transacción se compromete a realizar (*commit*) o se da marcha atrás (*rollback*) en cada una de ellas. Es evidente en este ejemplo, que cada una de las bases de datos participantes en la transacción deben de realizar sólo una de esas dos operaciones y que debe ser la misma en todas.

En palabras simples diremos que un sistema distribuido es un conjunto de procesos[1] que pueden comunicarse entre si y se organizan para resolver un problema en común. Organizar todo ese sistema presenta repetidamente problemas de acuerdo entre los procesos.

Los sistemas distribuidos computacionales se componen de procesadores que varian en su arquitectura, velocidad y desempeño unos con otros. Además de que no comparten un reloj central y sus canales de comunicación no están exentos de fallas y límites en la capacidad de transmisión de datos. Asimismo, los sistemas distribuidos han proporcionado toda una gama de posibilidades en la resolución de problemas modernos, tanto en la ciencia como en la ingeniería. Han dado acceso a diversos recursos antes no disponibles para cualquier usuario, como lo son el almacenamiento distribuido de gran cantidad de información, el acceso remoto a una estructura de control, o bien, el uso de varios procesadores en paralelo para la resolución de problemas “pesados”, computacionalmente hablando; dando con ello una ventaja tanto en costo como en tiempo.

Post hoc, podemos pensar en ejemplos de sistemas distribuidos muy utilizados en la actualidad, como lo son Internet y los Clusters. El primero engloba una serie de redes de computadoras entre las cuales hay una gran cantidad de intercambio de información; en el segundo caso utilizamos diversos procesadores a fin de que funcionen como una sola supercomputadora, con gran poder de procesamiento y memoria.

[1] Por proceso entendemos a aquella unidad que realiza algún cálculo o trabajo, como aritmético/lógico o de comunicación. Comúnmente puede ser un procesador computacional.

Así como estos, podemos encontrar numerosos ejemplos y aplicaciones, y es precisamente la heterogeneidad en los elementos que conforman un sistema distribuido la que hace que sea difícil la situación de sincronizar el trabajo.

El problema del consenso ha sido estudiado bajo una gran variedad de suposiciones en sus maneras de comunicación y sus tolerancias a fallos; en algunos casos mostrando cotas inferiores (como en [2] y [7]), en algunos mostrando resultados de imposibilidad (como en [11]); y en otros diseñando algoritmos eficientes (como en [10]).

Las investigaciones relacionadas a este problema han sido descritas como de "una sola pasada" i.e. donde los procesos comienzan con su valor propuesto y tienen que resolver consenso una sola vez. Pero los sistemas distribuidos reales tienen que resolver consenso repetidamente en entradas recibidas una después de la otra. De esa manera, los investigadores han estudiado versiones del consenso continuo o también llamado consenso de larga vida; y han propuesto soluciones las cuales son más eficientes que simplemente repetir ejecuciones independientes del consenso de "una sola pasada".

## 1.2. Los requerimientos a cumplir en un algoritmo de consenso

En el problema estandar del consenso de "una sola pasada", cada uno de los $n$ procesadores propone su valor binario de entrada y después de algunas comunicaciones entre ellos, tienen que acordar un valor binario como salida. Estas entradas son representadas como vectores binarios $n-$dimensionales; id est, elementos del conjunto $\mathbb{Z}_2^n$.

Si bien el problema puede resolverse de manera trivial, por ejemplo: que todos los procesos decidan cero; aunque para fines prácticos esa no sería una buena estrategía.

Los sistemas de consenso deben satisfacer esencialmente tres requerimientos a fin de conseguir el consensus omnium:

1. Todos los procesos correctos deciden el mismo valor. (Acuerdo).
2. Tarde o temprano cada proceso no fallido debe decidir un valor. (Terminación).
3. El valor decidido tiene que ser el valor propuesto por algún proceso del sistema. (Validez).

Nosotros consideraremos también un parámetro de tolerancia a fallos $t$ tal que $0 \leq t \leq \lfloor n/2 \rfloor$ con un requerimiento adicional:

- Si a lo más $t$ entradas son iguales a un valor binario $b$, entonces la decisión de salida es igual a $1 - b$.

La idea principal que no debemos perder de vista en el estudio de la estabilidad en un sistema distribuido de consenso de larga vida, es que el valor de decisión puede cambiar a lo largo de varias y repetidas ejecuciones del algoritmo de consenso. Por esto mismo, es preferible el uso de algoritmos de consenso de larga vida en los cuales el número de veces que el valor de decisión es cambiado es lo más pequeño posible. Nuevamente imaginemos un ejemplo, en donde tenemos un conjunto de sensores replicados los cuales deben decidir si un dispositivo (un avión remotamente dirigido) debe realizar algún cambio en su ejecución (subir o bajar de altitud). Es en este sentido que nos interesa estudiar qué tan estable puede ser el sistema de consenso. Para entender los aspectos y los límites de la estabilidad tratamos de que el modelo sea lo más general posible. En pocas palabras, nos interesa que los cambios de decisión a lo largo de ejecuciones consecutivas del consenso, sea mínimo.

Otro aspecto fundamental de nuestra definición es la autoestabilización. Un sistema que tolera fallas transitorias es autoestabilizante. Una falla transitoria puede cambiar el estado actual de un proceso a otro diferente dentro de su espacio de estados posibles. La propiedad de autoestabilización de un sistema asegura la recuperación de un estado arbitrario a un estado que continúa con la ocurrencia de fallos inesperados. Estrictamente hablando, si la salida de un sistema no cambia cuando un limitado número de entradas cambia, entonces ese sistema es también tolerante a un limitado número de fallas transitorias que ocasionen cambios en los valores de las entradas.

Si bien, en el presente trabajo nos enfocaremos en representaciones que asumen entradas binarias, es menester señalar que también se ha estudiado, los modelos de situaciones en que los valores a decidir por el sistema distribuido, forman parte de un rango multivaluado (vea el artículo [5]). Además se dan unas demostraciones detalladas del lema 1 (página 11) y del lema 2 (página 12), las cuales son alternativas a las presentadas en [2].

Cabe mencionar que el desarrollo de esta tesis sirvió como base para un artículo presentado en el Encuentro de Estudiantes de Ciencias de la Computación 2007 en el IPN, donde obtuvo per se el premio de segundo lugar al mejor artículo. También sirvió de base para una presentación y un artículo fast abstract en el Third Latin-American Symposium on Dependable Computing 2007; llevado a cabo en la Universidad de San Nicolás de Hidalgo en Morelia, conjunto al Encuentro Internacional de Computación 2007.

## 1.3. Objetivos planteados y descripción de resultados obtenidos

Basados en la formalización, estudiaremos la estabilidad de sistemas sin memoria donde ejecuciones consecutivas de consenso son independientes; y compararla con la estabilidad de sistemas que pueden mantener memoria de ejecuciones de consenso previas. Posteriormente, generalizamos las entradas al sistema, de manera que no estén restringidas a la definición 2 (página 10) como en [5] y [2]; sino que realizamos una modificación tal y como se presenta en [4]. Con ello, se formalizan unos nuevos sistemas con memoria y sin memoria, para los cuales realizamos unas simulaciones en computadora, a fin de estudiar su comportamiento.

Otra clasificación que resulta ser relevante, es si el sistema es simétrico o no. En un sistema simétrico las decisiones son tomadas solamente en base a la distribución de los distintos valores de entrada, pero no en cuál proceso específico se produjo un particular valor de entrada. Sin embargo, en este trabajo sólo consideraremos sistemas simétricos.

Hasta este momento, los investigadores han mostrado la convergencia de la inestabilidad de estos modelos. Sin embargo, las simulaciones computacionales pueden aportarnos información adicional que no ha sido estudiada hasta ahora, como lo puede ser la rapidez con la que la inestabilidad de los sistemas converge al valor predicho en las ecuaciones.

Estas simulaciones en computadora, son presentadas en los capítulos 4 y 5, las cuales fueron realizadas en lenguaje MATLAB, bajo el ambiente *Windows XP* y *Mac Os X* con arquitecturas *Intel* Centrino Duo e *Intel* Xeon de doble núcleo respectivamente. Vea los apéndices B.1 y B.2.

El problema del consenso de larga vida es una versión dinámica del problema del consenso de “una sola pasada” donde las entradas cambian en el tiempo y los resultados de estas simulaciones sirvieron para comparar la inestabilidad mostrada en los modelos sin memoria vs. los modelos con memoria. Notese que un algoritmo de consenso de “una sola pasada” que tolere $t$ fallas para un modelo particular de comunicación, puede ser usado para resolver la versión del consenso de larga vida, simplemente invocando una nueva versión del algoritmo en cada fase. Pero este tipo de solución produce un sistema de consenso de larga vida con estabilidad no garantizada en el siguiente sentido: si los vectores de entrada de dos fases consecutivas no cambian, los valores de decisión pueden cambiar, debido a retardos de ejecución.

Nos interesa minimizar las variaciones de las salidas de fase a fase tanto como sea posible. Por supuesto, si el vector de entrada cambia de $0^n$ en una fase, a un vector $1^n$ en la siguiente fase, entonces la decisión de salida tiene que cambiar de $0$ a $1$. Pero si el vector de entrada cambia en unos pocos bits, es posible que el cambio de decisión sea evitado.

Comenzaremos pues, con una serie de definiciones que usaremos a lo largo de esta tesis. Después daremos un panorama general sobre el estado del arte ad usum referente a los trabajos realizados por los investigadores en el área; concluyendo el presente capítulo con una reseña de los resultados previos importantes que debemos tener presentes al adentrarnos en el tema.

El resto de la tesis está organizada de la siguiente forma: en el capítulo 2 veremos a detalle la estabilidad mostrada por algunos de los primeros sistemas de consenso de larga vida (vea [2]). El capítulo 3 muestra los cambios realizados en la interpretación de las entradas a fin de hacer más poderoso el modelado de los sistemas (vea [4]). El capítulo 4 enseña los experimentos realizados por las simulaciones en computadora sobre los modelos previos. Y el capítulo 5 presenta un estudio sobre la velocidad de convergencia de los modelos, basado en las simulaciones del capítulo anterior y en el artículo [4].

De las simulaciones hechas también observamos el comportamiento de los sistemas a lo largo de las ejecuciones; notamos el valor más grande en la inestabilidad cuando el parámetro de tolerancia a fallos $t$ es igual a la mitad del valor de $n$. Asimismo, notamos la convergencia predicha por los investigadores y una gran variación en los valores tempranos de salida de la ejecución.

A fin de que constituya una aportación original en el estudio de los sistemas distribuidos de consenso de larga vida; mostramos los resultados en términos de una técnica estadística utilizada por los investigadores de simulaciones computacionales. No olvidemos que una simulación es el resultado de un modelo con variables estocásticas; por lo tanto, para decir qué tan rápido converge nuestra simulación al valor esperado, es necesario decir en qué momento alcanza su estado de estabilidad (al que llamaremos punto de fricción). Y lo importante aquí es comparar ese punto de fricción cuando a nuestro programa de simulación le damos distintos parámetros $n$ de entrada.

Anticipando algunos resultados al lector, diremos que los experimentos muestran que conforme el número $n$ de procesos participantes en nuestro sistema distribuido de consenso continuo aumenta, el punto de frición se posterga. Y este comportamiento se observó tanto en el modelo con memoria como sin memoria.

## 1.4. Descripción del problema: consenso de larga vida y estabilidad

El consenso continuo o de larga vida, es una versión dinámica del consenso de “una sola pasada”; esto quiere decir que el algoritmo de consenso se tiene que realizar de forma repetida conforme los valores aportados por los procesos participantes cambian en el tiempo. Si estos valores cambian, entonces la salida (el valor acordado) debe cambiar; sin embargo, si sólo unos pocos de esos valores de entrada cambian, es posible evitar el cambio en el valor de salida. Así pues, cuando un cambio en la salida se realiza, entonces la inestabilidad del sistema aumenta. Por lo tanto, buscamos las condiciones necesarias para describir la peor situación en la inestabilidad de un sistema distribuido de consenso de larga vida.

Uno de los primeros modelos propuestos es el de entradas geodésicas [2], donde los procesos participantes en el sistema pueden cambiar su valor propuesto a lo más una sola vez. Dado lo muy limitado que este modelo puede ser, los investigadores mostraron comportamientos interesantes. Posteriormente, el modelo sufrió una variación; ahora los procesos participantes estaban organizados como vectores que formaban un conjunto de vértices, los cuales se relacionaban si y sólo si entre dos vectores había sólo una entrada distinta. Dando con ello el modelo de un hipercubo [4] donde cada vértice tiene asociada una probabilidad de desplazamiento de acuerdo a la cantidad de entradas de un valor dado.

Para un camino aleatorio sobre el hipercubo, nos interesa la cantidad de pasos realizados por un vértice en particular (el vértice con $t$ entradas de un valor dado). Ese número de pasos dividido entre la longitud del camino aleatorio es ahora lo que llamaremos inestabilidad.

En la práctica existen varias razones para preferir un sistema de consenso continuo estable: algunos dispositivos son discretos, como aquellos que determinan si una compuerta se abre o se cierra. Existe también la posible alargamiento operacional del tiempo de uso, digamos al encender o apagar un motor. La energía u otro recurso de consumo es proporcional al número de transiciones entre encendido y apagado de un motor, reduciendo con ello su tiempo de vida útil.

Existen muchas aplicaciones del consenso de larga vida, dando con ello toda una rama de investigación y aplicaciones para el desarrollo de los sistemas distribuidos computacionales.

## 1.5. Una breve reseña de la investigación relacionada al consenso

Hablar de los sistemas distribuidos de consenso es hablar sin lugar a dudas de algunos de los trabajos de Leslie Lamport [20]. Ab initio, cuando se acopló libremente a un conjuntos de procesadores que interactuaban mediante una red de comunicación, y que desde el punto de vista de un procesador en específico, el resto de los procesadores y sus respectivos recursos son remotos, fue la instauración de lo que hoy conocemos como sistema distribuido [8]. Las investigaciones de Lamport han contribuido esencialmente en los fundamentos de la teoría del cómputo distribuido. Entre sus artículos más destacados podemos mencionar: "Tiempos, relojes y ordenación de eventos en un sistema distribuido" [24] y "Fotos Instantaneas Distribuidas: Determinando el Estado Global de un Sistema Distribuido" [26]; en los cuales introduce la noción de causalidad de eventos, consistencia de relojes lógicos y cortes en ejecuciones. Estos conocimientos se han utilizado en la sincronización de relojes físicos entre procesadores remotos, implementación de sistemas distribuidos replicados, y la creación de algoritmos distribuidos con exclusión mutua; donde un procesador $p_i$ al cual se le ha asignado un recurso, lo tiene que liberar antes de que se pueda asignar a algún otro procesador $p_j$, las peticiones a recursos se atienden en el orden en que se hicieron y se garantiza que si todo procesador libera su recurso entonces todo pedido es atendido tarde o temprano.

También podemos mencionar el artículo: "El Problema de los Generales Bizantinos" [23], donde su estudio es el manejo de un sistema distribuido, cuyos procesos pueden presentar comportamientos completamente arbitrarios, sea por errores o por ejecuciones malintencionadas. A esto podemos añadir también que se han desarrollado algoritmos que toleran un número determinado de fallas bizantinas, como en el artículo [1] donde se muestra un algoritmo distribuido que resuelve consenso si el número de procesos con comportamiento impredecible es menor a un cuarto del total de procesos.

El término se refiere al problema de acuerdo entre generales del imperio Bizantino, quienes deben de decidir si atacar o no a un enemigo [27]. El problema es complicado por la separación geográfica de los generales, quienes se comunican enviando mensajes unos con otros, y por la presencia de traidores entre los generales. Estos traidores pueden actuar arbitrariamente, incitando a algunos generales a atacar, o forzando una decisión que no es consistente con el deseo de la mayoría.

Otro de los trabajos más mencionados de Lamport es el algoritmo de Paxos ($\Pi\alpha\chi o\sigma$) [13] y [14]. En este, se resuelve el acuerdo en los valores de entrada sobre un conjunto de máquinas de estados replicadas [22]. Hacia finales de los años 80's del siglo XX, la Systems Research Center implementó un sistema de archivos tolerantes a fallos, y afirmaban que dicho sistema podía mantener consistencia a pesar de cualquier número de fallas no bizantinas, y podría hacer progreso si cualquier mayoría de los procesadores hubiesen trabajado. Como muchos de tales sistemas, este era bastante simple cuando no había errores, pero tenía un algoritmo complicado para el manejo de fallas basado en tomar con cuidado todos los casos que los implementadores

pudieran pensar. A raíz de ello, Leslie Lamport descubrió el algoritmo de Paxos, el cual es un protocolo de consenso de tres fases. Reconoció que son necesarias tres fases para evitar un bloqueo del sistema en la presencia de una falla arbitraria simple.

Inspirado por el éxito y la popularización que el problema del consenso tenía al describirse con generales bizantinos, decidió dar el algoritmo en términos de un parlamento de una antigua isla griega [29]. Se sugirió para ello el nombre de la isla de Paxos; y dió a los legisladores griegos nombres de científicos de la computación trabajando en el área utilizando un falso dialecto griego. Así pues, escribió acerca de una antigua civilización egea y su perdido protocolo parlamentario. Además, el mismo Leslie Lamport expuso algunas pláticas y presentaciones del algoritmo de Paxos vestido como un arqueólogo.

Sin embargo, su sentido del humor no fue apreciado; la gente que oía sus conferencias recordaba al "Indiana Jones" y su encuentro arqueológico pero no el algoritmo. Los que leían el artículo se distraían tanto en la parábola griega que no entendían el algoritmo. Incluso, a la gente que afirmaba haber leido el artículo, Lamport les escribía un par de meses después la siguiente pregunta: ¿Puede usted implementar una base de datos distribuida tal que pueda tolerar las fallas de cualquier número de sus procesadores (posiblemente de todos ellos) sin perder consistencia, y que reanude un comportamiento normal cuando más de la mitad de los procesadores estén trabajando con propiedad de nuevo? ¡Ninguno de ellos notó cualquier conexión entre esta pregunta y el algoritmo de Paxos!

El artículo original de "The Part-Time Parliament" fue presentado para TOCS en 1990, no fue aceptado para su publicación diciendo los revisores que el artículo, ad usum, era medianamente interesante y no muy importante, pero que todas las cosas de Paxos debían ser quitadas. Sin embargo, Lamport, un tanto molesto, puso el artículo a remotis. Pero unos años más tarde, gente de la Systems Research Center necesitaban algoritmos para los sistemas distribuidos que habían construido y Paxos proporcionaba justo lo que ellos necesitaban.

Efectivamente, no hubo problema alguno con el algoritmo implementado y fue entonces cuando Lamport pensó en que el momento de ponerlo a publicación había llegado.

Mientras tanto, a pesar de todo lo que había pasado, hubo alguien que si reconoció la significancia del algoritmo: Butler Lampson. Lo mencionó en sus conferencias y en sus artículos[19]; además de que interesó a otros investigadores para publicar su versión de especificaciones y pruebas. Convencieron a Lamport de que su artículo necesitaba revisión para tomar en cuenta el trabajo que se había publicado entre esos años. Así, finalmente, con algunas revisiones y comentarios de otros investigadores, el artículo apareció publicado [13].

No todos los artículos han pasado las peripecias que ha pasado un algoritmo tan importante como lo es Paxos.

Uno de los resultados más importantes dentro del cómputo distribuido, con respecto a la imposibilidad de diseñar una solución determinística al problema del consenso, en un sistema distribuido asíncrono propenso a fallas de caida en sus procesos; la dieron Michael J. Fischer, Nancy A. Lynch y Michael S. Paterson [11]. En el mismo sentido, Idit Keidar y Sergio Rajsbaum, dieron una prueba simple e intuitiva sobre una cota inferior de $f + 2$ rondas para resolver el problema del consenso uniforme [7]. Id est, que para cada algoritmo de consenso que tolere $t$ fallas y que para cada $f \leq t - 2$, existe una ejecución con $f$ fallas que requiere de $f + 2$ rondas para resolverlo.

Algunas versiones de algoritmos de consenso de larga vida aparecieron en "The Part-Time Parliament" [13], y en "Revisando el Algoritmo de Paxos" [19]; donde ya se consideraban algunos modelos. El manejo de inestabilidad en estos sistemas, tanto con entradas binarias como multivaluadas fue desarrollado en [2] y [5] respectivamente.

## 1.6. Resultados previos del consenso de larga vida y aplicaciones

Actualmente, en muchos algoritmos de consenso, existe alguna libertad en la forma en que los procesadores pueden escoger su decisión. Típicamente, después de algunas comunicaciones, un procesador descubre un conjunto de valores de entrada y entonces decide en uno de esos valores, usando algún criterio, tales como: decidir el máximo, el más frecuente ó el de mayor prioridad, etc. Usualmente, hay más de un criterio para el cual el algoritmo trabaja correctamente. Para entender cuáles son las estrategias más estables al momento de escoger los valores de decisión, comenzaremos por abstraer la forma del modelo de comunicación y el algoritmo específico de consenso utilizado, de la siguiente manera:

En cada fase el sistema comienza en algún estado $s$, y recibe un vector de entrada $x$. Como resultado de una ejecución particular, los procesadores convienen en algún valor binario $f(s, x) = b$, satisfaciendo los requerimientos del consenso de "una sola pasada", y llevando el sistema a algún estado $g(s, x) = s'$. En un sistema distribuido especial, las funciones $f$ y $g$ pueden depender de la ejecución en particular, y de ahí que puedan ser no deterministas.

Es necesario abstraer el modelo de sistema distribuido a fin de hacerlo lo más general posible y no restringirlo a propiedades tales como el tipo de canales de comunicación o la arquitectura de los procesadores.

Estos resultados pueden ser interpretados como "lo mejor posible" en el sentido de que muestran cuál es la mejor estabilidad posible que puede ser alcanzada en principio, abstrayendo las propiedades del sistema distribuido en particular. Esto es; hacemos estas funciones deterministas escogiendo una ejecución con el más grande número de cambios de decisión.

Ha habido diversas caracterizaciones acerca de la estabilidad en los sistemas de acuerdo a sus propiedades de simetría y de memoria. Entre los resultados destacados en el artículo [2], se han obtenido cotas superiores e inferiores para la estabilidad de sistemas con memoria; mostrando por ejemplo, que un bit de memoria es suficiente para obtener un sistema de consenso óptimo. De igual manera, se mostró la estabilidad realizable para un sistema simétrico sin memoria.

Para el caso de estabilidad en sistemas de consenso de larga vida con entradas multivaluadas [5], los autores tuvieron que recurrir a técnicas topológicas para complejos de alta dimensión (es decir $\geq 3$). Uno de los requerimientos de validez del consenso multivaluado es que la decisión puede estar entre los valores de dos procesos correctos. Así pues, estudian dos tipos de sistemas: los sistemas de valor exacto ($EV$), donde el valor de decisión es el valor de un proceso correcto; y los sistemas de rango de valores ($RV$), donde el valor de decisión del sistema está en el rango de valores que aportan procesos correctos.

Basados en el resultado de que el consenso multivaluado puede reducirse a consenso binario [3] nos enfocaremos en modelos que asumen entradas binarias.

Veremos el desarrollo de un sistema simple llamado $BP$, tal que dado un parámetro $t$, resuelve consenso de larga vida tolerando $t$ fallas. Este parámetro $t$, donde $0 \leq t \leq \frac{n}{2}$, tiene algunos requerimientos adicionales tales como: si a lo más $t$ entradas en el vector son iguales a un valor $b$, entonces la decisión que se toma es igual a $1 - b$; cuando $t = 0$ el conjunto de requisitos de tolerancia a fallos es vacio; y cuando $2t + 1 = n$ entonces requiere que la función de decisión de los procesos sea mayoría.

Este sistema $BP$ comienza con inestabilidad igual a 1 cuando $n > 4t$; y converge hasta una inestabilidad igual a $2t + 1$ de forma monótona cuando $t$ se acerca a satisfacer la ecuación $n = 2t + 1$.

De igual forma veremos un sistema simétrico sin memoria $BS$ cuya inestabilidad es $2t+2$ para cualquier $n \geq 2t + 2$. En general los sistemas simétricos sin memoria poseen una inestabilidad muy grande.

En el artículo [2] los autores dejan abierta la pregunta de si existe un sistema sin memoria con inestabilidad menor que $2t + 1$ para cuando $n > 2t + 1$.

Estos modelos servirán para cuando más adelante realicemos algunas variantes en la forma de restricción de las entradas; dando con ello sistemas de mayor flexibilidad con comportamientos interesantes.

Concluimos el capítulo señalando que el uso de sensores replicados de una manera exacta y confiable ha desarrollado una gran atención de los sistemas distribuidos de tiempo real [21], y para largo tiempo [28]. Usualmente se utilizan funciones de promedio de manera independiente de ejemplo a ejemplo, algunas veces combinándolas con protocolos de consenso como en [25].

Un tipo especial de sensores, son los sensores de tiempo; es decir, dispositivos para medir el tiempo. El problema de sincronización de relojes es realizar la lectura de distintos dispositivos de reloj (sensores) y producir un sólo valor de tiempo, el cual sea exacto y confiable tanto como sea posible y de la manera más eficiente. Existe mucha literatura acerca del problema de sincronización de relojes y sus correspondientes funciones de promedio (vea [34]). En los sistemas que requieren sincronización de relojes, un tipo de estabilidad a menudo es deseado, pues se trata de evitar brincos drásticos en los valores de tiempo, principalmente por amortizar el ajuste del reloj en tiempo real [35].

# Capítulo 2

# Estabilidad en sistemas de consenso continuo

Los sistemas distribuidos de consenso continuo (o también llamados de *larga vida*) son una colección de elementos cuyo modelo matemático tiene el proposito de realizar acuerdo entre los valores propuestos por un conjunto de procesos; los cuales están a su vez, posiblemente cambiando su propio valor propuesto conforme pasa el tiempo. Si bien, este problema se presenta comúnmente en los sistemas computacionales, las soluciones para consenso de "una sola pasada" propuestas por varios investigadores como Leslie Lamport [13] y [14], Rachid Guerraoui [10] y Michel Raynal [1] pueden aplicarse a cualquier sistema distribuido en general. El consenso *continuo* o de *larga vida* aparece en los trabajos de Lior Davidovitch, Shlomi Dolev y Sergio Rajsbaum [5]; Shlomi Dolev y Sergio Rajsbaum [2]; posteriormente Florent Becker, Sergio Rajsbaum, Ivan Rapaport y Eric Rémila [4] y se trata de la versión dinámica y cambiante en el tiempo del consenso de "una sola pasada".

En este capítulo veremos las nociones introducidas sobre inestabilidad en sistemas con modelos basados en entradas geodésicas (i.e. que cambian su valor de entrada a lo más una sola vez). La idea principal es mostrar un escenario de las investigaciones y resultados previos al modelo de hipercubo[1] donde veremos cotas alcanzables para modelos con memoria y sin memoria.

Comenzaremos con una serie de definiciones (sección 2.1) y posteriormente trataremos la inestabilidad de un sistema con memoria llamado $BP$ (sección 2.2.1). A continuación, haremos el análisis de inestabilidad para $BS$, el cual es un sistema sin memoria (sección 2.3.1). Ambos sistemas se basan en el modelo donde los procesos participantes pueden cambiar su valor de aportación a lo más una sola vez y esto nos servirá de base para las modificaciones que realizaremos en el capítulo 3.

## 2.1. Definiciones

Empezaremos con una formalización abstracta de un sistema de consenso de larga vida y las medidas de estabilidad. La formalización es independiente de un modelo específico de cómputo distribuido, a fin de entender las cuestiones básicas de estabilidad. Asumiremos entradas binarias en los vectores, i.e. que los valores propuestos por los procesos son $0$ ó $1$ (v. gr. realizar *commit* o *rollback*).

[1] En el modelo de hipercubo se piensa en los posibles caminos que podría tomar una secuencia de vectores de entrada si la $i-$ésima componente cambia. Esta modificación la estudiaremos en el capítulo 3

**Definición 1** *Un sistema de consenso de larga vida $D$ se representa como una tupla $D = (n, t, S, g, f)$ donde:*

- *$n$ es el número de procesos participantes en el sistema.*
- *$t$ es el parámetro de tolerancia a fallos tal que $0 \leq t \leq \lfloor n/2 \rfloor$; con un requerimiento adicional: si a lo más $t$ entradas son iguales a un valor $b$, entonces la decisión de salida es igual a $1 - b$.*
- *$S$ es el conjunto de posibles estados del sistema y contiene al estado especial inicial $s_0$.*
- *$g$ es la función de transición tal que $g : \mathbb{Z}_2^n \times S \to S$.*
- *$f$ es la función de decisión tal que $f : \mathbb{Z}_2^n \times S \to \mathbb{Z}_2$.*

La función de decisión mapea un vector de $n$ entradas de procesos y un estado del sistema (si es con memoria), a un valor de decisión. La función de transición mapea un vector de $n$ entradas de procesos y el estado actual del sistema, al siguiente estado del sistema. El parámetro $t$ puede ser usado para modelar una situación donde a lo más $t$ procesos pueden fallar.

Si un vector de valores introducidos por los procesos contiene más de $t$ $0's$ ó $t$ $1's$, este es necesariamente el caso en que dos valores distintos de entrada existen, y de ahí que el sistema sea libre de decidir en alguno de ellos. Sin embargo, si a lo más $t$ procesos regresan un valor $b$, es posible que ellos estén fallidos y la fuente produce sólo valores $1 - b$. En este caso el sistema es forzado a decidir $1 - b$.

En el caso de consenso multivaluado, es necesario definir dos clases de funciones: una de valor exacto y otra de rango de valores (Tal como se explica en [5]).

**Definición 2** *Una secuencia de vectores de entrada es geodésica si la entrada de cada proceso es cambiada a lo más una sola vez.*

Por ejemplo: la siguiente secuencia de vectores en $\mathbb{Z}_2^5$ tiene a lo más un sólo cambio en cada entrada.

$$[0, 1, 0, 1, 1] \quad \to \quad [\underline{1}, 1, 0, 1, 1] \quad \to \quad [\underline{1}, \underline{0}, 0, 1, 1] \quad \to \quad [\underline{1}, \underline{0}, \underline{1}, 1, \underline{0}]$$

La entrada subrayada ya ha cambiado su valor.

**Definición 3** *Una fase es el intervalo entre dos cambios de entrada.*

Ejemplo: vemos el cambio en la segunda entrada

$$[1, 1, 0, 1, \underline{1}] \quad \underset{fase}{\to} \quad [1, \underline{0}, 0, 1, \underline{1}]$$

En cada fase, el procesador obtiene un $n-$vector de entrada, se comunica con los otros procesadores y produce un valor (o un vector) de salida satisfaciendo los requerimientos del consenso de "una sola pasada".

**Definición 4** *Definimos la inestabilidad de un sistema de consenso de larga vida como el peor caso en el número de cambios en la decisión de salida sobre cualquier secuencia geodésica de entrada.*

Así pues, si el número más grande de cambios de decisión para cualquier ruta geodésica de entrada es $d$; entonces la inestabilidad del sistema distribuido de consenso de larga vida $D$ es igual a $d$. Lo escribimos $Inestabilidad(D) = d$.

**Notación:** para $x \in \mathbb{Z}_2^n$ y $b \in \{0, 1\}$ sea $\#b(x)$ el número de componentes de $x$ con valor $b$.

## 2.2. Estabilidad en sistemas con memoria

Comenzaremos diferenciando los modelos de acuerdo a su propiedad de poder recordar ejecuciones previas o no; es decir, modelos con memoria y modelos sin ella. Para el caso con memoria (llamado $BP$) y sin memoria (llamado $BS$) se explican su inestabilidad respectiva dada siempre una secuencia de vectores geodésicos de entrada.

Recordemos que por inestabilidad llamamos al peor número de cambios en la decisión de salida del sistema, dada cualquier entrada geodésica.

