UNIVERSIDAD DE MURCIA

Departamento de Informática y Sistemas

ASPECTOS COMPUTACIONALES DE LA RESOLUCIÓN Y OBTENCIÓN DE MODELOS DE ECUACIONES SIMULTÁNEAS

José Juan López Espín

Diciembre 2009

Dirigida por:
Domingo Giménez Cánovas

Veo esta tesis como la culminación de muchos años de trabajo y de ilusión. Y en este momento me vienen al pensamiento las personas que de alguna forma me han apoyado, ayudado y consolado, e incluso regañado si ha hecho falta.

En primer lugar, me acuerdo de mi madre, que nos dejó cuando solo tenía 12 años, y que estaría muy orgullosa de verme defender esta tesis. También me acuerdo de mis tías Chon y Trini (ya fallecida), y de mi Tata, que tomaron las riendas de mi vida haciendo de madres. A ellas mi agradecimiento eterno. También de mi padre, de Paca (es casi imposible contar la de botes de comida que me ha preparado para pasar la semana), y de sus tres hijas que han sido como hermanas para mí. De mis amigos del pueblo, de la facultad y de la Universidad Miguel Hernández, y de tantas y tantas personas. También de la familia de mi mujer (no habría conseguido la tesis sin las comidas de mi suegra), que son mi familia.

Y por supuesto tengo en mi mente a Maribel, mi compañera incansable que ha estado a mi lado todos estos años aguantando teoremas, computaciones paralelas y demás cosas que podría llevarle cada día. Gracias por tu apoyo.

Y de mi tesoro, mi hija Andrea, que con tan solo un año y poco ya quiere sentarse en el ordenador, escuchar música, tocar la guitarra... (¿a quién le habrá salido?)

En el plano profesional quiero agradecer a mi director de tesis la gran paciencia que ha tenido conmigo (a veces yo también con él). Ha sabido guiarme, enseñarme, y alguna vez regañarme. Pero siempre he notado que confiaba en mí y en mis posibilidades. Gracias Domingo.

A mi hija Andrea

Seamos realistas. Hagamos lo imposible
Ernesto Che Guevara

Índice general

Capítulo 1

Introducción

En este trabajo se desarrollan, estudian y comparan algoritmos para la resolución de Modelos de Ecuaciones Simultáneas (M.E.S.) tanto en secuencial como en paralelo, utilizando diferentes estimadores. Además se estudian descomposiciones matriciales para acelerar los algoritmos básicos en secuencial llevándolas a continuación a la versión paralela.

También se da respuesta a otra carencia encontrada en el software desarrollado hasta ahora, la de una herramienta capaz de encontrar un M.E.S. eficiente a partir de un conjunto de datos de variables.

1.1. Breve introducción histórica

Los Modelos de Ecuaciones Simultáneas forman parte del conjunto de técnicas estadísticas desarrolladas a partir de los modelos de ecuaciones de regresión [37, 38] con motivo de abarcar problemas que estos, de forma aislada, no podían resolver. Los Modelos de Ecuaciones Simultáneas nacieron en la primera mitad del siglo XX [27], siendo desarrollada la mayor parte de su cuerpo teórico en los años posteriores (en unos veinte años se había desarrollado la mayor parte de lo que se conoce hoy). Su desarrollo práctico y la aplicación a problemas reales se retrasó debido al problema computacional asociado, ya que las ecuaciones podían hacerse relativamente grandes y las técnicas de resolución necesitaban de un nivel de computación mayor al prestado en su momento por los ordenadores de la época [24, 63].

Con el paso de los años el mundo de la informática iba mejorando en todos sus aspectos. Las computadoras ganaban en velocidad, memoria, prestaciones, etc., con lo que los modelos de ecuaciones simultáneas fueron tomando una gran importancia ya que su implementación experimental no se veía excesivamente mermando por la computación.

El paso gigante fue la invención del ordenador personal y la facilidad de manejo del sistema operativo, programas, compiladores, etc. Esto hizo que muchísimos económetras no se vieran en la necesidad de trabajar conjuntamente con un equipo

de especialistas informáticos, sino que pudieran desarrollar sus propios programas y manejar sus propios ordenadores.

Los modelos de ecuaciones simultáneas son métodos estadísticos que nacieron dentro del mundo de la economía [27]. De hecho tuvieron un papel muy importante en el nacimiento de la rama de econometría, puesto que daban unas expectativas enormes en la estimación de variables económicas.

Desde su nacimiento (en los años treinta) hasta su declive (a finales de los setenta, principio de los ochenta), los Modelos de Ecuaciones Simultáneas fueron tratados, estudiados y desarrollados expresamente por económetras, es decir, se desarrollaron como una rama de la economía. Por esto hasta muy recientemente todas las aplicaciones se han desarrollado en economía y todas las conclusiones se han dado en este mismo campo, hasta el punto de entenderse esta herramienta como una herramienta económica y no estadística.

En los años cincuenta y sesenta, el peso de la econometría y de los modelos de ecuaciones simultáneas era inmenso. Se pensaba que se podría modelar cualquier proceso econométrico por complejo que este pareciera. Se desarrollaron modelos de ecuaciones simultáneas gigantescos que pretendían modelar la economía de un producto, de una región, de un país... Solo le ponía fronteras a dichos modelos la capacidad computacional del momento.

A principios de los setenta había modelos a escala mundial y las computadoras ya daban respuesta a estos modelos.

Pero la crisis del petróleo de mediados de los setenta dio un vuelco total a la econometría. Ningún modelo económico fue capaz de predecir tal desastre y, lo que es aún peor, a partir de tal punto muchos de ellos se resintieron (en el sentido de que no se ajustaban bien a la nueva situación económica) y hubo que reajustarlos, cambiarlos o, simplemente, desecharlos.

En este punto se introdujo un nuevo pensamiento en la economía y más concretamente en el mundo de la econometría. Esta nueva corriente pensaba que no era posible modelar a gran escala debido a que el mundo de la economía es un mundo caótico (en el que un atentado en un cierto instante puede cambiar el curso de la economía mundial) y el problema era que las ecuaciones simultáneas, que surgieron por tener una precisión que la regresión no daba, eran muy sensibles a elementos que por naturaleza son caóticos.

A finales de los setenta y principios de los ochenta se comenzaron a desechar modelos de ecuaciones simultáneas grandes (y han seguido en desuso hasta no hace mucho) debido a que al utilizar muchas variables en los modelos, se tenía más posibilidades de que alguna de ellas sufriera alteraciones y contaminara al modelo entero. La filosofía era usar pocas ecuaciones y pocas variables pero que fueran las verdaderamente importantes para la predicción. Con esto no se aseguraba que las estimaciones fueran precisas, pero sí que no estaban influenciadas por las fluctuaciones de otras variables de menos peso en el modelo.

Y como hasta el momento los modelos de ecuaciones simultáneas solo se habían usado en economía, cayeron en desuso. Se usaban poco y de tamaño pequeño (con

lo cual el nivel de computación exigido también era pequeño). Además, como se asumía el error que se cometía, no era necesario el usar técnicas desarrolladas para los modelos de ecuaciones simultáneas sino que simplemente se resolvían aplicando técnicas de regresión (Mínimos Cuadrados Ordinarios) aceptando que se estaba cometiendo un error de partida y que dichas técnicas vienen acompañadas de un sesgo importante. Pero merecía la pena puesto que el error no era significativo frente al ya asumido y porque el problema se simplificaba.

Actualmente se abre una nueva esperanza para los modelos de ecuaciones simultáneas. Otras ramas de tipo científico han comenzado a usar estas técnicas estadísticas para dar respuesta a problemas que no pueden ser resueltos con modelos de regresión. Por ejemplo, en el estudio de redes y el movimiento de paquetes a través de nodos [39], se ha comprobado que los modelos de ecuaciones simultáneas dan buena respuesta a la hora de modelar dicho problema, con lo que se pueden predecir tiempos de espera, tamaño de colas en los distintos nodos, etc. También se han desarrollado trabajos en medicina [35, 58, 59], en psicología [40], e incluso para modelar el tráfico aéreo en Nueva York [60] o el número de divorcios con respecto a los ingresos femeninos [67].

1.2. Estado del arte

Básicamente, con lo que respecta a sofware desarrollado para la resolución de Modelos de Ecuaciones Simultáneas, podemos hablar de trabajos anteriores refiriéndonos a software de tipo comercial (o en algunos casos libre) en el cual se implementen las herramientas de resolución de Modelos de Ecuaciones Simultáneas.

En marzo de 1994, QMS inició una revolución en el software de econometría con el lanzamiento de Eviews 1.0 [10], con una moderna interface gráfica orientada a objetos para el usuario. Actualmente está implementada su versión 7.0.

Eviews lleva incorporadas la gran mayoría de técnicas de resolución de regresión y de modelos de ecuaciones simultáneas (además de series temporales y demás partes de la econometría). Sin embargo, no contempla herramientas de búsqueda del mejor modelo. Tampoco hasta ahora ha desarrollado una versión paralela de su software.

Otro programa destinado también a la econometría y de difusión gratuita es Ox [5] (OxEdit, OxMetric, y demás productos) perteneciente a doornik y que se puede descargar directamente de su web `www.doornik.com/download.html`. En este caso se disponen de menos herramientas que en el anterior (con las mismas carencias).

Actualmente muchos paquetes estadísticos llevan implementado Mínimos Cuadrados en dos Etapas (MC2E), que es uno de los estimadores más usados en los modelos de ecuaciones simultáneas. Sin embargo su implementación no es para resolución de estos sistemas sino para resolver una ecuación de regresión usando la información almacenada en otras variables que no entren en el modelo. Es decir, exactamente lo que hace MC2E al resolver una ecuación en un modelo pero de forma aislada. Por ejemplo la versión de SPSS 15.0 [12] ya lleva implementado MC2E.

En [48] se desarrollan y estudian algoritmos para estimadores de información completa de los M.E.S., más concretamente MC3E [37, 38]. Sin embargo, hasta donde nosotros sabemos, no se han abordado desde el prisma computacional algoritmos de información limitada (que resuelvan cada una de las ecuaciones por separado), por lo que este se convierte en uno de los objetivos del presente trabajo.

1.3. Objetivos del presente trabajo

Los **objetivos** del presente trabajo son:

- **Resolver Modelos de Ecuaciones Simultáneas** de manera más eficiente reduciendo el tiempo de ejecución y los recursos en memoria.
- **Desarrollar una herramienta** que permita **encontrar el mejor modelo** posible a partir de un conjunto de datos de variables.
- **Aplicar** las herramientas desarrolladas a **problemas reales**.

Para el primer objetivo:

- Se utilizan descomposiciones matriciales, como la descomposición de la matriz inversa y la descomposición QR, para reducir el coste computacional del algoritmo.
- Se desarrollan versiones paralelas en memoria compartida y distribuida.

Esto permitirá abordar Modelos de Ecuaciones Simultáneas con gran número de ecuaciones y de variables. La gran cantidad de memoria y de tiempo de ejecución requeridos por este tipo de modelos hace de los clusters y de los algoritmos de memoria distribuida una herramienta importante para su resolución. También es importante desarrollar versiones para memoria compartida dado el auge actual de los multicores.

Además de desarrollar algoritmos que se ejecuten en sistemas de altas prestaciones, la finalidad de la obtención de dichos algoritmos es que sean lo más portables posible y que puedan ser ejecutados en diversos sistemas. Para ello se utilizarán llamadas a las librerías de LAPACK [6, 16] y BLAS [2, 25] en la versión secuencial, y a ScaLAPACK [11, 18] y PBLAS [8, 23] en las de paso de mensajes.

Para alcanzar el segundo objetivo, se utilizan:

- Técnicas Metaheurísticas. Dado que se tiene una necesidad computacional inmensa debido a lo extenso del espacio de soluciones donde se debe buscar el mejor modelo, se evitan métodos de búsqueda exhaustiva utilizando en su lugar metaheurísticas.

- Uso de Avance Rápido. Se desarrollan versiones híbridas en que se combinan algoritmos genéticos con técnicas de avance rápido, consiguiendo una mejora muy importante en la solución encontrada. El problema de este nuevo algoritmo es la gran carga computacional que añade, lo que justifica el desarrollo y estudio de una versión en paralelo.

También se plantea como objetivo el aplicar las técnicas estudiadas a problemas reales, lo que adicionalmente servirá para validar los métodos de obtención de modelos.

1.4. Aportaciones

Las aportaciones hechas por este trabajo y que tienen un carácter novedoso se pueden clasificar en los siguientes puntos:

- Una primera aportación ha sido el estudio y comparación teórica y experimental y desarrollo en paralelo de algoritmos para la resolución de M.E.S. basados en estimadores de información limitada. De hecho, solo tenemos constancia de que se hayan estudiado y desarrollado algoritmos en paralelo para estimadores de información completa [48].

- La segunda aportación ha sido utilizar descomposiciones matriciales para acelerar los algoritmos descritos, estudiándolos y comparándolos teórica y experimentalmente. Se han utilizado para ello técnicas e ideas usadas en modelos de regresión, pero que hasta ahora no se habían usado en M.E.S. Algunas de las técnicas usadas son la descomposición QR, la transformada de Householder, las rotaciones de Givens [34], creación de árboles para reducir el número de retriangularizaciones [73], etc. Otra aportación es el uso de una descomposición de la inversa y su comparación con la QR. Que nosotros sepamos, esta descomposición de la inversa no ha sido utilizada para la resolución de modelos basados en mínimos cuadrados de ningún tipo.

- La tercera aportación ha sido diseñar un algoritmo capaz de encontrar un Modelo de Ecuaciones Simultáneas lo más eficiente posible de acuerdo con un parámetro establecido, que se ajuste a un conjunto de datos de variables. Para este fin se ha usado algoritmos genéticos y se han comparado diferentes parámetros de criterio (*parameter criteria*) desarrollados previamente para M.E.S. También se ha desarrollado una versión más eficiente de ésta metaheurística, aunque más costosa computacionalmente, utilizando avance rápido en la mutación.

- Por último, se han utilizado las herramientas desarrolladas para encontrar el mejor M.E.S. posible acorde a un conjunto de datos, en dos problemas reales. En primer lugar se ha tomado uno de los modelos clásicos de econometría

propuesto por Keynes [37, 66] y un conjunto de datos de las variables que participan en él, siendo comparado con el modelo obtenido por el programa para los mismos datos y variables. En segundo lugar se ha trabajado junto con un grupo de ginecología del Hospital Virgen de la Arrixaca (perteneciente al grupo *Harris Birthright Research Center for Fetal Medicine* del *King´s College Hospital* de Londres) en la obtención de un M.E.S. en el que interrelacione un conjunto de variables utilizadas posteriormente para predecir preeclamsia [19, 26, 45, 68].

1.5. Metodología

La metodología utilizada en el primer objetivo (sección 1.3) ha sido la siguiente:

- Se han desarrollado algoritmos para la resolución de M.E.S. realizando un estudio teórico del tiempo de ejecución. El estudio del tiempo de ejecución permite la comparación del coste computacional de los algoritmos.
- Se han mejorado los algoritmos mediante descomposiciones matriciales, consiguiendo algoritmos que obtienen los mismos resultados con menor coste computacional.
- Se han desarrollado algoritmos paralelos de los algoritmos ya estudiados, puesto que el coste computacional es alto aún usando descomposiciones matriciales.
- En todos los casos, se han implementado dichos algoritmos realizando a continuación un estudio experimental en diferentes sistemas. El uso de diferentes sistemas tiene la finalidad de evitar la dependencia de los tiempos de ejecución del sistema en el que han sido tomados.
- Se ha contrastado el estudio experimental con el estudio teórico.

En los diferentes estudios teóricos, se han considerado aproximaciones para facilitar su comparación. En el estudio experimental se han medido los tiempos variando diferentes parámetros (como el tamaño de la muestra, el número de ecuaciones ...)

La metodología utilizada en el segundo objetivo ha sido la siguiente:

- Se ha desarrollado un algoritmo genético para la obtención automática de un M.E.S. eficiente a partir de un conjunto de datos.
- Se ha adaptado el algoritmo (*tuning*) al problema con que se trabaja. Para este ajuste se han variado diferentes parámetros propios del algoritmo genético, y se ha contrastando el tiempo de ejecución y la calidad de la solución obtenida.
- Se han desarrollado mejoras del algoritmo utilizando avance rápido. Con esto se consigue un algoritmo híbrido que conjuga las ventajas de ambos métodos y que obtiene mejores soluciones aunque aumenta el coste considerablemente.

- Debido al incremento computacional que supone el uso del avance rápido, se ha desarrollado una versión en paralelo del algoritmo estudiado.
- En todos los casos, se han implementado los algoritmos realizando a continuación un estudio experimental.

La metodología utilizada en las aplicaciones (tercer objetivo) ha sido la siguiente:

- Aplicar el programa a los datos del problema.
- Contrastar el modelo. En el caso del Modelo Keynesiano Simple con el ya propuesto y en el caso médico mediante su análisis junto con el grupo de ginecología.
- Obtención de conclusiones de los modelos.

1.6. Contenido de la tesis doctoral

La tesis doctoral se estructura en 9 capítulos. El primero, como ya se ha visto, introduce al lector en el problema a abordar.

El capítulo 2 describe las herramientas básicas usadas en el trabajo. Se describen los clusters utilizados para los experimentos y las librerías software usadas en ellos. También se describen herramientas matemáticas que serán utilizadas a lo largo del trabajo. Por último se dedica la mitad del capítulo a introducir los Modelos de Ecuaciones Simultáneas y los tipos de estimadores que se abordarán en éste trabajo.

El capítulo 3 analiza y compara los algoritmos secuenciales basados en los estimadores MCI y MC2E. También se analiza una nueva versión del algoritmo basado en el estimador MC2E utilizando una descomposición de la matriz inversa. Con esta descomposición se reutilizan cálculos hechos previamente, con el consiguiente ahorro computacional.

En el capítulo 4 de estudian las versiones en paralelo de los algoritmos analizados en el capítulo 3. Los algoritmos son desarrollados en memoria distribuida y evaluados en diferentes clusters.

El capítulo 5 analiza diferentes versiones del algoritmo basado en el estimador MC2E, utilizando la descomposición QR. Se introduce la idea de obtener la descomposición QR de la matriz asociada a una ecuación a partir de la descomposición QR usada en los cálculos hechos al principio del algoritmo, evitando realizar una descomposición QR completa en cada ecuación y obteniendo por tanto el consiguiente ahorro computacional. También se desarrollan versiones paralelas en memoria compartida de dichos algoritmos.

En el capítulo 6 se estudian mejoras de los algoritmos planteados en el capítulo 5 mediante el diseño de un árbol donde hay nodos que corresponden a ecuaciones del Modelo de Ecuaciones Simultáneas y nodos ficticios que se introducen para reducir el número de operaciones a realizar al obtener la descomposición QR. Se

estudia experimentalmente para qué valores del tamaño del problema es rentable la introducción de nodos ficticios y cual es la mejora obtenida. También se estudia la paralelización en memoria compartida.

El capítulo 7 estudia la obtención de un M.E.S. a partir de un conjunto de datos de variables. Se utilizan metaheurísticas (concretamente algoritmos genéticos) para este fin, dado que el extenso espacio de posibles soluciones no hace eficiente el uso de una búsqueda exhaustiva. También se desarrolla una versión híbrida que combina algoritmos genéticos con avance rápido y que obtiene mejores resultados en cuanto a la solución hallada. Se estudia también la paralelización del algoritmo en memoria compartida.

El capítulo 8 recoge aplicaciones del trabajo realizado. Concretamente se utiliza el programa desarrollado en el capítulo 7 en dos casos diferentes. En el primero se aplica el algoritmo a uno de los problemas clásicos de econometría propuestos por Keynes [37]. Se compara el modelo hallado por el programa con el modelo de Keynes comentando las diferencias. En el segundo caso se trabaja con investigadores del grupo de ginecología del Hospital Virgen de la Arrixaca de Murcia para hallar un M.E.S. que ayude en la predicción de las variables implicadas en preeclampsia [19, 26, 45, 68].

El capítulo 9 recoge las principales conclusiones obtenidas con relación a esta tesis, el entorno donde se ha trabajado, los proyectos y las aportaciones realizadas a lo largo de este trabajo. Finalmente, se relacionan algunas posibles líneas futuras de trabajo.

Capítulo 2

Herramientas básicas y descripción teórica

Se describen en este capítulo las diferentes herramientas computacionales, tanto software como hardware, utilizadas en el trabajo. También se hace una descripción de varias herramientas matemáticas básicas que serán utilizadas como base para diferentes algoritmos en el presente trabajo. Éste es el caso de Mínimos Cuadrados y la descomposición QR.

La otra parte de capítulo se encarga de introducir de manera teórica los Modelos de Ecuaciones Simultáneas y los diferentes estimadores que serán usados a lo largo del trabajo.

2.1. Herramientas computacionales

Se resume en esta sección las máquinas utilizadas para el desarrollo y estudio de los algoritmos propuestos, y el software utilizado en dichas máquinas para este fin.

2.1.1. Hardware

En este trabajo se han utilizado clusters formados por nodos multiprocesadores con un número de cores entre de 2 y 8. Por lo tanto, se dispone de dos tipos o niveles de paralelismo, un nivel global que se corresponde con un esquema de paso de mensajes, puesto que los nodos están conectados mediante una red, y un nivel local que se corresponde con un esquema compartido, pues los procesadores o cores que componen cada nodo comparten una misma memoria física.

Los clusters utilizados son:

- Kefren (Universidad Politécnica de Valencia)

  Kefren es un cluster de memoria distribuida que trabaja bajo el ambiente Linux, Red Hat 8 (Figura 2.1). Consta de 20 nodos biprocesadores Pentium Xeon a 2 GHz, interconectados mediante una red SCI con topología de Toro

2D en malla de 4×5. Cada nodo consta de 1 Gigabyte de memoria RAM. Todos los nodos están disponibles para cálculo científico.

Figura 2.1: La máquina Kefren.

Kefren es gestionado por el Departamento de Sistemas Informáticos y Computación de la Universidad Politécnica de Valencia.

- Marenostrum (Barcelona Supercomputing Center)

  Marenostrum es un supercomputador basados en procesadores PowerPC, arquitectura BladeCenter y una red de interconexión Myrinet. Las principales características son: 10240 procesadores IBM Power PC 970MP de 2.3 GHz de velocidad (2560 JS21 blades), 20 TB de memoria RAM, 280 TB de almacenamiento en disco y un pico ejecución de 94.21 Teraflops.

  Marenostrum fue en el año 2005 el supercomputador más potente de Europa, llegando a estar también en el cuarto puesto mundial, aunque actualmente se sitúa en el puesto 61 (según la lista de TOP500 [13]). Está situado en una ermita en Barcelona (figura 2.2) y es gestionado por el *Barcelona Supercomputing Center* [1].

- Rosebud (Universidad Politécnica de Valencia)

  Rosebud es un cluster heterogéneo compuesto por 8 ordenadores conectados mediante una red Fast-Ethernet. Los 8 ordenadores se agrupan en 4 pares de computadores, diferentes entre sí en cuanto a potencia de cálculo y en cuanto a arquitectura. Los dos primeros, denominados rosebud01 y rosebud02, están formados por procesadores Pentium IV con velocidades de 1.6 GHz y 1.7 GHz

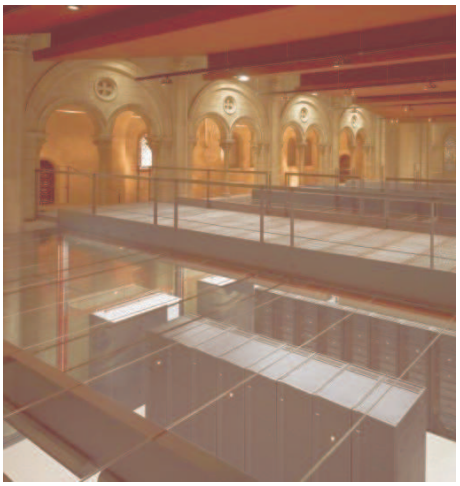

Figura 2.2: Marenostrum vista desde dentro de la ermita.

respectivamente, y una memoria RAM de 1 Gbyte. Los computadores del segundo par, denominados rosebud03 y rosebud04, son biprocesadores Xeon con una velocidad de 2.2 GHz y con 3.5 Gbyte de memoria RAM (figura 2.3 izquierda). Estos dos primeros pares presentan una arquitectura i386. El tercer par está formado por dos máquinas Fujitsu Primergy RXI600 (denotados por rosebud05 y rosebud06), cada una con 4 procesadores Dual-Core Intel Itanium2 a 1.4 GHz compartiendo 8 GByte de memoria RAM (figura 2.3 derecha). El procesador Itanium2 presenta una arquitectura de 64 bits. El cuarto par está formado por Intel Core 2 Quad, presentando cada core una velocidad de 2.66 GHz y compartiendo 4 GByte de memoria RAM. En total, en el cluster Rosebud se cuenta con 30 núcleos computacionales.

En este trabajo, todos los experimentos realizados en Rosebud han sido hechos en rosebud05 y rosebud06, por lo que se omitirá el número nombrando tan solo la máquina Rosebud.

Rosebud es gestionado por el Departamento de Sistemas Informáticos y Computación de la Universidad Politécnica de Valencia.

2.1.2. Software

Las herramientas software usadas para el presente trabajo se pueden dividir en dos grupos diferentes, una, los lenguajes de programación utilizados, y otra, las librerías usadas.

Como lenguajes de programación, se ha utilizado ANSI C para la programación secuencial, la librería de paso de mensajes MPI (Message Passing Interface) [15, 64, 71], para la programación en memoria distribuida, y el API OpenMP para memoria compartida [15, 21, 22, 64].

Figura 2.3: Máquina Rosebud. A la izquierda los nodos rosebud03 y rosebud04, a la derecha el nodo rosebud05.

MPI

Los algoritmos del capítulo 3 han sido desarrollados usando la librería MPI debido a su actualidad y disponibilidad de distribuciones en prácticamente cualquier arquitectura. Esta biblioteca proporciona tipos de datos, procedimientos y funciones que facilitan desarrollar aplicaciones paralelas para sistemas multiprocesadores, tanto para redes locales (LAN) como para redes de área amplia (WAN).

OpenMP

La librería OpenMP es un API que permite añadir concurrencia a los procesos mediante paralelismo de memoria compartida. Con OpenMP es posible desarrollar programas a un nivel superior que hacerlo creando hilos controlando la concurrencia de éstos. Con las directivas de OpenMP se hacen transparente las operaciones de concurrencia de threads así como la creación, inicialización, destrucción, etc. de threads.

OpenMP consta de tres componentes API principales: directivas del compilador, rutinas de librería de tiempo de ejecución y variables de entorno. Es una librería portable, pues está especificada para los lenguajes C/C++ y Fortran y se ha implementado en múltiples plataformas incluyendo Unix y Windows.

Librerías de álgebra lineal

Debido a la gran importancia que tiene el álgebra lineal dentro de los problemas que se puedan plantear en cualquiera de las ciencias, ingenierías, e incluso en aplicaciones de índole comercial, a principios de los años setenta se comenzaron a desarrollar aplicaciones para resolver problemas de este tipo, de manera más eficiente tanto en secuencial como en paralelo.

La librería BLAS [2, 25] se compone de funciones que se basan en la construcción de bloques para efectuar operaciones básicas entre vectores y matrices. BLAS

está construido en tres niveles: el nivel 1, que efectúa operaciones vector-vector y es llamado BLAS 1, el nivel 2, que efectúa operaciones matriz-vector, que es llamado BLAS 2, y el nivel 3, que efectúa operaciones matriz-matriz y es llamado BLAS 3. Esta herramienta es eficiente, portable y de amplia disponibilidad, por lo que se utiliza comúnmente en el desarrollo de software del álgebra lineal de altas prestaciones. La librería PBLAS [8, 23] contiene versiones en paralelo de las funciones desarrolladas en BLAS.

La librería LAPACK [6, 16] proporciona rutinas para resolver sistemas de ecuaciones, problemas de mínimos cuadrados, problemas de valores propios, vectores propios y valores singulares. También proporciona rutinas para factorización de matrices, tales como LU, Cholesky, QR, etc., tanto para matrices con estructuras específicas o matrices generales. La librería ScaLAPACK [11, 18] contiene versiones en paralelo de las funciones desarrolladas en LAPACK.

2.2. Herramientas matemáticas

Se explican en este apartado varias herramientas matemáticas básicas que serán utilizadas a lo largo del presente trabajo. En primer lugar se define la expresión del estimador de Mínimos Cuadrados añadiéndole de aquí en adelante, para evitar confusión con otros tipos de Mínimos Cuadrados que se estudiarán en este trabajo, el apelativo de Ordinarios. A continuación se trata la descomposición QR y dos métodos para su obtención, Householder y Givens.

2.2.1. Mínimos Cuadrados Ordinarios (MCO)

La expresión del estimador de Mínimos Cuadrados para el problema de regresión múltiple es descrita a continuación. Se considera una única ecuación donde la variable expresada o variable dependiente será denotada por z, y las variables en la expresión como explicativas o independientes, serán denotadas por $w_1, \ldots, w_n$. La variable aleatoria de ruido blanco será denotada por v. Así, tenemos:

$$z_t = \beta_1 w_{1,t} + \ldots + \beta_n w_{n,t} + v_t \tag{2.1}$$

siendo $z_t, w_{1,t}, \ldots, w_{n,t}, v_t$ elementos de las variables $z, w_1, \ldots, w_n, v$ respectivamente, con $t = 1, ..., d$ y siendo d el tamaño muestral.

La ecuación 2.1 puede expresarse en forma matricial de la siguiente manera:

$$z = W\beta + v \tag{2.2}$$

siendo:

$$z = \begin{pmatrix} z_1 \\ \vdots \\ z_d \end{pmatrix} \ , \ W = \begin{pmatrix} w_{1,1} & \ldots & w_{n,1} \\ \vdots & & \vdots \\ w_{1,d} & \ldots & w_{n,d} \end{pmatrix} \ , \ \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix} \ , \ v = \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} \tag{2.3}$$

La expresión del estimador MCO es $(W^TW)^{-1}W^Tz$. El vector resultante es el estimador de β, que denotaremos por $\hat{\beta}$. Con esta estimación se consigue calcular los coeficientes que relacionan la variable dependiente con las explicativas. Una vez obtenidos los coeficientes se pueden hacer estimaciones de la variable z a partir de W de la forma $\hat{z} = W\hat{\beta}$.

2.2.2. Descomposición QR

La descomposición QR de una matriz es una descomposición matricial que expresa la matriz original como producto de dos matrices tal como expresa la siguiente definición [34].

Proposición La descomposición QR de una matriz $A \in \mathbb{R}^{m\times n}, m \geq n$ es $A = QR$ (también se puede expresar como $Q^TA = R$) donde $Q \in \mathbb{R}^{m\times m}$ es una matriz ortogonal ($Q^TQ = I$) y $R \in \mathbb{R}^{m\times n}$ es una matriz triangular superior de la forma $\begin{pmatrix} R_1 \\ 0 \end{pmatrix} \begin{matrix} n \times n \\ (m-n) \times n \end{matrix}$, donde R_1 es una matriz triangular superior.□

Por lo tanto, se necesitan algoritmos para calcular la matriz Q y la matriz R a partir de una matriz dada A de forma que ambas matrices cumplan las condiciones de la definición. A continuación se exponen dos métodos para el cálculo de la descomposición QR de una matriz. Cada uno de ellos tiene características diferentes y serán utilizados en diferentes algoritmos para la resolución de M.E.S.

Descomposición QR mediante reflexiones de Householder

Una transformación de Householder o reflexión de Householder es una transformación que refleja el espacio con respecto a un plano determinado. Esta propiedad se puede utilizar para realizar la transformación QR de una matriz si tenemos en cuenta que es posible diseñar la matriz de Householder de manera que un vector elegido quede con una única componente no nula tras ser transformado (es decir, premultiplicando por la matriz de Householder). Gráficamente, esto significa que es posible reflejar el vector elegido respecto de un plano de forma que el reflejo quede sobre uno de los ejes de la base cartesiana.

La elección del plano de reflexión y la construcción de la matiz de Householder asociada, es de la forma siguiente: Sea $x \in \mathbb{R}^m$ un vector columna arbitrario tal que $||x||_2 = |\alpha|$, donde α es un escalar. Entonces, siendo e_1 el vector $(1, 0, ..., 0)^T$, se define $u = x - \alpha e_1, v = \frac{u}{||u||_2}$ y $Q = I - 2vv^T$, donde v es un vector unitario perpendicular al plano de reflexión elegido, Q es una matriz de Householder asociada a dicho plano y $Qx = (\alpha, 0, \cdots, 0)^T$.

Esta propiedad se puede usar para transformar gradualmente los vectores columna de una matriz A en una matriz triangular superior. En primer lugar se multiplica A con la matriz de Householder Q_1 que obtenemos al elegir como vector x la primera columna de la matriz. Esto proporciona una matriz QA con ceros en la primera columna (excepto el elemento de la primera fila):

$$Q_1 A = \begin{pmatrix} \alpha_1 & \star & \dots & \star \\ 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{pmatrix}$$

El procedimiento se puede repetir para A' (que se obtiene de A eliminando la primera fila y columna), obteniendo así una matriz de Householder Q'_2. Hay que tener en cuenta que Q'_2 es menor que Q_1. Para conseguir que esta matriz opere con $Q_1 A$ en lugar de A' es necesario expandirla hacia arriba y hacia la izquierda, completando con un uno en la diagonal, o en general:

$$Q_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & Q'_k \end{pmatrix}$$

donde I_k es la matriz identidad de dimensión $k \times k$.

Tras repetir el proceso t veces, donde $t = \text{mín}(m-1, n), R = Q_t \cdots Q_2 Q_1 A$ es una matriz triangular superior. Tomando $Q = Q_1 Q_2 \cdots Q_t, A = QR$ es una descomposición QR de la matriz A.

El coste de calcular la descomposición QR mediante el método de Householder de una matriz $A \in \mathbb{R}^{m \times n}, m \geq n$, viene dado por la siguiente expresión [7, 34]:

$$C_{HH}(n, m) = \frac{2}{3} n^2 (3m - n) \tag{2.4}$$

donde se ha contado el número de flops (floating point operation) necesarios para completar la descomposición QR de la matriz A.

Descomposición QR mediante rotaciones de Givens

Una rotación de Given situada en el plano (i, j) que reduce a cero el elemento $b_{j,k}$ cuando se aplica por la izquierda a la matriz $B \in \mathbb{R}^{m \times n}$ será denotada por $G^{(k)}_{i,j}$ donde $1 \leq i, j \leq m$ y $1 \leq k \leq n$. La rotación $G^{(k)}_{i,j} B$ afecta únicamente a la i-ésima y j-ésima filas de B. Los cambios en dichas filas pueden ser expresados como sigue:

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} b_{i,:} \\ b_{j,:} \end{pmatrix} = \begin{pmatrix} \tilde{b}_{i,1} & \dots & \tilde{b}_{i,k} & \dots & \tilde{b}_{i,n} \\ \tilde{b}_{j,1} & \dots & \tilde{b}_{j,k} & \dots & \tilde{b}_{j,n} \end{pmatrix} \tag{2.5}$$

donde $b_{j,k} \neq 0$, $c^2 + s^2 = 1$, $c = b_{i,k}/\tau$, $s = b_{j,k}/\tau$, $\tau^2 = b^2_{i,k} + b^2_{j,k}, \tilde{b}_{i,k} = \tau$ y $\tilde{b}_{j,k} = 0$.

Así, la matriz $G^{(k)}_{i,j} \in \mathbb{R}^{m \times m}$ queda definida de la siguiente forma:

$$G_{i,j}^{(k)} = \begin{array}{c} \\ i \rightarrow \\ \\ j \rightarrow \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & i & & j & & \\ & & \downarrow & & \downarrow & & \end{array} \\ \begin{pmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ 0 & \dots & c & \dots & s & \dots & 0 \\ 0 & & \vdots & \ddots & \vdots & & 0 \\ 0 & \dots & -s & \dots & c & \dots & 0 \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix} \end{array} \tag{2.6}$$

Puesto que esta matriz es ortogonal, si multiplicamos B por el conjunto de rotaciones de Givens que reducen a ceros los elementos por debajo de la diagonal principal en el orden descrito en la figura 2.4, habremos obtenido la descomposición QR de B. En la figura 2.4 se muestra el orden de eliminación de elementos en el proceso de descomposición QR mediante rotaciones de Givens de una matriz $A \in \mathbb{R}^{10\times 5}$. La entrada i (i=1,...,35) indica el elemento reducido a cero en la i-ésima rotación.

$$\begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ 9 & \bullet & \bullet & \bullet & \bullet \\ 8 & 17 & \bullet & \bullet & \bullet \\ 7 & 16 & 24 & \bullet & \bullet \\ 6 & 15 & 23 & 30 & \bullet \\ 5 & 14 & 22 & 29 & 35 \\ 4 & 13 & 21 & 28 & 34 \\ 3 & 12 & 20 & 27 & 33 \\ 2 & 11 & 19 & 26 & 32 \\ 1 & 10 & 18 & 25 & 31 \end{pmatrix}$$

Figura 2.4: Orden de anulación en la descomposición QR aplicada a $A \in \mathbb{R}^{10\times 5}$.

El número de rotaciones de Givens necesarias para calcular la descomposición QR de una matriz $A \in \mathbb{R}^{m\times n}$ es $\sum_{i=1}^{n}(m-i) = \frac{n}{2}(2m-n-1)$, y Q^T es definida como sigue:

$$Q^T = \prod_{i=1}^{n}\prod_{j=1}^{m-i} G_{m-j,m-j+1}^{(i)} \tag{2.7}$$

La construcción de una rotación de Givens requiere seis flops. El mismo tiempo es el empleado en aplicar la rotación a dos elementos, por lo que si tenemos en cuenta que al aplicar Givens en la expresión 2.5 no se calcula la primera columna (a $b_{i,k}$ se le asigna el valor de τ y $b_{j,k}$ se hace cero), el número de operaciones total será $6(n-k)$.

La complejidad de calcular 2.7 viene dada en la siguiente expresión:

$$C_G(n,m) = 6\sum_{i=1}^{n}(m-i)(n-i+1) = n(3m(n+1)-n^2-3n-2) \approx n^2(3m-n) \quad (2.8)$$

2.3. Descripción teórica

En esta sección se da una descripción básica de lo que son los Modelos de Ecuaciones Simultáneas, los tipos de variables que aparecen en ellos y los estimadores usados en este trabajo para resolverlos.

2.3.1. Modelo de Ecuaciones Simultáneas

Básicamente un Modelo de Ecuaciones Simultáneas es un conjunto de ecuaciones de regresión donde existe influencia simultánea entre variables y ecuaciones, y donde una variable que está en una ecuación (de dependiente o de explicativa) puede aparecer en otras ecuaciones [38]. Se supone que una variable solo aparecerá como dependiente en una ecuación, aunque puede aparecer como explicativa en varias de ellas.

Tres tipos de variables aparecen en un sistema de ecuaciones simultáneas:

- Endógenas: Son variables que influyen en el modelo y se ven influenciadas por él. Las representaremos por y y supondremos que el total de variables, N, coincide con el total de ecuaciones (a este tipo de sistemas se les denomina completos).

- Exógenas: Son variables que influyen en el modelo pero no se ven influenciadas por él. Las representaremos con x.

- Predeterminadas: Son aquellas formadas por las exógenas y endógenas retardadas. Una variable endógena retardada es una variable endógena que entra en una ecuación con datos retardados en el tiempo. Por ejemplo, en un sistema de predicción de la oferta y demanda de un artículo, el precio del mes pasado puede entrar en la ecuación de predicción de la demanda, y al ser un valor pasado no puede verse influenciado y variar por culpa del sistema, pero tampoco es una variable exógena, puesto que el precio actual sí se ve influenciado por el sistema. Es, por tanto, una variable predeterminada. Es decir, las variables predeterminadas son una extensión de las exógenas. Por comodidad de notación asumiremos que todas las variables predeterminadas en el sistema son exógenas.

- Ruido blanco: Son las variables de error que se acepta que existen en un modelo de ecuaciones simultáneas. Una variable de ruido blanco es una variable cuyos datos están idénticamente distribuidos según una normal de media cero y de

desviación típica constante. Obviamente, cuanto menor sea dicha desviación típica menores serán los errores en la relación entre las variables endógenas y sus ecuaciones en el modelo y, por lo tanto, mejor será su predicción, y más fiable su uso y las conclusiones derivadas a partir de él.

El esquema de un modelo con N ecuaciones, N variables endógenas y K variables exógenas es:

$$\begin{array}{c} y_1 = \gamma_{1,1}x_1 + \ldots + \gamma_{1,K}x_K + \beta_{1,2}y_2 + \beta_{1,3}y_3 + \ldots + \beta_{1,N}y_N + u_1 \\ y_2 = \gamma_{2,1}x_1 + \ldots + \gamma_{2,K}x_K + \beta_{2,1}y_1 + \beta_{2,3}y_3 + \ldots + \beta_{2,N}y_N + u_2 \\ \ldots \\ y_N = \gamma_{N,1}x_1 + \ldots + \gamma_{N,K}x_K + \beta_{N,1}y_1 + \ldots + \beta_{N,N-1}y_{N-1} + u_N \end{array} \tag{2.9}$$

donde $x_1, x_2, \ldots, x_K$ son variables exógenas, $y_1, y_2, \ldots, y_N$ son variables endógenas, y $u_1, u_2, \ldots, u_N$ son variables de ruido blanco. Todas ellas son vectores de dimensión d, siendo d el tamaño de la muestra.

El problema a resolver es estimar $\gamma_{1,1}, \ldots, \gamma_{N,K}, \beta_{1,2}, \beta_{1,3}, \ldots, \beta_{N,N-1}$ a partir de los datos tomados en cada una de las variables.

Cada ecuación tiene una variable endógena que está despejada en la parte izquierda y que llamaremos endógena principal. Cada ecuación está diseñada para expresar su endógena principal respecto a un conjunto de variables endógenas y exógenas.

La ecuación 2.9 se dice que está en forma estructural y puede ser expresada también de forma matricial:

$$BY^T + \Gamma X^T + u^T = 0 \tag{2.10}$$

siendo:

$$\begin{array}{c} Y = (y_1 \ldots y_N) \ , \ X = (x_1 \ldots x_K) \ , \ u = (u_1 \ldots u_N) \\ B = \begin{pmatrix} \beta_{1,1} & \ldots & \beta_{1,N} \\ & \ldots & \\ \beta_{N,1} & \ldots & \beta_{N,N} \end{pmatrix}, \ \Gamma = \begin{pmatrix} \gamma_{1,1} & \ldots & \gamma_{1,K} \\ & \ldots & \\ \gamma_{N,1} & \ldots & \gamma_{N,K} \end{pmatrix} \end{array} \tag{2.11}$$

o, equivalentemente, si se despejan las endógenas principales, como:

$$Y^T = \tilde{B}Y^T + \Gamma X^T + u^T \tag{2.12}$$

siendo todas las matrices las de antes y teniendo la matriz $\tilde{B}$ ceros en la diagonal principal.

El modelo estructural (ecuacion 2.10) se puede expresar también en forma reducida como sigue:

$$Y = X\Pi + v \tag{2.13}$$

con:

$$\Pi^T = -B^{-1}\Gamma \ , \ v^T = -B^{-1}u^T \tag{2.14}$$

Los parámetros que se usarán en los algoritmos propuestos serán los siguientes:

- El tamaño de la muestra, d.
- El número de variables endógenas, N.
- El número de variables exógenas, K.
- El número de variables endógenas en la ecuación i, n_i (o lo que es lo mismo, el número de coeficientes distintos de cero en la fila i-ésima de la matriz B).
- El número de variables exógenas en la ecuación i, k_i (o lo que es lo mismo, el número de coeficientes distintos de cero en la fila i-ésima de la matriz Γ).
- La matriz de variables exógenas X, de dimensión $d \times K$.
- La matriz de variables endógenas Y, de dimensión $d \times N$.

Para aplicar MCO sin obtener estimadores sesgados es necesario que no haya correlación entre el término aleatorio o variable de ruido blanco y las variables explicativas de la ecuación. En la teoría básica de los M.E.S. se ha demostrado que si una variable endógena entra de explicativa en una ecuación cuya principal es explicativa en la ecuación donde esta endógena es endógena principal, entonces existe correlación entre el término aleatorio y ella misma, y por lo tanto no se puede aplicar MCO. Esta es la razón por la que en los M.E.S. no se aplica directamente MCO y hay que buscar otros estimadores.

2.3.2. El problema de la identificación

Antes de resolver una ecuación en un modelo de ecuaciones simultáneas hay que tener en cuenta que no todas las ecuaciones pueden ser resueltas. El problema de la identificación pretende establecer si las estimaciones numéricas de los parámetros de una ecuación estructural (ecuación 2.9) pueden ser obtenidos de los coeficientes estimados de la forma reducida (ecuación 2.13). Si puede hacerse para una ecuación, se dice que dicha ecuación está **identificada**, y en caso contrario la ecuación considerada está **subidentificada**.

Para poder calcular la matriz Π en la expresión 2.13 es necesario que el número de ecuaciones planteadas $(N \cdot d)$ sea mayor o igual que el número de incógnitas a calcular $(N \cdot K)$, que no son otras que los valores de la matriz Π, con lo que se deduce que es necesario que $K \leq d$ para que se pueda calcular la expresión reducida de un M.E.S.

Para calcular la matriz B y la matriz Γ en la expresión 2.14 se tienen NK ecuaciones, y las incógnitas a calcular serán los valores de B distintos de cero (como máximo $N(N-1)$) y los valores de Γ distintos de cero (como máximo NK). Por lo tanto, una ecuación podrá ser resuelta si el número de igualdades que aporta (K) es mayor o igual que el número de incógnitas que tiene ($k_i + n_i - 1$).

