INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INNOVACIÓN Y DESARROLLO TECNOLÓGICO EN CÓMPUTO

CONSTRUCCIÓN Y PROGRAMACIÓN DE UN ALGORITMO PARALELO PARA DETERMINAR EL EXPONENTE DE HURST

TESIS QUE PARA OBTENER EL GRADO DE MAESTRÍA EN TECNOLOGÍA DE CÓMPUTO

PRESENTA

Lic. Adauto Israel Ortiz Romero

DIRECTORES

M. en C. Jesús Antonio Álvarez Cedillo
Dr. Oswaldo Morales Matamoros

MÉXICO D. F. JUNIO 2008

SIP-14

INSTITUTO POLITECNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISION DE TESIS

En la Ciudad de México, D.F. siendo las 12:00 horas del día 18 del mes de junio de 2008 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del CIDETEC para examinar la tesis de grado titulada:

CONSTRUCCIÓN Y PROGRAMACIÓN DE UN ALGORITMO PARALELO PARA DETERMINAR EL EXPONENTE DE HURST

Presentada por el alumno:

ORTIZ	ROMERO	ADAUTO ISRAEL
Apellido paterno	materno	nombre(s)

Con registro: B 0 5 0 9 1 7

aspirante al grado de:

MAESTRÍA EN TECNOLOGÍA DE CÓMPUTO

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACION DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISION REVISORA

DRA. MAGDALENA MARCIANO MELCHOR
Presidente

M. EN C. EDUARDO VEGA ALVARADO
Secretario

M. EN C. JESÚS A. ÁLVAREZ CEDILLO
Primer Vocal
(Director de Tesis)

DR. OSWALDO MORALES MATAMOROS
Segundo Vocal
(Director de Tesis)

M. EN C. ROLANDO FLORES CARAPIA
Tercer Vocal

M. EN C. EDUARDO RODRÍGUEZ ESCOBAR
Suplente

EL PRESIDENTE DEL COLEGIO

DR. VÍCTOR MANUEL SILVA GARCÍA

S. E. P.
INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INNOVACIÓN Y DESARROLLO TECNOLÓGICO EN CÓMPUTO

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 18 del mes de Junio del año 2008, el que suscribe Adauto Israel Ortiz Romero, alumno del Programa de Maestría en Tecnología de Cómputo con número de registro B050917, adscrito al Centro de Innovación y Desarrollo Tecnológico en Cómputo, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. Oswaldo Morales Matamoros y del M. en C. Jesús Antonio Álvarez Cedillo y cede los derechos del trabajo intitulado **Construcción y programación de un algoritmo paralelo para determinar el exponente de Hurst**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección iortiz@ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Adauto Israel Ortiz Romero

RESUMEN

El presente trabajo tiene por objeto construir programar un algoritmo computacional que calcule el exponente de Hurst en un ambiente de programación en paralelo. La importancia de este trabajo recae en el volumen de datos que se utilizan para el cálculo de dicho exponente. Existen herramientas computacionales que solucionan este problema, sin embargo cuando se requiere procesar un conjunto mayor a los sesenta mil datos, dichas aplicaciones tienen un bajo desempeño ya que el número de iteraciones que se tienen que realizar crece considerablemente y el proceso de las mismas se vuelve lento e inoperable. Debido a que estas aplicaciones corren en un ambiente “mono usuario” basadas en equipos de este mismo ambiente (como computadoras personales, portátiles y de escritorio), las operaciones las realiza un solo procesador, entonces se requiere de una computadora de muy alto desempeño en cuanto a capacidad de procesamiento, lo que implica un costo de inversión adicional en la adquisición de equipos de nueva tecnología. En el Centro de Innovación y Desarrollo Tecnológico en Cómputo del Instituto Politécnico Nacional (CIDETEC-IPN), se cuenta con un Cluster de computadoras de alto desempeño, que tiene como característica principal tener una red de alta velocidad interconectada a un número de computadoras en las cuales se pueden distribuir las operaciones en cada una de ellas y realizar el procesamiento de los datos en forma simultánea logrando el procesamiento en paralelo.

La aplicación construida puede ejecutarse en este cluster, pues ha sido programada para que realice las operaciones en forma paralela. Como se mencionó anteriormente, cuando crece el número de datos a procesar, crece el número de iteraciones; la presente aplicación fue programada para que estas iteraciones

puedan ser distribuidas en cada uno de los procesadores disponibles en el cluster, lo que permite entre otras cosas, disminuir considerablemente el tiempo de procesamiento y procesar el número de datos que sea necesario sin restricción alguna; por otro lado los resultados obtenidos pueden ser portables y/o compatibles ya que se genera un archivo en formato estándar, el cual puede ser consultado en cualquier aplicación que maneje formato de texto ASCII (American Standard Code for Information Interchange).

ABSTRACT

The present work intends constructing and programming a computer algorithm that calculates the Hurst's exponent in a programming parallel environment. The importance of this work falls to the volume of data that are used for this exponent calculation. There are computer tools that allow solving this problem, nevertheless when a greater set to the sixty thousand data is required to process, these applications have a low performance since the number of iterations that must processing grows considerably and the process from their selves becomes slow and inoperable. Because these applications run in a "mono user" environment based on equipment of this same atmosphere (like personal, portables and desktops computers), operations are computing by a single processor, then it is required a high performance computer respecting to processing capacity, which implies a cost of additional investment in the acquisition of new technology equipments. In Center of Innovation and Technological Development in Computing of National Polytechnical Institute, is available a high performance cluster of computers, that it has as basic characteristics having a high speed interconnected network to a number of computers in which they are possible distributing operations in each of them and processing data in a simultaneous form obtaining parallel processing.

Developed application can run in this cluster, because it has been programmed to carries out its operations in parallel environment. As previously it has mentioned, when the number of data to process grows, the number of iterations grows; present application was programmed for these iterations can be distributed in each one available processors in cluster, which allows things among others, to diminish considerably the time of processing and processing a necessary number of data

without any restriction; by another way the obtained results can be portable and compatible since a file is generated in standard format, which can be opened in any application which handle ASCII (American Standard Code for Information Interchange) text format.

AGRADECIMIENTOS

A MI MADRE:

Por su tenacidad e ilusión de verme desarrollar profesionalmente y porque seguimos recogiendo los frutos de una herencia invaluable como lo es el estudio.

A MI ESPOSA:

Con quien comparto adversidades y malos ratos pero también dicha, felicidad y mucho amor.

A MIS HIJOS, JIMENA E ISRAEL:

Que son el motivo para levantarme y seguir siempre adelante y porque soy todo para ellos.

A MIS HERMANOS:

No quiero ser su ejemplo, pero si quisiera que siguieran este camino...

A MIS SUEGROS

Por todo su apoyo y porque sé que puedo contar con ustedes.

A MIS CUÑADOS, CUÑADAS, SOBRINOS TIOS Y TODA LA FAMILA,

Pues siempre es bueno contar con todos ustedes...

A MIS AMIGOS...

Que no los nombro porque saben peferctamente quiénes son...

I N D I C E

INTRODUCCIÓN.

El exponente de Hurst es una medida que, basada en una serie de datos o medidas en el transcurso del tiempo, permite analizar los distintos comportamientos que tiene un sistema a lo largo del tiempo, lo que permite conocer si dicha serie muestra procesos persistentes o correlacionados (un periodo de crecimiento es seguido de otro similar) o si corresponden a un proceso anti-persistente o anti-correlacional (a un periodo de crecimiento le sigue otro de decrecimiento), así como también puede evidenciar una ausencia de correlaciones entre los incrementos, considerándose como una serie aleatoria donde sus incrementos y decrementos son independientes unos de otros.

Si la serie muestra procesos persistentes, el sistema evidencia un comportamiento fractal, que tiene como definición a un conjunto de formas generadas normalmente por un proceso de repetición y se caracterizan por tener similitud en diferentes escalas, por no ser diferenciables y por exhibir dimensión fraccional; al ser evidenciadas estas características y dependiendo del sistema que se esté analizando, por un lado, podría pronosticarse el comportamiento futuro del sistema, así como la relación y dependencia de los valores que se obtienen del mismo. Para llevar acabo dicho análisis es necesario procesar la serie de datos que se obtiene del sistema a lo largo del tiempo.

El objeto de investigación de este trabajo es el procesamiento que se ha de realizar a dicha serie. Basado en la técnica del rango reescalado (R/S), se obtiene el exponente de Hurst. Para calcular dicho exponente se implementó un algoritmo que funciona de forma paralela, distribuyendo sus operaciones de cálculo en los diferentes procesadores que dispone una máquina paralela.

El alcance de esta investigación, es comprobar que una serie de datos muy grande puede ser procesada en menor tiempo que en la forma como se ha venido realizando.

El presente trabajo comprende de cinco capítulos que permiten obtener bases suficientes para elaborar una solución alterna a la existente y motivar el uso de herramientas de cómputo disponibles sin necesidad de realizar inversiones complementarias de equipo.

En el primer capítulo se introduce a la materia de estudio consistente en indicar qué es el exponente de Hurst, sus aportaciones y aplicaciones; además se plantea el problema, que se deriva de la necesidad que se tiene para aplicar dicha técnica en sistemas complejos considerando la disponibilidad de los recursos existentes, como lo son los equipos en los que se realiza el cálculo y las herramientas de software ó aplicaciones con que se cuenta.

El segundo capítulo presenta el marco conceptual de lo que son los sistemas complejos, los fractales, la dimensión fractal; y el exponente de Hurst; así como los tipos o categorías de fractales existentes como, fractales auto-similares y fractales auto-afines. Se describen los fractales auto-afines y los métodos en que se puede calcular el exponente de Hurst.

En el tercer capítulo se revisa el marco teórico de los lenguajes y los ámbitos de programación, se exponen las particularidades y los entornos de la programación paralela, así como se describen las diferentes arquitecturas de computadoras paralelas existentes y se detallan las características de la programación en paralelo con MPI.

En el cuarto capítulo se describe el método detallado del Rango Reescalado (R/S) y derivado de estos pasos, se propone una solución basada en la solución mediante un algoritmo y se construye una aplicación basada en un conjunto de funciones en lenguaje de programación C utilizando las interfaces de procesamiento paralelo de MPI.

El capítulo cinco muestra los resultados que se obtienen de la aplicación y se discuten los mismos, conjeturando y evidenciando las diferencias entre esta nueva aplicación y las aplicaciones con que se ha llevado a cabo el análisis de las series de datos.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1
INTRODUCCIÓN

Desde la década de los años setenta del siglo pasado el concepto de fractal ha venido a dar un gran significado a diversas áreas de estudio tanto de las ciencias exactas como de las sociales; con el desarrollo tecnológico computacional a la par de estos conceptos se han logrado conjuntar una serie de técnicas y cálculos que han permitido conocer desde otra perspectiva fenómenos que la geometría Euclidiana sólo aproximaba o era difícil realizar su cálculo.

Su aportación en algunas áreas de las ciencias como el análisis espectral, la hidrología, las finanzas y la meteorología ha sido fundamental, pues se ha permitido que, a partir de análisis matemáticos derivados con el comportamiento de ciertos sistemas, se logren pronosticar y determinar sucesos y fenómenos que pudieran ocurrir. Sin embargo, entre más detallado sea el análisis de ese comportamiento, se requerirá de un mayor número de elementos de cálculo derivando cálculos complejos que, a pesar de la existencia de computadoras veloces y capaces de procesar infinidad de información, se requiere de un conjunto de técnicas y/o algoritmos que simplifiquen el procesamiento y permitir la ejecución o generación de resultados en un lapso de tiempo ideal con el fin de tomar una decisión correcta.

Dadas las características del cálculo de la dimensión fractal, con ciertos métodos se pueden aplicar técnicas de programación basada en procesamiento paralelo que permitan obtener resultados de manera rápida y precisa, además de poder ejecutar dicha programación en equipos de alto desempeño a bajo costo si se toman en cuenta equipos científicos instalados en planteles de educación e investigación pública especializados en esta materia, como lo puede ser un *cluster HPC.*[1]

[1] High Performace Cluster. Cluster de computadora de alto rendimiento conformado por elementos de cómputo interconectados que distribuyen la capacidad de cómputo en diferentes nodos permitiendo el procesamiento computacional de cálculos complejos.

1.1 APORTACIONES DEL EXPONENTE DE HURST

En 1906 el ingeniero hidrólogo inglés Harold Edwin Hurst, comenzó a estudiar las fluctuaciones del Río Nilo, en Egipto, con el fin de determinar las dimensiones y los cálculos apropiados para la construcción de una nueva presa en Asuán. En esa época, se comprendían bien las fluctuaciones fluviales estacionales. Pero la variación anual en un río tan vasto era un problema muy distinto. El caudal del Río Nilo variaba ampliamente, desde los 151,000 millones de metros cúbicos de un año húmedo, hasta los 42,000 millones de metros cúbicos durante la sequía. Además, a ese periodo seco le siguió otro sólo dos años más tarde. Las épocas de abundancia también tendían a agruparse, sin embargo, no existía una periodicidad obvia.

El diseño de presas era una tarea trascendente desde el siglo XIX, que, como sucede hoy en día con las finanzas, se prefería la vía matemáticamente más fácil de realizar sus estimaciones. Los expertos asumían que las variaciones anuales del caudal eran estadísticamente independientes, como en el lanzamiento de una moneda. Con la observación a largo plazo y con datos recavados de épocas distantes, Hurst determinó un nuevo método para evaluar la dependencia a largo plazo sobre una serie de medidas. Hurst quería caracterizar el caudal total del agua con un promedio histórico a lo largo de muchos siglos (largo plazo), concluyendo que para intervalos temporales de longitud se pueden tener características de *escalamiento* o *fractalidad* [1].

Derivado de esta teoría, se determina el exponente de Hurst (*H)*, y se aplica a series de datos históricos, con el objeto de determinar si un sistema o fenómeno natural, muestra procesos persistentes o correlacionados que cumplen con ciertas características fractales.

Los usos diversos que se le han dado al cálculo del exponente de Hurst, ha permitido el desarrollo de nuevas herramientas para cálculos complejos, una de las aportaciones más notables de estos cálculos es pronosticar posibles eventos que ocurren dentro de un sistema real o de la naturaleza; ejemplos de aplicaciones de este cálculo son en el campo de las matemáticas, tales como el análisis espectral, la hidrología, impulsora de esta técnica, análisis financieros de los indicadores de

precios de productos e insumos diversos como el petróleo, índices de inflación, etc; y muchas otras aplicaciones de diversas disciplinas.

1.2 LA NATURALEZA FRACTAL

Un Fractal es, matemáticamente hablando, una figura geométrica compleja y detallada en estructura a cualquier nivel de magnificación. A menudo los fractales son semejantes a sí mismos; esto es, poseen la propiedad de que cada pequeña porción del fractal puede ser visualizada como una réplica a escala reducida del todo. La característica por la que son llamados fractales es su dimensión fraccionaria. No tienen dimensión uno, dos o tres como la mayoría de los objetos a los cuales estamos acostumbrados. Los fractales tienen usualmente una dimensión que no es entera, ni uno, ni dos [2].

La noción de dimensión fractal o fraccional provee una manera de medir qué tan rugosa es una curva (que tanto "serpentea" entre un punto y otro); normalmente en la geometría euclidiana se establece que los puntos carecen de dimensión, las líneas de dimensión uno, las superficies de una dimensión de valor dos y los volúmenes de tres. Esta idea de dimensión es llamada comúnmente dimensión topológica[2]. Sin embargo, una *curva rugosa* que recorre una superficie puede ser tan rugosa que casi llene la superficie en la que se encuentra.

La hipótesis de *fractalidad* de la naturaleza implica "*auto-similitud*" en el sentido de que las partes replican en una escala menor al total. La *fractalidad* podrá ser *determinista* (como el triángulo de Sierpinski mostrado en la figura 2.3 o el copo de nieve de Koch que se mostrará en la figura 2.2) o *estocástica* (como las ramas de árboles, las que crecen de acuerdo a una escala fractal) [3].

[2] La topología es una rama de las matemáticas que se ocupa de aquellas propiedades de las figuras geométricas que no cambian cuando se estira o se dobla el objeto. Una forma de definir una dimensión topológica es la dimensión de cubierta que es cuando se cubre una línea con intervalos chicos siempre hay puntos que se encuentran en al menos dos intervalos.

1.3 HERRAMIENTAS QUE PERMITEN EL CÁLCULO DEL EXPONENTE DE HURST

Debido a que el campo de aplicación de cálculo del exponente de Hurst, se reduce a las áreas de las matemáticas, existen pocas aplicaciones comerciales que permitan realizar los cálculos para el análisis fractal. La mayoría de las existentes son elaboradas a la medida y difícilmente existe un mercado para su difusión.

La empresa TruSoft Int'l Inc desarrolló una herramienta llamada "BENOIT Fractal Análisis System" con las siguientes características:

BENOIT Versión 1.2 es una aplicación que permite calcular la dimensión fractal y/o el exponente de Hurst de un conjunto de datos usando once distintos métodos:
Para el análisis de patrones auto-similares:

1. Caja (box),
2. Dimensión perímetro-área,
3. Información (information),
4. Dimensión de masa (Mass dimension),
5. Regla (ruler),

Para el análisis de trazado auto-afin:

6. Análisis Rango escalar(R/S),
7. Espectro de potencia (Power Spectrum),
8. La longitud de rugosidad (roughness-length),
9. El variograma, y
10. Las ondoletas (Wavelets)
11. Dimensión de fragmentación por frecuencia de datos.

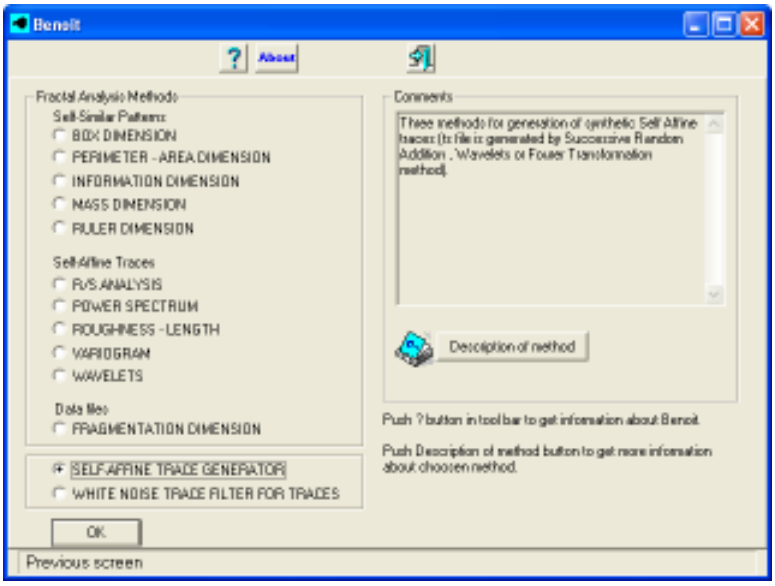


Figura 1.1 Métodos de Análsis Fractal tomado de la pantalla principal del software Benoit 1.2

La filtración se proporciona en el programa al permitir retirar ruido blanco de su trazo usando las técnicas de Fourier o de las ondoletas.

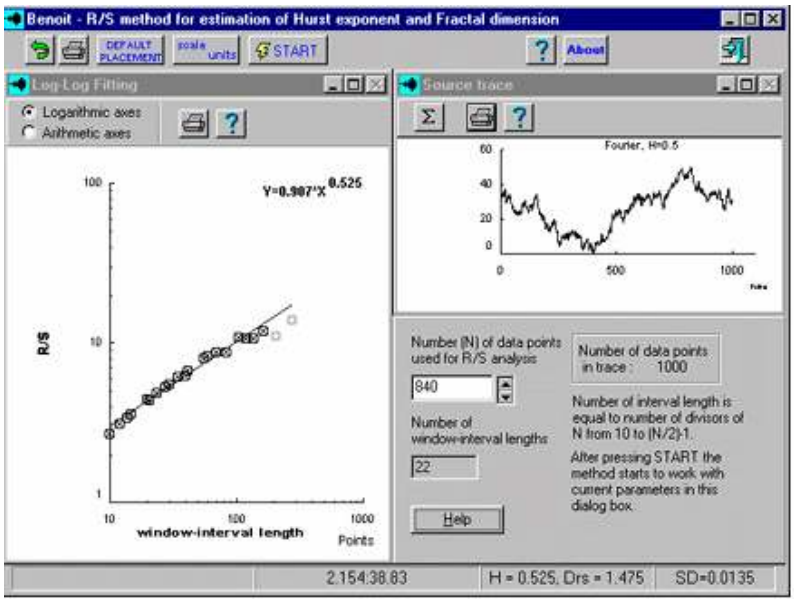


Figura 1.2 Gráficas de Análisis Fractal generadas por el software Benoit 1.2

BENOIT es una interfaz visual amigable y una herramienta de investigación y de estudio para cualquier persona interesada en las propiedades fractales de una serie de datos. El usuario realiza algunos clics y *Benoit* calcula automáticamente los

valores predeterminados para varios parámetros así como se controla el funcionamiento de cada método de forma manual. Las gráficas interactivas que genera de los datos permiten ajustar puntos de referencias fuera del rango fractal para aumentar exactitud.

Los formatos de datos que *Benoit* acepta y permite realizar sus cálculos son los siguientes:

1. Archivos *Bitmap* (*.*bmp*). Los patrones de datos para analizar con los métodos de auto-similitud (a excepción del método de Fragmentación) deben tener formato de archivo *.*bmp*.

2. Archivos para trazado (propietario de *Benoit*) (*.*ts*). Deben de tener la siguiente distribución: La primera línea es el nombre de la serie de datos (debe tener menos de 512 caracteres incluyendo espacios); a partir de la segunda línea, los valores de cada punto, seguidos de un salto de línea o *enter*, y tendrán que ser valores reales positivos, incluyendo el punto decimal.

3. Hojas de cálculo de MS Excel 4.0 (*.*xls*). El formato será el siguiente: cada línea, el valor de cada dato, seguido de un salto de línea o *enter*, y tendrán que ser valores reales positivos, incluyendo el punto decimal. Solo una columna de la hoja de datos puede ser seleccionada y la longitud del nombre del archivo de Excel no podrá exceder los ocho caracteres.

Los datos que se analizarán por el método de auto-afinidad, o los datos del método de fragmentación podrán tener el mismo formato y deberán ser archivos de tipo trazo (*.*ts*) o bien archivos de *Excel* Versión 4.0 o inferior.

1.4 DEFINICIÓN DEL PROBLEMA

Derivado de los métodos para el análisis fractal, particularmente de los fractales autoafines, se ha logrado simular la conducta que siguen algunos sistemas relacionados con las finanzas, demostrando en varios casos que reflejan un comportamiento determinando la *fractalidad* de sus elementos. Un caso en particular son los índices de precios del petróleo a nivel mundial, que al ser estudiados

históricamente, y que cumplen con características fractales, se podrían realizar pronósticos, acercado los resultados a un posible suceso futuro.

Sin embargo, un conjunto de datos históricos de este tipo de aplicaciones puede ser desde una cotización de precios dada por día; así como los datos podrían descomponerse jornadas de horas, y hasta en minutos ó segundos, generando un extenso número de elementos de análisis que habrá de considerase en el análisis del propio sistema.

1.4.1 ANTECEDENTES

En la Escuela Superior de Ingeniería Mecánica y Eléctrica, unidad Zacatenco del Instituto Politécnico Nacional, existe un grupo de investigación enfocado al análisis fractal de la dinámica de sistemas complejos, como las fluctuaciones de los precios internacionales del petróleo; para ello se requiere analizar bases de datos muy grandes, caracterizarlas y simular su comportamiento; con las herramientas computacionales con las que este grupo de investigación cuenta se obtienen resultados de una forma lenta, tediosa y no se pueden suministrar mas allá de 65,536 datos, ya que es la capacidad máxima de las celdas disponibles en *MS Exel,* que interactúa con el *Software Benoit 1.2.* Por lo mismo, se ha planteado la necesidad de construir y programar un algoritmo computacional que disminuya el tiempo de respuesta y que sea adaptable a las necesidades del grupo de investigación.

Por ende se pretende dar solución aplicando un algoritmo basado en programación paralela, distribuyendo el procesamiento de los datos en diferentes entidades de procesamiento, pero corriendo o ejecutándose en un mismo instante de tiempo posibilitando el uso de técnicas y herramientas de procesamiento computacional en paralelo.

1.4.2 JUSTIFICACIÓN

El CIDETEC cuenta con una infraestructura física de cómputo que permite la ejecución de programas en paralelo; dentro de la currícula de su programa de Maestría en Tecnología de Cómputo se tienen programadas materias relacionadas

con procesamiento paralelo, a nivel teórico y práctico. Se pretende implementar los conocimientos adquiridos en esas materias proponiendo una solución al problema planteado anteriormente, ya que se cuenta con el personal académico con experiencia en el ámbito de paralelismo y programación, impulsando el desarrollo e implementación de dichas herramientas e identificando las ventajas que se tienen con respecto al uso de herramientas basadas en la programación lineal.

1.4.3 OBJETIVO GENERAL

Obtener una herramienta no secuencial para determinar si un sistema complejo es predecible a diferentes escalas de tiempo-espacio.

1.4.4 OBJETIVOS PARTICULARES

1. Analizar el software disponible en el mercado para determinar el exponente de Hurst.
2. Establecer el estado del arte de herramientas no secuenciales para determinar el exponente de Hurst.
3. Construir el algoritmo en paralelo computacional para determinar el exponente de Hurst.
4. Programar el algoritmo computación paralelo computacional.
5. Validar el algoritmo paralelo computacional desarrollado.