### 2.2.1. El sistema $BP$

A continuación se presenta un sistema de consenso de larga vida, denominado $BP$, con sólo un bit de memoria. En el artículo [2] se prueba que la inestabilidad de $BP$ es una función la cual comienza en 1 cuando $n > 4t$, y converge de manera monótona hacia $2t + 1$, conforme $t$ se acerca a $\frac{n-1}{2}$. En el límite, cuando $n = 2t + 1$, la inestabilidad de $BP$ es $2t + 1$. También se probó que esta inestabilidad es mínima: i.e. que ningún sistema tiene mejor estabilidad que $BP$. Otra característica interesante de $BP$ es que es simétrico en el sentido de que la decisión es tomada en función del número de $0's$ y de $1's$ en el vector de entrada y en el estado actual del sistema; pero no en la posición específica de esos bits (es decir, que ningún procesador tiene preferencia sobre los otros).

Veamos el primer resultado interesante sobre la inestabilidad mostrada por $BP$ cuando se tiene que $n > 4t$.

**Lema 1** *Sea $BP = (n, t, S, g, f)$ un sistema de consenso de larga vida donde $t < \frac{n}{4}$ y cuya función de decisión es:*

$$f(x,s) = \begin{cases} 0 & \textit{si } \#1(x) \leq t \\ 1 & \textit{si } \#0(x) \leq t \\ maj(x) & \textit{si } s = s_0 \\ s & \textit{en otro caso} \end{cases}$$

*Entonces su inestabilidad es a lo más 1.*

**Prueba:** sea $x_0, x_1, x_2 \ldots$ una secuencia geodésica de entrada cualquiera, donde $b_0, b_1, b_2 \ldots$ son los valores decididos en cada fase (suponemos que las entradas son valores binarios).

$\#b_0(x_0) \geq \frac{n}{2} + 1$ i.e. Se decide lo que propone la mayoría.

Llegamos hasta la decisión $b_r$ donde se realiza el primer cambio; entonces $\#b_0(x_r) = t$. ($b_r$ es el complemento de $b_0$).

En otras palabras; del número original de bits propuestos por la mayoria en el vector $x_0$, solamente quedaron $t$ en el vector $x_r$.

$\therefore$ el número $\kappa$ de bits que cambiaron al valor $b_r$ fue: $\kappa \geq \frac{n}{2} + 1 - t$

Además sabemos que $t < \frac{n}{4}$ por ende tenemos que:

$$\kappa \quad \geq \quad \frac{n}{2} + 1 - t \quad > \quad \frac{n}{2} + 1 - \frac{n}{4} \quad = \quad \frac{n}{4} + 1$$

En este punto nos encontramos en un estado $s \neq s_0$; por lo tanto, según la función de decisión, para poder cambiar otra vez el valor de decisión es necesario que $\#b_r(x_i) \leq t$ para alguna $i > r$.

Pero vemos que ($\frac{n}{4} + 1$) el número de bits que cambiaron su valor a $b_r$ es mayor que el máximo necesario ($t$) para cambiar la decisión y como los vectores son geodésicos:

$\therefore$ Ya no pueden volver a cambiar su valor de decisión.

□

Ahora consideraremos la situación donde $n = 2t + 1$. Note que en este caso no hay libertad para diseñar $f$; todo sistema de consenso de larga vida es idéntico, con la función de decisión de mayoría ($maj(x)$):

$$f(x, s) = \begin{cases} 0 & \text{si } \#1(x) \leq t \\ 1 & \text{si } \#0(x) \leq t \end{cases}$$

De ahí que $g$ sea irrelevante.

**Lema 2** *La inestabilidad de cualquier sistema de consenso de larga vida con $t = \frac{n-1}{2}$ es de $2t + 1$.*

**Prueba:** la siguiente entrada geodésica produce $2t + 1$ cambios de decisión. Las decisiones se indican bajo los vectores de entrada.

$$\begin{array}{ccccccc} 0^t1^{t+1} & \rightarrow & 0^{t+1}1^t & \rightarrow & 10^t1^t & \rightarrow \\ 1 & & 0 & & 1 & \end{array}$$

$$\begin{array}{ccccccc} 10^{t+1}1^{t-1} & \rightarrow & 1^20^t1^{t-1} & \rightarrow & 1^20^{t+1}1^{t-2} & \rightarrow \\ 0 & & 1 & & 0 & \end{array}$$

$$\begin{array}{ccc} \ldots & \rightarrow & 1^t0^{t+1} \\ \ldots & & 0 \end{array}$$

En realidad, cualquier ruta geodésica que comienza digamos con $t$ $0's$ y $(t+1)$ $1's$ toma como decisión el valor 1. Posteriormente, un proceso cambia su valor propuesto de 1 a 0 dando como resultado $t$ $1's$ y $(t+1)$ $0's$; y según la función de decisión, el sistema decide 0. A continuación otro proceso cambia su valor de 0 a 1, teniendo así nuevamente $t$ $0's$ y $(t+1)$ $1's$; y por ello según la función de decisión, el sistema vuelve a cambiar su salida a 1.

Esta serie de pasos con cambio de decisión en la salida se puede repetir únicamente cuando se puede alternar los valores $t$ y $(t + 1)$ entre los $0's$ y los $1's$. i.e. que un bit con valor 1 cambia a 0 y a continuación un bit con valor 0 cambia a 1; repitiendo este proceso $2t + 1$ veces.

Una secuencia geodésica puede cambiar la decisión a lo más $2t + 1$ veces, ya que una ruta geodésica puede cambiar cada bit a lo más una vez; y de ahí que contenga a lo más $n = 2t + 1$ vectores.

□

## 2.3. Estabilidad en sistemas sin memoria

Los sistemas $(n, t, S, g, f)$ sin memoria, son aquellos cuyo conjunto $S = \{s_0\}$. Ya que sólo hay un estado, la función de transición $g$ es irrelevante; por ello denotamos a un sistema sin memoria con $(n, t, f)$.

Los resultados para sistemas con memoria se mantienen también para sistemas sin memoria, con las mismas pruebas: si $t < \frac{n}{4}$ la inestabilidad es 1 al menos (lema 1), y si $t = \frac{n-1}{2}$ la inestabilidad es $2t + 1$ (lema 2). Por otro lado, note que para $2t + 1 < n \leq 4t$ la inestabilidad del sistema con memoria $BP$ es estrictamente menor que $2t + 1$.

Shlomi Dolev y Sergio Rajsbaum [2] mostraron que cualquier sistema simétrico sin memoria tiene inestabilidad de al menos $2t + 2$. También presentaron un sistema no-simétrico sin memoria con inestabilidad $2t + 1$. En su artículo se deja abierta la pregunta de que si existe un sistema sin memoria con inestabilidad menor que $2t + 1$ para cuando $n > 2t + 1$.

### 2.3.1. El sistema $BS$

Comenzaremos describiendo un sistema simétrico sin memoria $BS$ con inestabilidad $2t + 2$ para cualquier $n \geq 2t+2$. Entonces se mostrará que no hay sistema simétrico sin memoria que tenga mejor estabilidad. Esta cota inferior implica que sistemas simétricos sin memoria tengan estrictamente estabilidad más baja que sistemas con un bit de memoria, ya que el sistema $BP$ de la sección (2.2.1) tiene inestabilidad de a lo más $2t + 1$ para cualquier $n \geq 2t + 1$.

La función de decisión para $BS$ es:

$$f(x, s) = \begin{cases} 0 & \text{si } \#1(x) \leq t \\ 1 & \text{en otro caso} \end{cases}$$

**Teorema 1**

$$Inestabilidad(BS) = \begin{cases} 2t+1 & si \;\; n = 2t+1 \\ 2t+2 & si \;\; n > 2t+1 \end{cases}$$

**Prueba:** la primer línea fue probada en lema 2. Para probar la segunda línea ahora asumamos que $n \geq 2t + 2$.

Primero probaremos que $Inestabilidad(BS) \geq 2t + 2$. Consideremos la siguiente entrada geodésica, comenzando con:

$$x_0 = 1^{t+1}0^{n-t-1}$$

e intercambiamos un bit a la vez, comenzando de 1 a 0 y luego de 0 a 1, etc.

$$\begin{array}{ccccc} 1^{t+1}0^{n-t-1} & \rightarrow & 01^t0^{n-t-1} & \rightarrow \\ 1 & & 0 & \end{array}$$

$$\begin{array}{ccccc} 01^{t+1}0^{n-t-2} & \rightarrow & 0^21^t0^{n-t-2} & \rightarrow \\ 1 & & 0 & \end{array}$$

$$\begin{array}{ccc} \ldots & \rightarrow & 0^{t+1}1^{t+1}0^{n-2t-2} \\ \ldots & & 1 \end{array}$$

Note que $n - 2t - 2 \geq 0$, porque esta es una ruta geodésica bien definida. Se sigue entonces, que $Inestabilidad(BS) \geq 2t + 2$.

Ahora probaremos que $Inestabilidad(BS) \leq 2t + 2$. Para ello consideremos cualquier ruta geodésica que (sin pérdida de generalidad ya que $BS$ es simétrico) comience:

$$x_0 = 1^l 0^{n-l}$$

Si $f(x_0) = 1$ entonces $l \geq t + 1$. Para obtener un vector de entrada $x_1$, con un valor de decisión diferente, al menos $l - t$ 1's tienen que ser cambiados a 0, de manera que a lo más $t$ 1's queden en $x_1$ y $f(x_1) = 0$. Entonces, es necesario alternar entre pasos de intercambio de un bit 0 a un bit 1, y un bit 1 a un bit 0. Ahora en $x_1$ hay a lo más $t$ 1's que pueden cambiar todavia, el segundo paso puede ser ejecutado a lo más $t$ veces, dando un número total de intercambios de 1 a 0 de $t + 1$. Ya que el número de ejecuciones del primer tipo de paso está acotado por el número de ejecuciones del segundo tipo de paso, a lo más $t + 1$ intercambios de 0 a 1 pueden ser realizados. Así el número total de cambios de decisión es a lo más $2(t + 1)$.

De otra forma, si $f(x_0) = 0$ y además $l < t + 1$, en este caso es necesario comenzar por intercambiar al menos $t + 1 - l$ bits con valor 0 a valor 1 a fin de obtener un vector $x_1$ con al menos $t + 1$ bits iguales a 1, y de ahí con $f(x_1) = 1$. Así, $x_1$ comienza con a lo más $n - l - (t + 1 - l) = n - t - 1$ bits 0 que pueden intercambiarse. Sin embargo empezando con $x_1$, a lo más $l$ cambios de 1 a 0 pueden ser desarrollados, y de ahí a lo más $t$ cambios de decisión de 1 a 0. Ya que esta cota en el número de cambios de decisión de 0 a 1 empieza en $x_1$, el número total de cambios de decisión comenzando con $x_1$ es a lo más $2t$, y por lo tanto, en este caso, el número total de cambios de decisión es a lo más $2t + 1$.

□

Como conclusión a este capítulo podemos decir que, para el modelo de consenso de *larga vida* con entradas *geodésicas*, se mostraron sistemas en los cuales se diferenció la inestabilidad presentada de acuerdo a su propiedad de poseer memoria o no. Para el caso con memoria (llamado sistema $BP$), se mostró una inestabilidad que comienza en 1 cuando el el parámetro de tolerancias a fallos $t$ es menor a un cuarto del número $n$ de procesos participantes en el sistema; y dicha inestabilidad converge de manera monótona hacia $2t + 1$ cuando $t \leq \frac{n-1}{2}$. En el límite cuando $n = 2t + 1$ la inestabilidad de $BP$ es precisamente: $2t + 1$.

En el caso sin memoria (llamado sistema $BS$), se mostró que la inestabilidad es mayor que en el caso con memoria, es decir $2t+2$ cuando el parámetro de tolerancias a fallos $t \leq \frac{n-2}{2}$. Ahora bien, recordemos que en este modelo, por inestabilidad entendemos el peor número de cambios en la decisión de salida del sistema, dada cualquier entrada geodésica. En el siguiente capítulo haremos una modificación a nuestra definición de inestabilidad (llamandola inestabilidad promedio) y a nuestros vectores de entrada, a fin de que no se restrinja únicamente a entradas cuyo cambio se permite una sola vez (entradas geodésicas).

# Capítulo 3

# Inestabilidad promedio y modelo con cadenas de Markov

Sea $D = (n, t, S, g, f)$ un sistema de consenso de larga vida con $n$ procesadores participantes y una tolerancia a fallos $t$, operando en rondas síncronas. Tenemos por tanto una secuencia de vectores de entrada (recordemos que $n$ es el número de entradas en el vector y que cada una de ellas representa el valor binario propuesto por algún proceso); sin embargo hasta aquí hemos asumido que esa secuencia es geodésica, dando con ello la limitante de que la secuencia puede ser finita, pues el total de cambios de entrada posibles es el número de entradas en el vector, ya que cada uno de las $n$ entradas puede cambiar a lo más una vez.

Shlomi Dolev y Sergio Rajsbaum [2] mostraron la inestabilidad para los modelos cuyas rutas geodésicas eran vectores con entradas binarias. Posteriormente ellos mismos junto con Lior Davidovitch [5] trataron la inestabilidad pero en el caso donde las secuencias geodésicas eran vectores cuyas entradas podian cambiar de entre un rango de valores a lo más una sola vez.

El caso de vectores con entradas multivaluadas se vuelve mucho más difícil, por ello, sabiendo que el consenso multivaluado puede reducirse a problemas de consenso binario [3], seguiremos manejando entradas binarias, pero quitaremos el requerimiento de secuencias geodésicas de los vectores de entrada.

La necesidad que surge al sustituir el requerimiento de secuencias geodésicas, y pensando precisamente en el (o los) camino(s) que podría tomar una nueva secuencia si la $i-$ésima entrada cambia; nos plantea la idea de que debemos relacionar a cada vector con tantos elementos como número de entradas haya en él.

Es decir, necesitamos idear un grafo $G_n = \langle V_n, E_n \rangle$ cuyo conjunto de vértices $V_n$ sea exactamente el conjunto de vectores; y que el conjunto de aristas $E_n$ sea la relación entre vectores que difieran en exactamente una sola entrada.

Tenemos que cada vértice de $V_n$ es un vector de dimensión $n$, y que $\mid V_n \mid$ es igual al número de posibles vectores binarios de dimensión $n$; entonces $\mid V_n \mid = 2^n$. Si además cada vértice tendrá grado $n$, el grafo $G_n$ debe tener las características de un hipercubo de dimensión $n$.

El término de hipercubo [15] fue acuñado por el matemático inglés Charles Howard Hinton en 1888 y se obtiene del desarrollo del polinomio $(2x + 1)^n$ donde $n$ es el valor de la dimensión y $x$ es el valor del largo, ancho, alto, ..., etc. de la figura polidimensional equilátera.

Ahora bien, podemos representar el sistema $D$ mediante un hipercubo con $2^n$ vértices, donde cada uno de esos vértices representa un estado con los valores binarios propuestos por cada proceso; y con esto podemos considerar nuevas secuencias no geodésicas que pueden tomarse de manera aleatoria sobre aristas del hipercubo.

Dando una secuencia aleatoria de vectores de entrada (vértices en el hipercubo), la inestabilidad promedio del sistema estará dada por una razón de las veces que la función $f$ cambia su valor de salida sobre dicha

secuencia. El objetivo es que tal función minimice la inestabilidad en dos posibles escenarios: sin memoria (que llamaremos Sistema $D_0$) y con memoria (que llamaremos Sistema $D_1$).

Resumiendo: dados los enteros no negativos $n$ y $t$, con $n \geq 2t + 1$; el hipercubo de dimensión $n$ es un grafo cuyos vértices $V_n = \{0, 1\}^n$ son vectores binarios $n-$dimensionales y el conjunto de aristas $E_n$ se conforma de todos los pares de vértices cuyos vectores difieren en exactamente una componente. En otras palabras, un vector de entrada del sistema está relacionado con otro si y sólo si existe solamente un cambio en la entrada de uno de los procesos.

Veamos un ejemplo con $n = 4$ procesadores. El hipercubo relacionado a ese sistema tendrá $2^4 = 16$ vértices (figura 3.1).

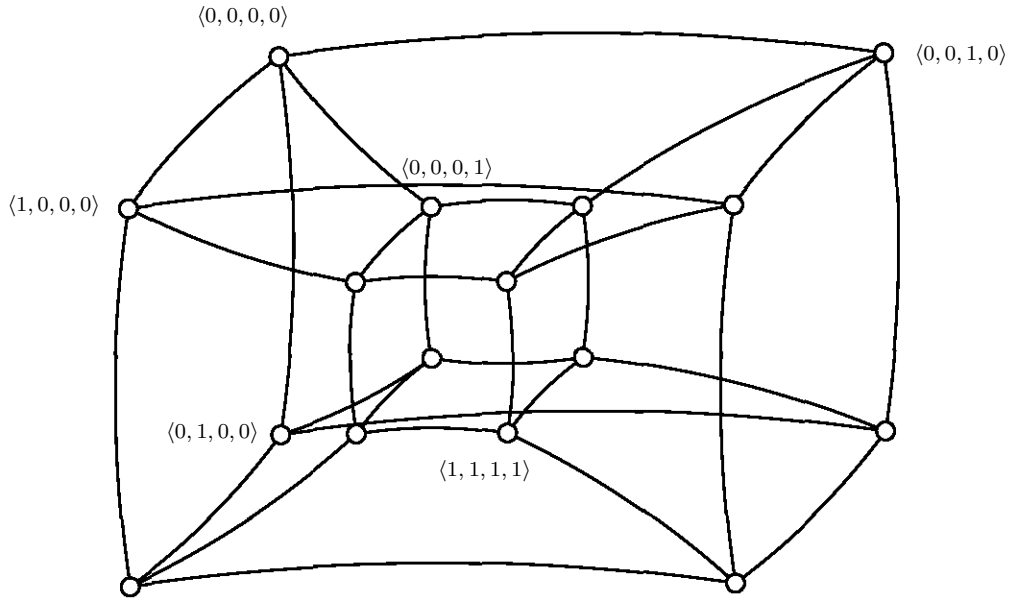


Figura 3.1: Ejemplo de hipercubo asociado a un sistema con $4$ procesos.

$E_n$ es el conjunto de aristas del hipercubo. Como tenemos $2^n$ vértices y cada uno de ellos está relacionado con otros $n$, entonces tenemos $n \cdot 2^n$ aristas. Sin embargo, de esta forma estamos contando aristas de manera repetida (dos veces para ser exacto). Por consiguiente, tenemos $\frac{n \cdot 2^n}{2}$ aristas; i.e. que $\mid E_n \mid = n \cdot 2^{n-1}$.

Sean $x_0, x_1, x_2 \ldots$ los vértices de un *camino aleatorio* en el hipercubo. Decimos que se trata de un camino aleatorio porque el vértice inicial se escoge de acuerdo a alguna distribución probabilística $\lambda$ y posteriormente los siguientes vértices del camino se seleccionan aleatoriamente de acuerdo a una probabilidad asociada. Consideraremos también que la función de decisión $f$ asigna un valor de salida $d$ a cada $x_i$ de modo que satisface el requerimiento de tolerancia a fallos: $\#d(x_i) \geq t + 1$. En general, la función $f$ da como salida el valor $d$ basado en los vértices previos del camino.

Con esta representación de un sistema de larga vida $D$ mediante un hipercubo de dimensión $n$, podemos escindir la nececidad en el requerimiento de secuencias geodésicas. A partir de ahí podemos también tener una base para realizar investigación cuando veamos el comportamiento de modelos sin memoria y con memoria, en experimentos de simulación realizados a caminos aleatorios sobre el hipercubo.

Cabe señalar que esta serie de trabajos [4] deja abiertos varios problemas de investigación interesantes, tales como manejar el caso de consenso multivaluado o utilizar otras distribuciones probabilísticas en la

selección del camino aleatorio.

## 3.1. Representación e inestabilidad del sistema $D_0$

Basados en la definición 1 de sistema de consenso de larga vida; introduciremos la siguiente:

**Definición 5** *Una ejecución de un sistema $D$ es una secuencia $(x_0, s_0, d_0) \rightarrow (x_1, s_1, d_1) \rightarrow \ldots$ donde $s_{i+1} = g(x_i, s_i)$ y $d_i = f(x_i, s_i)$.*

Una tripleta $(x_i, s_i, d_i)$ es una configuración.

**Definición 6** *Sea el sistema sin memoria $D_0 = (n, t, f_0)$ con $f_0(x) = 1$ si y sólo si $x \in \Gamma_i$ con $i > t$ donde $\Gamma_i$ es el conjunto de vértices del hipercubo con $i$ 1's (y con $(n - i)$ 0's).*

Ya que serán útiles en el uso del modelo del hipercubo, sobre todo para la deducción de la curva teórica esperada, daremos la siguiente notación:

La distancia $dis(x, y)$ entre dos vértices $x, y$ es igual al número de entradas en las cuales ellos difieren.

Así pues, $dis(x, y) = h$ si y sólo si la ruta más corta entre $x$ y $y$ en el hipercubo es de longitud $h$.

Las esquinas del hipercubo son los vectores $0^n$ y $1^n$.

**Definición 7** *La $h-$vecindad $N^h(x)$ de un vértice $x$ del hipercubo, es el conjunto de vértices a distancia a lo más $h$ de $x$.*

Así, $N^t(0^n) = \{x \mid \#1(x) \leq t\}$, y similar para $1^n$. Como por lo general $n > 2t + 1$ entonces $N^t(0^n) \cap N^t(1^n) = \emptyset$

Asumimos que si $x$ es el vértice actual (que representa un vector de entrada), el próximo vértice $x'$ es tomado de una manera aleatoria uniforme, de entre los vértices a distancia uno de $x$ en el hipercubo. El vértice inicial se escoge de acuerdo a alguna distribución $\lambda$. Una vez que el vértice inicial $x_0$ es determinado, también lo es la configuración inicial $(x_0, s_0, d_0)$. La siguiente configuración se produce al escoger aleatoriamente un vecino de $x_0$, digamos $x_1$ y así la siguiente configuración $(x_1, s_1, d_1)$, donde $s_1 = g(x_0, s_0)$ y $d_1 = f(x_0, s_0)$. Esto es, un *camino aleatorio* en el hipercubo define una ejecución.

Formalmente, definimos el siguiente proceso de Markov: el conjunto de estados es $V_n \times S$ el cual es finito, y hay una transición de $(x, s)$ a $(x', s')$ si la arista $(x, x') \in E_n$ y $g(x, s) = s'$.

Se sigue que la secuencia de variables aleatorias $X_0, X_1, X_2, \ldots$ con posibles valores en $V_n \times S$, eligiendo $X_0$ de acuerdo a la distribución $\lambda$, es una cadena de Markov (como referencia vea el apéndice A.6).

Cada estado $(x, s)$ tiene asociado un valor de salida $f(x, s)$ que escribiremos $f(X_i) = d_i$. Sea $c_{\lambda,l}(D)$ la variable aleatoria definida por:

$$c_{\lambda,l}(D) = \frac{1}{l} \sum_{k=0}^{l-1} \mid d_{k+1} - d_k \mid$$

Desglosemos el significado de esta variable aleatoria: recordemos que la selección del vértice inicial depende de la distribución $\lambda$ de los vértices del hipercubo y de la longitud $l$ del camino aleatorio. Lo que nos indica la expresión dentro de la sumatoria es el número de cambios en los valores de decisión entre estados contiguos a lo largo del camino aleatorio. Sin embargo, como este número puede crecer tanto como la longitud $l$ del camino aleatorio crezca, resulta necesario normalizarlo dividiéndolo entre $l$.

**Definición 8** *La inestabilidad promedio de un sistema de consenso de larga vida $D$ es*

$$c(D) = \mathbb{E}(\lim_{l\to\infty} c_{\lambda,l}(D))$$

*donde $\mathbb{E}$ es el valor esperado (vea el apéndice A.3).*

Notemos que la inestabilidad promedio $c(D)$ ya no depende de la distribución $\lambda$ que tengan los vértices del hipercubo. Esto se debe a que la selección del vértice inicial no influirá en el valor esperado de $c_{\lambda,l}(D)$ conforme la longitud del camino tiende a infinito. Esta afirmación es válida para el modelo sin memoria (como lo es el sistema $D_0$), ya que al no recordar decisiones de estados previos resulta indistinto dónde comenzar el recorrido de un camino aleatorio infinito.

Ahora bien, es necesario probar la siguiente proposición antes de demostrar la inestabilidad que presenta el sistema $D_0$ (teorema 2).

**Proposición 1** *La inestabilidad promedio de un sistema de consenso de larga vida sin memoria $D$ es*

$$c(D) = \frac{\sum_{\{u,v\}\in E_n} \mid f(u) - f(v) \mid}{n \cdot 2^{n-1}}$$

**Prueba:** ya que la definición 8 nos dice que $c(D) = \mathbb{E}(\lim_{l\to\infty} c_{\lambda,l}(D))$ Entonces necesitamos probar que

$$c_{\lambda,l}(D) \underset{l\to\infty}{\longrightarrow} \frac{\sum_{\{u,v\}\in E_n} |f(u)-f(v)|}{n\cdot 2^{n-1}}.$$

$X_0, X_1, X_2, \ldots$ es un camino aleatorio por el hipercubo y su distribución estacionaria $\pi$ es la uniforme: $\pi_x = 1/2^n$, para todo $x \in V_n$. Recordemos que la inestabilidad de $f$ cuenta el número de veces que la función $f$ cambia de decisión a lo largo del camino aleatorio. En el artículo [4] los autores se basan en el teorema ergódico para decir que, la fracción de veces que el camino aleatorio cruza aristas en las que hay cambio de decisión entre la longitud del camino, tiende al número de estas aristas dividido entre $\mid E_n \mid \; = n2^{n-1}$.

Formalmente consideramos la cadena de Markov $(X_0, X_1), (X_1, X_2), \ldots$ en la cual cada estado es un arco $e = (X_i, X_{i+1}) \in \vec{E_n}$ de parejas ordenadas de vértices vecinos del hipercubo. Por lo tanto, la función definida sobre el conjunto de estados (arcos) es $f(X_k, X_{k+1}) \; = \; \mid f(X_{k+1}) - f(X_k) \mid$. Hasta aquí, los autores de [4] indican que es necesario verificar que la cadena es irreducible y que su única distribución invariante es la uniforme, con lo cual concluimos que:

$$c_{\lambda,l}(D) \; = \; \tfrac{1}{l} \sum_{k=0}^{l-1} f(X_k, X_{k+1}) \; \underset{l\to\infty}{\longrightarrow} \; \sum_{e\in\vec{E_n}} \pi_e f(e) \; = \; \tfrac{1}{n2^{n-1}} \sum_{e\in\vec{E_n}} f(e)$$

□

**Teorema 2** *La inestabilidad promedio del sistema $D_0$ está dada por:*

$$c(D_0) = \frac{\binom{n-1}{t}}{2^{n-1}}$$

**Prueba:** es necesario contar el número de aristas $(x, y)$ con $x \in N^t(0^n)$ y $y \in V_n - N^t(0^n)$.

Este número es $(n-t)\cdot\binom{n}{t}$ porque hay $\binom{n}{t}$ de tales vértices $x$ y cada uno tiene $(n-t)$ aristas adyacentes a vértices $y$.

Posteriormente, basándonos en la proposición 1, de que $c(D_0) = \frac{(n-t)\binom{n}{t}}{n\cdot 2^{n-1}}$ tenemos

$\frac{(n-t)}{n\cdot 2^{n-1}} \cdot \frac{n!}{t!(n-t)!} = \frac{(n-t)}{n\cdot 2^{n-1}} \cdot \frac{n(n-1)!}{t!(n-t)(n-t-1)!} = \frac{1}{2^{n-1}} \cdot \frac{(n-1)!}{t!(n-t-1)!} = \frac{\binom{n-1}{t}}{2^{n-1}}$

□

**Proposición 2** *Sea $D = (n, t, f)$ un sistema simétrico sin memoria. Entonces,*

$$c(D) = \sum_{i=0}^{n-1} \binom{n-1}{i} \cdot \frac{\mid f(i+1) - f(i) \mid}{2^{n-1}}$$

**Prueba:** basándonos en el artículo [4] sabemos que:

$c_{\lambda,l}(D) \longrightarrow \sum_{i=0}^{n-1} \pi_{(i,i+1)} \mid f(i+1) - f(i) \mid + \sum_{i=0}^{n-1} \pi_{(i+1,i)} \mid f(i+1) - f(i) \mid$

Se sigue del hecho de que $\pi_{(i,i+1)} = \pi_{(i+1,i)} = \frac{\binom{n-1}{i}}{2^n}$

□

## 3.2. Representación e inestabilidad del sistema $D_1$

Ahora veremos un sistema con un bit de memoria llamado $D_1$:

**Definición 9** *Sea el sistema $D_1 = (n, t, S_1, f_1, g_1)$ donde $S_1 = (s_0, 0, 1)$ y*

$$f_1(x, s) = g_1(x, s) = \begin{cases} 0 & \textit{si } \#1(x) \leq t \\ 1 & \textit{si } \#0(x) \leq t \\ maj(x) & \textit{si } s = s_0 \\ s & \textit{en otro caso} \end{cases}$$

Aquí, $maj(x)$ regresa el bit más común en $x \in \{0,1\}^n$, o bien 1 en caso de empate.

**Teorema 3** *La inestabilidad promedio del sistema $D_1$ está dada por:*

$$c(D_1) = \left(2^{n-1} \sum_{k=t}^{n-t-1} \frac{1}{\binom{n-1}{k}}\right)^{-1}$$

**Prueba:** ya que $f_1$ es simétrica, podemos hacer una proyección del sistema a estados de la forma $(i, s) \in (\{0, \ldots, t-1\} \times \{0\} \cup \{t, \ldots, n-t\} \times \{0, 1\} \cup \{n-t+1, \ldots, n\} \times \{1\})$,

donde $i$ denota el número de 1's que los procesos están leyendo y $s$ representa la última salida de $f_1$ (no se considera al caso $s = s_0$ porque este aparece sólo una vez al inicio). Las transiciones son las siguientes:

$$\mathbb{P}\{(i,s) \to (i+1,s')\} = \tfrac{n-i}{n} \qquad \mathbb{P}\{(i,s) \to (i-1,s')\} = \tfrac{i}{n}$$

Dando una cadena de Markov irreducible que tiene una distribución estacionaria $\pi$. Sabemos que $c_{\lambda,l}(D_1)$ corresponde a la fracción de veces que la cadena ha estado en estado $(t,1)$ o $(n-t,0)$ al tiempo $l$. En efecto, el sistema cambia de decisión cada vez que lee $t$ 1's mientras su ultima decisión haya sido 1; o bien, cuando lee $t$ 0's mientras su última decisión haya sido 0.

El artículo [4] explica que $c_{\lambda,l}(D_1)$ converge casi seguramente a $(\pi_{t,1} + \pi_{n-t,0})$. Por simetría, se sigue que $c(D_1) = 2\pi_{t,1}$.

Denotamos el tiempo esperado necesario para pasar de un estado $(i,a)$ hacia un estado $(j,b)$ de la siguiente forma: $\mathbb{E}_{(i,a)}(T_{j,b})$.