Hay tres tipos de ecuaciones: **subidentificadas** (no se pueden resolver), **sobreidentificadas** (las soluciones de los coeficientes de la forma estructural 2.10 no se obtienen de forma única de la forma reducida 2.13) y **exactamente identificadas** (solución única).

Para estudiar la identificación de una ecuación se usan dos condiciones, la **condición de orden** y la **condición de rango**. La condición de orden es una mera comprobación cuya computación es mínima pero es una condición necesaria aunque no suficiente. La condición de rango, que sí es necesaria y suficiente, tiene necesidades computacionales grandes.

Condición de Orden

Si la ecuación i-ésima cumple $n_i - 1 > K - k_i$ es una ecuación subidentificada, si cumple $n_i - 1 = K - k_i$ es una ecuación exactamente identificada, y si cumple $n_i - 1 < K - k_i$ es una ecuación sobreidentificada [38].

Condición de Rango

En un modelo que contiene N ecuaciones y N variables endógenas, una ecuación está identificada sí y solo sí puede construirse al menos una matriz no singular de tamaño $(N-1)\times(N-1)$ con los coeficientes de las variables (endógenas y exógenas) excluidas de esa ecuación particular e incluidas en otras del modelo [38].

2.3.3. Tipos de estimadores

En el estudio teórico de los M.E.S. se han desarrollado diferentes expresiones de estimadores, cada uno bajo diferentes hipótesis y con diferentes propiedades en la inferencia del parámetro buscado. También dichos estimadores tienen diferentes necesidades de computación y de memoria. Ejemplos de ellos son el estimador de Máxima Verosimilitud (MV), Mínimos Cuadrados Indirectos (MCI), Mínimos en dos Etapas (MC2E), Mínimos Cuadrados en tres Etapas (MC3E), etc. [37, 38]

Tradicionalmente se suelen agrupar estos estimadores en dos conjuntos diferenciados: métodos de información completa y métodos de información limitada. Los primeros calculan la estimación de todas las ecuaciones del sistema a la vez, es decir, operan y resuelven todas las ecuaciones al mismo tiempo. Estos métodos gozan de una gran precisión en su estimación puesto que reflejan la influencia de unas ecuaciones en otras. En contrapartida son métodos muy costosos tanto en memoria como computacionalmente, debido al tamaño de las matrices que deben manejar, y además son muy sensibles a posibles errores. Es decir, si una ecuación está mal expresada

o tiene errores en una variable, su influencia será notoria en el sistema completo sesgando gravemente las estimaciones obtenidas, incluso en ecuaciones donde no aparezca dicha variable. Estas dos desventajas hicieron que desde casi el principio quedaran en desuso a favor de los métodos de información limitada. El método más común de este tipo es MC3E.

Los métodos de información limitada, por el contrario, operan ecuación a ecuación, estimando los coeficientes en cada una de ellas, utilizando la información de la ecuación mas una información global que se ha calculado a partir del sistema. Por lo tanto, este tipo de estimador es menos preciso que los anteriores, pero menos vulnerable a errores y con menos necesidades computacionales. Estas razones llevaron en los años 60 y 70 (años donde el uso de los M.E.S. gigantes llegó a su apogeo) a utilizar básicamente estas técnicas [27], y hasta en algunos casos, y aún sabiendo el error que cometían, a usar Mínimos Cuadrados Ordinarios directamente por no poder computar estimadores más complejos. Los métodos más comunes en este grupo son MCI (que aún siendo el más simple no se puede usar siempre) y MC2E.

2.3.4. Mínimos Cuadrados Indirectos (MCI)

El estimador MCI es solo aplicable a aquellas ecuaciones que son exactamente identificadas. Esto restringe mucho su utilidad puesto que el número de ecuaciones exactamente identificadas en un modelo puede ser poco numeroso. Sin embargo, tal y como se verá en el capitulo 2, las exigencias computacionales de este estimador son menores que las de MC2E, por lo que es útil usarlo siempre que se pueda.

MCI estima los valores de Π de la ecuación 2.13 mediante MCO utilizando todas las variables endógenas de dependientes y todas las exógenas como explicativas. Una vez estimada la matriz Π se obtienen los valores de la forma estructural de la ecuación a ser resuelta. Para obtener los parámetros de la forma estructural, MCI resuelve las ecuaciones derivadas de la igualdad $-B_i\Pi = \Gamma_i$ (deducida en 2.14), donde B_i es la fila de B, y Γ_i la fila de Γ correspondiente a la ecuación que se está resolviendo, y Π es común a todas las ecuaciones.

El número de ecuaciones que surgirán de la igualdad $-B_i\Pi = \Gamma_i$ es K, que es el número de columnas de Π, y el número de incógnitas que habrá en la forma estructural, y que por lo tanto habrá en B_i y Γ_i, es $n_i + k_i - 1$, que es el número de endógenas mas el número de exógenas en la ecuación i-ésima menos la endógena principal, puesto que su coeficiente está igualado a uno. El hecho de que la ecuación sea exactamente identificada exige, según la condición de orden, que ambos valores coincidan ($n_i + k_i - 1 = K$) y por lo tanto asegura una única solución a la estimación.

2.3.5. Mínimos Cuadrados en Dos Etapas (MC2E)

El estimador MC2E, a diferencia de otros estimadores usados en Modelos de Ecuaciones Simultáneas, puede ser utilizado en cualquier ecuación del M.E.S. que esté identificada, sin importar si es exactamente identificada o sobreidentificada.

Según la condición de orden, para resolver una ecuación por MC2E se tiene que cumplir que $n_i - 1 + k_i \leq K$.

Sabemos que no es posible usar MCO directamente en un Modelo de Ecuaciones Simultáneas debido a la correlación de las variables endógenas con el término aleatorio. MC2E lo que hace es sustituir las variables endógenas que actúan como variables explicativas en todas las ecuaciones del sistema por otras variables que, pareciéndose mucho a las endógenas originales, no están correlacionadas con el termino de error. A estas variables se les denomina *proxy* (por ser muy próximas a las variables endógenas originales). Una vez hecha la sustitución, su puede usar MCO para resolver las ecuaciones.

La variable *proxy* de una endógena es calculada mediante la estimación de dicha endógena utilizando MCO, con todas las exógenas del sistema como variables explicativas. Debido a que una variable endógena puede aparecer en varias ecuaciones del M.E.S. es necesario almacenar las *proxy* calculadas para su reutilización.

A la matriz de variables *proxy* se la denota por $\hat{Y}$, y tiene las mismas dimensiones que Y $(d \times N)$. Se cálcula mediante la expresión:

$$\hat{Y} = X(X^T X)^{-1} X^T Y \tag{2.15}$$

Una vez calculadas las variables *proxy* procedemos a la resolución de las ecuaciones. Para resolver la ecuación i-ésima, la cual se puede expresar de la forma $y_i = X_i b_i + Y_i \Gamma_i + \epsilon_i$ (siendo la variable y_i la endógena principal de la ecuación), primero se sustituyen las variables endógenas que hay en dicha ecuación (salvo la principal) por las variables *proxy* calculadas (Y_i por $\hat{Y}_i$). Las matrices X_i, Y_i e $\hat{Y}_i$ son las matrices asociadas a la ecuación i-ésima que tienen por columnas los datos de las variables exógenas, endógenas y *proxy* que aparecen en dicha ecuación.

Definimos la matriz $X_{e_i} = [X_i | \hat{Y}_i]$, es decir, la matriz formada por las columnas de las variables exógenas y las *proxy* que aparecen en la ecuación.

Por lo tanto, para la estimación de $\beta_i = [b_i^T | \Gamma_i^T]$ basta con calcular la expresión $(X_{e_i}^T X_{e_i})^{-1} X_{e_i}^T y_i$ (expresión de MCO aplicada a X_{e_i}).

Como se observa, se aplican dos veces MCO para resolver cada una de las ecuaciones del sistema, una en el calculo de las variables *proxy*, y la otra en la resolución de cada una de las ecuaciones. De ahí el nombre del estimador, Mínimos Cuadrados en Dos Etapas.

2.4. Conclusiones

En este capítulo se han descrito las herramientas básicas que serán usadas a lo largo del presente trabajo. Se han descrito el hardware donde se han realizado los experimentos y las librerías software que han sido utilizadas.

También se describen, como herramientas matemáticas que serán utilizadas a lo largo del trabajo, el método de Mínimos Cuadrados Ordinarios, la descomposición

QR de una matriz y los algoritmos para obtenerla (transformación de Householder y rotaciones de Givens).

Se han descrito matemáticamente los Modelos de Ecuaciones Simultáneas desde sus expresiones más comunes (forma estructural y forma reducida), y también los tipos de ecuaciones que lo forman (problema de la identificación).

La parte final del capítulo se ha centrado en describir matemáticamente los dos estimadores de M.E.S. que serán utilizados en los algoritmos del presente trabajo, MCI y MC2E, ambos pertenecientes al grupo de estimadores llamados de información limitada.

Capítulo 3

Algoritmos secuenciales para la resolución de Modelos de Ecuaciones Simultáneas

En este capitulo se estudiará la resolución de Modelos de Ecuaciones Simultáneas por Mínimos Cuadrados Indirectos y Mínimos Cuadrados en dos Etapas. La razón por la que se desarrollan algoritmos para estos estimadores es por pertenecer al grupo de los más utilizados históricamente, y porque computacionalmente se han desarrollado estudios para los estimadores de información completa pero no para los de información limitada [48].

Se desarrollan y estudian dichos algoritmos de manera secuencial, con el objetivo de minimizar los recursos utilizados tanto en tiempo de CPU como en memoria. Además, utilizando descomposiciones matriciales se llegará a expresiones del mismo estimador menos costosas computacionalmente por aprovechar mejor los cálculos realizados en operaciones anteriores a él.

Se estudiarán diversos algoritmos para el cálculo de los estimadores de un modelo de ecuaciones simultáneas (MCI y MC2E). Como dichos estimadores están basados en Mínimos Cuadrados, se estudiarán primero éstos referenciándolos como Mínimos Cuadrados Ordinarios (MCO).

El capítulo aporta una nueva forma de calcular el estimador MC2E basado en una expresión para la inversa de una matriz. Utilizando dicha expresión se pueden reutilizar cálculos y reducir considerablemente el tiempo de ejecución. Se demuestra matemáticamente que el número de operaciones en coma flotante utilizado al resolver una ecuación usando dicha descomposición de la inversa, es siempre menor que utilizar la expresión normal del estimador. Además, en el estudio experimental mostrado en el capítulo se comprueba que se puede conseguir un coste hasta 10 veces menor.

Se ha desarrollado un módulo de Excel con los algoritmos presentados en este capítulo, mediante el cual se pueden resolver M.E.S. utilizando todas las ventajas que da una hoja de cálculo y sin necesidad de exportar los datos a otro programa.

Dicho módulo puede ser usado para fines científicos (problemas que requieran la resolución de M.E.S. de tamaño pequeño) o docentes. En particular ha sido usado por los alumnos de la asignatura de Econometría de 5º de Administración y Dirección de Empresas de la Universidad Miguel Hernández como software de prácticas.

3.1. Mínimos Cuadrados Ordinarios

A continuación se desarrolla y estudia un esquema algorítmico para MCO. El algoritmo 1 muestra un esquema para calcular MCO y, si es requerido, también a partir de él la estimación de y, $\hat{y}$. El esquema se diseña basado en la expresión de MCO dada en el capítulo 1 sin usar descomposiciones matriciales, por lo que será denotado por MCO_{bas}.

Es posible usar LAPACK y BLAS para optimizar las operaciones matriciales y hacer el programa más portable. En los experimentos realizados en la sección 3.6 se ha usado la rutina *dgemm* para la multiplicación de matrices y *dgesv* para la resolución de un sistema de ecuaciones, ambas pertenecientes a la librería LAPACK.

Algoritmo 1 Algoritmo MCO_{bas}

Entrada: $X \in \mathbb{R}^{d\times K}, Y \in \mathbb{R}^{d\times N}$
Salida: $\beta \in \mathbb{R}^{K\times N}, \hat{Y} \in \mathbb{R}^{d\times N}$
1: Hacer X^TX $\{\text{coste} \rightarrow 2K^2d\}$
2: Hacer X^TY $\{\text{coste} \rightarrow 2NKd\}$
3: Hacer $(X^TX)^{-1}(X^TY)$ resolviendo el sistema $X^TY = (X^TX)\beta$
$\{\text{coste} \rightarrow \frac{2}{3}K^3 + 2K^2N\}$
4: **Si** estimación=**Verdadero Entonces**
5: Hacer $\hat{Y} = X\beta$ $\{\text{coste} \rightarrow 2NKd\}$
6: **Fin si**

Se muestra en cada línea del algoritmo su coste computacional. Solo se tiene en cuenta el número de flops. El coste total del algoritmo sumando todas sus partes es:

$$T_{MCO_{bas}}(N, d, K, \delta_{est}) = \frac{2}{3}K^3 + 2K^2(d+N) + 2NKd + \delta_{est}2NKd \tag{3.1}$$

donde δ_{est} es 1 si *estimación* es requerida en el algoritmo, es decir, cuando se requiera la estimación de la variable dependiente Y, y 0 en otro caso.

3.2. Mínimos Cuadrados Indirectos

El estimador MCI es solo aplicable a aquellas ecuaciones que son exactamente identificadas. MCI estima los valores de Π de la ecuación 2.13 mediante MCO y a

continuación deduce los valores de la forma estructural siguiendo la relación $-B_i\Pi = \Gamma_i$ (deducida en la ecuación 2.14).

El algoritmo 2 muestra el esquema para resolver las ecuaciones exactamente identificadas de un sistema mediante MCI.

Algoritmo 2 Algoritmo MCI

Entrada: $X \in \mathbb{R}^{d\times K}, Y \in \mathbb{R}^{d\times N}, B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$
Salida: $B \in \mathbb{R}^{N\times N}, \Gamma \in \mathbb{R}^{N\times K}$
1: $\Pi^T = MCO_{bas}$(Y,X,estimación=**Falso**)
2: **Para** i=1...N **Hacer**
3: **Si** ecuación i es exactamente identificada **Entonces**
4: Resolver $-B_i\Pi = \Gamma_i$
5: **Fin si**
6: **Fin Para**

El algoritmo requiere como entradas, además de las matrices de datos X e Y, las matrices B y Γ que están formadas por ceros, unos e incognitas, que corresponden a los coeficientes de las variables en las ecuaciones. El fin del algoritmo es estimar dichas incognitas, por lo que en la salida dichas matrices contendrán los valores de los coeficientes de las variables en cada ecuación.

En el primer paso, se calcula la matriz Π, que se utiliza en la resolución de cada ecuación. Para obtenerla, se llama a MCO_{bas} utilizando la matriz de variables endógenas como matriz de variables dependientes (con dimensión $d \times N$) y la matriz de variables exógenas (de dimensión $d \times K$) como variables explicativas.

En la línea 4 se procede a la resolución de la ecuación i-ésima, que se supone exactamente identificada (por lo que $K = n_i + k_i - 1$). Se tiene que B_i, fila i-ésima de B, es de la forma $B_i = (B_{i,1}, ..., B_{i,N})$ con $B_{i,j} = 0$ si la variable endógena j-ésima no aparece en la ecuación i-ésima, $B_{i,j} = 1$ si la variable aparece en la ecuación y $j = i$ (esto indica que es la variable principal de la ecuación y por lo tanto su coeficiente vale uno), e incognita en el resto de casos. Por lo tanto, puesto que la ecuación i-ésima tiene n_i variables endógenas, B_i tendrá $n_i - 1$ incógnitas y el resto ceros.

Igualmente, la matriz Γ_i, fila i-ésima de Γ, es de la forma $\Gamma_i = (\Gamma_{i,1}, ..., \Gamma_{i,K})$ con $\Gamma_{i,j} = 0$ si la variable exógena j no aparece en la ecuación i-ésima, y uno en caso contrario. Por lo tanto, puesto que la ecuación i-ésima tiene k_i variables exógenas, Γ_i tendrá k_i incógnitas y el resto ceros.

Para resolver el sistema $-B_i\Pi = \Gamma_i$ se realiza la multiplicación simbólica $-B_i\Pi$ resultando, tras igualar a Γ_i, un sistema de ecuaciones. Ahora se toman solo las ecuaciones cuyo valor en Γ_i sea cero. El número de ecuaciones es $K - k_i$ puesto que es el número de ceros en Γ_i, y como solo están las incognitas de B_i, el número total de ellas es $n_i - 1$. Como la ecuación se supone exactamente identificada, el sistema se puede resolver, puesto que coincide el número de incógnitas con el número de ecuaciones. Una vez calculadas las incógnitas de B_i, se calculan las de Γ_i simplemente

multiplicando $-B_i\Pi$.

El coste de resolver $-B_i\Pi = \Gamma_i$ es $\frac{2}{3}(K-k_i)^3 + 2(K-k_i)^2$ para calcular B_i, y $2Nk_i$ para calcular Γ_i. De esta forma, el coste total de resolver mediante MCI un sistema donde todas las ecuaciones son exactamente identificadas es:

$$T_{MCI}(N,d,K,\Omega_k) = T_{MCO_{bas}}(N,d,K,0) + \sum_{i=1}^{N}\left(\tfrac{2}{3}(K-k_i)^3 + 2(K-k_i)^2 + 2Nk_i\right) \tag{3.2}$$

donde $\Omega_k = (k_1, ..., k_N)$ es un vector de dimensión N cuyos elementos representan el número de exógenas que hay en cada una de las ecuaciones del sistema.

Los tiempos teóricos dependen de los valores de los parámetros pero se puede deducir que $T_{MCO_{bas}}$ es de orden cúbico y la resolución de ecuaciones tarda un orden cúbico en cada una de ellas (teniendo que resolver N) por lo que el orden será elevado a la cuarta. En conclusión el coste importante de MCI está en la resolución de las ecuaciones y no en los cálculos previos. Esto se comprueba también experimentalemente en la tabla 3.2 de la sección 3.6. Destacar también que el coste de resolución de las ecuaciones, tal y como se ve en la expresión 3.2, no depende del tamaño de la muestra d. Esto da como resultado que el tiempo computacional no se incremente (salvo lo que aporte MCO_{bas}) al incrementar el tamaño de la muestra, siendo muy eficiente MCI para modelos con valores de d muy grandes.

De nuevo la computación está basada en multiplicaciones e inversas de matrices, y por lo tanto se puede usar BLAS y LAPACK (las funciones *dgemm* y *dgesv* son llamadas en MCO_{bas} y al resolver $-B_i\Pi = \Gamma_i$).

3.3. Mínimos Cuadrados en Dos Etapas básico

El estimador MC2E, tal y como se ha explicado en el capítulo 1, puede ser utilizado en cualquier ecuación del M.E.S. que esté identificada. En el caso de las exactamente identificadas su estimación coincidirá con la calculada por MCI, pudiéndose utilizar cualquiera de los dos.

El algoritmo 3 muestra el esquema del algoritmo para la resolución de un M.E.S. utilizando la versión básica de MC2E, que está basada en su expresión sin usar descomposiciones matriciales. Como se vio en el capitulo anterior, es necesario el cálculo de las variables *proxy* al comienzo del algoritmo para su uso en cada una de las ecuaciones. Para ello se hace una llamada al algoritmo MCO con estimación=**Verdadero** almacenando el resultado en $\hat{Y}$. El coste computacional del cálculo de las variables *proxy* es $T_{MCO_{bas}}(N,d,K,1)$.

En cada ecuación se define la matriz $X_{e_i} = [X_i|\hat{Y}_i]$ de dimensión $d \times (n_i + k_i - 1)$, que está formada por las columnas de las variables exógenas y las variables *proxy* que sustituyen a las variables endógeas de la ecuación (salvo la endógena principal). La matriz X_i está formada por las columnas de X en cuya posición Γ_i tiene valor no nulo (son las variables que entran en la ecuación), y la matriz $\hat{Y}_i$ por las columnas

Algoritmo 3 Algoritmo $MC2E_{bas}$

Entrada: $X \in \mathbb{R}^{d\times K}, Y \in \mathbb{R}^{d\times N}, B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$
Salida: $B \in \mathbb{R}^{N\times N}, \Gamma \in \mathbb{R}^{N\times K}$
1: $\hat{Y} = MCO_{bas}(Y,X,$estimación=**Verdadero**$)$
2: **Para** $i = 1...N$ **Hacer**
3: Hacer $X_{e_i} = [X_i|\hat{Y}_i]$ {Usando B_i y Γ_i}
4: $\beta = MCO_{bas}(y_i,X_{e_i},$estimación=**Falso**$)$
5: Sustituir los valores de β en las incógnitas de B_i y Γ_i
6: **Fin Para**

de $\hat{Y}$ en cuya posición B_i tiene valor no nulo y distinto de uno (tampoco se copia la endógena principal). Por lo tanto, la columna j-ésima de X estará en X_i si $\Gamma_{i,j} \neq 0$, y la columna j-ésima de $\hat{Y}$ estará en $\hat{Y}_i$ si $B_{i,j} \neq 0$ y $B_{i,j} \neq 1$.

A continuación se vuelve a hacer una nueva llamada a MCO_{bas} utilizando la matriz X_{e_i} y la endógena principal de la ecuación, y_i.

Los valores calculados en la línea 4 son las estimaciones de los coeficientes de la ecuación. Están representados por incógnitas en B_i y Γ_i por lo que hay que sustituirlos en dichas matrices (línea 5).

El coste del algoritmo es:

$$T_{MC2E_{bas}}(N,d,K,\Omega_n,\Omega_k) = T_{MCO_{bas}}(N,d,K,1) + \sum_{i=1}^{N}(T_{MCO_{bas}}(1,d,k_i+n_i-1,0)) \tag{3.3}$$

donde $\Omega_n = (n_1,...,n_N)$, y $\Omega_k = (k_1,...,k_N)$ son dos vectores de dimensión N cuyos elementos son, respectivamente, el número de endógenas y exógenas que hay en cada una de las ecuaciones del sistema.

La ecuacion 3.3 muestra que el coste total de $MC2E_{bas}$ depende fuertemente del valor de d. De hecho tiene mayor influencia en $MC2E_{bas}$ que en MCI (ecuación 3.2). En los resultados mostrados en la tabla 3.3 de la sección 3.6 se puede observar la relación entre d y el coste total de $MC2E_{bas}$. También se puede observar en los resultados experimentales (tablas 3.3 y 3.5) la gran influencia que los parámetros N, K y d tienen en el coste del algoritmo $MC2E_{bas}$.

El principal problema del algoritmo descrito reside en el alto coste de las operaciones (dado el gran número de llamadas a MCO_{bas}) tanto en tiempo como en memoria, así como en la falta de precisión en los resultados, cosecuencia de la gran cantidad de operaciones. Estos dos problemas hacen aconsejable la utilización de técnicas de descomposición matricial que permitan reducir el número de operaciones y el coste computacional de los cálculos y aumentar la consistencia de los datos.

3.4. Mínimos Cuadrados en dos Etapas basado en la descomposición de la inversa

Se ha presentado una versión básica de MC2E ajustada a la expresión del estimador, pero que no es eficiente por tener un gasto computacional importante. Esto sumado a que MC2E es un algoritmo aplicable tanto a ecuaciones sobreidentificadas como a ecuaciones exactamente identificadas, y a que las primeras suelen ser más numerosas que las segundas, hace necesario mejorar dicho algoritmo. Para ello se van a utilizar descomposiciones matriciales, las cuales permitirán reducir el número de operaciones. Sin embargo, en determinadas situaciones no será posible utilizar dichas descomposiciones matriciales. En estos casos, que aún siendo muy poco comunes podrían darse, será necesario utilizar $MC2E_{bas}$.

Se definen a continuación dos objetivos a considerar para obtener un algoritmo de resolución de M.E.S. eficiente:

- Los algoritmos basados en MCI y MC2E deben compartir información. Puesto que se va a utilizar MCI en las ecuaciones exactamente identificadas, ya que se ha visto que tiene un coste teórico menor que $MC2E_{bas}$, es recomendable que compartan el máximo de información posible con $MC2E_{bas}$ y reducir así el número de operaciones.

- Puesto que se van a resolver muchas ecuaciones con MC2E, es recomendable que se comparta el máximo de información entre las distintas operaciones que se realicen al resolver cada una de ellas.

También es recomendable, puesto que se pretende realizar una futura implementación en paralelo, estructurar el algoritmo de forma que se facilite dicha implementación. Así, la realización de cálculos al comienzo permitirá una implementación paralela más eficiente, puesto que se realizará el máximo número de operaciones entre todos los procesadores al principio del algoritmo para poder ser usadas por cualquier procesador en cualquier otra parte del algoritmo.

El primer objetivo se consigue de manera satisfactoria con los algoritmos MCI y $MC2E_{bas}$, pero es posible obtener mejores resultados en el segundo usando alguna descomposición en las matrices usadas en el cálculo de MC2E. Se presenta a continuación un nuevo algoritmo basado en el estimador de MC2E utilizando una descomposición de la inversa de una matriz. Esta nueva versión será denotada por $MC2E_{inv}$.

En $MC2E_{inv}$ se usa una descomposición en cuatro partes de la inversa de una matriz, y será utilizada en el cálculo de X^TX. Mediante esta descomposición se reutilizarán operaciones realizadas previamente, con lo que el coste computacional total es menor.

Las siguientes operaciones son hechas al comienzo del algoritmo:

- Π se calcula como $\Pi^T = (X^TX)^{-1}X^TY$ (usando $MCO_{bas}(Y,X,$**Falso**$)$), y las matrices X^TX y X^TY se guardan para su uso posterior.

- Se calcula la matriz de variables *proxy*, $\hat{Y} = X\Pi^T$.
- Se calcula la matriz $\hat{Y}^T\hat{Y}$.

Utilizando MCO_{bas}(Y,X,**Verdadero**) y guardando Π, se consiguen los dos primeros objetivos.

Ahora, cuando MCI es utilizado para resolver una ecuación, la matriz Π está disponible, y cuando MC2E es utilizado, todas las variables *proxys* han sido calculadas y están disponibles.

Una vez realizadas las operaciones descritas al comienzo del algoritmo se resuelve cada una de las ecuaciones. Para ello hay que volver a realizar MCO_{bas} sobre las variables *proxy* y exógenas que toquen en cada momento, pero además de utilizar las variables *proxy* no se aprovecha ninguna información de la computada al comienzo.

La descomposición matricial que se va a usar está basada en el siguiente teorema:

Teorema 2.1

$$\begin{pmatrix} A & B \\ B^T & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -E \\ Id \end{pmatrix} (D - B^T A^{-1} B)^{-1} (-E^T | Id) \tag{3.4}$$

donde $E = A^{-1}B$

Demostración. La demostración del teorema se puede encontrar en el apéndice A de [61]

□

Supongamos que se va a resolver la ecuación i-ésima usando MC2E. Como todas las variables *proxy* han sido calculadas, el primer paso es sustituir las variables endógenas que actúan de explicativas en la ecuación por sus variables *proxy*:

$$y_i = \gamma_0 + \gamma_1 x_{i_1} + \ldots + \gamma_{k_i} x_{i_{k_i}} + \alpha_1 \hat{y}_{i_1} + \ldots + \alpha_{n_i} \hat{y}_{i_{n_i}} + \epsilon \tag{3.5}$$

El conjunto de variables *proxy* que aparecen en la ecuación será denotado por $\hat{Y}_i$ y el conjunto de variables exógenas por X_i. Las matriz usada al aplicar MCO en la ecuación es $[X_i|\hat{Y}_i]$ (con dimensión $d \times (n_i + k_i - 1)$), calculándose la expresión $([X_i|\hat{Y}_i]^T[X_i|\hat{Y}_i])^{-1}[X_i|\hat{Y}_i]^T y_i$.

Usando el teorema previo, la inversa de la matriz puede expresarse como sigue:

$$\left(\begin{bmatrix} X_i^T \\ \hat{Y}_i^T \end{bmatrix} \begin{bmatrix} X_i | \hat{Y}_i \end{bmatrix} \right)^{-1} = \begin{pmatrix} X_i^T X_i & X_i^T \hat{Y}_i \\ \hat{Y}_i^T X_i & \hat{Y}_i^T \hat{Y}_i \end{pmatrix}^{-1} = \begin{pmatrix} (X_i^T X_i)^{-1} & 0 \\ 0 & 0 \end{pmatrix} +$$
$$\begin{pmatrix} -(X_i^T X_i)^{-1} X_i^T \hat{Y}_i \\ Id \end{pmatrix} (\hat{Y}_i^T \hat{Y}_i - Y_i^T X_i (X_i^T X_i)^{-1} X_i^T \hat{Y}_i)^{-1} (-Y_i^T X_i (X_i^T X_i)^{-1} | Id_{n_i - 1}) \tag{3.6}$$

Esta nueva expresión de la inversa tiene submatrices que pueden ser tomadas de cálculos realizados al comienzo del algoritmo. Se analiza a continuación cuales de éstas son tomadas de cálculos previos y cuales hay que calcular (y su coste):

- $X_i^T X_i$ se toma de $X^T X$, que se obtiene al calcular Π.
- $X_i^T \hat{Y}_i$ se toma de $X^T Y$, que se obtiene al calcular Π. Esto es debido a que $X^T \hat{Y} = X^T X \Pi = X^T X (X^T X)^{-1} X^T Y = X^T Y$.
- $(X_i^T X_i)^{-1}[X_i^T \hat{Y}_i | Id_{k_i}]$ se tiene que calcular (coste de resolver el sistema $\frac{2}{3}k_i^3 + 2k_i^2(n_i - 1 + k_i) = \frac{8}{3}k_i^3 + 2k_i^2(n_i - 1)$). Con esta operación se consigue obtener por un lado la matriz resultante de multiplicar la inversa de $(X_i^T X_i)$ por $X_i^T \hat{Y}_i$, y por otro lado la inversa de $(X_i^T X_i)$, la cual es necesaria en el cálculo de $\begin{pmatrix} (X_i^T X_i)^{-1} & 0 \\ 0 & 0 \end{pmatrix}$.
- $\hat{Y}_i^T X_i (X_i^T X_i)^{-1} X_i^T \hat{Y}_i$ se tiene que calcular (coste de la multiplicación $2(n_i - 1)^2 k_i$).
- $\hat{Y}_i^T \hat{Y}_i$ se toma de $\hat{Y}^T \hat{Y}$, que es calculado antes.
- $\hat{Y}_i^T \hat{Y}_i - \hat{Y}_i^T X_i (X_i^T X_i) X_i^T \hat{Y}_i$ se tiene que calcular (coste de restar las dos matrices $(n_i - 1)^2$)
- $(\hat{Y}_i^T \hat{Y}_i - \hat{Y}_i^T X_i (X_i^T X_i)^{-1} X_i^T \hat{Y}_i)^{-1}(-\hat{Y}_i^T X_i (X_i^T X_i)^{-1} | Id_{n_i - 1})$ se tiene que calcular (coste de resolver el sistema $\frac{2}{3}(n_i - 1)^3 + 2(n_i - 1)^2(n_i - 1 + k_i) = \frac{8}{3}(n_i - 1)^3 + 2(n_i - 1)^2 k_i$).
- Multiplicar $\begin{pmatrix} -(X_i^T X_i)^{-1} X_i^T \hat{Y}_i \\ Id_{n_i - 1} \end{pmatrix}$ por la matriz resultante del paso anterior (coste de la multiplicación $2(n_i - 1)^2 k_i$).
- Sumar la matriz resultante del paso anterior a $\begin{pmatrix} (X_i^T X_i)^{-1} & 0 \\ 0 & 0 \end{pmatrix}$ (coste de la suma k_i^2).

El coste total es $\frac{8}{3}(n_i - 1)^3 + \frac{8}{3}k_i^3 + 6(n_i - 1)^2 k_i + 2k_i^2(n_i - 1) + (n_i - 1)^2 + k_i^2$.

Por lo tanto se tiene calculada la inversa de la matriz en la expresión de MCO para la ecuación i-esima $(([X_i|\hat{Y}_i]^T [X_i|\hat{Y}_i])^{-1}[X_i|\hat{Y}_i]^T y_i)$. Siguiendo con la expresión, la matriz $[X_i|\hat{Y}_i]^T y_i$ tiene que ser también calculada. $X_i^T y_i$ se toma de $X^T Y$, que se obtiene al calcular Π. $\hat{Y}_i^T y_i$ se toma de $\hat{Y}^T \hat{Y}$ puesto que $\hat{Y}^T \hat{Y} = \hat{Y}^T Y$, como se demuestra a continuación:

$$\hat{Y}^T Y = Y^T X (X^T X)^{-1} X^T Y = Y^T X (X^T X)^{-1} ((X^T X)(X^T X)^{-1}) X^T Y = (Y^T X (X^T X)^{-1} X^T)(X (X^T X)^{-1} X^T Y) = \hat{Y}^T \hat{Y}$$

Finalmente, se calcula la multiplicación de ambas matrices completando la expresión de MCO (coste $2(n_i - 1 + k_i)^2$).

En el algoritmo 4 se muestra el equema de MCO utilizado la descomposición de la inversa dada en la ecuación 3.6.

Algoritmo 4 Algoritmo MCO_{inv}

Entrada: $X^TX \in \mathbb{R}^{K\times K}, X^TY \in \mathbb{R}^{K\times N}, Y^TY \in \mathbb{R}^{N\times N}$, $B_i \in \mathbb{R}^N$ y $\Gamma_i \in \mathbb{R}^K$
Salida: la solución dada por mínimos cuadrados

1: Hacer $\begin{pmatrix} X_i^TX_i & X_i^T\hat{Y}_i \\ \hat{Y}_i^TX_i & \hat{Y}_i^T\hat{Y}_i \end{pmatrix}^{-1}$ {usando la ecuación 3.6}
2: Hacer $[X_i|\hat{Y}_i]^Ty_i$
3: Hacer $([X_i|\hat{Y}_i]^T[X_i|\hat{Y}_i])^{-1}[X_i|\hat{Y}_i]^Ty_i$

El coste de MCO_{inv} al resolver la ecuación i-ésima, es:

$$T_{MCO_{inv}}(n_i, k_i) = \frac{8}{3}(n_i-1)^3 + \frac{8}{3}k_i^3 + 2(n_i-1+k_i)^2 + 6(n_i-1)^2k_i + 2k_i^2(n_i-1) + (n_i-1)^2 + k_i^2 \tag{3.7}$$

Como se observa, el coste de usar MCO_{inv} al resolver una ecuación tiene orden cúbico y depende del número de variables endógenas y exógenas que tiene la ecuación. Sin embargo dicho coste no depende del tamaño de la muestra d, y por lo tanto, tal y como le ocurría al usar MCI, este algoritmo es más eficiente que MCO_{bas} para ecuaciones con tamaños de muestra grandes.

Este coste puede ser comparado con el coste total de resolver una ecuación en $MC2E_{bas}$, y es posible deducir que el tiempo de resolver una ecuación en $MC2E_{bas}$ es mayor que al hacerlo en $MC2E_{inv}$, tal como muestra la siguiente proposición.

Proposición
El número de flops realizados al resolver una ecuación de un M.E.S. por MCO_{bas} es mayor o igual que el número de flops realizados al resolver la misma ecuación por MCO_{inv}.

Demostración. Supongamos que se va a resolver una ecuación con $m_i = n_i - 1$ variables endógenas, k_i exógenas y un tamaño de muestra d. El coste en la versión básica de MCO es:

$$T_{MCO_{bas}}(1, d, m_i + k_i, 0) = \frac{2}{3}(m_i+k_i)^3 + 2(m_i+k_i)^2d + 2(m_i+k_i)^2 + 2(m_i+k_i)d,$$

y el coste de hacerlo por MCO_{inv} es:

$$T_{MCO_{inv}}(m_i+1, k_i) = \frac{8}{3}(m_i^3+k_i^3) + 2(m_i+k_i)^2 + 6m_i^2k_i + 2k_i^2m_i + m_i^2 + k_i^2 = \frac{2}{3}m_i^3 + \frac{2}{3}k_i^3 + 2(m_i^3+k_i^3) + 2(m_i+k_i)^2 + 6m_i^2k_i + 2k_i^2m_i + m_i^2 + k_i^2.$$

Tenemos que

$$\frac{2}{3}(m_i+k_i)^3 = \frac{2}{3}(m_i^3+k_i^3+3m_i^2k_i+3k_i^2m_i) \geq \frac{2}{3}(m_i^3+k_i^3) + m_i^2 + k_i^2 \ (*)$$

por lo que es suficiente con demostrar que:

$$2(m_i^3+k_i^3) + 6m_i^2k_i + 2k_i^2m_i < 2(m_i+k_i)^2d + 2(m_i+k_i)d.$$

Para calcular Π, $(X^TX)^{-1}$ debe ser calculado, y es necesario que $K \leq d$ (X tiene dimension $d \times K$), y $0 < m_i + k_i \leq K \leq d$ debido a que la ecuación está identificada.

Usando la inecuación previa, se deduce que:

$$(m_i + k_i)^3 < d(m_i + k_i)(m_i + k_i + 1) \text{ y } m_i^3 + k_i^3 + 3m_i^2k_i + k_i^2m_i \leq (m_i + k_i)^3$$

Entonces $m_i^3 + k_i^3 + 3m_i^2k_i + k_i^2m_i < d(m_i + k_i)(m_i + k_i + 1)$. Se tiene, por tanto, que:

$$2(m_i^3 + k_i^3) + 6m_i^2k_i + 2k_i^2m_i < 2(m_i + k_i)^2d + 2(m_i + k_i)d \text{ (**)}.$$

Sumando ambos miembros de las desigualdades (*) y (**), y $2(m_i + k_i)^2$ a cada lado de la desigualdad, da como resultado que

$$T_{MCO_{inv}} = \tfrac{2}{3}(m_i^3 + k_i^3) + 2(m_i^3 + k_i^3) + 2(m_i + k_i)^2 + 6m_i^2k_i + 2k_i^2m_i + m_i^2 + k_i^2 \leq \tfrac{2}{3}(m_i + k_i)^3 + 2(m_i + k_i)^2d + 2(m_i + k_i)^2 + 2(m_i + k_i)d = T_{MCO_{bas}}.$$

□

El algoritmo 5 muestra el esquema general de $MC2E_{inv}$. En $MC2E_{inv}$ hay algunas operaciones extra al comienzo del algoritmo que no es necesario hacer en $MC2E_{bas}$ (debe hacerse $\hat{Y}^T\hat{Y}$ en la línea 2 del algoritmo 5), teniendo esta operación extra un coste de $2N^2d$. Los resultados experimentales en 3.6 muestran que $MC2E_{inv}$ tiene un coste computacional menor que $MC2E_{bas}$ (tabla 3.5).

Algoritmo 5 Algoritmo $MC2E_{inv}$

Entrada: $X \in \mathbb{R}^{d\times K}, Y \in \mathbb{R}^{d\times N}, B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$
Salida: $B \in \mathbb{R}^{N\times N}, \Gamma \in \mathbb{R}^{N\times K}$
1: $\hat{Y} = MCO_{bas}(Y,X,$**Verdadero**$)$ {guardando Π, X^TX y X^TY}
2: Hacer $\hat{Y}^T\hat{Y}$
3: **Para** j=1...N **Hacer**
4: $\beta = MCO_{inv}(y_j,\hat{Y},X,X^TX,X^TY,\hat{Y}^T\hat{Y})$
5: Sustituir los valores de β en las incógnitas de B_i y Γ_i
6: **Fin Para**

Cuando $k_i > 0$ y $n_i > 1$, la ecuación puede ser resuelta por MCO_{inv}. Cuando $k_i = 0$ o $n_i = 1$, lo cual se da en muy raras ocasiones, no se puede aplicar esta segunda versión debido a que no se puede construir la matriz X_i o la matriz Y_i, por lo que no es posible la descomposición 3.6, por lo que hay que volver a usar la primera versión, o más concretamente aplicar MCO_{bas} a dicha ecuación.

El coste total de $MC2E_{inv}$ es:

$$T_{MC2E_{inv}}(N,d,K) = T_{MCO_{bas}}(N,d,K,1) + 2N^2d + \sum_{i=1}^{N}\left(\tfrac{8}{3}(k_i^3 + (n_i-1)^3) + 2(n_i - 1 + k_i)^2 + 6(n_i-1)^2k_i + 2k_i^2(n_i-1) + (n_i-1)^2 + k_i^2\right) \quad (3.8)$$

En la expresión 3.8 se observa que los costes previos al sumatorio, costes que corresponden a las operaciones previas a la resolución de ecuaciones en el M.E.S., tienen un orden cúbico, mientras que la resolución de las ecuaciones tiene un orden elevado a la cuarta (cada sumando dentro del sumatorio tiene orden cúbico, y se repite N veces). Esto indica que el mayor coste computacional del algoritmo se encuentra, al igual que los costes descritos para otros algoritmos en secciones anteriores, en la resolución de ecuaciones y no en los cálculos previos.

En la sección 3.6 (tabla 3.4) se muestra una comparación entre el coste de los cálculos preliminares del algoritmo (antes de comenzar a resolver ecuaciones) y el total.

3.5. Algoritmo para la resolución de Modelos de Ecuaciones Simultáneas

En el algoritmo 6 se expone un esquema de resolución de M.E.S. En un modelo general habrá ecuaciones tanto identificadas como no identificadas. Las no identificadas no se resolverán y para las otras se usará MCI o $MC2E_{inv}$ dependiendo de si la ecuación es exactamente identificada o sobreidentificada. En los casos excepcionales en que no se pueda usar $MC2E_{inv}$ (cuando el número de variables exógenas sea cero o cuando la ecuación no tenga variables endógenas salvo la principal), se usará $MC2E_{bas}$ en su lugar.

Algoritmo 6 Algoritmo MES

Entrada: $X \in \mathbb{R}^{d\times K}, Y \in \mathbb{R}^{d\times N}, B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$
Salida: $B \in \mathbb{R}^{N\times N}, \Gamma \in \mathbb{R}^{N\times K}$

$\hat{Y} = MCO_{bas}(Y, X,$**Verdadero**$)$ {guardando Π, X^TX y X^TY}
Hacer $\hat{Y}^T\hat{Y}$
Para i=1...N **Hacer**
  Si la ecuación i es exactamente identificada **Entonces**
    Resolver $-B_i\Pi = \Gamma_i$ {Se utiliza MCI}
  Si no, **Si** la ecuación i es sobreidentificada **Entonces**
    Si $k_i > 0$ y $n_i > 1$ **Entonces**
      $\beta = MCO_{inv}(y_i,\hat{Y},X,X^TX,X^TY,\hat{Y}^T\hat{Y})$ {Se utiliza $MC2E_{inv}$}
    Si no
      $\beta = MCO_{bas}(y_i, X_e,$**Falso**$)$ {Se utiliza $MC2E_{bas}$}
    Fin si
    Sustituir los valores de β en las incógnitas de B_i y Γ_i
  Si no
    no hacer nada {la ecuación i no está identificada}
  Fin si
Fin Para

En la línea 7 se comprueba si el número de variables exógenas y el número de variables endógenas aparte de la principal en la ecuación es mayor que cero. Si es así se pueden construir las submatrices usadas en $MC2E_{inv}$, si no es así hay que usar $MC2E_{bas}$.

En la tabla 3.1 se muestra un resumen del coste teórico de cada uno de los algoritmos de resolución de Modelos de Ecuaciones Simultáneas expuestos en las secciones anteriores. Para las aproximaciones se ha tomado el promedio de variables exógenas ($\bar{k}$) y endógenas ($\bar{n}$) que aparece en cada ecuación.

Algoritmo	coste
MCI	$T_{MCI}(N,d,K) = T_{MCO_{bas}}(N,d,K,\delta_{est}=0)+$ $+\sum_{i=1}^{N}\left(\frac{2}{3}(K-k_i)^3+2(K-k_i)^2+2Nk_i\right)$ $\approx N(K-\bar{k})^3$
$MC2E_{bas}$	$T_{MC2E_{bas}}(N,d,K) = T_{MCO_{bas}}(N,d,K,\delta_{est}=1)+$ $+\sum_{i=1}^{N}\left(T_{MCO_{bas}}(1,d,k_i+n_i-1,\delta_{est}=0)\right)$ $\approx N(\frac{2}{3}(\bar{k}+\bar{n})^3+2(\bar{k}+\bar{n})^2 d)$
$MC2E_{inv}$	$T_{MC2E_{inv}}(N,d,K) = T_{MCO_{bas}}(N,d,K,\delta_{est}=1)+2N^2d+$ $\sum_{i=1}^{N}\left(\frac{8}{3}(k_i^3+(n_i-1)^3)+2(n_i-1+k_i)^2+\right.$ $\left.6(n_i-1)^2k_i+2k_i^2(n_i-1)+(n_i-1)^2+k_i^2\right)$ $\approx N(\frac{8}{3}(\bar{k}^3+\bar{n}^3)+6\bar{n}^2\bar{k}+2\bar{k}^2\bar{n})$

Tabla 3.1: Resumen de los tiempos teóricos de los algoritmos MCI, $MC2E_{bas}$ y $MC2E_{inv}$

Se puede ver en la tabla que los costes se aproximan en todos los casos al coste más alto de dentro del sumatorio multiplicado por N. Esto, como se ha dicho, indica que el coste de resolver las ecuaciones es mayor que el coste de las operaciones previas en todos los algoritmos, ya que los dos tienen órdenes cúbicos, pero el primero va multiplicado por el número de ecuaciones. Esto también es corroborado experimentalmente en la sección 3.6, tablas 3.2, 3.3 y 3.4 donde se ve que el coste de las operaciones previas es despreciable frente al coste de resolver las ecuaciones.

En ecuaciones exactamente identificadas donde $n_i + k_i - 1 = K$, el coste de $MC2E_{inv}$ es proporcional a NK^3, mientras que el de MCI es proporcional a $N(K-\bar{k})^3$. Como se observa, la diferencia puede ser considerable sobre todo en sistemas donde la mayoría de las ecuaciones lleven muchas variables exógenas.