1.4.5 HIPÓTESIS

Si se *paraleliza* el método para el cálculo de la dimensión fractal mediante un algoritmo paralelo computacional, se logrará obtener una respuesta en menor tiempo en comparación con el método secuencial, permitiendo realizar aplicaciones con problemas complejos que requieren del análisis fractal a partir del cálculo del exponente de Hurst.

1.4.6 METODOLOGÍA

El trabajo se basó en la investigación documental; se estableció la base teórica del problema, investigando y documentando los conceptos de Fractales, dimensión fractal, las técnicas para su cálculo y la selección de la más adecuada, la cual se tendrá que analizar con el fin de verificar si los cálculos se pueden basar en operaciones paralelas.

Se investigaron las diferentes técnicas de paralelismo, que serán mencionadas más adelante que se puedan implementar con la infraestructura disponible. Se documentan los tipos de programación paralela existentes y se seleccionó la más adecuada para la implementación del problema planteado. Es pertinente tomar en cuenta que, derivado de estas técnicas, existen herramientas de desarrollo que permiten el diseño e implementación de los algoritmos paralelos; sin embargo, optar por la más factible, depende, por un lado, de los equipos y sistemas con los que se cuenta y, por el otro, del costo en que incurre el pago de una licencia de uso de un producto de desarrollo de este tipo, siendo el software libre una herramienta factible.

Se utilizó un Cluster HPC con 8 nodos equipados con dos procesadores Pentium III con una capacidad de procesamiento de 1 GHz cada uno, interconectados en una red de alta velocidad, con un sistema operativo Linux, compilando las aplicaciones con el compilador *GNU gcc y mpigcc* basado en el lenguaje de programación ANSI C.

CAPÍTULO 2
EL EXPONENTE DE HURST

El exponente de Hurst es una técnica que se utiliza para realizar análisis de los distintos comportamientos de un sistema en un período de tiempo; estimando si una serie de datos sigue un comportamiento *autosimilar* a lo largo de ese período. Con los resultados obtenidos, se pude conocer si el comportamiento de un sistema ó un fenómeno natural, muestra procesos persistentes o correlacionados.

El exponente de Hurst se originó a partir de las observaciones que llevó a cabo el ingeniero Harold Edwin Hurst para analizar las corrientes del rio Nilo en Egipto, para lo cual estudió el comportamiento de las series temporales del caudal producido por el mismo; este análisis lo llevó a desarrollar un método estadístico llamado Análisis R/S (Rescaled Range Analysis-Análisis de rango Reescalado) [5].

Por medio de este análisis se puede encontrar el parámetro de Hurst, el cual es un valor numérico que permite determinar la auto-correlación en una serie de datos.

2.1 SISTEMAS COMPLEJOS

Los fenómenos en muchas áreas de las ciencias, como la administración, las finanzas, la ingeniería, la biología, la medicina entre otras, son de naturaleza compleja y dinámica y requieren ser estudiados en base a la teoría de la complejidad que como aportación consiste en que dicha teoría proporciona una nueva manera de pensar sobre estas estructuras o fenómenos, presentando un nuevo paradigma al hacer énfasis en aspectos no lineales y flexibles en los sistemas que se han de estudiar y analizar.

Formalmente no existe definición única de complejidad, sin embargo se pueden mencionar algunas:

“Los sistemas complejos consisten en un gran número de elementos o agentes que actúan recíprocamente para producir los cambios dinámicamente a través del tiempo. La interacción, en cualquier elemento del sistema, puede influir y ser influido por otro elemento u otros elementos, resultando por ello los efectos no lineales. Los agentes reciben la información y actúan recíprocamente con otros vecinos.”(Cillers) [6].

“El tema se hace más complejo cuando las partes componentes de una organización pueden variar individualmente en su grado de complejidad” (Hall, 1993 p. 75) [6].

“La complejidad de cualquier sistema es la respuesta al medio ambiente que la rodea” (Robbins 1990 p. 12) [6].

“La complejidad es una propiedad intrínseca de sistemas del universo que evolucionan al adquirir mayor y más diversificado número de elementos que interactúan entre ellos. Entre mayor es la organización, crece la necesidad de cuidar el control de sus elementos y la comunicación que se da entre ellos” (Nelly) [6].

De acuerdo con estas definiciones, “un virus es más complejo que un meteorito, porque la complejidad estriba no sólo en la acumulación de elementos, sino en la diversidad de éstos y la calidad de sus interacciones. Así, al comparar el virus y el meteorito, la calidad de meteorito será invariable, sin que importase la cantidad de materia que se le removiese en un amplio rango, en tanto que el virus dejaría de serlo al remover un mínimo porcentaje de su materia” (Holbrook, p. 78, 2003) [6].

Todos los sistemas complejos reales generalmente exhiben *invariancia* de escala; es decir, su comportamiento no cambia por el reescalado de las variables que gobiernan su dinámica. Esto nos posibilita a emplear conceptos fractales, como es el caso del enfoque de escalamiento dinámico, para estudiar, caracterizar, modelar y predecir (desde el punto de vista estadístico) tanto la cinética del crecimiento de interfases rugosas, como series de tiempo que contengan datos de las fluctuaciones de los mercados financieros (ver figura 2.1) [7].

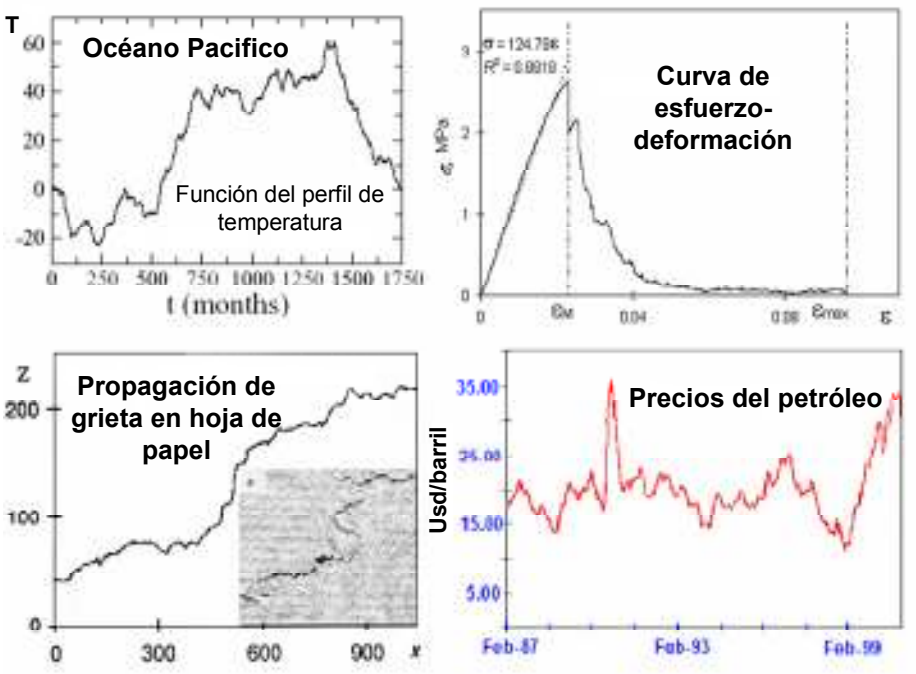


Figura 2.1. Sistemas complejos que exhiben invariancia de escala y que pueden ser caracterizados, analizados y modelados con la teoría de fractales.

Antes que nada, es necesario definir qué se entiende por interfase. Una interfase es una superficie que separa dos fases de la materia, cada una de los cuales puede ser sólida, líquida o gaseosa. Una interfase no es una superficie geométrica, sino más bien es una capa delgada que contiene propiedades diferentes a las de las fases (o superficies) de la materia que separa. Los procesos que ocurren en las interfases incluyen la evaporación de líquidos, la acción de los detergentes y los catalizadores químicos, así como la absorción de gases en los metales.

En muchos casos, las superficies (o fases) de las interfases que crecen son rugosas (como la superficie de los fragmentos de los vidrios cuando se rompen). El crecimiento de interfases rugosas bajo condiciones de no-equilibrio es un fenómeno (crítico) muy común en la naturaleza. Ejemplos de tales procesos incluyen la difusión, el quemado, el crecimiento de grietas, el fraguado de materiales granulares y el flujo de fluidos en medios porosos.

La rugosidad de interfases que crecen en no-equilibrio es sólo uno de los campos en los cuales la invariancia de escala se ha observado como una característica común y básica. La invariancia de escala de las interfases en crecimiento, los eventos y la información sobre un amplio intervalo de escalas de

longitud y tiempo se presenta de tal manera que no importa cuál es el tamaño de la escala considerado, ya que siempre se observa sorprendentemente una riqueza en estructuras complejas.

Debido a que la extensión de una superficie (interfase) está relacionada en la forma en qué tan rugosa es esta, el concepto de rugosidad puede ser descrito dentro de un marco matemático denominado geometría fractal [7], por lo que en el siguiente apartado se presenta una breve introducción sobre la teoría de fractales.

2.2 FRACTALES

En años recientes, para entender la naturaleza de las estructuras desordenadas y su formación mediante procesos aleatorios, se han desarrollado los conceptos fractales, introducidos por *Benoit Mandelbrot* en una teoría llamada geometría fractal [8]. La *auto-similitud* y la *auto-afinidad* son los conceptos que unifican áreas como fractales, leyes de potencia y caos. La auto-similitud es una de las simetrías fundamentales que rigen el universo. De igual manera, la auto-afinidad, o invariancia bajo cambios de escala o tamaño *anisotrópico*, es un atributo de muchas superficies e interfases que se presentan en algunos fenómenos naturales y económicos.

La geometría fractal, o teoría de fractales, es un lenguaje matemático empleado para describir geometrías complejas e irregulares que permanecen invariantes a los cambios de escala y cuya longitud infinita se encuentra contenida en un área finita. El propósito de la geometría fractal es el de caracterizar cuantitativamente cómo el espacio es ocupado por una curva o una geometría particular. La geometría fractal se aplica para caracterizar los fenómenos críticos que presentan invariancia de escala, como es el caso de las interfases, mediante la explotación de su principal propiedad: la ausencia de una escala característica, por lo que no existe ninguna unidad asociada, es decir, el mismo modelo es válido para todas esas escalas [7].

Este tipo de fenómenos críticos, independientes de la escala, son conocidos así porque fueron inicialmente observados en los puntos críticos de las transiciones

de fases continuas. La física de los procesos críticos está únicamente dominada por las simetrías del sistema. Así pues, distintos sistemas críticos, compuestos por distintas sustancias y a diferentes escalas, pueden ser descritos por un solo modelo. Esta idea fue la que dio origen a las clases de universalidad, que es la manera en como son clasificados los distintos sistemas críticos. Cada una de dichas clases aparece como resultado de unas simetrías distintas propias de un grupo de fenómenos críticos [7].

A fines del siglo XIX y principios del XX, los fractales fueron descubiertos como un grupo de estructuras que presenta una paradoja para la teoría de la medición: la imposibilidad de observar curvas de longitud infinita encerradas dentro de áreas finitas, por lo que a estas curvas (fractales) se les llamó galería de monstruos. La solución a tal paradoja llegó con la modificación del concepto de dimensión, aplicando la dimensión de Hausdorff, o dimensión fractal; Supóngase que μ es la magnitud a ser medida: longitud, área o volumen de un cierto conjunto. El método usado para estimar μ es cubriendo la estructura original con pequeños conjuntos del propio μ, es decir μi, y de longitud l. Si $\mu(l) = \sum i \mu i$ es la aproximación de μ en la escala l, la dimensión de Hausdorff está definida como:

$$D = \lim_{l \to 0} \frac{\log(\mu(l))}{\log(1/l)} \qquad \text{..... (2.1)}$$

Donde $\mu(l)$ es el número de pequeños segmentos de longitud l necesarios para cubrir un segmento de longitud *L*.

La dimensión fractal, es un entero para los objetos geométricos clásicos, como puntos (*D=0*), líneas (*D=1*), cuadrados (*D=2*), o cubos (*D=3*), y toma valores no enteros (o fraccionarios) para los fractales. Por ejemplo, para la curva de *Koch* (figura 2.2) $D = \log(4)/\log(3) = 1.2618$, un valor que se encuentra entre la dimensión de una línea, *D=1*, y el de una superficie, $D = 2$.

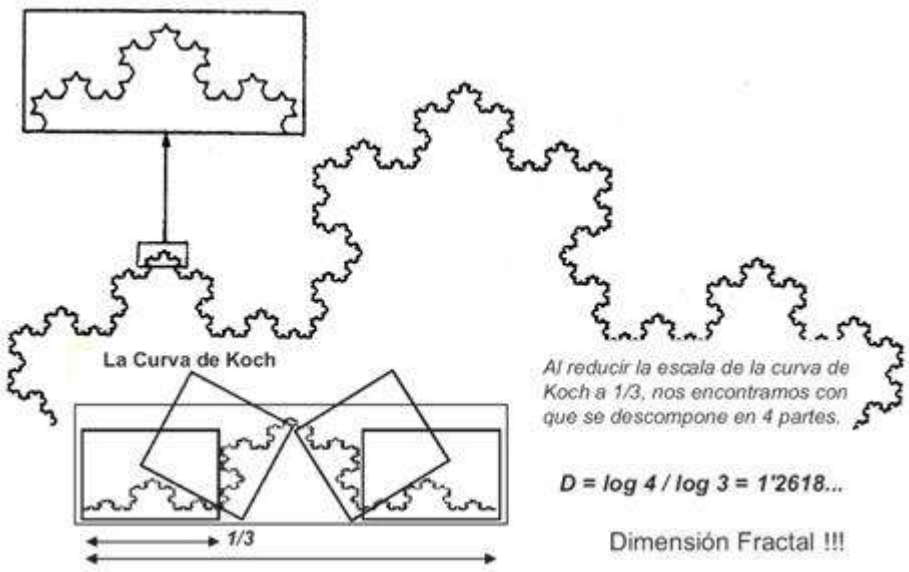


Figura 2.2. Fractal auto-similar deterministico: la curva de Koch.

Es importante destacar que $\mu(l)$ se comporta como una ley de potencia. La característica relevante de las leyes de potencia, es que éstas son las únicas funciones cuya forma es de invariancia de escala (simetría que define la geometría fractal). Por consiguiente, la mayoría de las propiedades de los fractales están expresadas mediante leyes de potencia.

Un fractal puede ser construido por dos métodos distintos. En el primer procedimiento el fractal crece, es decir, una figura es seleccionada y copias de esta son agregadas al conjunto, manteniendo algunas geometrías preestablecidas. Existen muchos sistemas reales en el mundo que emplean una técnica similar de crecimiento, algunos ejemplos son, la electro-deposición, las colonias de bacterias o el crecimiento de cristales. Por otro lado, la construcción de un fractal puede iniciar también a partir de una figura con una dimensión *d* mayor que la del conjunto futuro; la figura inicial es perforada, por lo que en la misma se producen hoyos de todos tamaños (figura 2.3). Si la distribución de los hoyos es una ley de potencia, $P(s) \sim s^{-\gamma}$, el conjunto final es un fractal con una dimensión $D = d(\gamma - 1)$. Este proceso es similar a la erosión en las rocas. Los dos métodos ofrecen diferentes clases de fractales. Aquellos fractales que son producidos mediante hoyos tienen un tamaño finito en el espacio que los alberga (estos pueden ser encerrados dentro de un hiper-volumen finito), por lo que más allá de cierta escala estos fractales pierden su fractalidad. Por

el contrario, aquellos fractales que son generados por la adición de piezas pequeñas presentan un tamaño infinito, pero también poseen un rango más bajo en la resolución y en la invariancia de escala [9].

Figura 2.3. El triángulo de Sierpinski, después de cuatro iteraciones.

Existes dos tipos de fractales: *auto-similares* y *auto-afines*. Los fractales *auto-similares* son *isotrópicos*, es decir, permanecen invariantes cuando la escala se cambia uniformemente en todas las direcciones. La invariancia puede ser literal, al realizar una ampliación de una parte se observa exactamente la misma estructura, o puede ser estadística: no se ve exactamente lo mismo, pero el resultado de la ampliación es indistinguible del conjunto original. Cualquier estimador estadístico que se aplique a dicho conjunto da lugar a los mismos resultados que si se aplica al conjunto incompleto (ver figura 2.4).

No obstante, es fácil encontrar situaciones en donde dos o más direcciones no son equivalentes; en este caso, el conjunto pertenece a un tipo diferente de fractales, llamados fractales *auto-afines*.

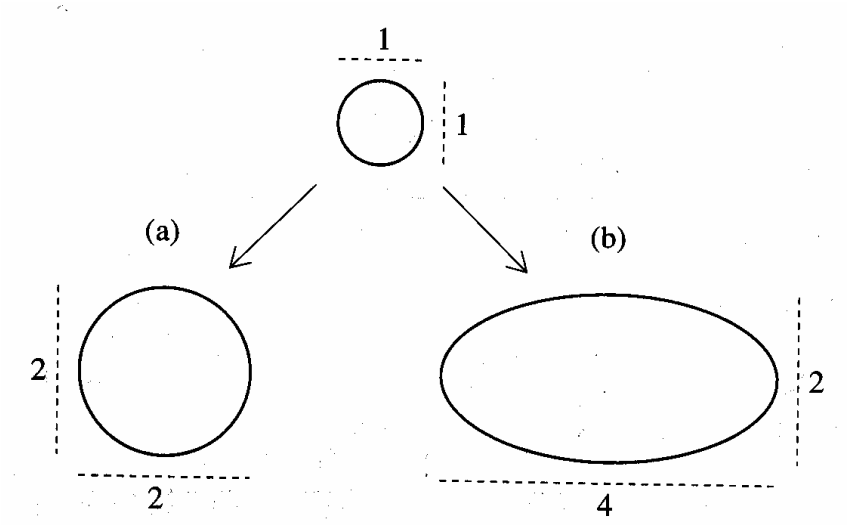


Figura 2.4. El efecto de los reescalados isotrópico y anisotrópico de un simple objeto, el círculo. (a) El diámetro es alargado por un factor de dos en el caso del cambio isotrópico de escala. (b) En el caso del reescalado anisotrópico, el diámetro (de este a oeste) es alargado por un factor de cuatro; mientras que de norte a sur el diámetro es alargado por un factor de dos, dando como resultado una elipse.

Los fractales *auto-afínes* son conjuntos que permanecen invariantes bajo una escala (estadística o literalmente) de transformación *anisotrópica*. A pesar de sus diferencias, en una escala de transformación las direcciones no son completamente independientes. Si al hacer una ampliación, uno de los ejes de coordenadas se transforma en un factor b, $x \to bx$ (2.2), el resto de los ejes coordenados deben ser reescalados en un factor b^{α_i}, $x_i \to b^{\alpha_i} x_i$ (2.3), con el objeto de preservar al conjunto invariante (figura 2.4). Los exponentes α_i son llamados exponentes de rugosidad, o exponentes de Hurst (H), y nos indican cuál es el grado de *anisotropía* (o rugosidad) del conjunto.

En cuanto al crecimiento de interfases, la dimensión especial corresponde a la dirección del crecimiento. La existencia de una dirección privilegiada implica que sólo existe un exponente de Hurst [7]. En la figura 2.5 se aprecia que el movimiento *Browniano*, al igual que una interfase rugosa, es un fractal *auto-afín* estadístico; en esta figura se observan tres curvas diferentes del movimiento *Browniano*, cuyo comportamiento está en función del valor de H con el que fueron generadas, ya que H indica el grado de rugosidad de las curvas.

Como puede apreciarse (figura 2.5), el exponente de Hurst es también un indicador para determinar si un fenómeno o una serie de tiempo presentan un comportamiento fractal y mide la intensidad de dependencia a largo plazo de una serie de tiempo. Se dice que el fenómeno analizado es aleatorio cuando $H = 0.5$ (ruido blanco), que es persistente cuando $0.5 < H < 1$ (existe invariancia de escala asociada a correlaciones positivas a largo plazo), y que es antipersistente cuando $0 < H < 0.5$ (existe invariancia en la escala asociada a correlaciones negativas a largo plazo) [10].

Finalmente, es importante mencionar que el exponente de Hurst está relacionado a la dimensión (fractal) local del conteo de cajas de la interfase como $D_B = 2 - H$ [11]. …… (2.4)

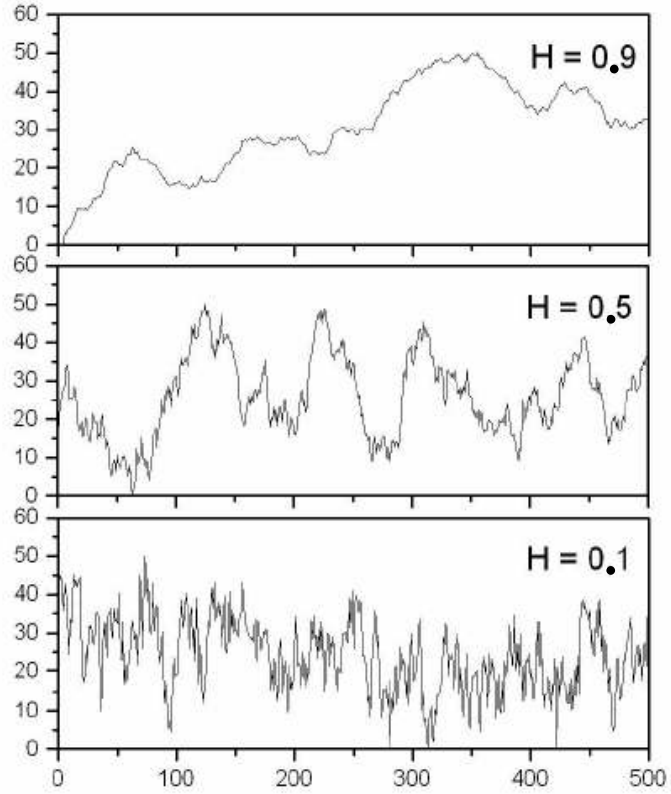


Figura 2.5. Fractal auto-afín estadístico: curvas fractales para diferentes valores de exponentes de Hurst (H) o de rugosidad (α):H=0.9, comportamiento persistente;H=0.5, comportamiento aleatorio; y H=0.1, comportamiento antipersistente.

Ahora bien, el hallazgo de que el crecimiento estocástico de superficies (o curvas rugosas o complejas), exhiba un comportamiento en escalamiento no trivial y

que evoluciona naturalmente a un estado estacionario, que no tiene una escala de tiempo o espacio característica, ha conducido al desarrollo del concepto fractal de escalamiento dinámico. Por lo tanto, en el siguiente apartado se explica qué es el escalamiento dinámico, los tipos de escalamiento dinámico, así como los modelos desarrollados para caracterizar y predecir el crecimiento del crecimiento de interfases rugosas, a partir del escalamiento dinámico.

Al hablar del escalamiento dinámico en el crecimiento de interfases rugosas, una interfase es una superficie que separa dos fases de la materia. Cada una de los cuales puede ser sólida, líquida o gaseosa. Una interfase no es una superficie geométrica, sino más bien es una capa delgada que contiene propiedades diferentes a las de las fases de la materia que separa. Los procesos que ocurren en las interfases incluyen la evaporación de líquidos, la acción de los detergentes y los catalizadores químicos, y la absorción de gases en los metales [11].

El crecimiento de interfases rugosas bajo condiciones de no-equilibrio es un fenómeno muy común en la naturaleza. Ejemplos de tales procesos incluyen la difusión, el quemado, el crecimiento de grietas, fraguado de materiales granulares y flujo de fluidos en medios porosos. En la figura 2.6 se presenta las configuraciones de la evolución de una interfase en diferentes intervalos de tiempo [12].

Algunos trabajos experimentales [13] sugieren que realmente muchos sistemas físicos desarrollan espontáneamente correlaciones con el comportamiento de leyes de potencia en el espacio - tiempo. Estos sistemas, con muchos grados de libertad, generalmente son tan complejos que su comportamiento a escalas macroscópicas no se puede predecir a partir de su comportamiento a escalas microscópicas. Surgen nuevos tipos de comportamientos colectivos y su comprensión representa una de las áreas más desafiantes en la física estadística moderna. La rugosidad de interfases que crecen en no-equilibrio es sólo uno de los campos en los cuales la invariancia de escala se ha observado como una característica común y básica. La invariancia de escala de las interfases en crecimiento, los eventos y la información sobre un amplio intervalo de escalas de longitud y tiempo, se presenta de tal manera que no importa cual es el tamaño de la

escala considerado, ya que siempre se observa sorprendentemente una riqueza en estructuras complejas.