Ya que el inverso del tiempo de retorno esperado corresponde a la distribución estacionaria, tenemos que $c(D_1) = \frac{2}{\mathbb{E}_{(t,1)}(T_{t,1})}$. Puesto que cada ciclo que pasa por $(t,1)$ también pasa por $(n-t,0)$, tenemos:

$$\mathbb{E}_{(t,1)}(T_{t,1}) \;=\; \mathbb{E}_{(t,1)}(T_{n-t,0}) + \mathbb{E}_{(n-t,0)}(T_{t,1})$$

Por simetría tenemos que $\mathbb{E}_{(t,1)}(T_{n-t,0}) = \mathbb{E}_{(n-t,0)}(T_{t,1})$. Por otro lado para cada $i \neq t$ tenemos que $\mathbb{E}_{(t,1)}(T_{i,0}) = \mathbb{E}_{(t,0)}(T_{i,0})$. Así obtenemos:

$$c(D_1) = \tfrac{2}{2\mathbb{E}_{(t,0)}(T_{n-t,0})} = \tfrac{1}{\mathbb{E}_{(t,0)}(T_{n-t,0})}$$

El tiempo esperado para alcanzar el estado $(n-t,0)$ empezando de $(t,0)$ es igual al tiempo esperado de alcanzar el estado $n-t$ comenzando del estado $t$ en el modelo clásico de Ehrenfest (como referencia puede leer [30]). En efecto, en esta parte del camino el sistema siempre estará en estado 0 y sólo cambiará a 1 cuando alcance la posición $n-t$.

En [4] los autores muestran que el tiempo esperado de ir de un estado $k$ a un estado consecutivo $k+1$ en el modelo clásico de Ehrenfest es igual a $\frac{1}{\binom{n-1}{k}} \sum_{j=0}^{k} \binom{n}{j}$. Así

$$\frac{1}{c(D_1)} \;=\; \sum_{k=t}^{n-t-1} \frac{1}{\binom{n-1}{k}} \sum_{j=0}^{k} \binom{n}{j}$$

Indicando que $k' = n-1-k$ (para poder hacer un cambio de límites) y usando la propiedad de que $\binom{x}{y} = \binom{x}{x-y}$ obtenemos:

$$\frac{1}{c(D_1)} \;=\; \sum_{k'=t}^{n-t-1} \frac{1}{\binom{n-1}{n-1-k'}} \sum_{j=0}^{n-1-k'} \binom{n}{j} \;=\; \sum_{k'=t}^{n-t-1} \frac{1}{\binom{n-1}{k'}} \sum_{j=k'+1}^{n} \binom{n}{j}$$

Sumando ambas expresiones de $\frac{1}{c(D_1)}$ y utilizando la propiedad de que $\sum_{i=0}^{w} \binom{w}{i} = 2^w$ tenemos:

$$\frac{2}{c(D_1)} \;=\; \sum_{k=t}^{n-t-1} \frac{2^n}{\binom{n-1}{k}}$$

□

## 3.3. Proyección del camino aleatorio del hipercubo en una recta

Al determinar aleatoriamente primero una configuración inicial y posteriormente una ejecución del sistema, se va obteniendo una secuencia de vértices los cuales forman un *camino aleatorio* en el hipercubo. El *camino aleatorio* se vuelve particularmente interesante cuando la función de decisión $f$ es simétrica, i.e. cuando depende sólo de la cantidad de $0's$ ó $1's$ en el vector de entrada. Recordemos que la función $f$ da como salida el valor $d$ basado en los vértices previos del camino.

Recordando el ejemplo del hipercubo asociado a un sistema con $4$ procesos (figura 3.1, página 16) podemos realizar una función proyección $F_p$ cuyo dominio son los vértices del hipercubo $V_H$ y cuya imagen son los puntos enteros $i \in \{0, \ldots, n\}$ en una semirecta $R$.

Haremos una partición en el conjunto $V_H$ de vértices del hipercubo, de acuerdo al número de elementos de un valor $b \in \{0, 1\}$ presentes en el vector de entrada asociado a cada vértice. Entonces la función proyección $F_p$ asociará a cada elemento $x$ de $V_H$ con un punto $i$ en la semirecta $R$, de la siguiente manera:

$$F_p : V_H \to R$$

$$F_p(x) = \{i \mid \#b(x) = i\}$$

Asignamos a $b$ el valor de $1$; por lo tanto seleccionaremos los vértices del hipercubo dependiendo del contenido de $1's$ dentro de su vector.

Así pues, tendremos entonces una representación del sistema de nuestro ejemplo de la figura 3.1 de la siguiente manera:

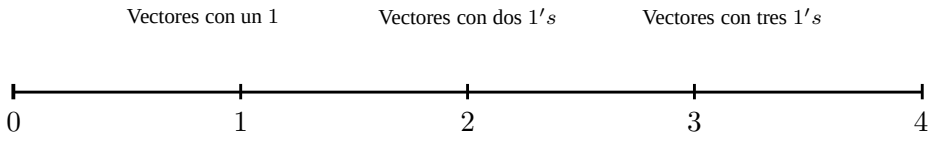


Figura 3.2: Proyección en una recta del hipercubo asociado a $4$ procesos.

De esta forma, tenemos que asumir que $f$ está definida sobre este conjunto de vértices en la semirecta, en lugar de los vértices del hipercubo; diremos entonces que $f(i) = d$ si y sólo si $f(x) = d$ cuando $\#1(x) = i$.

Si aplicamos la función proyección $F_p$ al *camino aleatorio* por el hipercubo sobre un *camino aleatorio* en la semirecta, obtenemos una cadena de Markov (vea [6]) con transiciones:

$$\mathbb{P}\{i \to (i+1)\} = \tfrac{n-i}{n} \quad ; \quad \mathbb{P}\{i \to (i-1)\} = \tfrac{i}{n} \tag{3.1}$$

La probabilidad $\mathbb{P}$ de moverse de $i$ a $i+1$ es simplemente la probabilidad de moverse desde un vértice con $i$ entradas iguales a $1$ hacia un vértice con $i+1$ entradas iguales a $1$. Existen $n-i$ de tales aristas que pueden hacer que esto suceda.

Este proceso es conocido como el modelo de urnas de Ehrenfest (como referencia puede leer [30]), el cual es utilizado por los físicos teóricos para modelos de difusión de moléculas entre dos contenedores; y puede ser descrito como sigue: dos urnas, $U_1$ y $U_2$ contienen juntas $m$ pelotas. En cada instante de tiempo $t = 1, 2, \ldots$, una de las $m$ pelotas es tomada al azar y desplazada de la urna de la cual fue tomada y depositada

en la otra urna. Sea $E_k$ el estado en donde $k$ pelotas se encuentran digamos, sin pérdida de generalidad, en la urna $U_2$, para $k = 1, 2, \ldots, m$. Así pues se estudia la distribución de el número de pelotas que hay en cada urna después de que ha transcurrido un largo periodo de tiempo.

Fácilmente podemos identificar nuestra proyección con el modelo de urnas de Ehrenfest, donde el número de $1's$ presentes en cada vértice del hipercubo (número asociado en la semirecta) representaría la cantidad de pelotas que hay en una de las urnas. Y este a su vez como una cadena de Markov, donde la transición entre estados consecutivos, digamos de $E_i$ a $E_{i+1}$, está dada solamente por la cantidad $i$ de pelotas presentes en nuestra urna $U_2$ en ese momento.

## 3.4. Caminos aleatorios por el hipercubo vistos como cadenas de Markov

Una de las generalizaciones más importantes del concepto de sucesión de magnitudes aleatorias independientes, es la noción de sucesión en cadenas de Markov [6]. Ya que esta es una serie de eventos en la que, la probabilidad asociada a un evento en particular depende del evento inmediato anterior, induce el hecho de que los eventos mantengan memoria sobre el último evento y por ello condicione las posibilidades de los eventos futuros (vea el apéndice A.5).

Uno de los puntos centrales en el manejo de los caminos aleatorios en el hipercubo, es poderlos asociar a caminos aleatorios sobre la semirecta, y de ahí manejarlos como procesos estocásticos discretos que cumplan con la propiedad de Markov.

Así pues, por todo lo comentado hasta el momento, los caminos aleatorios por el hipercubo (y por ende en la semirecta) están asociados con los estados $E_k$ del modelo de urnas de Ehrenfest. La probabilidad con la cual se tendrá el siguiente estado depende únicamente del estado del sistema en ese momento; lo cual nos conduce a la idea del modelo de cadenas de Markov.

En el capítulo 4 se explicará de qué manera las cadenas de Markov, las cuales representan los caminos aleatorios por el hipercubo, nos servirán para realizar las simulaciones computacionales de los modelos.

Un camino aleatorio por el hipercubo (y por ende en la proyección sobre la semirecta) se vuelve particularmente interesante cuando la función $f$ es simétrica, i.e. cuando depende únicamente de la cantidad de $0$'s y $1$'s presentes en el vector de entrada. Esta función es del siguiente tipo: "si $i$ sensores están leyendo el valor $1$ entonces la salida es $d$". Es por esa razón que podemos hacer la proyección del hipercubo a un camino unidimensional cuyos vértices son $\{0, \ldots, n\}$. Podemos por lo tanto asumir que $f$ está definida sobre este conjunto de vértices en vez de sobre el conjunto de vértices en el hipercubo. En lugar de considerar el estado $x$ (vértice $x$) consideraremos el estado $i = \#1(x)$. Así es como obtenemos lo que hemos dicho más arriba: una cadena de Markov con transiciones

$$\mathbb{P}\{i \to (i+1)\} = \tfrac{n-i}{n} \qquad \mathbb{P}\{i \to (i-1)\} = \tfrac{i}{n}$$

Y este proceso es conocido como el modelo de urnas de Ehrenfest. La distribución invariante del modelo de Ehrenfest (fácilmente calculable por proyección de la distribución uniforme del hipercubo) se sabe que es $\pi_i = \frac{\binom{n}{i}}{2^n}$. Por lo tanto, la distribución invariante de la pareja de estados correspondientes a los arcos $(i, i+1)$ y $(i+1, i)$ son $\pi_{(i,i+1)} = \frac{\binom{n}{i}}{2^n} \cdot \frac{n-1}{n}$, y $\pi_{(i+1,i)} = \frac{\binom{n}{i+1}}{2^n} \cdot \frac{i+1}{n}$. Así, $\pi_{(i,i+1)} = \pi_{(i+1,i)} = \frac{\binom{n-1}{i}}{2^n}$.

# Capítulo 4

# Simulaciones de los sistemas $D_0$ y $D_1$

Una de las principales contribuciones de este trabajo es la realización de experimentos acerca del comportamiento e inestabilidad que van presentando los sistemas $D_0$ y $D_1$ mediante simulaciones computacionales.

Así pues, presentamos a continuación una reseña de las pruebas realizadas. El capítulo se encuentra dividido es dos partes: la primera trata sobre las simulaciones hechas para el modelo sin memoria y la segunda parte el modelo con memoria.

En ambos casos comenzamos con una explicación de cómo se pensó el respectivo modelo y cuáles eran los parámetros de entrada para cada uno de nuestros programas. A continuación de esto, mostramos las gráficas de los resultados obtenidos dando una explicación intuitiva de su interpretación.

Cabe señalar que estas simulaciones fueron realizadas en lenguaje MATLAB, en una computadora PC bajo el ambiente *Windows XP* con arquitectura *Intel* Centrino Duo y también en una Apple Macintosh con sistema operativo *Mac Os X* y arquitectura *Intel* Xeon de doble núcleo. Se escogieron estas características primero, porque el lenguaje MATLAB ofrece gran versatilidad en el manejo de una amplia gama de funciones matemáticas que optimizan el desempeño de las simulaciones computacionales; y segundo, porque los sistemas operativos para los cuales MATLAB estaba diseñado, eran *Windows XP* y *Mac Os X* con sus respectivas arquitecturas.

Basta decir que las características físicas de velocidad y memoria RAM, fueron también de las más adecuadas al momento de realizar los experimentos: 1.66 GHz de velocidad con 1 Gb de espacio en memoria para el caso de la PC; y 2.66 GHz de velocidad con 2 Gb de espacio en memoria para el caso de la Apple Macintosh. Aun así y como comentario al margen, muchas de estas simulaciones requieren de algunas horas de tiempo de procesador para llevar a cabo su tarea.

Los programas se realizaron siguiendo las bases de la programación orientada a procedimientos (vea el apéndice B.2) y se aplicaron fases del Proceso Personal de Desarrollo de Software [1] (PSP) a fin de obtener estimados de calidad y tiempo en la realización de los programas. Por ello mismo, se obtuvieron programas lo suficientemente flexibles para adecuarlos a distintos experimentos en los cuales se buscaba obtener cierta información modificando la perspectiva de la simulación.

Con lo dicho hasta aquí, ahora pasemos a ver la explicación de los modelos de simulación para los programas y las gráficas resultantes en cada experimento. En dichas gráficas, se puede observar comportamientos y tendencias que son predichas por las ecuaciones analíticas de los artículos en los que está basado este trabajo. De igual manera (y ese es uno de nuestros objetivos) podemos referir aquellos comportamientos no previstos y que puedan aportar información acerca de nuestros modelos.

[1] Las métricas y bases del Proceso Personal de Desarrollo de Software (PSP) se encuentran fuera del alcance y objetivos de la presente tesis.

## 4.1. Simulaciones del modelo $D_0$ (sin memoria)

Recordemos la proyección del *camino aleatorio* por el hipercubo sobre un *camino aleatorio* en una semirecta (sección 3.3, página 21). En esa proyección representamos los vértices de 1 a $n$ donde además cada posición $i$ tiene asociada una probabilidad de desplazamiento (tal y como lo indica la ecuación 3.1): $\frac{i}{n}$ si su siguiente movimiento será hacia la izquierda, y $\frac{n-i}{n}$ si su siguiente movimiento será hacia la derecha.

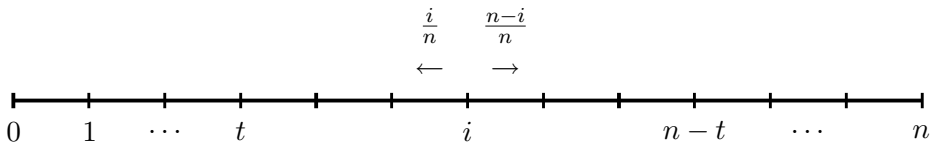


Figura 4.1: Proyección del hipercubo para el modelado de un sistema sin memoria.

Teniendo en mente esta idea de los movimientos con probabilidad asociada a su posición, realizamos un programa en lenguaje MATLAB que simule el comportamiento de nuestro modelo. Damos como entrada el número $n$ de procesos participantes en el sistema, el parámetro de tolerancia a fallos $t$, y la longitud $l$ del camino aleatorio, que no es otra cosa sino el número de tránsitos a realizar sobre los vértices de la semirecta. Inicialmente el programa elegirá un punto al azar sobre la recta y se desplazará hacia uno de sus vecinos (izquierda o derecha) de acuerdo a la probabilidad asociada descrita en la ecuación 3.1. Posteriormente repetirá el mismo procedimiento hasta completar el número $l$ de tránsitos establecidos.

Podemos notar fácilmente que si la posición $i$ actual se encuentra más a los extremos, entonces tenderá a desplazarse hacia las posiciones centrales. La razón de que esto suceda se debe precisamente a que mientras más cerca esté de los vértices extremos, más pequeña es la probabilidad de continuar su camino en esa dirección.

Este modelo está diseñado para el sistema $D_0$ de la definición 6. Por lo tanto, del teorema 2 (página 19) sabemos que su inestabilidad está dada por:

$$c(D_0) = \binom{n}{t} \cdot \frac{n-t}{n} \cdot \frac{1}{2^{n-1}} \tag{4.1}$$

(como referencia vea [4]).

En particular, para observar los cambios de decisión del sistema, nos interesa los tránsitos realizados por el vértice $t$. Recordemos que de acuerdo a la función de decisión del sistema $D_0$, los cambios de decisión (la inestabilidad) se realizarán cuando haya al menos $t$ elementos de valor 1 en el vector de entrada.

Nuestro programa, el cual realiza la simulación del modelo, recibe como entradas numéricas a tres parámetros: 1) El número $n$ de procesos participantes en nuestro sistema; 2) Un parámetro particular $t_1$ de tolerancias a fallos (el cual utilizaremos más adelante en las gráficas de convergencia); y 3) La longitud $l$ de un camino aleatorio por recorrer.

Así pues, a continuación mostramos la gráfica de la ecuación 4.1 dando a $n$ como parámetro de entrada y teniendo al parámetro de tolerancia a fallos $t$ como función de $n$. (Notemos que para realizar esta gráfica (figura 4.2), no utilizamos ni a $t_1$ ni a $l$).

En el eje $x$ tenemos los valores que va tomando $t$ hasta llegar a $n$ (la cual es fija).

En el eje $y$ los valores de la función $c(D_0) = \frac{\binom{n-1}{t}}{2^{n-1}}$.

Para esta gáfica en particular (figura 4.2) se utilizó como entrada una $n = 100$, la cual fue elegida como una constante indistinta; y se fue variando al parámetro $t$ para que tomara valores entre $0$ y $n$, con el fin de graficar $c(D_0)$.

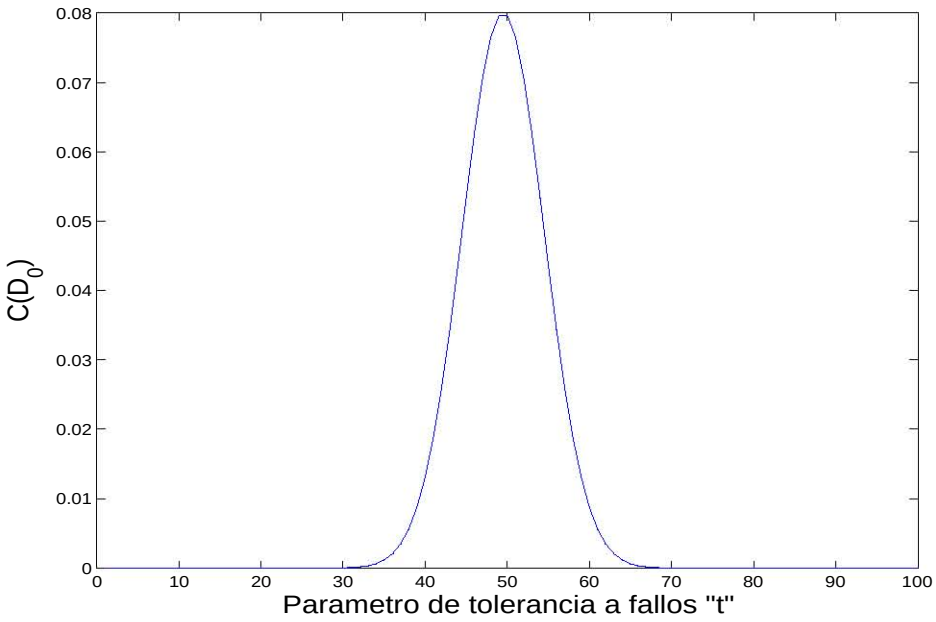


Figura 4.2: Inestabilidad mostrada por el sistema $D_0$, conforme $t$ toma valores entre $1$ y $n$.

Podemos apreciar que la inestabilidad de $D_0$ aumenta conforme $t$ se aproxima a $n/2$.

Un punto a considerar es ¿Porqué el parámetro $n$ lo elegimos como una constante cualquiera? En realidad es que dando el parámetro $n$ como fijo, lo que nos interesa son los valores que $t$ va tomando entre $0$ y $n$ a fin de obtener los valores de inestabilidad en ese intervalo.

Para conocer algunos detalles y descripciones de los programas de simulación, vea el apéndice B.2.

Al ejecutar el programa de simulación, también obtuvimos la distribución que iba tomando el número de tránsitos realizados por cada vértice de la semirecta.

Recordemos que el parámetro $n$ fue elegido como una constante cualquiera y la siguiente gráfica (figura 4.3), utiliza ese parámetro para evaluar la distribución de pasos por cada vértice. Para este experimento, ya fue necesario hacer uso de la longitud $l$ del camino aleatorio, para lo cual es preferible utilizar un valor lo suficientemente grande.

Por lo tanto, en el eje $x$ tenemos el parámetro $n$ (los vértices obtenidos de la proyección).

En el eje $y$, tenemos el número de tránsitos realizados en cada punto, dividido entre el total de pasos realizados.

Para esta gáfica en particular (figura 4.3) utilizamos como parámetros de entrada a nuestro programa la misma $n = 100$ que en el caso anterior y una longitud del camino $l = 50000$.

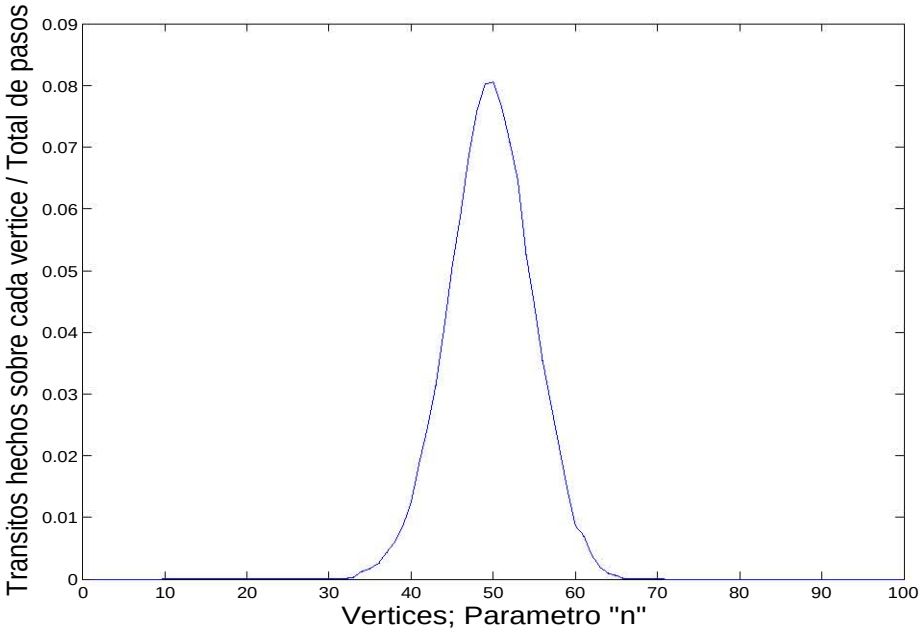


Figura 4.3: Distribución de pasos realizados por cada vértice, en la simulación con $n = 100$ y $l = 50000$.

Fijemos un punto, digamos un vértice $q$. Ahora observemos de la gráfica 4.3 la razón de la cantidad de veces que el recorrido pasó por ese vértice entre el total de pasos. Y como comentamos más arriba, de acuerdo con las probabilidades asociadas a cada vértice durante el recorrido, si la posición de $q$ se encuentra más a los extremos, entonces tenderá a desplazarse hacia las posiciones centrales ya que mientras más cerca esté de los vértices extremos, más pequeña es la probabilidad de continuar su camino en esa dirección.

Por otro lado, sabemos que la función de decisión $f_0$ indica los cambios de decisión de acuerdo al número de $1's$ que haya en un vector de entrada (en este caso $\#1(x) = q$). Por lo tanto, para llegar a ese vértice $q$ a lo largo de un camino aleatorio en el hipercubo, es necesario que sea un vector cuya cantidad de $1's$ sea un número cercano a $n/2$ (que es cuando hay más vértices posibles por dónde transitar).

Es por ello que observamos una gran similitud en las figuras 4.2 y 4.3.

Uno de los objetivos planteados al realizar las simulaciones es el de mostrar la convergencia predicha por las ecuaciones; sin embargo no es la única finalidad, ya que podemos también obtener información sobre el comportamiento que va teniendo la inestabilidad a lo largo de las ejecuciones y además saber qué tan rápido tiende al valor anunciado (dicho conocimiento no nos lo dice la ecuación).

Ahora bien, para esta simulación ya fue necesario hacer uso del parámetro de entrada $t_1$. Este valor es recomendable que sea cercano a la mitad de $n$ por la razón que nos enseña la figura 4.2: que la inestabilidad del sistema es mayor en esos valores. En otras palabras, nos interesa monitorear un vértice $t_1$ el cual muestre una alta inestabilidad o bien una alta frecuencia en el número de tránsitos realizados por él a lo largo de la simulación.

Por lo tanto, teniendo en cuenta lo anterior, la figura 4.4 nos muestra una ejecución con parámetros de entrada $n = 100$, $t_1 = 50$ y $l = 50000$ iteraciones.

En el eje $x$ de la gráfica 4.4 mostramos el número total de tránsitos realizados.

En el eje $y$ ponemos simplemente, el número de veces que el programa de simulación ha pasado por el vértice $t = t_1$, dividido entre el total de tránsitos realizados en el experimento.

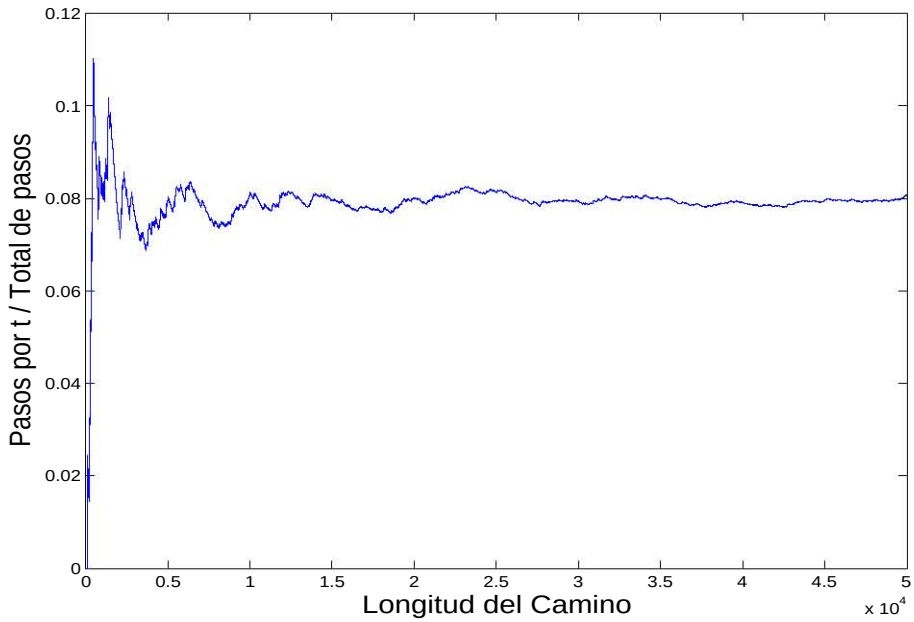


Figura 4.4: Convergencia mostrada por la simulación en los tránsitos hechos por $t$.

Sin embargo esta gráfica muestra sólo una ejecución en particular, por lo tanto no podemos concluir algún comportamiento del modelo; ya que una única ejecución está expuesta a los posibles efectos que el azar va teniendo sobre ella. Por eso mismo, la información que nos puede aportar no es representativa del comportamiento global del sistema.

Ahora, es necesario observar la forma que va teniendo la curva de la figura 4.4 después de realizar varias ejecuciones del programa de simulación y de esta forma tener información más acertada sobre el comportamiento del sistema.

La figura 4.5 al igual que la anterior, muestra el número total de tránsitos realizados (eje $x$) vs. el número de veces que el programa de simulación ha pasado por el vértice $t_1$ y dividido eso entre el total de tránsitos realizados en el experimento (eje $y$). La principal diferencia entre ambas gráficas, radica precisamente en que se realizaron $100$ ejecuciones y se promediaron sus resultados obtenidos.

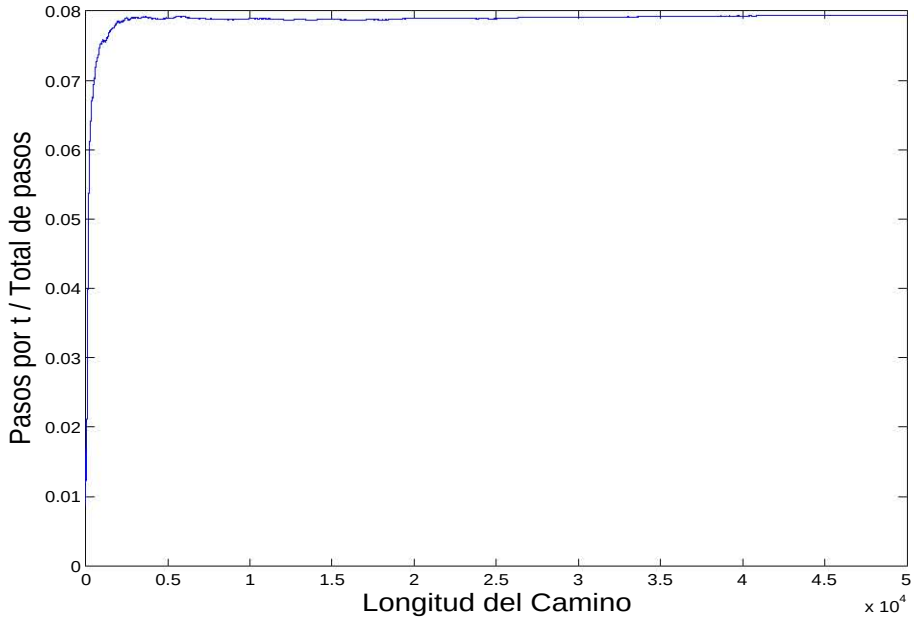


Figura 4.5: Convergencia mostrada por la simulación a lo largo de varias ejecuciones.

Notemos que la curva se ha suavizado y ha mostrado una convergencia más marcada hacia el valor esperado. En el capítulo 5 realizaremos un estudio de la velocidad con la que convergen hacia dicho valor.

Sin embargo, de nuevo esta simulación representa un exclusivo valor de $t_1$; por lo tanto resultaría interesante observar el comportamiento del sistema cuando variamos los valores del parámetro de tolerancia a fallos.

Para ello, pongamos un tercer eje y obtengamos una figura en tercera dimensión que nos muestre diferentes valores de los tres parámetros al mismo tiempo: figura 4.6.

La figura 4.6 nos muestra diversas ejecuciones de la simulación con la variación del parámetro $t$.

El eje derecho y el eje vertical son similares a los ejes $x$ y $y$ respectivamente de la figura 4.5.

Por el otro lado, el eje izquierdo y el eje vertical son los respectivos ejes $x$ y $y$ de la figura 4.3.

Nuevamente el parámetro de entrada $n$ fue fijado arbitrariamente en $100$, pues nos interesa observar el comportamiento del sistema cuando $t$ toma valores entre $0$ y $n$.

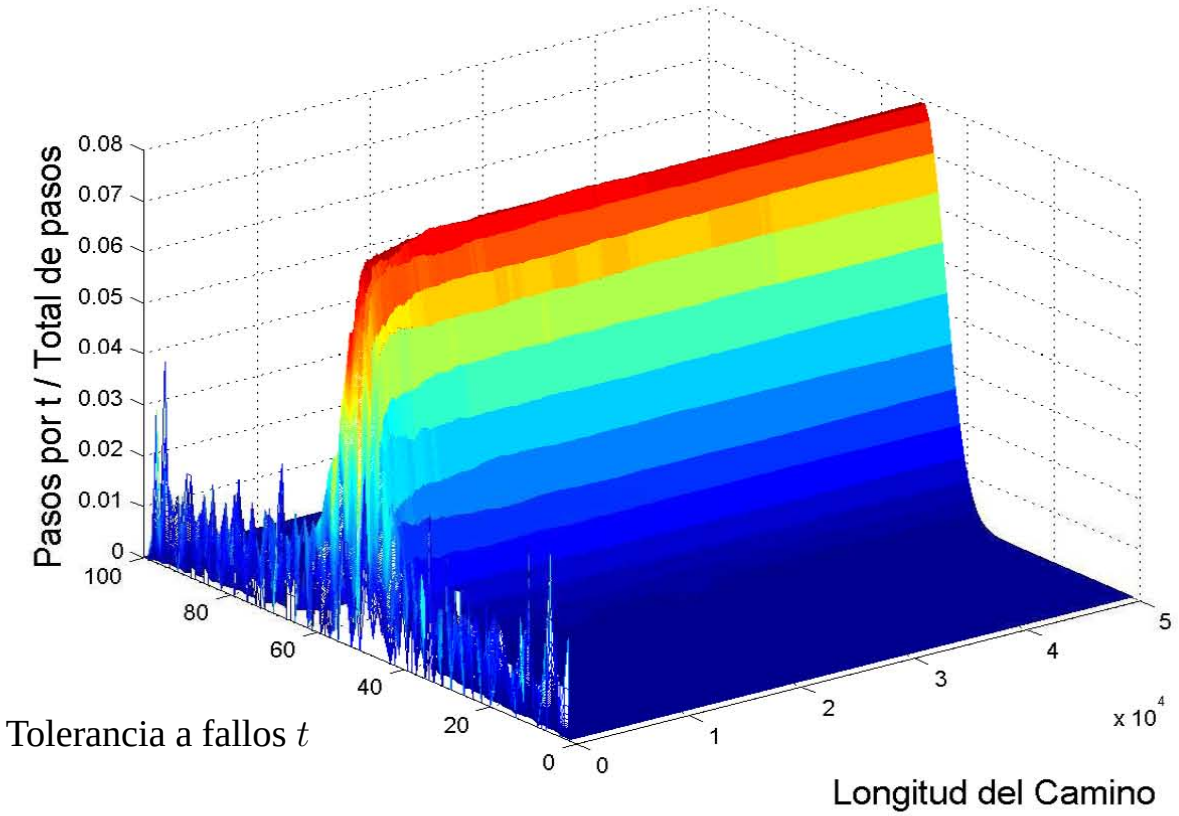


Figura 4.6: Simulación en 3D variando el parámetro $t$.