También se puede observar que en las aproximaciones de los tiempos de MCI y $MC2E_{inv}$ se ha eliminado el término correspondiente al tamaño de la muestra (d), mientras que no ocurre lo mismo en $MC2E_{bas}$. Teniendo en cuenta que $d \geq K$ y, casi siempre también mayor que N, la mejora en coste es considerable.

3.6. Estudio experimental

En esta sección se van a describir diversos experimentos mediante los cuales se estudian los algoritmos propuestos anteriormente. Se pretende estudiar los costes de las diferentes partes de dichos algoritmos, así como realizar comparaciones entre ellos, variando el tamaño del problema. Para ello es necesario aplicar los esquemas propuestos a Modelos de Ecuaciones Simultáneas creados previamente, por lo que se ha tenido que desarrollar un generador de M.E.S. Los modelos generados en este tema, han sido creados aleatoriamente sin ningún patrón a seguir en las ecuaciones, tan solo se ha exigido que no haya ninguna ecuación no identificada. En el caso del estudio de MCI, se ha exigido también a los modelos generados que todas las ecuaciones sean exactamente identificadas. El procedimiento de creación ha sido el siguiente: En cada ecuación, se genera aleatoriamente $n_i \leq N$ y $k_i \leq K$ tal que $K \geq n_i - 1 + k_i$. Al terminar todas las ecuaciones se comprueba la condición de rango en cada una de ellas y si existe alguna no identificada se vuelve a generar otra vez. Una vez acabado el M.E.S. se generan los datos de las variables aleatoriamente.

Para simplificar los experimentos, se han generado los modelos con valor de K un 40 % del valor de N y se ha variado N y d comparando los tiempos resultantes. En esta sección se han tomado todas las medidas en las máquina Kefren (ver sección 2.1.1).

Se pretenden estudiar experimentalmente los siguientes puntos:

- La influencia del parámetro d en los diferentes algoritmos.
- El peso en el tiempo de ejecución que tienen los cálculos previos a la resolución de ecuaciones.
- La reducción del tiempo de ejecución obtenido al usar $MC2E_{inv}$ en lugar de $MC2E_{bas}$.
- La reducción del tiempo de ejecución obtenido al usar MCI en lugar de $MC2E_{inv}$ en las ecuaciones exactamente identificadas.
- La diferencia entre las soluciones encontradas por $MC2E_{bas}$ y $MC2E_{inv}$ para detectar inestabilidad numérica en los cálculos.
- La precisión en la estimación del tiempo de ejecución utilizando las expresiones del tiempo teórico.

La tabla 3.2 muestra los tiempos de ejecución del algoritmo MCI (algoritmo 2) cuando varía el valor de d para $N = 1000$ y $K = 400$ (se observa el mismo comportamiento para otros valores de N y K). Se compara el tiempo total con el de los cálculos previos a la resolución de las ecuaciones (cálculo de la matriz Π). Tal y como se demuestra teóricamente (los costes teóricos de MCI dados en la expresión 3.2 y en la tabla 3.1), el tiempo del algoritmo MCI se ve poco afectado

por incrementos en el valor de d debido a que d no afecta en la resolución de las ecuaciones. Sin embargo, sí que es importante el valor de d en los cálculos previos, por lo que se hace necesario comprobar su influencia. Como se ve en la tabla 3.2 es cierto que el coste de Π sí se ve influenciado pero, puesto que dicho coste no supera (salvo que d sea excesivamente grande con respecto a N y K) el 1 % del coste total, el algoritmo no se ve influenciado por el incremento de d.

d:	500	1000	1500	2000
Tiempo MCI	229.63	230.44	228.54	231.26
Cal. de Π	0.96	1.42	1.95	2.48
Porcentaje	0.42 %	0.62 %	0.85 %	1.07 %

Tabla 3.2: Tiempo de ejecución (en segundos) del algoritmo MCI (algoritmo 2) en Kefren, con N=1000 y K=400, cuando varía el tamaño muestral (d).

La tabla 3.3 muestra los tiempos de ejecución del algoritmo $MC2E_{bas}$ (algoritmo 3) cuando varía el valor de d. Se compara el tiempo total con el de los cálculos previos a la resolución de las ecuaciones (cálculo de la matriz de variables *proxy*, $\hat{Y}$). La tabla 3.3 muestra el coste de ejecución de resolver un sistema por $MC2E_{bas}$ (algoritmo 3). El coste total, tal y como se había dicho en el análisis teórico, depende fuertemente de d, al contrario que ocurría con MCI. El incremento del coste en el cálculo de la matriz $\hat{Y}$ (línea 1 del algoritmo 3) es de menor grado que el incremento total, por lo que, cuando d aumenta se hace despreciable el tiempo empleado en su cálculo.

d:	500	1000	1500	2000
Tiempo $MC2E_{bas}$	790.26	1754.95	4356.69	11601.32
Calc. de $\hat{Y}$	1.14	1.73	2.51	3.12
Porcentaje	0.14 %	0.10 %	0.06 %	0.03 %

Tabla 3.3: Tiempo de ejecución (en segundos) del $MC2E_{bas}$ (algoritmo 3) en Kefren, con N=1000 y K=400, cuando varía el tamaño muestral (d).

La tabla 3.4 muestra los tiempos de ejecución del algoritmo $MC2E_{inv}$ (algoritmo 5) cuando varía el valor de d. Se compara el tiempo total del algoritmo con el de los cálculos previos a la resolución de las ecuaciones (cálculo de la matriz de variables *proxy* $\hat{Y}$ y de la matriz $\hat{Y}^T\hat{Y}$). Como se observa en la tabla 3.4, el tiempo de ejecución del algoritmo $MC2E_{inv}$ depende muy ligeramente de d (tal y como se explicó teóricamente en la expresión 3.8 y en la tabla 3.1). Los cálculos previos a la resolución de ecuaciones en $MC2E_{inv}$ (líneas 1 y 2 del algoritmo 5) tienen un tiempo mayor con respecto al total del algoritmo que los estudiados en las tablas anteriores. Sin embargo no pasan del 3 % (y esto en los casos donde el valor de d es muy grande en comparación con N y K), y aunque dichos tiempos se ven muy influenciados por

d, no tienen una gran influencia en el tiempo total. Otra conclusión que se extrae de la tabla 3.4 es la gran reducción en el coste total frente a la tabla 3.3. Tal y como se demostró teóricamente (a cambio de hacer algunos cálculos previos ($\hat{Y}^T\hat{Y}$) se consigue reducir muchísimo el coste de resolución de ecuaciones). Las mismas conclusiones se pueden extraer observando la tabla 3.5 donde dicha reducción se observa variando además N (y por lo tanto K) y d.

d:	500	1000	1500	2000
Tiempo $MC2E_{inv}$	197.43	202.78	203.35	208.08
Calc. de $\hat{Y}$ y $\hat{Y}^T\hat{Y}$	2.05	3.40	4.99	6.29
Porcentaje	1.04 %	1.68 %	2.45 %	3.02 %

Tabla 3.4: Tiempo de ejecución (en segundos) del algoritmo $MC2E_{inv}$ (algoritmo 10) en Kefren, con N=1000 y K=400, cuando varía el tamaño muestral (d).

La tabla 3.5 muestra los tiempos de ejecución de $MC2E_{bas}$ y $MC2E_{inv}$ para la resolución de un M.E.S., así como el ratio entre ellos, cuando se varía el número de variables endógenas (y por lo tanto el de exógenas), y el tamaño de la muestra. Tal y como se observa en la tabla 3.5, el coste del algoritmo $MC2E_{inv}$ siempre es menor que el coste del algoritmo $MC2E_{bas}$. Para tamaños del problema grandes puede ser hasta 10 veces menor. Esta diferencia es muy significativa sobre todo cuando el valor de d crece con respecto a N y K.

N:	500	1000	1500	2000	2500
d:	500	500	1000	1000	1500
$MC2E_{bas}$	72.82	790.93	7031.96	19337.92	97874.21
$MC2E_{inv}$	12.91	198.16	1225.33	4192.10	10217.02
Ratio	5.64	3.99	5.74	4.61	9.58

Tabla 3.5: Tiempo de ejecución (en segundos) y ratio entre el algoritmo $MC2E_{bas}$ (algoritmo 3) y $MC2E_{inv}$ (algoritmo 4) en Kefren, cuando varía el número de variables endógenas (N) y el tamaño muestral (d).

La tabla 3.6 muestra los tiempos de ejecución del algoritmo MCI (algoritmo 2) y el algoritmo $MC2E_{inv}$ (algoritmo 4) aplicados ambos a sistemas con todas las ecuaciones exactamente identificadas, cuando se varía el tamaño del problema. Tal y como se observa en la tabla, y como ya se anunció en la tabla 3.1, MCI sigue siendo más rentable en las ecuaciones exactamente identifiadas que la segunda versión de MC2E. Cuando el tamaño del problema crece, el tiempo de ejecución del primero es casi la mitad que el del segundo. También se puede observar que los tiempos de $MC2E_{inv}$ en la tabla 3.6 son superiores a los dados para el mismo algoritmo y tamaños del problema en la tabla 3.5. Esto es debido, tal y como se

vio en la expresión 3.7, a que el coste computacional del algoritmo $MC2E_{inv}$ es proporcional en cada ecuación a $n_i + k_i - 1$, que siempre es menor que K. En las ecuaciones exactamente identificadas ocurre que $n_i + k_i - 1 = K$, por lo que el coste por ecuación es mayor que en las ecuaciones sobreidentificadas.

N:	500	1000	1500	2000	2500
d:	500	500	1000	1000	1500
$MC2E_{inv}$	22.76	289.12	1665.99	6165.98	14316.93
MCI	18.17	230.78	1073.21	3118.85	7181.54
Ratio	1.25	1.25	1.55	1.98	1.99

Tabla 3.6: Tiempo de ejecución (en segundos) y ratio comparativo entre el algoritmo MCI (algoritmo 2) y el algoritmo $MC2E_{inv}$ (algoritmo 4) en Kefren, cuando varía el número de variables endógenas (N) y el tamaño muestral (d).

Para comparar la precisión numérica de las dos versiones de MC2E, en la tabla 3.7 se muestra el promedio de las diferencias, en valor absoluto, de las soluciones encontradas por $MC2E_{bas}$ para un M.E.S. de tamaño dado, y las encontradas por $MC2E_{inv}$ para el mismo M.E.S. Se puede observar en la tabla que los casos con mayor diferencia tienen siete decimales en común, teniendo la mayoría 11 decimales en común. Esto indica que desde el punto de vista de la precisión numérica no hay diferencias significativas entre resolver un sistema por $MC2E_{bas}$ o hacerlo por $MC2E_{inv}$. El interés de este estudio radica en que la gran cantidad de operaciones realizadas en $MC2E_{inv}$ pueden hacer sospechar que existe cierta inestabilidad numérica. Sin embargo las diferencias con $MC2E_{bas}$ son pequeñas. También hay que tener en cuenta el campo científico en el que se utilizan los Modelos de Ecuaciones Simultáneas puesto que, en la gran mayoría de los casos, el error cometido en la toma de la muestra (al no cumplir con exactitud el muestreo programado por el estadístico) es mayor que el que se pudiera obtener al perder precisión numérica en los cálculos realizados.

En el capítulo 5 se ha realizado la misma comparación entre $MC2E_{inv}$ y el algoritmo basado en MC2E desarrollado en dicho capítulo que utiliza la descomposición QR.

Para comprobar la validez del modelo teórico del tiempo de ejecución, la tabla 3.8 muestra los tiempos correspondientes a los algoritmos MCI y $MC2E_{inv}$, y las estimaciones dadas por las expresiones mostradas en la tabla 3.1, para distintos valores de N (y por lo tanto K) y d. También muestra la diferencia entre el tiempo real y el estimado y el porcentaje que representa dicha diferencia respecto al tiempo real. Se pretende contrastar la validez de las expresiones dadas en la tabla 3.1 para la estimación de los tiempos de ejecución de los algoritmos estudiados. Como se observa en la tabla, las estimaciones de los costes de ejecución quedan relativamente cerca del tiempo real medido en el caso de MCI (ligeramente por debajo de los reales), sin embargo en el caso de $MC2E_{inv}$ la estimación del tiempo parece no ser

N	K	d	diferencia
500	200	500	$9.08442 \cdot 10^{-12}$
1000	400	500	$3.00927 \cdot 10^{-12}$
1000	400	1000	$4.64279 \cdot 10^{-13}$
1500	600	1000	$2.18451 \cdot 10^{-8}$
1500	600	1500	$2.49918 \cdot 10^{-9}$
2000	800	1500	$7.78951 \cdot 10^{-12}$
2000	800	2000	$2.79031 \cdot 10^{-12}$

Tabla 3.7: Promedio de las diferencias (en valor absoluto) entre las soluciones calculadas por $MC2E_{bas}$ y las calculadas por $MC2E_{inv}$ para un Modelo de Ecuaciones Simultáneas.

tan buena, distando, en la mayoría de los casos, alrededor del 32 % del tiempo real. Esta sobrestimación del tiempo de ejecución en $MC2E_{inv}$ puede deberse a que la reducción en operaciones supone también una reducción en fallos de caché que no ha sido tenida en cuenta.

	N:	500	1000	1500	2000	2500
	d:	500	500	1000	1000	1500
Algoritmo	Tipo de medida	Tiempo				
MCI	real	18.17	230.78	1073.21	3118.85	7181.54
	estimado	17.99	217.62	1018.15	3088.93	6907.37
	diferencia	0.18	13.16	55.06	29.92	274.17
	porcentaje	0.99 %	5.70 %	5.13 %	0.96 %	3.82 %
$MC2E_{inv}$	real	11.80	196.63	1219.02	4217.42	10498.93
	estimado	14.43	218.37	1184.61	5400.73	13956.72
	diferencia	-2.63	-21.74	34.41	-1183.31	-3457.79
	porcentaje	22.29 %	11.06 %	2.82 %	28.06 %	32.93 %

Tabla 3.8: Tiempo de ejecución (en segundos) y tiempo estimado de los algoritmos MCI (algoritmo 2) y $MC2E_{inv}$ (algoritmo 4) en Kefren, cuando varía el número de variables endógenas (N) y el tamaño muestral (d)

.

En futuras estimaciones del tiempo de ejecución de ambos algoritmos, debe ser tenido en cuenta la sobrestimación producida en $MC2E_{inv}$ para evitar posibles desajustes.

3.7. Conclusiones

En este capítulo se han desarrollado y estudiado algoritmos secuenciales para la resolución de Modelos de Ecuaciones Simultáneas, basados en los estimadores MCI y MC2E. Se han presentado versiones básicas de dichos algoritmos y además, en el caso de MC2E, se ha presentado una nueva versión utilizando descomposiciones matriciales cuyo coste computacional es menor que el del algoritmo básico.

Dicha nueva forma de calcular el estimador MC2E basado en una expresión para la inversa de una matriz es la aportación más importante del capítulo. Utilizando dicha expresión se pueden reutilizar cálculos anteriores y reducir considerablemente el tiempo de ejecución. Se ha demostrado matemáticamente que el número de operaciones en coma flotante utilizado al resolver una ecuación usando la descomposición de la inversa es siempre menor que en el caso de utilizar la expresión normal del estimador. Además, en el estudio experimental se ha mostrado que el tiempo puede llegar a ser hasta 10 veces menor.

Capítulo 4

Algoritmos paralelos para la resolución de Modelos de Ecuaciones Simultáneas

En este capítulo se desarrollan y analizan versiones en paralelo de los algoritmos mostrados en el capítulo anterior. Las grandes necesidades de computación motivan el desarrollo de algoritmos paralelos que reduzcan el tiempo de ejecución utilizando sistemas de altas prestaciones con gran número de CPUs.

La paralelización de los algoritmos se ha realizado en memoria distribuida y en la implementación se ha utilizando la librería MPI [15, 64, 71] descrita en el capítulo 2. La paralelización se ha llevado a cabo a diferentes niveles. En las operaciones matriciales se han utilizado llamadas a PBLAS [8] y ScaLAPACK [11]. A niveles más altos, se han realizado operaciones al comienzo del algoritmo y se han distribuido los datos resultantes de manera que la resolución de las ecuaciones del sistema se pueda hacer de manera independiente, con lo cual se consigue una paralelización a alto nivel, ya que que todos los procesadores pueden estar resolviendo diferentes ecuaciones a la vez.

El capítulo aporta el desarrollo de algoritmos distribuidos para la resolución de M.E.S. en sistemas de altas prestaciones. Se realiza un estudio de los tiempos de ejecución y del speed-up para distintos tamaños del problema, y se analizan los tiempos de comunicación, comparando su coste en relación con el coste de las operaciones aritméticas cuando varía el tamaño del problema y el número de procesadores.

4.1. Modelado de las comunicaciones

En el desarrollo de los algoritmos se va a utilizar el esquema "maestro-esclavo". Se considera un sistema con p procesadores con identificadores desde 0 hasta $p-1$. El procesador P_0 será considerado el maestro y el resto (desde P_1 hasta P_{p-1}) los esclavos.

Supondremos que las matrices de los datos de las variables, X e Y, están distribuidas en todos los procesadores en el estilo ScaLAPACK. Cuando el algoritmo termina su ejecución la solución estará distribuida en la misma forma.

En el estudio teórico se utilizará un modelo clásico de las comunicaciones [42, 43, 44] en el que se asume que t_s es el *start-up time*, o tiempo de establecimiento de contacto en comunicaciones punto a punto, y t_w es el *word-sending time*, o tiempo de envío de un dato.

En algunos casos se realizan comunicaciones colectivas, y el coste de establecimiento de la comunicación y de envío de un dato no coincide con los valores de las comunicaciones punto a punto, por lo que puede ser preferible utilizar unos parámetros particulares para la comunicación colectiva que se esté realizando. Por ejemplo, para un broadcast se puede utilizar t_{s_b} para representar el tiempo se establecimiento de las comunicaciones y t_{w_b} para el coste de envío de un dato.

En algunas partes de los algoritmos es necesario distribuir una matriz a todos los procesadores en el estilo ScaLAPACK, y para ello se utiliza la rutina de MPI `MPI_ Alltoall`. Al utilizarse esta rutina directamente y no saber cómo está implementada en diferentes sitemas, puede ser preferible considerarla como una caja negra. El coste de esta rutina es representado por $T_{A2A}(T, p)$, donde T es el total de datos a enviar y p es el número de procesadores. Por ejemplo, en comunicaciones punto a punto donde un mismo procesador mande T datos al resto de procesadores, el coste de la rutina es aproximadamente $T_{A2A}(T, p) = (p-1)t_s + Tt_w$.

4.2. Paralelización del Algoritmo MCO_{bas}

En el algoritmo 7 se muestra la versión paralela de MCO_{bas}. La multiplicación de matrices y la resolución del sistema de ecuaciones se realizan de forma paralela utilizando todos los procesadores disponibles. Para ello se pueden utilizar las librerías de ScaLAPACK y PBLAS. En los experimentos realizados en la sección 4.7 se ha utilizado la rutina *pdgemm* para las multiplicaciones y *pdgesv* para resolver el sistema de ecuaciones (línea 3), ambas rutinas de PBLAS y ScaLAPACK respectivamente.

Algoritmo 7 Algoritmo $PMCO_{bas}$

Entrada: $X \in \mathbb{R}^{d\times K}$ e $Y \in \mathbb{R}^{d\times N}$ (distribuidas al estilo de ScaLAPACK)
Salida: $\beta \in \mathbb{R}^{K\times N}$ e $\hat{Y} \in \mathbb{R}^{d\times N}$ (distribuidas al estilo de ScaLAPACK)
1: Hacer X^TX {Multiplicación en paralelo}
2: Hacer X^TY {Multiplicación en paralelo}
3: Hacer $(X^TX)^{-1}(X^TY)$ resolviendo el sistema $X^TY = (X^TX)\beta$ en paralelo
4: **Si** estimación=**Verdadero Entonces**
5: Hacer $\hat{Y} = X\beta$ {Multiplicación en paralelo}
6: **Fin si**

El algoritmo 7 coincide plenamente con el algoritmo 1 salvo que las operaciones matriciales se realizan en paralelo entre todos los procesadores. Las llamadas se hacen a funciones de ScaLAPACK en lugar de LAPACK, y se asume que las matrices X e Y están distribuidas al comienzo del algoritmo. Al terminar $PMCO_{bas}$, la solución queda distribuida en el sistema al estilo ScaLAPACK.

El coste teórico de multiplicar dos matrices en paralelo o de resolver un sistema de ecuaciones también en paralelo usando p procesadores, es el tiempo de realizar las operaciones aritméticas (T_A) más el tiempo de realizar las comunicaciones (T_C). Al utilizarse las rutinas *pdgemm* y *pdgesv* directamente y no saber cómo están implementadas en diferentes sistemas, se aproximarán los costes aritméticos por los dados para el caso secuencial divididos por p y se considerarán de nuevo los costes de comunicación como una caja negra.

Por lo tanto el coste de $PMCO_{bas}$ cuando se usan p procesadores es el coste de la ecuación 3.1 dividido por p más un coste en comunicaciones que es de menor orden:

$$T_{PMCO_{bas}}(N,d,K,\delta_{est}) = \frac{2}{3}\frac{K^3}{p} + 2\frac{K^2}{p}(d+N) + 2\frac{N}{p}Kd + \delta_{est}2\frac{N}{p}Kd + T_{C_{MCO}} \quad (4.1)$$

donde δ_{est} es 1 si *estimación* es requerida en el algoritmo (cuando se realiza la estimación de la variable dependiente Y) y 0 en otro caso.

4.3. Paralelización del Algoritmo de MCI

En el algoritmo 8 se muestra la versión paralela de MCI ($PMCI$). En el primer paso se obtiene la matriz Π mediante una llamada a la rutina de $PMCO_{bas}$ presentada en el punto anterior (algoritmo 7). Π se usará en la resolución de las ecuaciones. Al resolver cada ecuación, se forma un sistema de ecuaciones que será resuelto a continuación, y para ello la única información necesaria es la matriz Π y las filas correspondientes a esa ecuación de la matriz B y Γ. Como cada procesador resuelve un subconjunto del conjunto total de ecuaciones, cada procesador necesita solo acceder a Π, B_{p_k} y Γ_{p_k}, siendo estas últimas matrices las filas de B y Γ que corresponden a las ecuaciones que le toca resolver al procesador P_k. De esta forma el procesador P_k no necesita recibir datos de la muestra, sino solo la matriz Π y datos de la estructura del sistema.

Si B y Γ están distribuidas en todos los procesadores al comienzo del algoritmo, todos los procesos tendrán la estructura completa del sistema y no será necesario enviarles la estructura de las ecuaciones que deben resolver desde otro procesador.

Si B y Γ están en el procesador P_0, como cada procesador solo necesita una parte de B y Γ ($\frac{N}{p}$ filas de cada matriz), P_0 envía solo esa parte.

Debido a que normalmente ecuaciones contiguas tienen estructuras similares en el sentido de tamaño, número de endógenas y exógenas, etc. (esto es debido a la forma de construir los M.E.S., que no es otra que mediante la construcción de ecuaciones por expertos en el tema, que modelan diversos problemas, y a continuación la

unión de todas las ecuaciones en un mismo sistema), ecuaciones adyacentes no son asignadas al mismo procesador, sino que se asignan de forma cíclica para evitar que un procesador tenga muchas ecuaciones grandes y otro tenga muchas pequeñas.

Algoritmo 8 Algoritmo $PMCI$

Entrada: $X \in \mathbb{R}^{d\times K}$ e $Y \in \mathbb{R}^{d\times N}$ (distribuidas al estilo de ScaLAPACK), $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$ (distribuidas $\frac{N}{p}$ filas por procesador)
Salida: $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$ (distribuidas $\frac{N}{p}$ filas por procesador)

```
1: Π^T = PMCO_bas(Y,X,Falso)
2: Distribuir Π en todos los procesadores
3: EN PARALELO cada procesador q = 0, ..., p − 1 HACE
4: Para j=1...N/p Hacer
5:    i = q + (j − 1)p + 1
6:    Si ecuación i es exactamente identificada Entonces
7:       Resolver −B_i Π = Γ_i
8:    Fin si
9: Fin Para
10: FIN PARALELO
```

Para obtener Π, se hace una llamada a la rutina $PMCO_{bas}$ usando la matriz de endógenas como matriz dependiente (con tamaño $d \times N$) y la matriz de exógenas como matriz explicativa (con tamaño $d \times K$). El coste teórico de esta llamada es $T_{PMCO_{bas}}(N, d, K, 0)$. Cuando $PMCO_{bas}$ finaliza, la matriz Π está guardada en el estilo de ScaLAPACK y repartida por tanto entre todos los procesadores con lo que se hace necesario enviar la matriz completa a todos ellos. El coste de este envío es $T_{A2A}(KN, p)$.

El coste de resolver $-B_i\Pi = \Gamma_i$ es $\frac{2}{3}(K - k_i)^3 + 2(K - k_i)^2$ para calcular B_i, y $2Nk_i$ para calcular Γ_i (tal y como se explicó en la versión secuencial). De esta forma, el coste total si todas las ecuaciones del sistema pudieran ser resueltas por MCI es:

$$T_{PMCI}(N, d, K) = T_{PMCO_{bas}}(N, d, K, 0) + T_{A2A}(KN, p) + \max_{q=0,\ldots,p-1} \left\{ \sum_{j=1}^{\frac{N}{p}} \left(\tfrac{2}{3}(K - k_{q+(j-1)p+1})^3 + 2(K - k_{q+(j-1)p+1})^2 + 2Nk_{q+(j-1)p+1} \right) \right\} \approx T_{PMCO_{bas}}(N, d, K, 0) + \sum_{i=1}^{\frac{N}{p}} \left(\tfrac{2}{3}(K - k_i)^3 + 2(K - k_i)^2 + 2Nk_i \right) \tag{4.2}$$

Para simplificar la expresión se asume que la estructura de todas las ecuaciones es similar, y que el coste de resolver $\frac{N}{p}$ ecuaciones es similar al de resolver cualesquiera otras $\frac{N}{p}$ ecuaciones, por lo que se asume que todos los procesadores tardan lo mismo.

De la misma forma que ocurría con T_{MCI}, el coste total de T_{PMCI} dependerá del valor de K y N y en menor medida de d. También se ha conseguido dividir por p el

tiempo de resolución de las ecuaciones, el cual representa la mayor parte del tiempo total del algoritmo. En las tablas 4.2 y 4.3 de la sección 4.7 se muestra un gran incremento en el coste total cuando K y N aumentan y d queda fijada, mientras que en el capítulo anterior se vio que el coste no depende de d (o lo hace en una medida insignificante).

Cuando el algoritmo MCI finaliza, cada procesador tiene la solución de $\frac{N}{p}$ ecuaciones. Si es necesario mandar los resultados al procesador P_0, se podrían usar, por ejemplo, comunicaciones punto a punto, siendo en este caso el coste aproximadamente $(p-1)t_s + t_w \sum_{i=1}^{N-\frac{N}{P}} (n_i + k_i - 1) \approx (p-1)t_s + t_w NK$ (ya que $n_i + k_i - 1 = K$, puesto que las ecuaciones deben ser exactamente identificadas).

4.4. Paralelización del Algoritmo $MC2E_{bas}$

A continuación se presenta la paralelización del algoritmo 3 (algoritmo $MC2E_{bas}$). La paralelización de dicho algoritmo se hace necesaria debido a que pueden existir ecuaciones en las que $MC2E_{inv}$ no puede ser usado (cuando ocurre que $k_i = 0$ o $n_i = 1$), utilizándose en su lugar $MC2E_{bas}$.

Debido a que para resolver una ecuación mediante MC2E es necesario aplicar MCO en dicha ecuación una vez se han sustituido las endógenas explicativas por sus variables *proxy*, el procesador que deba resolver dicha ecuación debe calcular dichas variables *proxy* previamente o disponer de ellas. Este cálculo tiene un coste alto, por lo que se debe de hacer en paralelo. Además, puede ocurrir que otro procesador haya calculado ya dicha variable, por lo que no sería necesario volverla a calcular. Por estas razones, y tal y como se explicó en el algoritmo secuencial, es recomendable calcular todas las variables *proxy* al comienzo del algoritmo aprovechando todos los procesadores, y distribuirlas a continuación a todos los procesadores de manera que estén disponibles para poder resolver cada ecuación.

El algoritmo $PMC2E_{bas}$ (algoritmo 9) muestra la paralelización de $MC2E_{bas}$ resolviendo un sistema completo. La matriz X_{e_i} (de dimensión $d \times (n_i - 1 + k_i)$) está formada por las variables *proxy* que han sustituido a las endógenas de la ecuación y por las variables exógenas.

Cuando MCO_{bas} finaliza, las matrices $\hat{Y}$ y X están distribuidas en todos los procesadores al estilo de ScaLAPACK y es necesario que cada uno tenga las variables que necesita. Puesto que no todos los procesadores usarán todas las variables *proxy* y todas las variables exógenas, cada procesador enviará a otro solo las partes que tenga él de las variables que necesite el otro procesador. Por simplicidad se supondrá que todos los procesadores necesitan todas las variables, por lo que se enviarán todas a todos. El coste de este envío es $T_{A2A}(d(N+K), p)$.

Además también hay que enviar la estructura de las ecuaciones que debe resolver cada procesador (filas de las matrices B y Γ que le correspondan). Si B y Γ están en el procesador P_0, éste envía solo $\frac{N}{p}$ filas de cada una de las matrices a cada uno

de los procesadores restantes. Cuando el algoritmo finaliza, cada procesador tiene la solución de $\frac{N}{p}$ ecuaciones.

Algoritmo 9 Algoritmo $PMC2E_{bas}$

Entrada: $X \in \mathbb{R}^{d\times K}$ e $Y \in \mathbb{R}^{d\times N}$ (distribuidas al estilo de ScaLAPACK), $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$ (distribuidas $\frac{N}{p}$ filas por procesador)

Salida: $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$ (distribuidas $\frac{N}{p}$ filas por procesador)

1: $\hat{Y} = PMCO_{bas}(Y,X,$**Verdadero**$)$
2: Distribuir $\hat{Y}$ y X a todos los procesadores
3: EN PARALELO cada procesador $q = 0, ..., p-1$ HACE
4: **Para** j=1...$\frac{N}{p}$ **Hacer**
5: $\quad i = q + (j-1)p + 1$
6: $\quad \beta_i = MCO_{bas}(y_i,X_e,$**Falso**$)$
7: $\quad$ Sustituir los valores de β_i en las incógnitas de B_i y Γ_i
8: **Fin Para**
9: FIN PARALELO

El coste total considerando la misma aproximación que en la ecuación 4.2, es:

$$\begin{aligned} T_{PMC2E_{inv}}(N,d,K) &= T_{PMCO_{bas}}(N,d,K,1) + T_{A2A}(d(N+K),p) + \\ &\max_{q=0,\ldots,p-1}\left\{\sum_{j=1}^{\frac{N}{p}}\left(T_{MCO_{bas}}(1,d,k_{q+(j-1)p+1}+n_{q+(j-1)p+1}-1,0)\right)\right\} \approx \\ &T_{PMCO_{bas}}(N,d,K,1) + \sum_{i=1}^{\frac{N}{p}}\left(T_{MCO_{bas}}(1,d,k_i+n_i-1,0)\right) \end{aligned} \tag{4.3}$$

Al igual que en su versión secuencial, el coste total de $T_{PMC2E_{bas}}$ depende del valor de K, N y d. Se ha conseguido dividir por p el tiempo de resolución de las ecuaciones, el cual representa la mayor parte del tiempo total del algoritmo. En la sección 4.7 tablas 4.4 y 4.5, se muestra el crecimiento del tiempo de ejecución cuando aumenta el tamaño del problema.

4.5. Paralelización del algoritmo $MC2E_{inv}$

Se ha hecho hincapié en la importancia del estimador MC2E (que es aplicable en todas las ecuaciones identificadas) y en los grandes problemas que presenta la versión básica de él. Además se han fijado una serie de objetivos a conseguir para obtener un algoritmo más eficiente, objetivos que se han logrado con la descomposición de la matriz inversa basada en el teorema 2.1. A continuación se presenta la versión paralela de dicho algoritmo (algoritmo 10). La paralelización se hace relativamente

sencilla debido a la disposición de los cálculos previos al comienzo, cálculos que ahora se pueden realizar utilizando todos los procesadores.

Algoritmo 10 Algoritmo $PMC2E_{inv}$

Entrada: $X \in \mathbb{R}^{d\times K}$ e $Y \in \mathbb{R}^{d\times N}$ (distribuidas al estilo de ScaLAPACK), $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$ (distribuidas $\frac{N}{p}$ filas por procesador)

Salida: $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$ (distribuidas $\frac{N}{p}$ filas por procesador)

1: $\hat{Y} = PMCO_{bas}(Y,X,$**Verdadero**$)$ {guardando Π, X^TX y X^TY}
2: Hacer $\hat{Y}^T\hat{Y}$ {multiplicaciones en paralelo}
3: Distribuir $\Pi, X^TX, X^TY, \hat{Y}, \hat{Y}^T\hat{Y}$ a todos los procesadores
4: EN PARALELO cada procesador $q = 0, ..., p-1$ HACE
5: **Para** j=1...$\frac{N}{p}$ **Hacer**
6: $\quad i = q + (j-1)p + 1$
7: $\quad \beta_i = MCO_{inv}(y_i,\hat{Y},X,X^TX,X^TY,\hat{Y}^T\hat{Y})$
8: $\quad$ Sustituir los valores de β_i en las incógnitas de B_i y Γ_i
9: **Fin Para**
10: FIN PARALELO

Al igual que ocurría en los algoritmos anteriores, al terminar $PMCO_{bas}$ la matriz resultante se encuentra repartida en todos los procesadores al estilo de ScaLAPACK y es necesario que todos la tengan al completo, por lo que cada procesador debe envíar a los demás su parte de la matriz. De igual forma ocurre con las matrices $\Pi, X^TX, X^TY, \hat{Y}, \hat{Y}^T\hat{Y}$ que estando distribuidas al estilo de ScaLAPACK, deben de ser envíadas al completo a cada procesador. Al igual que en los algoritmos anteriores, cuando el algoritmo finaliza, cada procesador tiene la solución de $\frac{N}{p}$ ecuaciones,

La función MCO_{inv} que aparece en la línea 7 del algoritmo 10 es la mostrada en el algoritmo 4 del capítulo anterior, y por lo tanto es una función secuencial que es llamada por cada uno de los procesadores al resolver las diferentes ecuaciones que le corresponden.

El coste total de este algoritmo es:

$$\begin{gathered} T_{PMC2E_{inv}}(N,d,K) = \\ T_{PMCO_{bas}}(N,d,K,1) + \tfrac{2N^2d}{p} + T_{C_{\hat{Y}^T\hat{Y}}} + T_{A2A}(K^2+N^2+2KN+dN,p) + \\ \max_{q=0,\ldots,p-1} \left\{ \sum_{j=1,\ t=q+(j-1)p+1}^{\frac{N}{p}} \left(\tfrac{8}{3}(n_t-1)^3 + \tfrac{8}{3}k_t^3 + 2(n_t-1+k_t)^2 + \right.\right. \\ \left.\left. 6(n_t-1)^2k_t + 2k_t^2(n_t-1) + (n_t-1)^2 + k_t^2) \right\} \approx \\ T_{PMCO_{bas}}(N,d,K,1) + \tfrac{2N^2d}{p} + \\ \sum_{i=1}^{\frac{N}{p}} \left(\tfrac{8}{3}n_i^3 + \tfrac{8}{3}k_i^3 + 2(n_i+k_i)^2 + 6n_i^2k_i + 2k_i^2n_i + n_i^2 + k_i^2 \right) \end{gathered} \tag{4.4}$$

El coste de realizar $\hat{Y}^T\hat{Y}$ en paralelo (línea 2 del algoritmo), tal y como se explicó al principio del capítulo, está formado por el coste de las operaciones aritméticas más el coste de las comunicaciones. El coste aritmético es representado por la suma del coste de las operaciones aritméticas (aproximado por el coste de una multiplicación secuencial dividido por el número de procesadores usados). Y el coste de las comunicaciones es representado por $T_{C_{\hat{Y}^T\hat{Y}}}$ y es despreciado al hacer la aproximación por ser un coste de orden menor al aritmético. De nuevo, se considera la misma aproximación que en la expresión 4.2 para simplificar la expresión 4.4.

4.6. Algoritmo paralelo para la resolución de Modelos de Ecuaciones Simultáneas

En el algoritmo 11 se expone un esquema paralelo de resolución de M.E.S. Este algoritmo no es más que el ya expuesto anteriormente (algoritmo 6) pero en su versión paralela. Al igual que en la versión secuencial, en el algoritmo se encontrarán ecuaciones tanto identificadas como no identificadas. Las no identificadas no se resolverán y para las otras se usará $PMCI$ si la ecuación es exactamente identificada o $PMC2E_{inv}$ si es sobreidentificada y se dan las condiciones en la ecuación para que $PMC2E_{inv}$ pueda ser utilizado. En las ecuaciones sobreidentificadas que no pueda ser usado $PMC2E_{inv}$ se utilizará $PMC2E_{bas}$ en su lugar.

Al igual que en Algoritmo MES, en la línea 7 de $PMES$ se comprueba si el número de variables exógenas y el número de variables endógenas (aparte de la principal) en la ecuación es mayor que cero. Si es así se pueden construir las submatrices usadas en $MC2E_{inv}$, y si no es así hay que usar $MC2E_{bas}$.

La tabla 4.1 muestra el resumen de los costes de resolver un M.E.S. mediante el uso de los algoritmos paralelos propuestos en esta sección.

Para las aproximaciones se ha tomado el promedio de variables exógenas ($\bar{k}$) y endógenas ($\bar{n}$) que aparece en cada ecuación.

Las conclusiones expuestas para la tabla 3.1 siguen siendo válidas para las versiones paralelas, aunque los tiempos en las comunicaciones pueden hacer cambiar ciertas conclusiones expuestas en la versión secuencial. Por ejemplo, $MC2E_{inv}$ tiene menor coste que $MC2E_{bas}$, lo que da que pensar que en paralelo seguirá ocurriendo lo mismo con los algoritmos $PMC2E_{inv}$ y $PMC2E_{bas}$. Sin embargo, en un primer análisis del algoritmo $PMC2E_{inv}$ se observa una cantidad de comunicaciones muy superior a las usadas por $PMC2E_{bas}$ (se envían $K^2+N^2+2KN+dN$ datos mientras que en $MC2E_{bas}$ se envían dN). Esto en un principio puede hacer pensar que este algoritmo deja de ser rentable en paralelo mientras que su versión secuencial sí lo era. Sin embargo, este tiempo se ve compensado por la reducción del coste de la multiplicación $\hat{Y}^T\hat{Y}$ que se realiza en paralelo (aunque por otra parte también aporta más comunicaciones). Sin embargo, una vez analizados los tiempos de comunicación en la sección experimental se concluye que, siempre que no se utilicen un número de procesadores excesivamente alto, los tiempos de comunicación no suponen un coste

Algoritmo 11 Algoritmo $PMES$

Entrada: $X \in \mathbb{R}^{d\times K}$ e $Y \in \mathbb{R}^{d\times N}$ (distribuidas al estilo de ScaLAPACK), $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$ (distribuidas $\frac{N}{p}$ filas por procesador)

Salida: $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$ (distribuidas $\frac{N}{p}$ filas por procesador)

1: $\hat{Y} = PMCO_{bas}(Y, X,$**Verdadero**$)$ {guardando Π, X^TX y X^TY}
2: $\hat{Y}^T\hat{Y}$ {multiplicación en paralelo}
3: Distribuir $\Pi, X^TX, X^TY, \hat{Y}, \hat{Y}^T\hat{Y}$ en todos los procesadores
4: EN PARALELO cada procesador $q = 0, ..., p-1$ HACE
5: **Para** j=1...$\frac{N}{p}$ **Hacer**
6: $i = q + (j-1)p + 1$
7: **Si** la ecuación i es exactamente identificada **Entonces**
8: Resolver $-B_i\Pi = \Gamma_i$ {Se utiliza MCI}
9: **Si no**, **Si** la ecuación i es sobreidentificada **Entonces**
10: **Si** $k_i > 0$ y $n_i > 1$ **Entonces**
11: $\beta_i = MCO_{inv}(y_i,\hat{Y},X,X^TX,X^TY,\hat{Y}^T\hat{Y})$ {Se utiliza $MC2E_{inv}$}
12: **Si no**
13: $\beta_i = MCO_{bas}(y_i,X_e,$**Falso**$)$ {Se utiliza $MC2E_{bas}$}
14: **Fin si**
15: Sustituir los valores de β_i en las incógnitas de B_i y Γ_i
16: **Si no**
17: no hacer nada {la ecuación i no está identificada}
18: **Fin si**
19: **Fin Para**
20: FIN PARALELO

Algoritmo	coste
MCI	$T_{PMCI}(N,d,K) = T_{PMCO_{bas}}(N,d,K,0)+$ $\sum_{i=1}^{\frac{N}{p}} \left(\frac{2}{3}(K-k_i)^3 + 2(K-k_i)^2 + 2Nk_i\right)$ $\approx \frac{N}{p}(K-\bar{k})^3$
$PMC2E_{bas}$	$T_{PMC2E_{bas}}(N,d,K) = T_{PMCO_{bas}}(N,d,K,1)+$ $\sum_{i=1}^{\frac{N}{p}} (T_{MCO_{bas}}(1,d,k_i+n_i-1,0))$ $\approx \frac{N}{p}(\frac{2}{3}(\bar{k}+\bar{n})^3 + 2(\bar{k}+\bar{n})^2 d)$
$PMC2E_{inv}$	$T_{PMC2E_{inv}}(N,d,K) = T_{PMCO_{bas}}(N,d,K,1) + \frac{2N^2d}{p}+$ $\sum_{i=1}^{\frac{N}{p}} \left(\frac{8}{3}(n_i-1)^3 + \frac{8}{3}k_i^3 + 2(n_i-1+k_i)^2 + 6(n_i-1)^2k_i+\right.$ $\left. 2k_i^2(n_i-1) + (n_i-1)^2 + k_i^2\right)$ $\approx \frac{N}{p}(\frac{8}{3}\bar{n}^3 + \frac{8}{3}\bar{k}^3 + 6\bar{n}^2\bar{k} + 2\bar{k}^2\bar{n} + \bar{n}^2 + \bar{k}^2)$

Tabla 4.1: Resumen de los tiempos teóricos de los algoritmos $PMCI$, $PMC2E_{bas}$ y $PMC2E_{inv}$.

destacable frente al coste aritmético y pueden ser despreciados. De hecho, han sido omitidos en las aproximaciones mostradas en la tabla 4.1. Además es importante resaltar que el mayor coste en todos los algoritmos reside en la resolución de las ecuaciones y no en los cálculos previos (tal y como se vio en los experimentos del capítulo anterior).

La tabla 4.1 muestra que $PMCI$ es el algoritmo con menor coste (cuando es posible usarlo). El número de operaciones aritméticas sigue siendo menor, tal y como se explicó en la tabla resumen del capítulo anterior, y la cantidad de datos a mandar entre los procesadores también es menor que en $PMC2E_{bas}$ (puesto que $PMCI$ envía KN y $PMC2E_{bas}$ dN datos) y muchísimo menor que en $PMC2E_{inv}$ (que envía $K^2 + N^2 + 2KN + dN$ datos).

4.7. Estudio experimental

En esta sección se van a describir diversos experimentos mediante los cuales se estudian los algoritmos propuestos anteriormente. Los modelos para dichos experimentos han sido generados de la misma forma que en la sección de experimentos del capítulo anterior.

Al igual que en el estudio secuencial, para simplificar los experimentos, se han generado los modelos con valor de K un 40 % del valor de N, y se ha variado N, d y el número de procesadores usados, comparando los tiempos resultantes y los speed-ups. En esta sección se han tomado todas las medidas en las máquinas Kefren

y Marenostrum (ver sección 2.1.1).

Se pretenden estudiar experimentalmente los siguientes puntos:

- La influencia de los parámetros N, K y d en el tiempo de ejecución en los diferentes algoritmos.
- La reducción del tiempo de ejecución cuando se usan varios procesadores.
- El coste que suponen las comunicaciones respecto al coste de cada uno de los algoritmos cuando se incrementa el número de procesadores.

Las tablas 4.2 y 4.3 muestran el tiempo de ejecución y el speed-up obtenido con el algoritmo $PMCI$ al ser ejecutado en Kefren y Marenostrum. Los sistemas generados tienen todas las ecuaciones exactamente identificadas para poder ser resueltas mediante MCI. Como se observa en ambas tablas, los speed-ups son bastante altos cuando el tamaño del problema es grande, e incluso cuando es pequeño y no se usan muchos procesadores. Esto se debe a que el tiempo de comunicación si se usan muchos procesadores y un tamaño del problema pequeño deja de ser despreciable. Se observa un crecimiento en el coste cuando se incrementa N (y por lo tanto K), al contrario de lo que ocurre al aumentar d (tal y como se vio en los experimentos del capítulo anterior).