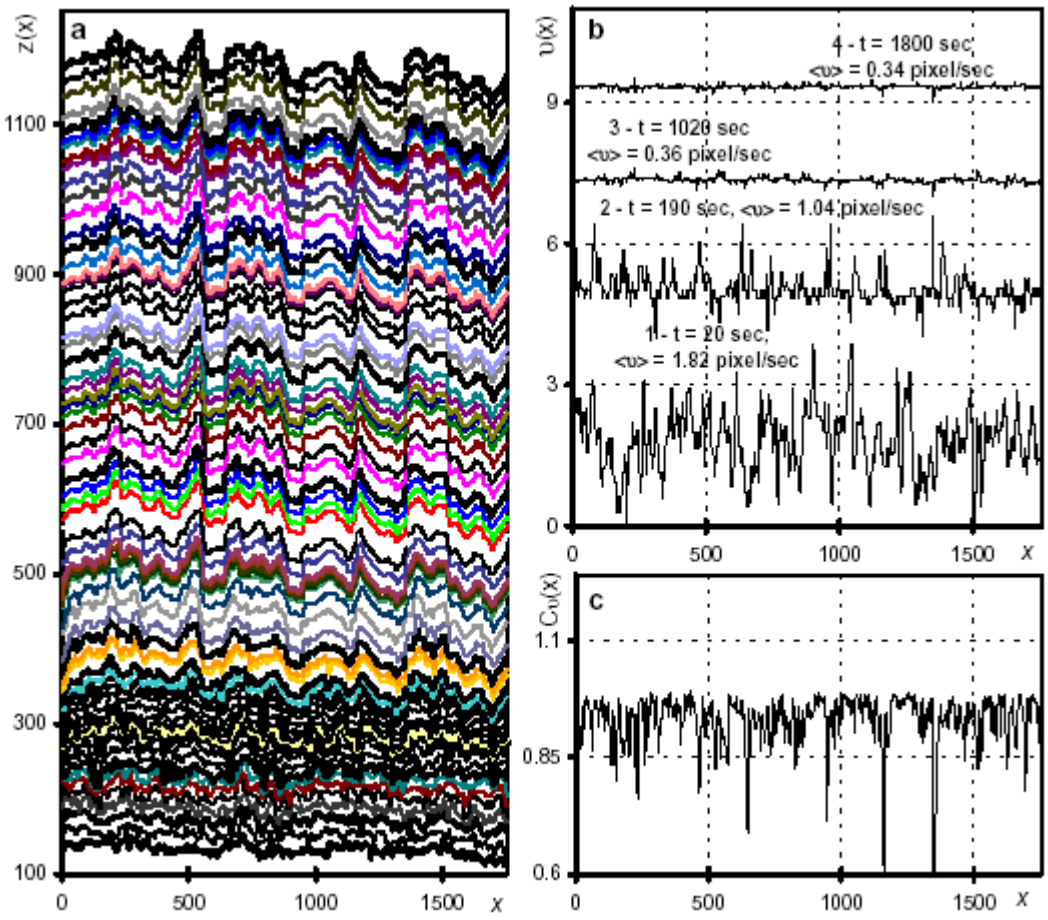


Figura 2.6. (a) Configuraciones de la evolución de una interfase en intervalos de tiempo: Δt = 10 segundos hasta t = 300 segundos, y Δt = 60 segundos hasta que la interfase se detiene; (b) Configuraciones de la velocidad de la interfase a diferentes tiempos, la distribución (1) corresponde a la etapa inicial, la distribución (2) corresponde al régimen transitorio, y las distribuciones (3) y (4) corresponden al estado estacionario (régimen saturado).

El hecho de que el crecimiento de interfases en sistemas en no-equilibrio exhiba propiedades de escalamiento complejas no nos dice nada acerca del por qué sucede así. Por lo tanto, un punto crucial para comprender este fenómeno es el origen de la invariancia de escala general de la rugosidad de la interfase. Esto correspondería a la comprensión del origen de las estructuras fractales y de las propiedades de Criticidad Auto-Organizada (SOC, siglas en inglés), a partir del conocimiento de los procesos físicos microscópicos en la base de estos fenómenos.

No obstante, actualmente no existe un formalismo sistemático que trate los procesos de crecimiento en no-equilibrio. Esto implica que los métodos de análisis tradicionales de la mecánica estadística no son los adecuados para describir el fenómeno de crecimiento de interfases. Sin embargo, el hallazgo de que el crecimiento estocástico de superficies exhibe un comportamiento en escalamiento no trivial y que evoluciona naturalmente a un estado estacionario, que no tiene una escala de tiempo o espacio característica, ha conducido al desarrollo de un método general de escalamiento [14] para describir de manera cuantitativa el crecimiento de interfases. Este formalismo, que se basa en conceptos generales de invariancia de escala y fractales se ha convertido en una herramienta estándar en el estudio del crecimiento de interfases. En particular, el método de escalamiento dinámico se ha aplicado al estudio de una gran variedad de modelos teóricos de crecimiento de interfases, incluyendo la deposición balística [14, 15, 16] y el modelo Eden [17].

Con el fin de simplificar el trabajo teórico, las interfases son comúnmente representadas por funciones de un solo valor, $h(x, t)$, donde h es la altura de la superficie sobre la posición del substrato x en el tiempo t. El procedimiento para establecer un perfil de h no siempre está bien establecido. En muchas situaciones reales, la superficie puede encorvarse sobre si misma, produciendo entonces una distribución de alturas multi-valores muy complicada. Algunas soluciones han sido probadas para resolver esta situación. La posibilidad más simple es sólo considerar el valor máximo de h en cada posición. Mientras los máximos tengan un tamaño característico, este método, o alguno otro similar, puede ser válido.

2.3 EXPONENTE DE HURST

En muchos casos, el comportamiento de escalamiento espacial se atribuye a la invariancia estadística de las interfases saturadas bajo la transformación de escala *auto-afín* ($\lambda x, \lambda^{\alpha} z$). La invariancia *auto-afín* implica que $z(\lambda x) \cong \lambda^{\alpha} z(x)$, donde "$\cong$" denota la igualdad en el sentido estadístico, y no hay escala de longitud característica en la interfase en comparación al tamaño del sistema [18]. Por lo tanto, en este caso la magnitud de las fluctuaciones de las alturas locales sobre una ventana de tamaño $l < L$ se mantiene saturada en el tiempo $t_s \sim L^z$,

independientemente del tamaño del sistema; o sea, $h(L, t > t_s) \sim L^{\zeta}$, donde $\zeta = \alpha$ es el exponente de rugosidad local. Si es así, la morfología de una interfase de una grieta es caracterizada por el exponente de rugosidad único, $0 \leq \alpha = \zeta \equiv H \leq 1$, el cual se llama comúnmente el exponente de Hurst [10,15]. Este último está relacionado a la dimensión (fractal) local del conteo de cajas de la interfase como $D_B = 2 - H$.

El exponente de Hurst es también un indicador para determinar si un fenómeno o una serie de tiempo presentan un comportamiento fractal y mide la intensidad de dependencia a largo plazo de una serie de tiempo. Se dice que el fenómeno analizado es aleatorio cuando $H = 0.5$ (ruido blanco), que es persistente cuando $0.5 < H < 1$ (existe invariancia de escala asociada a correlaciones positivas a largo plazo), y que es antipersistente cuando $0 < H < 0.5$ (existe invariancia en la escala asociada a correlaciones negativas a largo plazo) [11].

A continuación se describen brevemente algunos métodos de análisis fractal de trazado auto-afín para determinar el valor de *H*.

2.4 MÉTODOS DE TRAZADO AUTO-AFIN

Las señales complejas que analiza el grupo de investigación enfocado al análisis fractal de la dinámica de sistemas complejos, son de tipo fractal auto-afín, ó multifractales, por lo que es necesario aplicar las técnicas específicas conocidas para este tipo de fractales.

2.4.1 ANÁLISIS DEL RANGO ESCALADO (R/S)

El método estadístico del rango reescalado (R/σ) ó (R/S), propuesto por Mandelbrot y Wallis [19] y basado en un previo análisis hidrológico de Hurst [21], es el más antiguo y conocido método para determinar el valor de *H*. Este método permite calcular el parámetro de auto-similitud *H* para medir la intensidad de dependencia a largo plazo de una serie de tiempo.

Para una serie de tiempo de longitud *n*, $X = \{X_t : t = 1,2,...,n\}$, *R/σ* está definido como el cociente del rango máximo normalizado de la señal integrada *R(n)* entre la desviación estándar *S(n)*:

$$\frac{R(n)}{S(n)} = \frac{\max\{0, r_t : t = 1,2,...,n\} - \min\{0, r_t : t = 1,2,...,n\}}{\sqrt{S^2(n)}}, \qquad .....(2.5)$$

donde

$\max\{0, r_t : t = 1,2,...,n\} - \min\{0, r_t : t = 1,2,...,n\}$ Es el recorrido de los valores

$$r_k = \sum_{t=1}^{k} X_t - \frac{k}{n}\sum_{t=1}^{n} X_t \qquad ... \qquad ..... \ (2.6)$$

Es el valor máximo menos el mínimo

y $$S(n) = \left[\frac{1}{n}\sum_{t=1}^{n}\left(X_t - \frac{1}{n}\sum_{t=1}^{n} X_t\right)^2\right]^{1/2}. \qquad .....(2.7)$$

Es la desviación estándar

Una medición confiable de *S(n),* o σ, requiere de una muestra de datos con un intervalo constante d_n, ya que la diferencia esperada entre los valores constantes de *X* es una función de la distancia que separa a éstas.

Si el trazo es auto-afín, el valor esperado de *R/σ* tiene un escalamiento n^H cuando $n \to \infty$:

$$R/\sigma \propto \tau^H . \qquad .....(2.8)$$

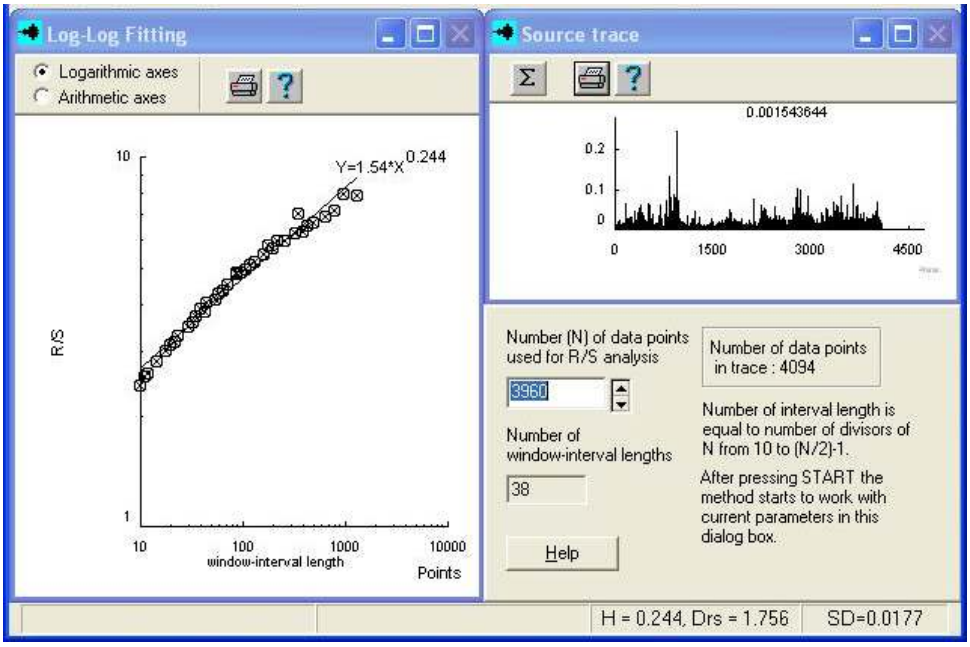


Figura 2.7 Estimación del Exponente de Hurst y la dimensión fractal por el método R/S, tomada del Software Benoit 1.2

La exactitud en la determinación de *H*, en todos los intervalos de tiempo, depende del número de datos que sean utilizados en el cálculo. Si dicho número es razonablemente grande (es decir, cuando el intervalo del tiempo máximo es trazado varias veces), se espera que la curva R/σ proporcione información sobre la auto-similitud de todos los intervalos de tiempo. Si el tiempo registrado posee sólo correlaciones a corto plazo, entonces la gráfica log-log de es también una línea recta con pendiente de 0.5.

2.4.2 MÉTODO DE RUGOSIDAD-LONGITUD (R/L)

Este método es muy parecido al R/σ. En el método de rugosidad-longitud (SD) se toma en cuenta la desviación estándar, o rugosidad de la raíz cuadrada de la media de los datos, en las escalas de tiempo τ, en vez del rango vertical [4]. Para un trazado auto-afín, la rugosidad *SD*, (donde SD es la desviación estándar) medida en una escala de tiempo τ, está relacionada con *H* como:

$$SD \propto \tau^{H}. \qquad .....(2.9)$$

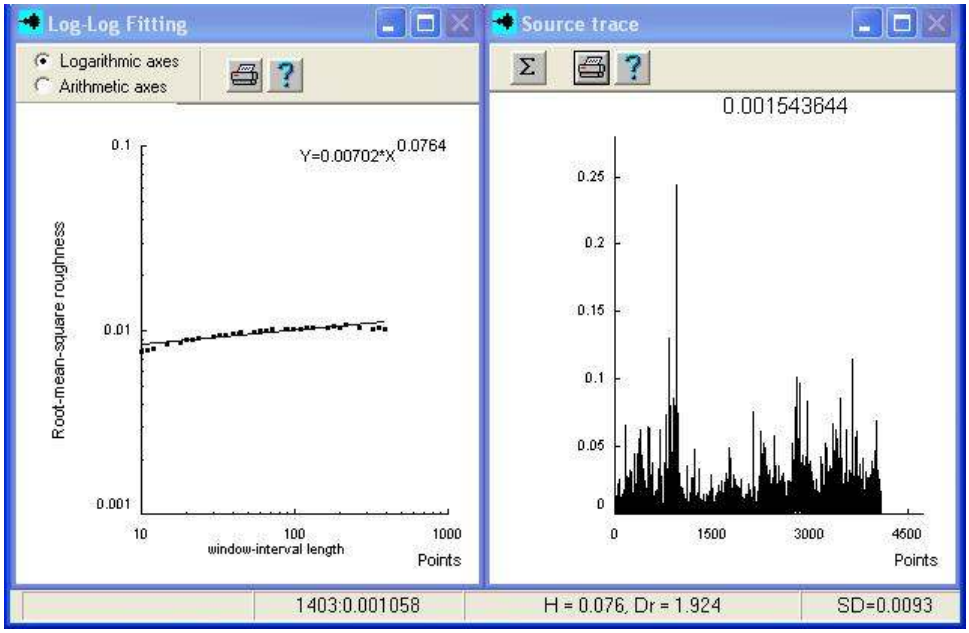


Figura 2.8 Estimación del Exponente de Hurst y la dimensión fractal por el método (R/L), tomada del Software Benoit 1.2

2.4.3 MÉTODO DEL VARIOGRAMA (Vg)

La distribución del tiempo de X_i puede ser caracterizada por una función de *semivarianza*, llamada *variograma*, la cual se define como:

$$V(n) = \frac{1}{2N}\sum_{i=0}^{N/2}(X_i - X_{i+n})^2, \qquad .....(2.10)$$

donde *N* es el número de parejas de puntos separados por un intervalo *n* (el desfasamiento del intervalo).

Y V(n) es la función variograma de la serie.

Cuando la *semivarianza* estimada es graficada contra *n*, ésta puede aproximarse *asintóticamente* a un valor constante (umbral) o puede incrementarse sin límites conforme aumente *n*. Los *variogramas* sin límites sugieren que la variación está dándose en un rango continuo de escalas de tiempo. Tanto el *variograma* transitivo, con un umbral finito, y los *variogramas* sin límites pueden ser analizados

en una gráfica log-log. Si el logaritmo de una *semivarianza* es graficado contra el logaritmo de *n*, entonces la pendiente es *2H*. Por consiguiente:

$$V \propto \tau^{2H} . \qquad .....(2.11)$$

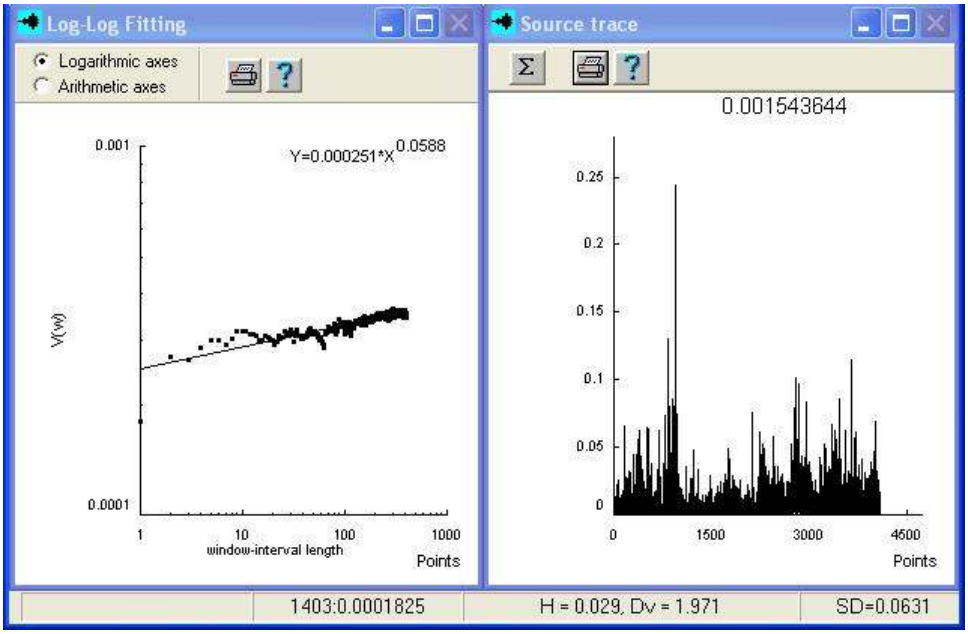


Figura 2.9 Estimación del Exponente de Hurst y la dimensión fractal por el método (Vg) , tomada del Software Benoit 1.2

Un problema con el método del variograma es que el intervalo de la muestra *n* y la determinación de la pendiente pueden afectar la estimación de *H*.

2.4.4 MÉTODO DEL ESPECTRO DE POTENCIA (P/S)

Otro método que determina la existencia de dependencia a largo plazo en las series de tiempo, se basa en la forma espectral de un proceso dependiente a largo plazo. El método del espectro de potencia, que tiene su origen en el análisis espectral, puede ser aplicado a los datos de las series de tiempo. La función de densidad del análisis espectral para datos aleatorios describe los datos en términos de la densidad espectral del valor de su media elevado al cuadrado para diferentes frecuencias. La función de densidad del espectro de potencia es la transformada de *Fourier* de la función de autocorrelación:

$$S(\omega) = C(0) + 2\sum_{n=1}^{\infty} C(n)\cos(2\pi\omega n)\,. \qquad \text{.....(2.12)}$$

Si $C(n)$ obedece a un escalamiento de ley de potencia, entonces

$$S(\omega) \propto \omega^{-\alpha}\,, \qquad \text{.....(2.13)}$$

donde $\alpha = 1 - \beta$ para pequeños ω. Para las series auto-afines tenemos que

$$\alpha = 1 - \beta = 1 + 2H. \qquad \text{.....(2.14)}$$

Por lo tanto, tenemos:

$$P \propto \tau^{-2H-1}\,. \qquad \text{.....(2.15)}$$

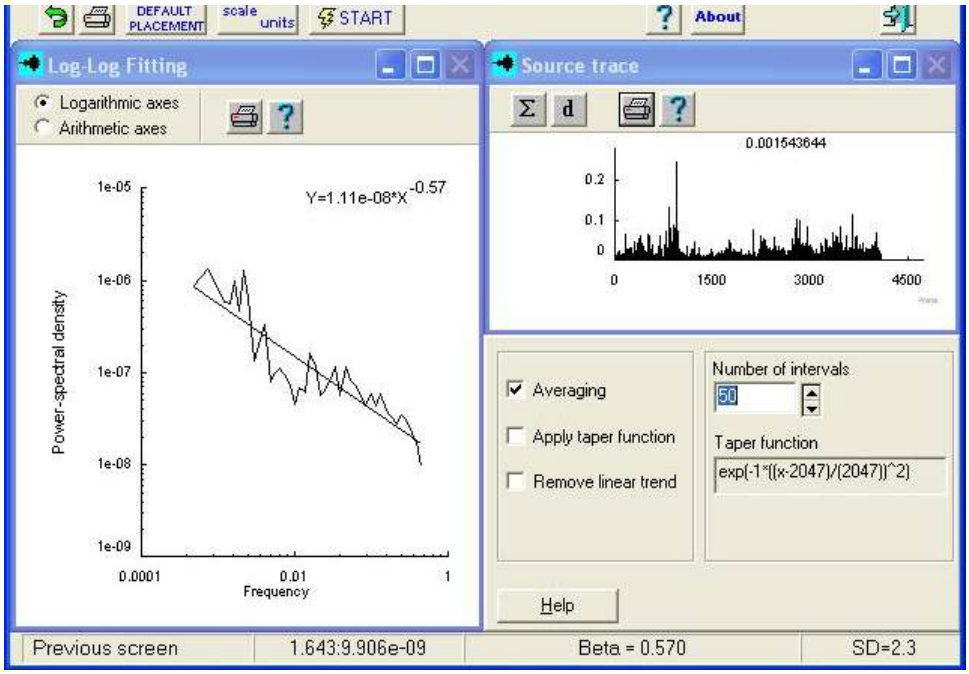


Figura 2.10 Análisis del espectro de potencias (P/S), tomada del Software Benoit 1.2

2.4.5 MÉTODO DE LAS ONDOLETAS Ú ONDULACIONES (Wv)

Las ondoletas o *Wavelets* son una extensión del análisis de *Fourier*, y la transformada de las ondoletas es computacionalmente similar, en principio, a la transformada rápida de *Fourier* (*FFT: Fast Fourier transform*). La *FFT* utiliza cosenos, senos y exponentes para representar una señal, y es la más usada para analizar funciones lineales. A partir de que se sabe que muchas series de tiempo despliegan un comportamiento caótico no lineal, el análisis de *Fourier* es menos apropiado para

analizar dichas series. El objetivo de la transformada de las ondoletas es el de expresar una señal de entrada en una serie de exponentes de "energía" especificada. Los números discretos, asociados con cada exponente, contienen toda la información necesaria para describir completamente la serie, en la que uno conoce qué análisis de ondoletas fue empleado para la descomposición.

El método del exponente promedio de la ondoleta (*AWC, the Average Wavelet Coefficient Method*) utiliza la transformada de la ondoleta para medir la auto-afinidad temporal de las correlaciones; en otras palabras, es el método para *H* [21]. Esto se hace mediante la transformación de la serie de tiempo, *X(t)*, al dominio de las ondoletas $W[X](a;b)$, donde a denota un parámetro de escala y b indica trazados. El método AWC consiste, para una escala dada de a, en encontrar una "energía" (ondoleta) representativa o amplitud a una escala específica. Esto puede ser realizado al tomar el promedio aritmético de $W[X](a;b)$ sobre todos los trazados del parámetro b, correspondiente este último a un valor de a en la misma escala. Se puede, por ende, construir, a partir de la transformada de la ondoleta de *X(t)*, el espectro de AWC $W[X](a)$, que sólo depende de la escala a, y que fue definido anteriormente. Si *X(t)* es un proceso auto-afín, caracterizado por *H*, este espectro se escalaría como sigue:

$$W[X](a) = \left\langle |W(a,b)| \right\rangle_b \propto a^{H+1/2}. \qquad \text{.....(2.16)}$$

Al graficar $W[X](a)$, en una escala log-log, la pendiente debería ser *H* + ½, si la señal es auto-afín (de multi-escala, en el sentido de que el comportamiento a diferentes escalas no tiene influencia de ninguna manera significativa; o sea, el método desacopla escalas).

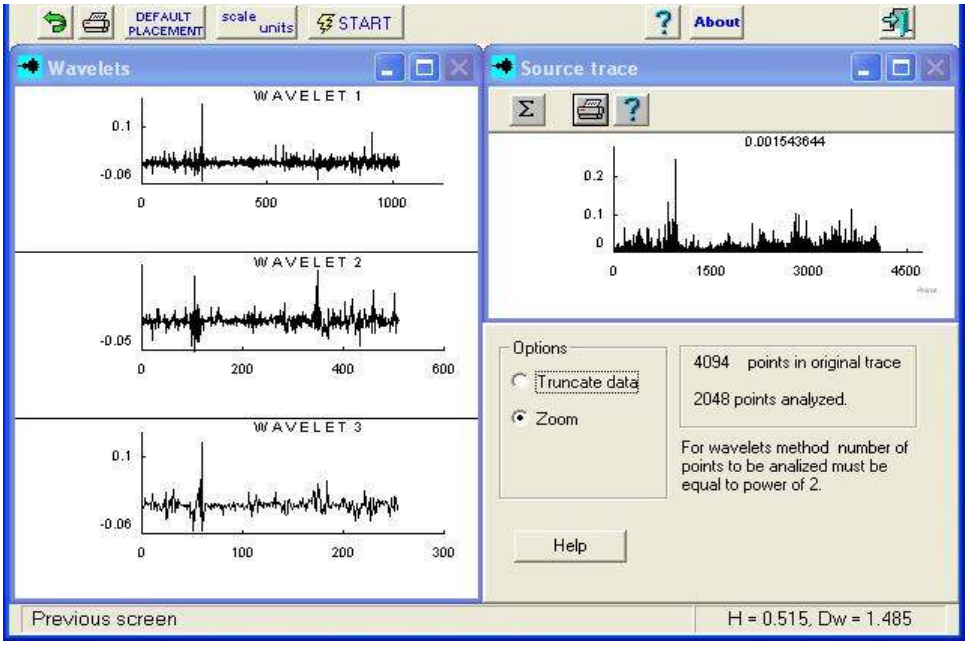


Figura 2.11 Estimación del Exponente de Hurst y la dimensión fractal por el método (Wv) , tomada del Software Benoit 1.2

El análisis de ondoletas es una herramienta para analizar cambios de potencia localizadas, mediante la descomposición de una serie de tiempo en el espacio de frecuencia de tiempo, a fin de determinar los modos dominantes de la variación y cómo éstos se alteran con el tiempo. Este método es apropiado para el análisis de series de tiempo no estacionarias; es decir, en donde la varianza no permanece constante con el tamaño de la muestra de datos analizados. Las propiedades fractales están presentes donde el espectro de potencia de la ondoleta es una función de ley de potencia de la frecuencia.