Observamos un comportamiento interesante en la inestabilidad mostrada durante los inicios de cada recorrido. Todos muestran una gran variación temprana independiente del valor que tome $t$. Podemos atribuir este comportamiento a que la elección del vértice incial está asociada a una distribución uniforme y por lo tanto es igualmente probable iniciar un recorrido en cualquier vértice.

Tal y como pensamos ab initio, la elección de este vértice inicial, no influye en el comportamiento del sistema cuando la longitud de los recorridos tiende al infinito.

También podemos notar que las respectivas convergencias se alcanzan casi al mismo tiempo en todos los casos. Y comparando esto con la gráfica 4.5 podemos asegurar que no hay que esperar demasiado para alcanzar dicha convergencia. Sin embargo, en el capítulo 5 realizaremos un estudio de la velocidad en la convergencia que muestran estos experimentos.

## 4.2. Simulaciones del modelo $D_1$ (con memoria)

Nuevamente, utilizaremos la proyección del *camino aleatorio* por el hipercubo sobre un *camino aleatorio* en una semirecta (sección 3.3, página 21). Hacemos una partición en el conjunto $V_H$ de vértices del hipercubo, de acuerdo al número de $1's$ presentes en el vector de entrada asociado a cada vértice. Entonces la función proyección $F_p$ asociará a cada elemento $x$ de $V_H$ con un punto $i$ en la semirecta $R$, de la siguiente manera:

$$F_p : V_H \to R$$

$$F_p(x) = \{i \mid \#b(x) = i\}$$

Recordemos que en esa proyección representamos los vértices de $1$ a $n$ donde además cada posición $i$ tiene asociada una probabilidad de desplazamiento (tal y como lo indica la ecuación 3.1): $\frac{i}{n}$ si su siguiente movimiento será hacia la izquierda, y $\frac{n-i}{n}$ si su siguiente movimiento será hacia la derecha. Pero para las simulaciones del caso con memoria se pensó en un modelo representado por dos semirectas; donde cada una significa el estado de la decisión en ese momento. En particular, la semirecta de arriba representa el estado $0$ y la de abajo el estado $1$. De igual manera existe una probabilidad de desplazamiento asociada a cada posición, similar al caso sin memoria. La principal diferencia radica en que cuando la posición llega a $n-t$ con valor de decisión $0$ entonces cambia su valor de decisión a $1$. De manera similar, si llega al punto $t$ con valor de decisión $1$ entonces cambia su decisión a $0$ (Tal como lo muestra el siguiente esquema).

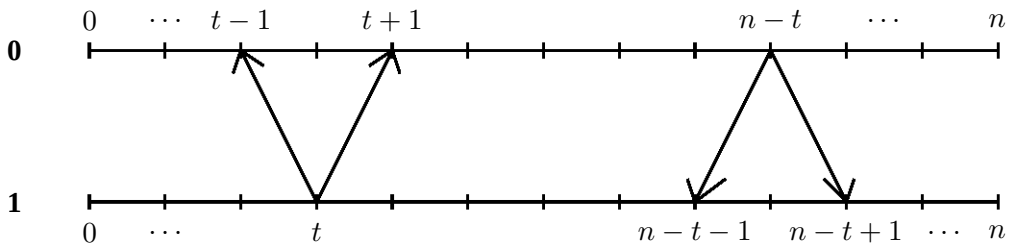


Figura 4.7: Modelado de un sistema con memoria.

Notemos que ambas rectas representan los mismos vértices, sólo que en estado diferente.

Este modelo está diseñado para el sistema $D_1$ de la definición 9. Por lo tanto, del teorema 3 (página 19) sabemos que su inestabilidad está dada por:

$$c(D_1) = \binom{n-1}{t} \cdot \frac{\sum_{k=t}^{n-t-1} \frac{\sum_{j=0}^{k} \binom{n}{j}}{\binom{n-1}{k}}}{2^{n-1}} \tag{4.2}$$

En este modelo de sistema con memoria representado en la figura 4.7 también queremos observar los cambios de decisión (la inestabilidad) que va teniendo a lo largo del recorrido en su camino aleatorio. De acuerdo con la función $f$ del sistema $D_1$, los cambios de decisión se realizarán cuando haya al menos $t$ elementos de valor $1$ en el vector de entrada y además el valor decidido hasta el momento sea $0$; o bien, que haya al menos $n-t$ elementos de valor $1$ en el vector de entrada y además el valor decidido hasta el momento sea $1$.

Con la idea de los movimientos con probabilidad asociada a su posición y su estado, realizamos otro programa en lenguaje MATLAB que simule el comportamiento de nuestro modelo con memoria. Damos como entrada el número $n$ de procesos participantes en el sistema, el parámetro de tolerancia a fallos $t$, y la longitud $l$ del camino aleatorio, que no es otra cosa sino el número de tránsitos a realizar sobre la línea correspondiente a su estado.

El programa elegirá un punto al azar sobre la línea y se desplazará hacia uno de sus vecinos (izquierda o derecha) de acuerdo a la probabilidad asociada descrita en la ecuación 3.1. Posteriormente repetirá el mismo procedimiento hasta completar el número $l$ de tránsitos impuestos.

El programa de simulación tendrá de igual forma como parámetros de entrada tres valores numéricos: 1) El número $n$ de procesos participantes en nuestro sistema; 2) Un parámetro particular $t_1$ de tolerancias a fallos (el cual utilizaremos más adelante en las gráficas de convergencia); y 3) La longitud $l$ de un camino aleatorio por recorrer.

La primera gráfica que presentamos en las simulaciones para este modelo (figura 4.8) nos muestra la distribución en el número de tránsitos realizados por cada vértice. Este experimento requirió del uso únicamente de los parámetros $n$ y $l$.

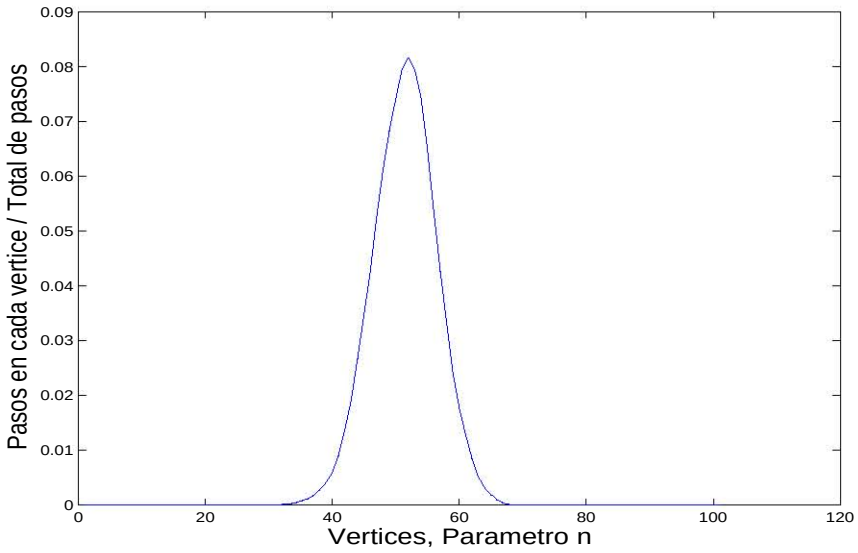


Figura 4.8: Distribución de pasos realizados por cada vértice, en el modelo con memoria.

Nuevamente explicamos que el valor de $n$ es una constante entera cualquiera y $l$ lo seleccionamos como un valor lo suficientemente grande.

Así pues, en el eje $x$ de esta gráfica tenemos los vértices obtenidos de la proyección (parámetro $n$). En el eje $y$ tenemos el número de tránsitos realizados por cada vértice, dividido entre el total de pasos realizados, a fin de normalizar los valores. Para la gráfica 4.8 en particular se utilizaron como parámetros de entrada al programa una $n = 100$ y una $l = 50000$.

Posteriormente en el capítulo 5 realizaremos una comparación de este modelo con distintos valores del parámetro $n$ así como un estudio de la velocidad en la convergencia que muestran estos experimentos.

Las siguientes dos gráficas (figura 4.9 y figura 4.10) muestran la convergencia predicha por la ecuación de inestabilidad para el modelo con memoria. Recordemos que debemos sacar información de ellas sobre el comportamiento del sistema a lo largo de las ejecuciones, así como qué tan rápido convergen.

Para esta simulación hicimos uso del parámetro de entrada $t_1$. Si observamos la figura 4.8 podemos apreciar que la mayor concentración de pasos está en los vértices próximos a $n/2$, por lo tanto sería recomendable el uso de uno de esos vértices como parámetro de tolerancia a fallos, ya que mostraría un alto número de cambios de decisión a lo largo de su ejecución.

Teniendo esto en cuenta, para el experimento mostrado en las gráficas 4.9, 4.10 y 4.11 se utilizaron como entradas: $n = 100$, $t_1 = 50$ y $l = 50000$.

En el eje $x$ de estas gráficas mostramos el total de tránsitos realizados. En el eje $y$, el número de veces que el programa de simulación ha pasado por el vértice $t_1$ en estado $1$ (o el vértice $n - t_1$ en estado $0$) dividido entre el total de tránsitos realizados por el experimento.

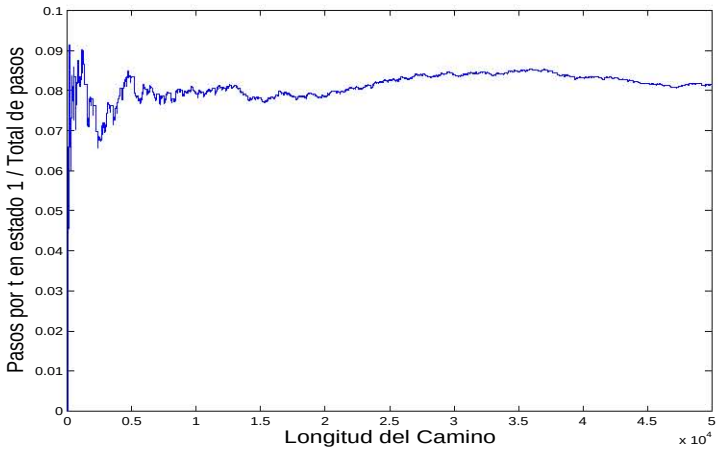


Figura 4.9: Convergencia mostrada por la simulación en los tránsitos hechos por $t$ en estado $1$.

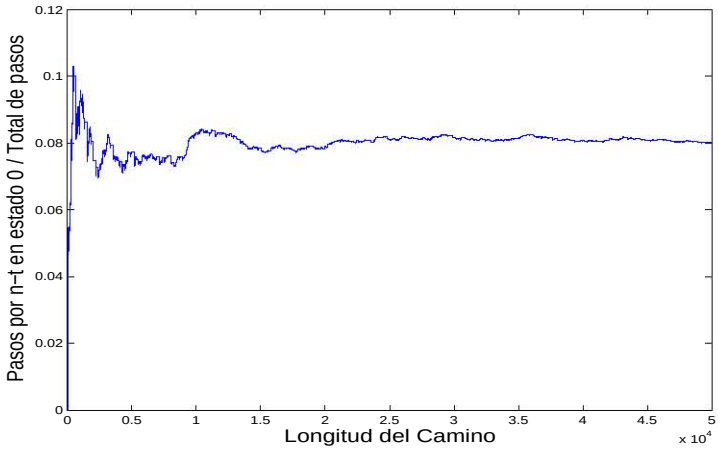


Figura 4.10: Convergencia mostrada por la simulación en los tránsitos hechos por $n - t$ en estado $0$.

Las gráficas 4.9 y 4.10 son similares en el sentido de que representan la inestabilidad en dos vértices que resultan ser complementarios dentro de un sistema simétrico. No obstante, para este caso como $t = n/2$ entonces $n - t = n/2$; y por ende representan la misma información.

La siguiente gráfica (figura 4.11) muestra la suma de la información de las dos gráficas anteriores (figura 4.9 y figura 4.10). En el eje $x$ tenemos nuevamente el total de tránsitos realizados. En el eje $y$, el número de veces que el programa de simulación ha pasado ya sea por el vértice $t_1$ en estado $1$ o por el vértice $n - t_1$ en estado $0$ y dividido entre el total de pasos.

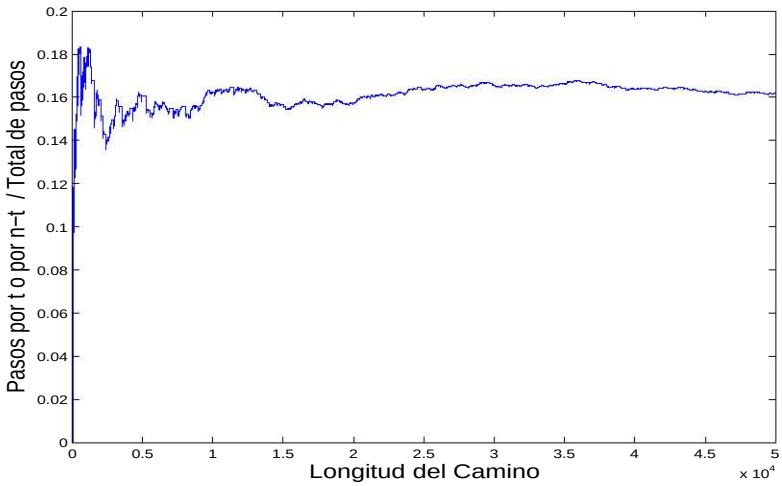


Figura 4.11: Convergencia mostrada en los tránsitos hechos por $t$ en estado $1$ o por $n - t$ en estado $0$.

Tal y como lo presentamos en el caso sin memoria, estas ejecuciones que acabamos de ver muestran el comportamiento de una ejecución en particular. Teniendo esto en cuenta y al estar asociada una probabilidad a su movimiento en el camino sobre el hipercubo (y por ende en el camino sobre la recta), es muy arriesgado sacar conclusiones de una sola ejecución, pues no muestran el comportamiento global del sistema. Por eso mismo realizamos las mismas simulaciones varias veces, obteniendo con ello que las curvas se fueron suavizando y mostrando una tendencia rápida hacia el valor de convergencia; para ello veamos las siguientes tres gráficas: figuras 4.12, 4.13 y 4.14.

Utilizamos como parámetros de entrada los valores $n = 100$, $t_1 = 50$, $l = 50000$ y se realizaron $100$ ejecuciones promediando los resultados obtenidos.

Posteriormente, en el siguiente capítulo (5) haremos una comparación con distintos valores de $n$ tanto en el modelo con memoria como sin memoria.

En el eje $x$ de estas gráficas mostramos el total de tránsitos realizados.

En el eje $y$, el número de veces que el programa de simulación ha pasado por el vértice $t_1$ en estado $1$ (figura 4.12) o el vértice $n - t_1$ en estado $0$ (figura 4.13) dividido entre el total de tránsitos realizados por nuestro programa de simulación.

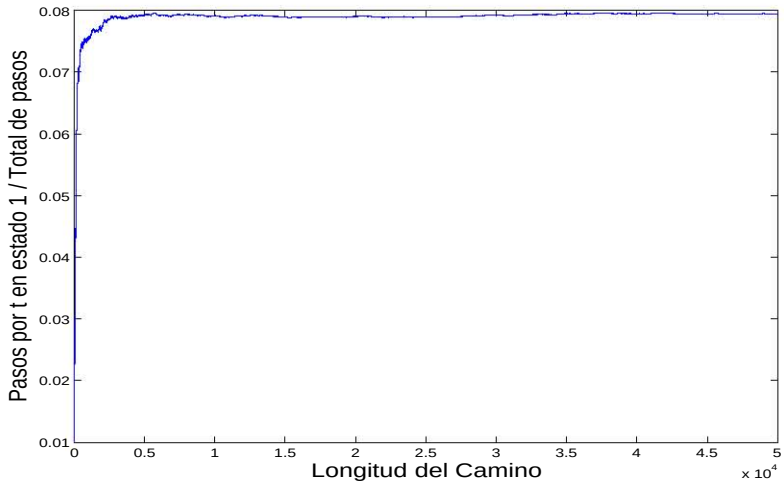


Figura 4.12: Convergencia mostrada en los tránsitos por $t$ en estado $1$ a lo largo de varias ejecuciones.

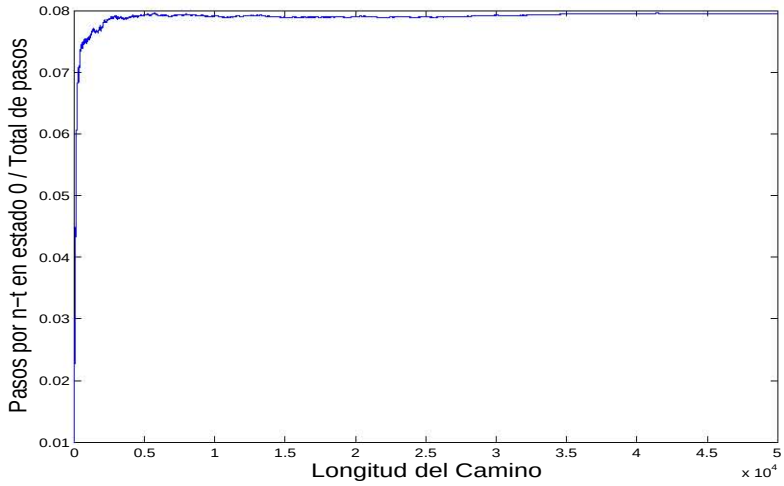


Figura 4.13: Convergencia mostrada en los tránsitos por $n - t$ en estado $0$ a lo largo de varias ejecuciones.

Podemos decir de las gráficas 4.12 y 4.13 lo mismo que de las gráficas 4.9 y 4.10: son similares en el sentido de que representan la inestabilidad en dos vértices que resultan ser complementarios dentro de un sistema simétrico. Es decir, que para estos experimentos, como tenemos que $t = n/2$ entonces $n - t = n/2$; y por lo tanto representan ejecuciones iguales.

La figura 4.14 es el resultado de varias ejecuciones, mostrando de forma similar lo representado en la figura 4.11. Muestra la suma de la información de las gráficas 4.12 y 4.13.

En el eje $x$ representamos el total de tránsitos realizados. En el eje $y$, el número de veces que el programa de simulación ha pasado por el vértice $t_1$ en estado $1$ o por el vértice $n - t_1$ en estado $0$.

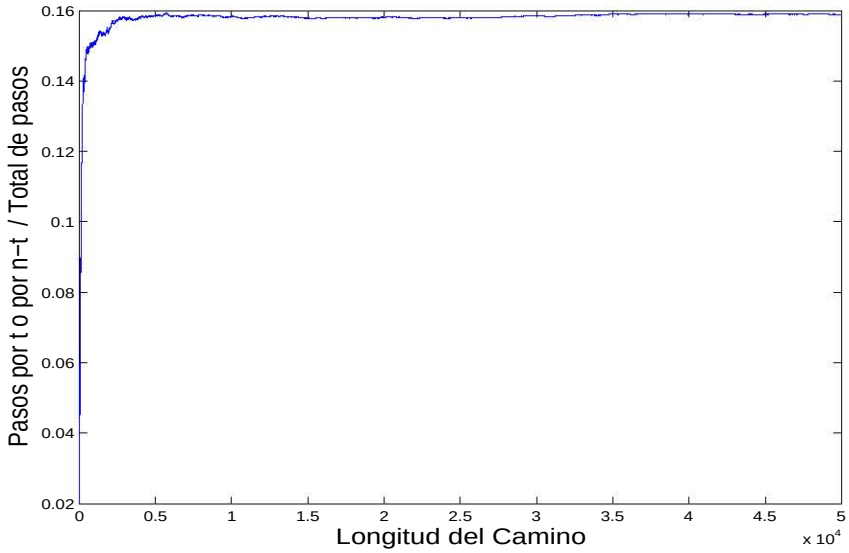


Figura 4.14: Convergencia mostrada en los tránsitos hechos por $t$ en estado $1$ o por $n - t$ en estado $0$ a lo largo de varias ejecuciones.

Notemos nuevamente que la curva se ha suavizado por el hecho de promediar las distintas ejecuciones y que además ha marcado mucho su convergencia hacia el valor esperado. Sin embargo, seguimos teniendo la restricción de que esta simulación se hizo para un valor particular de $t$. Luego entonces resultaría interesante ver el comportamiento del sistema con memoria variando el parámetro de tolerancia a fallos.

Recordemos que $t$ puede tomar valores entre $0$ y $n$. Así pues, las siguientes figuras muestran el comportamiento promedio obtenido por la simulación del sistema $D_1$ a lo largo de varias ejecuciones y además añadimos otro eje con los valores de $t$. Con esto obtenemos, las gráficas en 3D a fin de tener la información de tres parámetros al mismo tiempo.

Notemos algunos comportamientos interesantes.

La figura 4.15 es el resultado de la simulación del sistema con memoria $D_1$ a lo largo de varias ejecuciones dando diversos valores al parámetro de tolerancia a fallos $t$.

El número de procesos participantes en nuestra simulación del sistema fue $n = 100$ y una longitud de camino de $l = 10000$.

El eje derecho y el eje vertical son similares a los ejes $x$ y $y$ respectivamente de la figura 4.12. El eje izquierdo muestra los valores que va tomando $t$ entre $0$ y $n$.

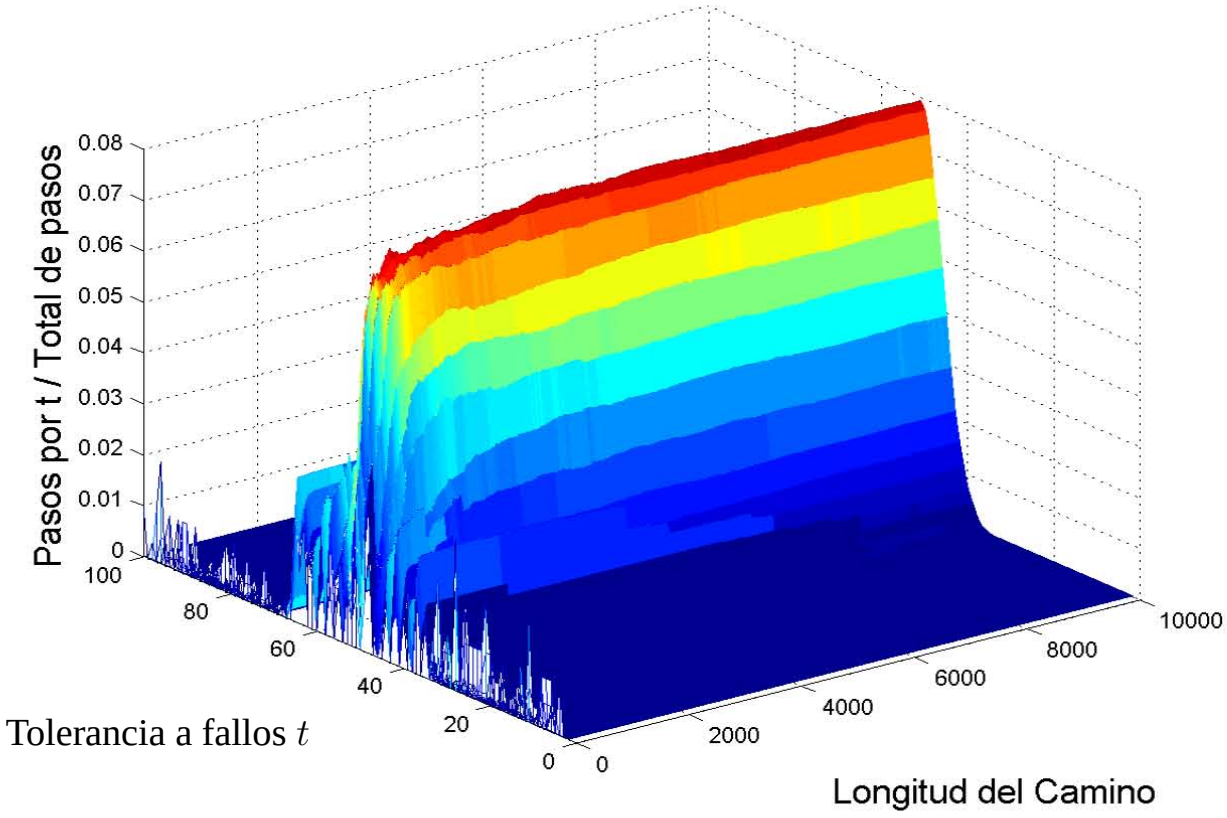


Figura 4.15: Simulación del modelo con memoria en 3D, de los pasos por $t$ en estado $1$, variando el parámetro $t$.

Observemos el comportamiento de la inestabilidad al inicio de las ejecuciones; es muy similar al mostrado por el modelo sin memoria. La elección del vértice inicial en el camino aleatorio está asociada a una distribución uniforme y no causa efectos en la tendencia de la inestabilidad cuando el número de pasos tiende a infinito.

Por otro lado, recordemos que en esta gráfica el eje vertical sólo representa la razón de los pasos hechos por el vértice $t$ en estado $1$ entre el total de pasos, por eso mismo es necesario mostrar también (en ese eje), la razón de los tránsitos realizados por el vértice $n - t$ en estado $0$ entre el total de pasos (figura 4.16) y la suma de ambos resultados (figura 4.17).

Para la siguiente gráfica (figura 4.16) el eje derecho y el eje vertical son similares a los ejes $x$ y $y$ respectivamente de la figura 4.13. Y nuevamente utilizamos los mismos valores de entrada: $n = 100$ y $l = 10000$.

Aquí, a diferencia de la gráfica anterior, el eje vertical muestra la razón de los pasos hechos por el vértice $n - t$ en estado $0$ entre el total de pasos. No obstante, por el comentario hecho más arriba, estas dos gráficas (4.15 y 4.16) también son similares en el sentido de que representan la inestabilidad en dos vértices complementarios dentro de un sistema simétrico.

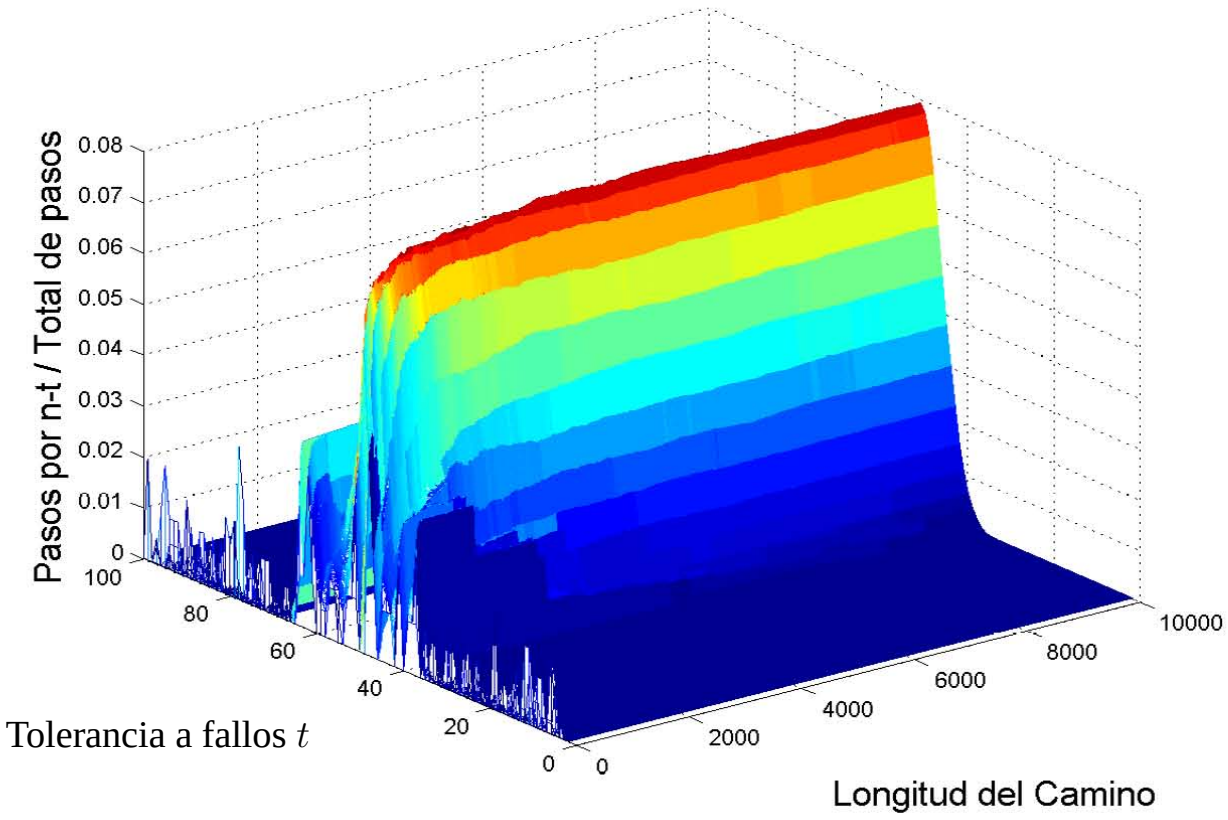


Figura 4.16: Simulación del modelo con memoria en 3D, de los pasos por $n - t$ en estado $0$, variando el parámetro $t$.

Ahora mostramos la gráfica que enseña la suma de la información en las dos gráficas previas.

En esta gráfica, nuestro eje vertical muestra normalizada la unión de los dos experimentos anteriores. El eje derecho y el eje vertical son similares a los ejes $x$ y $y$ respectivamente de la figura 4.14; y el eje izquierdo sigue mostrando los valores de $t$.