N :	500	1000	1500	2000	2500
d :	500	500	1000	1000	1500
p	Tiempo				
1	18.17	230.78	1073.21	3118.85	7181.54
2	9.28	137.53	530.95	1613.91	3663.72
4	5.04	58.59	267.45	809.66	1839.56
8	2.64	29.77	136.09	395.17	914.82
16	2.02	17.78	67.52	198.47	453.58
Speed-up					
2	1.96	1.68	2.02	1.93	1.96
4	3.61	3.94	**4.01**	3.85	3.90
8	6.88	7.75	7.89	7.89	7.85
16	9.00	12.98	15.89	15.71	15.83

Tabla 4.2: Tiempo de ejecución (en segundos) y speed-up del algoritmo 8 ($PMCI$) en Kefren, variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

También se pueden observar varios super speed-ups, casi todos en la tabla 4.3. Los casos como el 4.01 de la tabla 4.2 y 4.06 y 8.07 de la tabla 4.3 son mínimos y no significativos, siendo el resto de super speed-ups más notables. Estos super speed-ups (marcados en negrita) se deben a una mejor eficiencia del uso de

N :	500	1000	1500	2000	2500
d :	500	500	1000	1000	1500
p	Tiempo				
1	6.43	68.37	346.06	1263.33	2455.03
4	1.73	18.52	87.18	310.81	550.55
8	1.01	9.56	44.52	156.62	287.36
16	0.63	5.25	24.53	74.63	147.30
32	0.45	4.78	12.62	38.99	77.82
64	0.33	1.90	7.70	21.40	40.39
Speed-up					
4	3.72	3.69	3.97	**4.06**	**4.46**
8	6.37	7.15	7.77	**8.07**	**8.54**
16	10.22	13.02	14.11	**16.93**	**16.67**
32	14.44	14.29	27.42	**32.40**	31.55
64	19.27	35.96	44.92	59.03	60.79

Tabla 4.3: Tiempo de ejecución (en segundos) y speed-up del algoritmo 8 ($PMCI$) en Marenostrum, variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

la memoria caché cuando se utilizan varios procesadores, reduciendo considerablemente el número de fallos de ésta. El super speed-up deja de darse cuando aumenta el número de procesadores (con respecto al tamaño del problema) y dejan de ser despreciables los tiempos de comunicación.

El algoritmo paralelo correspondiente a $PMC2E_{bas}$ (algoritmo 9) es analizado experimentalmente (tablas 4.4 y 4.5) en Kefren y Marenostrum. Al igual que en las dos tablas anteriores, se dan buenos resultados de speed-up y estos son excelentes cuando el tamaño del problema es grande. También se da super speed-up (marcado en negrita) dado que la cantidad de memoria para operaciones que maneja el procesador es muy grande (superior a la del algoritmo de MCI). Si aproximamos $n_i - 1 + k_i$ con K, en cada ecuación como mínimo se debe tener en memoria dK para poder hacer MCO y eso son ya 11MB, y además hay que hacer multiplicaciones, inversas, etc. y esto en cada una de las N ecuaciones. Por lo tanto, para tamaños grandes del problema se hace significativa la reducción de fallos de caché que se da al usar varios procesadores. También se observa un crecimiento en el coste similar al del algoritmo anterior cuando se incrementa N (y por lo tanto K).

Las tablas 4.6 y 4.7 muestran los tiempos de ejecución y el speed-up del algoritmo $PMC2E_{inv}$ (algoritmo 10), cuando se varía el tamaño del problema y el número de procesadores, en Kefren y en Marenostrum.

Tal y como se observa en las tablas, el speed-up no es malo para tamaños del problema grandes aunque en problemas pequeños ya no es tan satisfactorio. Además siempre es inferior a los dados en las tablas para MCI y $MC2E_{bas}$. Esto se debe a la

N :	500	1000	1500	2000	2500
d :	500	500	1000	1000	1500
p	Tiempo				
1	72.82	790.93	7031.96	19337.92	97874.21
2	38.77	412.97	3538.48	9767.20	48124.48
4	19.83	206.92	1916.58	5154.38	22541.92
8	10.70	108.68	1082.13	2639.33	10686.11
16	7.45	65.34	561.43	1493.48	5531.40
Speed-up					
2	1.88	1.92	1.99	1.98	**2.03**
4	3.67	3.82	3.67	3.75	**4.34**
8	6.81	7.28	6.50	7.33	**9.15**
16	9.77	12.10	12.53	12.95	**17.68**

Tabla 4.4: Tiempo de ejecución (en segundos) y speed-up del algoritmo 9 ($PMC2E_{bas}$) en Kefren, variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

N :	500	1000	1500	2000	2500
d :	500	500	1000	1000	1500
p	Tiempo				
1	148.59	1604.48	13762.89	45046.84	248294.57
4	39.92	409.74	3524.29	11813.29	63018.56
8	21.42	212.62	1891.09	6001.29	33575.22
16	15.86	138.49	926.25	3062.14	16466.12
32	8.16	72.18	493.78	1557.13	8476.88
Speed-up					
4	3.72	3.92	3.91	3.81	3.94
8	6.94	7.55	7.28	7.51	7.40
16	9.37	11.59	14.86	14.71	15.08
32	18.22	22.23	27.87	28.93	29.29

Tabla 4.5: Tiempo de ejecución (en segundos) y speed-up del algoritmo 9 ($PMC2E_{bas}$) en Marenostrum, variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

N :	500	1000	1500	2000	2500
d :	500	500	1000	1000	1500
p	Tiempo				
1	12.91	198.16	1225.33	4192.10	10217.02
2	7.59	105.05	639.04	2154.62	5363.62
4	4.17	53.82	333.30	1086.43	2700.53
8	2.85	29.45	185.37	562.03	1388.88
16	1.68	16.72	99.30	299.77	721.13
Speed-up					
2	1.70	1.89	1.92	1.95	1.90
4	3.10	3.68	3.68	3.86	3.78
8	4.53	6.73	6.61	7.46	7.36
16	7.68	11.85	12.34	13.98	14.17

Tabla 4.6: Tiempo de ejecución (en segundos) y speed-up del algoritmo 10 ($PMC2E_{inv}$) en Kefren, variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

N :	500	1000	1500	2000	2500
d :	500	500	1000	1000	1500
p	Tiempo				
1	21.20	352.22	2014.73	7005.57	18471.71
4	6.26	91.55	528.08	1820.22	4930.59
8	3.49	48.23	274.49	944.70	2536.07
16	2.12	26.97	148.08	521.12	1348.91
32	1.36	15.69	83.77	273.55	677.84
64	1.12	10.38	48.42	150.91	393.16
Speed-up					
4	3.39	3.85	3.82	3.85	3.75
8	6.07	7.30	7.34	7.42	7.28
16	10.00	13.06	13.61	13.44	13.69
32	15.55	22.45	24.05	25.61	27.25
64	18.96	33.94	41.61	46.42	46.98

Tabla 4.7: Tiempo de ejecución (en segundos) y speed-up del algoritmo 10 ($PMC2E_{inv}$) en Marenostrum, variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

gran cantidad de datos que hay que enviar a cada procesador. Por lo tanto, para que el tiempo empleado en comunicaciones sea despreciable la cantidad de operaciones aritméticas realizadas (o dicho de otra forma, el tamaño del problema) debe ser grande con respecto al número de procesadores. Aunque la cantidad de memoria usada sigue siendo importante, hay que tener en cuenta que por las características de este algoritmo, muchos datos se toman de operaciones previas y no hay que calcularlos. Esto reduce la cantidad de operaciones en coma flotante y aumenta el envío de datos entre procesadores, no alcanzando así el super speed-up que en los otros casos se ha dado. También en este caso se observa un crecimiento en el coste cuando se incrementa N (y por lo tanto K) más elevado que en los algoritmos anteriores.

Una vez estudiados los algoritmos desarrollados en el capítulo de forma experimental, procedemos a estudiar el coste que tienen en ellos las comunicaciones. El objetivo es poder predecir cuando deja de ser rentable utilizar más procesadores en la resolución de un problema por incrementar el tiempo de comunicaciones en mayor grado que se decrementa el tiempo de cálculos aritméticos. Para ello se ha medido la latencia y la tasa de transferencia de la red en Kefren, resultando los valores t_s=$0.11652 \cdot 10^{-5}$ y t_w=$2.14 \cdot 10^{-8}$.

Con el fin de comparar el tiempo de comunicación con el tiempo total de cada algoritmo, se han medido los tiempos de comunicaciones para la función T_{A2A} según el tamaño del problema (tabla 4.8). Como se observa en la tabla, la diferencia de tiempos entre los algoritmos es mínima. También el tiempo es despreciable comparado con los tiempos de ejecución para los mismos tamaños del problema dados en las tablas anteriores.

Sin embargo, si suponemos que Kefren pudiera tener infinitos procesadores sin variar los datos medidos para t_s y t_w, los tiempos para T_{A2A} serían los expuestos en la tabla 4.9. En este caso sí se observan diferencias significativas entre los tres algoritmos (incluso para tamaños del problema pequeños). Además, si suponemos que los tiempos de ejecución se seguirían reduciendo al utilizar más procesadores tal y como se ha estudiado en las tablas 4.2, 4.4 y 4.6, los tiempos de comunicación dados en esta tabla ya no son despreciables en comparación con los tiempos de ejecución que se darían en el caso de darse speed-ups parecidos a los estudiados en las tablas. Por ejemplo en el caso del algoritmo correspondiente a $MC2E_{inv}$, el tiempo de ejecución para 16 procesadores es 721.13 ($N = 2500$ y $d = 1500$ de la tabla 4.6). Este tiempo hemos visto que se corresponde casi por completo a tiempo de operaciones aritméticas puesto que el tiempo de comunicación es 2.09 segundos (tabla 4.8). Si empleáramos 256 procesadores el tiempo empleado en flops se reduciría aún más, por lo que sería inferior a 721.13, y sin embargo el tiempo de comunicación estaría alrededor de los 22000 segundos (estimando con la expresión de T_{A2A}) con lo que el tiempo empeoraría. Por lo tanto la conclusión es que hay un número de procesadores óptimo a partir del cual deja de ser rentable usar más para la ejecución del algoritmo. Este número óptimo depende del algoritmo, del tamaño del problema y del hardware usado, por lo que, antes de resolver un M.E.S., se

	N:	500	1000	1500	2000	2500
	d:	500	500	1000	1000	1500
p	Algoritmo	Tiempo				
2	$PMCI$	0.118	0.121	0.126	0.134	0.143
	$PMC2E_{bas}$	0.120	0.124	0.139	0.146	0.173
	$PMC2E_{inv}$	0.124	0.026	0.063	0.105	0.171
4	$PMCI$	0.351	0.356	0.364	0.375	0.390
	$PMC2E_{bas}$	0.355	0.361	0.383	0.395	0.434
	$PMC2E_{inv}$	0.361	0.389	0.444	0.507	0.606
8	$PMCI$	0.818	0.823	0.832	0.846	0.862
	$PMC2E_{bas}$	0.822	0.829	0.855	0.868	0.914
	$PMC2E_{inv}$	0.829	0.862	0.926	1.001	1.115
16	$PMCI$	1.750	1.756	1.767	1.782	1.801
	$PMC2E_{bas}$	1.755	1.763	1.793	1.808	1.860
	$PMC2E_{inv}$	1.764	1.800	1.874	1.958	2.090

Tabla 4.8: Tiempo de comunicación (en segundos) entre los distintos procesadores cuando se ejecutan los algoritmos $PMCI$, $PMC2E_{bas}$ y $PMC2E_{inv}$ en Kefren, variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

podrán utilizar las expresiones teóricas desarrolladas en este capítulo para estimar el número de procesadores óptimo a usar, haciendo un uso mas eficiente del hardware y optimizando a la vez el tiempo de ejecución.

Para la comparación entre los tiempos de procesamiento aritmético y los de comunicación de cada uno de los algoritmos se han representado ambos tiempos en gráficas (figuras 4.1, 4.2 y 4.3), variando el número de procesadores y el tamaño del problema, siendo K el 40 % de N en todos los casos. Los valores para 8 y 16 procesadores han sido tomados de las tablas anteriores, y para el resto de procesadores han sido estimados de la siguiente forma: el tiempo de procesamiento aritmético se ha estimado dividiendo el tiempo de ejecución para un procesador entre el número de procesadores usados, y el tiempo de comunicaciones ha sido estimado mediante la función $T_{A2A}(T,p) = (p-1)t_s + Tt_w$. Se observa en todos los algoritmos que los tiempos de procesamiento aritmético son mayores que los tiempos de comunicación para 8 y 16 procesadores, ocurriendo lo contrario para 64 y 128. Por ejemplo, para un tamaño del problema de $N = 2500$ y $d = 1500$, deja de ser rentable usar 64 procesadores para $PMCI$ y $PMC2E_{inv}$ mientras que sigue siendo rentable para el algoritmo $PMC2E_{bas}$.

	N:	500	1000	1500	2000	2500
	d:	500	500	1000	1000	1500
p	Algoritmo	Tiempo				
64	$PMCI$	15.852	41.738	84.880	145.279	222.936
	$PMC2E_{bas}$	37.423	67.623	188.422	248.821	460.219
	$PMC2E_{inv}$	71.075	219.484	517.167	856.266	1387.781
128	$PMCI$	49.469	153.832	327.772	571.286	884.377
	$PMC2E_{bas}$	136.438	258.196	745.226	988.741	1841.043
	$PMC2E_{inv}$	272.111	870.462	2070.642	3437.804	5580.735
256	$PMCI$	169.295	588.392	1286.888	2264.783	3522.076
	$PMC2E_{bas}$	518.543	1007.490	2963.279	3941.173	7363.804
	$PMC2E_{inv}$	1063.370	3466.196	8285.818	13775.997	22381.468

Tabla 4.9: Tiempo de comunicación estimado (en segundos) entre los distintos procesadores cuando se ejecutan los algoritmos $PMCI$, $PMC2E_{bas}$ y $PMC2E_{inv}$ en Kefren (simulado), variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

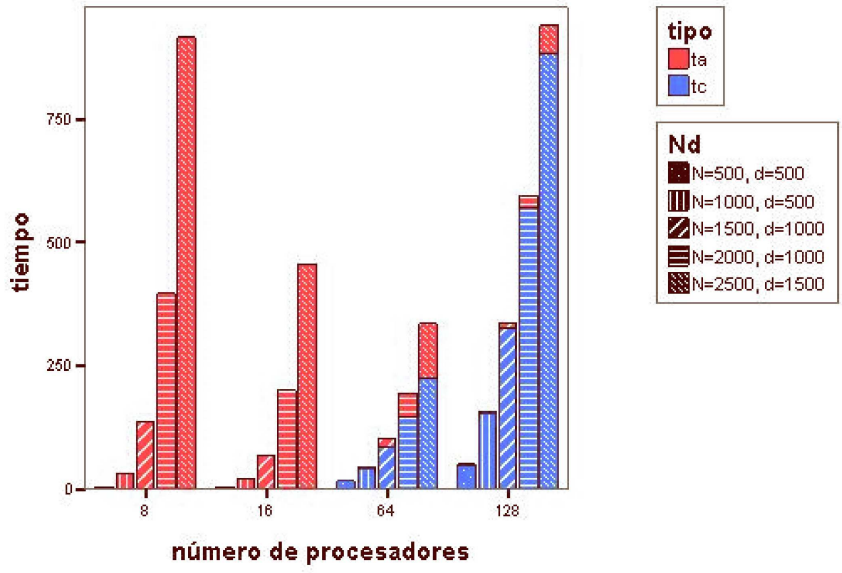

Figura 4.1: Gráfico comparativo de los tiempos de comunicación (tc) y de operaciones aritméticas (ta) del algoritmo $PMCI$ en Kefren, variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

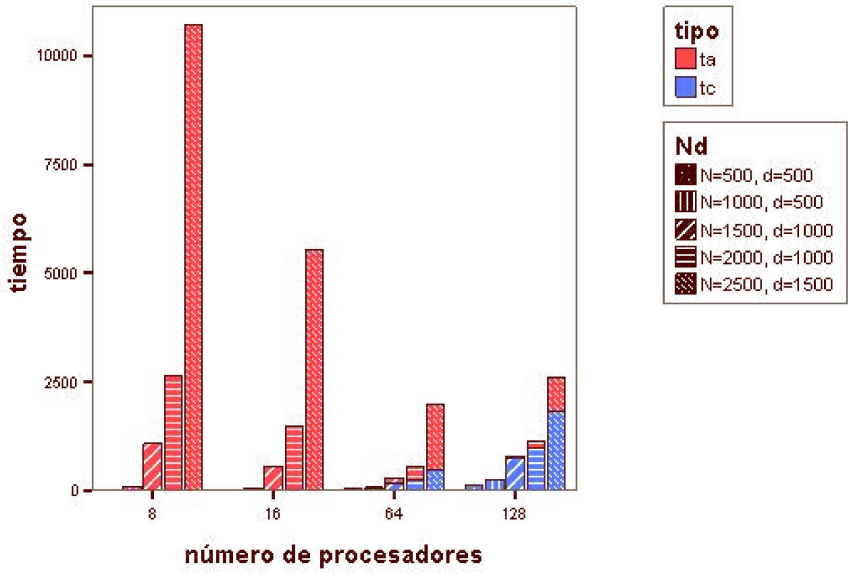

Figura 4.2: Gráfico comparativo de los tiempos de comunicación (tc) y de operaciones aritméticas (ta) del algoritmo $PMC2E_{bas}$ en Kefren, variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

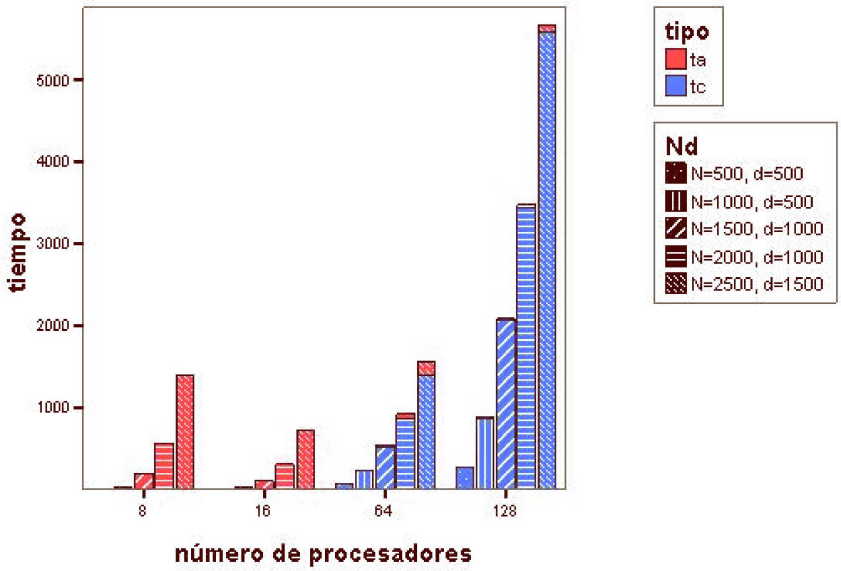

Figura 4.3: Gráfico comparativo de los tiempos de comunicación (tc) y de operaciones aritméticas (ta) del algoritmo PMC_{inv} en Kefren, variando el número de variables endógenas (N), el tamaño de la muestra (d) y el número de procesadores.

4.8. Conclusiones

Se han desarrollado versiones paralelas en memoria distribuida de los algoritmos mostrados en el capítulo anterior. Se han estudiado teóricamente los tiempos de las operaciones aritméticas y los tiempos de las comunicaciones, realizando comparaciones entres ellos.

Las implementaciones han sido desarrolladas en MPI, realizándose la paralelización a diferentes niveles. En las operaciones matriciales se han utilizado llamadas a PBLAS y ScaLAPACK, mientras que a niveles más altos, se han realizado operaciones al comienzo del algoritmo y distribuido los datos resultantes, de manera que la resolución de las ecuaciones del sistema se pueda hacer de forma independiente, con lo cual se consigue una paralelización a alto nivel, ya que todos los procesadores pueden estar resolviendo diferentes ecuaciones a la vez.

Finalmente, se han estudiado y comparado los tiempos de comunicación con los tiempos totales de cada algoritmo (tiempos de comunicación más operaciones aritméticas). Se concluye que existe en cada caso un número óptimo de procesadores a utilizar, dependiendo del algoritmo, el tamaño del problema y el hardware utilizado.

Capítulo 5

Algoritmos QR para la resolución de M.E.S.

En este capítulo se han desarrollado algoritmos para la resolución de Modelos de Ecuaciones Simultáneas mediante el estimador MC2E utilizando la descomposición QR expuesta en la sección 2.2.2. La razón de usar la descomposición QR en la expresión del estimador MC2E es que puede ser usado en cualquier tipo de ecuación (propiedad que no posee MCI).

Se han desarrollado algoritmos utilizando las reflexiones de Householder y los rotaciones de Givens, y se han comparado sus tiempos de ejecución. En la versión basada en rotaciones de Givens se ha reutilizado la descomposición QR utilizada para el cálculo de las variables *proxy* en la resolución de las ecuaciones, reduciéndose así considerablemente el tiempo de ejecución y consiguiéndose un algoritmo eficiente para M.E.S. con grandes tamaños de muestra (su tiempo de ejecución se ve mínimamente afectado por d).

La principal aportación del capítulo es la utilización de la descomposición QR (utilizada hasta ahora en muchas técnicas basadas en Mínimos Cuadrados) en Modelos de Ecuaciones Simultáneas. Hasta donde sabemos, no habían sido usados con anterioridad en la resolución de M.E.S. desarrollos basados en reflexiones de Householder y reflectores de Givens, aunque sí se han aplicado en otras técnicas (como la búsqueda del mejor modelo en Mínimos Cuadrados [31, 41], el cómputo de todos los modelos posibles en Mínimos Cuadrados [73], etc.)

5.1. Mínimos Cuadrados en dos Etapas mediante reflexiones de Householder

Utilizando la descomposición QR expuesta en la sección 2.2.2 se pueden rediseñar los cálculos en la expresión de MC2E para conseguir reducir el coste de operaciones y a la vez reducir la inestabilidad numérica. Esta descomposición ha sido usada con anterioridad en otras técnicas econométricas basadas (como éstas) en problemas de

mínimos cuadrados [29, 31, 73]. Desarrollamos ahora la misma idea en la resolución de Modelos de Ecuaciones Simultáneas.

Dada la matriz de variables exógenas X (de dimensión $d \times K$), existe una matriz ortogonal Q (de dimensión $d \times d$) y una matriz triangular R (de dimensión $d \times K$), tal que $X = QR$ o equivalentemente $Q^T X = R$. La matriz R tiene la forma $\begin{pmatrix} R_1 \\ 0 \end{pmatrix} \begin{matrix} K \times K \\ (d-K) \times K \end{matrix}$, donde R_1 es una matriz triangular superior.

Si usamos la descomposición QR en la expresión de la matriz de variables *proxy* $\hat{Y}$ dada en la ecuación 2.15, el estimador queda de la siguiente forma:

$$\begin{gathered} \hat{Y} = X(X^T X)^{-1} X^T Y = QR(R^T Q^T QR)^{-1} R^T Q^T Y = \\ QR\left([R_1^T|0]\begin{pmatrix} R_1 \\ 0 \end{pmatrix}\right)^{-1} [R_1^T|0] Q^T Y = Q\begin{pmatrix} R_1 \\ 0 \end{pmatrix}(R_1^{-1} R_1^{-T})[R_1^T|0] Q^T Y = \\ Q\begin{pmatrix} Id_K \\ 0 \end{pmatrix}[Id_K|0] Q^T Y = Q\begin{pmatrix} Id_K & 0 \\ 0 & 0 \end{pmatrix} Q^T Y \end{gathered} \tag{5.1}$$

Si en la expresión anterior llamamos $\tilde{Y}$ a la matriz resultante de multiplicar Q^T por Y, y si particionamos dicha matriz en dos tal que $\tilde{Y} = \begin{pmatrix} \tilde{Y}_1 \\ \tilde{Y}_2 \end{pmatrix}$ siendo $\tilde{Y}_1$ de dimensión $K \times N$, e $\tilde{Y}_2$ de dimensión $(d-K) \times N$, la expresión 5.1 queda:

$$\hat{Y} = Q\begin{pmatrix} Id_K & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \tilde{Y}_1 \\ \tilde{Y}_2 \end{pmatrix} = Q\begin{pmatrix} \tilde{Y}_1 \\ 0 \end{pmatrix} \tag{5.2}$$

Si se usa la descomposición QR en el algoritmo $MC2E_{bas}$ (algoritmo 3), el primer paso sería la descomposición QR de X y a continuación el cálculo de la matriz de variables *proxy*, $\hat{Y}$.

De la misma manera se puede usar la descomposición QR en la estimación de los coeficientes en cada una de las ecuaciones descritas en el algoritmo $MC2E_{bas}$ pero, en este caso, descomponiendo la matriz $X_{e_i} = Q_i R_i$, tal y como se muestra a continuación:

$$\begin{gathered} \hat{\beta}_i = (X_{e_i}^T X_{e_i})^{-1} X_{e_i}^T y_i = (R_i^T Q_i^T Q_i R_i)^{-1} R_i^T Q_i^T y_i = \\ \left([R_{i,1}^T|0]\begin{pmatrix} R_{i,1} \\ 0 \end{pmatrix}\right)^{-1} [R_{i,1}^T|0] Q_i^T y_i = (R_{i,1}^{-1} R_{i,1}^{-T})[R_{i,1}^T|0] Q_i^T y_i = \\ R_{i,1}^{-1}[Id_K|0] Q_i^T y_i = [R_{i,1}^{-1}|0] Q_i^T y_i = [R_{i,1}^{-1}|0]\tilde{y}_i = R_{i,1}^{-1}\tilde{y}_{i,1} \end{gathered} \tag{5.3}$$

siendo $\tilde{y}_{i,1}$ la submatriz de $\tilde{y}_i$ formada por las $n_i + k_i - 1$ primeras filas, donde $\tilde{y}_i$ es la matriz resultante de multiplicar $Q_i^T y_i$. $R_{i,1}$ es la submatriz de R_i formada por las k_i primeras filas.

En las ecuaciones 5.1 y 5.2 es necesaria la multiplicación de la matriz Q (o su transpuesta) por otra matriz. La construcción de dicha matriz a partir de los reflectores de Householder es costosa tanto en tiempo de computación como en necesidades de memoria, por lo que es recomendable evitarlo. En lugar de construir

la matriz Q y multiplicarla por otra matriz procederemos a aplicar directamente a esta última matriz los reflectores de Householder.

También hay que tener en cuenta que R_1 es una matriz triangular superior, y el coste a la hora de resolver el sistema de ecuaciones donde aparece la inversa de esta matriz es menor que en el caso de tener una matriz completa.

Tal y como se planteó al comienzo de la presente memoria se pretende hacer una aplicación lo más portable posible, y se usará siempre que sea posible librerías LAPACK y BLAS. En este caso pueden ser utilizadas tanto para la descomposición QR como en la resolución de un sistema cuya matriz de coeficientes es triangular. En los experimentos mostrados en 5.4 se han usado *dtrsv* para la resolver el sistema (inversa de R_1), *dgeqrf* para la descomposición QR y *dormqr* para aplicar los reflectores ya calculados a una matriz.

El coste de aplicar *dgeqrf* a la matriz X $(d \times K)$ es $\frac{2}{3}K^2(3d-K)$, el coste de aplicar *dorgqr* usando K reflectores a una matriz $d \times N$ es $2NK(2d-K)$, y el coste de aplicar *dtrsv* para resolver un sistema triangular superior de orden $n \times n$ sería n^2 [34].

El algoritmo 12 muestra el esquema de MC2E usando la descomposición QR. Tal y como se ha explicado anteriormente, en las líneas 2, 3 y 6 del algoritmo no se multiplica la matriz Q por la otra matriz, sino que se aplican los reflectores de Householder directamente. En el paso 3 del algoritmo no es necesario aplicar los reflectores a la matriz entera puesto que hay ceros por debajo de la fila K-ésima. En el paso 7 se resuelve un sistema de ecuaciones triangular superior. En las líneas 1 y 5 la descomposición QR usada es la descomposición de Householder descrita en la sección 2.2.2.

Algoritmo 12 Algoritmo $MC2E_H$

Entrada: $X \in \mathbb{R}^{d\times K}, Y \in \mathbb{R}^{d\times N}, B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$
Salida: $B \in \mathbb{R}^{N\times N}, \Gamma \in \mathbb{R}^{N\times K}$

1: Obtener Q y R (desc. QR de$X$) $\{\text{coste} \rightarrow \frac{4}{3}K^2(3d-K)\}$
2: $\tilde{Y} = Q^T Y$ $\{\text{coste} \rightarrow 2NK(2d-K)\}$
3: $\hat{Y} = Q \begin{pmatrix} \tilde{Y}_1 \\ 0 \end{pmatrix}$ siguiendo la ecuación 5.2 $\{\text{coste} \rightarrow 2NK^2\}$
4: **Para** i=1...N **Hacer**
5: Obtener Q_i y R_i (desc. QR de X_{e_i}) $\{\text{coste} \rightarrow \frac{4}{3}(n_i+k_i-1)^2(3d-(n_i+k_i-1))\}$
6: $\tilde{y}_i = Q_i^T y_i$ $\{\text{coste} \rightarrow 2(n_i+k_i-1)(2d-(n_i+k_i-1))\}$
7: $\beta_i = R_{i,1}^{-1}\tilde{y}_{i,1}$ $\{\text{coste} \rightarrow (n_i+k_i-1)^2\}$
8: Sustituir los valores de β en las incógnitas de B_i y Γ_i
9: **Fin Para**

En la ecuación i-ésima, la matriz X_{e_i}, formada por las variables exógenas y las variables *proxy* de la ecuación que se esté resolviendo, es la que se descompone en Q_i y R_i, e y_i es la endógena principal de dicha ecuación.

El coste total de $MC2E_H$ es:

$$T_{MC2E_H}(N,d,K,\Omega_n,\Omega_k) = \tfrac{4}{3}K^2(3d-K) + 4NKd + \sum_{i=1}^{N}\left(\tfrac{4}{3}(n_i+k_i-1)^2(3d-(n_i+k_i-1)) + 2(n_i+k_i-1)(2d-(n_i+k_i-1)) + (n_i+k_i-1)^2\right) \quad (5.4)$$

Se puede observar en la expresión 5.4 que el parámetro d tiene presencia en los términos previos al sumatorio y también en los de dentro del sumatorio. Esto se traduce en un aumento del coste computacional considerable cuando aumenta el tamaño de la muestra, por lo que $MC2E_H$ no es eficiente para un M.E.S. con tamaño de muestra grande.

5.2. Mínimos Cuadrados en dos Etapas mediante rotaciones de Givens

Se detecta una gran deficiencia en la versión presentada anteriormente de descomposición QR. Ésta reside en el no aprovechamiento de la descomposición QR de X en la descomposición en cada ecuación. También puede verse como deficiente el cálculo de la matriz $\hat{Y}$, puesto que exige aplicar dos veces los reflectores de Householder a la matriz Y.

La idea general del nuevo algoritmo consiste en aprovechar en cada ecuación que la matriz X ha sido ya triangularizada para, en lugar de usar Householder sobre la matriz X_{e_i}, aplicar rotaciones de Givens haciendo cero los elementos no nulos por debajo de la diagonal principal. Es decir, no tener que volver a usar Householder en cada ecuación y, en su lugar, obtener la descomposición QR a partir de la descomposición hecha de X para el cálculo de $\hat{Y}$.

Para resolver la ecuación i-ésima se construye la matriz X_{e_i}, la cual está formada por k_i columnas de X y $n_i - 1$ columnas de $\hat{Y}$ (que han sustituido a las variables endógenas de la ecuación salvo la principal), y a continuación se hace su descomposición QR con el correspondiente ahorro en el cálculo de MCO, aplicando lo expuesto en la ecuación 5.1 pero tomando en lugar de X, X_{e_i}, e y_i en lugar de Y. Es necesario por lo tanto construir la matriz $[X|\hat{Y}]$ a partir de la cual se obtendrán todas las X_{e_i}.

Sean Q y R tal que $Q^TX = R$, donde Q y R son calculadas mediante el método de Householder. Si aplicamos Q^T a $[X|\hat{Y}]$ se obtiene (usando la expresión de $\hat{Y}$ en la ecuación 5.2) $Q^T[X|\hat{Y}]=[Q^TX|Q^T\hat{Y}] = [R|Q^T\hat{Y}] = \begin{pmatrix} R_1 & \tilde{Y}_1 \\ 0 & 0 \end{pmatrix} \begin{matrix} K \\ d-K \end{matrix}$, con R_1 triangular superior.

Por lo tanto se tiene una matriz $[R_1|\tilde{Y}_1]$ formada por las primeras K filas de la matriz, donde aparecen los valores distintos de cero. El resto de la matriz no es relevante por ser todo valores nulos.

Esto tiene consecuencias importantes puesto que ya no es necesario calcular $\hat{Y}$, sino que basta con aplicar los vectores de Householder calculados en la descomposi-

ción QR de X a Y una sola vez, y a continuación quedarse con las K primeras filas de la matriz resultante.

Nos disponemos ahora a resolver cada una de las ecuaciones del modelo. Para resolver la ecuación i-ésima, la cual se puede expresar de la forma $y_i = X_i b_i + Y_i \Gamma_i + \epsilon_i$, definimos la matriz $X_{e_i} = [X_i|\hat{Y}_i]$, es decir la matriz formada por las columnas de las variables exógenas y las variables *proxy* que aparecen en la ecuación. La matriz se puede expresar de la forma $X_{e_i} = [X|\hat{Y}]S_i$ donde S_i (de dimensión $(K+N) \times (k_i + n_i - 1)$) es una matriz de selección formada por todo ceros salvo un 1 en cada columna, que estará en la fila correspondiente a la columna que se desee seleccionar. Con lo que S_i puede ser expresada como $(e_{\lambda_{i,1}}, ..., e_{\lambda_{i,n_i+k_i-1}})$, donde $e_{\lambda_{i,j}}$ es el $\lambda_{i,j}$-ésimo vector de la matriz identidad Id_{K+N}, $\forall j = 1, ..., n_i + k_i - 1$, siendo $\lambda_{i,j}$ la columna de $[X|\hat{Y}]$ correspondiente a la variable que entra en el j-ésimo lugar en la ecuación i-ésima (por lo que aparecerá en la j-ésima columna de X_{e_i}).

Si aplicamos la matriz Q^T a X_{e_i} se obtiene que $Q^T X_{e_i} = Q^T[X|\hat{Y}]S_i = [R|Q^T\hat{Y}]S_i$ $\begin{pmatrix} R_1 & \tilde{Y}_1 \\ 0 & 0 \end{pmatrix} S_i = \begin{pmatrix} R_{i,1} & \tilde{Y}_{i,1} \\ 0 & 0 \end{pmatrix}$, siendo $R_{i,1}$ e $\tilde{Y}_{i,1}$ las K primeras filas de la matriz resultante de la multiplicación. Hay que notar que $R_{i,1}$ tendrá k_i columnas de R_1 y que no necesariamente tiene por que ser triangular superior (solo ocurrirá esto en el caso en que dicha matriz esté formada por las k_i primeras columnas de R_1). La matriz $\tilde{Y}_{i,1}$ estará formada por columnas de $\tilde{Y}_1$ todas con K filas.

Es necesario obtener la descomposición QR de X_{e_i}, por lo que hay que encontrar una matriz ortogonal que multiplicada a la anterior dé como resultado una matriz triangular superior. Para esto utilizaremos las rotaciones de Givens descritas en el apartado 2.2.2.

Supongamos que queremos retriangularizar X_{e_i} cuya matriz de selección es $S_i = (e_{\lambda_{i,1}}, ..., e_{\lambda_{i,n_i+k_i-1}})$. El número de rotaciones de Givens necesarias para triangularizar $[R_{i,1}|\tilde{Y}_{i,1}]$ es $\sum_{j=1}^{k_i}(\lambda_{i,j} - j)$ para las primeras k_i columnas y $\sum_{j=1}^{n_i-1}(K - j - k_i)$ para el resto. La matriz ortogonal obtenida a partir de multiplicar rotaciones de Givens sería:

$$\tilde{Q}_i^T = \left(\prod_{n=1}^{n_i-1} \prod_{j=1}^{K-n} G^{(i)}_{K-j-k_i, K-j-k_i+1} \right) \left(\prod_{n=1}^{k_i} \prod_{j=1}^{\lambda_{i,n}-n} G^{(n)}_{\lambda_{i,n}-j, \lambda_{i,n}-j+1} \right) \tag{5.5}$$

El coste de la retriangularización de $[R_{i,1}|\tilde{Y}_{i,1}]$ es:

$$C_i(\lambda_i, k_i, n_i) = 6 \left(\sum_{j=1}^{k_i} (\lambda_{i,j} - j)(k_i + n_i - j) + \sum_{j=1}^{n_i-1} (K - j - k_i)(n_i - j) \right) \tag{5.6}$$

siendo $\lambda_i = (\lambda_{i,1}, ..., \lambda_{i,n_i+k_i-1})$.

La figura 5.1 muestra la secuencia de eliminación de elementos en el proceso de retriangularización mediante rotaciones de Givens de una matriz $X \in \mathbb{R}^{7\times5}$ siendo $\lambda_i =$

$(1, 3, 4, 6, 7)$. La entrada i (i=1,...,6) indica el elemento reducido a cero en la i-ésima rotación, multiplicando a la izquierda por la matriz $\tilde{Q}_i^T = G_{5,6}^{(5)} G_{5,7}^{(5)} G_{4,5}^{(4)} G_{4,6}^{(4)} G_{3,4}^{(3)} G_{2,3}^{(2)}$.

$$\begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & 1 & \bullet & \bullet & \bullet \\ & & 2 & \bullet & \bullet \\ & & & 4 & \bullet \\ & & & 3 & 6 \\ & & & & 5 \end{pmatrix}$$

Figura 5.1: Orden de anulación en la retriangularización de una matriz $X \in \mathbb{R}^{7\times 5}$.

La matriz $Q_i^T = \tilde{Q}_i^T Q^T$ es una matriz ortogonal tal que $Q_i^T X_{e_i} = R_i$ con $R_i = \begin{pmatrix} R_{i,1} \\ 0 \end{pmatrix}$, siendo $R_{i,1}$ triangular superior de dimensión $(n_i + k_i - 1) \times (n_i + k_i - 1)$.

Para que una ecuación pueda ser resuelta tiene que estar identificada y por lo tanto se tiene que dar que $n_i + k_i - 1 \leq K$ (condición de orden) o, lo que es lo mismo, el número de columnas que tomará una ecuación de $[R_1|\tilde{Y}_1]$ es a lo sumo K. Esto asegura que la matriz triangular resultante será invertible puesto que tendrá a lo sumo K filas (con más filas tendría obligatoriamente filas formadas por ceros y la matriz no sería invertible, no pudiendo ser resuelta la ecuación).

Para calcular el valor de $\hat{\beta}_i$ tal y como se ve en su expresión en la ecuación 5.1, es necesario resolver el sistema $\hat{\beta}_i = R_{i,1}^{-1} \tilde{y}_{i,1}$, siendo ${\tilde{y}_i}^T = [\tilde{y}_{i,1}^T, \tilde{y}_{i,2}^T]$ la matriz resultante de multiplicar Q_i^T por y_i, e $\tilde{y}_{i,1}$ la submatriz de $\tilde{y}_i$ formada por las $n_i + k_i - 1$ primeras filas.

Se tiene que $\tilde{y}_i = Q_i^T y_i = \tilde{Q}_i^T Q^T y_i$, y puesto que y_i es la columna i-ésima de la matriz Y, la matriz $Q^T y_i$ es la columna i-ésima de la matriz $Q^T Y$ (que denotamos $\tilde{Y}_i$) calculada anteriormente. Por lo tanto el cálculo de $\tilde{y}_i$ se reduce a multiplicar la columna i-ésima de $Q^T Y$ por $\tilde{Q}_i^T$ o, lo que es lo mismo, a aplicar las rotaciones de Givens dadas en la expresión 5.5 a $\tilde{Y}_i$.

El algoritmo 13 muestra el esquema de $MC2E_G$ (retriangularizando en cada ecuación mediante rotaciones de Givens). Tal y como se ha explicado anteriormente, en la línea 2 del algoritmo no se multiplica la matriz Q por la otra matriz sino que se aplican los reflectores de Householder directamente a esta última. En la línea 6 se aplican las rotaciones de Givens calculadas en la línea 5 a la i-ésima columna de $\tilde{Y}$ denotada por $\tilde{Y}_i$, que ha sido calculada en el paso 2. En la línea 7 se resuelve un sistema de ecuaciones triangular superior.

El coste total del algoritmo $MC2E_G$ es:

Algoritmo 13 Algoritmo $MC2E_G$

Entrada: $X \in \mathbb{R}^{d\times K}, Y \in \mathbb{R}^{d\times N}, B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$
Salida: $B \in \mathbb{R}^{N\times N}, \Gamma \in \mathbb{R}^{N\times K}$

1: Obtener Q y R (desc. QR de Householder de X) {coste $\rightarrow \frac{4}{3}K^2(3d-K)$}
2: $\tilde{Y} = Q^T Y$ {coste $\rightarrow 2NK(2d-K)$}
3: Crear la matriz $[R_1|\tilde{Y}_1]$
4: **Para** i=1...N **Hacer**
5: Retriangularizar la matriz $[R_{i,1}|\tilde{Y}_{i,1}]$ mediante las rotaciones de Givens dadas en la expresión 5.5 {coste $\rightarrow C_i(\lambda_i, k_i, n_i)$}
6: $\tilde{y}_i = \tilde{Q}_i^T \tilde{Y}_i$ {aplicando las rotaciones de Givens calculadas en el paso anterior} {coste $\rightarrow 6\left(\sum_{j=1}^{k_i}(\lambda_{i,j}-j) + \sum_{j=1}^{n_i-1}(K-j-k_i)\right)$}
7: $\hat{\beta}_i = R_{i,1}^{-1}\tilde{y}_{i,1}$ {coste $\rightarrow (n_i+k_i-1)^2$}
8: Sustituir los valores de β_i en las incógnitas de B_i y Γ_i
9: **Fin Para**

$$T_{MC2E_G}(N,d,K,\Omega_n,\Omega_k) = \tfrac{4}{3}K^2(3d-K) + 2NK(2d-K) + \sum_{i=1}^{N}\left(C_i(\lambda_i,k_i,n_i) + 6\left(\sum_{j=1}^{k_i}(\lambda_{i,j}-j) + \sum_{j=1}^{n_i-1}(K-j-k_i)\right) + (n_i+k_i-1)^2\right) \tag{5.7}$$

Se puede observar en la expresión que el parámetro d no tiene presencia dentro del sumatorio, por lo que se reduce considerablemente su influencia respecto a la expresión 5.4, haciendo de $MC2E_G$ un algoritmo eficiente para M.E.S. con tamaño de muestra grande.

5.3. Paralelización del algoritmo $MC2E_G$

Se presentan en esta sección la versión en paralelo del algoritmo 13 desarrollado en secuencial en la sección anterior. La paralelización está diseñada para memoria compartida aunque el cambio a memoria distribuida es casi inmediato.

No se han desarrollado una verión en paralelo del algoritmo $MC2E_H$ porque se entiende que será usado en su lugar $MC2E_G$ que mejora el tiempo de ejecución para cualquier tamaño del problema (ver tabla 5.1).

El algoritmo 14 muestra un esquema paralelo del algoritmo $MC2E_G$ para p procesadores con memoria compartida.

En las líneas 1 y 2 se pueden usar las funciones de LAPACK *dgeqrf* y *dorgqr* dado que están desarrolladas en numerosas versiones de LAPACK en paralelo (en memoria compartida). A continuación se hace un reparto cíclico de las ecuaciones entre los p procesadores. El hecho de hacerlo cíclico es, como se ha dicho en el capítulo anterior, porque en muchas ocasiones un sistema real se forma a partir de

Algoritmo 14 Algoritmo $PMC2E_G$

Entrada: $X \in \mathbb{R}^{d\times K}$, $Y \in \mathbb{R}^{d\times N}$, $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$
Salida: $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$

1: Obtener Q y R (desc. QR de Householder de X en paralelo) {coste $\rightarrow \frac{4K^2}{3p}(3d-K)$}
2: Calcular $\tilde{Y} = Q^T Y$ {Aplicar los reflectores de Householder en paralelo a la matriz} {coste $\rightarrow \frac{2NK}{p}(2d-K)$}
3: Crear la matriz $[R_1|\tilde{Y}_1]$
4: EN PARALELO cada procesador $q = 0, ..., p-1$ HACE
5: **Para** j=1...$\frac{N}{p}$ **Hacer**
6: $i = q + (j-1)p + 1$
7: Retriangularizar la matriz $[R_{i,1}|\tilde{Y}_{i,1}]$ mediante las rotaciones de Givens dadas en la ecuación 5.5 {coste $\rightarrow C_i(\lambda_i, k_i, n_i)$}
8: Calcular $\tilde{y}_i = \tilde{Q}_i^T \tilde{Y}_i$ {aplicando las rotaciones de Givens calculadas en el paso anterior} {coste $\rightarrow 6\left(\sum_{j=1}^{k_i}(\lambda_{i,j}-j) + \sum_{j=1}^{n_i-1}(K-j-k_i)\right)$}
9: Calcular $\hat{\beta}_i = R_{i,1}^{-1}\tilde{y}_{i,1}$ {coste $\rightarrow (n_i+k_i-1)^2$}
10: Sustituir los valores de β_i en las incógnitas de B_i y Γ_i
11: **Fin Para**
12: FIN PARALELO

varios más pequeños concentrando las ecuaciones más grandes, que se añaden para interrelacionar las existentes, de forma consecutiva.

El coste total del algoritmo $PMC2E_G$ es:

$$T_{PMC2E_G}(N,d,K,\Omega_n,\Omega_k,p) = \tfrac{4}{3}\tfrac{K^2}{p}(3d-K) + 2\tfrac{N}{p}K(2d-K) +$$
$$\max_{q=0...p-1}\left\{\sum_{s=1,\ i=q+(s-1)p+1}^{\frac{N}{p}}\left(C_i(\lambda_i,k_i,n_i) + 6\left(\sum_{j=1}^{k_i}(\lambda_{i,j}-j) + \sum_{j=1}^{n_i-1}(K-j-k_i)\right) + (n_i+k_i-1)^2\right)\right\} \approx$$
$$\tfrac{4}{3}\tfrac{K^2}{p}(3d-K) + 2\tfrac{N}{p}K(2d-K) +$$
$$\sum_{i=1}^{\frac{N}{p}}\left(C_i(\lambda_i,k_i,n_i) + 6\left(\sum_{j=1}^{k_i}(\lambda_{i,j}-j) + \sum_{j=1}^{n_i-1}(K-j-k_i)\right) + (n_i+k_i-1)^2\right) \tag{5.8}$$

Al igual que en el capítulo anterior, se ha considerado la siguiente aproximación: El tiempo total del algoritmo es el tiempo del último procesador que termine, por lo que es necesario tomar el máximo variando los procesadores $q = 0, ...p-1$. Sin embargo, si se hace una asignación suficientemente balanceada se asume que todos los procesadores acaban simultáneamente, por lo que se puede tomar el tiempo de uno de ellos como aproximación.