CAPÍTULO 3
TÉCNICAS DE PROGRAMACIÓN PARALELA

Una vez que se ha decidido construir un algoritmo computacional paralelo, es importante tener el conocimiento de cómo se puede construir dicho algoritmo. El simple hecho de tocar el tema de la programación o construcción de algoritmos computacionales, nos remite a pensar en un algoritmo en secuencia estructurado paso por paso, donde el flujo de las operaciones lleva un orden de arriba hacia abajo; si embargo, como lo veremos más adelante se tiene que pensar en una forma de programación y construcción de la solución de forma diferente. Algo conocido como el paradigma de la programación, permitirá distinguir que la programación basada en técnicas paralelas, distará mucho de la programación estructurada y secuencial; además es importante recalcar que no por ser un algoritmo paralelo se puede pensar en que sea una solución óptima, pues se tiene que determinar, según las características físicas de la herramienta computacional su funcionamiento, su forma de almacenamiento, accesos y administración, que puedan ser adaptables al conjunto de herramientas de software disponibles y que puedan ser capaces de cumplir con las expectativas para la construcción de la solución adecuada.

3.1 LENGUAJES DE PROGRAMACIÓN

Comencemos por definir de forma general lo que es un lenguaje de programación; el cual puede ser considerado como lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora; consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Aunque muchas veces se usa lenguaje de programación y lenguaje informático como si fuesen sinónimos, no tiene por qué ser así, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como, por ejemplo, el *HTML* (*lenguaje para el marcado de páginas Web*).

Un lenguaje de programación permite a uno o más programadores especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural, tal como sucede con el lenguaje Léxico. Una característica relevante de los lenguajes de programación es precisamente que más

de un programador puedan tener un conjunto común de instrucciones que puedan ser comprendidas entre ellos para realizar la construcción del programa de forma colaborativa.

Los lenguajes de programación pueden clasificarse empleando distintos métodos y puntos de vista. Esta clasificación se basa en el paradigma que utilizan

3.2 PARADIGMAS DE PROGRAMACIÓN

Un paradigma de programación provee (*y determina*) la visión y métodos de un programador en la construcción de un programa o subprograma. Diferentes paradigmas resultan en diferentes estilos de programación y en diferentes formas de pensar la solución de problemas (con la solución de múltiples "problemas" se construye una aplicación).

3.2.1 PARADIGMAS DE PROGRAMACIÓN PARALELA

Los paradigmas de programación no están unívocamente ligados a las arquitecturas paralelas, aunque sí que guarden relación como se indicará, existen dos paradigmas principales de programar computadores paralelos:

- Programación paralela mediante memoria compartida (espacio de direcciones único)

  El programador escribe el programa, bien empleando directivas de paralelismo de datos bien asumiendo explícitamente diferentes procesos que pueden acceder a un espacio de direcciones único, con variables globales compartidas.

  Este paradigma es apropiado para arquitecturas de memoria compartida, aunque el espacio de direcciones único también puede ser simulado sobre arquitecturas de memoria distribuida.

- Programación paralela mediante paso de mensajes

  El programador explícitamente divide el trabajo y los datos entre varios procesos y debe gestionar la comunicación de datos entre ellos.

  Esta aproximación es completamente flexible, si bien requiere un mayor trabajo del programador. Puede ser empleado en arquitecturas de memoria distribuida e incluso en redes de computadoras, y también en arquitecturas de memoria

compartida, aunque en ese caso los diferentes procesos utilizan partes de memoria independientes y separadas.

3.3 ENTORNOS DE PROGRAMACIÓN PARALELA

Al implementar un algoritmo paralelo, se requiere construir un programa paralelo. El entorno en el que los programas paralelos se construyen, se conoce como Entorno de Programación Paralela.

Existen muchos entornos y clasificaciones de programación paralela. Para entender sus características es apropiado organizarlos en torno de una categoría según sus modelos de programación:

MODELOS FORMALES		
CLASIFICACIÓN	DESCRIPCIÓN	EJEMPLOS
MODELOS DE PROGRAMACIÓN LÓGICA	Basados en programación lógica declarativa. Se basan en el cálculo de primer orden del predicado de Prolog. La concurrencia se define de tres maneras: el "y-paralelismo" (se ejecutan múltiples predicados), el "o-paralelismo" (se ejecutan múltiples protectores), y con mapeo de predicados ligados a variables de una sola asignación.	• Parlog • Strand
MODELOS DE PROGRAMACIÓN FUNCIONAL	Utilizan la semántica declarativa y una cierta forma de cálculo de la lambda para expresar la operación de un programa. LISP es quizás el más conocido. Al ejecutarse una función tan pronto como los datos son requeridos, será capaz de proporcionar una manera natural de llevar a cabo la ejecución concurrente.	• Haskell • Sisal
MODELOS COMPOSICIO-NALES	Estos modelos se basan en distinguir explícitamente entre las formas en las que los programas se componen (juntado ciertas partes para formar programas más grandes). En este contexto, las formas pertinentes de composición son secuenciales (en qué programas se ejecuta en secuencia) y paralelas (en qué programas se ejecutan "concurrentemente", donde la ejecución concurrente se modela con frecuencia como ejecución interpolada de los elementos de la composición).	• CC++ • Fortran M • PCN
MODELOS INFORMALES Estos modelos son pragmáticos y motivados por las experiencias de programadores en comparación con las teorías computacionales. Estos modelos no clasifican tan cuidadosamente como los modelos formales, y habrá un cierto traslape entre las categorías. Por ejemplo, una biblioteca de MPI se puede utilizar para escribir algoritmos paralelos procesamiento de múltiples datos (SPMD).		
MODELOS DE DATOS PARALELOS	Modela la visión de cómputo paralelo en términos de un solo flujo de instrucciones que funciona en múltiples conjuntos de datos. Los conjuntos múltiples de datos, proporcionan la fuente de la concurrencia.	• C* • Data Parallel C • HPF

MODELOS COORDINADOS BASADOS EN COMUNICACIÓN	La comunicación entre las tareas concurrentes ocurre con el intercambio de mensajes o algunos eventos discretos de comunicación. La semántica para la comunicación proporciona la sincronización.	• CHARM • MPI • PVM • TCGMSG
MODELOS COORDINADOS BASADOS EN MEMORIA COMPARTIDA	La coordinación es proporcionada por la semántica de las operaciones en las estructuras de datos compartidas.	• GA • Linda • SDDA
MODELOS ORIENTADOS A OBJETOS	Los objetos se utilizan normalmente para proporcionar una abstracción de los datos. Sin embargo, el mismo concepto básico se puede utilizar para proporcionar abstracción de la concurrencia. Un sistema orientado a objetos proporciona un marco dentro de el cual un programa paralelo se construye.	• HPC++ • Mentat • CHARM++ • POOMA
MODELOS DE MEMORIA COMPARTIDA CON TAREAS EXPLÍCITAS	Están basados en la idea de procesos muy ligeros que comparten un espacio de dirección. Estos procesos se llaman los hilos. Puesto que se comparte el espacio de dirección, estos sistemas ignoran la comunicación y se centran generalmente las formas de controlar las relaciones entre los hilos y proteger el orden del acceso a los datos. La creación y la interacción entre los hilos son explícitas en estos sistemas.	• ACE • Java • Pthreads • Threads.h++ • Win32
MODELOS DE MEMORIA COMPARTIDA CON TAREAS IMPLÍCITAS	Algunos sistemas de memoria compartida utilizan constructores de alto nivel en comparación con los hilos controlados por la ejecución. Por ejemplo, un sistema pudo expresar concurrencia en términos de iteraciones distribuidas o secciones paralelas.	• OpenMP • PAMS • Sthreads

Tabla 3.1 Clasificación de los entornos de programación paralela, propuesta por el Departamento de Ingeniería y Ciencias de la Computación de la Universidad de Florida, E.U.A[3]

3.4 ARQUITECTURAS DE PROCESAMIENTO PARALELO

En 1966 Michael Flynn propuso un mecanismo de clasificación de las computadoras. La taxonomía de *Flynn* es la manera clásica de organizar las computadoras, y aunque no cubre todas las posibles arquitecturas, proporciona una importante penetración en varias arquitecturas de computadoras. El método de *Flynn* se basa en el número de instrucciones y de la secuencia de datos que la computadora utiliza para procesar información. Puede haber secuencias de instrucciones sencillas o múltiples y secuencias de datos sencillas o múltiples. Esto da lugar a cuatro tipos de computadoras, de las cuales solamente dos son aplicables a las computadoras paralelas [23]:

[3] Departamento de Ingeniería y Ciencias de la Computación de la Universidad de Florida, E.U.A http://www.cise.ufl.edu/. Último acceso, 4 de junio del 2008.

1.- SISD (Single Instruction Single Data).

Este es el modelo tradicional de computación secuencial donde una unidad de procesamiento recibe una sola secuencia de instrucciones que operan en una secuencia de datos.

Ejemplo: Para procesar la suma de N números a1, a2, ... aN, el procesador necesita accesar a memoria N veces consecutivas (para recibir un número). También son ejecutadas en secuencia N-1 adiciones. Es decir los algoritmos para las computadoras SISD no contienen ningún paralelismo, éstas están constituidas de un procesador.

2.- SIMD (Single Instruction Multiple Data).

A diferencia de SISD, en este caso se tienen múltiples procesadores que sincronizadamente ejecutan la misma secuencia de instrucciones, pero en diferentes datos. El tipo de memoria que estos sistemas utilizan es distribuida.

Aquí hay N secuencias de datos, una por procesador, así que diferentes datos pueden ser utilizados en cada procesador. Los procesadores operan sincronizadamente y un reloj global se utiliza para asegurar esta operación. Es decir, en cada paso todos lo procesadores ejecutan la misma instrucción, cada uno en diferente dato.

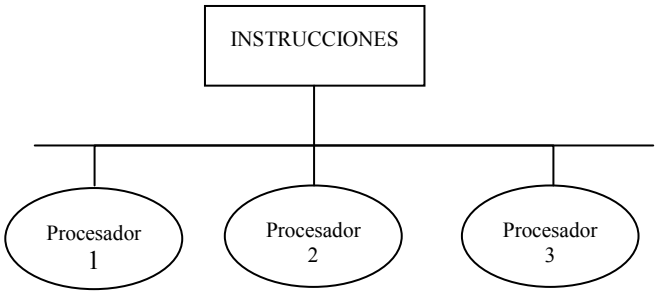


Figura 3.1 Modelo SIMD(Single Instruction Multiple Data), tomada de la página del departamento de Cómputo del CICESE, Ensenada México, http://telematica.cicese.mx/computo/.

Máquinas con arreglos de procesadores tales como *ICL DAP* (*Distributed Array Processor*) y computadoras vectoriales canalizadas como *CRAY 1 & 2* y *CIBER 205* son de arquitectura SIMD.

Ejemplo: Sumando dos matrices A + B = C. Siendo A y B de orden 2 y teniendo 4 procesadores:

A11 + B11 = C11 A12 + B12 = C12

A21 + B21 = C21 A22 + B22 = C22

La misma instrucción es ejecutada en los 4 procesadores (sumando dos números) y los 4 ejecutan las instrucciones simultáneamente. Esto toma un paso en comparación con cuatro pasos en una máquina secuencial.

3.- MIMD (Multiple Instruction Multiple Data).

Este tipo de computadora es paralela al igual que las SIMD, la diferencia con estos sistemas es que MIMD es asíncrono. No tiene un reloj central. Cada procesador en un sistema MIMD puede ejecutar su propia secuencia de instrucciones y tener sus propios datos. Esta característica es la más general y poderosa de esta clasificación.

Se tienen N procesadores, N secuencias de instrucciones y N secuencias de datos. Cada procesador opera bajo el control de una secuencia de instrucciones, ejecutada por su propia unidad de control, es decir cada procesador es capaz de ejecutar su propio programa con diferentes datos. Esto significa que los procesadores operan asíncronamente, o en términos simples, pueden estar haciendo diferentes cosas en diferentes datos al mismo tiempo.

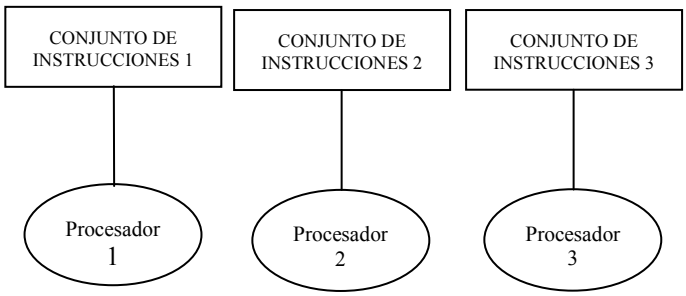


Figura 3.2 Modelo MIMD(Multiple Instruction Multiple Data), tomada de la página del departamento de Cómputo del CICESE, Ensenada México, http://telematica.cicese.mx/computo/.

Los sistemas MIMD se clasifican en tres:

- ***Sistemas de Memoria Compartida.***

En este tipo de sistemas cada procesador tiene acceso a toda la memoria, es decir hay un espacio de direccionamiento compartido. Se tienen tiempos de acceso a memoria uniformes ya que todos los procesadores se encuentran igualmente comunicados con la memoria principal y las lecturas y escrituras de todos los procesadores tienen exactamente las mismas latencias; y además el acceso a memoria es por medio de un ducto o canal común. En esta configuración, debe asegurarse que los procesadores no tengan acceso simultáneamente a regiones de memoria de una manera en la que pueda ocurrir algún error.

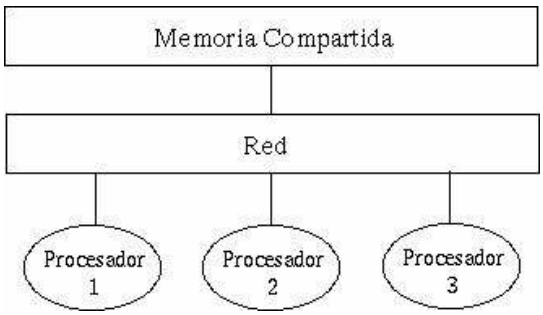


*Figura 3.3 Sistema MIMD de Memoria Compartida, tomada de la página del departamento de Cómputo del CICESE, Ensenada Méx*ico, *http://telematica.cicese.mx/computo/*.

Desventajas:

- ☆ El acceso simultáneo a memoria es un problema.
- ☆ Poca escabilidad de procesadores, debido a que se puede generar un cuello de botella al incrementar el numero de CPU's.
- ☆ En computadoras vectoriales como Crays, etc. Todos los CPUs tienen un camino libre a la memoria y No hay interferencia entre CPUs.
- ☆ La razón principal por el alto precio de Cray es la memoria.
- ☆ Ventaja:
- ☆ La facilidad de la programación. Es mucho más fácil programar en estos sistemas que en sistemas de memoria distribuida.

Las computadoras MIMD con memoria compartida son sistemas conocidos como de multiprocesamiento simétrico (SMP) donde múltiples procesadores comparten un mismo sistema operativo y memoria. Otro término con que se le conoce es máquinas firmemente juntas o de multiprocesadores. Ejemplos son: *SGI/Cray Power Challenge, SGI/Cray C90, SGI/Onyx, ENCORE, MULTIMAX, SEQUENT y BALANCE*, entre otras.

- ***Sistemas de Memoria Distribuida.***

Estos sistemas tienen su propia memoria local. Los procesadores pueden compartir información solamente enviando mensajes, es decir, si un procesador requiere los datos contenidos en la memoria de otro procesador, deberá enviar un mensaje solicitándolos. Esta comunicación se le conoce como Paso de Mensajes.

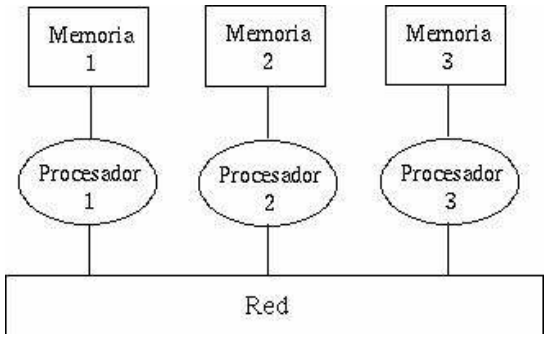


Figura 3.4 Sistema MIMD de Memoria Distribuída, tomada de la página del departamento de Cómputo del CICESE, Ensenada México, http://telematica.cicese.mx/computo/

Ventajas:

- ☆ La escalabilidad. Las computadoras con sistemas de memoria distribuida son fáciles de escalar, mientras que la demanda de los recursos crece, se puede agregar más memoria y procesadores.

Desventajas:

- ☆ El acceso remoto a memoria es lento.
- ☆ La programación puede ser complicada.

Las computadoras MIMD de memoria distribuida son conocidas como sistemas de procesamiento en paralelo masivo (MPP) donde múltiples procesadores trabajan en

diferentes partes de un programa, usando su propio sistema operativo y memoria. Además se les llama multicomputadoras, máquinas libremente juntas o cluster. Algunos ejemplos de este tipo de máquinas son IBM SP2 y SGI/Cray T3D/T3E.

- ***Sistemas de Memoria Compartida Distribuida.***

Es un cluster o una partición de procesadores que tienen acceso a una memoria compartida común pero sin un canal compartido. Esto es, físicamente cada procesador posee su memoria local y se interconecta con otros procesadores por medio de un dispositivo de alta velocidad, y todos ven las memorias de cada uno como un espacio de direcciones globales.

El acceso a la memoria de diferentes clusters se realiza bajo el esquema de Acceso a Memoria No Uniforme (*NUMA*), la cual toma menos tiempo en accesar a la memoria local de un procesador que accesar a memoria remota de otro procesador.

Ventajas:

- ☆ Presenta escalabilidad como en los sistemas de memoria distribuida.
- ☆ Es fácil de programar como en los sistemas de memoria compartida.
- ☆ No existe el cuello de botella que se puede dar en máquinas de sólo memoria compartida.

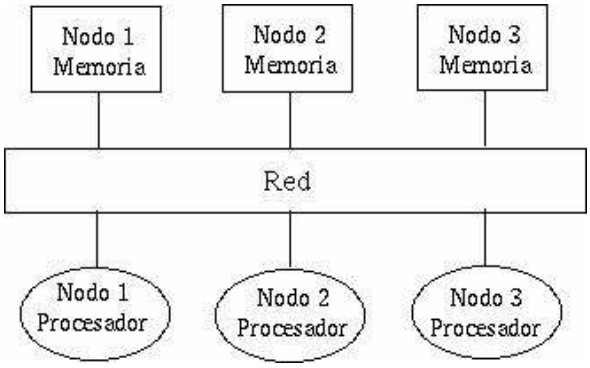


Figura 3.5 Sistema MIMD de Memoria Compartida Distribuida, tomada de la página del departamento de Cómputo del CICESE, Ensenada México, http://telematica.cicese.mx/computo/.

Algunos ejemplos de este tipo de sistemas son *HP/Convex SPP-2000 y SGI/Cray Origin2000*.

4.- MISD (Multiple Instruction Single Data).

Aquí hay N secuencias de instrucciones (algoritmos/programas) y una secuencia de datos. El paralelismo es alcanzado dejando que los procesadores realicen diferentes cosas al mismo tiempo en el mismo dato.

Las máquinas MISD son útiles en cómputos donde la misma entrada esta sujeta a diferentes operaciones.

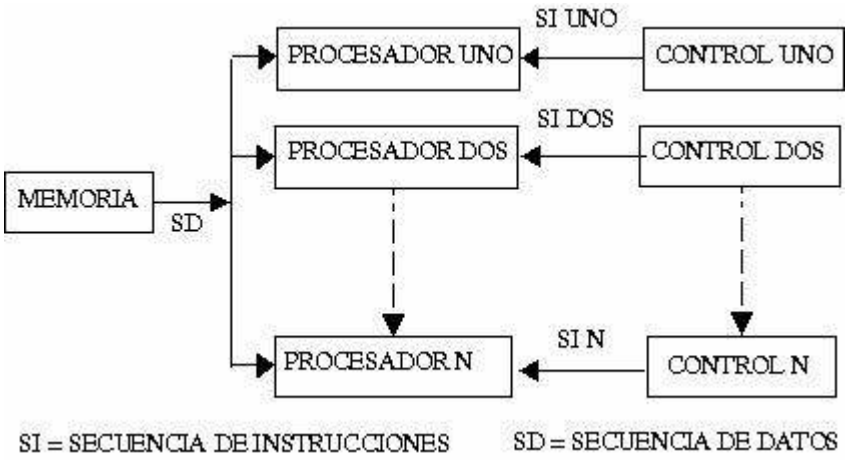


Figura 3.6 Sistema MISD (Multiple Instruction Single Data), tomada de la página del departamento de Cómputo del CICESE, Ensenada México, http://telematica.cicese.mx/computo/.

3.5 MPI (INTERFACE DE PASO DE MENSAJES/MESSAGE PASSING INTERFACE)

MPI no es un lenguaje de programación sino un conjunto de funciones y macros que conforman una biblioteca estándar de C y C++, y subrutinas en Fortran.

MPI ofrece una interfaz de programación de aplicaciones (*API*), junto con especificaciones de sintaxis y semántica que explican como sus funcionalidades deben añadirse en cada implementación que se realice (tal como almacenamiento de mensajes o requerimientos para entrega de mensajes). MPI incluye operaciones punto a punto y colectivas, todas destinadas a un grupo específico de procesos.

MPI proporciona muchas características encaminadas a mejorar el rendimiento de los programas, como lo son la portabilidad, la estandarización y la implementación eficiente de la ejecución paralela.

El envío de un mensaje de un proceso a otro involucra el movimiento de información de un espacio de direccionamiento a otro.

proceso 1 movimiento de datos proceso 2
send(&x,2); ----------> recv(&y,1);

El envío de un mensaje involucra además:

a) El uso de los *buffers*.

b) Identificación de los mensajes, lo cual se hace a través de una selección por medio de etiquetas, además necesita del uso de comodines (*wildcards*) para seleccionar cualquier tipo de mensaje.

c) Nombramiento de procesos.

d) Sincronización .

Existen dos términos básicos utilizados en el MPI:

- Síncrono que se utiliza en aquellas rutinas que regresan cuando la transferencia del mensaje ha terminado.
- Bloqueo (*blocking*), que se usa para describir funciones que no regresan hasta que termina la transferencia.

Las funciones de no bloqueo (*non blocking*) inician la solicitud de transferencia y regresa el control sin esperar que la Transferencia concluya

El paso de mensajes es una tarea ampliamente usada en ciertas clases de máquinas paralelas, especialmente aquellas que cuentan con memoria distribuida. Aunque existen muchas variaciones, el concepto básico en el proceso de comunicación mediante mensajes es bien entendido. En los últimos 10 años, se ha logrado un progreso substancial en convertir aplicaciones significativas hacia este tipo de tareas. Más recientemente diferentes sistemas han demostrado que un

sistema de paso de mensajes puede ser implementado eficientemente y con un alto grado de portabilidad.

Al diseñar el lenguaje estándar por facto llamado MPI, se tomaron en cuenta las características más atractivas de los sistemas existentes para el paso de mensajes, en vez de seleccionar uno sólo de ellos y adoptarlo como el estándar. Resultando así, en una fuerte influencia para en la construcción de MPI los trabajos hechos por *IBM, INTEL NX/2, Express, nCUBE's Vernex, p4 y PARMACS.* Otras contribuciones importantes provienen de *Zipcode, Chimp, PVM, Chameleon y PICL*.

La meta de MPI fue la de desarrollar un estándar para escribir programas que implementen el paso de mensajes. Por lo cual la Interfase intenta establecer para esto un estándar práctico, portable, eficiente y flexible. El esfuerzo para estandarizar MPI involucra cerca de 60 personas de 40 organizaciones diferentes principalmente de Estados Unidos y Europa. La mayoría de los vendedores de computadoras concurrentes estaban involucrados con MPI, así como con investigadores de diferentes universidades, laboratorios del gobierno e industrias. Se llegó a una propuesta preliminar conocida como MPI1, enfocada principalmente en comunicaciones punto a punto sin incluir rutinas para Comunicación colectiva y no presentaba tareas seguras.

ESTANDARIZACIÓN DEL MPI

MPI se ha popularizado como un estándar en la programación paralela. Los programadores no se deben preocupar por la versión del sistema operativo o la máquina en la cual están ejecutando, ya que las llamadas MPI se comportan de igual manera a pesar de la implementación que se utilice.

El estándar final para el MPI fue presentado en la conferencia de supercomputación en Noviembre de 1993, constituyéndose así el foro para el MPI.

En un ambiente de comunicación con memoria distribuida en la cual las rutinas de nivel más alto y/o las abstracciones son construidas sobre rutinas de paso de mensajes de nivel bajo, los beneficios de la estandarización son muy notorios. La principal ventaja al establecer un estándar para el paso de mensajes es la

portabilidad y el ser fácil de utilizar. MPI es un sistema complejo, el cual comprende 129 funciones, de las cuales la mayoría tienen muchos parámetros y variantes.