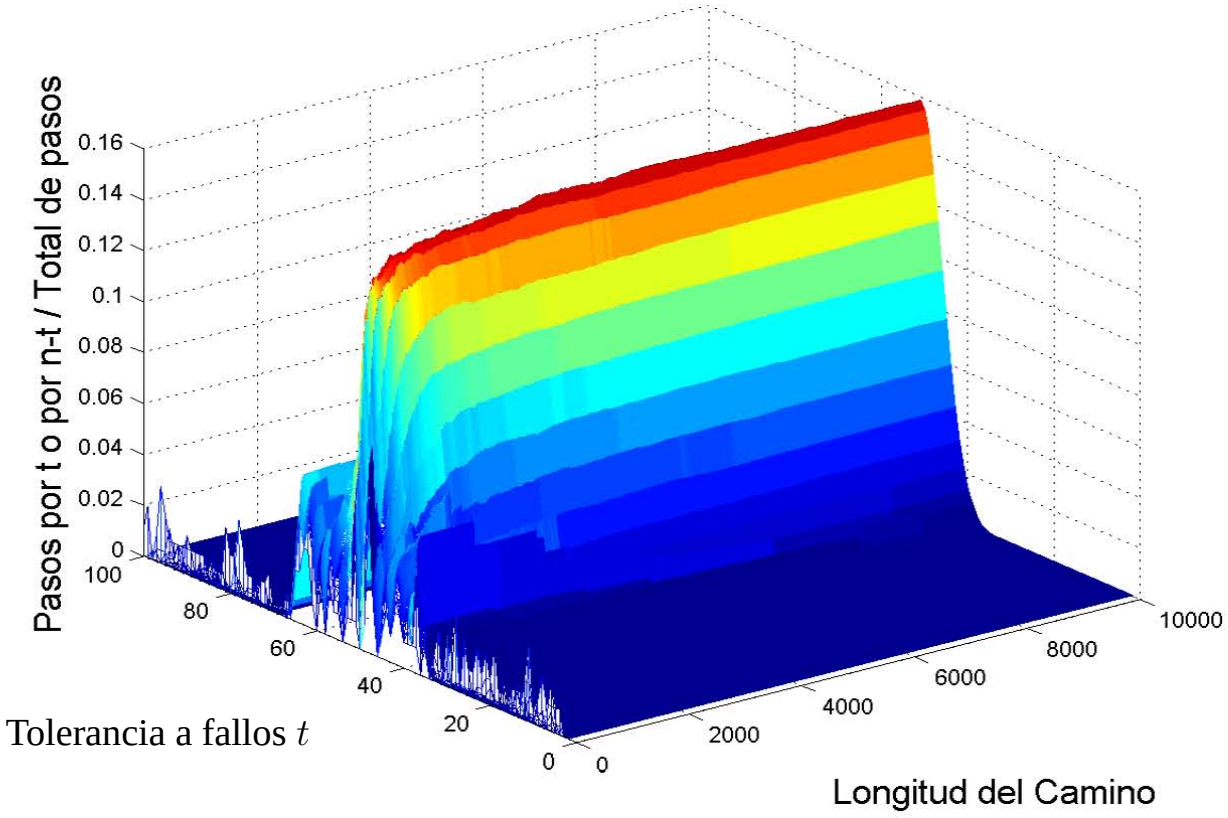


Figura 4.17: Simulación del modelo con memoria en 3D, pasando por $t$ o por $n - t$, variando el parámetro $t$.

En realidad las gráficas (4.15 y 4.16) muestran la misma información, tal y como pasa con las figuras (4.12 y 4.13) y con las figuras (4.9 y 4.10). Su similitud radica en que representan la inestabilidad en dos vértices complementarios dentro del sistema simétrico; por lo tanto su ejecución proporciona la misma información estadística; y esta última gráfica muestra el doble de los valores de las dos figuras anteriores.

Ahora bien, es necesario dar una interpretación estadística de los resultados obtenidos por estos experimentos de simulación. Para ello, en el siguiente capítulo vamos a comparar las ejecuciones de nuestros programas de simulación cuando el número de procesos participantes cambia. ¿Qué pasa cuando $n$ es cada vez más grande? Las simulaciones enseñan que efectivamente los modelos convergen al valor predicho por los investigadores, pero ¿qué criterio utilizar para la rapidez de convergencia? ¿Será necesario utilizar alguna técnica de sucesiones sobre las salidas de nuestro programa? ¿El orden de convergencia se modifica de manera proporcional con el parámetro $n$? Y de ser así ¿Qué métrica nos dice que tan acertada es esa proporción? Con estas preguntas es que cerramos este capítulo e iniciamos la interpretación de nuestros resultados.

# Capítulo 5

# Análisis de los resultados de las simulaciones de los modelos

En el capítulo anterior pudimos observar las gráficas resultantes de algunos experimentos de simulación. Corroboramos la convergencia predicha por los investigadores [4] para ambos modelos. Vimos también que la elección del vértice inicial en el camino aleatorio no afecta la tendencia de la inestabilidad en los sistemas. Notamos los puntos de mayor inestabilidad cuando el parámetro de tolerancias a fallos $t$ vale la mitad del número de procesos participantes $n$. Y surgieron las preguntas clásicas acerca del comportamiento de los sistemas en función a sus parámetros de entrada. ¿Qué pasa cuando $n$ cambia de valores? Sabemos que la convergencia hacia su valor esperado se logra en cualquiera de los casos; pero, cuando los procesos participantes son pocos ¿Se llegará primero a dicho valor que cuando son varios? Uno esperaría una respuesta afirmativa, pero de ser cierto eso ¿Qué tan rápido lo harán?

La idea es que una vez obtenidos los datos de la simulación, podamos dar una interpretación adecuada de los mismos. Es de nuestro interés observar la mayor inestabilidad alcanzada por los modelos, es decir cuando $t = n/2$, para cada respectiva $n$ de entrada.

Para ello debemos tener una definición exacta de lo que entendemos por alcanzar el valor de convergencia, para así poder comparar los resultados. Posteriormente haremos un diagrama de dispersión sobre los valores obtenidos, a fin de tener una curva de ajuste y así encontrar cuantitativamente una relación entre las variables de nuestros experimentos. Tengamos en cuenta que las salidas de nuestro experimento son valores aportados por variables estocásticas; por ende un manejo estadístico de la información resulta ser lo más adecuado.

En la literatura relacionada al tema de simulación computacional resulta útil la aplicación de los conceptos de alcance del estado estable, periodo de calentamiento y punto de fricción; los cuales definiremos más adelante.

El presente capítulo está organizado de la siguiente forma: la siguiente sección (5.1) mostrará a los valores dados por nuestros programas como resultados de variables aleatorias independientes e idénticamente distribuidas; posteriormente (sección 5.2) haremos un breve análisis de la aportación que hace el realizar diversas réplicas del experimento y promediarlas. Después (sección 5.3) utilizaremos un procedimiento para reducir la autocorrelación de nuestros datos a fin de suavizar la curva de su gráfica (llamado Procedimiento de Welch [32]). Una vez realizado esto, nos basaremos en el Proceso de Control Estadístico [33] con el fin de encontrar intervalos de confianza para nuestros experimentos (sección 5.4). Por último, concluimos haciendo comparaciones con distintos resultados donde el parámetro $n$ es nuestra variale independiente (esto lo hacemos tanto para el modelo con memoria como sin memoria, (secciones 5.5 y 5.6)).

## 5.1. Las simulaciones como experimentos

Las simulaciones se realizaron a fin de comparar la rapidez con la que convergen al valor esperado los modelos de sistemas distribuidos de consenso de larga vida, dando como entrada distintos valores en su parámetro $n$ y teniendo la mayor inestabilidad para esa $n$ particular (id est $t = n/2$).

Primero notemos que los datos aportados por nuestro programa de simulación están almacenados en un vector unidimensional. Segundo, apriori sabemos que el modelo converge a un valor dado por las fórmulas; por lo tanto este vector contendrá una sucesión convergente de datos. Debemos observar que esta sucesión es el resultado de variables estocásticas; así pues, resulta conveniente el manejo de estos datos por alguna técnica utilizada en el área de simulaciones y modelado computacional, con el objetivo de medir la rapidez con que ellos convergen. Y tercero, post hoc, poder hacer una comparación para saber si un determinado modelo alcanza antes su valor esperado si el número de procesos participantes cambia.

En muchos estudios de simulación se invierte una gran cantidad de tiempo y dinero en el desarrollo de modelos y programación, y se realiza un pequeño esfuerzo para analizar los datos de salida apropiadamente. De hecho, un modo muy común de operación es realizar una sola ejecución con alguna longitud arbitraria y entonces tratar de estimar los resultados de la simulación como las características del modelo "real". Desde que las muestras aleatorias de distribuciones de probabilidad son típicamente usadas para manejar un modelo de simulación a través del tiempo, estas estimaciones son precisamente realizaciones particulares de variables aleatorias que pueden tener grandes varianzas. Como resultado, estas estimaciones pueden, en alguna ejecución particular, diferir grandemente de las correspondientes características verdaderas del modelo. El efecto neto es, por supuesto, que pueda haber una probabilidad significante de inferencias erroneas acerca del sistema que se estudia. Consecuentemente, muchos "estudios" de simulaciones comienzan construyendo un modelo heurístico y terminan con una simple ejecución del programa para producir las "respuestas".

Sin embargo, una simulación es un experimento de muestreo estadístico basado en computadora [32]. Así, si los resultados de un estudio de simulación tienen algún significado, apropiadas técnicas de estadística pueden ser usadas para analizar los experimentos.

Describamos más precisamente la naturaleza aleatoria de las salidas de nuestra simulación. Sean $Y_1, Y_2, \ldots$ las salidas de un proceso estocástico como la ejecución simple de nuestro programa de simulación (por ejemplo vea la figura 4.4, página 27). Estas $Y_i$'s son variables aleatorias que, en general, pueden no ser independientes.

Ahora bien, sea $Y_{1,1}, Y_{1,2}, \ldots, Y_{1,l}$ el resultado de una sola ejecución de longitud $l$. Podemos hacer otra ejecución del mismo experimento, así tendriamos $Y_{2,1}, Y_{2,2}, \ldots, Y_{2,l}$. En general podemos hacer $m$ réplicas independientes de nuestro experimento dando como resultado la siguiente matriz de observaciones:

$$\begin{array}{ccccc} Y_{1,1} & \ldots & Y_{1,i} & \ldots & Y_{1,l} \\ Y_{2,1} & \ldots & Y_{2,i} & \ldots & Y_{2,l} \\ \vdots & & \vdots & & \vdots \\ Y_{m,1} & \ldots & Y_{m,i} & \ldots & Y_{m,l} \end{array}$$

Las observaciones de una ejecución en particular (una fila) claramente no son todas independientes e idénticamente distribuidas. Sin embargo, notemos que $Y_{1,i}, Y_{2,i}, \ldots, Y_{m,i}$ (la $i$-ésima columna) son observaciones indepenientes e idénticamente distribuidas de la misma variable aleatoria $Y_i$. Esta independencia através de las ejecuciones es la clave de lo relativamente simple de los métodos de análisis de los datos de salida. Entonces, estrictamente hablando, el objetivo del análisis de los datos de salida es el uso de las observaciones $Y_{j,i}$ para inferir sobre las variables aleatorias $Y_1, Y_2, \ldots, Y_l$; utilizando por ejemplo un estadístico como $\bar{Y}_i = \sum_{j=1}^{m} \frac{Y_{ji}}{m}$ como un estimado del valor esperado $E(Y_i)$.

## 5.2. Análisis de los datos de salida en los programas de simulación

El objetivo es presentar un método para el análisis estadístico de los datos de salida de nuestro experimento.

Sean $Y_1, Y_2, \ldots$ las salidas de un proceso estocástico como las de nuestro programa de simulación; y sea $F_i(y \mid I) = P(Y_i \leq y \mid I)$ para $i = 1, 2, \ldots$ donde $y$ es un número real e $I$ representa las condiciones iniciales utilizadas para comenzar la simulación al tiempo 0. Llamamos a $F_i(y \mid I)$ la distribución transitoria de un proceso al tiempo discreto $i$ para condiciones iniciales $I$.

Para $y$ e $I$ fijos, las probabilidades $F_1(y \mid I), F_2(y \mid I), \ldots$ son justamente una secuencia de números. Si $F_i(y \mid I) \to F(y)$ a medida que $i \to \infty$ para toda $y$ y para cualesquiera condiciones iniciales $I$, entonces $F(y)$ se llama la distribución de estado estable de la salida de los procesos $Y_1, Y_2, \ldots$ Estrictamente hablando, la distribución de estado estable $F(y)$ se obtiene solamente en el límite cuando $i \to \infty$.

En la práctica, sin embargo, habrá a menudo un índice de tiempo, digamos $k$, tal que la distribución en ese punto será aproximadamente la misma que en cualquier otro índice siguiente. Es por eso que decimos que el estado estable comienza al tiempo $k$; lo cual nos da un parámetro de referencia para saber cuál simulación, con determinados valores de entrada $(n, t)$, alcanza primero su estado estable. Notemos que estado estable no significa que las variables aleatorias $Y_k, Y_{k+1}, Y_{k+2} \ldots$ tomarán el mismo valor en una ejecución de la simulación en particular; más bien significa que todas tendrán aproximadamente la misma distribución.

Para nuestros experimentos utilizamos la media como estadístico de prueba; i. e. realizamos $m$ réplicas de la simulación, cada una de longitud $l$. De esa forma tendremos que $Y_{ji}$ será la $i-$ésima observación de la $j-$ésima réplica (con $i = 1, 2, \ldots, l$ y $j = 1, 2, \ldots, m$).

Después hacemos $\bar{Y}_i = \sum_{j=1}^{m} \frac{Y_{ji}}{m}$ para $i = 1, 2, \ldots, l$

$$\begin{array}{ccccc} Y_{1,1} & \ldots & Y_{1,i} & \ldots & Y_{1,l} \\ Y_{2,1} & \ldots & Y_{2,i} & \ldots & Y_{2,l} \\ \vdots & & \vdots & & \vdots \\ Y_{m,1} & \ldots & Y_{m,i} & \ldots & Y_{m,l} \\ \hline \\ \bar{Y}_1 & \ldots & \bar{Y}_i & \ldots & \bar{Y}_l \end{array}$$

Los procesos promedio $\bar{Y}_1, \bar{Y}_2, \ldots$ tienen medias $E(\bar{Y}_i) = E(Y_i)$ y varianzas $Var(\bar{Y}_i) = Var(Y_i)/m$. Así, los procesos promedio tienen la misma curva de medias transitoria que el proceso original, pero su gráfica tendrá solamente $\frac{1}{m}$ de varianza. Escogimos la simulación con mayor inestabilidad para una $n$ en particular y aplicamos este procedimiento (Figura 5.1).

Ahora bien, nosotros deseamos estimar el estado estable de medias $v = E(Y)$, el cual es generalmente definido como $v = \lim_{i \to \infty} E(Y_i)$. Así pues, el transito de medias converge al estado estable de medias. La consecuencia más seria del problema es que tal vez $E[Y(n)] \neq v$ para alguna $n$. Los investigadores del área de simulación computacional sugieren para tratar este problema, una técnica comúnmente utilizada llamada: *periodo de calentamiento (warmup period)* o *eliminación de datos iniciales*. La idea es eliminar (o ignorar) algún número de observaciones del inicio de la ejecución y utilizar solamente el resto a fin de estimar $v$.

Por ejemplo, dadas las observaciones $Y_1, Y_2, \ldots, Y_l$ se recomienda utilizar a $\bar{Y}(l, wp)$ en vez de $\bar{Y}(l)$ como un estimador de $v$.

Donde $\bar{Y}(l, wp) = \frac{\sum_{i=wp+1}^{l} Y_i}{l - wp}$ y $(1 \leq wp \leq l - 1)$.

En general, uno esperaría que $\bar{Y}(l, wp)$ sea menos sesgado que $\bar{Y}(l)$; ya que las observaciones cercanas al inicio de la simulación pueden no ser muy representativas del comportamiento del estado estable.

La pregunta obvia que surge es ¿Cómo elegir el periodo de calentamiento $wp$?

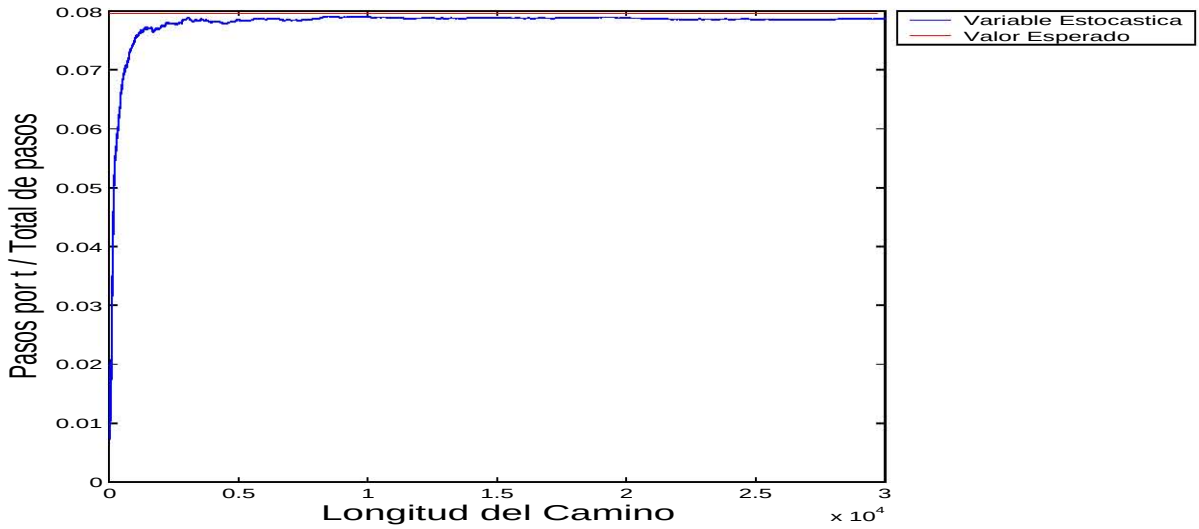


Figura 5.1: Salida estocástica de la simulación donde la inestabilidad es mayor; i.e. $t = n/2$.

Quisieramos elegir a $l$ y a cada vértice $wp$ tales que $E[\bar{Y}(l, wp)] \approx v$. Si se escogen $l$ y $wp$ muy chicos, entonces $E[\bar{Y}(l, wp)]$ puede ser significativamente diferente de $v$. Por otro lado, si $wp$ se escoge más grande de lo necesario, entonces $\bar{Y}(l, wp)$ probablemente tenga una varianza innecesariamente grande [32] o bien se desperdicie información de interés.

## 5.3. Método de Welch

Hasta el momento tenemos lo siguiente:

- $m$ réplicas de la simulación computacional; cada una de longitud $l$ (con $l$ lo suficientemente grande). Y tenemos $Y_{ji}$ ($i-$ésima observación de la $j-$ésima réplica) como las entradas de nuestra matriz de observaciones.
- $\bar{Y}_i = \sum_{j=1}^{m} \frac{Y_{ji}}{m}$ para $i = 1, 2, \ldots, l$ Los procesos promedio $\bar{Y}_1, \bar{Y}_2, \ldots, \bar{Y}_l$ con medias $E(\bar{Y}_i) = E(Y_i)$ y varianzas $Var(\bar{Y}_i) = Var(Y_i)/m$. Con esto, los procesos promedio tienen la misma curva de medias que el proceso original, pero su gráfica tendrá solamente $\frac{1}{m}$ de varianza.

Ahora bien, para suavizar las oscilaciones de alta frecuencia en $\bar{Y}_1, \bar{Y}_2, \ldots, \bar{Y}_l$ (pero dejando las oscilaciones de baja frecuencia o la tendencia de largas ejecuciones de interés), definimos aún más, el movimiento promedio $\overline{Y_i(w)}$ donde $w$ se denomina ventana y es un número entero positivo tal que $w \leq \lfloor l/2 \rfloor$ como sigue:

$$\overline{Y_i(w)} = \begin{cases} \frac{\sum_{s=-w}^{w} \bar{Y}_{i+s}}{2w+1} & \text{si i = w+1, w+2, \ldots, l-w} \\ \\ \frac{\sum_{s=-(i-1)}^{i+1} \bar{Y}_{i+s}}{2i-1} & \text{si i = 1, 2, \ldots, w} \end{cases}$$

Este procedimiento debido a P. D. Welch [37], sirve como filtro a fin de eliminar el "ruido" en nuestro proceso estocastico y suavizar la curva de medias (vea la curva resultante en la figura 5.2 después de aplicar el método de Welch a las salidas mostradas en la figura 5.1) y así poder obtener posteriormente un intervalo de confianza en el cual encontraremos los valores de nuestras variables estocásticas en un estado estable de medias.

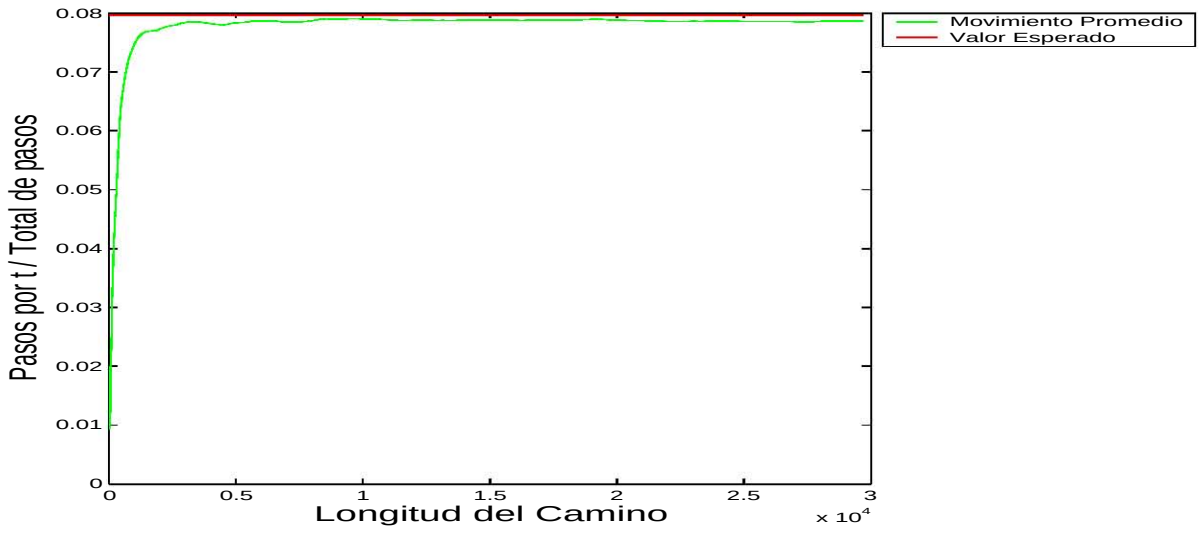


Figura 5.2: Resultado después de aplicar el método de Welch a la salida estocástica a fin de reducir la autocorrelación.

## 5.4. Proceso de control estadístico

En este punto, nosotros tenemos la curva de medias no autocorrelacionada; por ello, basándonos en el artículo [33] describiremos los pasos realizados para encontrar el periodo de calentamiento *(warmup period)* de nuestras simulaciones. Este método denominado *SPC* por sus siglas en ingés: Statistical Process Control (Proceso de Control Estadístico) fue presentado por Stewart Robinson en el artículo antes señalado.

Como hemos visto, muchas simulaciones que muestran una tendencia hacia un *estado estable* pasan por una serie de resultados iniciales, los cuales podemos decir se encuentran "fuera de control" o lejos del valor esperado. Esto debido a las condiciones iniciales y tendencias de inicialización de las simulaciones. Ese intervalo de salidas iniciales se llama *periodo de calentamiento*.

Entendemos como tal que al finalizar el *periodo de calentamiento* entonces nuestro experimento de simulación ha alcanzado el *estado estable*. Notemos que para nuestro caso tanto el periodo de calentamiento como el estado estable, se esperaria fuesen únicos, ya que el modelo apriori converge y por lo tanto una vez alcanzado su estado de estabilidad no saldrá de él.

Así pues es necesario definir el punto donde ocurre dicha transición y lo llamaremos *punto de fricción*.

Comencemos por describir el método: hasta aquí disponemos de los procesos promedio $\bar{Y}_1, \bar{Y}_2, \ldots, \bar{Y}_l$ donde $l$ es la longitud de la simulación; y los disponemos en un vector $y(l) = [\bar{Y}_1, \bar{Y}_2, \ldots, \bar{Y}_l]$

Ahora, una vez aplicado el método de Welch para disminuir su autocorrelación, disponemos las observaciones en lotes, las medias de los lotes se hacen aproximadamente no correlacionadas conforme el tamaño del lote ($k$) crece. La serie de datos del vector $y(l)$ se combinan en una serie de lotes para generar el vector $y(k)$ de la siguiente manera:

$$y(k) = \left( \frac{\sum_{i=1}^{k} \bar{Y}_i}{k}, \frac{\sum_{i=k+1}^{2k} \bar{Y}_i}{k}, \ldots, \frac{\sum_{i=(b-1)k+1}^{bk} \bar{Y}_i}{k} \right)$$

el cual podemos representar como $y(k) = [y(1), y(2), \ldots, y(b)]$ donde $b$ es el número de lotes y cada uno es de longitud $k$. Lo que significa que $l = bk$ El autor recomienda utilizar el método de Fishman [38] para determinar el tamaño de $k$: comenzando con lotes de tamaño 1, el valor de $k$ se incrementa hasta que la autocorrelación del intervalo 1 es cercana a 0.

Otro detalle importante del método es que los datos se suponen están distribuidos normalmente; nuevamente, el autor propone la técnica de Kolmogorov-Smirnov como prueba de normalidad. Sin embargo, por el desempeño de una serie de varias réplicas, se puede asumir (mediante el Teorema del Límite Central) que la muestra de medias tiende hacia la normalidad incrementando el tamaño de lote $k$. De igual manera se recomienda un número de lotes $b$ mayor a 60 (y por ende un número mayor de longitud de la simulación $l$). Debemos notar que a medida que $k$ se incrementa, la media de los lotes tiende a normalizarse [33].

Una muestra de tamaño más grande refuerza esta suposición. Por lo tanto, más réplicas del experimento de simulación pueden desempeñar un mejor resultado. De nuevo, el autor nos recomienda hacer al menos diez réplicas si es posible.

Una vez que tenemos nuestros valores $l$, $k$ y $b$ (donde $l = bk$), notemos que deberiamos usar el mínimo valor para $k$ tal que pase las pruebas de autocorrelación y normalidad.

La media poblacional ($\widehat{\mu}$) y la desviación estandar ($\widehat{\sigma}$) de los datos en $y(k)$ se estiman de la segunda mitad de la serie. En otras palabras, si el número de lotes ($b$) es 100, entonces la media y la desviación estandar se calculan para los datos en el rango $b = 51$ hasta $b = 100$. Es por esto que se recomienda [33] que al menos tengamos 60 datos sobre $y(k)$. Obviamente el número de datos aportados por nuestra simulación es necesario que sea multiplo de $k$ a fin de asegurar que hay 60 datos en la serie de lotes de medias.

Con todo esto en cuenta, procedemos a calcular la desviación estandar de la siguiente manera:

$$\widehat{\sigma} = \sqrt{\frac{1}{\lfloor \frac{b}{2} \rfloor} \cdot \sum_{j=b-\lfloor \frac{b}{2} \rfloor + 1}^{b} s_j^2}$$

Donde $s_j^2$ es la desviación estandar de las medias individuales en $y(k)$, y se calcula de la observación individual obtenida de cada réplica. Esto es, si se han desarrollado 10 réplicas, $s_j^2$ debe ser la desviación estandar de los 10 datos en $y(j)$.

Con estos datos obtenemos dos intervalos de control. El primero se denomina *Límites de Advertencia (Warning Limits)*

$$WL = \widehat{\mu} \pm \frac{1{,}96\widehat{\sigma}}{\sqrt{n}}$$

Y el segundo se denomina *Límites de Acción (Action Limits)*

$$AL = \widehat{\mu} \pm \frac{3{,}09\widehat{\sigma}}{\sqrt{n}}$$

Con esto se crean intervalos de confianza donde se espera que 1 de 1000 datos caigan fuera de los Límites de Acción, mientras que 1 de 40 caiga fuera de los Límites de Advertencia. Stewart Robinson [33] propone a continuación una tabla de control a fin de determinar el final del periodo de calentamiento (o en otras palabras, el inicio del estado estable).

Recordemos que al inicio de nuestras simulaciones los datos de $y(l)$ y por supuesto de $y(k)$ estarán "fuera de control" o en periodo de calentamiento. Una vez que el modelo ha alcanzado el estado estable, los datos estarán "bajo control" y las reglas que propone para identificar en la tabla de control a los valores o datos que se encuentran "fuera de control" son las siguientes:

- Los datos de la serie violan los Límites de Acción.
- Dos valores consecutivos violan cualquiera de los límites inferior o superior del intervalo de Advertencia.
- Valores frecuentes en relativa cercania violan los límites de Advertencia.
- Hay excesivo zigzagueo en bastantes puntos cercanos a la media $\widehat{\mu}$.

En nuestro caso, observamos de las simulaciones una sucesión de datos convergentes hacia el valor esperado, por ello basándonos en las reglas anteriores y según las necesidades experimentales de nuestro experimento de simulación, procedemos con la siguiente:

**Definición 10** *Sea $S_n$ una sucesión de datos convergente hacia $E(y)$ obtenidos por nuestra simulación, y sean $WL$ y $AL$ sus intervalos de Advertencia y de Acción respectivamente, entonces si existe un único punto en el camino que intersecte el límite inferior de $AL$ y a partir de él se cumplen las cuatro reglas de la tabla de control, se denominará* Punto de Fricción *y se denotará $F_P$.*

Ahora bien, veamos el resultado (Figura 5.3) de aplicar todo el Proceso de Control Estadístico (SPC) a los datos mostrados por la figura 5.2:

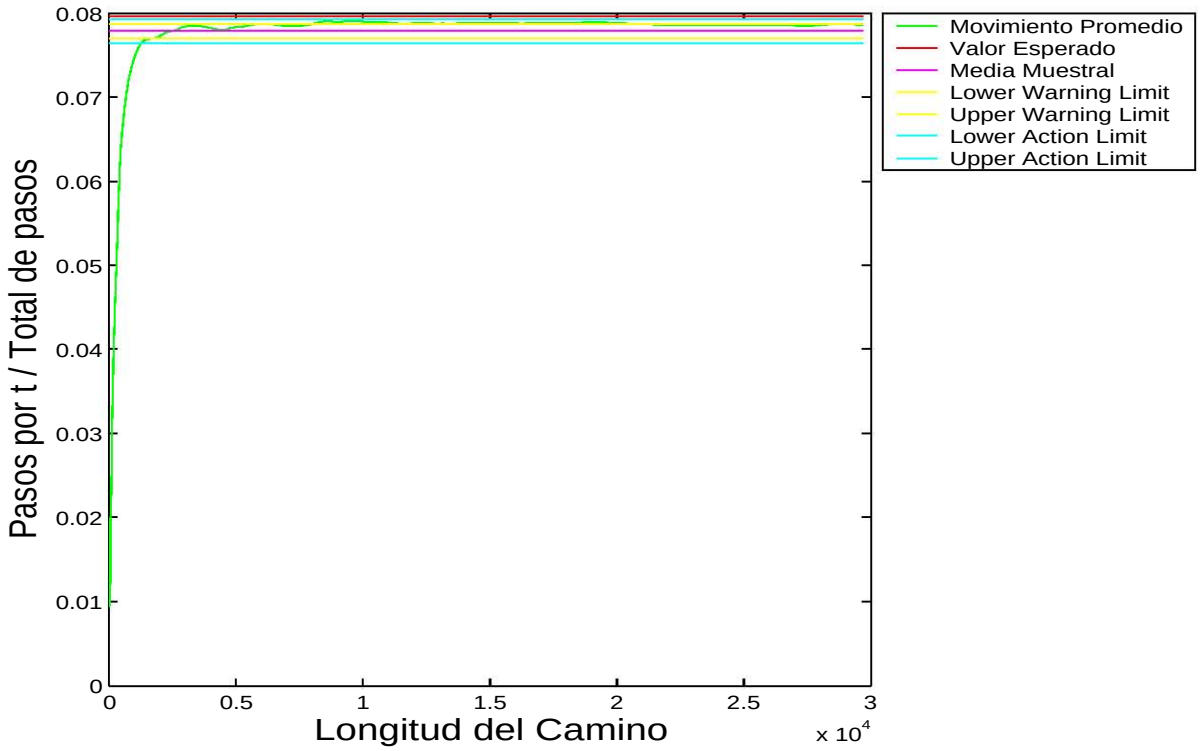


Figura 5.3: Intervalo de confianza encontrado al aplicar el Proceso de Control Estadístico.