5.4. Estudio experimental

En esta sección se van a describir diversos experimentos mediante los cuales se estudian los algoritmos propuestos en los apartados anteriores. Los Modelos de Ecuaciones Simultáneas han sido generados de la misma forma que en el capítulo anterior.

Se pretenden estudiar experimentalmente los siguientes puntos:

- La influencia de los parámetros N, K y d en $MC2E_H$ y $MC2E_G$.
- Comparar los tiempos de ejecución de $MC2E_H$ y $MC2E_G$ para diferentes tamaños del problema.
- Comparación con los tiempos de ejecución de los algoritmos expuestos en capítulos anteriores.
- La diferencia entre las soluciones encontradas por $MC2E_{inv}$ y $MC2E_H$, para detectar inestabilidad numérica en los cálculos.

La tabla 5.1 muestra una comparativa entre los tiempos de ejecución de la resolución de un M.E.S. usando los algoritmos $MC2E_H$ y $MC2E_G$. Se han tomado para cada tamaño del problema 5 modelos que han sido resueltos por ambos algoritmos, anotando los tiempos de ejecución y el ratio. A continuación se han calculado los promedios y las desviaciones típicas tanto de los tiempos como de los ratios, mostrándolos en la tabla. El hecho de tomar promedios ha sido para evitar que el tiempo de ejecución quede sesgado por el M.E.S. generado aleatoriamente. A priori, puede parecer que el algoritmo $MC2E_G$ tiene una gran dependencia respecto al M.E.S. que se está resolviendo. Esto es debido a que el coste de las retriangularizaciones depende de las variables que se hayan tomado en cada ecuación. Sin embargo, una vez vistos los resultados, se puede concluir que la variabilidad es mínima y que tal dependencia no afecta considerablemente al tiempo total del algoritmo.

Se puede observar que el tiempo de resolución de un sistema utilizando retriangulaciones de Givens es siempre menor que el tiempo utilizado por el algoritmo que usa la descomposición de Householder en cada ecuación. Esta diferencia se hace especialmente efectiva cuando el tamaño de la muestra d es muy grande con respecto al número de endógenas y exógenas. El ratio mayor se da en el caso en que mayor diferencia hay entre N, K y d ($N = 100$, $K = 100$ y $d = 1000$) y el caso menor en el que menor diferencia hay ($N = 200$, $K = 400$ y $d = 500$). Puesto que la condición $K \leq d$ pone un límite a K, se podría concluir que el algoritmo $MC2E_G$ (algoritmo 13) es más eficiente en general que $MC2E_H$.

El algoritmo $MC2E_G$ no se ve afectado prácticamente por el aumento de d, pero su tiempo de ejecución aumenta considerablemente con el aumento de N y K. También es de subrayar la poca variabilidad a la que se ven afectados ambos algoritmos (y por supuesto el ratio). Esto índica que dichos algoritmos se ven afectados mínimamente por la estructura del sistema en cuanto a tiempo de ejecución, siendo

Tam. del problema			Tiempo	Tiempo	
N	K	d	Householder	Givens	Ratio
100	100	500	$0.50_{0.02}$	$0.15_{0.00}$	$3.35_{0.06}$
100	100	1000	$1.06_{0.03}$	$0.16_{0.00}$	$6.63_{0.21}$
100	200	500	$1.39_{0.06}$	$0.67_{0.02}$	$2.09_{0.09}$
100	200	1000	$3.23_{0.11}$	$0.70_{0.00}$	$4.60_{0.13}$
200	200	500	$3.39_{0.08}$	$1.90_{0.03}$	$1.79_{0.01}$
200	200	1000	$7.85_{0.32}$	$1.94_{0.05}$	$4.05_{0.07}$
200	400	500	$10.25_{0.14}$	$9.16_{0.13}$	$1.12_{0.02}$
200	400	1000	$23.28_{0.20}$	$9.31_{0.16}$	$2.50_{0.05}$
400	400	1000	$56.60_{2.47}$	$28.05_{0.41}$	$2.02_{0.07}$
400	400	1500	$98.85_{2.84}$	$28.62_{0.31}$	$3.45_{0.08}$
400	600	1000	$108.68_{3.94}$	$74.55_{1.18}$	$1.46_{0.03}$
400	600	1500	$200.76_{5.57}$	$75.13_{0.91}$	$2.67_{0.05}$
800	800	2000	$1319.30_{12.58}$	$438.14_{2.24}$	$3.01_{0.01}$
800	800	2500	$1889.01_{20.66}$	$440.61_{3.57}$	$4.29_{0.02}$
800	1000	2000	$2067.94_{16.62}$	$741.28_{4.60}$	$2.79_{0.01}$
800	1000	2500	$2894.90_{28.64}$	$741.77_{5.43}$	$3.90_{0.02}$

Tabla 5.1: Tiempo de ejecución (en segundos) y ratio de los algoritmos $MC2E_H$ y $MC2E_G$ en Rosebud, cuando se varía el número de variables endógenas (N), el número de variables exógenas (K) y el tamaño de la muestra (d). Se muestra el promedio (número grande) y la desviación típica (subíndice) de 5 medidas para el mismo tamaño del problema.

dicho tiempo muy similar para todos los sistemas con el mismo número de variables endógenas y exógenas y mismo tamaño de muestra.

La tabla 5.2 muestra una comparación entre los tiempos de ejecución de tres de los algoritmos desarrollados para el estimador MC2E en Kefren. Como se observa, la versión del algoritmo basada en la descomposición QR tiene menor tiempo de ejecución que $MC2E_{bas}$ pero no que $MC2E_{inv}$.

Se pretende comparar la precisión numérica de los algoritmos desarrollados en este capítulo con los del capítulo 3. Para ello se muestra en la tabla 5.3 el promedio de las diferencias, en valor absoluto, de las soluciones encontradas por $MC2E_{inv}$ para un M.E.S. de tamaño dado, y las encontradas por $MC2E_H$ para el mismo M.E.S. Se puede observar en la tabla que, al igual que ocurría en la tabla 5.3, los casos con mayor diferencia tienen siete decimales en común, coincidiendo la mayoría hasta en 11 decimales. Esto indica que, al contrario de lo que pueda parecer en un principio, no hay diferencias significativas entre resolver un sistema por $MC2E_{inv}$ o hacerlo por $MC2E_H$, desde el punto de vista de la precisión numérica. Tal y como se dijo en el capítulo anterior, las diferencias encontradas entre ambos algoritmos suelen ser muy inferiores a aquellas dadas por el error cometido en la toma de la

N:	500	1000	1500	2000	2500
d:	500	500	1000	1000	1500
Alg.	Tiempo	Tiempo	Tiempo	Tiempo	Tiempo
$MC2E_{bas}$	72.82	790.93	7031.96	19337.92	97874.21
$MC2E_{inv}$	12.91	198.16	1225.33	4192.10	10217.02
$MC2E_H$	26.53	257.10	2363.64	5457.78	16825.37
			Ratio		
$MC2E_{bas}/MC2E_{inv}$	5.64	3.99	5.74	4.61	9.58
$MC2E_H/MC2E_{inv}$	2.05	1.30	1.93	1.30	1.65

Tabla 5.2: Tiempo de ejecución (en segundos) y ratio comparativo entre el algoritmo $MC2E_{bas}$ (algoritmo 3), $MC2E_{inv}$ (algoritmo 5) y $MC2E_H$ (algoritmo 12) en Kefren, cuando varía el número de variables endógenas (N) y el tamaño muestral (d), siendo el número de variables exógenas (K) el 40 % de N.

muestra (al no cumplir con exactitud el muestreo programado por el estadístico).

N	K	d	diferencia
500	200	500	$9.13657 \cdot 10^{-12}$
1000	400	500	$3.00996 \cdot 10^{-12}$
1000	400	1000	$4.65709 \cdot 10^{-13}$
1500	600	1000	$2.13886 \cdot 10^{-8}$
1500	600	1500	$2.63000 \cdot 10^{-9}$
2000	800	1500	$7.81023 \cdot 10^{-12}$
2000	800	2000	$2.78960 \cdot 10^{-12}$

Tabla 5.3: Promedio de las diferencias entre las soluciones calculadas por $MC2E_{inv}$ y las calculadas por $MC2E_H$ para un Modelo de Ecuaciones Simultáneas.

La tabla 5.4 muestra los tiempos de ejecución (en segundos) y el speed-up del algoritmo $PMC2E_G$. Puesto que la paralelización del algoritmo es relativamente sencilla por la independencia que se tiene entre ecuaciones, los speed-up resultantes son bastante altos incluso para tamaños del problema pequeños.

5.5. Conclusiones

Se han desarrollado algoritmos del estimador MC2E utilizando las reflexiones de Householder y las rotaciones de Givens. Se han comparado los tiempos de ejecución tanto teóricamente como experimentalmente. En la comparación de ambos algoritmos, se observa que el algoritmo basado en rotaciones de Givens tiene un coste menor que el basado en las reflexiones de Householder. Además, también se

N	K	d	1 proc.	2 proc.	Sp	4 proc.	Sp	8 proc.	Sp
400	400	1000	27.87	14.22	1.96	7.18	3.88	3.71	7.51
400	400	1500	28.03	14.33	1.96	7.25	3.87	3.74	7.49
400	600	1000	75.82	37.90	2.00	19.36	3.92	10.01	7.57
400	600	1500	76.40	38.01	2.01	19.48	3.92	10.08	7.58
800	800	2000	435.11	218.16	1.99	112.75	3.86	57.42	7.58
800	800	2500	439.88	220.03	2.00	113.72	3.87	58.12	7.57
800	1000	2000	741.75	370.05	2.00	188.08	3.94	96.36	7.70
800	1000	2500	736.48	368.26	2.00	187.07	3.94	95.62	7.70
1000	1000	2500	1058.59	531.54	1.99	266.22	3.98	135.76	7.80
1000	1200	3000	1634.33	816.77	2.00	418.91	3.90	210.10	7.78
1200	1200	2500	2191.15	1095.92	2.00	552.06	3.97	281.44	7.79
1200	1200	3000	2176.38	1092.76	1.99	550.75	3.95	279.97	7.77

Tabla 5.4: Tiempos de ejecución (en segundos) en Rosebud y Speed-up correspondiente al algoritmo $PMC2E_G$, cuando se varía el número de variables endógenas (N), el número de variables exógenas (K) y el tamaño de la muestra (d).

han comparado el tiempo de ejecución y la estabilidad numérica de los algoritmos de este capítulo con los algoritmos desarrollados en capítulos anteriores. Se concluye que $MC2E_{inv}$ tiene tiempos de ejecución inferiores a los de $MC2E_H$, pero no que los de $MC2E_G$. En cuanto a la estabilidad numérica, se obtienen valores idénticos por debajo del séptimo dígito (en el peor caso) de las soluciones calculadas por los algoritmos descritos en este capítulo y los de capítulos anteriores. Por lo tanto, no hay diferencia significativa en la estabilidad numérica de los distintos algoritmos teniendo en cuenta el campo en el que se usan los M.E.S.

Con $MC2E_G$ se obtiene un nuevo algoritmo para el estimador MC2E que reduce considerablemente el tiempo de ejecución (incluso menor que $MC2E_{inv}$ en muchas ocasiones), tiene la estabilidad numérica similar a la de los algoritmos basados en la QR (se evitan inversas, multiplicaciones grandes, etc.), y además con su versión paralela se obtienen speed-ups satisfactorios.

Capítulo 6

Algoritmos QR con nodos ficticios para la resolución de M.E.S.

En el capítulo anterior se ha explotado la propiedad de que la matriz de datos de cada ecuación está formada por columnas de la matriz X (de la cual ha sido calculada su descomposición QR) y la matriz Y. En lugar de volver a realizar la descomposición QR en cada ecuación se retriangulariza usando rotaciones de Givens, siendo el ahorro computacional muy notable. Para reducir aún más el número de operaciones, y por lo tanto el tiempo de ejecución, tendremos en cuenta dos considerariones.

La primera es la de no resolver las ecuaciones en cualquier orden, puesto que se podría dar el caso de que una ecuación contuviera todas las variables de otra y se pudiera retriangularizar a partir de la matriz asociada a la primera (y por lo tanto aprovechar los cálculos hechos para ella) obteniéndose la matriz asociada a la segunda. Sin embargo, se verá en este capítulo que solo en muy raras ocasiones se puede dar en un M.E.S. una ecuación identificada, y por lo tanto que se pueda resolver, cuyas variables, salvo su endógena principal, estén todas incluidas en otra ecuación.

La segunda consideración consiste en observar que la retriangularización expuesta en el capítulo anterior tiene mucho más coste cuanto más diferentes son las matrices correspondientes a cada ecuación, por lo que la creación de matrices intermedias podría reducir el número de retriangularizaciones a usar para hallar la descomposición QR en cada ecuación. En este capítulo, se explota esta idea mediante la creación de un árbol cuyos nodos corresponden a conjuntos de variables. En unos casos corresponderán a las ecuaciones del M.E.S. y en otros casos (los nodos llamados ficticios) no corresponden a ninguna ecuación del M.E.S. pero contribuyen a reducir considerablemente el número de rotaciones de Givens a utilizar para obtener las descomposiciones QR en cada ecuación.

Puesto que la finalidad de la construcción del árbol no es otra que la de recortar la computación requerida a la hora de resolver las ecuaciones del sistema, es importante que el tiempo de obtención del árbol más el tiempo de la resolución de las ecuaciones utilizando el árbol no sea superior al tiempo de resolver las ecuaciones

retriangularizando directamente. Puesto que el número de posibles nodos en el árbol podría ser muy grande y el número de caminos o ejes entre ellos también, se hace impracticable el uso de algoritmos exhaustivos, siendo la solución propuesta en su lugar un algoritmo heurístico.

La principal aportación del capítulo es la utilización de nodos ficticios que permitan reducir el número de rotaciones de Givens en la descomposición QR de cada ecuación. Esta idea ha sido utilizada en la generación y resolución de un subconjunto de modelos de todos los posibles modelos en Mínimos Cuadrados a partir de un conjunto de variables [73]. Hasta donde sabemos, no se han usado nodos ficticios y árboles de mínimo coste en la resolución de M.E.S.

6.1. Orden de resolución de las ecuaciones de un Modelo de Ecuaciones Simultáneas

Tal y como se ha dicho al comienzo del capítulo, lo idóneo sería resolver las ecuaciones en un orden de forma que si la ecuación i-ésima tiene todas sus variables incluidas en la ecuación j-ésima, se resolviera primero la segunda y después la primera. Esto permitiría obtener la descomposición QR de la matriz asociada a la ecuación i-ésima a partir de la descomposición de la matriz asociada a la ecuación j-ésima, ahorrando en el número de rotaciones de Givens respecto a retriangularizar desde la descomposición QR de la matriz $[X|\hat{Y}]$.

Sin embargo, el orden de resolución es prácticamente indiferente puesto que solo en muy raras ocasiones se puede dar en un M.E.S. una ecuación identificada cuyas variables, salvo su endógena principal, estén todas incluidas en otra ecuación.

De hecho no se puede dar una ecuación identificada cuyas variables estén todas (endógena principal incluida) en otra ecuación. Esta propiedad se puede deducir de la condición de rango (sección 2.3.2), la cual dice que una ecuación de un M.E.S. con N variables endógenas está identificada sí y solo sí existe al menos una matriz de dimensión $(N-1)\times(N-1)$ de coeficientes de variables excluidas en dicha ecuación con determinante distinto de cero. Supongamos que la ecuación i-ésima tiene todas sus variables en la ecuación j-ésima. Cuando se construye la matriz de tamaño $(N-1)\times(N-1)$ para comprobar la condición de rango en la j-ésima ecuación, la fila correspondiente a la ecuación i-ésima contiene todo ceros (puesto que hay que poner solo coeficientes excluidos) por lo que el determinante es cero.

Por lo tanto, la única posibilidad de que se dé una ecuación con todas las variables explicativas en otra es que la endógena principal no esté contenida en ella. Un ejemplo podría ser el siguiente:

$$\begin{aligned} y_1 &= \beta_{1,3}y_3 + \gamma_{1,2}x_2 + \gamma_{1,3}x_3 + u_1 \\ y_2 &= \beta_{2,3}y_3 + \gamma_{2,3}x_3 + u_2 \\ y_3 &= \beta_{3,1}y_1 + \gamma_{3,1}x_1 + u_3 \end{aligned} \tag{6.1}$$

donde la ecuación 2 tiene todas sus variables explicativas en la ecuación 1, y por

lo tanto si resolvemos primero la ecuación 1 se puede retriangularizar su matriz llegando a la ecuación 2 con menos coste. El problema es que estas ecuaciones no son comunes puesto que tiene que darse la propiedad de que no estando la endógena en la otra ecuación sí lo estén todas las variables explicativas (situadas a la derecha de la igualdad) lo que no tiene mucho sentido estadístico. En el ejemplo se incluyen las variables y_3 y x_3 en la primera ecuación, y por lo tanto se admite que ambas influyen en y_1. Sin embargo no se incluye y_2, admitiendo en este caso que y_2 no influye en y_1. Luego resulta que hay una ecuación que dice que y_2 se puede expresar en función de y_3 y x_3 más un error. No obstante, si se diera el caso de una ecuación de este tipo es claro que la reducción en el coste computacional en ella sería grande.

6.2. Árbol de Mínimo Coste

Tal y como se ha explicando al comienzo del capítulo, surge el problema de obtener el árbol de mínimo coste que permita obtener la descomposición QR de la matriz asociada a cada ecuación usando matrices intermedias. Dichas matrices (tanto las asociadas a las ecuaciones como las intermedias) formarán los nodos del árbol. En esta sección se formaliza el problema y se desarrollan las reglas necesarias para construir un algoritmo que obtenga un árbol que se aproxime al de mínimo coste.

Definición *Árbol asociado a un M.E.S.*.
Definimos el árbol asociado a un M.E.S. como un conjunto de nodos y aristas donde los nodos representan conjuntos de variables del sistema tanto exógenas como endógenas. Las aristas representan una relación de contenido o subconjunto de variables. Si existe una arista de un nodo N_i a otro nodo N_j indica que todas las variables de N_j se encuentran en N_i y, por lo tanto, se puede llegar de este último nodo al primero y obtener los datos de sus variables.□

Un nodo tiene asociada una matriz formada por un conjunto de columnas de la matriz $R = \begin{pmatrix} R_1 & \tilde{Y}_1 \\ 0 & 0 \end{pmatrix} \begin{matrix} K \\ d-K \end{matrix}$ definida en el capítulo anterior (y que será la matriz asociada al nodo raíz o nodo cero) retriangularizadas. Es decir, se calcula la descomposición QR de la matriz resultante de seleccionar las columnas correspondientes a las variables del nodo, y la R resultante es la matriz asociada a dicho nodo.

Sea M_i la matriz asociada al nodo N_i, el cual suponemos que tiene q variables. Tenemos que:

- Si $q \leq K$ entonces M_i es triangular superior de dimensión $d \times q$.
- Si $K \leq q \leq K+N$, entonces $M_i = \begin{pmatrix} R_{i,1} & \tilde{Y}_{i,1} \\ 0 & 0 \end{pmatrix} \begin{matrix} K \\ d-K \end{matrix}$ siendo $R_{i,1} \in \mathbb{R}^{K \times K}$ triangular superior y $\tilde{Y}_{i,1} \in \mathbb{R}^{K \times (q-K)}$.

El pasar de un nodo a otro en el árbol a través de las aristas existentes da como resultado en las matrices asociadas la eliminación de las columnas del primer nodo que no están en el segundo, y por lo tanto es necesaria la retriangularización hasta conseguir la matriz asociada al nodo. La retriangularización se realizará usando rotaciones de Givens, y el número de operaciones a realizar depende en gran medida del camino por el que se llegue al nodo. Se realizarán menos operaciones cuantas menos columnas hayan sido eliminadas del nodo origen al nodo destino o, dicho de otra forma, cuanto más se parezcan ambos. Puesto que tendremos varios caminos para llegar al mismo nodo surge el problema de decidir cual de ellos utilizar (el que menos coste en número de operaciones tenga) y por lo tanto la eliminación de los otros, por lo que de todos los posibles grafos resultará un árbol cuyo coste sea el mínimo posible.

A los nodos correspondientes a las ecuaciones los llamaremos nodos ecuación y a los nodos que no correspondan a ninguna ecuación los llamaremos *nodos ficticios*. Supondremos que en total (sumando nodos ficticios y nodos ecuación) hay G nodos más el raíz.

Representaremos las variables de un nodo por su número si son variables exógenas y por su número con gorro si son endógenas.

Ejemplo N_i=[1 2 5 $\hat{2}$ $\hat{4}$] es un nodo asociado a la matriz cuyas columnas corresponden a las variables exógenas 1, 2 y 5 y a las variables endógenas 2 y 4, una vez que han sido sustituidas por las variables *proxy*.

Definición Sea v una variable, decimos que $v \in N_i$ si la matriz asociada a N_i tiene a v como una de sus columnas.□

Definición *Nivel del nodo.*
Decimos que q es el nivel de un nodo N_i, o que $N_i \in L_q$, si el número de variables o columnas de la matriz asociada a N_i es q.□

Nota No existe N_i con $i = 0, ..., G$ tal que $N_i \in L_1$.

El nodo raíz del árbol tendrá asociada una matriz con todas las variables posibles y su nivel será $N + K$, es decir, $M_0 = \begin{pmatrix} R_1 & \tilde{Y}_1 \\ 0 & 0 \end{pmatrix} \begin{matrix} K \\ d-K \end{matrix} \in L_{N+K}$.

En general, en un nivel q puede existir un número de nodos $C_q^{N+K} = \frac{(N+K)!}{q!(N+K-q)!}$, y por lo tanto en total en el árbol pueden existir un máximo de $|V|_{max} = \sum_{i=1}^{N+K} C_i^{N+K}$ $= \sum_{i=0}^{N+K} C_i^{N+K} - 1 = 2^{N+K} - 1$ nodos.

Desde el nivel q el número máximo de aristas que se pueden construir desde nodos de ese nivel a nodos inferiores es $C_q^{N+K}(2^{N+K-q} - 2)$, y por lo tanto en total en el árbol pueden existir un máximo de $|A|_{max} = \sum_{i=0}^{N+K-2} C_i^{N+K}(2^{N+K-i} - 2) =$ $3^{N+K} - 2^{N+K+1} + 1$ aristas.

Este último cálculo está basado en la igualdad $\sum_{i=0}^{n} C_i^n C_{k-i}^{n-i} t^i = C_k^n (1+t)^k$ para cualesquiera n, k y t números enteros (que se puede encontrar en la página 76 de [28]). Tomando $k = n$ y teniendo presente que $C_i^n = C_{n-i}^n$, se obtiene la propiedad de que $\sum_{i=0}^{n} C_{n-i}^n t^i = (1+t)^n$, y llamando $j = n - i$ se obtiene que $\sum_{j=0}^{n} C_j^n t^{n-j} = (1+t)^n$, que es usada en el cálculo de la expresión anterior.

Definición *Distancia Asimétrica.*
Definimos la distancia asimétrica de un nodo N_i a otro N_j ($d_i(N_j)$) como el número de variables que están en N_i y no están en N_j:

$$d_i(N_j) = |\{x_k \in N_i \; con \; k = 1, ..., q \; y \; x_k \notin N_j \; siendo \; q \; / \; N_i \in L_q\}| \; .\square$$

Nota $d_0(N_i) = 0, \; \forall \; i = 1, ..., G.$

Nota Si existe un arco de N_j a N_i entonces $d_i(N_j) = 0$.

Definición *Distancia Total.*
Definimos la distancia total (d_{ij}) de dos nodos como la suma de las distancias asimétricas entre ellos: $d_{ji} = d_{ij} = d_i(N_j) + d_j(N_i).\square$

Definición *Coste de un arco.*
El coste del arco $a_{i,j}$ que va desde el nodo N_i al nodo N_j se representa por $C_{i,j}$, y es el número de operaciones necesarias para conseguir la matriz asociada al nodo N_j a partir de la matriz asociada al nodo N_i mediante retriangulaciones de Givens (ver sección 2.2.2).$\square$

El coste de un arco desde el nodo raíz hasta un nodo N_i viene dado por la siguiente expresión, siendo n_i en número de variables endogenas y k_i el número de exógenas de la ecuación i-ésima:

$$C_{0,i} = \sum_{j=1, \; m_j \in N_i}^{k_i} (m_j - j)(n_i + k_i - j + 1) + \sum_{j=k_i+1}^{n_i+k_i} (K - j)(n_i + k_i - j + 1) \quad (6.2)$$

siendo $m_j \in N_i$ la variable j-ésima del nodo N_i (cuya posición en el nodo raíz sería m_j).

Para triangularizar las columnas tomadas de la submatriz R_1 de M_0 (primer sumatorio), se necesitan $m_j - j$ eliminaciones para la columna j-ésima. En cada una de ellas se utiliza una rotación de Givens que actualiza dicho elemento y los $n_i + k_i - j + 1$ que hay hasta el final de la matriz. Para triangularizar las columnas tomadas de la submatriz $\tilde{Y}_1$ de M_0 (segundo sumatorio), se necesitan siempre $K - j$ eliminaciones puesto que $\tilde{Y}_1$ tiene K filas, y el número de elementos hasta el final son $n_i + k_i - j + 1$, puesto que j toma valores a partir de $k_i + 1$.

Y en el caso general, el coste de un arco desde el nodo N_s hasta un nodo N_i viene dado por la expresión:

$$\begin{aligned} C_{s,i} &= \sum_{j=1,\ m_j \in N_i}^{k_i+n_i} (L(m_j, N_s) - j)(n_i + k_i - j + 1) \ si\ K > L(m_{k_i+n_i}, N_s) \\ C_{s,i} &= \sum_{j=1,\ m_j \in N_i}^{j_c} (L(m_j, N_s) - j)(n_i + k_i - j + 1) + \sum_{j=j_c+1}^{k_i+n_i} (K - j)(n_i + k_i - j + 1) \\ & si\ K \leq L(m_{k_i+n_i}, N_s) \end{aligned} \tag{6.3}$$

donde j_c es el valor para el que $L(m_{j_c}, N_s) = K$, donde $L(m_j, N_s)$ es el lugar que ocupa m_j en N_s.

En el cálculo de 6.3 se utiliza el mismo razonamiento que para la expresión 6.2, teniendo en cuenta que m_j, variable que ocupa la columna j-ésima en M_i y que por lo tanto debe ser triangularizada hasta tener ceros por debajo del elemento j-ésimo, ya no procede de una matriz donde ocupaba la columna m_j-ésima (como ocurría en M_0), sino que en M_s ocupa la posición $L(m_j, N_s)$. En este caso la variable m_j tendrá $L(m_j, N_s)$ valores distintos de cero y el número de elementos a eliminar será $L(m_j, N_s) - j$, y en cada uno de ellos habrá que actualizar hasta el final de la matriz.

Puede ocurrir que M_s sea una matriz triangular, o que la última variable de N_i, tome una posición en N_s menor que K. En ambos casos (el primero es un caso particular del segundo) las variables a triangularizar tendrán un número de elementos distintos de cero igual a su posición, y no sería necesario eliminar elementos desde la fila K (como ocurría en $C_{0,i}$). Si por el contrario hubiera un valor de j (que llamaremos j_c) a partir del cual las variables que se incluyen en N_i tienen posiciones en N_s superiores a K, sería necesario incluir un segundo sumatorio que contara las operaciones de eliminar desde el elemento K hasta el j (como se observa en la segunda expresión para $C_{s,i}$).

Nota Si existe $a_{s,i}$, entonces $m_j \geq L(m_j, N_s) \geq j$. Esto es debido a que la posición de la variable j-ésima en un nodo N_i, es menor o igual que la que ocupa en el nodo N_s, la cual es a su vez menor que la que ocupa en el nodo raíz.

Definición *Árbol de Mínimo Coste.*
Definimos el Árbol de Mínimo Coste (AMC) de un M.E.S. como el árbol, de entre todos los asociados al M.E.S., que minimiza la suma total de los costes de todos los arcos del árbol.□

6.3. Construcción del Árbol de Mínimo Coste

En un principio se debe formar un árbol colocando los nodos correspondientes a cada una de las ecuaciones y el nodo raíz, y añadir los arcos que unen este último

nodo con el resto. Cualquier árbol que se plantee con nodos ficticios tiene que tener los nodos del árbol descrito dentro de él. De esta forma lo único que variará serán los nodos ficticios añadidos y los arcos entre dichos nodos y los nodos ya existentes. El fin será que la suma del coste total de dichos arcos sea lo menor posible.

Un algoritmo que obtenga el mínimo absoluto pasa por un alto coste computacional. Sería necesario, por cada combinación de nodos ficticios posibles, encontrar el árbol de mínimo coste y el AMC sería el que minimizara el coste de todos los encontrados para todas las combinaciones posibles de nodos ficticios. El alto número de nodos ficticios y de arcos posibles hace que dicho algoritmo no sea eficiente. También hay que subrayar que lo que se busca es una forma de resolver las ecuaciones que reduzca el coste de retriangularizar desde el nodo cero a todas ellas, y si el coste de encontrar dicho algoritmo es superior a resolver dichas ecuaciones directamente, no tendría sentido el aplicarlo.

Por lo tanto, se propone un algoritmo heurístico basado en una serie de reglas que se detallan a continuación, mediante el cual se pretende, si no encontrar el mínimo global, encontrar una solución que esté suficientemente cerca del mínimo y con la que se reduzca, al menos en ciertos problemas, el tiempo de computación.

Primero se detallarán y justificarán las reglas heurísticas que serán utilizadas en el algoritmo, y seguidamente se explicará el algoritmo y como se usan las reglas en él.

Regla 1 Sean N_1, N_2 y N_3 tres nodos tal que $N_1 \in L_{p_1}$, $N_2 \in L_{p_2}$ y $N_3 \in L_{p_3}$. Supongamos que existen los arcos $a_{1,3}$ y $a_{2,3}$. Si existierá $a_{1,2}$ se podría eliminar $a_{1,3}$ puesto que siempre ocurrirá que $C_{2,3} \leq C_{1,3}$ y el camino para ir a N_3 sería a través de N_2.

Ejemplo En un sistema con $N = 6$ y $K = 8$, si se tienen los nodos $N_1 = (2\ 3\ 7\ \hat{4}\ \hat{5})$ y $N_2 = (3\ 4\ 7\ \hat{5}\ \hat{6})$, el coste de retriangularizar desde N_0 es de 13 para N_1, es decir, $C_{0,1} = 13$, y 15 para N_2 ($C_{0,2} = 15$). En total el coste de ambos es de 15+13=28. Si creamos el nodo $N_3 = (2\ 3\ 4\ 7\ \hat{4}\ \hat{5}\ \hat{6})$ los costes son $C_{0,3} = 12$, $C_{3,1} = 3$ y $C_{3,2} = 7$, lo que da un total de 22 con lo que añadiendo el nodo ficticio N_3 y aplicando la regla 1 dos veces, una para los nodos N_0, N_1 y N_3 y otra para los nodos N_0,N_2 y N_3 (donde en ambos casos N_0 hace de nodo N_1 en la regla, N_3 de N_2, y el nodo restante de nodo N_2), se eliminan los caminos $a_{0,1}$ y $a_{0,2}$, y se ahorran 6 operaciones.

Regla 2 Dados dos nodos N_i y N_j, de todos los nodos ficticios N_k tal que $d_k(N_i) = d_k(N_j) = 0$ siempre se creará el que minimiza la suma de las distancias asimétricas: $\min_k\{d_i(N_k) + d_j(N_k)\}$.

La regla 2 dice que el nodo ficticio que, una vez dados dos nodos, más se parece a ambos (es decir el que más variables comunes tiene) es el nodo a tomar como camino intermedio para ir a ambos. Con esta regla se intenta que la retriangularización sea lo menos costosa posible por parecerse el nodo origen (que es el ficticio que se crea) a los dos de destino (que pueden ser nodos ficticios o ecuaciones). Pero los nodos

ficticios pueden variar según el orden de nodos dos a dos que se tomen, por lo que lo lógico es comenzar por aquellos que se parezcan más y por lo tanto la retriangulación desde el nodo nuevo que se cree sea menor.

En el ejemplo anterior, N_3 es el nodo (de todos para los que ocurre que $d_1(N_3) = d_2(N_3) = 0$) que minimiza la suma de las distancias, $d_3(N_1) + d_3(N_2)$. Esto hace que sea el más cercano a N_1 y N_2 a la vez, y que por lo tanto sea menor el número de operaciones necesarias para la retriangularización desde él.

Nota Dados $N_i \in L_q$ y $N_j \in L_{q'}$ tal que $d_i(N_j) = r$ y $d_j(N_i) = r'$, el nodo ficticio N_k creado según la regla 2 pertenecerá a $L_{q+r'} = L_{q'+r}$, es decir, N_k subirá r' niveles respecto a L_q y r niveles respecto a $L_{q'}$.

Regla 3 Para crear un nodo ficticio se seleccionan aquellos dos nodos cuya distancia total sea mínima.

En realidad esto nos dará un orden en el algoritmo de actuación, buscando siempre a la hora de crear un nuevo nodo ficticio los nodos con menor distancia total. También implica un coste de inicialización alto (orden cúbico) de comparaciones puesto que es necesario calcular las distancias entre todos los nodos, teniendo que comparar en cada cálculo las variables de los dos nodos. Sin embargo, se ha comprobado experimentalmente que, salvo para tamaños del problema pequeños, este tiempo no es significativo en comparación con el tiempo del resto del algoritmo.

Regla 4 Si existe un nodo ficticio N_i con un solo arco de salida este puede ser eliminado (puesto que no representa mejora ninguna) junto con dicho arco creando un nuevo arco desde el nodo padre al hijo.

Ejemplo Veamos con este ejemplo como la regla 4 da lugar a una reducción en el número de nodos ficticios usados. Denotaremos un nodo ficticio cuyos descendientes sean N_i y N_j por $NF_{i,j}$ En el mismo sistema anterior con $N = 6$ y $K = 8$, si se tienen los nodos $N_1 = (2\ 3\ 4\ 7\ \hat{4}\ \hat{5}\ \hat{6}) \in L_7$, $N_2 = (2\ 3\ 4\ 6\ \hat{4}\ \hat{5}\ \hat{6}) \in L_7$ y $N_3 = (2\ 3\ 4\ 8\ \hat{4}\ \hat{5}\ \hat{6}) \in L_7$, ocurre que $d_1(N_2) = d_1(N_3) = d_2(N_1) = d_3(N_1) = d_2(N_3) = d_3(N_2) = 1$. Como todas las distancias son iguales podemos crear cualquier nodo ficticio. Los nodos ficticios que se pueden crear son $NF_{1,2} = (2\ 3\ 4\ 6\ 7\ \hat{4}\ \hat{5}\ \hat{6}) \in L_8$, $NF_{1,3} = (2\ 3\ 4\ 7\ 8\ \hat{4}\ \hat{5}\ \hat{6}) \in L_8$ y $NF_{2,3} = (2\ 3\ 4\ 6\ 8\ \hat{4}\ \hat{5}\ \hat{6}) \in L_8$. Por ejemplo, a N_2 se puede llegar tanto desde $NF_{1,2}$ (cuyo coste es de 10) como desde $NF_{2,3}$ (cuyo coste es de 11). Como el coste de llegar a N_2 desde ambos es 3, se crea $NF_{1,2}$ ($NF_{2,3}$ no se crea siguiendo la regla 4). Lo mismo se hace con $NF_{1,3}$ cuyo coste es 12, siendo los costes desde él a N_1 y N_3 3 y 4, respectivamente. Por lo tanto, a N_1 se irá desde $NF_{1,2}$, y $NF_{1,3}$ no se crea por la regla 4.

Para la siguiente regla se necesitan dos definiciones adicionales:

Definición $N_i \in L_q$ está más a la izquierda que $N_j \in L_q$ si existe $1 \leq j_0 \leq q$ tal que $x_k = x'_k$, $\forall\ k = 1, ..., j_0 - 1$ y $x_{j_0} < x'_{j_0}$, siendo $x_k \in N_i$ y $x'_k \in N_j\ \forall\ k = 1, ..., j_0$.□

Definición Definimos los descendientes de un nodo N_i como el conjunto $Desc(N_i)$= $\{N_j$ con $j = 0, ..., G$ y $j \neq i$ / $d_j(N_i) = 0\}$.□

Regla 5 En caso de igualdad de distancia entre n nodos del mismo nivel, se deben colocar de izquierda a derecha según el orden dado en la definición anterior y crear $\lfloor \frac{n}{2} \rfloor$ nodos ficticios de forma que cada uno de ellos tenga dos de los n nodos como descendientes. En caso de igual distancia entre nodos de diferente nivel, se añadirán los nodos ficticios por orden de coste (del arco a este nodo ficticio) de menor a mayor.

En el ejemplo anterior el orden es N_2, N_1, N_3 puesto que todos coinciden en las tres primeras variables y difieren en la cuarta. Por lo tanto, según la regla 5 se debería crear el nodo ficticio entre N_2 y N_1, es decir $NF_{1,2}$ sin necesidad de comprobar los costes.

También en el ejemplo anterior puede darse el caso de que exista un nodo $N_4 =$ (6 8 $\hat{6}$) cuyo coste es $C_{0,4} = 16$. Al decidir incorporar $NF_{1,2}$ en lugar de $NF_{2,3}$ se ganaba en una operación, pero en este (existiendo N_4) caso, si se hubiera incorporado $NF_{2,3}$ se podría haber colocado N_4 como descendiente suyo (puesto que $d_4(NF_{2,3}) =$ 0) obteniéndose N_4 con coste 11. Por lo tanto, haber incorporado $NF_{2,3}$ en lugar de $NF_{1,2}$ habría costado 4 operaciones menos en el coste total del árbol. Para intentar paliar en gran parte este problema se enuncia la regla 6.

Regla 6 En caso de igualdad de distancia entre nodos, se deben crear los nodos ficticios que tengan mayor número de descendientes.

La regla 6 servirá como primer criterio para crear nodos ficticios habiendo igualdad de distancia entre nodos, y si se diera igualdad también en esta regla pasaríamos a usar la 5.

Regla 7 Si $N_k, N_j \in Desc(N_i)$ y $N_i \in L_s$, un nodo ficticio $NF \in L_q$ cuyos descendientes sean N_k, N_j no se creará si $q \geq s$.

En la figura 6.1 se muestra el árbol de resolución de un M.E.S. con 5 variables endógenas y 8 exógenas. Las variables endógenas han sido representadas en cada nodo por su número más K, también se les coloca un gorro para su distinción y para indicar que han sido sustituidas por las variables *proxy*. El nodo N_0 está formado por todas las variables exógenas y todas las endógenas. El acceso a cada una de las ecuaciones se hace desde el nodo cero retriangularizando la matriz resultante por rotaciones de Givens.

En la figura 6.2 se representa el mismo árbol que en la figura 6.1, pero se han añadido nodos ficticios (representados por rectángulos) siguiendo las reglas descritas hasta ahora y aplicadas en el orden que describe el algoritmo que se verá en la siguiente sección. El primer nodo ficticio en ser añadido ha sido N_6 ya que, según la regla 3 hay que añadir nodos ficticios entre nodos con distancia mínima. Puesto que hay igualdad en la distancia menor ($d_{1,2} = d_{3,4}$) se aplica la regla 5. Después se

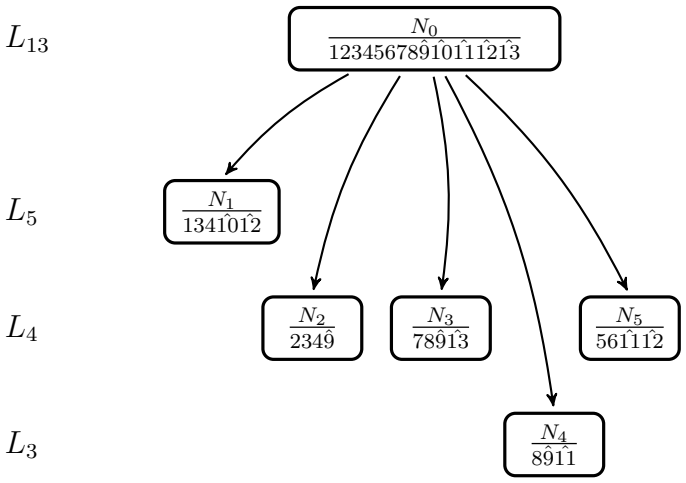

Figura 6.1: Árbol de resolución de un M.E.S. con N=5 y K=8 donde se representan los nodos ecuaciones y el nodo raíz por niveles (mostrados a la izquierda). En cada nodo se representan las variables que lo componen, siendo las que llevan gorro las variables endógenas y las que no las exógenas.

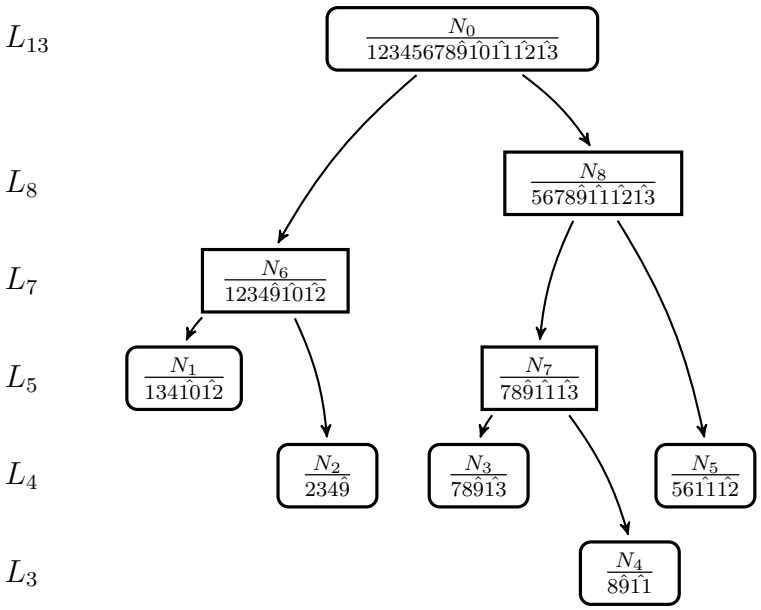

Figura 6.2: Árbol de resolución del M.E.S. dado en la figura 6.1 donde se han añadido tres nodos ficticios.

han añadido los nodos N_7 y N_8 en este orden siguiendo también la regla 3. No se ha añadido un nodo ficticio que tuviera como descendientes a N_7 y N_8 por coincidir con N_0 e incumplir la regla 7.

Regla 8 Un nodo ficticio se añade al árbol si reduce el coste entre un nodo existente y sus hijos.

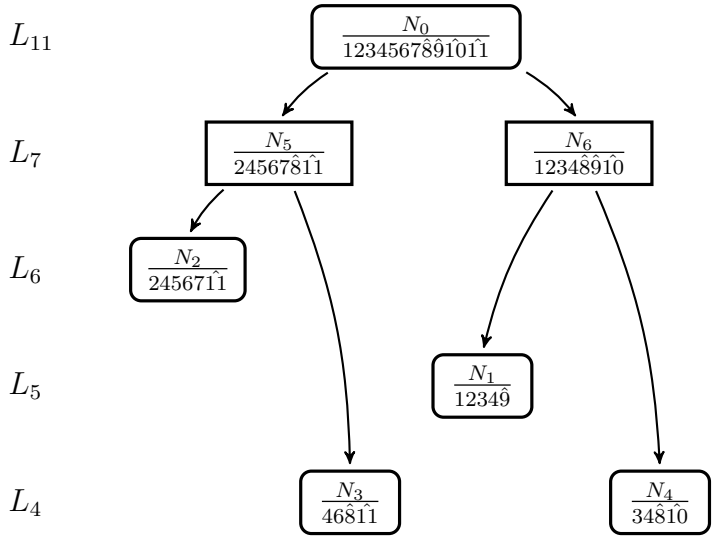

Figura 6.3: Árbol de resolución de un M.E.S. donde se han añadido dos nodos ficticios. El nodo N_5 mejora el coste global y el nodo N_6 lo empeora.

La regla 8 viene a decir que no todos los nodos reducen el coste del árbol, por lo que solo se añadirán los que lo hagan. En el árbol mostrado en la figura 6.3 se observan cuatro nodos ecuaciones y dos nodos ficticios (representados por un rectángulo), uno de los cuales se incluye (el nodo N_5) y otro no (N_6) por la regla 8. En la parte izquierda del árbol, si no se añade N_5 se tienen los costes $C_{0,3} = 35$ y $C_{0,2} = 35$, que suman 70, y al añadir el nodo N_5 se tienen $C_{0,5} = 45$, $C_{5,3} = 19$ y $C_{5,2} = 1$, lo que suma 65. Por lo tanto el añadir el nodo N_5 disminuye el coste global del árbol. En la parte derecha, si no se añade N_6 se tienen los costes $C_{0,4} = 25$ y $C_{0,1} = 2$, lo que suma 27, y al añadir el nodo N_6 los costes son $C_{0,6} = 8$, $C_{6,4} = 21$ y $C_{6,1} = 1$, que suman 30. Por lo tanto al añadir el nodo N_6 el coste global del árbol aumenta.

Se presentan a continuación el algoritmo heurístico que aproximará un Árbol de Coste Mínimo (AMC) y el algoritmo de resolución de Modelos de Ecuaciones Simultáneas que lo usa ($MC2E_{NF}$). La complejidad de éstos algoritmos es estudiada experimentalmente en la siguiente sección, no estudiándose la complejidad teórica dada su alta dependencia del problema abordado (el coste de resolver un sistema con nodos ficticios depende en gran medida de los nodos obtenidos, los cuales dependen de las ecuaciones y por lo tanto del problema que se aborde).