Con esto, se alcanzaron las siguientes características en el diseño de estándar:

- Diseñar una Interfase de programación aplicable.
- Permite una Comunicación eficiente: Evitando el copiar de memoria a memoria y permitiendo la sobreposición de computación y comunicación, además de aligerar la comunicación con el procesador.
- Permite implementaciones que puedan ser utilizadas en un ambiente heterogéneo.
- Permite enlaces convenientes en C y Fortran 77 para la interfase.
- Asume una interfase de comunicación segura.
- Define una interfase que no sea muy diferente a los sistemas actuales, tales como *PVM, NX, Express, p4*, etc., y provee de diversas extensiones que permitan mayor flexibilidad.
- Define una interfase que pueda ser implementada en diferentes plataformas, sin cambios significativos en el software y las funciones internas de comunicación.
- La semántica de la interfase debe ser independiente del lenguaje.
- La Interfase debe ser diseñada para producir tareas seguras.

PORTABILIDAD

Debido a los esfuerzos encaminados, MPI tiene desarrolladores en muchas plataformas, que basados en los lineamientos descritos en el estándar, desarrollan las bibliotecas que proporcionen esa portabilidad [24].

POTENCIAL

MPI proporciona subrutinas para comunicación asíncrona, comunicación colectiva, topologías virtuales, manejo de mensajes vía buffer, grupos de comunicación, por mencionar algunas; proporcionándole una potencialidad y flexibilidad atractiva para los programadores de aplicaciones de cómputo paralelo [24].

En el modelo de programación MPI, un cómputo comprende de uno o más procesos comunicados a través de llamadas a rutinas de biblioteca para mandar (*send*) y recibir (*receive*) mensajes a otros procesos.

En la mayoría de las implementaciones de MPI, se crea un conjunto fijo de procesos al inicializar el programa, y un proceso es creado por cada tarea. Sin embargo, estos procesos pueden ejecutar diferentes programas.

De ahí que, el modelo de programación MPI es algunas veces referido como MIMD (*multiple program multiple data*) para distinguirlo del modelo SIMD, en el cual cada procesador ejecuta el mismo programa.

Debido a que el número de procesos en un sistema de cómputo de MPI es normalmente fijo, se puede enfatizar en el uso de los mecanismos para comunicar datos entre procesos. Los procesos pueden utilizar operaciones de Comunicación punto a punto para mandar mensajes de un proceso a otro, estas operaciones pueden ser usadas para implementar comunicaciones locales y no estructuradas. Un grupo de procesos puede llamar colectivamente operaciones de Comunicación para realizar tareas globales tales como *broadcast*, etc. La habilidad de MPI para probar mensajes da como resultado el soportar comunicaciones asíncronas. Probablemente una de las características más importantes del MPI es el soporte para la programación modular. Un mecanismo llamado comunicador permite al programador del MPI definir módulos que encapsulan estructuras internas de comunicación (estos módulos pueden ser combinados secuencialmente y paralelamente).

ESTRUCTURA BÁSICA DE UN PROGRAMA PARALELO CON MPI

En resumen los programas de MPI siguen, al menos estos pasos generales y las funciones que se pueden utilizar:

1. Inicializar la comunicación. La función *MPI_Init()* inicializa el ambiente de programación MPI.
2. Comunicar para compartir datos entre procesos. Al invocar la función *MPI_Comm_size()* retorna el número de procesos. Y *MPI_Comm_rank()* regresa como resultado el número de cada procesador. *MPI_Send()* premite enviar un mensaje y *MPI_Recv()* permite recibir un mensaje.

3. Finalizar el ambiente paralelo con la función *MPI_Finalize()* [24].

GRUPO DE FUNCIONES

Aunque MPI es un sistema complejo, es posible resolver un amplio rango de problemas usando seis de sus funciones, estas funciones inician y terminan un cómputo, identifican procesos, además de mandar y recibir mensajes.

- *MPI_INIT*: Este proceso Inicia el entorno de MPI.
- *MPI_FINALIZE*: Termina el MPI.
- *MPI_COMM_SIZE:* Determina el número de procesos en un cómputo.
- *MPI_COMM_RANK*: Determina el identificador del proceso actual "mi proceso".
- *MPI_SEND*: Manda un mensaje.
- *MPI_RECV:* Recibe un mensaje.

Todas las funciones con excepción de las dos primeras, toman un manejador "comunicador" como argumento. El comunicador identifica el grupo de procesos y el contexto en el cual la operación se debe realizar. Los comunicadores proveen un mecanismo para identificar subconjuntos de procesos durante el desarrollo de programas modulares y para garantizar que los mensajes provistos con diferentes propósitos no sean confundidos. El valor por omisión es llamado *MPI_COMM_WORLD*, el cual identifica todos los procesos [25].

Las funciones *MPI_INIT* y *MPI_FINALIZE* son usadas para iniciar y terminar MPI, respectivamente *MPI_INIT* debe ser llamada antes que cualquier otra función MPI y debe ser llamada solamente una vez por proceso. Ninguna función MPI puede ser llamada después de *MPI_FINALIZE*. Las funciones *MPI_COMM_SIZE* y *MPI_COMM_RANK* determinan el número de procesos en él cómputo actual y el identificador (entero) asignado al proceso actual, respectivamente. (Los procesos en un grupo de procesos son identificados con un único y continuo número (entero) empezado en 0).

La necesidad por tener una comunicación asíncrona puede presentarse cuando un cómputo necesita acceder a los elementos de un dato estructurado compartido en una manera no estructurada. Una implementación aproximada es el encapsular los datos estructurados en un conjunto de tareas de datos

especializados, en la cual las peticiones de lectura y escritura pueden ser ejecutadas. Este método no es eficiente en MPI debido a su modelo de programación MPMD.

Una implementación alternativa con MPI, es el distribuir las estructuras de datos compartidas entre los procesos existentes, los cuales deben solicitar periódicamente las solicitudes pendientes de lectura y escritura. Para esto MPI presenta tres funciones *MPI_IPROBE*, *MPI_PROBE*, *MPI_GET_COUNT*.

MPI_IPROBE checa la existencia de mensajes pendientes sin recibirlos, permitiéndonos escribir programas que generan cómputos locales con el procesamiento de mensajes sin previo aviso. El mensaje puede ser recibido usando *MPI_RECV*.

MPI_PROBE es utilizado para recibir mensajes de los cuales se tiene información incompleta.

MPI soporta la programación modular a través de su mecanismo de comunicador (*comm*, el cual provee la información oculta necesaria al construir un programa modular), al permitir la especificación de componentes de un programa, los cuales encapsulan las operaciones internas de Comunicación y proveen un espacio para el nombre local de los procesos.

Una operación de Comunicación MPI siempre especifica un comunicador. Este identifica el grupo de procesos que están comprometidos en el proceso de comunicación y el contexto en el cual la comunicación ocurre. El grupo de procesos permite a un subconjunto de procesos el comunicarse entre ellos mismos usando identificadores locales de procesos y el ejecutar operaciones de comunicación colectivas sin meter a otros procesos. El contexto forma parte del paquete asociado con el mensaje. Una operación *receive* puede recibir un mensaje sólo si éste fue enviado en el mismo contexto. Si dos rutinas usan diferentes contextos para su Comunicación interna, no puede existir peligro alguno en confundir sus comunicaciones.

Con *MPI_COMM_DUP*: Un programa puede crear un nuevo comunicador, conteniendo el mismo grupo de procesos pero con un nuevo contexto para asegurar que las comunicaciones generadas para diferentes propósitos no sean confundidas, Este mecanismo soporta la composición secuencial.

Usando *MPI_COMM_SPLIT*: Un programa puede crear un nuevo comunicador, conteniendo sólo un subconjunto del grupo de procesos. Estos procesos pueden comunicarse entre ellos sin riesgo de tener conflictos con otros cómputos concurrentes. Este mecanismo soporta la composición paralela.

Aplicando *MPI_INTERCOMM_CREATE*: Un programa puede construir un intercomunicador, el cual enlaza procesos en dos grupos. Soporta la composición paralela.

La función *MPI_COMM_FREE*: Puede ser utilizada para liberar el comunicador creado al usar las funciones anteriores.

TIPOS DE DATOS

Los datos utilizados para la comunicación entre procesadores deben ser del mismo tipo, es decir, tanto la operación de envío como la de recepción deben utilizar el mismo tipo de datos.

Los tipos de datos reconocidos por MPI son definidos en la tabla siguiente [26].

Argc	Número de argumentos en la línea de comandos (C).
Argv	Argumentos en la línea de comandos (C).
Buf	Variable que contiene la información a comunicar.
Comm	Comunicador que incluye el número de cada procesador miembro de un ambiente de trabajo en paralelo. Al iniciar la ejecución de un programa en paralelo, la instrucción MPI_COMM_SIZE regresa un comunicador por default: MPI_COMM_WORLD, el cual contiene los números que corresponden a cada uno de los procesadores que se asignan para la ejecución del programa.
Count	Cantidad de elementos contenidos en buf.
Datatype	Tipo de la variable buf.
Dest	Número lógico del procesador al cual se ha transferido información.
Errorcode	Entero que identifica una situación asociada con error en proceso. Algunas implementaciones de MPI regresan este valor

	como si fuera producto de una instrucción return errorcode.
Ire	Código de error de implementación que es igual a MPI_SUCCESS sí la función termina con éxito; de otra manera, el valor de ierr corresponde a un valor que depende de la implementación de MPI.
Rank	Número lógico del procesador.
Op	Operación a ejecutar.
Outsize	Tamaño de buffer de salida.
position	Posición del último elemento alimentado o recuperado de un buffer.
Recvbuf	Variable que contiene la información a recibir.
recvcount	Cantidad de elementos contenidos en recvbuf.
recvdatatype	Tipo de la variable recvbuf
Request	En combinación con las funciones MPI_TEST y MPI_WAIT proporciona información sobre el estado de una función MPI_ISEND o MPI_IRECV.
sendbuf	Variable que contiene la información a comunicar.
sendcount	Cantidad de elementos contenidos en senbuf.
senddatatype	Tipo de la variable senbuf. Ver tabla II.
Size	Número de procesadores asignados al programa.
Source	Número lógico del procesador que ha enviado información.
Status	Arreglo de tamaño MPI_STATUS_SIZE. Auxiliar en conocer el estado de ejecución de una función MPI.
Tag	Identifica el envío. Generalmente es cero y sólo cambia cuando se ha de comunicar más de un envío.
Target	Número lógico del procesador a quien se envía información.

CAPÍTULO 4
CONSTRUCCIÓN DEL ALGORITMO COMPUTACIONAL PARALELO

Considerando que, de los métodos para la obtención de la dimensión fractal y el exponente de Hurst (H) para los fractales auto-afínes, el más conocido y difundido es el de rango reescalado (R/S), detallemos el conjunto de pasos y métodos que han de seguirse para tal fin. Por lo tanto, cuando se construye el algoritmo, se debe de conocer a detalle los elementos de entrada y de salida que tendrá el procedimiento; en la figura 3.1 se muestra en forma general lo que se desea obtener.

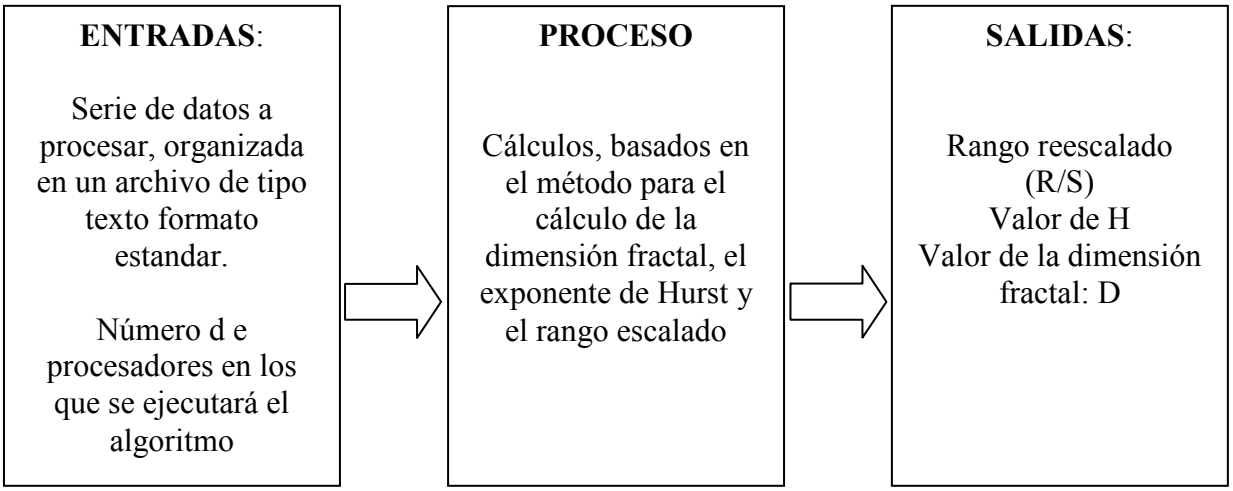


Figura 4.1 Esquema general del algoritmo

ENTRADAS:

La parte principal de entrada del proceso, será el conjunto de datos que se tendrán que procesar, llamada serie de datos N. Dicha serie, son los datos que se requiere sean computados para obtener los resultados esperados. Estos datos pueden ser tan diversos, dependiendo el tipo de datos que se trate; una de las aplicaciones más comunes que se le han dado a este método son en los datos estadísticos reflejados diariamente en los índices financieros, tanto de rendimientos de bolsas de valores como en los índices de precios de productos y servicios. Sin embargo no es privativo del área de las finanzas este tipo de datos. También lo pueden ser los pulsos cardiacos obtenidos en un periodo de tiempo de un paciente, los datos arrojados por un sismógrafo, etc.

N pude ser tan grande como sea necesario; sin embargo no puede ser tan pequeño (menos de 200 datos no es pertinente), ya que en muestras pequeñas, el método de rango escalado R/S no puede evidenciar la existencia correlación entre

los mismos, ya que su naturaleza está determinada por periodos de tiempo largos y que a través de estos periodos se puede reflejar esa correlación que puede determinar o no persistencia en un sistema.

El conjunto de datos a ser capturados se almacena en un archivo de formato de texto estándar ANSI, donde la separación entre uno y otro dato estará determinado por un salto de línea, el cual tendrá como nombre serie.txt.

Por otro lado, se requerirá capturar el número de procesadores a utilizar ya que el algoritmo tendrá la capacidad de ser ejecutado en cualquier supercomputadora que administre n procesadores y entre cada supercomputadora existen diferencias, así como disponibilidad o restricción de procesadores y que cumpla con los requisitos mínimos de ejecución, como lo es contar con las herramientas y bibliotecas de MPI; pues una de las ventajas de este algoritmo es poder ser manejado en cualquier computadora de código abierto; y que su aplicación es general, esto es, que puede correr sin problemas en cualquier cluster de computadoras o supercomputadoras dedicados a cálculos complejos.

En el caso particular en el CIDETEC, se cuenta con un Cluster HPC basado en Linux el cual cuenta con una red de de interconexión ocho computadoras dual de procesadores de dos cada una, lo cual nos da una capacidad total de 16 procesadores del modelo Pentium III a una velocidad de 800 MHz; si multiplicamos esta capacidad de cómputo, se podría lograr una capacidad de procesamiento de hasta 12800 MHz, esto es, 12.8 GigaHertz; sin embargo, el rendimiento de las supercomputadoras se basa en el número de instrucciones en punto flotante que se pueden realizar en un segundo reduciendo las cifras en unidades de medida conocidas como Flops (Megaflops, Gigaflops, Teraflops).

SALIDAS

Las salidas nos muestran el valor de H que se espera obtener, que es el exponente de Hurst encontrado en la serie de datos procesada; además, un conjunto de datos conocido como rango reescalado (R/S) que permitirá realizar una gráfica de dispersión mostrando el comportamiento de la serie aplicado al rango reescalado; dicho rango se almacena en un archivo de texto de salida, el cual se puede exportar fácilmente a aplicaciones gráficas fáciles de usar y conocidas, como lo pueden ser

Excel, Oen Ofice, Mtlab, etc. Basado en el valor del exponente de Hurst H se determina la dimensión fractal, que es igual a 2-H, el cual también es generado por el algoritmo.

PROCESO

El proceso es discutido con detalle, describiendo los pasos y procedimientos a seguir, derivado de las convenciones que existen en el cálculo del rango escalado.

R/S tiene media cero y se expresa en términos de la desviación estándar; esto es, que los puntos del rango reescalado que se encuentre se encontrarán muy cerca de esa media. En general, los valores de R/S se incrementan con n, por el valor de la ley de potencias igual al exponente Hurst, esta es la primera conexión del fenómeno Hurst y la geometría Fractal. Hay que mencionar además que el método R/S es un análisis no paramétrico que no requiere de una distribución específica. Para que una serie pueda ser considerada como fractal, el requisito clave que debe de cumplir es una escala de ley de potencia. El exponente de Hurst se determina por medio de una regresión lineal de los puntos de ln(R/S) contra ln(n), como se muestra en la siguiente ecuación.

$$\ln(R/S)n = \log(c) + H\log(n) \qquad \ldots\ldots 3.1$$

Donde R/S es el rango reescalado encontrado, c es un valor constante y H es el exponente de Hurst encontrado en una serie para n elementos.

Si el sistema tuviera la característica de independencia entonces H = 0.50. Sin embargo, como resultado de su investigación en el río Nilo, Hurst encontró un exponente de H = 0.91. Si se comparara el ejemplo de una partícula errática descrita por Einstein como un camino aleatorio, con un H = 0.91 entonces esta partícula cubriría una distancia mayor que otra partícula con un proceso aleatorio en el mismo periodo. Mandelbrot demostró empíricamente que en series de tiempo cuyas observaciones son independientes, el estadístico (R/S) es asintóticamente proporcional a la raíz cuadrada, es decir si H = 0.5 resulta un evento aleatorio puro. Hurst supuso como hipótesis nula que el comportamiento de fenómeno sea de una caminata aleatoria o un movimiento browniano, si este fuera el caso la ecuación del rango reescalado estaría dado por:

……3.2

METODOLOGÍA PARA ENCONTRAR EL EXPONENTE DE HURST Y EL RANGO REESCALDADO (R/S):

A continuación se describen los pasos que propuso Edgar Petters [27], basado en los trabajos anteriores de Hurst, Maldelbrot y Wallis; así como su posible solución ya sea en un diagrama o un segmento de programa o pseudocódigo:

Paso 1. Se inicia una serie de datos M de tiempo de la cual, si las diferencias entre un valor y otro son muy distantes, es conveniente reducir la serie original a una serie de tiempo llamada N=M-1 donde la serie es "suavizada" para tener diferencias pequeñas de tal forma que:

$$N_i = \log\left(\frac{M_{i+1}}{M_i}\right) \qquad \text{……..3.3}$$

donde i={1,2,3,4,…,N}

Paso 2. Se divide el periodo de tiempo N en A subperiodos divisores de N contiguos con una longitud n cada uno, de tal forma que (A*n=N). Se puede nombrar a cada uno de estos subperiodos como I_a, donde a=1,2,3,…,A. Cada elemento I_a puede ser nombrado o etiquetado como $N_{k,a}$, donde K=1,2,3,…,n. Para cada subperiodo I_a de longitud n, se define el valor promedio e_a de ese subperiodo de la siguiente forma:

$$e_a = \frac{1}{n}\sum_{k=1}^{n} N_{k,a} \qquad \text{……3.4}$$

donde:

e_a: es el valor promedio para cada subrango a

N es el número de elementos de ese subrango

N_{ka} es cada elemento de es subrango

Paso 3. Las diferencias entre la media e_a y cada elemento $N_{k,a}$ para cada subperiodo I_a se van sumando para obtener una serie de tiempo acumulada $X_{k,a}$ definida como:

$$X_{k,a} = \sum_{i=1}^{k} (N_{i,a} - e_a) \qquad \text{.....3.5}$$

donde:

K=(1,2,3,…,n)

Paso 4. Se define el rango RI_a como la diferencia entre el valor máximo y el valor mínimo del valor $X_{k,a}$ encontrados en cada subserie I_a:

$$R_{I_a} = \text{Max}(X_{k,a}) - \text{Min}(X_{k,a}) \qquad \text{.....3.6}$$

Paso 5. Se calcula la desviación estándar SI_a muestra por cada periodo I_a, de la siguiente forma:

$$S_{I_a} = \left(\frac{1}{n} \sum_{k=1}^{n} (N_{k,a} - e_a)^2 \right)^{\frac{1}{2}} \qquad \text{.....3.7}$$

que es lo mismo que:

$$S_{I_a} = \sqrt{\frac{1}{n} \sum_{k=1}^{n} (N_{k,a} - e_a)^2} \qquad \text{.....3.8}$$

Paso 6. Se obtiene un rango RI_a y para cada perido I_a, este rango se normaliza dividiéndolo por su desviación estándar muestral SI_a correspondiente. Así, el rango reescalado para cada subperiodo I_a es: RI_a/SI_a. Como existen A periodos contiguos de longitud n, se toma el valor promedio R/S para los periodos de longitud, definido como:

.....3.9

Paso 7. La longitud n o el tamaño del subperiodo se incremente al siguiente valor (divisor) posible, de tal forma que N/n sea un valor entero; se continúa con el valor mas pequeño con respecto al anterior y se repiten los pasos del 2 al 6 hasta que n=(N/2)-1.

Paso 8. Al rango reescalado $(RI_a/SI_{a)}$ para a=1,2,3,…,A se aplica una regresión por mínimos cuadrados de log(RI/SI)n contra log(n). La ordenada al origen es el log(c) y la pendiente de la ecuación es la estimación del exponente de Hurst.

Para determinar la regresión por mínimos cuadrados tenemos lo siguiente:
Se definen los valores log(Ia) como valores x y los valores de $\log(RI_a/SI_{a)}$ como los valores de y; siguiendo el procedimiento general de la regresión por mínimos cuadrados se realizará lo siguiente:

Paso mc1. Se calcula la media aritmética μ_x y la desviación típica σ_x de los datos de la primera variable. En este caso, se debe calcular la media aritmética y la desviación típica de las coordenadas "X" (o el rango log(Ia)).

$$\mu_x = \frac{X_1+X_2+\ldots+X_n}{n} \qquad \sigma_x = \frac{\sqrt{(x_1-\mu_x)^2+(x_2-\mu_x)^2+\ldots+(x_n-\mu_x)^2}}{n} \qquad \ldots\ldots 3.10,\ 3.11$$

Paso mc2. Se calcula la media aritmética μ_y y la desviación típica σ_y de los datos de Y, la segunda variable; "Y" son los puntos de log(RIa/SIa).

$$\mu_y = \frac{Y_1+Y_2+\ldots+Y_n}{n} \qquad \sigma_y = \frac{\sqrt{(y_1-\mu_y)^2+(y_2-\mu_y)^2+\ldots+(y_n-\mu_y)^2}}{n} \qquad \ldots\ldots 3.12,\ 3.13$$

Paso mc3. Se calcula la llamada covarianza σ_{xy} entre las dos variables(x e y) de la siguiente manera:

$$\sigma_{xy} = \frac{(X_1-\mu_x)(Y_1-\mu_y)}{n} + \frac{(X_2-\mu_x)(Y_2-\mu_y)}{n} + \ldots + \frac{(X_n-\mu_x)(Y_n-\mu_y)}{n} \qquad \ldots\ldots 3.14$$

Paso mc4. Hecho todo lo anterior, se obtiene la ecuación de la recta de regresión por mínimos cuadrados de la forma siguiente:

$$y - \mu_y = m(x - \mu_x) \qquad \ldots\ldots 3.15$$

Donde la pendiente (que determina el exponente H) m es igual a:

$$\frac{\sigma_{xy}}{\sigma_x^2} \qquad \ldots\ldots 3.16$$

Es decir que la recta de regresión por mínimos cuadrados es la recta que pasa por el punto (μ_x, μ_y) y que tiene por pendiente a m (Fórmula 3.16).