Podemos apreciar perfectamente que los datos de la serie cumplen con las reglas de la tabla de control y así nuestro programa puede encontrar el punto de fricción.

## 5.5. Comparaciones con distintos valores del parámetro $n$ para el modelo sin memoria

Teniendo ya definidas todas las herramientas a utilizar, procedemos a describir ahora el comportamiento de una secuencia de datos aportada por nuestra simulación, como el de la figura 4.5 (página 28).

Aplicamos a diversas simulaciones (las cuales se efectuaron con distintos valores del parámetro $n$) las técnicas antes descritas. La figura 5.4 muestra la comparación hecha a 11 simulaciones.

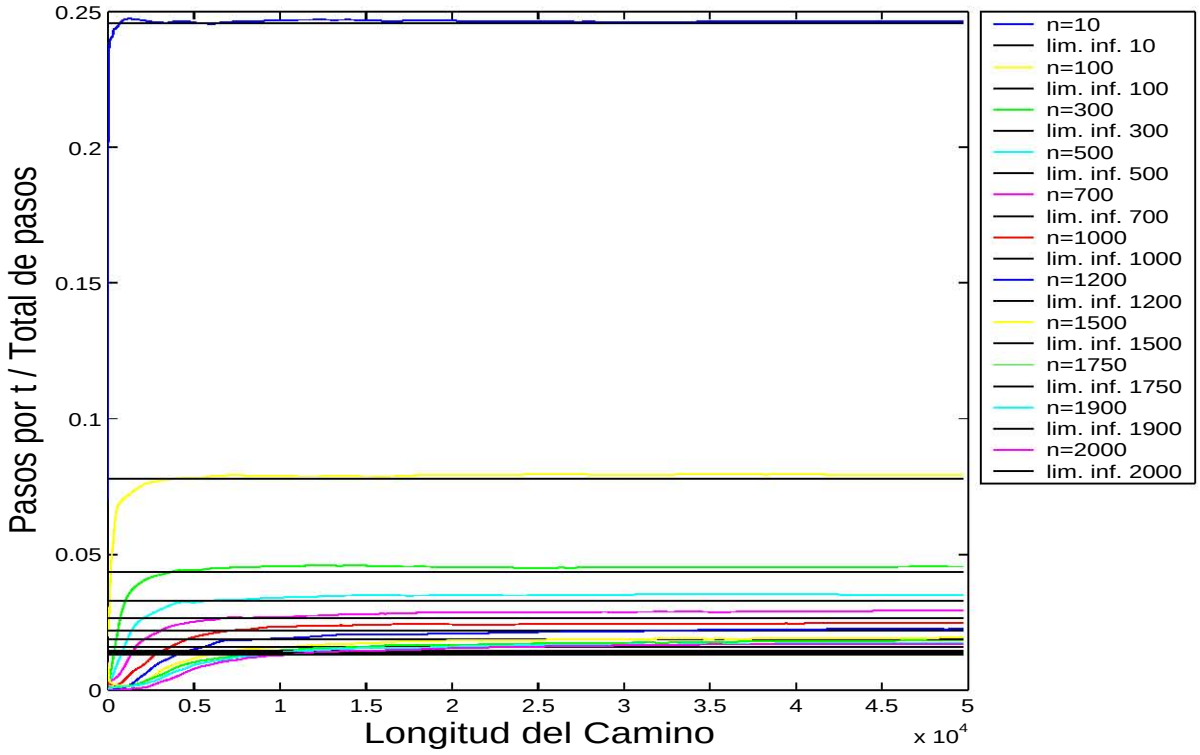


Figura 5.4: Comparacion de convergencia en el modelo sin memoria variando el parámetro $n$.

Los resultados de este experimento los podemos apreciar en la figura 5.5 y en la tabla 5.1.

Muy a menudo en la práctica se encuentra que existe una relación entre dos (o más) variables, y se desea expresar esta relación en forma matemática determinando una ecuación que conecte las variables (en nuestro caso, el valor del parámetro $n$ y el punto de fricción $F_P$).

Generalmente, más de una curva de un tipo dado parece ajustar un conjunto de datos. Para evitar el juicio individual en la construcción de curvas de aproximación, es necesario obtener una definición de la "recta de mejor ajuste".

Consideremos los datos $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ y una curva de ajuste $C$. Así, habrá una cierta diferencia entre el valor dado $(x_1, y_1)$ y el valor correspondiente en la curva $C$. Denotemos esta diferencia como $d_1$. Análogamente para cada uno de los valores tendremos las diferencias $d_2, d_3, \ldots, d_n$.

Una medida de la bondad del ajuste de la curva $C$ a los datos es la suma del cuadrado de esas diferencias $\sum_{i=1}^{n} d_i^2$. Si el resultado es pequeño el ajuste es bueno. Por lo tanto, requerimos de que dicho valor sea el mínimo posible.

Empleando lo anterior, lo que intentamos es obtener la recta de mínimos cuadrados que se aproxime a nuestros datos.

En particular, la figura 5.5 es un diagrama de dispersión donde se relacionan los valores de $n$ con los de sus respectivos $F_P$.

De este diagrama es posible visualizar una curva que se aproxime a los datos. Dicha curva se llama curva de ajuste (y se muestra en la figura 5.6).

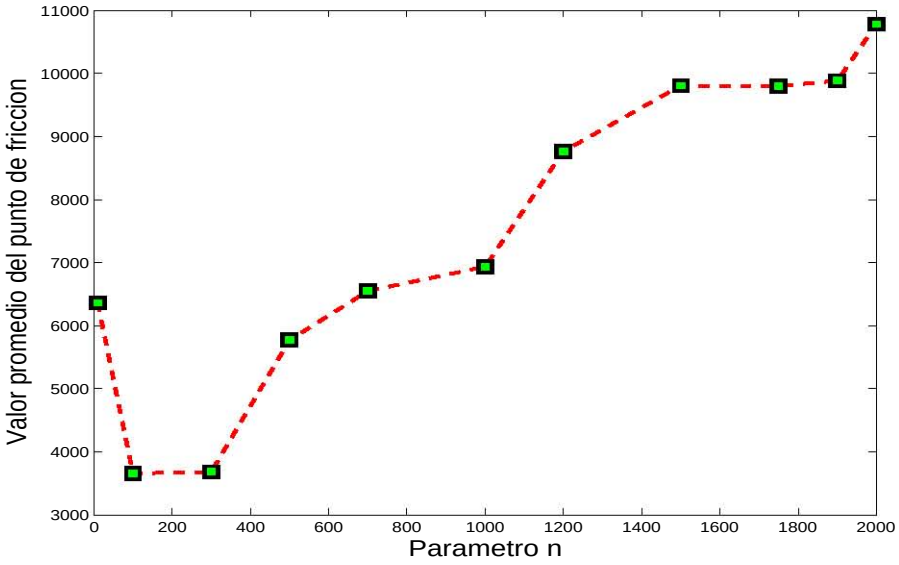


Figura 5.5: Puntos de fricción encontrados en los experimentos de simulación para el caso sin memoria.

| Procesos participantes ($n$) | Valor del punto de fricción |
|---|---|
| 10 | 6362 |
| 100 | 3659 |
| 300 | 3685 |
| 500 | 5773 |
| 700 | 6552 |
| 1000 | 6938 |
| 1200 | 8764 |
| 1500 | 9808 |
| 1750 | 9801 |
| 1900 | 9886 |
| 2000 | 10784 |

Cuadro 5.1: Comparación de los valores de $n$ y sus respectivos puntos de fricción en el modelo sin memoria.

Por lo tanto, de los datos de la tabla 5.1 y las fórmulas mencionadas en el apéndice A.9 podemos obtener la pendiente $m$ y la ordenada al origen $b$ de la recta de mínimos cuadrados que mejor se ajusta a nuestros datos de la simulación para el modelo sin memoria:

$$y = 3.2x + 4255.9$$

Y con un coeficiente de correlación lineal de

$$r^2 = 0.8583$$

Podemos observar la recta en la siguiente figura:

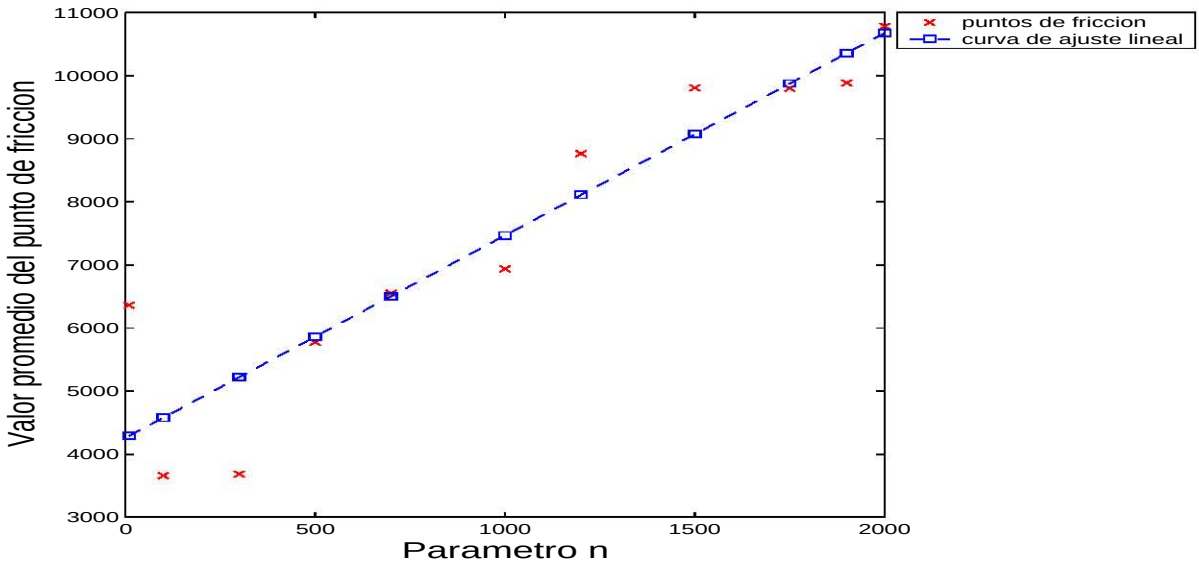


Figura 5.6: Curva de ajuste lineal para los puntos de fricción en el modelo sin memoria.

Recordemos que el coeficiente de correlación $r$ nos dice qué tan bien la recta de regresión de mínimos cuadrados se ajusta a nuestros datos muestrales. Si la variación total se explica totalmente por la recta de regresión, entonces $r^2 = 1$ ó $r = \pm 1$ significa que hay una correlación lineal perfecta (y en tal caso una regresión lineal perfecta). De otro modo, si la variación total no se puede explicar entonces la variación explicada (vease el apéndice A.9) es cero; y así $r = 0$. Esta cantidad $r^2$ llamada algunas veces en la práctica coeficiente de determinación, se encuentra entre 0 y 1; y nos dice cuantitativamente que tan bueno es el ajuste de la curva a nuestros datos.

## 5.6. Comparaciones con distintos valores del parámetro $n$ para el modelo con memoria

Nuevamente, teniendo ya definidas todas las herramientas a utilizar, describiremos ahora el comportamiento de las secuencias de datos como las aportadas en la simulación mostradas en las figuras 4.12 y 4.13 (página 34).

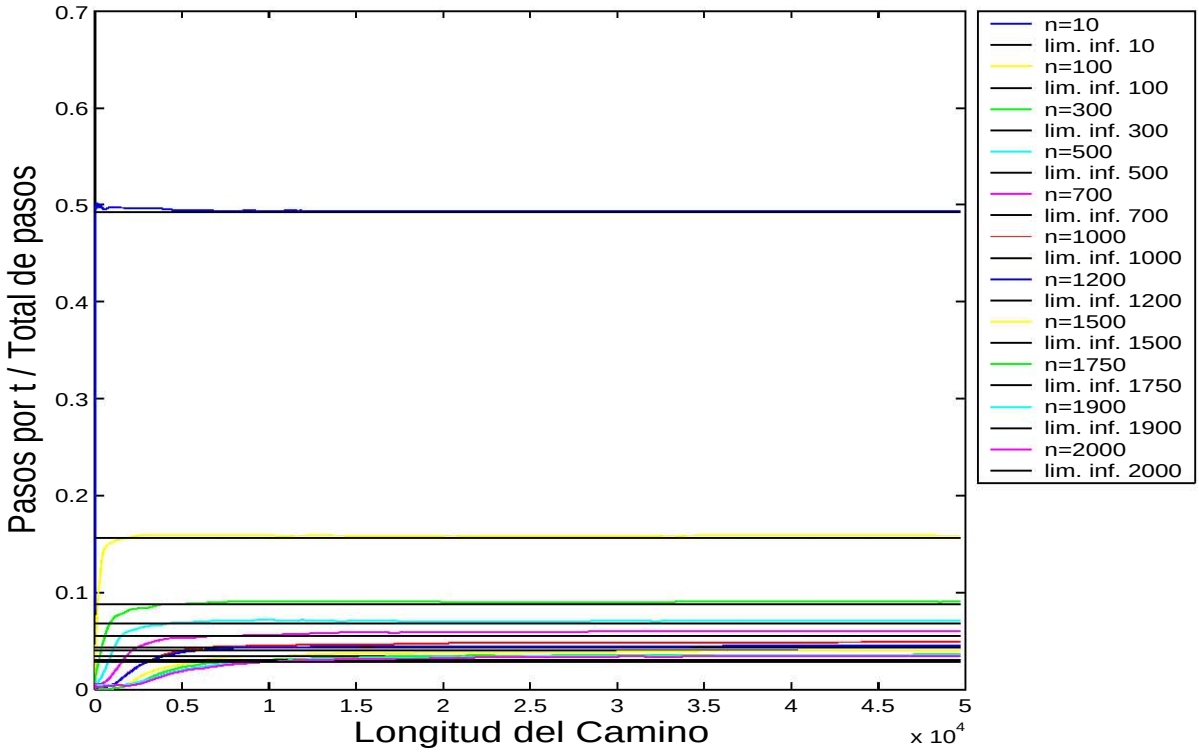


Figura 5.7: Comparacion de convergencia en el modelo con memoria variando el parámetro $n$.

Los resultados de este experimento los podemos apreciar en la figura 5.8 y en la tabla 5.2.

Realicemos el mismo análisis que hicimos al modelo sin memoria con estos datos. Encontremos si existe una relación entre las dos variables (el valor del parámetro $n$ y el punto de fricción $F_P$), y expresemos esta relación en forma matemática determinando una ecuación lineal que las relacione.

Consideremos los datos $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ y una curva de ajuste $C$. Habrá una cierta diferencia entre el valor dado $(x_1, y_1)$ y el valor correspondiente en la curva $C$. Denotemos esta diferencia como $d_1$. Análogamente para cada uno de los valores tendremos las diferencias $d_2, d_3, \ldots, d_n$.

Una medida de la bondad del ajuste de la curva $C$ a los datos es la suma del cuadrado de esas diferencias $\sum_{i=1}^{n} d_i^2$. Si el resultado es pequeño el ajuste es bueno. Por lo tanto, requerimos que dicho valor sea el mínimo posible.

De esta forma, evitamos el juicio individual en la construcción de curvas de aproximación.

Otra vez, empleando todo lo anterior, lo que intentamos es obtener la recta de mínimos cuadrados que se aproxime a nuestros datos.

En particular, la figura 5.8 es un diagrama de dispersión donde se relacionan los valores de $n$ con los de sus respectivos $F_P$.

De este diagrama es posible visualizar una curva que se aproxime a los datos. Dicha curva se llama curva de ajuste (y se muestra en la figura 5.9).

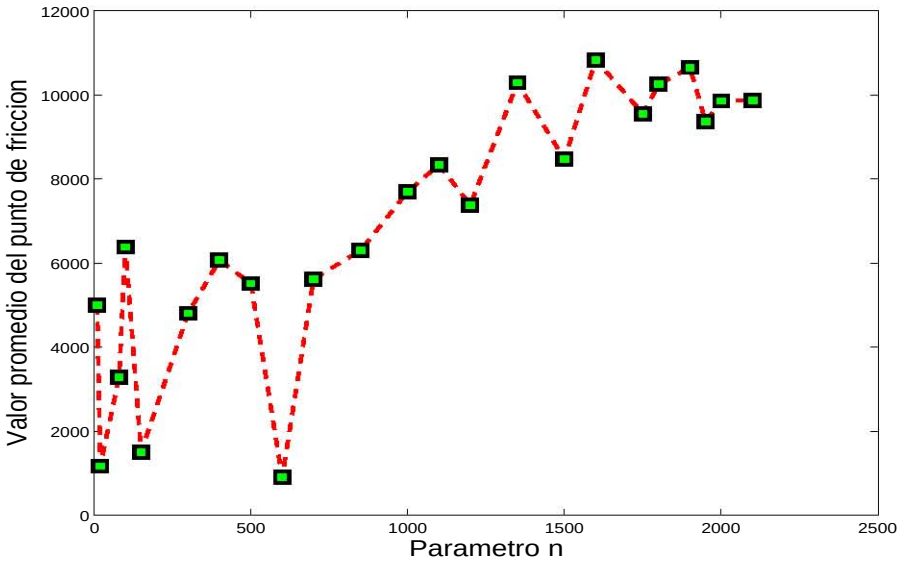


Figura 5.8: Puntos de fricción encontrados en los experimentos de simulación para el caso con memoria.

La siguiente tabla, nos muestra los resultados experimentales obtenidos de la simulación. Como comentario al margen, la duración de uso de CPU en este experimento fue de 12 hrs. 09 min. y 58 seg.

| Procesos participantes ($n$) | Valor del punto de fricción |
|---|---|
| 10 | 5000 |
| 20 | 1178 |
| 80 | 3292 |
| 100 | 6385 |
| 150 | 1508 |
| 300 | 4807 |
| 400 | 6070 |
| 500 | 5520 |
| 600 | 905 |
| 700 | 5624 |
| 850 | 6303 |
| 1000 | 7704 |
| 1100 | 8342 |
| 1200 | 7383 |
| 1350 | 10292 |
| 1500 | 8472 |
| 1600 | 10837 |
| 1750 | 9554 |
| 1800 | 10253 |
| 1900 | 10654 |
| 1950 | 9368 |
| 2000 | 9859 |
| 2100 | 9862 |

Cuadro 5.2: Comparación de los valores de $n$ y sus respectivos puntos de fricción en el modelo con memoria.

Por lo tanto, de los datos de la tabla 5.2 y las fórmulas mencionadas en el apéndice A.9 podemos obtener la pendiente $m$ y la ordenada al origen $b$ de la recta de mínimos cuadrados que mejor se ajusta a nuestros datos de la simulación para el modelo sin memoria:

$$y = 3.84x + 2996.5$$

Y con un coeficiente de correlación lineal de

$$r^2 = 0.7517$$

Podemos observar la recta en la siguiente figura:

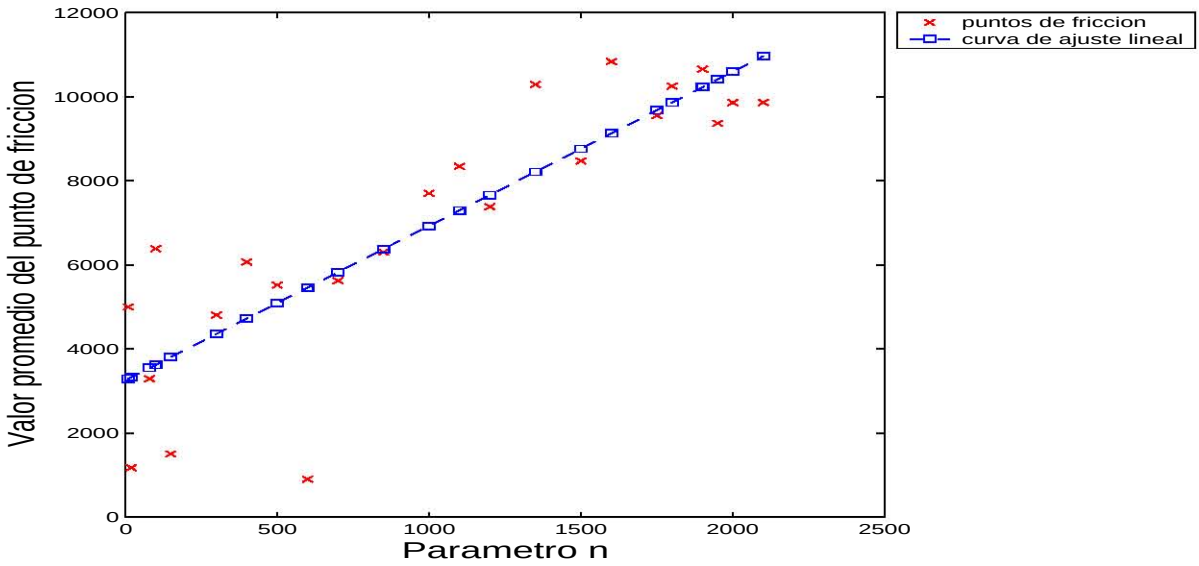


Figura 5.9: Curva de ajuste lineal para los puntos de fricción en el modelo con memoria.

Recapitulando lo dicho en la sección 5.5, el coeficiente de correlación $r$ nos dice qué tan bien la recta de regresión de mínimos cuadrados se ajusta a nuestros datos muestrales.

Si la variación total se explica totalmente por la recta de regresión, entonces $r^2 = 1$ ó $r = \pm 1$ significa que hay una correlación lineal perfecta (y en tal caso una regresión lineal perfecta). De otro modo, si la variación total no se puede explicar entonces la variación explicada (vease el apéndice A.9) es cero; y así $r = 0$.

Esta cantidad $r^2$ llamada algunas veces en la práctica coeficiente de determinación, se encuentra entre 0 y 1; y nos dice de manera cuantitativa que tan bueno es el ajuste de la curva a nuestros datos.

Notemos el valor de nuestro coeficiente de correlación es algo menor que en el caso sin memoria. Por lo tanto, una conclusión a extraerse es que existe una menor correlación lineal entre los datos; sin embargo sigue siendo considerablemente de ajuste lineal.

# Capítulo 6

# Conclusiones y trabajo futuro

En este trabajo pudimos estudiar los sistemas distribuidos de consenso continuo desde la perspectiva de su inestabilidad; es decir, de la cantidad de cambios de decisión a lo largo de su ejecución. Comenzamos haciendo una introducción al problema clásico del consenso en los sistemas distribuidos de cómputo. Posteriormente, vimos con un poco de más formalidad los requerimientos que debe cumplir cualquier algoritmo de consenso; haciendo énfasis en lo que consiste el problema del consenso continuo o de larga vida. De igual manera, abordamos una reseña cronológica de los trabajos previos y resultados obtenidos en el área.

En la parte inicial del documento, dimos las definiciones necesarias para manejar los sistemas de consenso de larga vida y mostramos los resultados de inestabilidad cuando hacemos uso de entradas geodésicas.

Post hoc, estudiamos la parte medular que sirve como sustento de las simulaciones desarrolladas, es decir la modificación de los modelos a una representación por hipercubo y la definición de inestabilidad promedio. Presentando además, la inestabilidad que en este nuevo modelo tienen los sistemas tanto con memoria como sin memoria. Hecho esto, procedimos a explicar los experimentos de simulación resaltando las predicciones dadas por los investigadores en la convergencia de la inestabilidad hacia un valor esperado.

Podemos mencionar, que durante el desarrollo de este trabajo se participó en dos congresos: en el primero, con un artículo donde se mencionan los primeros avances de esta tesis, como son, el cambio en el modelo de entradas geodésicas al modelo de hipercubo y las simulaciones computacionales de los sistemas con memoria y sin memoria; en el segundo, con la presentación de un fast abstract mencionando las aportaciones realizadas por los experimentos, id est, los resultados de obtener una ecuación matemática que relacione el valor de los parámetros de entrada con la velocidad de convergencia de los modelos.

Es precisamente esto último la principal aportación de la tesis: presentar un estudio de la velocidad de convergencia utilizando técnicas estadísticas de simulaciones y modelado computacional. Debido a que las salidas de nuestros experimentos son aportadas por variables aleatorias, entonces el concepto de rapidez, velocidad u orden de convergencia toma un sentido completamente diferente a los que tomaría en sucesiones analíticas. Por lo tanto, fue de utilidad la aplicación de los conceptos de alcance del estado estable, periodo de calentamiento y punto de fricción.

Para responder a la pregunta original de qué tan rápido converge un modelo de consenso de larga vida cuando el valor en su parámetro $n$ cambia, basta con mostrar que la mejor curva de ajuste obtenida por los experimentos en ambos modelos, muestra una relación directamente proporcional (aunque de cierto modo con mayor ajuste lineal en el caso sin memoria que en el caso con memoria).

Así pues, a prima facie uno puede concluir (con los datos obtenidos de la simulación y por el coeficiente de determinación menor en $D_1$ que en $D_0$) que existe menos correlación lineal para el caso de sistemas con memoria.

Existen diversas líneas de investigación asociadas con estos modelos; por ejemplo, el modelo de consenso continuo multivaluado [5]; en este caso las entradas no están restringidas a valores binarios sino que entre los requerimientos de validez está que la decisión puede encontrarse entre los valores de dos procesos correctos. Por eso mismo se estudian dos tipos de sistemas: los sistemas de valor exacto ($EV$), donde el valor de decisión es el valor de un proceso correcto; y los sistemas de rango de valores ($RV$), donde el valor de decisión del sistema está en el rango de valores que aportan procesos correctos [3].

Otra línea de investigación muy interesante relacionada a los sistemas de consenso de larga vida, son aquellas donde la elección en particular del vértice inicial y en general de los vértices de un camino aleatorio en el hipercubo sean de acuerdo a alguna distribución probabilística distinta de la distribución uniforme [4] (como la que se ha tratado en este trabajo y en los artículos previos).

¿Qué pasaría si cambiamos la elección de los vértices mediante alguna función de distribución no uniforme? Uno esperaria que se modificara el valor de convergencia y de igual manera su alcance del estado estable, pero ¿De qué manera?

Los caminos aleatorios por el hipercubo estan asociados con los estados $E_k$ del modelo de urnas de Ehrenfest. La probabilidad con la cual se tendrá el siguiente estado depende unicamente del estado del sistema en ese momento; lo cual nos conduce a la idea del modelo de cadenas de Markov. Este es un modelo ergódico y en el artículo [4] se deja abierta la pregunta de ¿Cómo se comportaría el modelo si no lo fuera? ¿Habría estados inalcanzables por un camino aleatorio? ¿Se crearía una partición en los estados de la cadena? Y de ser así ¿El consenso se resolvería unicamente en alguna de esas particiones?

Uno de los comentarios hechos por mi asesor el Dr. Sergio Rajsbaum, es que los experimentos de simulación se realizan a fin de obtener información adicional que no te da originalmente una ecuación. Así que por esa línea también hay mucho que explotar.

En el Instituto de Astronomía de la UNAM, bajo la supervisión del Dr. Raúl Michel, pude constatar que áreas aparentemente tan distantes como lo son computación y astronomía, no lo son tanto. Los clusters computacionales sirven para llevar a cabo simulaciones sobre modelos físicos que requieren grandes cálculos numéricos. Por eso mismo, dentro de la rama de las simulaciones, y recordando que muchos de estos experimentos requerían de algunas horas de tiempo de CPU para llevar a cabo su tarea, surge la idea natural de poder paralelizar la ejecución de los programas y así ver la posibilidad de ahorrar tiempo y aprovechar los recursos de supercómputo haciendo las simulaciones en algún cluster.

Teniendo en cuenta la idea de paralelización, el Dr. Micho Durdevich propone modificar el modelamiento de procesos a fin de utilizar algoritmos cuánticos y ver (de manera teórica) como se comportarían si fuesen sistemas dinámicos con base en la lógica cuántica.

Esto se modificaría a fin de emular una computadora cuántica formada por tres eventos [40]: Primero preparar $N$ qubits en un estado inicial determinado; segundo aplicar una transformación unitaria $U$ a los $N$ qubits, la cual es el producto de compuertas cuánticas convencionales; y tercero, medir el valor de los qubits colapsándolos en los estados $\{0, 1\}$.

En computación cuántica un algoritmo siempre es probabilístico, de hecho el resultado siempre es una distribución de probabilidades de los diferentes resultados. El colapso de resultados en el tercer evento, no es más que escoger o hacer lectura de los resultados más posibles [39].

Recordemos que una simulación es un experimento de muestreo estadístico basado en computadora [32]. Por ello, si los resultados de un estudio de simulación tienen algún significado, apropiadas técnicas de estadística pueden ser usadas para analizar los experimentos [6]. Así pues y a guisa de comicidad diremos que ningún experimento es un completo fracaso, siempre puede servir de mal ejemplo.

También existen muchas aplicaciónes prácticas del consenso de larga vida, dando con esto la posibilidad de que aparezcan más ramas de investigación en los sistemas distribuidos y los algoritmos de consenso.

Por ejemplo: en los dispositivos que determinan si una compuerta se abre o se cierra, la energía es proporcional al número de transiciones entre encendido y apagado de su motor, por ello, un adecuado uso del consenso de larga vida, daría el posible alargamiento del tiempo de vida útil del dispositivo. Imaginemos también un conjunto de sensores replicados los cuales deben decidir si un avión autodirigido debe realizar algún cambio en su altitud; lo cual conlleva a que es preferible que realice el menor número de cambios en la altura durante su trayecto.

Estas son algunas ideas planteadas sobre posibles trabajos futuros, con la finalidad de que surja una representación nueva, útil y válida sobre los modelos distribuidos de consenso; y que den la pauta para obtener algún conocimiento nuevo.

Pero como toda idea, debe de pasar por un riguroso escrutinio e intenso trabajo de investigación a fin de que se puedan convertir en herramientas escenciales para el manejo y comprensión de la teoría de cómputo.

# Capítulo 7

# Ejercicios propuestos

Para reforzar al lector en los temas tratados, propongo una serie de ejercicios. Esperando sean de ayuda en la comprensión de los conceptos.

## 7.1. Para el capítulo 1

1. Para construir un sistema distribuido robusto, es necesario saber qué clases de fallos pueden ocurrir. Mencione tres tipos posibles de fallos en un sistema distribuido.

2. Considere un sistema distribuido y dos de sus procesos, digamos $p_1$ y $p_2$. Determine si $p_1$ puede distinguir entre las siguientes situaciones:

   - $p_2$ deja de operar.
   - El canal de comunicación entre $p_1$ y $p_2$ deja de operar.
   - $p_2$ está sobrecargado y su tiempo de respuesta es 100 veces mayor que lo normal.

3. ¿Es siempre crucial saber si el mensaje que envia un proceso llega a su destino en forma segura?
   Si tu respuesta es “Sí”, explica ¿Porqué? si tu respuesta es “No”, da un ejemplo apropiado.

4. Leslie Lamport presentó su algoritmo de Paxos ($\Pi\alpha\chi o\sigma$), y argumentó que este garantizaba la tolerancia a fallas de cualquier número de procesos participantes en el sistema y la continuación de sus tareas cuando más de la mitad de ellos estén trabajando de manera apropiada otra vez. En términos de la robustez ¿Qué implicaría para una base de datos distribuida aplicar el algoritmo de Paxos?

5. Decimos que el sistema distribuido es síncrono si cada ejecución consiste en una secuencia de rondas; las cuales son identificadas por enteros consecutivos: $1, 2, \ldots$ etc. Para los procesos, el número de la ronda actual aparece en la variable global $r$. Una ronda es hecha por las tres consecutivas fases:

   - *Envio*.- Fase en la cual cada proceso manda mensajes a todos los demás.
   - *Recepción*.- Fase en la que los procesos reciben mensajes. La propiedad fundamental del modelo síncrono es que un mensaje enviado por el proceso $p_i$ al proceso $p_j$ en la ronda $r$, es recibido por $p_j$ en la misma ronda $r$.
   - *Cómputo*.- Durante esta fase, cada proceso realiza cálculos o procesa los datos que trajeron los mensajes recibidos durante esa misma ronda.