Algoritmo 15 Algoritmo *AMC*

Entrada: $B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$
Salida: AMC

Crear árbol con $N_0 = R$, N_i nodo correspondiente a la ecuación i, $\forall i = 1, ..., N$ y arcos desde N_0 a N_i
Crear tabla de distancias entre nodos
$G = N$, $i = 0$
Mientras $i \leq G$ **Hacer**
  $seguir =$ **Verdadero**
  Mientras ($|Desc(N_i)| \geq 2$) **Y** ($seguir =$ **Verdadero**) **Hacer**
    Buscar N_j, $N_k \in Desc(N_i)$ / $d_{j,k} = min\{d_{r,s}$ / $N_r, N_s \in Desc(N_i)\}$ {Regla 3}
    Si el mínimo no es único aplicar en este orden la regla 6 y la 5
    Si ($d_j(N_k) \geq q_i - q_j$) **O** ($d_k(N_j) \geq q_i - q_k$) **Entonces**
      seguir=**Falso** {Regla 7}
    Si no
      CrearNodo (N_{G+1}) {Según regla 2}
      Añadir N_{G+1} al árbol y a la tabla de distancias
      Calcular $a_{G+1,j}$ y $a_{G+1,k}$
      Eliminar $a_{i,j}$ y $a_{i,k}$ {Regla 1}
      Añadir N_j, N_k a $Desc(N_{G+1})$ y quitarlos de $Desc(N_i)$
      Mientras exista $N_{k'}$ / $d_{k'}(N_{G+1}) = 0$ **Hacer**
        Calcular $a_{G+1,k'}$
        Buscar N_s nodo tal que $N_{k'} \in Desc(N_s)$, y si $C_{s,k'} > C_{G+1,k'}$ eliminar $a_{s,k'}$
        Añadir N'_k a $Desc(N_{G+1})$ y quitarlo de $Desc(N_s)$
        Si $|Desc(N_s)| = 1$ **Entonces**
          Buscar N_a / $N_s \in Desc(N_a)$
          Si $N_d \in Desc(N_s)$ **Entonces** crear $a_{a,d}$ y añadir N_d a $Desc(N_a)$
          Borrar $a_{a,s}, a_{s,d}$ y N_s {Regla 4}
        Fin si
      Fin Mientras
      Si Se reduce el coste global del árbol **Entonces**
        $G = G + 1$
      Si no
        Quitar N_{G+1} y deshacer cambios {Regla 8}
      Fin si
    Fin si
  Fin Mientras
  $i = i + 1$
Fin Mientras

Los puntos 1 y 2 del algoritmo AMC (algoritmo 15) forman la inicialización del algoritmo y pueden ser relativamente costosos en problemas con muchas ecuaciones pues tiene un orden cúbico en número de comparaciones. La idea del algoritmo es tomar un nodo i con un número de descendientes mayor que 2, y crear nodos ficticios por cada dos de sus descendientes reduciéndolos hasta que posea a lo mucho dos (bucle interno, líneas 6-33) añadiendo cada vez que se crea un nodo ficticio todos los descendientes posibles (líneas 17-26). Esto se repite por cada nodo (bucle externo, líneas 4-35). Si un nodo queda con un único descendiente, se elimina del árbol (líneas 21-25). En la línea 28 se añade un nodo si reduce el coste global (lo que solo se puede averiguar una vez hechos los cálculos de las líneas anteriores). Si el nodo no reduce el coste se deshace lo calculado (líneas 12-20).

El algoritmo comienza con $i = 0$, es decir, añadiendo nodos ficticios para nodos descendientes de N_0. Una vez que ya no puede añadir más pasará a hacerlo a los descendientes del nodo N_1, y así hasta llegar a los nodos ficticios añadidos. Por la forma de crear dichos nodos (según la regla 2) la única posibilidad de que se puedan generar ficticios descendientes de ellos es que se les hayan añadido más descendientes en las líneas 17 a la 26. Cada vez que se añade un nodo ficticio se comprueba si se le pueden añadir más descendientes que los dos que lo generaron. Cuantos más descendientes se añadan al nodo, más rentable se hace y menor coste tendrá el árbol resultante, puesto que al añadirle un descendiente se le quita a otro nodo cuyo arco a él es más costoso. A su vez se comprueba que ese otro nodo no quede con un solo descendiente puesto que si se da dicho caso habrá que eliminarlo por la regla 4 (líneas 21-24). El algoritmo finaliza cuando no se pueden añadir más nodos.

El algoritmo no utiliza operaciones en coma flotante. Aun así su coste puede ser alto debido al gran número de comparaciones que debe realizar. En la inicialización del algoritmo (línea 2) se tiene un coste cúbico. Dentro del bucle, en la búsqueda dada en la línea 7 se tiene orden cuadrático, al añadir un nuevo nodo a la tabla de distancias (línea 13) también se tiene el mismo orden. También se tiene orden cuadrático en el cálculo de los costes de los arcos y en la búsqueda de un nodo ficticio que sea padre de uno dado. Por lo tanto, se tiene orden cuadrático dentro de dos bucles (líneas 4 y 6) con lo que el orden se incrementa hasta la cuarta en el peor de los casos.

El algoritmo 16 ($MC2E_{NF}$) muestra un esquema para el estimador MC2E con retriangularizaciones mediante reflexiones de Givens (similar al expuesto en el algoritmo 13) utilizando el Árbol de Mínimo Coste desarrollado en el punto anterior.

Al igual que antes, en la línea 2 del algoritmo $MC2E_{NF}$ se aplican los reflectores de Householder directamente a la matriz. En la línea 4 se usa el algoritmo 15 para aproximar el Árbol de Mínimo Coste.

En $MC2E_{NF}$, en cada nodo del árbol de AMC se retriangulariza la matriz mediante rotaciones de Givens (línea 8). Dichas rotaciones se aplican también a la matriz $\tilde{Y}'_{N_i}$ (si estamos en el nodo i-ésimo). Esta matriz está formada por las columnas de $\tilde{Y}$ correspondientes a las variables endógenas que están de endógenas principales en ecuaciones accesibles desde dicho nodo. La matriz $\tilde{Y}'_{N_i}$ está formada

Algoritmo 16 Algoritmo $MC2E_{NF}$

Entrada: $X \in \mathbb{R}^{d\times K}, Y \in \mathbb{R}^{d\times N}, B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$
Salida: $B \in \mathbb{R}^{N\times N}, \Gamma \in \mathbb{R}^{N\times K}$

1: Obtener Q y R {desc. QR de Householder de X}
2: Calcular $\tilde{Y} = Q^T Y$
3: **Si** no se ha hallado el AMC **Entonces**
4: Crear el Árbol de Mínimo Coste siguiendo el algoritmo 15
5: **Fin si**
6: **Repetir**
7: Recorrer el árbol y en cada nodo i hacer:
8: Retriangularizar la matriz $[R_{i,1}|\tilde{Y}_{i,1}]$ mediante las rotaciones de Givens
9: Crear $\tilde{Y}'_{N_i}$ submatriz de $\tilde{Y}_{N_s}$ ($N_i \in Desc(N_s)$) de variables endógenas principales cuyas ecuaciones se encuentren accesibles desde el nodo i-ésimo
10: Calcular $\tilde{Y}_{N_i} = \tilde{Q}_i^T \tilde{Y}'_{N_i}$ {aplicando las rotaciones de Givens calculadas en el paso anterior}
11: **Si** N_i es una ecuación **Entonces**
12: Calcular $\hat{\beta}_i = R_{i,1}^{-1} \tilde{Y}_{N_i,1}$ {$\tilde{Y}_{N_i,1}$ está formado por las primeras $n_i + k_i - 1$ filas de $\tilde{Y}_i$}
13: **Fin si**
14: **Hasta que** Fin del Árbol

por columnas de la matriz $\tilde{Y}$ que han sido multiplicadas por rotaciones de Givens en tantas ocasiones como número de nodos se hayan recorrido en el árbol de N_0 a N_i. En muchos casos no es necesario calcular todas las columnas de $\tilde{Y}_{N_i}$ puesto que se pueden obtener de la matriz asociada al nodo. Esto ocurre en nodos ficticios que tengan variables explicativas que son endógenas principales en ecuaciones que están por debajo de él. Cuando se llega al nodo ecuación, $\tilde{Y}_{N_i}$ será un vector columna que coincidirá con $\tilde{y}_i$ del paso 6 del algoritmo 13. En la línea 12 del algoritmo 16, se resuelve un sistema de ecuaciones triangular superior de la misma forma que ocurría en los algoritmos del capítulo anterior.

Como ejemplo del funcionamiento del algoritmo $MC2E_{NF}$ veamos como opera sobre el ejemplo dado en la figura 6.2. En las líneas 1 y 2 se obtienen la descomposición QR de la matriz X y la matriz de variables *proxy*, y por lo tanto se puede formar el nodo N_0 que tendrá de matriz asociada a $[R_{0,1}|\tilde{Y}_{0,1}]$. A continuación se crea el nodo N_6, para lo que se eliminan del nodo N_0 las columnas correspondientes a las variables $5, 6, 7, 8, \hat{11}$ y $\hat{13}$, y se retriangulariza la matriz utilizando rotaciones de Givens. Dichas rotaciones deberían ser aplicadas también a la matriz $\tilde{Y}'_{N_6}$ que es la matriz resultante de tomar las dos columnas de $\tilde{Y}_{0,1}$ correspondientes a las variables endógenas $\hat{9}$ y $\hat{10}$ (que son las dos primeras columnas de QY). Sin embargo, no es necesario aplicar las rotaciones puesto que estas columnas pueden ser copiadas de la matriz asociada al nodo N_6. El hecho de tomar estas columnas es porque por debajo del nodo N_6 están accesibles los nodos N_1 y N_2, que son nodos ecuación

cuyas endógenas principales son $\hat{9}$ y $\hat{10}$. Después de N_6 se llega a N_1, y en este nodo se eliminan las columnas de la matriz asociada a N_6 correspondientes a las variables 2 y $\hat{9}$, y se retriangulariza la matriz, aplicándose las rotaciones calculadas en dicha retriangularización a la matriz Y_{N_1} (que está formada por la primera columna de Y_{N_6}). En este caso sí es necesario hacer los cálculos puesto que no se dispone de esta variable en la matriz asociada al nodo. Como estamos en un nodo ecuación se resuelve la ecuación (línea 12 del algoritmo).

Al retroceder de nuevo al nodo N_6 se libera la memoria de las matrices creadas para N_1. El recorrido en este orden del árbol intenta minimizar en tamaño de la memoria usada puesto que para un árbol relativamente grande, el número de matrices en memoria puede ser muy considerable.

En el nodo N_2 se procede de la misma forma que en el N_1, y al retroceder hasta el N_0 se libera la memoria ocupada por las matrices de N_2 y N_6. A continuación se recorre la parte derecha del árbol siguiendo el mismo esquema.

6.4. Paralelización del algoritmo $MC2E_{NF}$

Se presenta en esta sección la versión en paralelo del algoritmo $MC2E_{NF}$ (algoritmo 16) desarrollado en secuencial anteriormente. La paralelización está diseñada para memoria compartida, pudiéndose reformular para memoria distribuida de forma sencilla.

No se ha desarrollado una verión en paralelo para el algoritmo AMC (algoritmo 15) porque no conseguiríamos mejorar en coste computacional al algoritmo $MC2E_G$ tal y como se demuestra a continuación. Para este razonamiento se usarán los datos tomados en el estudio experimental de este capítulo.

Supongamos que se consigue una versión en paralelo de AMC con la que se obtiene el speed-up óptimo teórico. Obviamente la usaríamos para los problemas dados en la tabla 6.1, en los que el tiempo del algoritmo AMC es alto. Tal y como se observa en la columna 8 de la tabla 6.1, el algoritmo $MC2E_G$ consigue tiempos 3, 4 e incluso 5 veces inferiores a la suma de tiempos dados por los algoritmos AMC y $MC2E_{NF}$. Además, este algoritmo, como se demuestra en la tabla 5.4 del capítulo anterior, tiene una paralelización muy buena, con lo que, aún consiguiendo reducir los tiempos del algoritmo AMC en la paralelización, siempre quedarían considerablemente por encima de los tiempos dados por la versión paralela de $MC2E_G$.

Por lo tanto, tan solo quedaría la posibilidad de desarrollar dicha paralelización para usarla en tamaños del problema en los que el algoritmo $MC2E_{NF}$ es más eficiente. Estos son los estudiados en la tabla 6.2. Pero en este caso los tiempos de cálculo del AMC no son muy altos debido al pequeño valor de N, y no es muy importante su paralelización. Además, tal y como se demuestra en las tablas 6.5 y 6.6, la paralelización del algoritmo $MC2E_{NF}$ se hace eficiente cuando se tienen muchos nodos ficticios en el AMC. Cuando no es el caso, pasar de dos procesadores no es muy eficiente, y es preferible la utilización de la versión paralela de $MC2E_G$.

El algoritmo 17 ($PMC2E_{NF}$) muestra el esquema paralelo del algoritmo 16 ($MC2E_{NF}$) para p procesadores con memoria compartida. Al igual que en el capítulo anterior, en las líneas 1 y 2 se usan funciones de LAPACK en paralalelo (puesto que en muchos sistemas están desarrolladas para poder ser usadas en memoria compartida). En la línea 4 se usa el algoritmo AMC para encontrar el Árbol de Mínimo Coste. En la línea 15 se crea la matriz $\tilde{Y}_{N_i}$ a partir de $\tilde{Y}_{N_s}$ con $N_i \in Desc(N_s)$ (lo cual es posible puesto que ésta última se ha construido a la vez que $[R_{j,1}|\tilde{Y}_{j,1}]$). La matriz $\tilde{Y}_{N_i}$ está formada por las variables endógenas principales cuyas ecuaciones se encuentran accesibles desde el nodo i-ésimo.

Algoritmo 17 Algoritmo $PMC2E_{NF}$

Entrada: $X \in \mathbb{R}^{d\times K}, Y \in \mathbb{R}^{d\times N}, B \in \mathbb{R}^{N\times N}$ y $\Gamma \in \mathbb{R}^{N\times K}$
Salida: $B \in \mathbb{R}^{N\times N}, \Gamma \in \mathbb{R}^{N\times K}$

1: Obtener Q y R {desc. QR de Householder de X en paralelo}
2: Calcular $\tilde{Y} = Q^T Y$ {Aplicar los reflectores de Householder en paralelo a la matriz}
3: **Si** No se ha hallado el AMC **Entonces**
4: Crear el Árbol de Mínimo Coste siguiendo el algoritmo 15
5: **Fin si**
6: EN PARALELO cada procesador $q = 0, ..., p-1$ HACE
7: **Repetir**
8: Recorrer el árbol y en cada nodo i hacer:
9: **Si** $i \equiv q \ mod \ p + 1$ **Entonces**
10: **Mientras** $[R_{j,1}|\tilde{Y}_{j,1}]$ no haya sido creada (siendo $i \in Desc(N_j)$) **Hacer**
11: Esperar
12: **Fin Mientras**
13: Crear la matriz $[R_{i,1}|\tilde{Y}_{i,1}]$ a partir de $[R_{j,1}|\tilde{Y}_{j,1}]$
14: Retriangularizar la matriz $[R_{i,1}|\tilde{Y}_{i,1}]$ mediante las rotaciones de Givens
15: Crear $\tilde{Y}'_{N_i}$ submatriz de $\tilde{Y}_{N_s}$ ($N_i \in Desc(N_s)$) de variables endógenas principales cuyas ecuaciones se encuentren accesibles desde el nodo i-ésimo
16: Calcular $\tilde{Y}_{N_i} = \tilde{Q}_i^T \tilde{Y}'_{N_{i-1}}$ {aplicando las rotaciones de Givens calculadas en el paso anterior}
17: **Si** N_i es una ecuación **Entonces**
18: Calcular $\hat{\beta}_i = R_{i,1}^{-1} \tilde{Y}_{N_i,1}$ {$\tilde{Y}_{N_i,1}$ está formado por las primeras $n_i + k_i - 1$ filas de $\tilde{Y}_i$}
19: **Fin si**
20: **Fin si**
21: **Hasta que** Recorrer todo el AMC
22: FIN PARALELO

La paralelización del algoritmo tiene una clara influencia en la forma en que se recorra el AMC, tanto en memoria como en tiempo de ejecución. A continuación se describien dos posibilidades:

$PMC2E_{NF}$ con recorrido del AMC en profundidad

Esta forma de paralelizar es muy acertada en los casos en los que el árbol es relativamente grande, puesto que en cada nodo que se resuelve es necesario guardar $[R_{i,1}|\tilde{Y}_{i,1}]$ e $\tilde{Y}_{N_i}$, lo que puede suponer una gran cantidad de datos y consecuentemente grandes necesidades de memoria. En un recorrido en profundidad, el algoritmo $PMC2E_{NF}$ sigue el mismo recorrido del árbol que el algoritmo $MC2E_{NF}$ pero resolviendo cada procesador un nodo. En este tipo de recorrido, se mantiene un menor número de datos en memoria que en otros recorridos, liberándose las matrices creadas en cada nodo en el momento en que se han procesado todos los nodos por debajo de él. El problema reside en que un nodo no puede ser resuelto si el nodo del cual desciende no ha sido resuelto previamente. Esto puede hacer que unos procesos tengan que esperar a otros. Sin embargo, si el árbol es suficientemente grande respecto al número de procesadores el tiempo de espera se reduce muchísimo.

En la figura 6.4 se muestra como opera el algoritmo $PMC2E_{NF}$ con recorrido en profundidad con 2 procesadores en el árbol de la figura 6.2. El número colocado encima de cada nodo indica el procesador que lo ha de resolver.

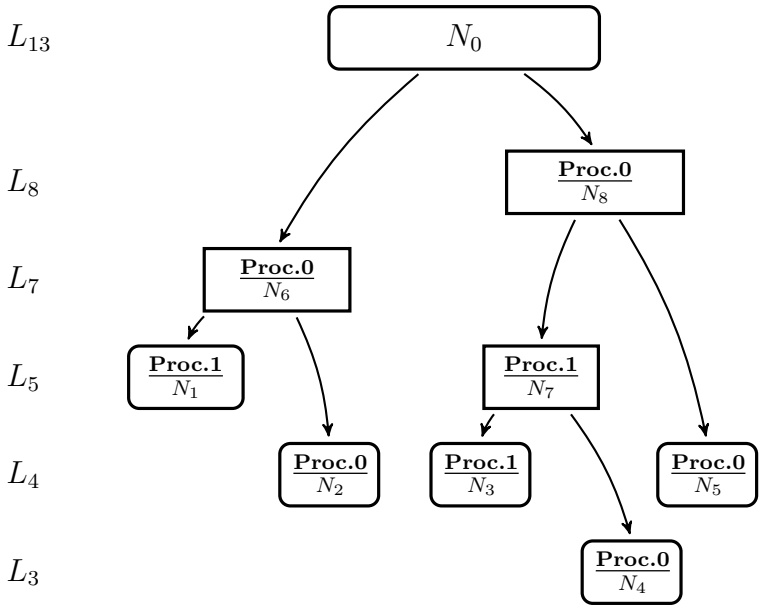

Figura 6.4: Recorrido en profundidad del algoritmo $PMC2E_{NF}$ del AMC dado en la figura 6.2.

Para la comparación del tiempo de ejecución de las paralelizaciones descritas anteriormente, simplificaremos los tiempos empleados en la resolución de cada nodo. Supondremos que se tarda 1 unidad de tiempo en hacer un nodo ficticio (retriangularizar la matriz asociada al nodo) y dos unidades de tiempo en hacer un nodo ecuación (puesto que además de retriangularizar hay que resolver un sistema), el tiempo en resolver el árbol en secuencial sería 13 unidades. En paralelo el procesador

0 hace el nodo N_6 mientras que el procesador 1 está situado en el nodo N_1 esperando. Cuando termina (llevamos por lo tanto una unidad de tiempo) el procesador 1 hace N_1 y el procesador 2 hace N_2 simultáneamente (dos unidades de tiempo). A continuación el procesador 0 hace el nodo N_8 mientras que el procesador 1 espera en N_7 (una unidad de tiempo). A continuación el procesador 1 hace el nodo N_7 mientras que el procesador 0 hace el nodo N_5. Cuando el procesador 1 termina (una unidad de tiempo), el procesador 0 todavía está a la mitad de N_5, por lo que comienza con N_3. Ahora, el procesador 0 termina una unidad de tiempo despúes de que el procesador 1 comience con N_3, por lo que hace N_4 (dos unidades de tiempo más puesto que el procesador 1 termina antes que él). En total se han usado 8 unidades de tiempo.

Sin embargo, se observa que hay una dependencia grande entre procesos (si en lugar de dos procesadores fueran más la dependencia es mayor) y parte del tiempo se pierde en la espera. No sería difícil reasignar los nodos para minimizar esta dependencia, sin embargo para árboles suficientemente grandes el tiempo de espera disminuye notablemente respecto al tiempo total.

Recorrido del árbol en amplitud

Esta forma de paralelizar es apropiada cuando el árbol es pequeño. En este caso, la cantidad de memoria no será excesivamente grande, reduciéndose el tiempo de espera considerablemente. En un recorrido en amplitud del árbol se resuelven los nodos por niveles, desde las capas más altas hasta las ecuación. Además hay que tener en cuenta que las matrices asociadas a los nodos de las capas altas son mayores que las de las capas bajas, siendo mayor el número de rotaciones de Givens necesarias para retriangularizarlas. Por ésto, el tiempo de espera de los procesos en el recorrido en profundidad se incrementa en las capas altas. En este caso, se evita dicha espera asignando procesadores a cada nodo de la capa (como se muestra en la figura 6.5). Se muestra como opera el algoritmo $PMC2E_{NF}$ con recorrido en amplitud con 2 procesadores en el árbol de la figura 6.2. De nuevo se indica encima de cada nodo el procesador que lo ha de resolver.

Suponemos, igual que en el caso anterior, que se tarda 1 unidad de tiempo en hacer un nodo ficticio y dos unidades de tiempo en hacer un nodo ecuación. El procesador 0 hace el nodo N_8 mientras que el procesador 1 hace el nodo N_6 (una unidad de tiempo). Cuando terminan, el procesador 0 es asignado a N_1 y el procesador 1 a N_7. Termina el procesador 1 (una unidad de tiempo) y el procesador 0 necesita otra unidad para terminar, por lo que se asigna a N_2. Una unidad de tiempo más tarde termina el procesador 0, que es asignado a N_3. Una unidad de tiempo después el procesador N_1 se asigna a N_5 y por último el N_0 a N_4. El tiempo total es 7, que es menor que las 8 unidades que tardaba el otro recorrido y las 13 del algoritmo secuencial.

Sin embargo, las matrices creadas en los nodos N_6, N_7 y N_8 no son eliminadas hasta el final, ocupando en todo el proceso una cantidad de memoria considerable.

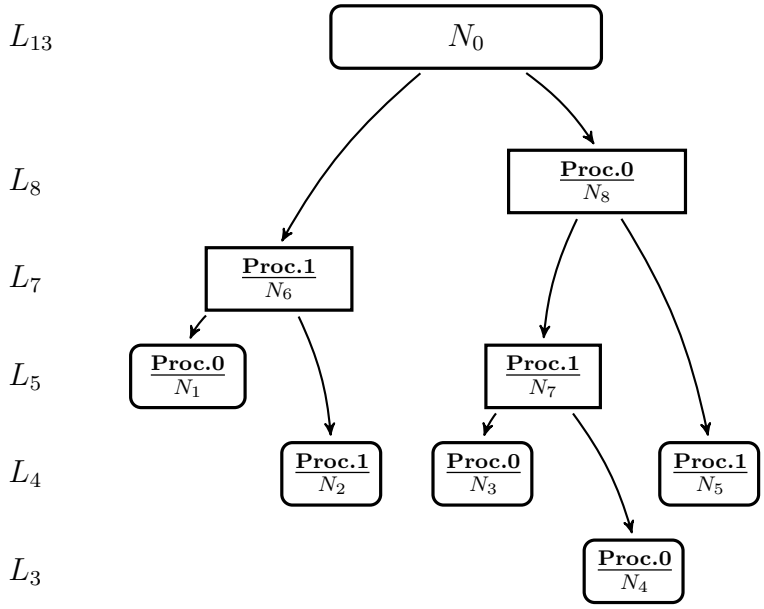

Figura 6.5: Recorrido en amplitud del algoritmo $PMC2E_{NF}$ del AMC dado en la figura 6.2.

6.5. Estudio experimental

En esta sección se van a describir diversos experimentos mediante los cuales se estudian los algoritmos propuestos en los apartados anteriores. Se han utilizado diferentes formas de generar los modelos de ecuaciones simultáneas dependiendo de los intereses que se tengan en el estudio de cada algoritmo.

Se pretenden estudiar experimentalmente los siguientes puntos:

- La influencia de los parámetros N, K y d en AMC y $MC2E_{NF}$.
- Comparar los tiempos de ejecución de $MC2E_{NF}$ y $MC2E_G$ para diferentes tamaños del problema, en los casos que se tenga creado el AMC y en los casos que se tenga que hallar llamando a AMC.
- Nodos añadidos y rechazados en la creación del árbol AMC, así como la reducción del número de operaciones en $MC2E_{NF}$ usando el árbol.
- Estudio del speed-up en $PMC2E_{NF}$.

Para realizar los experimentos cuyos resultados se muestran en las tablas 6.1, 6.4 y 6.5 se han generado M.E.S. de la misma forma que en el capítulo anterior.

La introducción de nodos ficticios se hace rentable, tal y como se mostrará y explicará en las tablas 6.2 y 6.3, cuando existen numerosas ecuaciones con variables similares dentro de un sistema grande. Este hecho se da en la realidad cuando se construye un modelo a partir de muchos modelos de menor tamaño. Por ejemplo,

en el modelado de variables de carácter económico a nivel europeo se construye un M.E.S. a partir de los M.E.S. de cada país añadiendo algunas ecuaciones de unión. Esto hace que el modelo europeo tenga una gran cantidad de variables (todas las de los países) pero, por ejemplo, en las que incorpora el modelo español la mayoría de las ecuaciones utilizarán variables locales. Por ejemplo, en el modelo a escala mundial (gestionado por el proyecto LINK en la universidad de Toronto [9]) incluye al modelo español (gestionado por el CEPREDE [3] o Centro de Predicción Económica y por el Instituto Lawrence R. Klein [4] perteneciente a la Universidad Autónoma de Madrid) más una serie de ecuaciones que relacionan variables del modelo español con otras globales.

Puesto que el fin, para que el simulador genere modelos similares a los descritos en el párrafo anterior, es intentar obtener un sistema grande donde haya unas pocas ecuaciones que tengan cualquier variable y una gran parte de las ecuaciones cuyas variables estén dentro de un subconjunto no muy grande de las del sistema, se generan sistemas aleatorios de la siguiente forma: Se genera un sistema donde las $\frac{N}{4}$ primeras ecuaciones tendrán todas la variables del sistema, y el resto de ecuaciones tendrán solo un número de variables endógenas y exógenas generado entre $3\frac{N}{4}$ y N y $3\frac{K}{4}$ y K, respectivamente. A continuación se procede igual que en la generación de los sistemas descritos anteriormente. Para realizar los experimentos cuyos resultados se muestran en las tablas 6.2, 6.3 y 6.6 se han generado los M.E.S. de esta forma.

La tabla 6.1 muestra una comparativa entre la resolución de un modelo por rotaciones de Givens y añadiendo nodos ficticios. Como se vio en el capítulo anterior, el algoritmo $MC2E_G$ se ve poco afectado en tiempo de ejecución por el aumento de d y se incrementa considerablemente al aumentar N o K. Esta misma propiedad se repite en el algoritmo $MC2E_{NF}$, lo cual es lógico puesto que se basa también en rotaciones de Givens sobre matrices con ceros desde la fila K-ésima hasta la d-ésima. El tiempo de ejecución del algoritmo AMC también es independiente de d, lo cual es razonable puesto que solo depende de la estructura del sistema y no de los datos de las variables. Se puede observar en la misma tabla que el tiempo de ejecución del algoritmo $MC2E_{NF}$ se ve afectado tanto por el aumento de K como por el aumento de N.

El primer ratio compara los tiempos de resolución del sistema por ambos algoritmos (sin considerar el tiempo de la heurística que busca el AMC). Como se observa, el coste de $MC2E_G$ es aproximadamente cuatro veces mayor que el coste de $MC2E_{NF}$. Utilizar en estos casos el algoritmo $MC2E_{NF}$ puede tener cierto interés. Por ejemplo, cuando se tienen sistemas que hay que resolver continuamente debido a cambios en las variables o incorporación de nuevos datos. En estos casos, en que el sistema no varía, el algoritmo AMC solo sería ejecutado la primera vez puesto que el árbol no variará, siendo el resto de veces más rentable el algoritmo $MC2E_{NF}$ que el $MC2E_G$.

Pero salvo en casos como el anterior, lo normal es que haya que incluir en el tiempo de ejecución la búsqueda del AMC y la resolución del sistema, por lo que el segundo ratio es el relevante, dando el algoritmo $MC2E_G$ aproximadamente cuatro

			Tiempos			Ratio de los tiempos	
N	K	d	$MC2E_G$	AMC	$MC2E_{NF}$	$\frac{MC2E_G}{MC2E_{NF}}$	$\frac{MC2E_G}{AMC+MC2E_{NF}}$
400	400	1000	27.87	94.12	7.71	3.61	0.27
400	400	1500	28.03	94.46	7.99	3.51	0.27
400	600	1000	75.82	151.82	19.80	3.83	0.44
400	600	1500	76.40	148.98	18.73	4.08	0.46
800	800	2000	435.11	2204.85	102.39	4.25	0.19
800	800	2500	439.88	2205.01	103.70	4.24	0.19
800	1000	2000	741.75	3130.04	167.41	4.43	0.22
800	1000	2500	736.48	3148.18	170.98	4.31	0.22

Tabla 6.1: Tiempos de ejecución (en segundos) en Rosebud de los algoritmos $MC2E_G$, AMC y $MC2E_{NF}$, cuando se varía el número de variables endógenas (N), el número de variables exógenas (K) y el tamaño de la muestra (d). Se muestran los ratios de resolución del sistema y del total de tiempos de los algoritmos.

veces menos tiempo de ejecución que el $MC2E_{NF}$.

No es necesario tomar varias medidas en la tabla 6.1 puesto que su fin es ver las grandes diferencias entre ambos algoritmos y lo poco eficiente que puede resultar usar nodos ficticios si la heurística se dispara en tiempo.

La tabla 6.2, sin embargo, estudia el comportamiento del uso de nodos ficticios para problemas en los que sale rentable calcular el árbol y resolverlo. En este caso sí se toman promedios y desviaciones típicas de cinco medidas. Se muestran en la tabla el número de nodos aceptados y rechazados, el ratio de operaciones y el tiempo de ejecución del algoritmo AMC y $MC2E_{NF}$.

Como se puede observar en la tabla 6.2, el número de nodos añadidos al árbol por el algoritmo AMC es mucho menor que el número de nodos que han sido rechazados. Por lo tanto, uno de los principales problemas de esta heurística surge de la gran cantidad de nodos que se evalúan para ser después desechados por no disminuir el coste global del árbol. Es de destacar la gran variabilidad que tiene el número de nodos rechazados y la poca variabilidad que tienen los nodos añadidos. Esto hace pensar que para un tamaño dado del problema, se pueden rechazar muchos o pocos nodos pero el número de nodos que se incorporarán al árbol será más o menos el mismo. También es de resaltar la poca influencia en ambos parámetros que tiene el aumento de K y d y la gran influencia que tiene el aumento de N en los nodos rechazados (que no en los aceptados).

El ratio de operaciones muestra el cociente entre el número de operaciones en coma flotante necesarias para resolver el M.E.S. usando el AMC y sin usarlo. Dicho ratio ha sido calculado dividiendo el total de los costes de los arcos del AMC por el total de los costes de los arcos del árbol inicial. Por ejemplo, el ahorro en el número de operaciones (en promedio) en el primer caso es del 24 %. Como se puede

Tam. del problema			Nodos	Nodos	Ratio de	Tiempo
N	K	d	Añadidos	Rechazados	Operaciones	AMC
20	1000	2000	$8.0_{1.2}$	$80.2_{20.3}$	$0.76_{0.08}$	$0.18_{0.01}$
20	2000	3000	$7.2_{1.1}$	$56.0_{25.5}$	$0.73_{0.07}$	$0.63_{0.06}$
20	3000	4000	$9.8_{1.8}$	$53.0_{20.6}$	$0.79_{0.04}$	$1.60_{0.30}$
20	4000	5000	$8.8_{1.5}$	$60.8_{14.3}$	$0.69_{0.11}$	$2.72_{0.40}$
40	1000	2000	$10.4_{2.9}$	$213.6_{58.3}$	$0.68_{0.03}$	$0.59_{0.07}$
40	2000	3000	$11.6_{3.4}$	$210.8_{86.9}$	$0.69_{0.06}$	$2.19_{0.43}$
40	3000	4000	$10.2_{3.3}$	$207.4_{30.2}$	$0.68_{0.09}$	$4.59_{0.31}$
40	4000	5000	$11.0_{5.2}$	$228.8_{49.5}$	$0.68_{0.08}$	$9.07_{2.05}$
80	1000	2000	$7.8_{3.0}$	$593.0_{49.5}$	$0.67_{0.06}$	$2.78_{0.38}$
80	2000	3000	$10.2_{2.9}$	$561.2_{91.5}$	$0.63_{0.07}$	$10.04_{2.66}$
80	3000	4000	$13.4_{3.6}$	$550.2_{203.0}$	$0.63_{0.07}$	$16.21_{3.69}$
80	4000	5000	$14.6_{3.7}$	$670.0_{122.7}$	$0.67_{0.02}$	$34.11_{6.74}$
160	1000	2000	$15.0_{4.3}$	$1865.6_{453.8}$	$0.65_{0.03}$	$14.75_{5.19}$
160	2000	3000	$12.6_{2.6}$	$1949.6_{493.4}$	$0.63_{0.05}$	$38.13_{12.11}$
160	3000	4000	$14.4_{1.7}$	$2171.0_{382.3}$	$0.62_{0.05}$	$80.38_{12.09}$
160	4000	5000	$16.0_{1.6}$	$2336.0_{385.2}$	$0.65_{0.06}$	$136.56_{27.46}$
320	1000	2000	$16.6_{4.4}$	$6757.2_{479.4}$	$0.64_{0.02}$	$162.46_{10.94}$
320	2000	3000	$13.2_{3.6}$	$7133.4_{800.4}$	$0.66_{0.02}$	$420.65_{144.57}$
320	3000	4000	$16.6_{2.1}$	$7479.8_{634.0}$	$0.63_{0.04}$	$556.28_{59.27}$
320	4000	5000	$15.8_{0.8}$	$7123.4_{672.1}$	$0.62_{0.05}$	$842.07_{73.73}$

Tabla 6.2: Nodos añadidos, rechazados y tiempo de ejecución (en segundos) en Rosebud de AMC, cuando se varía el número de variables endógenas (N), el número de variables exógenas (K) y el tamaño de la muestra (d). Se muestra el promedio (número grande) y la desviación típica (subíndice) de 5 medidas para el mismo tamaño del problema.

observar, la tendencia conforme el tamaño del problema aumenta es de reducir el ratio de operaciones, es decir, cuando el tamaño del problema aumenta, la diferencia entre el número de operaciones necesarias para resolver el M.E.S. usando AMC y sin usarlo, es mayor.

Como se había dicho anteriormente, el tiempo de ejecución del algoritmo AMC se ve muy afectado por el valor N y en menor medida por K (por d no se ve afectado). Hay que resaltar la gran variabilidad mostrada por este parámetro, que en algunas ocasiones llega a ser un tercio del valor promediado. Esto indica una gran dependencia de la estructura del sistema a resolver y, por lo tanto, una baja prescisión a la hora de predecir el tiempo de ejecución.

La tabla 6.3 muestra una comparación detallada de los algoritmos $MC2E_G$ y $MC2E_{NF}$ para M.E.S. con pocas ecuaciones y gran número de variables (de forma similar a los tamaños usados en la tabla anterior). La cuarta columna, denominada Parte Común, muestra los tiempos de las líneas 1 y 2 de ambos algoritmos. La siguiente columna muestra los tiempos de resolución de un sistema mediante el uso del algoritmo $MC2E_{NF}$ sin contemplar el tiempo de creación del árbol (mostrado en la tabla anterior). La siguiente columna hace lo mismo pero resolviendo el sistema mediante el algoritmo $MC2E_G$. Las dos últimas muestran los ratios entre los tiempos medidos. El primer ratio es entre los tiempos de resolución del sistema dados en las columnas 5 y 6, y el segundo muestra la relación entre los dos algoritmos completos, es decir se han sumado la parte común a cada uno y en el caso del algoritmo $MC2E_{NF}$ los tiempos de creación del árbol.

Como se observa en la tabla, el ratio de resolución del sistema es siempre favorable al algoritmo $MC2E_{NF}$, creciendo conforme crece el tamaño del problema y llegando en algunos casos a superar el 80 % de beneficio. Sin embargo, como se ha explicado anteriormente, este ratio solo es significativo en el caso en el que se disponga del Árbol de Mínimo Coste ya creado. En el caso normal (que haya que hallar el AMC), el ratio total indica una comparación entre ambos algoritmos. Se puede ver que existe una relación entre el tamaño de N y de K (el valor de d no afecta) para la cual el algoritmo $MC2E_{NF}$ mejora a $MC2E_G$. Por ejemplo, para $N = 20$ se ve que con $K = 1000$ ambos algoritmos tienen tiempos de ejecución similares, pero conforme se aumenta K el ratio va creciendo llegando a ser el algoritmo $MC2E_{NF}$ un 16 % más rentable. Sin embargo, si aumentamos N, por ejemplo para $N = 160$, un valor de $K = 1000$ daría un ratio a favor del algoritmo $MC2E_G$ pues tardaría el 70 % del tiempo que tarda AMC. Para un valor de $K = 2000$ el tiempo de ejecución de ambos es similar y a partir de ahí el algoritmo $MC2E_{NF}$ tiene un tiempo de ejecución menor.

Resaltar varias cosas, primero el pequeño valor del tiempo de la parte común de ambos algoritmos, lo que indica donde radica la mayor parte del coste computacional (en la resolución del sistema). Y resaltar también que, aunque la variabilidad en los tiempos de ejecución es grande en ambos algoritmos, no ocurre así con los ratios, que tienen una variabilidad baja. Esto tiene una fácil explicación, y es que la dependencia del problema es grande a la hora de medir el tiempo de ejecución en

N	K	d	Parte Común	Resol. sis. $MC2E_{NF}$	Resol. sis. $MC2E_G$	Ratio de resol. sis.	Ratio Total
20	1000	2000	$4.64_{0.04}$	$1.20_{0.18}$	$1.45_{0.22}$	$1.22_{0.12}$	$1.01_{0.02}$
20	2000	3000	$27.84_{0.02}$	$9.58_{1.56}$	$12.32_{2.15}$	$1.29_{0.13}$	$1.06_{0.03}$
20	3000	4000	$80.57_{0.03}$	$39.13_{2.85}$	$49.14_{4.73}$	$1.26_{0.06}$	$1.07_{0.02}$
20	4000	5000	$176.93_{0.21}$	$72.50_{20.75}$	$114.95_{17.57}$	$1.65_{0.31}$	$1.16_{0.05}$
40	1000	2000	$4.60_{0.02}$	$2.65_{0.42}$	$3.39_{0.35}$	$1.29_{0.10}$	$1.02_{0.03}$
40	2000	3000	$27.91_{0.16}$	$18.74_{3.45}$	$23.99_{2.87}$	$1.30_{0.12}$	$1.06_{0.04}$
40	3000	4000	$80.72_{0.13}$	$57.66_{15.39}$	$83.49_{11.66}$	$1.49_{0.21}$	$1.15_{0.05}$
40	4000	5000	$177.53_{0.25}$	$142.94_{32.17}$	$229.08_{22.03}$	$1.65_{0.27}$	$1.24_{0.08}$
80	1000	2000	$4.71_{0.04}$	$5.24_{0.72}$	$7.37_{0.63}$	$1.42_{0.14}$	$0.95_{0.03}$
80	2000	3000	$28.05_{0.14}$	$36.34_{4.52}$	$51.93_{4.63}$	$1.43_{0.07}$	$1.08_{0.05}$
80	3000	4000	$81.58_{0.12}$	$119.47_{26.79}$	$189.90_{21.42}$	$1.63_{0.24}$	$1.25_{0.06}$
80	4000	5000	$178.24_{0.13}$	$288.99_{7.47}$	$466.51_{17.56}$	$1.61_{0.07}$	$1.29_{0.03}$
160	1000	2000	$4.88_{0.12}$	$11.89_{1.09}$	$17.09_{1.21}$	$1.44_{0.07}$	$0.71_{0.13}$
160	2000	3000	$28.60_{0.10}$	$74.52_{8.15}$	$110.15_{10.60}$	$1.48_{0.10}$	$0.99_{0.10}$
160	3000	4000	$81.61_{0.07}$	$233.54_{2.83}$	$405.32_{49.14}$	$1.74_{0.20}$	$1.23_{0.15}$
160	4000	5000	$178.88_{0.15}$	$592.51_{45.85}$	$1008.26_{60.27}$	$1.71_{0.18}$	$1.31_{0.10}$
320	1000	2000	$4.96_{0.05}$	$28.14_{1.42}$	$42.87_{3.36}$	$1.52_{0.05}$	$0.24_{0.01}$
320	2000	3000	$29.09_{0.20}$	$172.06_{9.90}$	$250.99_{11.15}$	$1.46_{0.05}$	$0.47_{0.12}$
320	3000	4000	$82.82_{0.11}$	$511.22_{26.35}$	$892.16_{76.46}$	$1.75_{0.15}$	$0.85_{0.09}$
320	4000	5000	$180.43_{0.49}$	$1206.26_{71.12}$	$2242.92_{149.72}$	$1.87_{0.18}$	$1.09_{0.09}$

Tabla 6.3: Tiempos de ejecución (en segundos) en Rosebud y ratios comparativos de los algoritmos 13 y 15, cuando se varía el número de variables endógenas (N), el número de variables exógenas (K) y el tamaño de la muestra (d). Se muestra el promedio (número grande) y la desviación típica (subíndice) de 5 medidas para el mismo tamaño del problema.

ambos algoritmos, y sin embargo ambos se ven afectados de la misma forma, anulándose dicho efecto en la división (si el modelo a resolver necesita mucho tiempo, lo necesitará para ambos y el ratio seguirá siendo el mismo o muy similar).

Para el estudio experimental de los algoritmos paralelos mostrados en el apartado anterior se han utilizando los mismos generadores de Modelos de Ecuaciones Simultáneas que en el estudio secuencial. Se han desarrollado versiones de los algoritmos en paralelo para memoria compartida utilizando OpenMP y llamadas a LAPACK y BLAS (puesto que como ya se ha dicho, están desarrollados en muchos sistemas para poder ser usados en paralelo en memoria compartida). En los experimentos se han utilizado hasta 8 procesadores.

La tabla 6.4 muestra tiempos de ejecución y speed-ups del algoritmo $PMC2E_{NF}$ para los dos tipos de paralelización descritos en la sección anterior. Los tiempos tomados no contemplan el cálculo del AMC, y por lo tanto el tiempo de la llamada al algoritmo AMC no está sumado. La razón, tal y como se comentó en la tabla 6.1, es que para los tamaños del problema mostrados en la tabla, el tiempo de cálculo del AMC es demasiado grande y no hace rentable el uso del algoritmo $PMC2E_{NF}$ (es más rentable usar $PMC2E_G$). Tan solo sería recomendable su uso en el caso en el que se tuviera que resolver numerosas veces el mismo M.E.S. (por actualización de datos, constante aumento de la muestra, etc.) puesto que el cálculo del AMC solo se haría la primera vez, siendo el tiempo de ejecución el resto de veces similar a los tiempos mostrados en la tabla. En dicho caso hipotético se hace rentable el uso del algoritmo $PMC2E_{NF}$ y su paralelización.

Tal y como muestra la tabla 6.4, los speed-ups obtenidos son altos para tamaños del problema grande. Se puede observar que los tiempos de ejecución de los dos tipos de paralelización son muy similares debido a que el árbol utilizado es relativamente grande y los tiempos de espera en el recorrido en profundidad no sean significativos frente al tiempo total.

Sin embargo, si se consideran los tiempos del algoritmo AMC (que pueden ser observados en la tabla 6.1) los speed-ups son cercanos a uno (incluso con 8 procesadores) en ambos tipos de paralelización, tal y como muestra la tabla 6.5. Aun así esto no tiene una gran importancia puesto que, tal y como se demostró en la sección anterior, en caso de tener que construir el AMC para estos tamaños del problema, el algoritmo sin nodos ficticios tiene menor coste computacional, por lo que sería el usado.

Por lo tanto el interés se centra más en ver como paraleliza el algoritmo para tamaños en los cuales es rentable su uso (dados en la tabla 6.2). La tabla 6.6 muestra los tiempos de ejecución y speed-ups para tamaños del problema en los que se hace rentable el uso de nodos ficticios. Se observa que en general, el speed-up aumenta cuando se incrementa el tamaño del problema en ambos tipos de paralelización. También se observa que los speed-ups obtenidos por la paralelización en amplitud son ligeramente mejores que los obtenidos por la paralelización en profundidad.