4.1 CONSTRUCCIÓN

Basado en los métodos descritos anteriormente, podríamos ver gráficamente la solución del algoritmo de la siguiente manera:

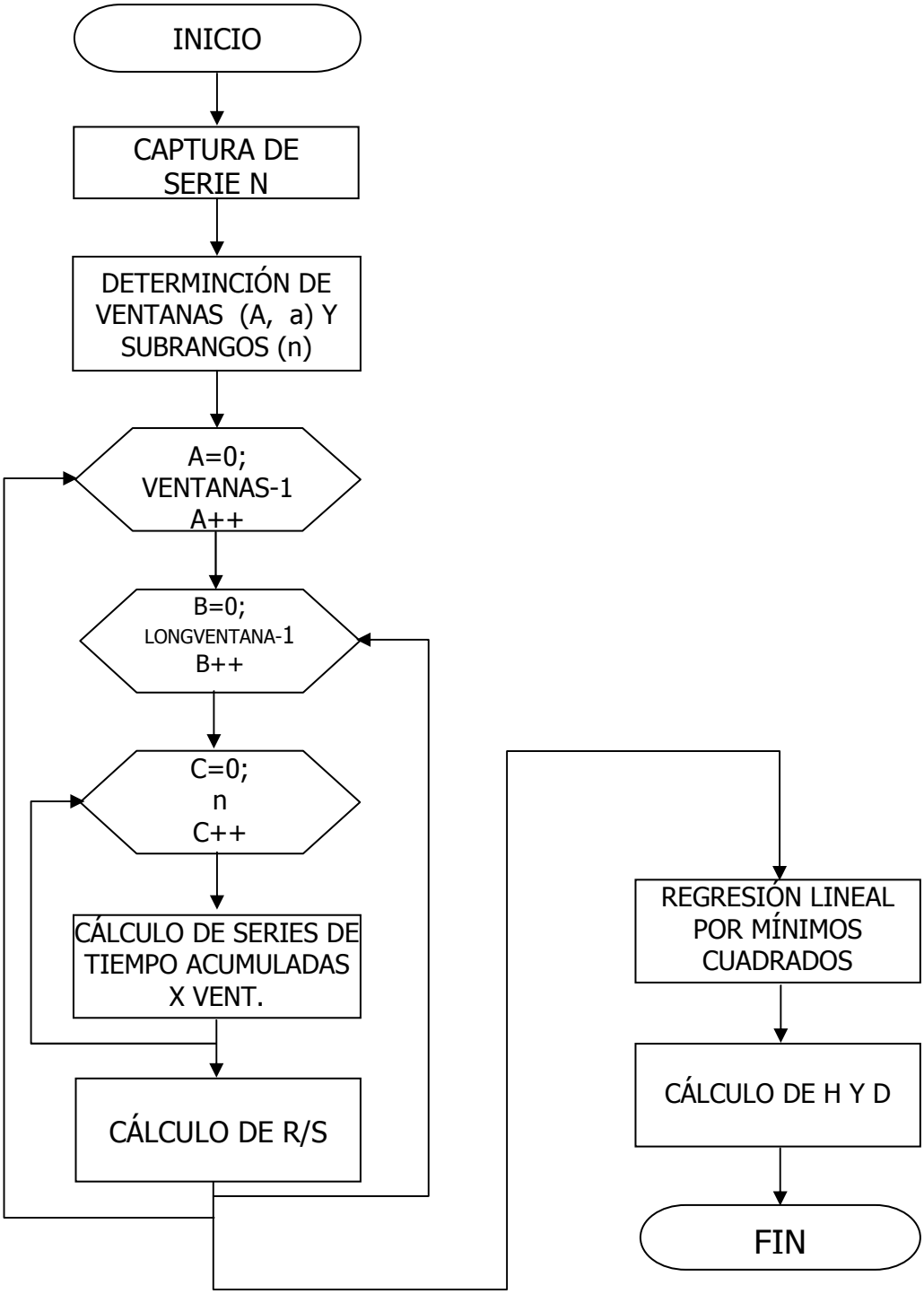


Figura 4.2 Diagrama de flujo, cálculo de H

El anterior diagrama considera una solución secuencial; sin embargo se puede tomar en cuenta una paralelización en los procesos de cálculo de R/S, ya que como se observó en las fórmulas de la 3.4 a la 3.8 este proceso se va a realizar con diferentes valores determinados de los rangos de las ventanas o subperiodos en los que se divide la serie; por lo tanto para los procesos de iteración segundo y tercero puede considerarse como un procedimiento general que se puede ejecutar en paralelo por varios procesos:

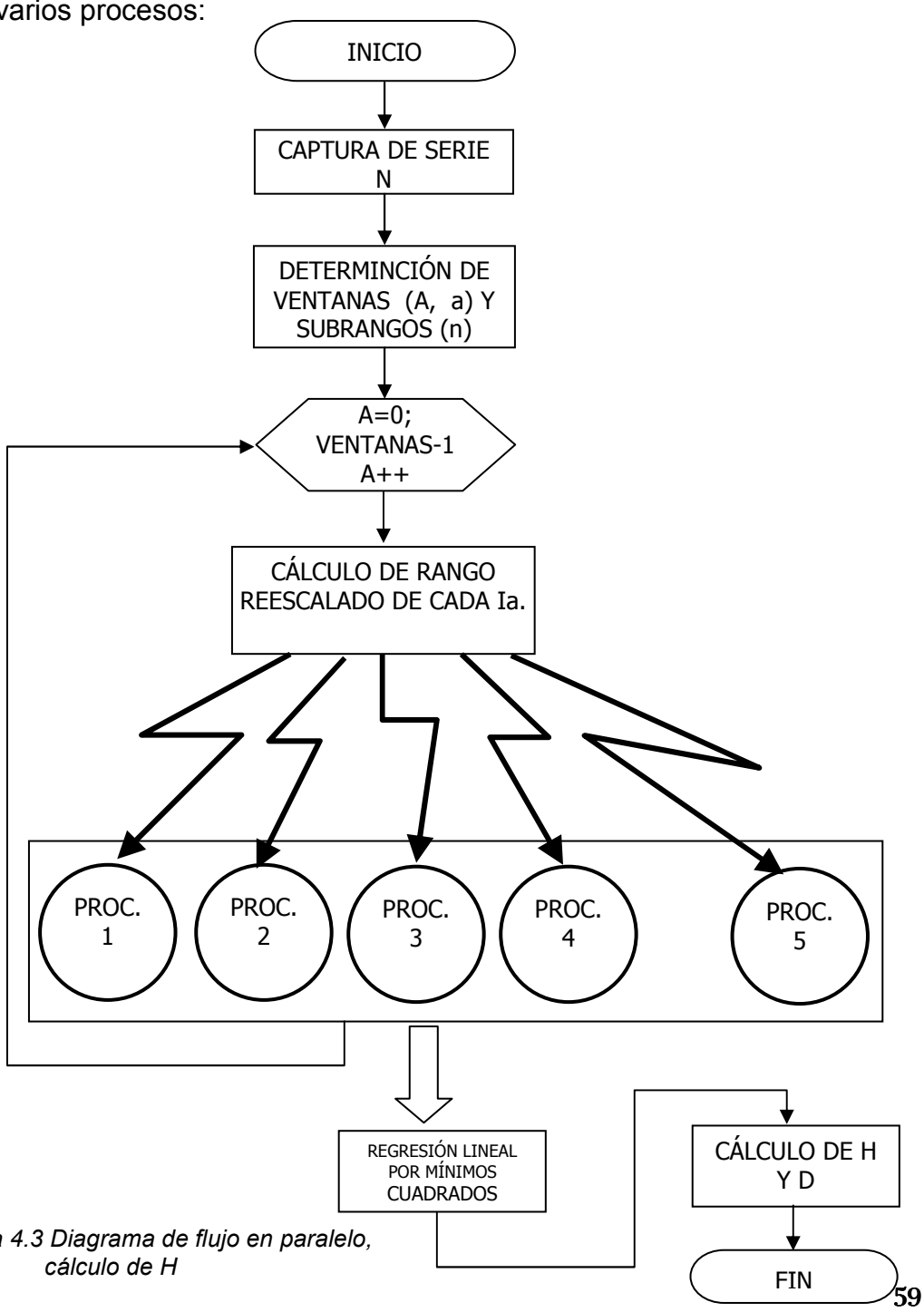


Figura 4.3 Diagrama de flujo en paralelo, cálculo de H

4.2 PROGRAMACIÓN

A continuación se resumen las partes más importantes de cada procedimiento, según los diagramas de flujo anteriores, resaltando operaciones trascendentes:

CAPTURA DE SERIE N

La serie que se analiza, se debe de encontrar en un archivo de tipo texto en formato ANSI con una separación de salto de línea por cada dato llamado “serie.txt”, el cual será capturado en un arreglo llamado serieN; a partir de esa captura se determina el número de datos introducidos:

```
long int captura()
{
 FILE *flujo;
 clrscr();
 if((flujo=fopen("serie.txt","r"))==NULL){
    puts("El archivo no se puede abrir\n");
    getch();
    exit(1);
    }
 else
 {
  printf("Archivo serie.txt abierto\n");
  printf("COMIENZA CAPTURA\n");
  for(x=0;!feof(flujo);x++)
  {
    fscanf(flujo,"%f",&Mm1);
    SerieN[x]=Mm1;
      //}
  }//del for
 }//del else

 fclose(flujo);
 printf("\n x vale cuando salio del for: %ld",x);
 return x-1;
 } //de funcion
```

Programación en ANSI C compilado con gcc y mpicc.
Ejecutado en el cluster HPC del CIDETEC

DETERMINCIÓN DE VENTANAS (A, a) Y SUBRANGOS (n)

Este proceso se divide en varias funciones, ya que el primer punto hay que considerar es que, derivado del número de datos capturados, cuál es el mejor número de datos a tomar en cuenta, ya que pueden existir valores que permitan un número mayor de subragos a dividir la serie. Como se menciona en el análisis de Husrt es conveniente tener un número significativamente grade de muestras. Derivado de un análisis hecho a los posibles rangos divisores que puede dar una

cantidad, se tienen los siguientes valores en los que un rango de datos original podría variar; esto es poder desechar algunos datos, para encontrar el número óptimo (esto es, que nos de un número mayor de divisores):

Rango de datos	Datos que pueden ser desechados
1,000-10,000	720
10,001-20,000	900
20,001-40,000	1,460
40,001-80,000	1,200
80,001-100,000	1,600
100,001-1,000,000	1,800
1,000,001-10,000,000	2,000
10,000,001-VALORES MAS GRANDES	2,200

Tabla 4.1 Valores a los que se puede reducir una Serie N de valores de entrada

```
/*
  numero_intervalos calcula el numero de intervalos en los que se puede
  dividir la serie N, tal que A*n=N
*/

long int numero_intervalos(long int N)
{
  long int resid,i=0,j,cv=0,vi=0,r;
     for(i=10;i<((N/2));i++)
    {
    resid=N%i;
    if(resid==0)
       {
       if(bandera1=='1')  //Se activa cuando se encotró el rango óptimo
         {
          numero_ventana[0][vi]=cv+1;
          numero_ventana[1][vi]=i;
          numero_ventana[2][vi]=(R_ventanas/i);
          vi++;
         }
       cv++;
       }
    }
    //printf("\nvalor devuelto, cv %ld\n",cv);
    return cv;
  }//fin de la funcion
```

Programación en ANSI C compilado con gcc y mpicc.
Ejecutado en el cluster HPC del CIDETEC

```
/*
  encuentra_ventanas
  Recibe el numero N de valores de la serie orignial y el numero de
  intervalos en los que se puede dividir ese numero.
  Determina cual es el rango óptimo de datos a utilizar; que es
  aquel en el que se puede dividir la serie N en un numero mayor de intervalos
*/
void encuentra_ventanas(long int SerieO,long int rangos)
   {
     long int va,ventanas;
     printf("\nSerieO vale: %ld y tienen %ld elementos divisores",SerieO,rangos);
     if(SerieO<1001)
       {va=25;   iteracion1(va);}
       if(SerieO>1000&&SerieO<10001)
   {va=720; iteracion1(va);}
       if(SerieO>10000&&SerieO<20001)
   {va=900; iteracion1(va);}
       if(SerieO>20000&&SerieO<40001)
   {va=1460; iteracion1(va);}
       if(SerieO>40000&&SerieO<80001)
   {va=1200; iteracion1(va);}
       if(SerieO>80000&&SerieO<100001)
   {va=1600; iteracion1(va);}
       if(SerieO>100000&&SerieO<1000001)
   {va=2000; iteracion1(va);}
       if(SerieO>1000000)
   {va=2200; iteracion1(va);}
   }

/*iteracion, busca aquellos valores que sean divisores del valor que entra
  permite conocer rango de divisores mas óptimos para el análisis
  se excluyen los valores impares, ya que siempre se encontrarán menos
  divisores que en los pares*/
long int iteracion1(long int valor)
{
  long int a,intervalos;
  for(a=2;a<=valor;a++)
  {
    intervalos=numero_intervalos(SN-a);
    if (intervalos>NVo)
      {
   NVo=intervalos;
   R_ventanas=SN-a;
   VENTAN=NVo;
   printf("\nElemento con mayor numero de ventanas: %ld\n",R_ventanas);
     }
   a++; //solo se corren valores pares, ya que sus rangos siempre son mayores a los nones
   }
   return 0;
}
```

Programación en ANSI C compilado con gcc y mpicc.
Ejecutado en el cluster HPC del CIDETEC

PROCESO POR RANGO PARA EL CÁLCULO DEL RANGO REESCALDO

Una vez definido el número de elementos, así como el número de subrangos o ventanas a utilizar, con sus intervalos respectivos, dichos datos se han almacenado en un arreglo bidimensional llamado “numero_ventana[8][230]” donde a cada columna le pertenecerán tanto datos de entrada que serán utilizados para los cálculos, así como datos de salida con el resultado del rango reescalado:

[0][0]	[1][0]	[2][0]	[3][0]	[4][0]	[5][0]	[6][0]	[7][0]
ventana	Valor vent.	Interv.	R/S	Log(n)	Log(R/S)	DesvX	DesvY

Figura 4.4 Representación del arreglo bidimensional numero_ventana(8 columnas, por 230 renglones)

El límite de 230 está definido por el número máximo de divisores que se encontraron en los números que van de 1 hasta 10,000,000; lo que significa que este puede ser el límite máximo de ventanas o subrangos que se pueden procesar.

CÁLCULO DEL RANGO ESCALADO

El cálculo se ha programado en una función llamada *calculaRS()* que implementa el método R/S descrito al principio de este capítulo.

```
/*****************************calculoRS() ************************************
Realiza las iteraciones correspondientes para obtener el Rango de cada venana
*****************************************************************************/

void calculoRS(void)
 {
  long int b,c,aux1=0,aux2=0,aux3=0,val,interv;
  float acumula=0.0,captura,acDesv1,val_min,val_max,valor_media,Ria,Sia,desv1,Rian;
  val=m1;
  interv=m2;
  printf("\nEntrando a funcion MEDIA");
  printf("\n val vale: %ld, interv vale: %ld b: %ld, c: %ld y vvj: %ld",val,interv,b,c,vvj);
  printf("\nCOMIENZA CALCULO   vvj vale:%ld y val(b) vale:%ld\n",vvj,val);
  /*CADA NODO RECIBIRA 2 VALORES, numero_ventana[0][vvj] vvj+1 y numero_ventana[1][vvj] val*/
  numero_ventana[0][vvj]=vvj+1;
  numero_ventana[1][vvj]=val;
  for(b=0;b<val;b++)
          {

           for(c=0;c<interv;c++)
             {
              captura=SerieN[c+aux2];
              acumula=acumula+captura;
              mediasi[b]=acumula/interv;
             }

           aux2=aux2+c;
```

```
            acumula=0.0;

            val_min=0.0;
            val_max=0.0;
            acDesv1=0.0;
            for(c=0;c<interv;c++)
              {

               captura=SerieN[c+aux3];
               valor_media=captura-mediasi[aux1];
               desv1=valor_media*valor_media;
               if (val_min>valor_media)
                        val_min=valor_media;
               if(val_max<valor_media)
                  val_max=valor_media;
               acDesv1=acDesv1+desv1;
              }
            aux3=aux3+c;
            aux1++;

            /*Calculo del rango RIa*/
            if(val_min<0)
              Ria=val_max+(val_min*-1);
            else
              Ria=val_max-val_min;

/**********Calculo de la desviación estándar muestral Sia para cada periodo***/
            Sia=sqrt((1.0/interv)*acDesv1);
            Rian=Ria/Sia;
            R_S=Rian+R_S;
           }
            printf("\n R/S vale:%f",R_S);
            R_S=R_S/val;
/****** MPI_Send se encargará de enviar los resultados de cada valor del Rango Reescalado R/S a través de la
variable R_S de cada procesador hacia el procesador maestro que por omisión es el procesador cero*********/
        MPI_Send(&R_S,1,MPI_FLOAT,0,0,MPI_COMM_WORLD);
```

Programación en ANSI C compilado con gcc y mpicc.
Ejecutado en el cluster HPC del CIDETEC }

REGRESIÓN LINEAL POR MÍNIMOS CUADRADOS Y CÁLCULO DEL EXPONENTE DE HURST H Y LA DIMENSIÓN FRACTAL D

Una vez que se ha determinado el rango reescalado y almacenado en el arreglo "*numero_ventana[8][230]*", se aplica el método de regresión por mínimos cuadrados y se determina H y D aplicando el método descrito a partir de la fórmula 3.10, tomado los valores de entrada correspondientes al logaritmo de cada ventana y el logaritmo del rango reescalado del citado arreglo (columnas cuatro y cinco respectivamente).

```
float calculoH()
{
long int valores;
float mediax=0.0,mediay=0.0,desvTx=0.0,desvTy=0.0,desvTxy=0.0,pendiente;
for(vvj=0;vvj<230;vvj++)
  {
   if(numero_ventana[0][vvj]!=NULL)
    {
      mediax=mediax+numero_ventana[4][vvj];
      mediay=mediay+numero_ventana[5][vvj];
      }
   else{
    break;
       }
  }
  printf("\nvalor acum de x: %f, valor acum y: %f, vvj:%ld",mediax,mediay,vvj);
  valores=(vvj);
  printf("\nvalores vale: %ld",valores);
  mediax=(mediax/valores);
  mediay=(mediay/valores);
  printf("\n Media de x: %f,  media de y: %f",mediax,mediay);

  /*Calculo de la desviacion tipica*/
  for(vvj=0;vvj<VENTAN;vvj++)
  {
   if(numero_ventana[0][vvj]!=NULL)
    {
      numero_ventana[6][vvj]=pow(numero_ventana[4][vvj]-(mediax),2);
      desvTx=desvTx+numero_ventana[6][vvj];
      numero_ventana[7][vvj]=pow(numero_ventana[5][vvj]-(mediay),2);
      desvTy=desvTy+numero_ventana[7][vvj];
      desvTxy=desvTxy+(((numero_ventana[4][vvj]-(mediax))*(numero_ventana[5][vvj]-(mediay)))/valores);
      }
   else{
    break;
       }
  }
  desvTx=sqrt(desvTx/valores);
  desvTy=sqrt(desvTy/valores);
  printf("\nDesv.Tipica X=%f, Desv.Tipica Y=%f, Desv.Tipica XY=%f",desvTx,desvTy,desvTxy);
  pendiente=desvTxy/(pow(desvTx,2));
  printf("\n\t\tLa pendiente es (o el exponente H): %f",pendiente);
getch();
return pendiente;
}
```

Programación en ANSI C compilado con gcc y mpicc.
Ejecutado en el cluster HPC del CIDETEC

En la paralelización del método del cálculo de R/S, la función calculoRS() será procesada por cualquier proceso; lo importante en esta sección es definir el control que permita identificar qué ventana se estará procesado.

```
/**** MPI ****/
MPI_Init(&argc,&argv);        /*Inicializacion del MPI*/
MPI_Status status;            /*Inicializacion para pso de mensajes en MPI*/
MPI_Comm_rank(MPI_COMM_WORLD,&minodo); /*comunicador de MPI para identificar los nodos*/
MPI_Comm_size(MPI_COMM_WORLD, &size); /*Definición del número de procesos, que se deriva del
número de ventanas que se han de procesar   */


     m1=(long int)numero_ventana[1][minodo];  //Numero de ventana a procesar
     m2=(long int)numero_ventana[2][minodo];  //Longitud del intervalo de la ventana a procesar

     calculoRS();


  if(minodo==0)
  {
    for(vvj=0;vvj<size;vvj++)
     {
/****** MPI_Recv se encargará de recibir en el proceso cero los resultados del valor del Rango Reescalado R/S
de cada procesador a través de la variable R_S ; de donde conjuntará todos los resultados en el arreglo
correspondiente*********/

      MPI_Recv(&R_S,1,MPI_FLOAT,vvj,0,MPI_COMM_WORLD,&status);
      numero_ventana[3][vvj]=R_S;
     }
calculoH();
imprime_arreglo();
genera_resultado();  //El rango reescalado resultante se guarda en un archivo de texto llamado serieR.txt

}

MPI_Finalize(); /*FINALIZA INTERACCION MPI */
```

Programación en ANSI C compilado con gcc y mpicc.
Ejecutado en el cluster HPC del CIDETEC

CAPÍTULO 5

ANÁLISIS Y DISCUSIÓN DE RESULTADOS

5.1 VALIDACIÓN

Comparando los resultados obtenidos en el programa contra los generados por el software *Benoit*, existe una diferencia mínima con los datos obtenidos. Al utilizar diferentes equipos cómputo para realizar los cálculos, es compresible que estas diferencias se evidencien, esto depende del sistema operativo sobre el que se esté trabajando ya que el sistema operativo es el que implementa de diferente forma el uso y representación de los datos en punto flotante; el comportamiento generado por el rango reescalado obtenido en ambos casos es el mismo y existen evidencias de acercamiento en el resultado final. Además, entre más grande es el número de datos procesados, se reduce la diferencia del los valores de H encontrados. Así por ejemplo, como se muestran los resultados de cada intervalo y sus diferencias en la tabla 5.1, cuando se procesaron 500 datos, *Benoit* reportó un valor de H igual a 0.132 y el programa un valor de 0.21107 siendo la diferencia de 0.079. Y cuando se procesaron 4095 datos el valor resultante de H en *Benoit* fue de 0.244 y en el programa de 0.23457, existiendo una diferencia de 0.00943 (tabla 5.2).

RESULTADO BENOIT:		RESULTADO programa		DIFERENCIAS
INTERV.	R/S	INTERV.	R/S	
10	2.484057	10	2.3200917	0.16397
12	2.579714	12	2.5253766	0.05434
15	2.822422	15	2.659071	0.16335
16	2.828779	16	2.8090477	0.01973
20	3.236361	20	2.9767318	0.25963
24	3.110043	24	3.1143398	-0.00430
30	3.336073	30	3.2151425	0.12093
32	3.286501	32	3.4062219	-0.11972
40	3.575674	40	3.4707701	0.10490
48	3.290963	48	3.7082736	-0.41731
60	3.864964	60	3.7014649	0.16350
80	3.601951	80	3.8889539	-0.28700
96	3.698305	96	3.929076	-0.23077
120	3.780979	120	4.1698208	-0.38884
160	3.33501	160	4.0956378	-0.76063
H=	0.132	H=	0.21107	0.079

Tabla 5.1 Diferencias de R/S encontrado entre Benoit y programa para una serie de 500 datos

RESULDADO BENOIT:		RESULDADO programa		DIFERENCIAS
INTERV.	R/S	INTERV.	R/S	
10	2.438254	10	2.3275995	0.11065
11	2.578994	11	2.5367172	0.04228
12	2.62976	12	2.6999753	-0.07022
15	2.831434	15	2.8116868	0.01975
18	3.009493	18	2.9942236	0.01527
20	3.138614	20	3.0724947	0.06612
22	3.199921	22	3.1352453	0.06468
24	3.324307	24	3.2054203	0.11889
30	3.502161	30	3.2482226	0.25394
33	3.621221	33	3.386553	0.23467
36	3.735672	36	3.5317416	0.20393
40	3.907409	40	3.6177592	0.28965
44	3.853496	44	3.7091291	0.14437
45	4.08432	45	3.7459705	0.33835
55	4.158335	55	3.9096951	0.24864
60	4.326796	60	3.9870694	0.33973
66	4.397726	66	4.071178	0.32655
72	4.576408	72	4.1974816	0.37893
88	4.882106	88	4.3123469	0.56976
90	4.898981	90	4.3034739	0.59551
99	4.897581	99	4.5019932	0.39559
110	5.023559	110	4.7024245	0.32113
120	5.159503	120	4.6759777	0.48353
132	5.22922	132	4.8145609	0.41466
165	5.483374	165	5.2041407	0.27923
180	5.850259	180	5.1413698	0.70889
198	5.717519	198	5.2480807	0.46944
220	6.01798	220	5.3575954	0.66038
264	6.008087	264	5.5685163	0.43957
330	6.30447	330	5.5657177	0.73875
360	7.091475	360	5.870297	1.22118
396	6.326413	396	6.3463435	-0.01993
440	6.56317	440	6.2247381	0.33843
495	6.724191	495	6.6355534	0.08864
660	6.947075	660	6.6469955	0.30008
792	7.224821	792	6.9314761	0.29334
990	8.025652	990	7.6971083	0.32854
1320	7.913161	1320	7.4990129	0.41415
H=	0.244	H=	0.23457	0.00943

Tabla 5.2 Diferencias de R/S encontrado entre Benoit y programa para una serie de 4094 datos

5.2 ANÁLISIS Y DISCUSIÓN DE RESULTADOS

Al realizar una comparación entre el algoritmo secuencial contra el paralelizado, se pueden encontrar grandes ventajas en el segundo.

- Si se toma en cuenta el tiempo que toma un programa en ejecutar podemos darnos cuenta la gran diferencia entre uno y otro.
- Se realizaron comparaciones entre el procesamiento paralelo y secuencial, describiendo los valores que representan las operaciones o cálculos que se deben realizar cuando se procesa una serie de 500 datos; la siguiente figura muestra el resultado obtenido de esas comparaciones determinando que:

  Sea un procesamiento secuencial, para una serie de datos de 500, reducida a 480, el cálculo del R/S implica un total de 44,741 cálculos; la figura 5.3 indica, según el número de procesadores que fueron utilizados en paralelo el número al que se redujo por procesador el número de instrucciones que ejecutó, y su equivalencia porcentual respecto al número total de cálculos para R/S. La gráfica 5.2 muestra el análisis gráfico de la tabla citada.