   Suponemos también que el sistema de comunicación subyacente es libre de fallas; es decir, que no hay creación, alteración, pérdida o duplicado de mensajes. Suponemos que el modelo es síncrono, todos a todos, con comunicación confiable y donde a lo más $t$ procesadores pueden fallar ($t < n$). Las entradas válidas para los vectores de almacenamiento son únicamente $0$ y $1$.
   Entonces, dado el siguiente protocolo:

```
Function Decide()
(1)    val ← input
(2)    send val to all
(3)    wait until at least (n − t) messages have been received
(4)    let v[j] be the val recived from process j else ⊥
(5)    return h(v) = largest value in v[...]
```

   - Defina un conjunto de entrada $C$ tan grande como sea posible para el cual el protocolo es correcto.
   - Muestre que el protocolo cumple con los tres requerimientos esenciales para este $C$ propuesto.
   - ¿Se requiere asumir que $t < \frac{n}{2}$?
   - Defina otra función h′ diferente y repita lo anterior.

6. El siguiente algoritmo (vea [1]) permite el consenso entre procesos de los cuales a lo más un cuarto del total de ellos presentan un comportamiento arbitrario (bizantino):

```
Function Consensus()
(1)    V_i ← v_i
(2)    When r = 1,3,...,2t-1,2t+1 do
(3)       begin round
(4)          for each j
(5)             send(V_i) to p_j;
(6)          end for
(7)          let rec_i the bag of values received during r;
(8)          most_freq_i ← most frequent value in rec_i;
(9)          occ_nb_i ← occurrence number of most_freq_i;
(10)      end round
(11)    When r = 2,4,...,2t,2t+2 do
(12)       begin round
(13)          if (i = r/2) then
(14)             for each j
(15)                send (most_freq_i) to p_j;
(16)             end for
(17)          end if
(18)          if (v received from p_{r/2}) then
(19)             cord_val_i ← v;
(20)          else
(21)             cord_val_i ← v_i;
(22)          end if
(23)          if occ_nb_i > n/2 + t then
(24)             V_i ← most_freq_i;
(25)          else
(26)             V_i ← cord_val_i;
(27)          end if
(28)       end round
(29)    return(V_i);
```

Muestre que cumple con los tres requerimientos (Acuerdo, Terminación y Validez) de un buen algoritmo de consenso.

## 7.2. Para el capítulo 2

1. Basado en la demostración del lema 1 (página 11), ¿Qué se necesita para que la inestabilidad del sistema BP sea a lo más $3$?

2. Escribe la secuencia geodésica de entrada con sus respectivos valores de decisión, para un sistema $BP$ con $n = 2t + 1$ procesadores, donde el parámetro de tolerancia a fallos $t \geq 3$. Verifica la inestabilidad(BP).

3. Si permitieramos que las entradas en los vectores de tu ejemplo anterior cambiaran a lo más dos veces en lugar de una; ¿Es posible construir una secuencia que genere inestabilidad(BP)$= 4t + 2$?
   Si tu respuesta es “Sí”, explica qué condición fue necesaria; si tu respuesta es “No”, explica ¿Porqué?

## 7.3. Para el capítulo 3

1. Demuestre que la ecuación 4.1 (página 24) se obtiene a partir del teorema 2 (página 19).

2. Demuestre que la ecuación 4.2 (página 30) se obtiene a partir del teorema 3 (página 19).

## 7.4. Para el capítulo 4

1. Calcule el valor al cuál converge la inestabilidad promedio de un sistema distribuido de consenso de larga vida, con 100 procesadores participantes y que tolere que a lo más, la mitad de ellos fallen:

   - Guardando memoria del valor de decisión en ejecuciones anteriores.
   - Sin importar lo que hayan decidido en ejecuciones anteriores.

## 7.5. Para el capítulo 5

1. En el método de Welch (sección 5.3) muestre que para $\bar{Y}_i = \sum_{j=1}^{m} \frac{Y_{ji}}{m}$ para $i = 1, 2, \ldots, l$ tenemos que $E(\bar{Y}_i) = E(Y_i)$ y que $Var(\bar{Y}_i) = Var(Y_i)/m$

# Apéndice A

# Conceptos de probabilidad y estadística

Como apoyo a los temas tratados en esta tesis, añadimos varios conceptos fundamentales en el manejo de la Probabilidad y la Estadística. Para un manejo más detallado y amplio de estos conceptos, recomiendo al lector el libro de V. S. Koroliuk [6] el cual es la base de este apéndice.

## A.1. Álgebra de sucesos

Consideremos un conjunto $M$ de los sucesos que pueden observarse en cierto experimento estocástico. Podemos determinar algunas operaciones que se realizan con tales sucesos. Pero antes destaquemos dos sucesos especiales: el *suceso cierto* $U$ el cual es aquel que aparece en cada realización del experimento; y el *suceso imposible* $V$ el cual no puede ocurrir nunca, cualquiera que sea la realización del experimento.

Con todo suceso $A$ de $M$ asociamos el *suceso opuesto* $\overline{A}$, el cual consiste en que el suceso $A$ no ha ocurrido. Un suceso que está formado por la situación de que al menos uno de dos sucesos $A$ o $B$ se produzca, se denomina *unión* o *suma* de los sucesos $A$ y $B$; y se denota $(A \cup B)$ o $(A + B)$. Un suceso consistente en que $A$ y $B$ tienen lugar simultaneamente se denomina *intersección* o *producto* de los sucesos $A$ y $B$; y se denota $(A \cap B)$ o $(A \cdot B)$.

Dos sucesos $A$ y $B$ son incompatibles si $(A \cup B)$ es un suceso imposible. El suceso $\overline{A \cdot B}$ se denomina *diferencia de sucesos* y se designa $(A - B)$.

Los sucesos $E_1, E_2, \ldots, E_n$ forman un *grupo completo de sucesos* si son incompatibles dos a dos; y además $E_1 \cup E_2 \cup \ldots \cup E_n = U$. En otras palabras, de todos estos sucesos ocurre por lo menos uno.

Ahora bien, un conjunto no vacio $M$ de sucesos que satisface las condiciones:

- Si $A \in M$ entonces $\overline{A} \in M$;
- Si $A, B \in M$ entonces $(A \cup B) \in M$

Se le llama *álgebra de sucesos.*

Suele decirse que el suceso $A$ origina la aparición del suceso $B$ (se escribe $A \supset B$), si el suceso $B$ ocurre siempre que aparece $A$. El suceso $E$ se llama elemental, si para todo suceso $A$ del experimento aleatorio, tenemos que $E$ provoca o bien a $A$ o bien a $\overline{A}$. Un experimento aleatorio se denomina *finito* si se tiene un grupo completo de sucesos elementales.

En teoría de probabilidades se consideran sólo aquellos experimentos aleatorios en los cuales cualquier suceso representa una suma de todos los sucesos elementales que conducen a la aparición del suceso mencionado. Tal experimento aleatorio se describe por el conjunto de sucesos elementales $\Omega$ y por cierta clase

de sus subconjuntos $C_l$, llamados sucesos que pueden ocurrir en el experimento. Esta clase de subconjuntos debe satisfacer las siguientes condiciones:

1. $\Omega_i \in C_l$ ($\Omega_i$ es un suceso cierto que ocurre al transcurrir cualquier suceso elemental).
2. $C_l$ contiene el subconjunto vacio $\emptyset$ que se interpreta como un suceso imposible.
3. Si $A \in C_l$ entonces $(\Omega_i - A)$ también está en $C_l$. ($\Omega_i - A$ es el suceso opuesto a $A$).
4. Si $A, B \in C_l$ entonces $(A \cup B)$ y $(A \cap B)$ también pertenecen a $C_l$.

La clase de subconjuntos $C_l$ que satisface las condiciones 1-4 se denomina *álgebra de conjuntos*.

En caso de que $\Omega$ sea finito $C_l$ coincide con la clase de todos los subconjuntos de $\Omega$.

Como ejemplo, veamos una prueba en la cual se mide cierta magnitud $\xi$. Como sucesos elementales puede considerarse aquí los datos de valor fijo obtenidos. Resulta natural por eso identificar el conjunto de sucesos elementales con un conjunto de puntos en la recta. A priori sabemos que $\xi$ sólo puede tomar valores de cierto conjunto $M$, el cual consideramos como el conjunto de sucesos elementales. Y es natural la posibilidad de observar los sucesos $\{a \leq \xi < b\}$ donde $a, b$ son número arbitrarios y $a < b$. Cualesquiera sumas finitas de tales semiintervalos pueden considerarse como álgebras de sucesos ligados con el experimento.

## A.2. Espacios probabilísticos

En aquellos experimentos aleatorios en los cuales el álgebra de sucesos contiene un número infinito de sucesos, hemos de considerar sucesiones infinitas de sucesos como las operaciones que se realizan con ellas. Las operaciones más sencillas son la unión y la intersección de una sucesión infinita de sucesos. Si el álgebra de sucesos es tal que, con cada sucesión infinita de los sucesos $A_k$ (con $k = 1, 2, \ldots$), contiene tanto a los sucesos $\bigcap_k A_k$ como a los sucesos $\bigcup_k A_k$, entonces esta álgebra se llama *$\sigma$-álgebra*.

El suceso $\bigcap_k A_k$ consiste en que todos los sucesos $A_k$ ocurren simultaneamente; el suceso $\bigcup_k A_k$ consiste en que de la sucesión se realiza por lo menos uno de los sucesos $A_k$.

La sucesión se denomina monótona decreciente si $A_k \supset A_{k+1}$ y monótona creciente si $A_k \subset A_{k+1}$.

El suceso $\bigcap_k A_k$ se denomina límite de la sucesión decreciente; mientras que el suceso $\bigcup_k A_k$ se denomina límite de la sucesión creciente. El límite de la sucesión monótona (creciente o decreciente) $A_k$ se denota $\lim A_k$.

Un álgebra de sucesos $M$ será $\sigma$-álgebra si a la par con toda sucesión monótona también contiene el límite de esta.

Se denomina *espacio probabilístico* (o *campo de probabilidades*) a la totalidad de tres objetos: primero, un conjunto de sucesos elementales $\Omega$; segundo, la $\sigma$-álgebra $\mathcal{U}$ de subconjuntos de $\Omega$; y tercero, la probabilidad $P(A)$ definida para cada elemento $A \in \mathcal{U}$ para la cual $P(\Omega) = 1$.

El espacio probabilístico se denota por los objetos citados: $\{\Omega, \mathcal{U}, P\}$.

Llamaremos *medida* de la $\sigma$-álgebra a una función no negativa del conjunto $P(A)$ tal que para la cual

$$P\left(\bigcup_k A_k\right) = \sum P(A_k)$$

cualesquiera que sea la sucesión de los conjuntos $A_k$ de $\mathcal{U}$, disjuntos dos a dos.

Si $P(\Omega) = 1$, la medida se llama *normada*.

Daremos el nombre de *espacio medible* a un par de objetos: un conjunto $\Omega$ y una $\sigma$-álgebra $\mathcal{U}$ de subconjuntos de $\Omega$. Y la denotaremos $\{\Omega, \mathcal{U}\}$. Así pues el *espacio probabilístico* es un *espacio medible*, el cual dispone de una *medida normada*.

## A.3. La esperanza matemática

Supongamos que realizamos un experimento aleatorio y observamos una magnitud $\xi$ que puede tomar un número finito de valores $a_1, a_2, \ldots, a_N$ con las probabilidades correspondientes $p_1, p_2, \ldots, p_N$. Sean $x_1, x_2, \ldots, x_n$ las observaciones de la magnitud en $n$ realizaciones sucesivas del experimento, por lo tanto, el valor medio de las observaciones es:

$$\frac{1}{n}(x_1 + x_2 + \ldots + x_n) = \sum_{k=1}^{n} a_k p_k$$

La cual se denomina *media probabilística, valor esperado* o simplemente *esperanza*.

Si $\xi$ es una magnitud discreta arbitraria que toma los valores $a_k = 1, 2, \ldots$ con probabilidades $p_k$ respectivas, entonces la expresión

$$M\xi = \sum_{k=1}^{\infty} a_k p_k$$

recibe el nombre de *esperanza matemática* de dicha magnitud; siempre que la sumatoria converja.

A continuación damos algunas propiedades de la esperanza matemática de una magnitud discreta $\xi$:

- Si existe $M\xi_1$ y $M\xi_2$, entonces existirá $M(\xi_1 + \xi_2) = M\xi_1 + M\xi_2$.
- Para cualquier $c$, siempre que $M\xi$ exista, tenemos que $M(c\xi) = cM(\xi)$.
- Si $\mathbb{P}(\xi_1) = \mathbb{P}(\xi_2) = 1$ entonces $M\xi_1 = M\xi_2$.
- $M\xi \geq 0$ si $\xi \geq 0$ y $M\xi$ existe.
- Si $\mathbb{P}(\xi = c) = 1$ entonces $M\xi = c$.

## A.4. Probabilidad condicional

La probabilidad condicional de que ocurra un suceso $A$ con respecto a un suceso dado $B$ (denotada $\mathbb{P}(A \mid B)$), para el cual $\mathbb{P}(B) > 0$ se define como:

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

De aquí se deduce la fórmula de multiplicación de las probabilidades:

$$\mathbb{P}(A \cap B) = \mathbb{P}(A \mid B) \cdot \mathbb{P}(B) = \mathbb{P}(B \mid A) \cdot \mathbb{P}(A)$$

y la expresión $\mathbb{P}(A \mid B)$ en términos de $\mathbb{P}(B \mid A)$:

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(B \mid A) \cdot \mathbb{P}(A)}{\mathbb{P}(B)}$$

Proporcionamos además, la fórmula general de multiplicación de las probabilidades:

$$\mathbb{P}(\cap_{k=1}^{n} A_k) = \mathbb{P}(A_1) \cdot \mathbb{P}(A_2 \mid A_1) \cdots \mathbb{P}(A_n \mid \cap_{k=1}^{n-1} A_k)$$

Sean $E_1, E_2, \ldots, E_n$ un grupo completo de sucesos; entonces para todo suceso $A$

$$\mathbb{P}(A) = \sum_{k=1}^{n} \mathbb{P}(A \mid E_k) \cdot \mathbb{P}(E_k)$$

La fórmula de Bayes proporciona la expresión para las probabilidades condicionales de uno de los sucesos $E_k$ a condición de que $A$ ya tuvo lugar:

$$\mathbb{P}(E_k \mid A) = \frac{\mathbb{P}(A \mid E_k) \cdot \mathbb{P}(E_k)}{\sum_{i=1}^{n} \mathbb{P}(A \mid E_i) \cdot \mathbb{P}(E_i)}$$

Esta fórmula se llama también fórmula para la probabilidad de la hipótesis después de la prueba.

## A.5. Cadenas de Markov

Este concepto, introducido por el matemático ruso Andrei Markov [18] en un artículo publicado en 1907, presenta una forma de dependencia útil entre variables aleatorias.

Sea $\{\Omega, \mathcal{F}, P\}$ un espacio probabilístico. La aplicación medible $y : \{\Omega, \mathcal{F}\} \to \{X, \mathcal{B}\}$ donde $\{X, \mathcal{B}\}$ es cierto espacio medible, se llama elemento aleatorio en $\{X, \mathcal{B}\}$.

La sucesión $\{y_n, n = 0, 1, 2, \ldots\}$ de elementos aleatorios en el espacio medible $\{X, \mathcal{B}\}$ se denomina *Cadena de Markov,* si para cualesquiera $y \in \mathcal{B}$ y $n = 0, 1, 2, \ldots$ se verifica con probabilidad $1$

$$\mathbb{P}(y_n \mid y_0, y_1, \ldots, y_{n-1}) = \mathbb{P}(y_n \mid y_{n-1})$$

El espacio $\{X, \mathcal{B}\}$ se denomina *espacio fásico* de la cadena.

Toda sucesión de elementos (aleatorios o no aleatorios) $\{y_n, n = 0, 1, 2, \ldots\}$ del espacio $\{X, \mathcal{B}\}$ puede considerarse como el movimiento de cierto sistema (de un punto o de una partícula) en el espacio fásico: del estado inicial $y_0$, el sistema pasa al estado $y_1$ en el momento de tiempo $1$, luego pasa al estado $y_2$ en el momento de tiempo $2$, etc. De este modo el concepto de cadena de Markov destaca en la clase de sistemas llamados *sistemas sin efecto residual*. Para el caso determinista son aquellos sistemas en los cuales el estado en el momento de tiempo $n$ se determina unívocamente por el estado de dicho sistema en el momento de tiempo $(n-1)$, independientemente del comportamiento en el movimiento hasta ese momento.

A diferencia de los sistemas deterministas, los sistemas estocásticos sin efecto residual poseen la propiedad de que por el estado del sistema en el momento de tiempo $(n-1)$ se determina sólo la probabilidad con la cual el sistema se encuentra en este momento de tiempo en uno u otro conjunto de estados.

## A.6. Criterios para distinguir cadenas de Markov

Sea una sucesión $\{y_n, n = 0, 1, 2, \ldots\}$ de elementos aleatorios en el espacio medible $\{X, \mathcal{B}\}$ (El espacio probabilístico $\{\Omega, \mathcal{F}, P\}$ se concidera fijado). Designemos mediante $\mathcal{F}_n$ la $\sigma$-álgebra mínima de sucesos, respecto a la cual son medibles los elementos aleatorios $y_1, y_2, \ldots, y_n$ y mediante $\mathcal{F}^n$ la $\sigma$-álgebra mínima de sucesos, respecto a la cual son medibles los elementos aleatorios $y_n, y_{n+1}, \ldots$ En otras palabras, $\mathcal{F}_n$ la $\sigma$-álgebra de todos los sucesos relacionados con la evolución de la sucesión hasta el momento $n$ inclusive; en tanto que $\mathcal{F}^n$ la $\sigma$-álgebra de todos los sucesos relacionados con la evolución de la sucesión después del momento $n$, incluyendo el propio momento $n$.

La $\sigma$-álgebra $\mathcal{F}_n$ se genera por los sucesos del tipo $\{y_k \in \Gamma\}$ para todo $k = 0, 1, 2, \ldots, n, \Gamma \in \mathcal{B}$. Análogamente, la $\sigma$-álgebra $\mathcal{F}^n$ se genera por los sucesos del tipo $\{y_k \in \Gamma\}$ cuando $k \geq n, \Gamma \in \mathcal{B}$.

La definición de la cadena de Markov significa, de este modo, que para todos los $n = 0, 1, 2, \ldots$ y $\Gamma \in \mathcal{B}$ con probabilidad 1 tenemos

$$\mathbb{P}(y_{n+1} \in \Gamma \mid \mathcal{F}_n) = \mathbb{P}(y_{n+1} \in \Gamma \mid y_n)$$

Así pues, teniendo una sucesión de elementos aleatorios $\{y_n, n = 0, 1, 2, \ldots\}$ en el espacio medible $\{X, \mathcal{B}\}$, las siguientes afirmaciones son equivalentes:

- La sucesión $\{y_n, n = 0, 1, 2, \ldots\}$ es una cadena de Markov.
- Para cualesquiera $n = 0, 1, 2, \ldots$; $m = 1, 2, \ldots$; $\Gamma \in \mathcal{B}$ con probabilidad 1 tenemos

$$\mathbb{P}(y_{n+m} \in \Gamma \mid \mathcal{F}_n) = \mathbb{P}(y_{n+m} \in \Gamma \mid y_n)$$

- Para cualquiera $A \in \mathcal{F}_{n-1}$, $B \in \mathcal{F}^{n+1}$ $(n = 1, 2, \ldots)$ con probabilidad 1, tenemos

$$\mathbb{P}(A \cap B \mid y_n) = \mathbb{P}(A \mid y_n) \cdot \mathbb{P}(B \mid y_n)$$

- Para toda magnitud aleatoria acotada $\mathcal{F}^{n+1}-$medible $\eta$ con probabilidad 1, tenemos

$$M(\eta \mid \mathcal{F}_n) = M(\eta \mid y_n)$$

Si convenimos en considerar el momento de tiempo $n$ "presente", entonces $\mathcal{F}_{n-1}$ será el "pasado", mientras que $\mathcal{F}^{n+1}$ será el "futuro". La tercer afirmación dice que para la cadena de Markov con "el presente" conocido, "el pasado" y "el futuro" son condicionalmente independientes.

## A.7. Teorema del límite central

El teorema del límite central establece los fundamentos para estimar parámetros poblacionales y pruebas de hipótesis [31]. Debemos tener en cuenta el hecho de que el teorema del límite central implica dos distribuciones diferentes: la distribución de la población original y la distribución de las medias de muestra. Tengamos en cuenta que:

- Una *variable aleatoria* es una variable que tiene un sólo valor numérico determinado por el azar, para cada ejecución de la simulación.
- Una distribución de probabilidad es una fórmula que da la probabilidad para cada valor que puede aportar una variable aleatoria.
- La distribución muestral de la media es la distribución de probabilidad de medias de muestra donde todas y cada una de las muestras tienen el mismo tamaño.

Consideremos un ejemplo de ilustración muy sencillo: sea la población de los diez dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 los cuales se seleccionan con reemplazo.

1. *Variable aleatoria:* si realizamos ensayos que consisten en la selección aleatoria de un solo dígito y representamos con $x$ el valor en cada ensayo del dígito seleccionado, entonces $x$ es una variable aleatoria (su valor depende del azar).
2. *Distribución de probabilidad:* si los dígitos se seleccionan aleatoriamente, la probabilidad de cada dígito es $1/10$ que puede expresarse con la fórmula $P(x) = \frac{1}{10}$. Esta es una distribución de probabilidad, ya que describe la posibilidad de selección de cada valor de la variable aleatoria $x$.

3. *Distribución muestral:* supongamos que seleccionamos aleatoriamente distintas muestras posibles de tamaño $n = 4$ (Recuerde que es muestreo con reemplazo). En cada muestra $i$ calculemos la media de muestra $\bar{x}_i$ (que también es una variable aleatoria). La distribución de probabilidad de las medias muestrales es una distribución muestral.

El tercer inciso del ejemplo ilustra una distribución muestral específica de medias de muestra. Cuando el tamaño de la muestra aumenta, las medias de muestra correspondientes tienden a variar menos. El *Teorema del Límite Central* nos dice que si el tamaño de la muestra es lo suficientemente grande, la distribución de medias de muestra se aproxima a una distribución normal. Es curioso pues esto ocurre aun si la población original no se distribuye normalmente. Para una prueba formal remito al lector a la siguiente referencia [36].

1. Tenemos que:

   *a*) La variable aleatoria $x$ tiene una distribución (que puede ser normal o no) con media $\mu$ y desviación estándar $\sigma$.

   *b*) Todas las muestras aleatorias del mismo tamaño $n$ se seleccionan de la población original.

2. Concluimos que:

   *a*) La distribución de medias de muestra $\bar{x}$ se aproxima a una distribución normal conforme el tamaño de la muestra aumente.

   *b*) La media de todas las medias de muestra es la media poblacional $\mu$.

   *c*) La desviación estándar de todas las medias de muestra es $\frac{\sigma}{\sqrt{n}}$.

3. Tenemos las siguientes reglas prácticas de uso común:

   *a*) Si la población original no se distribuye normalmente, la siguiente es una guia común: para muestras de tamaño $n$ mayores a 30, la distribución de las medias de muestra puede aproximarse razonablemente bien a una distribución normal. La aproximación mejora conforme el tamaño de muestra $n$ se incrementa.

   *b*) Si la población original se distribuye normalmente, entonces las medias de muestra se distribuiran normalmente para cualquier tamaño de muestra $n$ (no sólo los mayores a 30).

El *Teorema del Límite Central* implica dos distribuciones diferentes: la distribución de la población original y la distribución de las medias de muestra. Utilizamos los símbolos $\mu$ y $\sigma$ para denotar la media y la desviación estándar de la población original. A continuación daremos las notaciones para la media y la desviación estándar de la distribución de medias de muestra.

Si tenemos una población con media $\mu$ y desviación estándar $\sigma$, y seleccionamos todas las muestras aleatorias de tamaño $n$, entonces la media de las medias de muestra se denota con $\mu_{\bar{x}}$ de modo que:

$$\mu_{\bar{x}} = \mu$$

También la desviación estándar de las medias de muestra se denota con $\sigma_{\bar{x}}$ de tal manera que:

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

$\sigma_{\bar{x}}$ suele denominarse el error estándar de la media.

## A.8. Intervalos de confianza

A continuación describiremos el procedimiento estadístico estandar para dada una proporción de muestra, estimar el valor de la proporción poblacional (tal y como es mostrada en [31]).

Un estimado puntual es un valor o punto que se utiliza para aproximar un parámetro poblacional.

Llamemos $p$ a la proporción de la población. $\hat{p} = \frac{x}{n}$ es la proporción muestral de $x$ éxitos en una muestra de tamaño $n$. Y $\hat{q} = 1 - \hat{p}$ es la proporción muestral de fracaso en una muestra de tamaño $n$.

Ahora bien, la proporción muestral $\hat{p}$ es el mejor estimado puntual en la proporción poblacional $p$. Ya que no está sesgado y porque es el más consistente de los estimadores que puede usarse. No está sesgado en el sentido de que la distribución de las proporciones muestrales tiende al centro para el valor de $p$; esto es, en las proporciones muestrales $\hat{p}$ no tiende sistemáticamente a subestimar ni a sobreestimar $p$.

La proporción muestral $\hat{p}$ es el estimador más consistente en el sentido de que su desviación estándar tiende a ser menor que la desviación estándar de cualquier otro estimador sin sesgo.

Aunque $\hat{p}$ es nuestro mejor estimado puntual de la proporción poblacional $p$, no tenemos indicación precisa de qué tan bueno es nuestro mejor estimado. Un estimado puntual tiene el grave defecto de no revelar nada acerca de qué tan bueno es, los estadísticos diseñaron ingeniosamente otro tipo de estimado. Este estimado se llama *Intervalo de Confianza* y consiste en un rango de valores en lugar de un solo valor.

Dicho de otro modo, un *Intervalo de Confianza* es una gama o intervalo de valores que se usan para estimar el valor real de un parámetro de la población. Así, un intervalo de confianza se asocia con un nivel de confianza (por ejemplo 95 %). El nivel de confianza nos da la tasa de sucesos del procedimiento.

El nivel de confianza suele expresarse como la probabilidad o área $1-\alpha$. El valor de $\alpha$ es el complemento del nivel de confianza. Para un nivel de confianza del 95 % (o 0.95), $\alpha = 0.05$

El *Nivel de Confianza* es la probabilidad $1 - \alpha$, que es la proporción de veces que el intervalo de confianza realmente contiene el parámetro de población, suponiendo que el proceso de estimación se repite un gran número de veces. El nivel de confianza también se conoce como *Grado de Confianza* o *Coeficiente de confianza*.

## A.9. Regresión lineal

Basados en la referencia [36], veamos los conceptos de curva de ajuste, regresión lineal y correlación.

Cuando en un experimento encontramos una relación entre variables, tratamos de expresar esa relación en forma matemática determinando una ecuación que conecte a esas variables.

Un primer paso es la colección de datos indicando los valores correspondientes a las variables. Por ejemplo, si $x$ y $y$ denotan la estatura y peso de un adulto; entonces una muestra de $n$ individuos resultaría en las estaturas $x_1, x_2, \ldots, x_n$ y los pesos correspondientes $y_1, y_2, \ldots, y_n$.

El paso siguiente es dibujar los puntos $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ en un sistema de coordenadas rectangulares. El conjunto resultante de puntos se llama *diagrama de dispersión.*

Del diagrama de dispersión es posible frecuentemente visualizar una curva que se aproxime a los datos. Puede ser que los datos se aproximen bien a una recta y en este caso decimos que existe una relación lineal entre las variables. Sin embargo, existen casos donde existe una relación entre las variables de forma que se aproximan a una curva no recta, por eso la llamamos relación no lineal.

El problema general de hallar ecuaciones de curvas de aproximación que se ajusten a conjuntos de datos se denomina curva de ajuste. En la práctica el tipo de ecuación se sugiere frecuentemente del diagrama de dispersión. Así pues, en el caso en que los datos se ajustan bien a una recta podríamos utilizar la ecuación

$$y = a + bx$$

Uno de los propósitos principales de la curva de ajuste es estimar la variable dependiente a partir de la variable independiente. El proceso de estimación se conoce como *regresión*.

Generalmente, más de una curva de un tipo dado parece ajustar un conjunto de datos. Para evitar el juicio individual en la construcción de rectas o curvas de aproximación, es necesario obtener una definición de "la mejor recta de ajuste".

Para motivar una definición considere los puntos de datos $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ y una curva de ajuste $C$. Así, habrá una cierta diferencia entre el valor dado $(x_1, y_1)$ y el valor correspondiente en la curva $C$. Denotemos esta diferencia como $d_1$; al que llamaremos *desviación* o *error* y notemos que puede ser positivo, negativo o cero. Análogamente para cada uno de los valores tendremos las diferencias $d_2, d_3, \ldots, d_n$.

Una medida de la "bondad del ajuste" de la curva $C$ a los datos es la cantidad $d_1^2 + d_2^2 + \ldots + d_n^2$. Si la suma es pequeña entonces el ajuste es bueno. Si es grande el ajuste es malo.

Por lo tanto tomamos la siguiente definición:

**La mejor curva de ajuste** es la que de todas las curvas de aproximación de un conjunto de puntos de datos, tenga la propiedad de que $d_1^2 + d_2^2 + \ldots + d_n^2$ es mínimo.

Una curva con esta propiedad se dice que se ajusta a los datos en el sentido de mínimos cuadrados y se llama curva de regresión de mínimos cuadrados.

Empleando esta definición es posible demostrar que la recta de mínimos cuadrados tiene la ecuación de recta $y = a + bx$ donde las constantes $a$ y $b$ se determinan solucionando simultaneamente el sistema de ecuaciones

$$\begin{aligned} \sum y &= an + b\sum x \\ \sum xy &= a\sum x + b\sum x^2 \end{aligned}$$

que se conocen como las *ecuaciones normales* para la recta de mínimos cuadrados. Observe que por brevedad hemos utilizado $\sum y$, $\sum xy$ en vez de $\sum_{j=1}^{n} y_j$, $\sum_{j=1}^{n} x_j y_j$.

Las ecuaciones normales se pueden recordar fácilmente; la primera ecuación se obtiene formalmente sumando ambos lados de la ecuación de recta $y = a + bx$, mientras que la segunda ecuación se obtiene primero multiplicando por $x$ ambos lados de la ecuación de recta y luego sumando. Logicamente esta no es una derivación de las ecuaciones normales sino solamente un medio para recordarlas.

Los valores de $a$ y $b$ que se obtienen a partir de las ecuaciones normales son:

$$a = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n\sum x^2 - (\sum x)^2}$$

$$b = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

El resultado para $b$ también puede escribirse como:

$$b = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$

Como es de uso común, la barra arriba de la variable aleatoria indica media, id est $\bar{x} = \frac{\sum x}{n}$.

Recordemos que aquí la variable $b$ es la pendiente de la recta.

### A.9.1. Recta de mínimos cuadrados en términos de varianzas y covarianzas muestrales

Las varianzas y covarianzas muestrales de $x$ y de $y$ están dadas por:

$$s_x^2 = \frac{\sum(x-\bar{x})^2}{n}$$

$$s_y^2 = \frac{\sum(y-\bar{y})^2}{n}$$

$$s_{xy} = \frac{\sum(x-\bar{x})(y-\bar{y})}{n}$$

Además definimos formalmente el *Coeficiente de Correlación Muestral* por:

$$r = \frac{s_{xy}}{s_x s_y}$$

Si denotamos por $y_{est}$ el valor estimado de $y$ para un valor dado de $x$, obtenido de la curva de regresión de $y$ sobre $x$, entonces una medida de la dispersión con respecto a la curva de regresión está suministrada por la cantidad:

$$s_{y\cdot x} = \sqrt{\frac{\sum(y-y_{est})^2}{n}}$$

que se llama *error típico de la estima de $y$ sobre $x$*. Donde $\sum(y-y_{est})^2 = \sum d_i^2$ y vemos que de todas las curvas de regresión posible, la curva de mínimos cuadrados tiene el más pequeño error típico de la estima.