N	K	d	1 proc.	2 proc.	Sp	4 proc.	Sp	8 proc.	Sp
Recorrido en profundidad									
400	400	1000	7.71	4.01	1.92	2.37	3.25	1.61	4.79
400	400	1500	7.99	4.08	1.96	2.43	3.29	1.67	4.78
400	600	1000	19.80	9.89	2.00	5.26	3.76	3.22	6.15
400	600	1500	18.73	9.74	1.92	5.11	3.67	3.21	5.83
800	800	2000	102.39	52.19	1.96	28.98	3.53	15.60	6.56
800	800	2500	103.70	52.69	1.97	29.30	3.54	15.70	6.61
800	1000	2000	167.41	87.80	1.91	46.16	3.63	22.99	7.28
800	1000	2500	170.98	89.73	1.91	46.64	3.67	22.86	7.48
Recorrido en amplitud									
400	400	1000	7.71	4.13	1.87	2.4	3.21	1.45	5.32
400	600	1000	18.18	9.33	1.95	5.44	3.34	3.33	5.46
800	800	2000	102.39	52.41	1.95	30.27	3.38	17.5	5.85
800	1000	2000	169.51	90.99	1.86	48.15	3.52	24.13	7.02

Tabla 6.4: Tiempos de ejecución (en segundos) en Rosebud y speed-up correspondientes al algoritmo $PMC2E_{NF}$ sin el cálculo del AMC (no se contempla el tiempo del algoritmo AMC) para ambos tipos de recorrido del árbol, cuando se varía el número de variables endógenas (N), el número de variables exógenas (K) y el tamaño de la muestra (d).

Recorrido en profundidad									
N	K	d	1 proc.	2 proc.	Sp	4 proc.	Sp	8 proc.	Sp
400	400	1000	101.83	98.13	1.04	96.49	1.06	95.73	1.06
400	400	1500	102.11	98.20	1.04	96.55	1.06	95.79	1.07
400	600	1000	171.62	161.71	1.06	157.08	1.09	155.04	1.11
400	600	1500	167.71	158.72	1.06	154.09	1.09	152.19	1.10
800	800	2000	2307.24	2257.04	1.02	2233.83	1.03	2220.45	1.04
800	800	2500	2308.55	2257.54	1.02	2234.15	1.03	2220.55	1.04
800	1000	2000	3297.45	3217.84	1.02	3176.20	1.04	3160.03	1.04
800	1000	2500	3319.16	3237.91	1.03	3194.82	1.04	3178.24	1.04
Recorrido en amplitud									
400	400	1000	102.01	98.25	1.04	96.52	1.06	95.57	1.07
400	600	1000	167.14	158.29	1.06	154.40	1.08	152.29	1.10
800	800	2000	2319.04	2257.26	1.02	2235.12	1.03	2222.35	1.04
800	1000	2000	3323.11	3246.59	1.02	3204.75	1.04	3188.13	1.04

Tabla 6.5: Tiempos de ejecución (en segundos) en Rosebud y speed-up comparativos correspondientes al algoritmo $PMC2E_{NF}$ con el cálculo del AMC (se contempla el tiempo de la llamada al algoritmo AMC) para ambos tipos de recorrido del árbol, cuando se varía el número de variables endógenas (N), el número de variables exógenas (K) y el tamaño de la muestra (d).

N	K	d	1 proc.	2 proc.	Sp	4 proc.	Sp	8 proc.	Sp
Recorrido en profunidad									
20	3000	4000	70.45	62.64	1.12	63.27	1.11	52.41	1.34
20	4000	5000	149.69	109.02	1.37	90.70	1.65	55.96	2.67
40	3000	4000	113.68	61.37	1.85	34.52	3.29	26.53	4.28
40	4000	5000	194.44	129.33	1.50	113.83	1.71	80.05	2.43
80	3000	4000	248.08	205.62	1.21	143.16	1.73	143.16	1.73
80	4000	5000	566.09	431.87	1.31	363.29	1.56	228.66	2.48
160	3000	4000	365.97	222.26	1.65	183.24	2.00	122.78	2.98
160	4000	5000	1329.29	779.17	1.71	521.41	2.55	350.01	3.80
320	3000	4000	854.49	554.00	1.54	357.17	2.39	212.85	4.01
320	4000	5000	2571.66	1482.43	1.73	976.61	2.63	599.32	4.29
Recorrido en amplitud									
20	3000	4000	70.46	58.77	1.20	41.63	1.69	39.34	1.79
20	4000	5000	171.90	159.48	1.08	122.51	1.40	121.85	1.41
40	3000	4000	109.79	60.20	1.82	30.51	3.60	24.23	4.53
40	4000	5000	194.44	128.51	1.51	110.10	1.77	77.80	2.50
80	3000	4000	189.90	118.42	1.60	89.15	2.13	49.54	3.83
80	4000	5000	566.09	328.96	1.72	259.97	2.18	189.38	2.99
160	3000	4000	383.88	252.51	1.52	146.99	2.61	100.61	3.82
160	4000	5000	1329.29	750.13	1.77	425.08	3.13	275.89	4.82
320	3000	4000	932.55	539.48	1.73	300.18	3.11	169.83	5.49
320	4000	5000	2510.03	1339.88	1.87	790.25	3.18	474.77	5.29

Tabla 6.6: Tiempos de ejecución (en segundos) en Rosebud y speed-up correspondientes al algoritmo $PMC2E_{NF}$ sin el cálculo del AMC (no se contempla el tiempo de la llamada al algoritmo AMC) para ambos tipos de recorrido del árbol, cuando se varía el número de variables endógenas (N), el número de variables exógenas (K) y el tamaño de la muestra (d) en valores similares a los de la tabla 6.2.

6.6. Conclusiones

Se han desarrollado nuevos algoritmos que, utilizando la descomposición QR y las rotaciones de Givens, mejoran en ciertos casos los tiempos dados por los algoritmos estudiados en el capítulo anterior. Esto se consigue mediante la inclusión de matrices intermedias que reducen el número de rotaciones de Givens necesarias para hallar la descomposición QR en cada ecuación.

Se ha desarrollado un algoritmo para obtener una aproximación del árbol de mínimo coste donde los nodos corresponden a las matrices asociadas a cada ecuación y otras incluidas para reducir el coste. También se ha desarrollado un nuevo algoritmo para el estimador MC2E que, utilizando el árbol hallado, resuelve las ecuaciones del M.E.S. Dicho algoritmo ha sido paralelizado en memoria compartida.

Se han estudiado y comparado los nuevos algoritmos describiendo en qué casos se obtiene una reducción del coste computacional respecto a los estudiados en capítulos anteriores. También se ha estudiado el speed-up de las dos versiones paralelas desarrolladas según el recorrido del árbol. Se concluye que ambas versiones dan resultados muy similares en problemas donde el árbol es grande, siendo ligeramente superior el speed-up de la paralelización con recorrido en amplitud cuando el árbol es relativamente pequeño.

Capítulo 7

Obtención de Modelos de Ecuaciones Simultáneas mediante Algoritmos Genéticos

Tradicionalmente, los Modelos de Ecuaciones Simultáneas han sido desarrollados por personas con una extensa experiencia en el problema particular que representa el modelo. La relación entre las variables se usa para la creación del M.E.S. a partir del criterio de dichas personas. Pero cuando el número de variables es muy grande o la relación entre algunas de ellas no es muy clara, la construcción del modelo se hace difícil y subjetiva.

En este capítulo se estudia como obtener un Modelo de Ecuaciones Simultáneas a partir de un conjunto de datos de variables. La idea es desarrollar un algoritmo el cual, una vez dadas las variables endógenas y exógenas, encuentre el mejor M.E.S. posible acorde con un criterio de comparación de modelos.

El espacio de posibles soluciones es muy extenso puesto que el número de ecuaciones del mejor modelo está entre 1 y el total de variables endógenas (N) de las que se parte, y en cada una de dichas ecuaciones hay $N(N+K)$ posibles variables. Por lo tanto, usar métodos exhaustivos de búsqueda no es eficiente en este problema, y se plantea en su lugar aplicar técnicas metaheurísticas [33]. Una revisión y clasificación de optimización heurística para problemas de modelos de regresión puede verse en [72]. Existen otros estudios sobre algoritmos de obtención del mejor modelo a partir de un conjunto de variables [32, 41, 74], pero todos ellos son para modelos de regresión y, por lo que conocemos, no se ha estudiado para M.E.S.

Se plantea en este capitulo un algoritmo genético para encontrar el mejor M.E.S. posible a partir de un conjunto de datos de variables [62] siguiendo un criterio de comparación de modelos. La solución puede no ser necesariamente la mejor, pero el coste de encontrar dicha solución es mucho menor que el coste de encontrar la mejor usando algoritmos de búsqueda exhaustivos. Primero se propone una versión básica del algoritmo genético y a continuación se usan algoritmos de avance rápido para mejorar dicho algoritmo creando un algoritmo híbrido. La idea es usar el avance

rápido en el algoritmo genético para explorar mejor el espacio de soluciones. Puesto que esta nueva versión del algoritmo tiene un alto coste computacional, se estudia una paralelización de él en memoria compartida.

7.1. Parámetros de criterio para la comparación de modelos de ecuaciones simultáneas

La medida de lo bueno o malo que es un modelo depende del criterio de expertos en el problema representado. Sin embargo, se pueden usar métodos numéricos para representar dicha medida. Los modelos de regresión con una sola ecuación han hecho uso de parámetros para indicar la bondad del modelo, por ejemplo Akaike Information Criterion (AIC), su versión corregida AIC (AICC), Schwarz BIC, Bozdogan ICOMP, etc. [14, 20, 70]. Algunos de ellos, particularmente AIC, BIC y AICC, han sido modificados para modelos de regresión multivariante [17, 30], y han sido adaptados a M.E.S. [36]. El problema de encontrar el mejor modelo usando los parámetros AIC y BIC y algoritmos genéticos ha sido estudiado para modelos de regresión [47], donde se comparan diferentes técnicas metaheurísticas, así como AIC y BIC que se comparan experimentalmente cuando se varían los parámetros del problema. El mismo autor usa algoritmos genéticos para obtener el conjunto óptimo de variables instrumentales a partir de un conjunto de ellas [46]. En este capítulo se utilizan ideas similares para obtener un Modelo de Ecuaciones Simultáneas eficiente a partir de los datos de las variables.

Dado dos conjuntos de datos de variables, uno el de variables endógenas representado por la matriz Y $(d \times N)$ donde sus columnas $(Y_1, ..., Y_N)$ son cada una de dichas variables, y el de variables exógenas representado por la matriz X $(d \times K)$ donde sus columnas $X_1, ..., X_K$ son cada una de estas variables, el problema consiste en encontrar el mejor modelo posible que muestre la relación existente entre dichas variables. Algunas de esas variables tienen influencia simultánea entre ellas, por lo que aparecerán en diferentes ecuaciones.

Puesto que AIC y BIC están basados en métodos de Máxima Verosimilitud, es necesario minimizar la matriz de covarianzas de los errores de predicción. Esto complica computacionalmente el algoritmo puesto que exige realizar el determinante de dicha matriz que es de tamaño $N \times N$, además de construirla.

Las expresiones de AIC y BIC son:

$$AIC = d \cdot \ln |\hat{\Sigma}_e| + 2 \sum_{i=1}^{N} (n_i + k_i - 1) + N(N+1) \tag{7.1}$$

$$BIC = d \cdot \ln |\hat{\Sigma}_e| + \ln\ d \cdot \left(\sum_{i=1}^{N} (n_i + k_i - 1) + \frac{N}{2}(N+1) \right) \tag{7.2}$$

donde $|\hat{\Sigma}_e|$ es el determinante de la matriz de covarianzas de los errores e_i, $i =$

$1, ..., N$, donde e_i es la diferencia entre Y_i y la estimación de Y_i dada en la ecuación i. Se observa una gran similitud entre AIC y BIC, que solo se diferencian en la ponderación del tamaño del problema (parte derecha de sus expresiones), que en el caso de AIC es multiplicado por la constante 2 y en el caso de BIC es multiplicado por $\ln d$. Esto tiene una influencia grande (sobre todo si d es muy grande como suele ser normal) a la hora de comparar dos M.E.S., puesto que BIC valorará mejor de lo que lo hará AIC aquellos con pocas variables y ecuaciones.

El algoritmo MES_{PC} (algoritmo 18) muestra un esquema para el cálculo del Parámetro de Criterio (*Parameter Criteria*) AIC o BIC de un M.E.S. En el comienzo, el algoritmo requiere resolver el Modelo de Ecuaciones Simultáneas (línea 1). Para esto se pueden usar cualquiera de los algoritmos propuestos en los capítulos anteriores. A continuación se sustituyen los valores estimados y los de las variables exógenas en la forma estructural del sistema dada en la expresión 2.10 del capitulo 2. Además se hace la operación $\alpha = \Gamma X^T$ que es una multiplicación de una matriz $N \times K$ por otra $K \times d$. A continuación se resuelve el sistema de ecuaciones dado por $0 = BY_e^T + \alpha$ donde B se conoce pues ha sido estimada en la línea 1, α ha sido calculada en el paso anterior, e Y_e es la incógnita a encontrar. Dicha matriz Y_e será la estimación que proporciona el modelo a las variables endógenas a partir de los datos de las exógenas y los coeficientes calculados.

Algoritmo 18 Algoritmo MES_{PC}

Entrada: $X \in \mathbb{R}^{d \times K}$, $Y \in \mathbb{R}^{d \times N}$, $B \in \mathbb{R}^{N \times N}$ y $\Gamma \in \mathbb{R}^{N \times K}$
Salida: AIC o BIC (o ambos)

1: Resolver el modelo
2: Calcular $\alpha = \Gamma X^T$
3: resolver el sistema $0 = BY^T + \alpha$ {la solución será Y_e matriz $d \times N$}
4: **Para** j=1...N **Hacer**
5: $\quad e_i = Y_i - Y_{e_i}$ {cada e_i es un vector $d \times 1$}
6: **Fin Para**
7: Calcular $\hat{\Sigma}_e = \frac{1}{d^2} e^T e$ siendo $e = [e_1, ..., e_N]$
8: Calcular $d \cdot \ln |\hat{\Sigma}_e|$
9: **Si** AIC **Entonces**
10: $\quad$ HACER $d \cdot \ln |\hat{\Sigma}_e| + 2 \sum_{i=1}^{N} (n_i + k_i - 1) + N(N+1)$
11: **Fin si**
12: **Si** BIC **Entonces**
13: $\quad d \cdot \ln |\hat{\Sigma}_e| + \ln\ d \cdot \left(\sum_{i=1}^{N} (n_i + k_i - 1) + \frac{N}{2}(N+1) \right)$
14: **Fin si**

Los errores e_i, calculados en la línea 5, son la diferencia entre los valores estimados por el modelo para las variables endógenas y los valores reales de éstas. Dichos errores son variables aleatorias cuya distribución es la del vector de ruido blanco u dado en

1.9. Es decir, cuyo valor esperado es cero puesto que no hay sesgo en la estimación y variabilidad la de u.

A continuación se calcula la matriz de covarianzas de los errores, esta matriz mide o estima la variabilidad de dichos errores. El valor de la posición (i, j) de $\hat{\Sigma}_e$ representa la variabilidad compartida entre e_i y e_j si $i \neq j$, o la variabilidad interna de e_i si $i = j$. En el primer caso el valor de $\hat{\Sigma}_e$ en (i, j) es la Covarianza entre e_i y e_j, y en el segundo caso la varianza de e_i.

En la línea 8 se realiza el logaritmo del determinante de $\hat{\Sigma}_e$. Para ello se pueden usar diferentes métodos como descomposiciones matriciales. En los experimentos dados en 7.4 se ha usado la descomposición LU. En el caso de grandes M.E.S, es posible que el determinante de la matriz $\hat{\Sigma}_e$ sea un número muy grande, tanto que haya que aproximarlo mucho para poder almacenarlo. Después se hace el logaritmo y el error cometido es muy grande. Este problema se resuelve de la siguiente forma: una vez hecha la descomposición LU de $\hat{\Sigma}_e$, el determinante es la multiplicación de los elementos de la diagonal principal de la matriz triangular superior U. En lugar de realizar el logaritmo de la multiplicación, se realiza la suma de los logaritmos de cada uno de los elementos de la diagonal. Esto conlleva realizar N logaritmos en lugar de uno solo pero se mejora muchísimo en la precisión cuando el determinante es un valor muy grande. Por último se calcula la expresión del parámetro que se desee (AIC o BIC).

7.2. Un algoritmo genético para la búsqueda de Modelos de Ecuaciones Simultáneas

El espacio de los diferentes M.E.S. posibles a partir de un conjunto de variables es muy extenso. En número total de combinaciones es $2^{N(N+K)}$ y, tal como se ha dicho al principio, los métodos de búsqueda exhaustiva no son eficientes en un espacio tan grande por el coste computacional que tienen. Como alternativa, se usa un algoritmo genético para aproximar el mejor M.E.S. a partir de un conjunto de datos de variables. Una población de M.E.S. creada como cromosomas que representan cada uno a un modelo, es explorada con el fin de desechar los no válidos y ordenar los posibles candidatos a mejor modelo. Si el cromosoma es válido, es decir, si puede ser resuelto y tiene todas sus ecuaciones identificadas, se le asigna un valor de calidad (mediante parámetros de criterio como AIC o BIC) y se compara con el resto de la población.

Un cromosoma se representa por una matriz con N filas y $N + K$ columnas. En cada fila se representa una ecuación mediante unos y ceros. Si la variable j aparece en la ecuación i, el valor para la posición (i, j) en el cromosoma es uno, y cero en caso contrario. Las primeras N columnas del cromosoma representan las variables endógenas y las otras K las variables exógenas.

En la ecuación i-ésima, la variable endógena principal será la que ocupa la columna i-ésima. Así, si la ecuación i-ésima está en el sistema, la posición (i, i) del

cromosoma tendrá valor uno. Por ejemplo, en un problema con $N = 2$ variables endógenas (Y_1 e Y_2) y $K = 3$ variables exógenas (X_1, X_2 y X_3), el modelo

$$\begin{aligned} Y_1 &= \beta_{1,2}Y_2 + \gamma_{1,1}X_1 + \gamma_{1,2}X_2 + u_1 \\ Y_2 &= \beta_{2,1}Y_1 + \gamma_{2,3}X_3 + u_2 \end{aligned} \tag{7.3}$$

es representado por el cromosoma:

$$\begin{matrix} 11110 \\ 11001 \end{matrix} \tag{7.4}$$

donde las primeras dos columnas corresponden a las variables endógenas, y las otras tres columnas a las exógenas.

El esquema general de un algoritmo genético [62] se representa en el algoritmo 19. El bucle desde la línea 2 a la línea 12 del algoritmo 19 se repite hasta que el proceso alcanza la condición de fin, que puede ser, por ejemplo, alcanzar el número máximo de iteraciones ($MaxIter$) permitidas o que el valor del cromosoma que obtiene la mejor puntuación en la función de fitness se repita un número sucesivo de veces máximo ($MaxBest$).

Algoritmo 19 Esquema de un algoritmo genético

```
1: Inicializar(S)
2: Mientras No CondicionesdeFin(S) Hacer
3:    Evaluar(S) {función de fitness}
4:    SS1=Selecionar los elementos mejores de S
5:    SS2=Cruce(SS1)
6:    Si Hay que mutar Entonces
7:       Seleccionar aleatoriamente c ∈ SS1
8:       Mutar(c)
9:       Introducir c en SS1
10:   Fin si
11:   S=SS1 ∪ SS2
12: Fin Mientras
```

Las funciones usadas así como los diferentes parámetros establecidos en el algoritmo son mostradas a continuación.

7.2.1. Cromosoma Válido

Tal y como se explicó en el capitulo 2, no todas las ecuaciones de un sistema pueden ser resueltas, sino que han de estar identificadas para ello. Por lo tanto se exige que el modelo a buscar tenga todas las ecuaciones identificadas para que pueda ser resuelto y evaluado. Además, tienen que cumplirse otras condiciones como, por ejemplo, que si se elimina una ecuación no aparezca la endógena principal en otra ecuación. Puesto que el conjunto de sistemas válidos es inferior al de cromosomas, es

necesario enunciar una serie de condiciones que dichos cromosomas deben cumplir. Las condiciones necesarias para que un cromosoma sea válido son:

- C1: El modelo tiene que tener al menos una ecuación.
- C2: Si el elemento (i, i) es cero, la columna i-ésima debe tener todo ceros. Esto significa que cuando la ecuación i-ésima no está en el sistema la variable i-ésima no formará parte de otras ecuaciones.
- C3: Cada ecuación en el modelo debe tener al menos dos variables, es decir, si el elemento (i, i) es uno, entonces existe j con $1 \leq j \leq N + K$ y $j \neq i$ tal que el elemento (i, j) es uno.
- C4 (Condición de Rango, ver 2.3.2 o [38]): La ecuación i-ésima está identificada sí y solo sí es posible encontrar una matriz de dimensión $(N-1) \times (N-1)$ de rango completo cuyos elementos son los coeficientes de la forma estructural $\beta_{1,2}, \beta_{1,3}, \ldots, \beta_{N,N-1}, \gamma_{1,1}, \ldots, \gamma_{N,K}$ que no aparecen en la ecuación i-ésima.

El algoritmo $NewCrom$ (algoritmo 20) muestra un esquema para la creación de un cromosoma válido donde se comprueban todas las condiciones descritas. Cuando se analiza un cromosoma para comprobar las condiciones anteriores y por lo tanto, si es un cromosoma válido, no se realiza ninguna operación en coma flotante. Sin embargo, el número de comparaciones puede ser muy alto y debe ser tenido en cuenta. El coste de comprobar si un cromosoma es válido depende del número de condiciones que sean comprobadas (puesto que cuando se falla en una no se comprueba el resto de condiciones) y, en el peor caso, el coste de evaluar C1 es N, el de C2 es $N(N-1)$ y el de C3 es $N(N+K)$. El alto coste de C4 en cada ecuación viene dado porque hay que encontrar una matriz $(N-1) \times (N-1)$ de rango $N-1$ con los coeficientes del resto de ecuaciones (usando solo las variables que no aparecen en la ecuación). El número máximo de posibles matrices es $\begin{pmatrix} K+N-2 \\ N-1 \end{pmatrix}$ (puesto que la ecuación ha de tener al menos dos variables), y $(N-1)!$ comparaciones son hechas en el peor caso para comprobar si la matriz tiene rango máximo.

Por lo tanto, el coste total de comprobar si un cromosoma es válido (en número de comparaciones) es:

$$\begin{gathered} T_{NewCrom}(N, K) = N + N(N-1) + N(N+K) + \\ N \begin{pmatrix} K+N-2 \\ N-1 \end{pmatrix} (N-1)! = 2N^2 + NK + \frac{(K+N-2)!}{(K-1)!} N \end{gathered} \tag{7.5}$$

7.2.2. Inicialización y Condición de Fin

Inicialmente, se crean aleatoriamente muchos cromosomas válidos para formar una población inicial. El tamaño de población ($PopSize$) es un parámetro que se puede variar para adaptar el algoritmo al problema. Cada cromosoma es generado

Algoritmo 20 Algoritmo *NewCrom*

Entrada: N y K
Salida: Un cromosoma válido

1: Generar los $N(N+K)$ elementos aleatoriamente (ceros y unos con la misma probabilidad)
2: {CONDICIÓN C1 y C3}
3: **Si** N o N-1 elementos $e_{(i,i)}$ son cero con i=1,...,N **Entonces**
4: Invertir todos los elementos $e_{(i,i)}$ con i=1,...,N
5: **Fin si**
6: {CONDICIÓN C2}
7: **Para** i=1...N **Hacer**
8: **Si** el elemento $e_{(i,i)}$ es cero **Entonces**
9: Hacer cero todos los elementos en la columna i
10: **Fin si**
11: **Fin Para**
12: {CONDICIÓN C4}
13: **Para** i=1...N **Hacer**
14: **Si** la ecuación i-ésima falla la condición de rango **Entonces**
15: Generar aleatoriamente la fila i-ésima e ir a la línea 3 del algoritmo
16: **Fin si**
17: **Fin Para**

siguiendo el algoritmo 20 y, si se dispone de alguna información previa (se conocen características que tendrá el mejor M.E.S.), se pueden insertar algunos modelos con dichas características en la población inicial y ver su evolución, es decir, observar si sobreviven a numerosas iteraciones o por el contrario mueren enseguida. Además si los modelos sobreviven lo suficiente darán lugar a hijos suyos enriqueciendo las poblaciones siguientes y mejorando la solución encontrada por el algoritmo o disminuyendo su tiempo de ejecución.

7.2.3. Evaluación de un Cromosoma (función de *Fitness*)

La función de fitness en el algoritmo 19 consistirá en la evaluación de un cromosoma, la cual tiene tres partes diferenciadas. Hay que resolver el sistema que representa el cromosoma. Se estiman los valores de las variables endógenas con el sistema. Y finalmente se calculan los parámetros AIC o BIC usando las ecuaciones 7.1 o 7.2. El algoritmo 18 muestra un esquema para evaluar un cromosoma siguiendo las tres partes descritas. La primera parte se realiza en la línea 1, la segunda en las líneas 2 y 3, y la tercera en el resto del algoritmo.

El coste de evaluar un cromosoma será la suma de dichas tres partes:

$$T_{CEv}(N, K, \Omega_n, \Omega_k) = T_{solve}(N, K, \Omega_n, \Omega_k) + \sum_{i=1}^{N} (n_i + k_i - 1) + 2N^2 d + \tfrac{2}{3}N^3 + N \tag{7.6}$$

La ecuación 7.6 considera el caso de que todas las variables endógenas y exógenas estén incluidas en el cromosoma, el cual daría el peor tiempo posible. La función T_{solve} representa el tiempo de resolución del M.E.S. que depende, además del tamaño del mismo M.E.S., del algoritmo de resolución de ecuaciones de los analizados en capítulos anteriores que se haya elegido.

7.2.4. Selección y Cruce

En cada generación de cromosomas, una proporción de la población existente es seleccionada para generar la nueva población. Por lo tanto es necesario una comparación y ordenación entre los cromosomas de los que se extraerá un subconjunto de tamaño *SurvSize* para su supervivencia y cruce. El resto de cromosomas serán eliminados.

Se seleccionan dos cromosomas (*ascendiente*1 y *ascendiente*2) del conjunto de cromosomas extraído para sobrevivir para generar dos nuevas soluciones (*descendiente*1 y *descendiente*2). Para combinar los cromosomas ascendientes y generar los descendientes se pueden usar numerosos métodos. La tabla 7.2 en la sección 7.4 muestra una comparación entre los tres siguientes métodos:

Cruce por elementos ($Cruce1$)

Este cruce es el más básico, y es propuesto como una primera aproximación al problema. Se selecciona aleatoriamente un número e con $1 \leq e \leq N(N+K)$ del total de elementos (ceros o unos) del cromosoma, y los primeros e elementos del cromosoma *ascendiente*1, es decir, las primeras $\lfloor \frac{e}{N} \rfloor$ ecuaciones más los primeros $e - \lfloor \frac{e}{N} \rfloor N$ elementos de la $\lceil \frac{e}{N} \rceil$ ecuación, y los últimos $(NK - e)$ elementos del cromosoma *ascendiente*2 son seleccionados para el cromosoma *descendiente*1. Los otros elementos de los cromosomas ascendientes se seleccionan para el cromosoma *descendiente*2.

Cruce por ecuaciones ($Cruce2$)

Este cruce es similar al *cruce*1 salvo que en lugar de tomar aleatoriamente un elemento, se selecciona una ecuación. Se genera aleatorimanete un número e con $1 \leq e \leq N$ y las primeras e ecuaciones del cromosoma *ascendiente*1 y las últimas $N - e$ ecuaciones del cromosoma *ascendiente*2 se selecionan para el cromosoma *descendiente*1, y el resto de ecuaciones de los cromosomas ascendentes serán seleccionadas para el cromosoma *descendiente*2.

Cruce dentro de una ecuación ($Cruce3$)

Se selecciona aleatoriamente una ecuación e y a continuación se generan dos números $v1$ y $v2$ también aleatoriamente con $1 \leq v1 \leq N$ y $1 \leq v2 \leq K$. Todas las ecuaciones del cromosoma *ascendiente*1 irán al cromosoma *descendiente*1 excepto la ecuación e, y de la misma forma ocurre para el otro cromosoma. La ecuación e se genera en el cromosoma *descendiente*1 copiando las primeras $v1$ variables endógenas de la ecuación e del cromosoma *ascendiente*1 y el resto de variables endógenas serán completadas de la ecuación e del cromosoma *ascendiente*2. De igual forma se copian las primeras $v2$ variables exógenas de la ecuación e del cromosoma *ascendiente*1 y se completa la ecuación con las exógenas del cromosoma *ascendiente*2. De forma similar se genera la ecuación e en el cromosoma *descendiente*2 usando el resto de elementos de la ecuación e de los cromosomas ascendentes.

En la tabla 7.1 se muestra un ejemplo con los tres tipos de cruces descritos anteriormente.

Ascendente		Cruce1 $e = 10$	
$asc1$	$asc2$	$desc1$	$desc2$
11110110	10100100	11110110	10100110
11110101	01110100	11110100	01110101
01110110	11110110	11110110	01110110
Cruce2 $e = 1$		Cruce3 $e = 2$, $v1 = 2$,$v2 = 3$	
$desc1$	$desc2$	$desc1$	$desc2$
11110110	10100100	11110110	10100100
01110100	11110101	11110100	01110101
11110110	01110110	01110110	11110110

Tabla 7.1: Un ejemplo de los tres métodos de cruce. Los valores de e, $v1$ y $v2$ se han tomado aleatorimante, y el número de variables endógenas y exógenas es $N = 3$ y $K = 5$.

El $Cruce1$ es el cruce básico utilizado en los algoritmos genéticos. El $Cruce2$ aporta soluciones a la población que son combiandción de ecuaciones de modelos que ya están en la población. Esto permite explorar otros M.E.S. y, en muchos casos, que los nuevos modelos mejoren a sus ascendientes. El $Cruce3$ es un cruce enfocado a perfilar un M.E.S. en el sentido que genera dos descendientes que son casi igual que los ascendientes salvo en una sola ecuación, explorándose así nuevos modelos que difieren poco de los ya existentes en la población.

7.2.5. Mutación

En cada una de las iteraciones se considera una pequeña probabiliad de mutación (P_{mut}). Si en una iteración hay que mutar, se elige aleatoriamente un cromosoma del conjunto de cromosomas dado por el cruce. Todos los cromosomas de dicho conjunto tienen la misma probabilidad de salir elegidos. A continuación se utiliza el algoritmo 21 para mutar el cromosoma y obtener otro válido que será incluido en la población.

Algoritmo 21 Algoritmo *Mutacion*

Entrada: Un cromosoma c de la población
Salida: El cromosoma c mutado

1: Generar e y v aleatoriamente, tal que $1 \leq e \leq N$ y $1 \leq v \leq N + K$
2: Invertir el elemento $c_{(e,v)}$ {elemento de c situado en la posición (e, v)}
3: **Si** $e = v$ **Y** $c_{(e,v)} = 0$ **Entonces**
4: **Si** algún $c_{(i,v)} \neq 0$ con $1 \leq i \leq N$ **O** $c_{(i,i)} = 0 \; \forall \; 1 \leq i \leq N$ **Entonces**
5: Invertir $c_{(e,v)}$ {No se cumple la condición C1 o C2}
6: Ir a la línea 1
7: **Fin si**
8: **Fin si**
9: **Si** $c_{(e,v)} = 0$ **Y** $|\{c_{(e,i)} \neq 0; \; 1 \leq i \leq N + K\}| < 2$ **Entonces**
10: Invertir $c_{(e,v)}$ {No se cumple la condición C3}
11: Ir a la línea 1
12: **Fin si**
13: **Si** la ecuación e-ésima falla la condición de rango **Entonces**
14: Invertir $c_{(e,v)}$ {No se cumple la condición C4}
15: Ir a la línea 1
16: **Fin si**

En la línea 1 del algoritmo 21 se generan aleatoriamente una ecuación y una variable. El elemento situado en dicha posición se invierte, es decir, si hay un cero se coloca un uno y si hay un uno se pone un cero. Por lo tanto, lo que se está haciendo es seleccionar una ecuación y una variable aleatoriamente y excluir dicha variable de la ecuación si es que está incluida, o incluirla si es que está excluida.

Una vez mutado el cromosoma da origen a un nuevo cromosoma que corresponde a un sistema distinto, por lo que es necesario comprobar si es válido. En las líneas siguientes del algoritmo se comprueba la validez de dicho cromosoma pero teniendo en cuenta que solo se ha variado una fila y que el resto no tienen que ser comprobadas pues antes eran válidas.

7.3. Avance Rápido

Cuando se muta un cromosoma, y por lo tanto se sitúa en una parte diferente del espacio de soluciones, puede ocurrir que el nuevo cromosoma no tenga un valor

de evaluación lo bastante bueno para sobrevivir suficientes iteraciones y crear otros cromosomas descendientes de él en esa área, incluso si la mejor solución está cerca de él.

Para evitar este problema, se propone un algoritmo de avance rápido que será aplicado en la mutación de un cromosoma tal y como se muestra en el algoritmo AR (algoritmo 22). En este algoritmo, una ecuación e y una variable v son generadas aleatoriamente en el cromosoma c (línea 2) el cual es el cromosoma tomado aleatoriamente para ser mutado, y por lo tanto se obtiene un nuevo cromosoma que denotaremos por c_1. Entonces, el mejor cromosoma de la vecindad de c_1 (aquellos cromosomas obtenidos invirtiendo un solo elemento de c_1) es seleccionado (líneas 6 a la 11). Si el mejor cromosoma coincide con c_1, el bucle termina y el cromosoma es incluido en la población. Si no, c_1 es sustituido por el mejor cromosoma encontrado (línea 15) y el proceso continúa. Este proceso se repite hasta que se generan NEG (Number de Equations en Greedy) ecuaciones diferentes. La tabla 7.5 muestra una comparación entre las diferentes soluciones encontradas cuando se varía el valor de NEG.

Algoritmo 22 Algoritmo AR

Entrada: Un cromosoma c de la población
Salida: El cromosoma c mutado

1: **Mientras** Número de ecuaciones generadas$\leq NEG$ **Hacer**
2: Generar $1 \leq e \leq N$ y $1 \leq v \leq N + K$ aleatoriamente
3: $c_1 = Mutar(c)$ {Invertir el elemento (e, v) del cromosoma c}
4: $c_{mejor} = c_1$ y *Condición de Fin*=**Falso**
5: **Mientras** *Condición de Fin*=**Falso Hacer**
6: **Para** v=1...$N + K$ **Hacer**
7: $c_2 = Mutar(c_1)$ {Invertir el elemento (e, v) del cromosoma c_1}
8: **Si** c_2 es válido **Y** *Evaluación*(c_2) <*Evaluación*(c_{mejor}) **Entonces**
9: $c_{mejor} = c_2$
10: **Fin si**
11: **Fin Para**
12: **Si** $c_{mejor} = c_1$ **Entonces**
13: *Condición de Fin*=**Verdadero**
14: **Si no**
15: $c_1 = c_{mejor}$
16: **Fin si**
17: **Fin Mientras**
18: **Fin Mientras**

7.4. Estudio experimental

En esta sección se describen diversos experimentos mediante los cuales se estudian los algoritmos propuestos anteriormente. Se pretenden estudiar experimentalmente los siguientes puntos:

- Comparación de los resultados obtenidos utilizando en el algoritmo genético los diferentes cruces.
- Comparación entre los tiempos necesarios para crear un nuevo cromosoma y evaluarlo.
- Comparación de los resultados obtenidos usando AIC y BIC.
- Comparación de los resultados obtenidos usando el algoritmo AR en la mutación y sin usarlo.
- La influencia en el tiempo de ejecución de N, K y $PopSize$.
- Estudio del speed-up de la paralelización del algoritmo genético.

Los M.E.S. usados en este estudio son generados aleatoriamente sin restringir el número de variables por ecuación pero obligando a que todas sean identificadas. Se generan aleatoriamente las N ecuaciones del M.E.S. tomando aleatoriamente n_i valores entre 0 y $N-1$ (serán las variables endógenas) además del valor i (será la endógena principal), y k_i valores entre 1 y K (serán las variables exógenas), con la condición de que la ecuación sea identificada. Una vez creado el modelo se generan los datos de forma que sigan la relación impuesta por el modelo. Por lo tanto se generan aleatoriamente K variables con d datos, cada una las cuales formarán la matriz de variables exógenas X. A continuación se calculan las N variables endógenas (matriz Y) siguiendo la ecuación 2.13 con la variable de ruido blanco u generada siguiendo una $N(0,\sigma)$. Se ha variado el valor de σ para estudiar su influencia en el comportamiento de los algoritmos, siendo mostrado en las tablas donde ha sido relevante su influencia.

El tiempo de la función de evaluación depende fuertemente del algoritmo usado para resolver el M.E.S. En los experimentos de este capítulo se ha utilizado $MC2E_{bas}$ (que es el algoritmo que obtiene tiempos de ejecución más altos de los algoritmos dados para el estimador MC2E). Utilizando otras versiones estudiadas en los capítulos anteriores se puede mejorar mucho el tiempo de evaluación (aunque la solución proporcionada por el algoritmo genético será la misma) mientras que el tiempo de comprobar si un cromosoma es válido seguirá siendo el mismo.

En los experimentos realizados en este capitulo se han establecido los valores de los siguientes parámetros: $SurvSize{=}\frac{PopSize}{2}$, $MaxIter = 150$, y la probabilidad de mutación igual al 1 por ciento. A estos valores se ha llegado después de realizar varias pruebas y comprobar que los resultados obtenidos con el algoritmo son más

eficientes (en relación a la solución hallada y el tiempo de ejecución empleado en su obtención) que en otros casos.

Para comparar los diferentes esquemas propuestos para el cruce de cromosomas, se han diseñado una serie de experimentos cuyos resultados se muestran en la tabla 7.2. En la obtención de dichos datos se ha utilizando en la función de evaluación el parámetro AIC como parámetro de criterio. Se muestra también el coste del algoritmo en segundos así como el número de iteraciones necesarias para que finalice. Se han marcado en negrita los mejores valores obtenidos tanto en tiempo de ejecución como en evaluación del mejor cromosoma. Salvo en un caso (en el que la diferencia no es significativa), el *cruce*1 encuentra mejores soluciones que el *cruce*2 y *cruce*3. Sin embargo, el *cruce*3 consigue soluciones muy próximas en evaluación al *cruce*1 pero en un tiempo mucho más reducido. Por ejemplo, para el tamaño del problema dado por $N = 10$, $K = 15$ y $d = 50$ se obtiene, utilizando el *cruce*3, una solución cuyo valor de evaluación es $2.833{\cdot}10^3$, el cual es un 6 % mayor que el valor de la solución obtenida por el *cruce*1, pero que sin embargo ha sido encontrada en un tiempo de 0.67 o, lo que es lo mismo, en aproximadamente un 20 % del tiempo que ha necesitado el *cruce*1. El *cruce*2 por su parte consigue peores resultados y en tiempos mayores que el *cruce*3. Por lo tanto, concluimos que el *cruce*3 es el más rentable ya que consigue muy buenos resultados en un tiempo reducido, y será el cruce usado a partir de ahora en el resto de experimentos.

			Cruce1			Cruce2			Cruce3		
N	K	d	t	*iter*	Ev	t	*iter*	Ev	t	*iter*	Ev
10	15	50	3.03	48	**2.683**	5.11	97	2.732	**0.67**	20	2.833
15	20	50	8.00	62	4.548	6.73	53	**4.540**	**1.9**4	40	4.709
30	40	100	58.33	50	**21.937**	87.54	72	22.120	**9.47**	17	22.765
40	50	100	325.87	111	**30.956**	294.19	102	31.262	**64.41**	24	32.975

Tabla 7.2: Comparación entre tres métodos diferentes de cruce variando el tamaño del problema. En cada caso, t es el tiempo de ejecución (en segundos), *iter* es el número de iteraciones que el algoritmo necesita para finalizar, y Ev es el valor (por 10^{-3}) de la evaluación del mejor cromosoma encontrado. El tamaño de la población es 100 y $MaxBest$=15.

La comprobación de si un cromosoma es válido está basada en comparaciones y no en operaciones en coma flotante. Sin embargo, la gran cantidad de operaciones que se realizan hace necesario su estudio para ver si su tiempo de ejecución es despreciable frente a otras partes del algoritmo o, por el contrario, aporta gran parte de la carga computacional de éste. La tabla 7.3 muestra una comparación entre el coste de conocer si un cromosoma es válido y el coste de evaluarlo. En la evaluación se usa AIC (aunque un coste similar se da con BIC). En Rt se muestra el ratio entre ambos costes (Función de evaluación/Comprobación de Validez). El coste de saber si un cromosoma es válido o no es menor que el coste de evaluarlo,

pero no es despreciable.

Tamaño del problema			Comprobación de	Función de	
N	K	d	Validez	Evaluación	Rt
50	100	500	0.07	0.79	11.29
50	100	1000	0.07	1.77	25.29
75	100	500	0.43	1.94	4.51
75	100	1000	0.42	3.5	8.33
100	200	500	1.43	7.7	5.38
100	200	1000	1.42	18.21	12.82
150	200	500	7.29	20.31	2.79
150	200	1000	7.28	45.32	6.23

Tabla 7.3: Comparación entre el tiempo (en segundos) de comprobación de si un cromosoma es válido y el tiempo de evaluarlo, en Rosebud, cuando se varía el número de variables endógenas (N), el número de variables exógenas (K) y el tamaño de la muestra (d).

La tabla 7.4 muestra una comparación de las soluciones encontradas por el algoritmo genético cuando se usa AIC o BIC en la función de evaluación, variando el tamaño de la población ($PopSize$), σ y el tamaño del problema. El error mostrado es la suma de los cuadrados de las diferencias entre los valores observados de las variables endógenas principales y aquellos obtenidos por la estimación de dichas variables utilizando el M.E.S. divididos por los valores observados. Los errores son calculados ejecutando 10 veces el algoritmo para los mismos datos de variables. Se muestra la media y la desviación estándar de los errores. Los errores calculados siguen una distribución χ-cuadrado con $(d-1)(N-1)$ grados de libertad si las estimaciones dadas por el modelo están suficientemente cerca de los datos observados (según la prueba de Pearson [69]). Por lo tanto, se obtiene una forma objetiva de comparar ambos errores además de contrastar si el modelo obtenido es suficientemente bueno para una probabilidad dada. Los errores mostrados en la tabla 7.4 son muy pequeños (debido a que σ es muy pequeño) y pasan sobradamente el test de Pearson.

Se observa que el error crece cuando crece el tamaño de N y K. En la mayoría de los casos, BIC obtiene modelos con menor error que AIC. Pero el comportamiento de BIC es irregular ya que hay muchos de ellos en los que, incrementando el valor de $PopSize$ y obteniendo soluciones con BIC menor, se incrementa el valor del error. Por ejemplo, en el caso N=30, K=40 y σ=0, $Error_{BIC}$ es menor cuando $PopSize = 100$ que cuando $PopSize = 500$, lo cual es una contradicción puesto que cuando se aumenta el tamaño de la población la solución encontrada es mejor. Puesto que se está estudiando una metaheurística, la aleatoriedad tiene una influencia muy grande en los resultados del algoritmo. Sin embargo, el hecho de que se hayan realizado 10 mediciones y que esto ocurra en promedio da por conclusión que el problema está en la expresión usada para evaluar. Además, la baja variabilidad indica que los

			$PopSize = 100$		$PopSize = 500$	
N	K	σ	$Error_{AIC}$	$Error_{BIC}$	$Error_{AIC}$	$Error_{BIC}$
30	40	0	$1.47_{0.72}$	$1.24_{0.65}$	$1.31_{0.31}$	$1.33_{0.54}$
30	40	0.01	$1.17_{0.32}$	$0.99_{0.39}$	$0.88_{0.28}$	$0.87_{0.36}$
30	40	0.1	$1.06_{0.32}$	$0.92_{0.42}$	$0.91_{0.35}$	$0.95_{0.31}$
40	50	0	$2.29_{0.52}$	$2.01_{0.43}$	$2.29_{0.64}$	$2.28_{0.78}$
40	50	0.01	$1.64_{0.46}$	$1.58_{0.40}$	$1.59_{0.49}$	$1.62_{0.27}$
40	50	0.1	$1.64_{0.37}$	$1.54_{0.34}$	$1.56_{0.38}$	$1.31_{0.19}$

Tabla 7.4: Comparación de las soluciones encontradas por el algoritmo genético basado en AIC o BIC, cuando se varía el tamaño de la población ($PopSize$), el número de variables endógenas (N) y el número de variables exógenas (K). El cruce utilizado es el $cruce3$, $d = 100$, $MaxBest$=15 y $MaxIter$=150. $Error_{AIC}$ y $Error_{BIC}$ son los errores de la solución utilizando AIC y BIC, respectivamente, en la función de evaluación.

errores son muy similares, reduciendo la influencia de la aleatoriedad. Por lo tanto, se utilizará AIC en el resto de experimentos, dado que tiene un comportamiento más natural.

La tabla 7.5 muestra una comparación entre la solución encontrada por el algoritmo genético con y sin usar avance rápido (AR en la tabla) en la mutación. Los datos mostrados son la media y la desviación típica de 10 ejecuciones del algoritmo para los mismos datos de variables. El parámetro NEG indica el número de ecuaciones generadas en el algoritmo de avance rápido. Se puede observar que cuando solo se genera una ecuación, la solución encontrada sin usar AR y usándolo es similar, pero el tiempo es casi el doble. Cuando el número de ecuaciones utilizadas en el algoritmo AR aumenta, la solución encontrada es mejor (incluso se llegan a conseguir soluciones negativas) puesto que lo que se trata es de minimizar el parámetro de calidad que toma valores en todo $\mathbb{R}$. Pero también ocurre que el coste del algoritmo es mucho mayor, llegando a ser casi siete veces mayor cuando se toman todas las ecuaciones en AR en el caso $N = 10$, y casi 30 veces mayor en el caso $N = 20$. Por lo tanto se observa como se puede mejorar mucho la solución encontrada a costa de mayor coste computacional, coste que aumenta muchísimo cuando se aumenta el tamaño del problema (cuando N y K se aumentan en 10, el tiempo pasa de ser 7 veces mayor que sin usar AR a 30 veces mayor).