	N=480		VENTANAS:	15			
	Cálculos a realzar: (por ventana o intervalo)						X VENTANA
	medias	dato-media	max y min	,min, max-	desv. Est.	(R/S)	**TOTALES**
1	10	480	960	1440	10	1	2901
2	12	480	960	1440	12	1	2905
3	15	480	960	1440	15	1	2911
4	16	480	960	1440	16	1	2913
5	20	480	960	1440	20	1	2921
6	24	480	960	1440	24	1	2929
7	30	480	960	1440	30	1	2941
8	32	480	960	1440	32	1	2945
9	40	480	960	1440	40	1	2961
10	48	480	960	1440	48	1	2977
11	60	480	960	1440	60	1	3001
12	80	480	960	1440	80	1	3041
13	96	480	960	1440	96	1	3073
14	120	480	960	1440	120	1	3121
15	160	480	960	1440	160	1	3201
	763	**7200**	**14400**	**21600**	**763**	**15**	**44741**

Figura 5.1 cálculos realizados por subrango o ventana para 500 datos

PROCESAMIENTO EN PARALELO			
Promedio de datos procesados	procesadores	Instrucciones por procesador	Porcentaje de diferencia
2,983	2	22,371	50%
	4	11,185	25%
	6	7,457	17%
	8	5,593	13%
	10	4,474	10%
	12	3,728	8%
	14	3,196	7%
	16	2,796	6%

Tabla 5.3 Distribución de las instrucciones por procesador

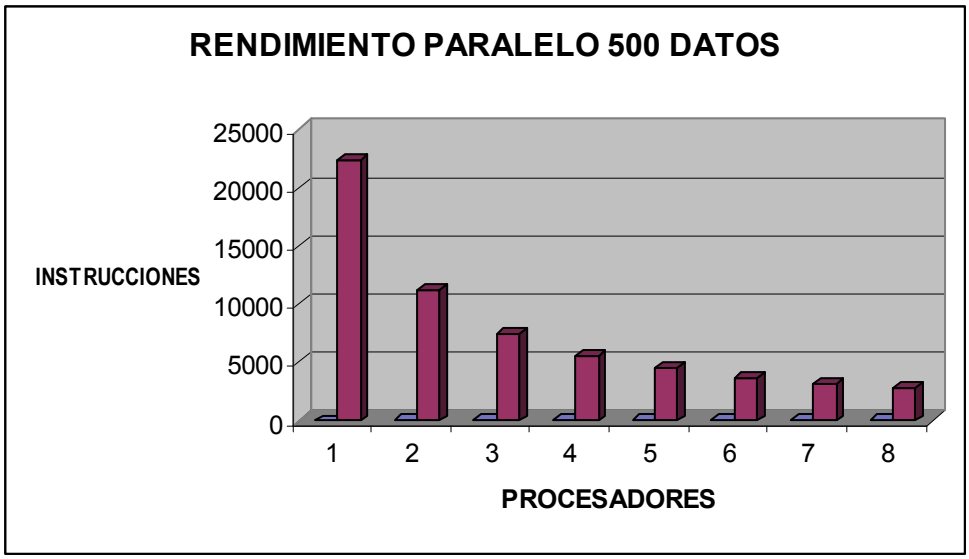


Figura 5.2 Gráfica de rendimiento según el número de procesadores para una serie de datos de 500.

Sea ahora, una serie de 3960 datos; el cálculo del R/S implica un total de 919,042 cálculos; la tabla 5.4 indica, según el número de procesadores que fueron utilizados en paralelo. Basado en el promedio de instrucciones por procesador, se redujo el número de instrucciones que cada procesador ejecutó a 24,185, y su equivalencia porcentual respecto al número total de cálculos para R/S. La figura 5.3 muestra el análisis de la tabla mencionada:

	N=3960		VENTANAS:	38			
	Cálculos a realzar: (por ventana o intervalo)						X VENTANA
	medias	dato-media*	max y min*ve	max,min, max	desv. Est.	(R/S)	**TOTALES**
1	10	3960	7920	11880	10	1	23781
2	11	3960	7920	11880	11	1	23783
3	12	3960	7920	11880	12	1	23785
4	15	3960	7920	11880	15	1	23791
5	18	3960	7920	11880	18	1	23797
6	20	3960	7920	11880	20	1	23801
7	22	3960	7920	11880	22	1	23805
8	24	3960	7920	11880	24	1	23809
9	30	3960	7920	11880	30	1	23821
10	33	3960	7920	11880	33	1	23827
11	36	3960	7920	11880	36	1	23833
12	40	3960	7920	11880	40	1	23841
13	44	3960	7920	11880	44	1	23849
14	45	3960	7920	11880	45	1	23851
15	55	3960	7920	11880	55	1	23871
16	60	3960	7920	11880	60	1	23881
17	66	3960	7920	11880	66	1	23893
18	72	3960	7920	11880	72	1	23905
19	88	3960	7920	11880	88	1	23937
20	90	3960	7920	11880	90	1	23941
21	99	3960	7920	11880	99	1	23959
22	110	3960	7920	11880	110	1	23981
23	120	3960	7920	11880	120	1	24001
24	132	3960	7920	11880	132	1	24025
25	165	3960	7920	11880	165	1	24091
26	180	3960	7920	11880	180	1	24121
27	198	3960	7920	11880	198	1	24157
28	220	3960	7920	11880	220	1	24201
29	264	3960	7920	11880	264	1	24289
30	330	3960	7920	11880	330	1	24421
31	360	3960	7920	11880	360	1	24481
32	396	3960	7920	11880	396	1	24553
33	440	3960	7920	11880	440	1	24641
34	495	3960	7920	11880	495	1	24751
35	660	3960	7920	11880	660	1	25081
36	792	3960	7920	11880	792	1	25345
37	990	3960	7920	11880	990	1	25741
38	1320	3960	7920	11880	1320	1	26401
	8062	**150480**	**300960**	**451440**	**8062**	**38**	**919042**

Figura 5.3 cálculos realizados por subrango o ventana para 3960 datos

PROCESAMIENTO EN PARALELO			
Promedio de datos procesados	procesadores	Instrucciones por procesador	Porcentaje de diferencia
24,185	2	459,521	50%
	4	229,761	25%
	6	153,174	17%
	8	114,880	13%
	10	91,904	10%
	12	76,587	8%
	14	65,646	7%
	16	57,440	6%

Tabla 5.4 Distribución de las instrucciones por procesador

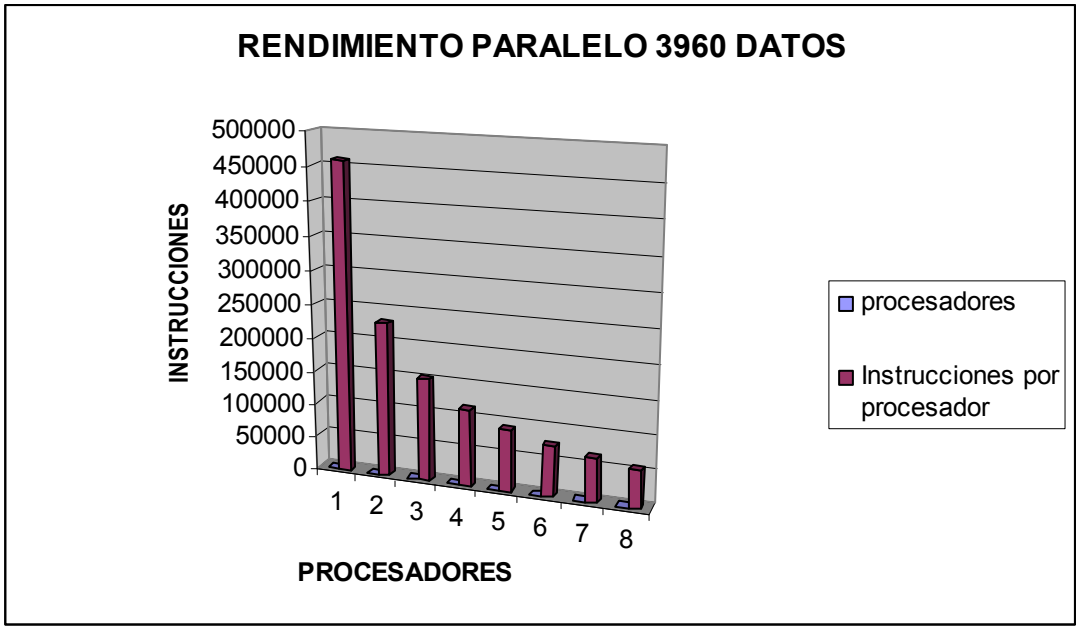


Figura 5.4 Gráfica de rendimiento según el número de procesadores para una serie de datos de 3960.

- Realizando cálculos con series de diferentes longitudes, se generaron los siguientes resultados indicando la diferencia entre *Benoit* y el programa en paralelo:

CALCULO CON BENOIT				CALCULO CON PROGRAMA				DIFERENCIA DE H
DATOS	VENTANAS	H		DATOS	VENTANAS	H		
500	15	0.132		500	15	0.21107		0.07907
4,095	38	0.244		4,095	38	0.23457		-0.00943
12,000	54	0.273		12,000	54	0.24648		-0.02652
20,000	110	0.274		20,000	110	0.27998		0.00598
30,000	70	0.284		30,000	70	0.2631		-0.0209
40,000	80	0.288		40,000	80	0.27322		-0.01478
50,000	86	0.282		50,000	86	0.27636		-0.00564
60,000	86	0.284		60,000	86	0.27862		-0.00538
72,000	97	0.278		72,000	90	0.28032		0.00232
80,000	98	0.281		80,000	98	0.28062		-0.00038
90,000	109	0.277		90,000	87	0.28016		0.00316
100,000	117	0.276		100,000	90	0.28086		0.00486
120,000	110	0.275		120,000	110	0.27998		0.00498

Tabla 5.5 Comparación de resultados de H entre Benoit y programa

CONCLUSIONES

1. Basado en el objetivo general, se pudo generar una herramienta no secuencial que permite el cálculo del exponente de Hurst. Derivado del marco teórico del presente trabajo, se motivó el desarrollo del producto final a través de la programación en paralelo el cual, al evaluar el producto final se destaca su utilidad en las aplicaciones que requieren su implementación.

2. Se logró demostrar que el método del rango reescalado puede ser resuelto mediante el método computacional en paraleo reduciendo el tiempo de procesamiento derivando un mayor aprovechamiento en el uso de los recursos computacionales disponibles sin necesidad de adquirir o invertir equipos más sofisticados.

3. Se cuenta con una herramienta computacional al alcance de la comunidad académica para la implementación en aplicaciones de cálculo de dimensión fractal basadas en el método del Rango Reescalado para el cálculo del exponente de Hurst.

4. El sistema se basa en código abierto y el uso de herramientas *GNU*, lo que motiva el uso de dichas herramientas abatiendo costos de licencias y dependencia de los fabricantes de software.

TRABAJOS A FUTURO

1. Del resultado obtenido, se propone como trabajo a futuro, crear la aplicación en un ambiente gráfico y fácil de implementar para los usuarios, basado en los sistemas operativos gráficos de Linux existentes; así como la creación de un módulo que permita la captura manual pero ágil, o por alguna interface con los datos que se tengan que cargar al sistema; y también generar un gráfico del rango reescaldado resultante desde la misma aplicación.
2. Adicionar al programa, un cálculo que permita eliminar “ruidos” a la serie resultante y actualizar el valor de H.
3. Derivado de la experiencia adquirida en este trabajo se propone la implementación de los otros cuatro métodos de trazado auto-afín restantes con el fin de reforzar y comparar los resultados que se obtienen a partir del método R/S.
4. La aplicación computacional resultante de este trabajo, se basó en la captura de la serie de datos en una estructura de almacenamiento indexado como lo es un arreglo; se pretende implementar el uso de apuntadores para hacer más eficiente el uso de la memoria del equipo de cómputo; permitiendo obtener procesamientos de series sin restricción de longitud o tamaño.
5. Implementar el procesamiento paralelo también en las funciones que realizan el cálculo para determinar el número de subrangos en las que se divide la serie, pues debido a que esta función se realiza en forma secuencial, cuando crece el número de datos a procesar se manifiesta un rendimiento no favorable.

GLOSARIO

Algoritmo Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema. (Diccionario de la lengua española de la Real Academia Española en línea: http://buscon.rae.es/)

Anisotropía La anisotropía (opuesta de isotropía) es la propiedad general de la materia según la cual determinadas propiedades físicas, tales como: elasticidad, temperatura, conductividad, velocidad de propagación de la luz, etc. varían según la dirección en que son examinadas. Algo anisótropo podrá presentar diferentes características según la dirección.

ANSI American National Standards Institute - Instituto Nacional Estadounidense de Estándares). Organización encargada de estandarizar ciertas tecnologías en EEUU. Es miembro de la ISO, que es la organización internacional para la estandarización. ANSI es una organización privada sin fines de lucro, que permite la estandarización de productos, servicios, procesos, sistemas y personal en Estados Unidos. Además, ANSI se coordina con estándares internacionales para asegurar que los productos estadounidenses puedan ser usados a nivel mundial. (http://www.ansi.org/)

BMP Es el formato propio del programa Microsoft Paint, que viene con el sistema operativo Windows. Puede guardar imágenes de 24 bits (16,7 millones de colores), 8 bits (256 colores) y menos. Puede darse a estos archivos una compresión sin pérdida de calidad: la compresión RLE (Run Length Encoding). El uso más común de este formato es generar imágenes de poco peso para crear fondos para el escritorio de Windows.

Cluster HPC High Performace Cluster. Cluster de computadora de alto rendimiento conformado por elementos de cómputo interconectados que distribuyen la capacidad de cómputo en diferentes nodos permitiendo el procesamiento computacional de cálculos complejos.

Dimensión Topológica La topología es una rama de las matemáticas que se ocupa de aquellas propiedades de las figuras geométricas que no cambian cuando se estira o se dobla el objeto. Una forma de definir una dimensión topológica es la dimensión de cubierta que es cuando se cubre una línea con intervalos chicos siempre hay puntos que se encuentran en al menos dos

intervalos. (extraído de Proyecto Universitario de Enseñanza Asistida por computadora de la UNAM http://interactiva.matem.unam.mx/)

GNU GNU es un acrónimo recursivo que significa GNU (No es Unix). El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre: el sistema GNU. Entre las motivaciones principales del desarrollo de este concepto destaca "volver al espíritu de cooperación que prevaleció en los tiempos iniciales de la comunidad de usuarios de computadoras". (http://www.gnu.org/)

Hardware Es la parte física de una computadora y más ampliamente de cualquier dispositivo electrónico. El término proviene del inglés y es definido por la Real Academia Española como el equipo de una computadora, sin embargo, es usual que sea utilizado en una forma más amplia, generalmente para describir componentes físicos de una tecnología, así el hardware puede ser de un equipo militar importante, un equipo electrónico, un equipo informático o un robot. En informática también se aplica a los periféricos de una computadora tales como el disco duro, CD-ROM, disquetera (floppy), etc... En dicho conjunto se incluyen los dispositivos electrónicos y electromecánicos, circuitos, cables, gabinetes, periféricos de todo tipo y cualquier otro elemento físico involucrado.

Hidrógrafo Persona que se dedica profesionalmente a la hidrografía o Hidrómetro que registra el nivel y caudal de las corrientes de agua.

FLOPS Acrónimo de las palabras Floating point Operations Per Second (operaciones en punto flotante por segundo), se usa como una medida del rendimiento de una computadora, especialmente en cálculos científicos que requieren un gran uso de operaciones de punto flotante.

Hidráulica Rama de la ingeniería que estudia la manera de conducir y aprovechar las aguas:

Invariancia Cualidad de invariante. Se denomina invariancia galileana al hecho derivado del principio de relatividad según el cual las leyes fundamentales de la física son las mismas en todos los sistemas de referencia inerciales. El término invariancia galileana usualmente se refiere a este principio aplicado a la mecánica newtoniana, en la cual las longitudes y tiempos no son afectados por el cambio en la velocidad, lo cual es descrito matemáticamente por una transformación galileana.

Isotropía En física, la isotropía se refiere al hecho de que ciertas magnitudes vectoriales medibles dan resultados idénticos con independencia de la dirección escogida para la medida. Cuando una determinada magnitud no presenta isotropía decimos que presenta anisotropía. En matemáticas, la isotropía se refiere a una propiedad geométrica de invariancia en una variedad diferenciable.

De la raíz iso - = "equitativo o igual"; algo isótropo es algo de proporciones idénticas.

Paradigma Un paradigma es —desde fines de la década de 1960— un modelo o patrón en cualquier disciplina científica u otro contexto epistemológico. El concepto fue originalmente específico de la gramática; en 1900 el diccionario Merriam-Webster definía su uso solamente en tal contexto, o en retórica para referirse a una parábola o a una fábula.

El término paradigma se origina en la palabra griega παράδειγμα (paradeigma), que significa "modelo" o "ejemplo". A su vez tiene las mismas raíces que παραδεικνύναι, que significa "demostrar".

Sofware Palabra proveniente del inglés (literalmente: partes blandas o suaves), que en nuestro idioma no posee una traducción adecuada al contexto, por lo cual se la utiliza asíduamente sin traducir y fue adoptada por la Real Academia Española. Se refiere al equipamiento lógico o soporte lógico de una computadora digital, comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware).

Taxonomía. Ciencia que trata de los principios, métodos y fines de la clasificación. Se aplica en particular, dentro de la biología, para la ordenación jerarquizada y sistemática, con sus nombres, de los grupos de animales y de vegetales.

REFERENCIAS

[1] Mandelbrot B.: *Fractales y Finanzas, Una aproximación a los mercados: arriesgar, perder y ganar.* Tusquets Editores S.A. Barcelona 2006.

[2] Barnsley, M. Fractals Everywhere. Ed. Academia Press, Inc. 1988, Londres

[3] Mandelbrot B.: *La geometría fractal de la naturaleza.* Tusquets Editores S.A. Barcelona 1997.

[4] Software Benoit Version 1.2, TruSoft Int'l Inc. http://www.trusoft.netmegs.com/index.html

[5] Quezada Len *Ariel Fractales y opinión Pública: Una aplicación del exponente de Hurst al estudio de la dinámica de la identificación ideológica.* Tesis de Doctorado. Universidad de Barcelona, Facultad de Psicología, Departamento de Psicología social, Barcelona, Junio de 2006.

[6] Gálvez Medina Ernesto T. *Análisis fractal del mercado de valores de México (1978-2004)*. . Tesis de Doctorado, ESCA-IPN 2005

[7] A.-L. Barábasi & H.E. Stanley. Fractal *Concepts in Surface Growths*; Cambridge University Press, Boston Massachusetts, 1995.

[8] Mandelbrot B.B. *The Fractal Geometry of Nature;* W.H. Freeman, Nueva York, 1982.

[9] Vicsek. *Fractal growth phenomena*; World Scientific, 1989.

[10] Mandelbrot B.B. *Fractals and Scaling in Finance;* Springer, New York, 1997.

[11] Mandelbrot B.B. *Fractals in Physics; Holland,* Amsterdam, 1986, pp. 3.

[12] J.J. Ramasco. *Invariancia de escala en el crecimiento de superficies e interfases*; Tesis doctoral 2002, Departamento de Física, Universidad de Cantabria, España.

[13] D. Morales. *Dinámica fractal de interfases en medios porosos*. Tesis doctoral 2001. Instituto Politécnico Nacional, México.

[14] A. Bunde & S. Havlin. *Fractals and Disordered Systems*; Springer, Heildelberg, 1996.

[15] F. Family & T. Vicsek. J. *Phys. A 18, L75*, 1985.

[16] T. Vicsek. *Fractal Growth Phenomena*; World Scientific, Singapore, 1991.

[17] J.G. Amar, P.-M. & F. Familiy. *Phys. Rev. A 43*, 1991, pp. 4548.

[18] J. Kertész, V.K. Horváth, & F. Weber; *Fractals 1, 67*, 1993..

[19] A. M. Sahimi; *J. Phys. I, France 4*, 1994, pp. 7113.

[20] B.B. Mandelbrot & J. Wallis. *Robustness of the Rescaled Range R/S in the Measurement of Noncyclic Long Run Statistical Dependence*; Water Resources Research 5, 1969d..

[21] H.E. Hurst. *Long-term Storage of Reservoirs; Transactions of the American Society of Civil Engineers* 116, 1951.

[22] A I. Simonsen. *Measuring Anti-Corrrelations in the Nordic Electricity Spot Market by Wavelets*; arXiv:cond-mat/0108033, v1, 2001.

[23] Álvarez Cedillo Jesús Antonio. *Apuntes de la asignatura de posgrado Procesamiento en paralelo de la Maestría en Tecnología de cómputo* del CIDETEC-IPN 2006.

[24] Torres Jiménez José, Rodríguez Tello Eduardo A. *Conceptos de cómputo paralelo*. Ed. Trillas, México D.F. 2000.

[25] Karniadakis George Em and Robert M. Kirby II. *Parallel Scientific Computing in C++ and MPI*, Cambridge University Press, 2003

[26] http://www.mpi-forum.org/

[27] Peters Edgar E. Fractal Market Analysis: Applying Chaos Theory to Investment and Economics. John Wiley and Sons New York 1994.

A N E X O S

PROGRAMA FUENTE COMPLETO:

```
/*************************************************************************
rs.c
Programa que implementa el metodo R/S para el cálculo del exponente de Hurst.
Recibe como argumento de entrada "-np" el número de procesadores.
Autor: Adauto Israel Ortiz Romero
Ultima modificacion: 23 de junio de 2008
*/
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<stdlib.h>
#include<mpi.h>
/*
SN guarda el numero de datos de la serie original
NVo Almacena el numero de intervalos que se pueden dividir con SN
EV guarda el rango optimo despues de explorar hacia abajo un numero de datos
que tenga un mayor numero de intervalos por dividir.
VENTAN: Es el n£mero de ventanas o subrangos que se obtiene a partir del rango optimo
*/
long int va,SN, NVo,EV,VENTAN,R_ventanas,x=0,par,vvi,vvj,ventana,valor,intervalo,m1,m2,control=0,ia;
float D,Ni,Mi,Mm1,numero_ventana[8][230],mediasi[230],SerieN[120000],pendiente,R_S=0.0;
int minodo;
char **p;
char bandera1='0',opcion='0';

int size;
/*Declaracion de prototipos */
void capturaD(void);
int numero_intervalos(long int);
int encuentra_ventanas(long int,long int);
int iteracion1(void);
void imprime_arreglo(void);
void genera_resultado(void);
void calculoRS(void);
int calculoH(void);
/****************** PROGRAMA PRINCIPAL   **********************/
int main(int argc, char *argv[])
{
 system("clear");
 capturaD();
 printf("\nDatos capturados: %ld \n",SN);
 par=SN;
 par=par%2;
 if(par!=0)
    {
    SN--;
    printf("\nSerie non, ajustado a serie par eliminando un elmento");
    printf("\nNuevo valor para SN=%ld",SN);
    }
 else
    {
    printf("\nvalor par=%ld",SN);
    }
```

```
 NVo=numero_intervalos(SN);
 printf("\nnvo= %ld\n",NVo);
 encuentra_ventanas(SN,NVo);

 /*generacion de rangos o ventanas para cada serie, se almacenan en el
   arreglo numero_ventana[8][230], donde la primera columna corresponde a la ventana,
   la segunda columna al divisor o intervalos de ese rango, la tercera es la
   longitud de cada intervalo, la cuarta el rango reescalado RS y la quinta el log
   del divisor, la sexta el log(R/S), septima y ocatava son de paso para calcular
   regresion por minimos cuadrados  */
 bandera1='1';
 numero_intervalos(R_ventanas);

 /**** MPI ****/
MPI_Init(&argc,&argv);        /*Inicializacion del MPI*/
MPI_Status status;            /*Inicializacion para pso de mensajes en MPI*/
MPI_Comm_rank(MPI_COMM_WORLD,&minodo); /*comunicador de MPI para identificar los nodos*/
MPI_Comm_size(MPI_COMM_WORLD, &size);

     m1=(long int)numero_ventana[1][minodo];
     m2=(long int)numero_ventana[2][minodo];
     calculoRS();

  if(minodo==0)
  {
    for(vvj=0;vvj<size;vvj++)
     {
      MPI_Recv(&R_S,1,MPI_FLOAT,vvj,0,MPI_COMM_WORLD,&status);
      numero_ventana[3][vvj]=R_S;
      printf("\n recibi el valor ----------------------------------------------------------------->%f",R_S);
     }
calculoH();
imprime_arreglo();
genera_resultado();
}

MPI_Finalize(); /*FINALIZA INTERACCION MPI */
return 0;
}
/****************** FIN PROGRAMA PRINCIPAL ********************/

/********************FUNCION captura D ***************************
  capturaD()
  Obiene los datos de una serie de datos M y genera una nueva serie N basada en
  N=log(mi+1/mi), guardandola en el archivo serieN.txt"
*********************************************************************/
void capturaD(void)
{
 FILE *flujo;
 if((flujo=fopen("serie.txt","r"))==NULL){
    puts("El archivo no se puede abrir\n");

    exit(1);
    }
 else
 {
```