Describamos ahora el significado del *Coeficiente de Correlación*. Resulta que tenemos de la definición de $s_{y\cdot x}$ y de $s_y$ (vea [36]) la recta de mínimos cuadrados en términos de la varianza y el coeficiente de correlación. Por ello obtenemos:

$$r^2 = 1 - \frac{\sum(y-y_{est})^2}{\sum(y-\bar{y})^2}$$

Y podemos probar que:

$$\sum(y-\bar{y})^2 = \sum(y-y_{est})^2 + \sum(y_{est}-\bar{y})^2$$

De esta última ecuación, el miembro izquierdo se denomina *Variación Total*. El primer sumando del miembro derecho se denomina *Variación No Explicada* y el segundo sumando se llama *Variación Explicada*. Esta terminología surge debido a que las desviaciones $y - y_{est}$ se comportan de una manera aleatoria o impredecible, en tanto que las desviaciones $y_{est} - \bar{y}$ se explican por la recta de regresión de mínimos cuadrados y así tienden a seguir un patrón definido.

Se deduce que:

$$r^2 \quad = \quad \frac{\sum(y_{est}-\bar{y})^2}{\sum(y-\bar{y})^2} \quad = \quad \frac{variación\ explicada}{variación\ total}$$

Por lo tanto, $r^2$ puede interpretarse como la fracción de la variación total que se explica por la recta de regresión de mínimos cuadrados. En otras palabras, $r$ mide qué tan bien la recta de regresión de mínimos cuadrados se ajusta a los datos muestrales. Si la variación total se explica por completo debido a la recta de regresión, es decir, si $r^2 = 1$ osea $r = \pm 1$, decimos que hay una correlación lineal perfecta. De otro modo, si la variación total no se puede explicar, entonces la variación explicada es cero y así $r = 0$.

En la práctica, la cantidad $r^2$ se denomina *Coeficiente de Determinación*; y se encuentra entre 0 y 1. Notemos que $r^2 \leq 1$, esto es que $-1 \leq r \leq 1$.

Así pues, tenemos que el coeficiente de correlación puede calcularse de la siguiente manera:

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2} \sqrt{\sum (y - \bar{y})^2}}$$

O bien, por las fórmulas equivalentes a la anterior, que se utilizan mucho en la práctica y que se denomina $r$ de Pearson:

$$r = \frac{n \sum xy - (\sum x)(\sum y)}{\sqrt{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]}}$$

y

$$r = \frac{\overline{xy} - \bar{x}\bar{y}}{\sqrt{(\overline{x^2} - \bar{x}^2)(\overline{y^2} - \bar{y}^2)}}$$

muy utilizados en computación.

El coeficiente de correlación lineal puede ser positivo o negativo. Si $r$ es positivo entonces la variable $y$ tiende a aumentar con $x$ (la pendiente de la recta de regresión es positiva). En tanto que si $r$ es negativa entonces $y$ está en proporción inversa con respecto a $x$ (y la pendiente de la recta de mínimos cuadrados es negativa).

Cualquiera de estas fórmulas para el coeficiente de correlación incluye solamente valores muestrales $x$ y $y$. En consecuencia da el mismo número para todas las formas de curvas de regresión, y con la excepción del caso de regresión lineal, no es útil como medida de ajuste para otro tipo de curvas.

Puesto que un coeficiente de correlación mide simplemente qué tan bien se ajusta una curva de regresión a los datos muestrales, es ilógico utilizar un coeficiente de correlación lineal donde los datos no son lineales. Sin embargo, supongamos que la aplicamos a datos no lineales y se obtiene un valor que es considerablemente menor que 1. Entonces la conclusión a extraerse no es que hay poca correlación sino que hay poca correlación lineal. En efecto, puede existir una gran correlación no lineal.

# Apéndice B

# Programación y graficación con MATLAB

En este apéndice especificamos algunos detalles técnicos del ambiente de simulación. Los programas se realizaron en lenguaje MATLAB bajo el ambiente *Windows XP* y *Mac Os X* con arquitecturas *Intel* Centrino Duo e *Intel* Xeon de doble núcleo respectivamente. La referencia [12] fue utilizada como consulta para la programación y la graficación de los experimentos.

## B.1. El laboratorio matricial

El software MATLAB es un programa de análisis numérico que incluye su propio lenguaje de programación. Fue desarrollado por la empresa *MathWorks* y desde 1984 ha sacado continuamente al mercado diversas versiones mejoradas (al momento se encuentra la versión 7 la cual está disponible para plataformas Unix, Windows y Mac).

MATLAB es un paquete de cálculo numérico orientado a matrices y vectores, de ahí su nombre, que es la abreviación de *Matrix Laboratory* (Laboratorio de Matrices). Además se pueden ampliar sus capacidades con herramientas (Toolboxes) orientadas al proceso digital de señales, manejo y adquisición de datos, inteligencia artificial, lógica, estadística, y también cuenta con paquetes como SIMULINK que sirve para realizar simulaciones de sistemas.

Su lenguaje de programación fue creado en 1970 junto con su propio compilador, y ha tenido algunas mejoras a lo largo del tiempo. De hecho la primera versión fue pensada con la idea de emplear unas subrutinas de álgebra lineal y análisis numérico escritas en Fortran sin necesidad de escribir programas en este lenguaje.

Debido a sus características de cálculo numérico y a sus funciones matemáticas, es un software muy utilizado en universidades y centros de investigación e ingeniería. Esta herramienta es de gran utilidad para la implementación de algoritmos que a través de números y reglas matemáticas simples pueden simular procesos matemáticos más complejos o del mundo real.

Estos algorítmos nos pueden llevar a una solución aproximada de un problema mediante un número finito de pasos que pueden ejecutarse de manera lógica.

Existe un programa libre equivalente a MATLAB para realizar cálculos numéricos llamado OCTAVE, el cual es parte del proyecto GNU. Y entre sus varias características que comparten se puede destacar que ambos ofrecen un interprete que permite ejecutar ordenes de modo interactivo. Otros programas alternativos para el manejo de álgebra computacional los cuales facilitan el cálculo simbólico en computadora son MATHEMATICA, MAXIMA o MAPLE.

## B.2. Descripción de los programas

A continuación haremos una breve descripción de los programas de simulación sin enfocarnos demasiado en el lenguaje, sino más bien en pseudocódigo. Recordemos que estas simulaciones están basadas en los modelos sin memoria $D_0$ (sección 4.1 página 24) y con memoria $D_1$ (sección 4.2 página 30).

```
Funcion Simulacion Del Modelo Sin Memoria (n, t, long, Ejc)

     /*
        Primer parametro (n): numero de procesos participantes en el sistema.
        Segundo parametro (t): parametro de tolerancia a fallos.
        Tercer parametro (long): longitud del camino aleatorio.
        Cuarto parametro (Ejc): numero de replicas del experimento.
     */

For ejecucion = 1 to Ejc

        Generamos un numero aleatorio A tal que (0 < A < n);

        Actualizamos el numero de pasos hechos por el vertice A;

        For pasos = 1 to long

                El siguiente movimiento (izquierda/derecha) se obtiene con
                    probabilidad de acuerdo a la posicion actual A;

                If movimiento es a la derecha
                        Actualizamos el contador del vertice A+1;
                End If

                If movimiento es a la izquierda
                        Actualizamos el contador del vertice A-1;
                End If

                Contabilizamos la razon de transitos por el vertice t
                    entre el numero de pasos en ese momento;

        End For

        Hacemos una normalizacion en esta ejecucion, del numero de pasos realizados
            por cada vertice entre el total de pasos realizados;

End For

Finalizamos graficando la distribucion de transitos realizados por cada vertice y
    la razon de de transitos hechos por el vertice t a lo largo de la ejecucion;
```

```
Funcion Simulacion Del Modelo Con Memoria (n, t, long, Ejc)

     /*
        Primer parametro (n): numero de procesos participantes en el sistema.
        Segundo parametro (t): parametro de tolerancia a fallos.
        Tercer parametro (long): longitud del camino aleatorio.
        Cuarto parametro (Ejc): numero de replicas del experimento.
     */

For ejecucion = 1 to Ejc

        Elegimos aleatoriamente uno de los dos estados posibles (0 / 1);

        Generamos un numero aleatorio A tal que (0 < A < n);

        Actualizamos el numero de pasos hechos por el vertice A;

        For pasos = 1 to long

                El siguiente movimiento (izquierda/derecha) se obtiene con
                    probabilidad de acuerdo a la posicion actual A;

                If movimiento es a la derecha
                        Actualizamos el contador del vertice A+1;
                End If

                If movimiento es a la izquierda
                        Actualizamos el contador del vertice A-1;
                End If

                If vertice actual es t y el estado es 1
                        estado cambia a 0 y contabilizamos la razon de transitos por
                            el vertice t entre el numero de pasos en ese momento;
                End If

                If vertice actual es (n-t) y el estado es 0
                        estado cambia a 1 y contabilizamos la razon de transitos por
                            el vertice (n-t) entre el numero de pasos en ese momento;
                End If

        End For

        Hacemos una normalizacion en esta ejecucion, del numero de pasos realizados
            por cada vertice entre el total de pasos realizados;

End For

Finalizamos graficando la distribucion de transitos realizados por cada vertice y
    la razon de de transitos hechos por el vertice t y el vertice (n-t)
    a lo largo de la ejecucion;
```

```
Funcion Comparacones (N, long, Ejc, Ventana, tipo)

     /*
        Primer parametro (N): lista de valores de n a comparar.
        Segundo parametro (long): longitud del camino aleatorio.
        Tercer parametro (Ejc): numero de replicas del experimento.
        Cuarto parametro (Ventana): tamanio de ventana utilizada en
                                        el metodo de Welch.
        Quinto parametro (tipo): si la simulacion se hara en el modelo
                                    con memoria o sin memoria.
     */

For cada valor n de la lista en N

        If tipo es sin memoria
                Simulacion Del Modelo Sin Memoria (n, n/2, long, Ejc);
        Caso contrario
        If tipo es con memoria
                Simulacion Del Modelo Con Memoria (n, n/2, long, Ejc);
        End If

        Aplicamos el metodo de Welch al resultado dado por la simulacion
            a fin de reducir la autocorrelacion en los datos;

        Posteriormente aplicamos el Proceso de Control Estadistico
            a fin de encontrar los intervalos de confianza del estado estable;

        Una vez hecho esto, procedemos a encontrar el punto de friccion para
            esta simulacion en particular;

End For

Aplicamos las ecuaciones de regresion lineal a fin de obtener la
    recta de minimos cuadrados que mejor se ajuste a los valores
    de N y sus respectivos puntos de friccion;

Calculamos el coeficiente de correlacion para determinar que tan bueno
    es el ajuste;

Graficamos los resultados obtenidos;
```

## B.3. Estructuras de datos

Como hemos visto, MATLAB es un paquete de cálculo numérico orientado a matrices y vectores. Por ello desde un inicio debemos pensar en hacer uso de las estructuras de datos en términos de vectores y matrices a fin de que sea más rápido y efectivo.

A final de cuentas la computadora va a manipular la información, así que el modo en que esta se organice es muy importante para el óptimo desempeño de nuestro programa. En general, tanto vectores como matrices son arreglos en localidades de memoria que están vinculadas y forman tipos de datos abstractos.

Las principales estructuras de datos utilizadas en los programas son: los vectores donde almacenamos la razón de tránsitos por el vértice $t$ o el vértice $n - t$; los vectores que contabilizan el número de tránsitos hechos por cada vértice; las matrices que sirven para graficar la información en $3D$ de diversas ejecuciones variando el parámetro de tolerancia a fallos $t$.

Estos TDA's son los que almacenan los resultados de los programas de simulación y son pasados como parámetros a procedimientos que manipulan su información y así obtener resultados interpretables desde el punto de vista de los experimentos.

## B.4. Detalles en la graficación

Como hemos visto, utilizamos una matriz o arreglo bidimensional para guardar los resultados de varias ejecuciones y con diversos valores de $t$. Si pensamos detenidamente un momento, esto lo podemos interpretar como una función de dos variables. Existen tanto en MATLAB como en OCTAVE las funciones diseñadas específicamente para evaluar y graficar funciones de dos variables. Primero es necesario evaluar la función de dos variables y después graficarla tridimensionalmente.

Para evaluar una función $f(x, y)$ de dos variables, primero tenemos que definir una retícula bidimensional en el plano $x, y$. A continuación evaluar la función en los puntos de la retícula para determinar puntos en la superficie tridimensional.

Definimos tanto en MATLAB como en OCTAVE una retícula bidimensional en el plano $x, y$ usando dos matrices. Una matriz contiene las coordenadas $x$ de todos los puntos de la retícula, y la otra contiene las coordenadas $y$ de todos los puntos de la retícula.

Veamos un ejemplo: supongamos que queremos definir una retícula en la que la coordenada $x$ varie de -2 a 2 en incrementos de 1; y la coordenada $y$ varie de -1 a 2 también en incrementos de 1. La matriz correspondiente de valores $x$ en la retícula es la siguiente:

$$\begin{matrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{matrix}$$

Y la matriz correspondiente de valores $y$ en la retícula es la siguiente:

$$\begin{matrix} -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \end{matrix}$$

Así, el punto de la esquina superior izquierda de la retícula tiene coordenadas $(-2, -1)$. Y el punto de la esquina inferior derecha de la retícula tiene coordenadas $(2, 2)$.

La función `meshgrid` genera las dos matrices que definen la retícula subyacente para una función bidimensional. Utilizada de la siguiente forma:

```
[x_grid, y_grid] = meshgrid(x,y)
```

Genera dos matrices de tamaño $n \times m$, con base en los valores de los vectores $x$ y $y$ que contienen $m$ valores y $n$ valores respectivamente. La matriz `x_grid` contiene los valores de $x$ repetidos en cada fila, y la matriz `y_grid` contiene los valores de $y$ repetidos en cada columna.

Así pues, para generar las dos matrices descritas en el ejemplo anterior, podemos utilizar las siguientes instrucciones:

```
x = -2:2;
y = -1:2;
[x_grid, y_grid] = meshgrid(x,y);
```

Una vez definidas las matrices de la retícula subyacente, podemos calcular los valores correspondientes de la función. Por ejemplo, supongamos que se desea evaluar la siguiente función para la retícula que acabamos de definir:

$$z = f(x, y) = \frac{1}{1 + x^2 + y^2}$$

Los valores correspondientes de la función se pueden calcular y almacenar en una matriz $z$ de cuatro filas y cinco columnas con estas instrucciones:

```
z = 1./(1 + x_grid.^2 + y_grid.^2);
```

Puesto que las operaciones son elemento por elemento, el valor de la matriz $z$ que tiene subíndices $(1, 1)$ se calcula usando los valores que están en `x_grid(1,1)` y en `y_grid(1,1)`, y así sucesivamente. Observe que no se requieren ciclos para calcular todos los valores de $z$. Un error común al calcular los valores de una función de dos variables es usar los vectores $x$ y $y$, en lugar de los valores de la retícula subyacente que están en `x_grid` y en `y_grid`.

Una vez hecho esto entonces procedemos a aplicar alguna de las varias formas que existen para graficar una superficie tridimensional, como pueden ser:

```
plot3(x_grid, y_grid , z);
mesh(x_grid, y_grid , z);
surf(x_grid, y_grid , z);
```

las cuales representan datos en superficie de diferente forma; por ejemplo `plot3` genera una superficie continua, `mesh` genera una superficie de cuadrícula abierta, y `surf` genera una gráfica de cuadrícula sombreada. Todo sobre los datos definidos en la matriz $z$.

# Bibliografía


[1] Michel Raynal. “Consensus in Synchronous Systems: A Concise Guided Tour”. Proc. 9th IEEE Pacific Rim Int. Symposium on Dependable Computing. (PRDC’02), Tsukuba (Japan), IEEE Computer Press, pp. 221-228, December 2002.

[2] Shlomi Dolev and Sergio Rajsbaum. “Stability of Long-lived Consensus”. Journal of Computer and System Sciences, Vol. 67, Issue 1, pp. 26-45, August 2003. A preliminary version appeared in Proc. of the $19^{th}$ annual ACM symposium on Principles of Distributing Computing. (PODC) 2000, pp. 309-318, 2000.

[3] Achour Mostefaoui, Michel Raynal and Frédéric Tronel. “From Binary Consensus to Multivalued Consensus in Asynchronous Message-Passing Systems”. Information Processing Letters. 73: 207-212 March 2000.

[4] Florent Becker, Sergio Rajsbaum, Ivan Rapaport and Eric Rémila. “Average Stability of Binary Long-Lived Consensus”. Manuscrito Marzo 2007. Partially supported by Programs Conicyt “Anillo de Redes” and Fondap on Applied Mathematics.

[5] Lior Davidovitch, Shlomi Dolev and Sergio Rajsbaum. “Stability of Multi-Valued Continuous Consensus”. To appear in SIAM Journal on Computing. Preliminary Proceedings of the Workshop on Geometry and Topology in Concurrency and Distributed Computing (GETCO 2004). Amsterdam, The Netherlands. October 4, 2004. pp. 21-24. P. Cousot, L. Fajstrup, E. Goubault, M. Herlihy, M. RauBen, V. Sassone (Eds.) BRICS Notes Series NS-04-2, September 2004, ISSN 0909-3206.

[6] V. S. Koroliuk. “Manual de la Teoría de Probabilidades y Estadística Matemática”. Academia de Ciencias de la República Socialista Soviética de Ucrania. Editorial MIR. Moscú. 1981.

[7] Idit Keidar and Sergio Rajsbaum. “A Simple Proof of the Uniform Consensus Synchronous Lower Bound”. Information Processing Letters. 85: 47-52, 2003.

[8] Abraham Silberschatz et. al. “Sistemas Operativos”. ISBN: 968-18-6168-x Limusa Wiley. México. 2004.

[9] Ralph P. Grimaldi. “Matemáticas Discreta y Combinatoria”. ISBN: 968-444-324-2 Prentice Hall. México. 1998.

[10] Rachid Guerraoui and Michel Raynal. “The Alpha of Indulgent Consensus”. The Computer Journal. Published by Oxford University Press on behalf of The British Computer Society. August 3, 2006.

[11] Fischer M. J., Lynch N. and Paterson M. S. “Impossibility of Distributed Consensus with One Faulty Process”. Journal of the ACM. 32(2):374-382. April, 1985.

[12] Delores M. Etter. “Solución de Problemas de Ingeniería con MATLAB”. ISBN: 9701701119 Pearson Education. México. 1998.

[13] Leslie Lamport. “The Part-Time Parliament”. ACM Transactions on Computer Systems. 16(2):133-169. 1998.

[14] Leslie Lamport. “Paxos Made Simple”. ACM Sigact News, Distributed Computing Column. 32(4):34-58. 2001.

[15] Enciclopedia Libre (8 de Mayo del 2007). Disponible en: http://es.wikipedia.org/wiki/Hipercubo

[16] Hamdy A. Taha. “Investigación de Operaciones”. ISBN: 970-26-0498-2 Pearson Education. México. 2004.

[17] Richard L. Burden and J. Douglas Faires. “Numerical Analysis”. ISBN: 0-53491-585-X PWS-KENT Publishing Company. Boston Massachusetts.

[18] Enciclopedia Libre (8 de Mayo del 2007). Disponible en: http://es.wikipedia.org/wiki/Andrey_Markov

[19] R. De Prisco, B. Lampson and N. Lynch. “Revisiting the Paxos Algorithm”. In M. Mavronicolas and P. Tsigas (eds.), Proc. of 11th Int. Workshop on Distributed Algorithms (WDAG’97). Saarbrucken, Germany. September 1997. Vol. 1320 of Lecture Notes in Computer Science, 111-125. Springer-Verlag 1997.

[20] Enciclopedia Libre (11 de Mayo del 2007). Disponible en: http://en.wikipedia.org/wiki/Leslie_Lamport

[21] H. Kopetz and P. Veríssimo. “Real Time and Dependability Concepts”. chapter 16, pp. 411-446. Sape Mullender (ed.), Distributed Systems. ACM Press. 1993.

[22] Enciclopedia Libre (11 de Mayo del 2007). Disponible en: http://en.wikipedia.org/wiki/Paxos_algorithm

[23] Robert Shostak, Leslie Lamport and Marshall Pease. “The Bizantine Generals Problem”. ACM Transactions on Programming Languages and Systems. 4(3):382-401. 1982.

[24] Leslie Lamport. “Time, Clocks and the Ordering of Events in a Distributed System”. Comunications of the ACM. 21(7):558-565. 1978.

[25] K. Marzullo. “Tolerating failures of continuous-valued sensors”. ACM Trans. on Comp. Systems. 8(4):284-304, Nov. 1990.

[26] Chandy K. Mani and Leslie Lamport. “Distributed Snapshots: Determining Global States of a Distributed System”. ACM Transactions on Computer Systems. 3(1):63-75. 1985.

[27] Enciclopedia Libre (15 de Mayo del 2007). Disponible en: http://en.wikipedia.org/wiki/Byzantine_failure

[28] J. Von Newmann. “Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components”. pp. 43-98. in Automata Studies, C.E. Shannon and J. McMarthy (Eds.). Princeton Univ. Press. 1956.

[29] Leslie Lamport (15 de Mayo del 2007). Disponible en: http://research.microsoft.com/users/lamport/pubs/pubs.html

[30] Gunnar Blom, Lars Holst and Dennis Sandell. “Problems and Snapshots from the World of Probability”. ISBN: 0-387-94161-4 Springer-Verlag. New York. 1994.

[31] Mario F. Triola. “Probabilidad y Estadística”. ISBN: 970-26-0575-X Addison Wesley. México. 2004.

[32] Averill M. Law and W. David Kelton. “Simulation Modeling and Analysis”. Second Edition. ISBN: 0-07-100803-9 McGraw-Hill International Editions. Singapore 1991.

[33] Stewart Robinson. “A Statistical Process Control Approach for Estimating the Warm-Up Period”. Proceedings of the 2002 Winter Simulation Conference. Vol. 1. Dec. 2002. pp. 439-446.

[34] H. Attiya and J. Welch. “Distributed Computing: Fundamentals, Simulations and Advanced Topics”. McGraw-Hill, 1998.

[35] F. B. Schmuck and F. Cristian. “Continuous clock amortization need not affect the precision of a clock synchronization algorithm”. in Proceedings of the 9th Annual ACM Symp. on Principles of Distributed Comp. pp. 133-143, Quebec City, Quebec, Canada. 22-24 August 1990.

[36] Murray R. Spiegel. “Teoría y Problemas de Probabilidad y Estadística”. Serie Schaum. McGraw-Hill. ISBN: 968-422-923-2

[37] P. D. Welch. “The Statistical Analysis of Simulation Results”. The Computer Performance Modeling Handbook. Ed. S. Lavenberg, 268-328. New York. Academic Press. 1983.

[38] G. S. Fishman. “Grouping Observations in Digital Simulation”. Management Science. 24:510-521. 1978.

[39] Michael A. Nielse and Isaac L. Chuang. “Quantum Computation and Quantum Information”. ISBN 0521635039. Cambridge University Press. September, 2000.

[40] Computación Cuántica (13 de Noviembre del 2007). Disponible en: http://www.sargue.net/fitxers/quantum-es.pdf

# Convergence and Reach of Steady-State on Long-Lived Consensus Model

Jorge Figueroa Canales.
Maestría en Ciencia e Ingeniería de la Computación.
Universidad Nacional Autónoma de México.
jfigueroac@uxmcc2.iimas.unam.mx
www.mcc.unam.mx/∼jfigueroac

## 1. Fast Abstract of a Work in Progress.

The consensus problem in distributed systems, is an extensively studied subject in computer science. The continuous consensus system, also named long-lived consensus system, is a common variant of this model. This system executes the consensus algorithm in a subsequent way according to changes in contribution values of the involved processes [1].

This particular model has been studied under a wide variety of assumptions in order to get a more general representation. This assumptions are: The system is synchronous, it has reliable communication channels and it is restricted to binary input values. The last assumption is not a constraint, since multi-valued consensus can be restricted to binary consensus (multi-valued consensus stability is studied in [3]). Therefore, each one of the $n$ processors proposes its own binary input value, and after some communications among them, they must decide one binary exit value.

Stability notion is central to this kind of systems, since it reflects the system sensitiveness to changes in decision, from one consensus algorithm invocation to the next one according to the inputs. These inputs are represented as binary $n-$dimensional vectors; id est elements of the set $\mathbb{Z}_2^n$.

Among the scenes used for these systems, there are those that keep in memory the value from one invocation of consensus to the next invocation and also memoryless systems (that is, systems without memory).

Florent Becker, Sergio Rajsbaum, Ivan Rapaport and Eric Rémila [2] have proven that these systems converge as much in the model with memory like without memory. Besides they prove that the greater instability reached about these systems is obtained when the fault-tolerance parameter $t$ is equal to half of the number of participant processes $n$. One of the immediate goals of this work, is to make some simulations done over the distributed models execution, in order to obtain information of their behavior.

We should bear in mind that the main idea in the study of stability in distributed long-lived consensus systems, is that the exit value can change throughout several and repeated executions of the consensus algorithm. Because of this the use of long-lived consensus algorithms is preferable when decision values are changed a few number of times. Let us imagine that we have a set of replicated sensors, which must decide if a device (v. gr. an airplane remotely directed) must make some change in its operation (to raise or to lower its altitude). In this sense we are interested in studying how stable a consensus system can be.

Notice that if there is a one-shot consensus algorithm for the particular communication model, it could be used to solve the long-lived consensus version, by simply invoking a new version of the algorithm in each phase. However, this kind of solution produces a long-lived consensus system with no stability guarantees. In other words, we are interested in minimizing the fluctuations of the outputs from phase to phase as much as possible. Of course, if the input vector changes from the $0^n$ in one phase, to the $1^n$ vector in the next phase, then the decision will have to change from $0$ to $1$. But if the input vector changes by just a few bits, it might be possible to avoid changing the decision.

One of the aims when making the simulations is to show the convergence predicted by the equations [2]; but it is not the only goal, since in addition we can obtain data of the inestability behavior throughout executions and also to know how fast it tends to the announced value. (Equation does not give us this information).

Therefore, the main contribution of this work, is a study of how fast the models converge towards the value of stability. To achieve this, statistical techniques used by researchers in the study of simulations and computacional modeling are applied (see [5]). Among these techniques we find Welch's procedure and the Statistical Process Control (see [6]).

Let $Y_1, Y_2, \ldots$ the outputs of a stochastic process like our simulation program; and let $F_i(y \mid I) = P(Y_i \leq y \mid I)$ for $i = 1, 2, \ldots$ where $y$ is a real number and $I$ represents the initial conditions used to start the simulation at time 0. We call $F_i(y \mid I)$ the transient distribution of the output process at discrete time $i$ for initial conditions $I$.

For fixed $y$ and $I$, the probabilities $F_1(y \mid I), F_2(y \mid I), \ldots$ are just a sequence of numbers. If $F_i(y \mid I) \to F(y)$ as $i \to \infty$ for all $y$ and for any initial conditions $I$, then $F(y)$ is called the steady-state distribution of the process $Y_1, Y_2, \ldots$ Strictly speaking, the steady-state distribution $F(y)$ is only obtained in the limit as $i \to \infty$.

In practice, however, there will often be a finite time index, say $k$, such that the distribution from this point on will be approximately the same as following each other. It's by that we say the steady state start at time $k$; which gives us a reference parameter to know which of the simulations, with certain enter values $(n, t)$, reach first its steady-state.

Note that steady state does not mean that the random variables $Y_k, Y_{k+1}, Y_{k+2} \ldots$ will all take on the same value in a particular simulation run; rather, it means that they will all have approximately the same distribution.

For our experiments we used the mean like test statistic; id est we made $m$ replications of the simulations, each of length $l$. At this way we will have that $Y_{ji}$ be the $i$th observation from the $j$th replication (with $i = 1, 2, \ldots, l$ and $j = 1, 2, \ldots, m$).

Let $\bar{Y}_i = \sum_{j=1}^{m} \frac{Y_{ji}}{m}$ for $i = 1, 2, \ldots, l$

The averaged process $\bar{Y}_1, \bar{Y}_2, \ldots$ has means $E(\bar{Y}_i) = E(Y_i)$ and variances $Var(\bar{Y}_i) = Var(Y_i)/m$. Thus, the averaged process has the same transient mean curve as the original process, but its plot has only $\frac{1}{m}$th the variance.

More even, to smooth out the high-frequency oscillations in $\bar{Y}_1, \bar{Y}_2, \ldots$ (but leave the low-frequency oscillations or long-run trend of interest), we further define the moving average $\overline{Y_i(w)}$ where $w$ is called the window and it is a positive integer such that $w \leq \lfloor l/2 \rfloor$ as follows:

$$\overline{Y_i(w)} = \begin{cases} \frac{\sum_{s=-w}^{w} \bar{Y}_{i+s}}{2w+1} & \text{if i = w+1, w+2, \ldots, l-w} \\ \\ \frac{\sum_{s=-(i-1)}^{i+1} \bar{Y}_{i+s}}{2i-1} & \text{if i = 1, 2, \ldots, w} \end{cases}$$

This procedure due to P. D. Welch, serves as a filter in order to eliminate "noise" in our stochastic process and thus to obtain a confidence interval in which we will find the values of our stochastics variables in a steady-state means.

There is another way: The statistical process control [6] is also applied in order to determine first point in which our system is "in-control". Id est, simulation's warming-up period ends, once stochastic variables output values reach a steady-state means.

We will apply these techniques in order to show how fast the models converge when input parameter $n$ changes.

## Referencias


[1] Shlomi Dolev and Sergio Rajsbaum. "Stability of Long-lived Consensus". Journal of Computer and System Sciences, Vol. 67, Issue 1, pp. 26-45, August 2003. A preliminary version appeared in Proc. of the $19^{th}$ annual ACM symposium on Principles of Distributing Computing. (PODC) 2000, pp. 309-318, 2000.

[2] Florent Becker, Sergio Rajsbaum, Ivan Rapaport and Eric Rémila. "Average Stability of Binary Long-Lived Consensus". Manuscrito Marzo 2007. Partially supported by Programs Conicyt "Anillo de Redes" and Fondap on Applied Mathematics.

[3] Lior Davidovitch, Shlomi Dolev and Sergio Rajsbaum. "Stability of Multi-Valued Continuous Consensus". To appear in SIAM Journal on Computing. Preliminary Proceedings of the Workshop on Geometry and Topology in Concurrency and Distributed Computing (GETCO 2004). Amsterdam, The Netherlands. October 4, 2004. pp. 21-24. P. Cousot, L. Fajstrup, E. Goubault, M. Herlihy, M. RauBen, V. Sassone (Eds.) BRICS Notes Series NS-04-2, September 2004, ISSN 0909-3206.

[4] V. S. Koroliuk. "Manual de la Teoría de Probabilidades y Estadística Matemática". Academia de Ciencias de la República Socialista Soviética de Ucrania. Editorial MIR. Moscú. 1981.

[5] Averill M. Law and W. David Kelton. "Simulation Modeling and Analysis". Second Edition. ISBN: 0-07-100803-9 McGraw-Hill International Editions. Singapore 1991.

[6] Stewart Robinson. "A Statistical Process Control Approach for Estimating the Warm-Up Period". Proceedings of the 2002 Winter Simulation Conference. Vol. 1. Dec. 2002. pp. 439-446.