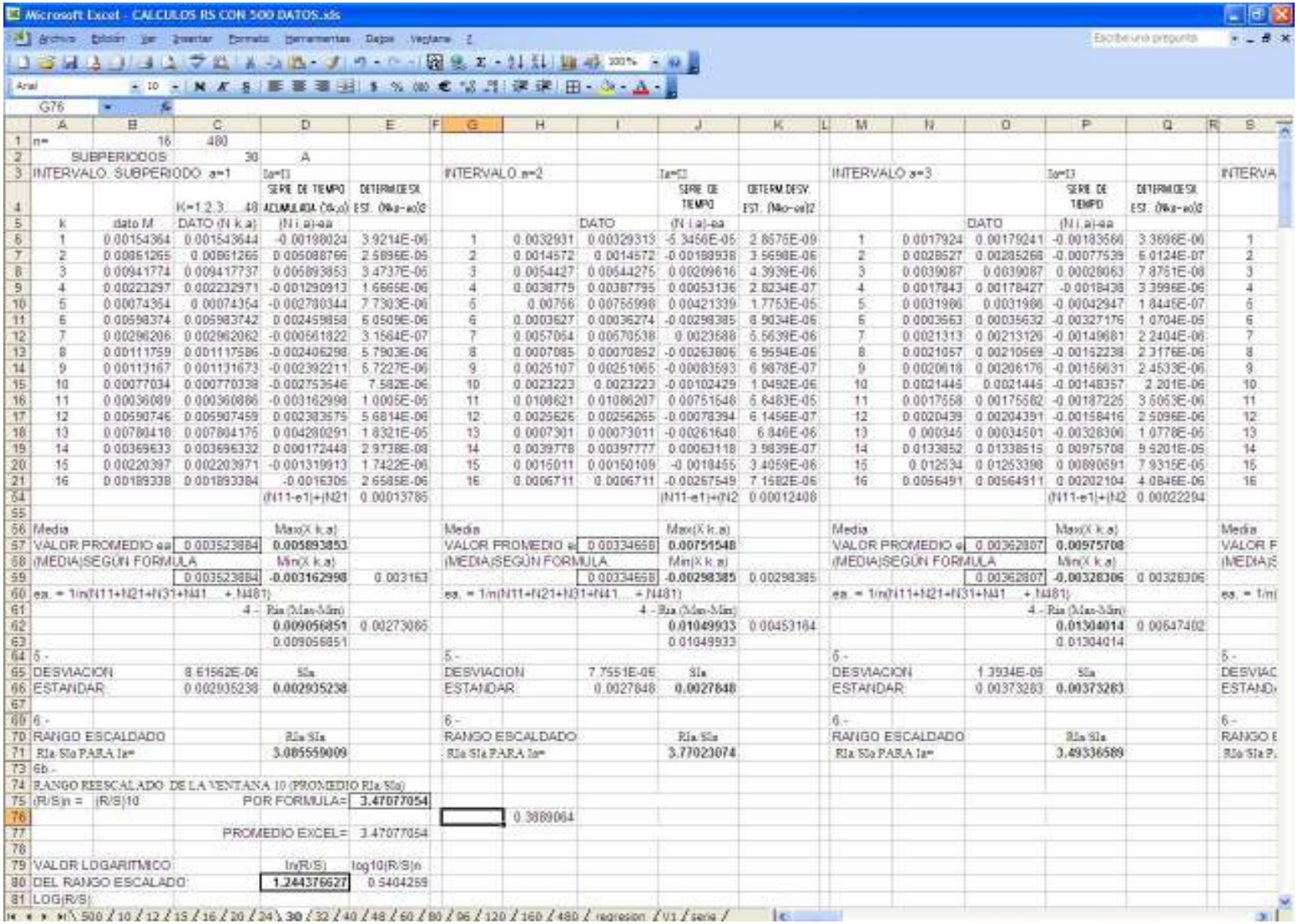
```
 printf("Archivo serie.txt abierto\n");
 printf("COMIENZA CAPTURA\n");
 for(x=0;!feof(flujo);x++)
 {
   fscanf(flujo,"%f",&Mm1);
   SerieN[x]=Mm1;
 }
 }
fclose(flujo);
SN=x-1;
 }

/*************************************************************************
 numero_intervalos calcula el numero de intervalos en los que se puede
 dividir la serie N, tal que A*n=N
*************************************************************************/
int numero_intervalos(long int N)
{
 long int i,resid,cv=0,vi=0;
   for(i=10;i<((N/2));i++)
          {
          resid=N%i;
          if(resid==0)
            {
            if(bandera1=='1')
              {
               numero_ventana[0][vi]=cv+1;
               numero_ventana[1][vi]=i;
               numero_ventana[2][vi]=(R_ventanas/i);
               vi++;
              }
            cv++;
            }
          }

   return cv;
  }

/*************************************************************************
 encuentra_ventanas
 Recibe el numero N de valores de la serie orignial y el numero de
 intervalos en los que se puede dividir ese numero.
 Determina cual es el rango optitmo de datos a utilizar; que es
 aquel en el que se puede dividir la serie N en un un numero mayor de intervalos
*************************************************************************/
int encuentra_ventanas(long int SerieO,long int rangos)
  {

   printf("\nSerieO vale: %ld y tienen %ld elementos divisores",SerieO,rangos);

   if(SerieO<1001)
     {va=25;   iteracion1();}
    if(SerieO>1000&&SerieO<10001)
          {va=720; iteracion1();}
    if(SerieO>10000&&SerieO<20001)
          {va=900; iteracion1();}
```

```
      if(SerieO>20000&&SerieO<40001)
            {va=1460; iteracion1();}
      if(SerieO>40000&&SerieO<80001)
            {va=1200; iteracion1();}
      if(SerieO>80000&&SerieO<100001)
            {va=1600; iteracion1();}
      if(SerieO>100000&&SerieO<1000001)
            {va=2000; iteracion1();}
      if(SerieO>1000000)
            {va=2200; iteracion1();}
  return 0;
  }

/*********************************************************************
iteracion1, busca aquellos valores que sean divisores del valor que entra
  permite conocer rango de divisores mas optimos para el analisis
  se excluyen los valores impares, ya que siempre se encontraran menos
  divisores que en los pares*********************************************/
int iteracion1(void)
{
  long int a,intervalos,valor;
  valor=va;
  for(a=2;a<=valor;a++)
  {
   intervalos=numero_intervalos(SN-a);
   if (intervalos>NVo)
     {
            NVo=intervalos;
            R_ventanas=SN-a;
            VENTAN=NVo;
     printf("\nElemento con mayor numero de ventanas: %ld\n",R_ventanas);
     }
  a++;
  }
  return 0;
}

/*********************************imprime_arreglo*************************/
void imprime_arreglo(void)
{
printf("\nVent. Valor  Interv     R/S       Log(n)      Log(R/S) ");
for(ia=0;ia<size;ia++)
  {
    printf("\n%4.0f    %5.0f                %5.0f                %2.7f                %1.7f                %1.7f
",numero_ventana[0][ia],numero_ventana[1][ia],numero_ventana[2][ia],
    numero_ventana[3][ia],numero_ventana[4][ia],numero_ventana[5][ia]);
  }
}

/*****************************************************************
genera_resultado abre un archivo llamado serieR.txt en el cual se almacena el
rango reescalado R/S ***************************************************/
void genera_resultado(void)
{
  FILE *fp;
  fp=fopen("serieR.txt","w");
```

```
  fprintf(fp,"Vent. Valor  Interv    R/S      Log(n)      Log(R/S)  \n");
for(vvj=0;vvj<size;vvj++)
  {
     fprintf(fp,"\n%4.0f   %5.0f    %5.0f  %2.7f   %1.7f
%1.7f",numero_ventana[0][vvj+1],numero_ventana[1][vvj],numero_ventana[2][vvj],
     numero_ventana[3][vvj],numero_ventana[4][vvj],numero_ventana[5][vvj]);
  }
fclose (fp);
}
/*****************************calculoRS() ************************************
Realiza las iteraciones correspondientes para obtener el Rango de cada venana
*****************************************************************************/
void calculoRS(void)
 {
  long int b,c,aux1=0,aux2=0,aux3=0,val,interv;
  float acumula=0.0,captura,acDesv1,val_min,val_max,valor_media,Ria,Sia,desv1,Rian;
  val=m1;
  interv=m2;
  /*CADA NODO RECIBIRA 2 VALORES, numero_ventana[0][vvj] vvj+1 y numero_ventana[1][vvj] val*/
  numero_ventana[0][vvj]=vvj+1;
  numero_ventana[1][vvj]=val;
  for(b=0;b<val;b++)
            {

             for(c=0;c<interv;c++)
               {
                captura=SerieN[c+aux2];
                acumula=acumula+captura;
                mediasi[b]=acumula/interv;
               }

             aux2=aux2+c;

             acumula=0.0;
             val_min=0.0;
             val_max=0.0;
             acDesv1=0.0;
             for(c=0;c<interv;c++)
               {
                captura=SerieN[c+aux3];
                valor_media=captura-mediasi[aux1];
                desv1=valor_media*valor_media;
                if (val_min>valor_media)
                        val_min=valor_media;
                if(val_max<valor_media)
                   val_max=valor_media;
                acDesv1=acDesv1+desv1;
               }
             aux3=aux3+c;
             aux1++;
             /*Calculo del rango RIa*/
             if(val_min<0)
               Ria=val_max+(val_min*-1);
             else
               Ria=val_max-val_min;
```

```
/**********Calculo de la desviacion estandar muestral Sia para cada periodo***/
            Sia=sqrt((1.0/interv)*acDesv1);
            Rian=Ria/Sia;
            R_S=Rian+R_S;
           }
            R_S=R_S/val;
        MPI_Send(&R_S,1,MPI_FLOAT,0,0,MPI_COMM_WORLD);
    /*EL VALOR DE R_S ES EL QUE ENVIARA CADA NODO */
            numero_ventana[3][minodo]=R_S;
 }

/************************* calculoH()  ***********************************
Calcula es estimado H, una vez obtenido el rango reescalado R/S por medio
de la regresion por minimos cuadrados
*************************************************************************/
int calculoH(void)
{
long int valores;
float mediax=0.0,mediay=0.0,desvTx=0.0,desvTy=0.0,desvTxy=0.0;
printf("\n\t\tR_ventanas vale: %ld",R_ventanas);
for(vvj=0;vvj<size;vvj++)
  {
    numero_ventana[4][vvj]=log(numero_ventana[1][vvj]);
    numero_ventana[5][vvj]=log(numero_ventana[3][vvj]);
    mediax=mediax+numero_ventana[4][vvj];
    mediay=mediay+numero_ventana[5][vvj];
  }
  printf("\nvalor acum de x: %f, valor acum y: %f, vvj:%ld",mediax,mediay,vvj);
  valores=(vvj);
  printf("\nvalores vale: %ld",valores);
  mediax=(mediax/valores);
  mediay=(mediay/valores);
  printf("\n Media de x: %f,  media de y: %f",mediax,mediay);
/***************Calculo de la desviacion tipica**********************/
  for(vvj=0;vvj<VENTAN;vvj++)
  {
    numero_ventana[6][vvj]=pow(numero_ventana[4][vvj]-(mediax),2);
    desvTx=desvTx+numero_ventana[6][vvj];
    numero_ventana[7][vvj]=pow(numero_ventana[5][vvj]-(mediay),2);
    desvTy=desvTy+numero_ventana[7][vvj];
    desvTxy=desvTxy+(((numero_ventana[4][vvj]-(mediax))*(numero_ventana[5][vvj]-(mediay)))/valores);
  }
  desvTx=sqrt(desvTx/valores);
  desvTy=sqrt(desvTy/valores);
  printf("\nDesv.Tipica X=%f, Desv.Tipica Y=%f, Desv.Tipica XY=%f",desvTx,desvTy,desvTxy);
  pendiente=desvTxy/(pow(desvTx,2));
  if (pendiente<0)
    pendiente=pendiente*-1;
   D=pendiente;
  D=2-D;
printf("\nEl exponente de Hurst es: %1.5f y La dimension fractal D es=%f\n",pendiente,D);
return 0;
}
```

MUESTRA DE CÁLCULOS PARCIALES MANUALES PARA UNA SERIE DE 500 DATOS DESDE LA HOJA DE CÁLCULO MS EXCEL 2003



Microsoft Excel - CALCULOS RS CON 500 DATOS.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	n=		16	480															
2	SUBPERIODOS		30	A															
3	INTERVALO SUBPERIODO a=1			Ia=[]			INTERVALO a=2			Ia=[]			INTERVALO a=3			Ia=[]			INTERVA
4			K=1,2,3,...48	SERIE DE TIEMPO ACUMULADA (Sk,o)	DETERMINO DESV. EST. (Mko-xo)^2					SERIE DE TIEMPO	DETERM DESV. EST. (Mko-xo)^2					SERIE DE TIEMPO	DETERMINO DESV. EST. (Mko-xo)^2		
5	k	dato M	DATO (N i,a)	(N i,a)-ea				DATO	(N i,a)-ea					DATO	(N i,a)-ea				
6	1	0.00154364	0.001543644	-0.00198024	3.9214E-06		1	0.0032931	0.00329313	-5.3456E-05	2.8575E-09		1	0.0017924	0.00179241	-0.00183566	3.3696E-06		1
7	2	0.00861265	0.00861265	0.005088766	2.5896E-05		2	0.0014572	0.0014572	-0.00188938	3.5698E-06		2	0.0028527	0.00285268	-0.00077539	6.0124E-07		2
8	3	0.00941774	0.009417737	0.005893853	3.4737E-05		3	0.0054427	0.00544275	0.00209616	4.3939E-06		3	0.0039087	0.0039087	0.00026063	7.8761E-08		3
9	4	0.00223297	0.002232971	-0.001290913	1.6665E-06		4	0.0038779	0.00387795	0.00053136	2.8234E-07		4	0.0017843	0.00178427	-0.0018438	3.3996E-06		4
10	5	0.00074354	0.00074354	-0.002780344	7.7303E-06		5	0.00756	0.00755998	0.00421339	1.7753E-05		5	0.0031986	0.0031986	-0.00042947	1.8445E-07		5
11	6	0.00598374	0.005983742	0.002459858	6.0509E-06		6	0.0003627	0.00036274	-0.00298385	8.9034E-06		6	0.0003563	0.00035632	-0.00327176	1.0704E-05		6
12	7	0.00296206	0.002962062	-0.000561822	3.1564E-07		7	0.0057054	0.00570538	0.0023688	5.5639E-06		7	0.0021313	0.00213126	-0.00149681	2.2404E-06		7
13	8	0.00111759	0.001117586	-0.002406298	5.7903E-06		8	0.0007085	0.00070852	-0.00263805	6.9594E-06		8	0.0021057	0.00210569	-0.00152238	2.3176E-06		8
14	9	0.00113167	0.001131673	-0.002392211	5.7227E-06		9	0.0026107	0.00251065	-0.00083593	6.9878E-07		9	0.0020618	0.00206176	-0.00156631	2.4533E-06		9
15	10	0.00077034	0.000770338	-0.002753546	7.582E-06		10	0.0023223	0.0023223	-0.00102429	1.0492E-06		10	0.0021445	0.00214445	-0.00148357	2.201E-06		10
16	11	0.00360889	0.003608886	-0.000162998	1.0005E-05		11	0.0108621	0.01086207	0.00751548	5.6483E-05		11	0.0017558	0.00175582	-0.00187225	3.5053E-06		11
17	12	0.00590746	0.005907459	0.002383575	5.6814E-06		12	0.0025626	0.00256265	-0.00078394	6.1456E-07		12	0.0020439	0.00204391	-0.00158416	2.5096E-06		12
18	13	0.00780418	0.007804175	0.004280291	1.8321E-05		13	0.0007301	0.00073011	-0.00261648	6.846E-06		13	0.000345	0.00034501	-0.00328306	1.0778E-05		13
19	14	0.00369633	0.003696332	0.000172448	2.9738E-08		14	0.0039778	0.00397777	0.00063118	3.9839E-07		14	0.0133852	0.01338515	0.00975708	9.5201E-05		14
20	15	0.00220397	0.002203971	-0.001319913	1.7422E-06		15	0.0016011	0.00160109	-0.0018455	3.4059E-06		15	0.012534	0.01253398	0.00890591	7.9315E-05		15
21	16	0.00189338	0.001893384	-0.0016305	2.6585E-06		16	0.0006711	0.0006711	-0.00267549	7.1582E-06		16	0.0056491	0.00564911	0.00202104	4.0846E-06		16
64				(N11-e1)+(N21	0.00013785					(N11-e1)+(N2	0.00012408					(N11-e1)+(N2	0.00022294		
55																			
56	Media			Max(X k,a)			Media			Max(X k,a)			Media			Max(X k,a)			Media
57	VALOR PROMEDIO ea		0.003523884	0.005893853			VALOR PROMEDIO ea		0.00334658	0.00751548			VALOR PROMEDIO ea		0.00362807	0.00975708			VALOR F
58	(MEDIA)SEGÚN FORMULA			Min(X k,a)			(MEDIA)SEGÚN FORMULA			Min(X k,a)			(MEDIA)SEGÚN FORMULA			Min(X k,a)			(MEDIA)S
59			0.003523884	-0.003162998	0.003163				0.00334658	-0.00298385	0.00298385				0.00362807	-0.00328306	0.00328306		
60	ea. = 1/n(N11+N21+N31+N41		+ N481)				ea. = 1/n(N11+N21+N31+N41		+ N481)				ea. = 1/n(N11+N21+N31+N41		+ N481)				ea. = 1/n(
61			4 -	Rla (Max-Min)					4 -	Rla (Max-Min)					4 -	Rla (Max-Min)			
62				0.009056851	0.00273085					0.01049933	0.00453164					0.01304014	0.00647402		
63				0.009056851						0.01049933						0.01304014			
64	5 -						5 -						6 -						5 -
65	DESVIACION		8.61562E-06	Sla			DESVIACION		7.7551E-06	Sla			DESVIACION		1.3934E-05	Sla			DESVIAC
66	ESTANDAR		0.002935238	0.002935238			ESTANDAR		0.0027848	0.0027848			ESTANDAR		0.0037326	0.0037326			ESTAND
67																			
68	6 -						6 -						6 -						6 -
70	RANGO ESCALDADO			Rla Sla			RANGO ESCALDADO			Rla Sla			RANGO ESCALDADO			Rla Sla			RANGO E
71	Rla Sla PARA Ia=			3.085559009			Rla Sla PARA Ia=			3.77023074			Rla Sla PARA Ia=			3.49336589			Rla Sla P
73	6b -																		
74	RANGO REESCALADO DE LA VENTANA 10 (PROMEDIO Rla Sla)																		
75	(R/S)n = (R/S)10			POR FORMULA=	3.47077054														
76									0.3889064										
77				PROMEDIO EXCEL=	3.47077054														
78																			
79	VALOR LOGARITMICO			ln(R/S)	log10(R/S)n														
80	DEL RANGO ESCALADO			1.244376627	0.5404259														
81	LOG(R/S)																		

MUESTRA DE 500 DATOS UTILIZADOS EN LA EJECUCIÓN Y PRUEBAS

0.001543644	0.005934078	0.005027698	0.008216736	0.014069071	0.007857835	0.000392069	0.008532989	0.018811608	0.012915551
0.00861265	0.005247061	0.004703209	0.009375724	0.003281244	0.013397332	0.003625164	0.000425714	0.01293086	0.02670219
0.009417737	0.004184076	3.65E-05	0.009784497	0.003688192	0.000414118	0.002917694	0.004987629	0.007885169	0.015650061
0.002232971	0.003115775	0.005100226	0.001865226	0.008686417	0.000414604	0.007714707	0.007220043	0.009243358	0.012648425
0.00074354	0.003869423	0.012913867	0.001122095	0.004120681	0.000413876	0.000411208	0.00136639	0.03143574	0.007115074
0.005983742	0.004819636	0.003282937	0.006030756	0.001249674	0.002907345	0.009452891	0.001354655	0.055842645	0.003727072
0.002962062	0.01055657	0.005720955	0.008157888	0.007242709	0.007177676	0.005646791	0.003637856	0.044112331	0.031011667
0.001117586	0.0051277	0.000364019	0.002653312	0.000834344	0.007952158	0.004797713	0.002196329	0.017838848	0.02788501
0.001131673	0.000675687	0.002874422	0.010367757	0.011383616	0.007069681	0.006322905	0.007220043	0.006725015	0.002842236
0.000770338	0.00661233	0.007245743	0.009430745	0.004606575	0.002940157	0.005572862	0.000900767	0.005292343	0.014446738
0.000360886	0.012053871	0.001076354	0.004204575	0.002086478	0.033215016	0.009718382	0.005440409	0.006756676	0.002767934
0.005907459	0.000661154	0.002883212	0.005265713	0.004537978	0.033241241	0.005572862	0.002837006	0.011891576	0.004823818
0.007804175	0.007549701	0.004679268	0.017778961	0.006247085	0.005133218	0.009739076	0.009153614	0.012423865	0.011770807
0.003696332	0.009820104	0.00682853	0.003785377	0.004904487	0.008497734	0.008985921	0.004163546	0.007846979	0.012208775
0.002203971	0.012045508	0.008355328	0.004187455	0.007563477	0.007761953	0.000441252	0.001840227	0.001973076	0.012734054
0.001893384	0.006648467	0.003945318	0.001126382	0.001629279	0.003832566	0.001785624	0.002323718	0.00558352	7.59E-05
0.003293129	0.006648467	0.00943791	0.002667612	0.002242781	0.015610055	1.65E-10	0.002702071	0.017334416	0.009163817
0.001457202	0.008786946	0.003265664	0.004842249	0.008595905	0.022129221	1.65E-10	0.005775216	0.010116149	0.000693019
0.005442745	0.014784901	0.004703209	0.003415065	0.011660447	0.015592623	0.00088278	0.012322548	0.003648168	0.003348566
0.003877945	0.015809886	0.003257542	0.007376854	0.013722585	0.00784375	0.022190891	0.026300391	0.005581783	0.013008902
0.007559976	0.005527473	0.002177551	0.005197844	0.001681586	0.000770689	0.020982573	0.024318311	0.013020748	0.030161883
0.000362736	0.0192338	0.001443076	0.00883175	0.005935436	0.002015385	0.005282585	0.00882115	0.002018864	0.034199821
0.005705381	0.025973453	0.009308882	0.017403014	0.002493038	0.001542218	0.004199261	0.007461625	0.006861743	0.001360008
0.000708524	0.024105894	0.006814676	0.00691823	0.004664589	0.00756388	0.006329021	0.001463485	0.002056192	0.011110773
0.002510653	0.003963243	0.01485011	0.025525471	0.005974384	0.008542855	0.002362133	0.013391273	0.010938887	0.019266264
0.002322295	0.000660228	0.004655569	0.003447406	0.002952433	0.031898892	0.000916535	0.035818044	0.004537698	0.01454671
0.010862069	0.023961039	0.009462399	0.003959395	0.002554272	0.031965071	0.02202342	0.014907257	0.038589268	0.003755808
0.002562647	0.012752792	0.000359394	0.033719303	0.022072477	0.004528793	0.038154875	0.006979522	0.019304401	0.007310761
0.000730105	0.002208474	0.003585742	0.017936454	0.027369359	0.004481821	0.010272918	0.00669342	0.032803497	0.011663353
0.003977767	0.008086421	0.000359576	0.002801302	0.005329976	0.00153056	0.011045016	0.006437348	0.063236489	0.000403946
0.001501085	0.001284399	0.004300713	0.045294934	0.009025097	0.006066152	0.0232657	0.011946071	0.039943646	0.003160606
0.000671095	0.005146496	0.003945318	0.064330867	0.012950218	0.00039294	0.045159299	0.002351673	0.019985288	0.001613479
0.001792414	0.002016466	0.015549201	0.002560438	0.010062491	0.001611635	0.01460919	0.04467423	0.016219317	0.004461256
0.002852679	0.008826219	0.000714972	0.000833857	0.002709225	0.000404177	0.003580295	0.042485635	0.009961997	0.000414122
0.0039087	0.003078342	0.015549201	0.017314319	0	0.001210914	0.004491845	0.001986256	0.006855651	0.009861004
0.001784272	0.001098155	0.004984661	0.019908869	0.002265644	0.005208924	0.022519821	0.004508666	0.01232163	0.00767066
0.0031986	0.015939882	0.004180361	0.008815337	0.007788103	0.006025706	0.007432061	0.000588533	0.008159015	0.019965736
0.000356315	0.023461321	0.008814709	0.061238861	0.020575083	0.01249409	0.003458124	0.022029209	0.006227287	0.024467073
0.002131264	0.004292191	0.005985181	0.066129041	0.016284986	0.01166803	0.003944207	0.014020604	0.001369036	0.020229406
0.002105694	0.004014472	0.001051982	0.001593481	0.021029085	0.000812767	0.003968804	0.01609569	0.004461256	0.004929256
0.00206176	0.001211443	0.003526724	0.011008126	0.017231196	0.015623695	0.008255104	0.024852074	0.010969409	0.009568573
0.0021445	0.009137468	0.004249151	0.016135481	0.001044644	0.015189525	0.00043988	0.022607289	0.001332907	0.001519027
0.001755822	0.00367329	0.004942354	0.027929451	0.005932704	0.013130755	0.009271572	0.018188461	0.010908991	0.003811906
0.002043909	0.00251687	0.010364652	0.023648485	0.006276206	0.00830366	0.005344056	0.011495735	0.014942791	0.004948609
0.000345014	0.000292008	0.00236961	0.002982678	0.005586409	0.002435502	0.01327487	0.029850781	0.015298304	0.016256251
0.013385151	0.001816355	0.00196717	0.014201086	0.017890276	0.003637596	0.005098793	0.01787044	0.006501181	0.012734054
0.012533979	0.00250938	0.008428183	0.015207184	0.002563532	0.003637596	0.015234147	0.007946538	0.035680686	0.001960315
0.00564911	0.006156666	0.008266757	0.007404327	0.017927167	0.001222887	0.011453227	0.019041866	0.043482511	0.001541949