La desviación típica es más alta en los casos de $NEG = \frac{N}{4}$ y $NEG = \frac{N}{2}$. Esto es debido a la gran influencia que tiene en el resultado la ecuación que se tome aleatoriamente. En el caso de $NEG = 1$ o $NEG = N$ la influencia es menor, en el primer caso porque se mejora la calidad del cromosoma encontrando otro que comparte todas las ecuaciones salvo una de ellas, por lo que se limita mucho el espacio de búsqueda y la mejora en la calidad del cromosoma encontrado. Y en el segundo caso, se toman todas las ecuaciones, con lo que la influencia de la aleatoriedad es

		$N = 10, K = 20$		$N = 20, K = 30$	
Modo	NEG	AIC	t	AIC	t
sin AR	-	$2138.93_{495.92}$	$5.10_{1.41}$	$4658.06_{722.33}$	$15.41_{5.20}$
con AR	1	$2143.54_{399.53}$	$9.79_{3.92}$	$4710.53_{580.39}$	$49.14_{21.72}$
	$\lfloor \frac{N}{4} \rfloor$	$1491.13_{488.72}$	$12.62_{2.54}$	$3072.98_{1135.88}$	$102.23_{45.53}$
	$\lfloor \frac{N}{2} \rfloor$	$-680.61_{642.75}$	$27.48_{8.89}$	$811.65_{784.60}$	$227.35_{83.93}$
	N	$-3586.46_{453.69}$	$34.17_{20.74}$	$-4920.00_{466.22}$	$449.78_{165.68}$

Tabla 7.5: Comparación entre la solución encontrada por el algoritmo genético usando y sin usar avance rápido en la mutación, cuando se varía el número de variables endógenas (N) y el número de variables exógenas (K). $PopSize$=100, d=100, el cruce usado es $cruce3$, $MaxBest$=30 y $MaxIter$=150.

menor (influye el orden en el que se tomen).

Cuando el tamaño del problema o el parámetro $PopSize$ crecen, el tiempo de ejecución se incrementa considerablemente. Por ejemplo, en la tabla 7.5, cuando se aumenta en 10 unidades N y K, el tiempo aumenta en más de tres veces cuando no se usa el algoritmo AR, y en más de trece veces cuando $NEG = N$. Para reducir el tiempo, se ha desarrollado una versión paralela del algoritmo en memoria compartida. Las partes más costosas del algoritmo 19 son la evaluación, el cruce y la mutación, y han sido paralelizadas asignando varios cromosomas a cada procesador. La población es inicializada una sola vez, y esta inicialización supone un alto coste, por lo que es también paralelizada. La tabla 7.6 muestra el speed-up y el tiempo de ejecución del algoritmo cuando se varían el número de procesadores, el tamaño de la población $PopSize$ y el tamaño del problema. Para estudiar el paralelismo, el algoritmo finaliza cuando el número máximo de iteraciones ($MaxIter$) es alcanzado.

$PopSize$	N	K	d	1 proc	2 proc	sp	4 proc	sp	8 proc	sp
100	10	20	100	17.25	10.61	1.63	6.48	2.66	3.73	4.63
100	20	30	100	123.04	63.74	1.93	33.41	3.68	20.72	5.94
100	30	40	100	717.75	370.99	1.94	190.42	3.77	98.48	7.30
500	10	20	100	71.2	41.74	1.71	24.66	2.89	16.29	4.37
500	20	30	100	280.09	144.82	1.93	97.48	2.87	54.06	5.18
500	30	40	100	1309.45	682.78	1.92	344.18	3.81	180.86	7.24

Tabla 7.6: Tiempo de ejecución (en segundos) y speed-up del algoritmo en memoria compartida cuando se varía el tamaño de la población ($PopSize$), el número de variables endógenas (N), el número de variables exógenas (K), el tamaño de la muestra (d) y el número de procesadores. El cruce usado es $cruce3$, $NEG = \lfloor \frac{N}{2} \rfloor$ y $MaxBest$=$MaxIter$=150.

.

Como se observa en la tabla, el speed-up tiene un comportamiento normal, dado que para $Popsize = 500$ es más alto que para $Popsize = 100$, y que incrementa su valor cuando aumenta el tamaño del problema, al igual que la eficiencia.

7.5. Conclusiones

Se ha desarrollado un algoritmo genético que, a partir de los datos de las variables, devuelve un M.E.S. acorde con dichos datos. La elección del M.E.S. se realiza minimizando un parámetro de criterio. Se han estudiado las diferentes partes del algoritmo genético y ajustado los valores de varios parámetros para obtener un algoritmo más eficiente. También se han comparado dos parámetros de criterio, AIC y BIC, obteniéndose un comportamiento más regular del primero que del segundo.

Además, se ha desarrollado un algoritmo híbrido que utiliza avance rápido en la mutación obteniéndose una gran mejora en la calidad de la solución encontrada, aunque también un aumento considerable en el tiempo de ejecución.

Por último, se ha desarrollado una versión paralela en memoria compartida del algoritmo híbrido, obteniéndose speed-ups satisfactorios.

Capítulo 8

Aplicaciones

En esta sección se utilizan los algoritmos desarrollados en el capítulo anterior para dos estudios reales. El primero trata de encontrar un M.E.S. a partir de los datos de las variables que aparecen en uno de los modelos clásicos de econometría, el Modelo Keynesiano Simple. Una vez obtenido dicho modelo se compara con el creado por Keynes. Los datos del Modelo Keynesiando Simple, su estimación, los valores R^2 y los datos de las variables que se usan en él han sido tomados de [37]. El segundo trata de obtener un M.E.S. eficiente para modelar un conjunto de variables que son usadas en la predicción de preeclampsia.

Los algoritmos usados son las desarrolladas en el capítulo 7. En la resolución de un M.E.S., lo cual es necesario dentro de la función de evaluación, se ha usado el algoritmo $MC2E_G$ desarrollado en el capítulo 5.

La principal aportación de este capítulo es el aplicar las herramientas estudiadas en esta tesis a dos casos reales. El primero otorga la posibiliad de contrastar modelos que han sido propuestos por especialistas y que han sido cuestionados posteriormente. El segundo muestra la aplicación como una herramienta muy útil a la hora de buscar un M.E.S. que modele un conjunto de variables. La aplicación puede dar un primer modelo para ser estudiado y retocado por especialistas en el tema.

8.1. El Modelo Keynesiano Simple

El Modelo Keynesiano Simple (MKS) es un modelo muy conocido, que es usado en las facultades de economía como ilustración básica de un Modelo Macroeconómico de Ecuaciones Simultáneas no complejo. Relaciona el comportamiento del consumo, la inversión y la demanda agregada. Fue desarrollado por John Maynard Keynes (1883-1946), economista británico cuyas ideas tuvieron una gran repercusión tanto en economía como en política. En su teoría, Keynes destacó el carácter ascendente de la curva de oferta, en contraposición con la visión clásica, y además la inestabilidad de la demanda agregada (proveniente de los shocks de mercados privados). Por esto, Keynes proponía políticas fiscales y monetarias intervencionistas para contrarrestar las perturbaciones de la demanda privada.

8.1.1. Descripción del Modelo Keynesiano Simple

El Modelo Keynesiano Simple viene expresado por el siguiente Modelo de Ecuaciones Simultáneas:

$$\begin{aligned}
C_t &= \alpha_0 + \alpha_1 C_{t-1} + \alpha_2 DY_t + \alpha_3 DY_{t-1} + u_{1t} \\
I_t &= \beta_0 + \beta_1 I_{t-1} + \beta_2 Y_t + \beta_3 Y_{t-1} + \beta_4 r_t + \beta_5 r_{t-1} + u_{2t} \\
r_t &= \gamma_0 + \gamma_1 r_{t-1} + \gamma_2 Y_t + \gamma_3 Y_{t-1} + \gamma_4 M_t + \gamma_5 M_{t-1} + u_{3t} \\
T_t &= t_0 + t_1 Y_t + u_{4t} \\
IMP_t &= m_0 + m_1 Y_t + u_{5t} \\
DY_t &= Y_t - T_t \\
Y_t &= C_t + I_t + G_t + X_t - IMP_t
\end{aligned} \tag{8.1}$$

donde C representa el consumo agregado, I es la inversión, Y es el producto interior bruto, DY es la renta disponible, G es el gasto total del gobierno, M es la masa monetaria, X son las exportaciones, IMP son las importaciones, T es total de ingresos por impuestos (federales, estatales y locales), r es el tipo de interés, y las variables u son los terminos de error estocástico. Para corregir el efecto de la inflación y la población, todas las variables son medidas en términos reales y per capita salvo r que está en puntos porcentuales. Las variables endógenas son C, I, r, DY, Y, T y IMP. Las variables predeterminadas son el término constante, las variables exógenas G, X, M y las variables endógenas retardadas en el tiempo.

La primera ecuación modela la función de consumo. La segunda la función de inversión. La tercera determina el tipo de interés y se deriva del equilibrio en el mercado de dinero. La cuarta y la quinta ecuación definen el impuesto y la función de importación. La sexta es una identidad que define el ingreso disponible. La última ecuación es la condición para el equilibrio de productos en el mercado.

Los resultados de las estimaciones para los coeficientes, una vez resuelto el modelo mediante el estimador MC2E, se muestran en la siguiente expresión:

$$\begin{aligned}
\hat{C}_t &= -0.4045 + 0.8521 C_{t-1} + 0.7414 DY_t - 0.5621 DY_{t-1} \\
\hat{I}_t &= -0.2156 + 0.4861 I_{t-1} + 0.6353 Y_t - 0.5561 Y_{t-1} + 0.0698 r_t - 0.0532 r_{t-1} \\
\hat{r}_t &= 3.4645 + 1.1159 r_{t-1} - 0.7935 Y_t + 2.5464 M_t + 3.421 M_{t-1} \\
\hat{T}_t &= -0.9653 + 0.3591 Y_t \\
\hat{IMP}_t &= -2.1553 + 0.2217 Y_t
\end{aligned} \tag{8.2}$$

Aunque el uso del parámetro R^2 no es del todo recomendable en los M.E.S., ya que es un parámetro especialmente diseñado para ecuaciones de regresión, puede ser una forma de intuir lo buena o mala que es una ecuación. En el modelo 8.2 los valores de R^2 resultantes mostrados en orden de ecuación, son: 0.998, 0.97, 0.933, 0.989 y 0.938.

8.1.2. El modelo obtenido por la aplicación

Se han introducido los datos de las variables mostradas anteriormente (facilitados en [66]), además de $N = 5$, $K = 12$ (la constante se cuenta como variable exógena) y $d = 34$, en el algoritmo híbrido dado en el capítulo anterior. Se ha usado AIC como parámetro de criterio y se han tomado $NEG = N$ en el algoritmo AR. El valor de los parámetros se ha establecido a $PopSize = 500$, $SurvSize$=$\frac{PopSize}{2}$, $MaxIter = 150$, y la probabilidad de mutación es igual al 1 por ciento.

Las ecuaciones de equilibrio (las dos últimas ecuaciones del sistema dado en 8.1) han sido utilizadas para eliminar variables y así hacer el modelo más simple. En este caso, se han dejado fuera las variables Y_t y DY_t. Para la comparación con el modelo 8.1 será necesaria la sustitución de dichas variables en el modelo utilizando las ecuaciones de equilibrio.

El modelo resultante, que llamaremos MOA (modelo obtenido por la aplicación) se muestra en la tabla 8.1, donde las ecuaciones son columnas de la tabla. Se muestra con 1 las variables que entran en la ecuación y con un 0 las que no. Se puede observar que el modelo resultante deja fuera la variable T_{t-1}, por lo que $K = 11$, mientras que no prescinde de T_t. También se observa que todas las ecuaciones están identificadas puesto que el número de unos por columna no supera a K. De hecho son todas sobreidentificadas (igual que en el modelo 8.1).

Existen muchas diferencias entre el modelo propuesto por el algoritmo y el Modelo Keynesiano Simple. Por ejemplo, en MOA solo la tercera y la cuarta ecuación tienen constante, mientras que en MKS tienen todas. La influencia de las variables en las endógenas principales de cada ecuación también es diferente. Por ejemplo en la tercera ecuación de MOA, las variables M_t y M_{t-1} no entran, mientras que en la misma ecuación de MKS sí lo hacen. Igual ocurre con la variable r_{t-1}, que también queda fuera de dicha ecuación en MOA y no en MKS. Sin embargo estas variables aparecen en la cuarta y quinta ecuación, por lo que influyen en T_t e IMP_t, y estas dos variables endógenas entran en la tercera ecuación. Por lo tanto hay una influencia indirecta entre M_t, M_{t-1} y r_{t-1} en r_t.

La tabla 8.2 muestra los coeficientes estimados del modelo MOA. El valor del AIC obtenido por dicho modelo es -372.214878.

Los valores de R^2 han sido calculados con la expresión del parámetro R^2 [69] una vez sustituidas las variables *proxy*. Los valores resultantes mostrados en orden de ecuación, son: 0.998, 0.919, 0.968, 0.993 y 0.989. Como se observa, los valores R^2 son muy altos, incluso superan los obtenidos por el modelo MKS. La conclusión es que la aplicación puede obtener modelos con un error menor o igual en la predicción que los ya existentes, pero sin embargo se puede perder la influencia directa entre ciertas variables al no ser incluidas en una ecuación, y sí en otras.

	C_t	I_t	r_t	T_t	IMP_t
C_t	1	0	1	1	1
I_t	0	1	1	0	0
r_t	1	1	1	0	1
T_t	0	1	1	1	0
IMP_t	0	0	1	0	1
C_{t-1}	1	0	0	0	0
T_{t-1}	0	0	0	0	0
I_{t-1}	1	1	1	0	0
r_{t-1}	1	0	0	0	1
M_t	0	1	0	1	1
M_{t-1}	0	0	0	1	1
X_t	0	1	0	1	1
G_t	1	1	1	1	0
X_{t-1}	0	0	0	1	1
G_{t-1}	0	1	1	1	1
IMP_{t-1}	0	0	1	1	0
cte	0	0	1	1	0

Tabla 8.1: Resultado de la salida de la aplicación para las variables del modelo 8.1. Con un 0 se representa que la variable no ha entrado en la ecuación y con un 1 que sí ha entrado

	C_t	I_t	r_t	T_t	IMP_t
C_t	-1	0	11.3363	0.6162	-0.5797
I_t	0	-1	21.9611	0	0
r_t	-0.1424	-0.0562	-1	0	0.5372
T_t	0	1.3498	-13.5651	-1	0
IMP_t	0	0	-71.3275	0	-1
C_{t-1}	1.1611	0	0	0	0
T_{t-1}	0	0	0	0	0
I_{t-1}	-0.5594	0.2416	-20.6502	0	0
r_{t-1}	0.1301	0	0	0	-0.3529
M_t	0	-0.2226	0	0.1621	1.0679
M_{t-1}	0	0	0	-0.1663	-0.8372
X_t	0	-0.8906	0	0.5089	-3.4129
G_t	-0.1049	-1.8848	21.2906	0.8705	0
X_{t-1}	0	0	0	-0.6231	4.6460
G_{t-1}	0	1.6456	-23.708	-0.6007	0.5653
IMP_{t-1}	0	0	53.5432	-0.5447	0
cte	0	0	-17.1907	-1.4432	0

Tabla 8.2: Resultado de la salida de la aplicación para las variables del modelo 8.1. Se representa el valor estimado por el programa para las variables que entran en el modelo.

8.2. Modelo para la preeclampsia

Actualmente trabajamos con el grupo de ginecología del Hospital Virgen de la Arrixaca de Murcia. Más concretamente con la doctora Catalina de Paco, perteneciente también al grupo de investigación del Harris Birthright Research Center for Fetal Medicine (King´s College Hospital, London, United Kingdom). Dicho grupo de ginecología investiga, entre otras cosas, enfermedades de parto prematuro y predicción en las primeras semanas de embarazo. Uno de los problemas más comunes es la preeclampsia (definida a continuación) y su predicción en los centros de salud. Por ahora, se ha conseguido predecir un alto riesgo de padecer preeclampsia de manera más o menos acertada a partir de variables no inmediatas, es decir, variables que deben ser medidas en laboratorios y que cuestan tiempo y dinero, y que incluso pueden poner en riesgo la salud del feto.

El proposito de la investigación de dicho grupo es la de dar herramientas de predicción de preeclampsia a los centros de salud de forma que, a partir de variables "fáciles de medir"(y que por lo tanto se puedan tomar en el mismo centro de salud), se pueda predecir si existe un alto riesgo de padecer preeclampsia.

En este trabajo se pretende modelar las variables que son usadas en la predicción de la preeclampsia con el fin de estimar las que son "difíciles de medir" y aportar parte de las herramientas necesarias para la predicción de preeclampsia en los centros de salud.

8.2.1. ¿Qué es la Preeclampsia?

La preeclampsia es una enfermedad heterogénea que se caracteriza por hipertensión y proteinuria, con efectos variables tanto en el feto como en la madre. En general, si la preeclampsia tiene un inicio precoz (antes de las 34 semanas de gestación), suele asociarse con una importante morbimortalidad maternofetal. Si el inicio es tardío (después de las 34 semanas de embarazo), el crecimiento fetal y la morbilidad materna no suelen ser diferentes a las de un embarazo normal. Afecta aproximadamente al 6 % de las mujeres que quedan embarazadas por primera vez. Es por ello que el motivo es intentar predecir la preeclampsia desde el primer trimestre del embarazo para un mejor control del embarazo e intentar disminuir la morbimortalidad maternofetal.

8.2.2. Descripción general de las variables medidas

Las variables las podemos dividir en aquellas que han sido tomadas de la historia clínica y otras que han sido medidas con la intención de predecir preeclampsia. En estudios previos se comprobó que el gasto cardiaco estaba alterado desde el primer trimestre de embarazo, por lo que la hipótesis del estudio era predecir preeclampsia midiendo el gasto cardiaco [19, 26, 45, 68] y la tensión arterial.

Las variables tomadas de la historia clínica son las siguientes: edad materna,

etnicidad, peso, altura, hábito tabáquico, enfermedades preexistentes (hipertensión arterial, diabetes, enfermedad tiroidea, asma, etc.) y toma de medicación, número de embarazos previos, historia familiar de preeclampsia (madre o hermanas), concepción del embarazo (espontánea o por fertilización in vitro).

Las variables que se han medido para este estudio son: índice de pulsatilidad de las arterias uterinas, gasto cardiaco y la tensión arterial.

El índice de pulsatilidad de las arterias uterinas ha sido medido con Doppler color a nivel del cérvix. Para medir el gasto cardiaco se siguió la guía de la Sociedad Americana de Ecocardiografía [65]. Se utilizó un ecógrafo Toshiba Aplio y una sonda de 3.5-MHz.

El tamaño de la muestra para las variables descritas es de 6117 individuos.

La tabla 8.3 muestra un esquema de las variables usadas en el programa. Se puede ver el nombre de las variables y una breve descripción de cada una de ellas.

Se han introducido en la aplicación los datos de dichas variables, además de $N = 6$, $K = 38$ (la constante se cuenta como variable exógena) y $d = 5714$ (individuos válidos una vez depurada la base de datos), en el algoritmo híbrido dado en el capítulo anterior. Se ha usado AIC como parámetro de criterio y se han tomado $NEG = N$ en el algoritmo AR. El valor de los parámetros se ha establecido a $PopSize = 500$, $SurvSize = \frac{PopSize}{2}$, $MaxIter = 150$, y la probabilidad de mutación es igual al 1 por ciento.

La tabla 8.4 muestra los coeficientes estimados para el modelo (que será denotado por MOA1) obtenido por la aplicación. Si la variable no entra en la ecuación se muestra un cero. Se observa, por ejemplo, que las variables Drugs4, Drugs9 y PET tardía no han entrado en el modelo, y que todas las ecuaciones están sobreidentificadas. También se puede ver que dos de las 6 ecuaciones no tienen término constante, o que existen variables como LVOT que entran en tres ecuaciones pero que en dos de ellas tiene una ponderación muy grande mientras que en la otra muy pequeña (casi insignificante).

El valor del parámetro AIC obtenido por este modelo es 222910.79, el cual resulta muy alto en comparación con el obtenido en el modelo de la sección anterior. Esto es debido fundamentalmente a la gran diferencia existente entre los tamaños de la muestra en ambos casos.

Para estudiar la bondad del modelo obtenido es necesario la utilización de un parámetro acotado en un rango de valores. El parámetro AIC no está acotado y por lo tanto, solo puede ser usado para comparar dos modelos y no para decidir si el mejor de ellos es suficientemente eficiente. Para tal fin se volverá a utilizar, tal y como se ha hecho en el Modelo Keynesiano Simple, el parámetro R^2.

Los valores de R^2 han sido calculados de igual forma que en la sección anterior, y en orden de ecuación son: 0.2, 0.3, 0.977, 0.629, 0.545 y 0.997. Como se observa, las ecuaciones 3 y 6 tienen un valor muy alto, la cuarta y quinta un valor bajo, aunque superior a 0.5, y las dos primeras un valor excesivamente bajo. A partir de estos valores de R^2 se puede concluir que las únicas variables endógenas que serán estimadas con garantía son SV y BSA. De hecho, la predicción se podría ver afectada por la

	Variable	**Grupo**	**Descripción**
1	MAP		Tensión arterial media
2	HR		Frecuencia Cardiaca
3	SV		Volumen de eyección sistólico
4	CO		Gasto cardiaco ($HR \cdot SV/1000$)
5	TPR		Resistencia periférica total ($MAP/(80 \cdot CO)$)
6	BSA		Body surface area ($\sqrt{altura(cm) \cdot peso(kg)}/3600$)
7	Ut PI		Media del índice de pulsatilidad de la art. uterina
8	LVOT		Diámetro de la aorta
9	LA		Diámetro de la aurícula
10	VTI		Velocidad de la aorta
11	ET		Tiempo de ejección
12	MA		Edad materna
13	CRL		Longitud del feto
14	RN	Raza	Raza Negra
15	RA		Raza Asiatica
16	RO		Raza Oriental
17	RM		Raza Mixta
18	RC		Raza Caucasiana
19	Peso		Peso de la madre
20	Altura		Altura de la madre
21	Fumador		La madre fuma durante el embarazo
22	FHPET		No existe historia familiar con PET
23	HC	Grupo	Hipertensión Crónica
24	Diab	Médico	Diabetes
25	No pat		No patología
26	Drugs 1		Antihipertensivos
27	Drugs 2		Insulina
28	Drugs 4	Tipo de	Betamiméticos
29	Drugs 5	medicación	Medicación para asma
30	Drugs 6	que toma	Tiroxina
31	Drugs 8	el paciente	Aspirina/Heparina
32	Drugs 9		Antiepilépticos
33	Drugs 10		Antidepresivos
34	Drugs 15		No medicación - 15
35	PET precoz	Tipo	Precoz
36	PET tardía	de	Tardía
37	No PET	PET	Ninguna
38	C2	Tipo	Medicación para la ovulación
39	C3	de	IVF
40	C7	Concep.	Espontáneo
41	Centile		Percentil del peso del recién nacido
42	NE		Número de embarazos
43	NP		Número de partos

Tabla 8.3: Nombre y descripción de las variables que han entrado en el análisis.

influencia del resto de ecuaciones que podrían sesgar dicha estimación. Por lo tanto, se plantea el generar un nuevo modelo, que se denotará por MOA2, con un número de ecuaciones menor, exactamente las dos que corresponden a los valores altos del R^2. La tabla 8.5 muestra los coeficientes estimados para el modelo obtenido por la aplicación. En este caso los R^2 obtenidos son 0.979 y 0.957, y el valor del parámetro AIC es 53160.51, el cual es menor que el AIC del modelo MOA1. En este modelo no han entrado las variables Ut PI, MA, CRL, RN, RO, Peso, Altura, Fumador, FHPET, HC, No Pat, Drugs2, Drugs4, Drugs5, Drugs6, Drugs9 y Drugs15. Las dos ecuaciones son sobreidentificadas, y ninguna de las ecuaciones tiene constante.

Con este segundo modelo se muestra un claro ejemplo de que la metaheurística no tiene por qué encontrar la mejor solución. En el espacio de busqueda de MOA1 se encuentra el modelo MOA2 que tiene un menor valor de AIC y sin embargo no ha sido obtenido por el algoritmo.

Este segundo modelo está siendo contrastado por el grupo de ginecología y estamos a la espera de obtener resultados empíricos de su validez.

Para la obtención MOA1 se han utilizado 8 procesadores y el tiempo de ejecución ha sido de 14 horas y 4 minutos aproximadamente. Para la obtención de MOA2 se han utilizado igualmente 8 procesadores y el tiempo de ejecución ha sido de 1 hora y 48 minutos aproximadamente.

8.3. Conclusiones

Se han utilizado aplicaciones desarrolladas en esta tesis en dos problemas diferentes, ambos con datos reales. En el primero se compara un modelo clásico de econometría, el Modelo Keynesiano Simple, con otro obtenido por la aplicación para los mismos datos y las mismas variables. En el segundo se propone un modelo para un conjunto de variables para las cuales no se había desarrollado ningún M.E.S. previamente. En este caso se obtiene un primer modelo donde hay ecuaciones cuya predicción no es muy fiable, por lo que se decide acotar las restricciones a menos ecuaciones. Se obtiene así un segundo modelo donde se observan mejores predicciones que en el anterior.

En ambos casos se observan modelos que consiguen predicciones precisas pero que pierden la influencia directa de algunas de las variables. Por lo tanto, la principal aportación de la aplicación es la de dar un primer modelo al especialista con el que comenzar la investigación. A partir de dicho modelo se podría llegar a otro más acorde con el problema y que tenga la visión subjetiva de dicho especialista.

	Variable	MAP	HR	SV	CO	TPR	BSA
1	MAP	-1	0.3498	0	0.0733	14.5201	-0.0007
2	HR	0.243	-1	0	0	0	0.0001
3	SV	-1.3128	-0.9613	-1	0.002	-13.9681	0
4	CO	0	7.7109	0	-1	0	0
5	TPR	0	-0.0235	0.002	-0.0046	-1	0
6	BSA	14.1719	0	0	0	0	-1
7	Ut PI	0	0	0.0444	0	30.1146	-0.0007
8	LVOT	98.759	0	76.4277	0	0	0.0067
9	LA	0	-0.1339	0	0	15.0689	0.001
10	VTI	4.2693	0	3.2444	0	3.7598	0.0002
11	ET	0	-0.0032	0	0	2.0222	0
12	MA	0.1417	0	0	-0.0021	1.151	0.0003
13	CRL	-0.0617	0	0	0	-0.7326	0
14	RN	-0.7276	0	-0.3072	0	0	0.0048
15	RA	0	0.416	0	0.109	0	0
16	RO	0	0	0	0.0695	0	0
17	RM	0	0	0	0	0	0.0042
18	RC	0	-0.1382	-0.3188	0	0	0.004
19	Peso	0	0	0	0	-2.2044	0.0125
20	Altura	-0.083	0	0	0	2.6028	0.0056
21	Fumador	-1.481	0	0.2106	0.024	0	-0.0014
22	FHPET	1.0564	0	0	-0.0352	0	0.0013
23	HC	14.5205	0	0	0	-45.8635	0.005
24	Diab	1.8739	1.0381	0	0	0	0
25	No pat	0	0.3538	0	0	0	0
26	Drugs 1	0	0	0	-0.1642	0	0
27	Drugs 2	0	0	0	0	0	-0.0035
28	Drugs 4	0	0	0	0	0	0
29	Drugs 5	-1.5611	0	0	-0.115	0	-0.0042
30	Drugs 6	1.6577	0	0	-0.0658	0	0
31	Drugs 8	-1.2465	0	-0.3725	-0.0942	-37.2814	0
32	Drugs 9	0	0	0	0	0	0
33	Drugs 10	0	1.2497	0.8167	0	0	0
34	Drugs 15	0	0	0	-0.0559	0	0
35	PET precoz	-3.1511	0	0	0	0	0
36	PET tardía	0	0	0	0	0	0
37	No PET	-1.5257	0	0	0	0	0
38	C2	0	0	0	-0.0269	0	0
39	C3	1.385	0	0.7234	0	0	0.002
40	C7	0	0	0	0	17.9228	0
41	Centile	-0.0171	-0.002	0	0	-0.1889	0
42	NE	0	0	0.0525	0	0	0
43	NP	0	0	0	0	-6.8714	-0.0006
	cte	-142.667	104.9953	-157.6854	5.0826	0	0

Tabla 8.4: Modelo MOA1. Se representa el valor estimado por el programa para las variables que entran en el modelo.

	SV	TPR
SV	-1	-15.8722
TPR	0.0004	-1
MAP	0	14.4689
HR	0	-15.9203
Ut PI	0	0
LVOT	74.742	0
LA	0	-2.1192
VTI	3.1787	0
ET	0.0023	0.0518
MA	0	0
CRL	0	0
RN	0	0
RA	0.2437	17.702
RO	0	0
RM	0.4088	0
RC	0	-3.5957
Peso	0	0
Altura	0	0
Fumador	0	0
FHPET	0	0
HC	0	0
Diab	0	17.9607
No pat	0	0
Drugs1	0.7894	0
Drugs2	0	0
Drugs4	0	0
Drugs5	0	0
Drugs6	0	0
Drugs8	-0.4328	-11.3899
Drugs9	0	0
Drugs10	0.8348	30.4829
Drugs15	0	0
PET precoz	0	2404.5607
PET tardía	0	2405.071
No PET	0	2407.0467
C2	-151.5793	0
C3	-150.8889	0
C7	-151.5818	0
Centile	0	-0.0621
NE	0.0908	0
NP	-0.0863	0
cte	0	0

Tabla 8.5: Modelo MOA2. Se representa el valor estimado por el programa para las variables que entran en el modelo.

Capítulo 9

Conclusiones y Trabajos Futuros

En este último capítulo resumimos las principales conclusiones que se pueden extraer del trabajo realizado con el fin de tener una visión general de los resultados obtenidos. También se enumeran los documentos generados durante la realización de la tesis, junto con los proyectos de investigación en los que se ha desarrollado. Por último, se explican las diferentes vías de investigación que se proponen como trabajos futuros.

9.1. Conclusiones

Como se ha visto en el capítulo de introducción, un Modelo de Ecuaciones Simultáneas es un conjunto de ecuaciones de regresión donde existe influencia simultánea entre variables y ecuaciones. Nacieron en econometría a mediados del siglo pasado y han ido creciendo de forma paralela al desarrollo de las computadoras debido a su altas necesidades computacionales.

Hoy en día, los M.E.S. se han comenzado a utilizar en otras disciplinas y tienen aplicaciones más diversas que aquellas para las que nacieron.

En este trabajo se han propuesto algoritmos de resolución de M.E.S. para dos de los estimadores de información limitada más utilizados, MCI y MC2E.

En primer lugar se han desarrollado algoritmos secuenciales basados en la expresión de cada uno de dichos estimadores. Estos algoritmos han sido comparados teórica y experimentalmente dando como resultado que el basado en MCI tiene menor coste que el basado en MC2E. Sin embargo, existen ciertas condiciones a la hora de poder usar un estimador u otro. Una ecuación tiene que estar exactamente identificada para poder ser resuelta por MCI y, por el contrario, MC2E puede resolver cualquier tipo de ecuación identificada.

Para reducir el coste computacional del algoritmo basado en la expresión del estimador MC2E, se proponen algoritmos basados en descomposiciones matriciales. Se desarrolla en primer lugar un algoritmo basado en una descomposición de la inversa. En este nuevo algoritmo pueden ser reutilizadas muchas de las operaciones realizadas al comienzo de él, reduciendo así el coste computacional. Este nuevo

esquema puede ser usado en todas las ecuaciones salvo en aquellas que no tengan variables exógenas o que solo tengan la endógena principal, ambos casos no muy comunes. En estas ecuaciones se usará el algoritmo desarrollado sin descomposición matricial.

El algoritmo basado en MCI sigue teniendo menor coste que este nuevo algoritmo basado en la descomposición de la inversa por lo que se propone un esquema general de resolución de M.E.S., donde se utiliza MCI en las ecuaciones exactamente identificadas y el algoritmo de MC2E basado en la descomposición de la inversa en las sobreidentificadas. En los casos donde este algoritmo no se pueda utilizar, se usará el algoritmo básico de MC2E basado en su expresión.

Debido al alto coste de estos algoritmos en M.E.S. grandes, se han desarrollado versiones en paralelo en memoria distribuida para todos ellos. Se han estudiado teóricamente y experimentalmente dichas versiones obteniéndose resultados de speed-ups satisfactorios.

Además de la descomposición de la inversa descrita, se ha utilizado la descomposición QR para proponer nuevos algoritmos basados en MC2E. Se han desarrollado dos nuevos esquemas, uno basado solamente en la descomposición QR mediante el método de las reflexiones de Householder y otro que, además de este método, utiliza rotaciones de Givens para retriangularizar la matriz en cada ecuación. Ambos algoritmos han sido comparados dando mejor resultado el que utiliza rotaciones de Givens. De hecho, aunque el basado exclusivamente en Householder da tiempos de ejecución superiores al basado en la descomposición de la inversa, el que utiliza rotaciones de Givens mejora a ambos.

Se ha desarrollado una versión paralela en memoria compartida del algoritmo basado en rotaciones de Givens obteniéndose speed-ups satisfactorios.

En el algoritmo basado en las rotaciones de Givens es posible reducir el número de rotaciones a usar si se comparten columnas entre matrices. Por ejemplo, si las variables que aparecen en una ecuación están incluidas en otra, se podría resolver primero la segunda y después la primera obteniendo su matriz asociada a partir de la matriz de la segunda, y reduciendo así el número de rotaciones de Givens usadas (y por lo tanto el tiempo de ejecución). Sin embargo, este caso no es muy común, por lo que el ahorro no sería grande. Por otro lado, se podrían utilizar matrices intermedias que, no correspondiendo a ninguna ecuación, redujeran el número de rotaciones de Givens necesarias para obtener la descomposición QR de otras matrices asociadas a ecuaciones.

Se plantea un algoritmo que obtiene un árbol cuyos nodos corresponden a las matrices descritas (llamados ficticios) y a las matrices correspondientes a cada una de las ecuaciones del M.E.S. Para la obtención del árbol se propone una heurística basada en un conjunto de reglas.

A continuación se propone un nuevo algoritmo basado, como el anterior, en rotaciones de Givens, pero que utiliza dicho árbol para obtener la descomposición QR de las matrices asociadas a cada ecuación con un número menor de retriangularizaciones. Este nuevo algoritmo tiene un coste de ejecución menor que el anterior.

Sin embargo, el tiempo de obtener dicho árbol puede ser considerable y hacer que el coste total empeore hasta el punto de dejar de ser rentable. Solo en los casos donde se tenga ya construido el árbol porque se resuelva el mismo modelo varias veces (por actualización de la muestra, por ejemplo), o aquellos donde el número de variables es muy grande en comparación con el número de ecuaciones, es más rentable obtener el árbol y resolver el M.E.S. Este último es el caso de muchos modelos macroeconómicos, los cuales están formados por la unión de modelos más pequeños. Al unir muchos modelos pequeños, el número de variables total es muy grande pero al resolver cada uno de ellos se observa que se tienen unas pocas ecuaciones donde se usa solo una parte pequeña de las variables. En este tipo de modelo la utilización del árbol reducirá considerablemente el coste computacional.

Se desarrollan dos versiones en paralelo de este algoritmo. Una recorriendo el árbol en profundidad y otra en amplitud. La primera da mejores resultados que la segunda en árboles grandes, puesto que el tamaño de memoria ocupada (que puede ser un problema debido a la cantidad de matrices intermedias que se pueden llegar a crear) es menor que en otros recorridos. Esto es debido a que se liberan las matrices asociadas a los nodos ficticios conforme se realizan todas las ecuaciones accesibles desde ellos. Sin embargo, es necesario que unos procesos esperen a otros siendo este tiempo despreciable en árboles grandes pero importante en árboles pequeños. El recorrido en amplitud reduce el tiempo de espera de unos procesos a otros pero aumenta el tamaño de memoria ocupada por el algoritmo. Este recorrido es más rentable en árboles pequeños donde el total de memoria no es importante y el tiempo de espera de un proceso a otro con respecto al tiempo total sí lo es.

Una vez se han obtenido diferentes algoritmos para la resolución de Modelos de Ecuaciones Simultáneas, se plantea la obtención de un M.E.S. a partir de los datos de las variables.

Puesto que el espacio de soluciones es demasiado grande, no es recomendable utilizar búsqueda exhaustiva en este problema, por lo que se ha utilizado en su lugar una metaheurística. Concretamente se ha desarrollado un algoritmo genético que obtiene un M.E.S. eficiente a partir de los datos de las variables, indicándole previamente qué variables son endógenas y cuales son exógenas. Para la comparación de modelos se han utilizado dos parámetros de criterio, AIC y BIC, siendo ambos comparados experimentalmente y resultando el segundo más irregular que el primero. En la función de fitness del algoritmo genético es necesario resolver el M.E.S. que representa cada cromosoma. Para dicha resolución se utilizan los algoritmos desarrollados en esta tesis para la resolución de M.E.S.

Para mejorar el algoritmo, se ha desarrollado una versión híbrida que utiliza avance rápido en la mutación. En los experimentos se observa que la solución obtenida por este algoritmo es mucho mejor que la obtenida sin usar avance rápido pero, sin embargo, la carga computacional crece considerablemente. Para disminuir el tiempo de ejecución se desarrolla una versión paralela en memoria compartida de dicho algoritmo híbrido.

Finalmente se ha utilizado está aplicación en dos casos reales. El primero ha con-

sistido en obtener un M.E.S. a partir de los datos del Modelo Keynesiano Simple, para su posterior comparación. Y el segundo ha consistido en obtener un M.E.S. para la estimación de algunas de las variables que intervienen en la predicción de preeclampsia. Con estos dos ejemplos se comprueba que las herramientas desarrolladas pueden ser útiles en diversos campos científicos ayudando a resolver eficientemente M.E.S. y a obtener, a partir de datos, modelos que pueden ser analizados por los expertos.

9.2. Resultados y entorno de trabajo

El trabajo de esta tesis se ha realizado en el marco de varios proyectos de investigación regionales y nacionales:

- Proyecto CICYT, Ministerio de Educación y Ciencia: "Desarrollo y optimización de código paralelo para sistemas de audio 3D"(TIC2003-08238-C02-02), de 1 de enero de 2004 a 31 de diciembre de 2006.

- Proyecto de la Fundación Séneca, Consejería de Cultura y Educación de la Región de Murcia: "Técnicas de optimización de rutinas paralelas y aplicaciones"(02973/PI/05), de 1 de enero de 2006 a 31 de diciembre de 2008.

- Proyecto de la Fundación Séneca, Consejería de Cultura y Educación de la Región de Murcia: "Adaptación y Optimización de Código Científico en Sistemas Computacionales Jerárquicos"(08763/PI/08), de 1 de enero de 2009 a 31 de diciembre de 2011.

- Proyecto CICYT, Ministerio de Educación y Ciencia: "Construcción y optimización automáticas de bibliotecas paralelas de computación científica" (TIN2008-06570-C04-02), de 1 de enero de 2009 a 31 de diciembre de 2011.

  En este proyecto CICYT y en el anterior se ha colaborado con el grupo de Computación Paralela de la Universidad Politécnica de Valencia, principalmente a través del investigador Antonio Manuel Vidal Maciá. También se ha utilizado el clúster Rosebud cuyo administrador es el investigador del mismo grupo Pedro Alonso Jordá.

  En este proyecto figura como EPO el grupo de Series Temporales y Econometría del departamento de Métodos Cuantitativos para la Econometría de la Universidad de Murcia, cuyo responsable es la investigadora Arielle Beyaert con la que se ha colaborado en las aplicaciones.

  También figura como EPO de este proyecto la "Asociación para la investigación de disfunciones sexuales en atención primaria" con sede en el Hospital Morales Meseguer y cuyo director es el doctor Luis García-Giralda Ruiz. Con dicho investigador se trabajó en una aplicación de los algoritmos desarrollados en la tesis y que no ha sido incluida en ella.

- La toma de datos para el modelo de preeclampsia se ha hecho a través de un proyecto otorgado por "The Fetal Medicine Foundation" (Charity No. 1037116) al grupo de investigación del Harris Birthright Research Center for Fetal Medicine (King´s College Hospital, London, United Kingdom) al que pertenece como investigadora la doctora Catalina de Paco.

Durante la realización de la tesis se han realizado varias publicaciones y comunicaciones que se enumeran y comentan a continuación:

- Solution of simultaneous equations systems by high performance methods. Jose J. López-Espín and Domingo Giménez. Computational Statistics and Data Analysis, 28-31 October 2005, Limassol, Cyprus. [51]

  En esta comunicación se presenta un análisis teórico y experimental de algoritmos secuenciales y paralelos basados en los estimadores MCI y MC2E (incluido en el capítulo 3). La paralelización se hace en memoria compartida de forma básica como una primera aproximación al problema.

- Solution of Simultaneous Equations Models in high performance systems. Jose J. López-Espín and Domingo Giménez. XXIX Congreso Nacional de Estadística e Investigación Operativa, 15-19 May 2006, Tenerife, Spain. [52]

  En esta comunicación se presenta el mismo análisis que en la comunicación anterior y se amplia la paralelización al caso distribuido, también de forma básica como una primera aproximación al problema (capítulo 4).

- Solution of Simultaneous Equations Models in high performance systems. Jose J. López-Espín and Domingo Giménez. Workshop on State of the Art in Scientific and Parallel Computing, 18-21 June 2006, Umeå, Sweden. [53]

  En esta comunicación se presenta un análisis completo del algoritmo paralelo en memoria compartida y distribuida descrito en la comunicación anterior. Se proponen varias soluciones, se estudia el speed-up tanto en la resolución de una ecuación como en un sistema completo. Además se estudian los tiempos de comunicación en memoria distribuida (incluido en el capítulo 4).

- Message-passing Two Steps Least Square Algorithms for Simultaneous Equations Models. Jose J. López-Espín and Domingo Giménez. Proceedings International Symposium on Parallel Processing and Applied Mathematics, Lecture Notes in Compute Science, 4967, 127-136, 2007. [49]

  Algoritmos de paso de mensajes para modelos de ecuaciones simultáneas. Jose J. López-Espín and Domingo Giménez. Meeting on Optimization of Parallel Routines and Applications, 10-12 June 2007, Murcia, Spain. [54]

  El primer trabajo, es una publicación donde se analizan tres algoritmos para MC2E, el basado en su expresión sin descomposiciones matriciales, el basado en la descomposición de la inversa y el basado en la descomposición QR utilizando

el método de Householder. Se realiza una comparación teórica y experimental entre ellos. También se analiza la paralelización en memoria compartida de los algoritmos tanto teórica como experimentalmente. Los capítulos 2 y 3 recogen el estudio de los dos primeros algoritmos, y el capítulo 4 el del basado en la QR. El segundo, es una comunicación donde se recoge una versión en español de la primera publicación.

- Meeting (Birds of a Feather) about use and management of clusters of processors. Jose J. López-Espín, Javier Cuenca, Miguel Bernabeu, Emmanuel Jeannot and Luis Pedro García. Meeting on Optimization of Parallel Routines and Applications, 10-12 June 2007, Murcia, Spain. [55]

  En esta comunicación se describió las diferencias existentes, como usuario, entre realizar experimentos en Marenostrum al que se tuvo acceso para resolver grandes problemas de M.E.S. y desarrollarlos en máquinas más pequeñas como Rosebud o Kefren. Se habló del tiempo máximo de ejecución por usuario, prioridad de la cola a la que estén adjudicados los procesos, etc.

- A genetic algorithm for estimation of simultaneous equation models. Jose J. López-Espín and Domingo Giménez. 2nd International Workshop on Computational and Financial Econometrics, 19-21 June 2008, Neuchâtel, Switzerland. [56]

  Se analiza el algoritmo genético presentado en el capítulo 7. Se analizan los parámetros del genético y como afecta su variación al tiempo de ejecución. También se comparan los parámetros de criterio y diferentes tipos de cruce. Finalmente se incluye una versión paralela en memoria compartida, la cual es analizada experimentalmente.

- Genetic Algorithms for Simultaneous Equation Models. Jose J. López-Espín and Domingo Giménez. Proceedings International Symposium on Distributed Computing and Artificial Intelligence 2008, Advance in Soft Computing, 215-225, 2008. [50]

  Obtención de modelos de ecuaciones simultáneas mediante algoritmos genéticos. Jose J. López-Espín and Domingo Giménez. XXXI Congreso Nacional de Estadística e Investigación Operativa, 10-13 Febrero 2009, Murcia, Spain. [57]

  El primer trabajo es una publicación que extiende el trabajo anterior al algoritmo híbrido que incluye avance rápido en la mutación (incluida en el capítulo 7). También se desarrolla una versión paralela en memoria compartida, la cual es analizada experimentalmente. El segundo trabajo es una versión en español del primero.

9.3. Trabajos futuros

A la vista de los resultados obtenidos, se proponen algunas líneas de investigación (en algunas ya se ha empezado a trabajar) relacionadas y complementarias con el trabajo realizado en esta tesis, que se consideran interesantes a corto y medio plazo y que se podrían agrupar en cuatro direcciones:

- Resolución de M.E.S. Se han desarrollado unos algoritmos en memoria compartida y distribuida. Se plantea como línea futura a corto plazo desarrollar versiones en paralelo tanto en memoria compartida como distribuida de todos los algoritmos secuenciales, También versiones híbridas y en sistemas heterogéneos. Con lo cual se dispondrá de una serie de funciones que son de utilidad para científicos de diversos campos y en distintos sistemas.

- Árbol de Mínimo Coste. Se ha visto en el capítulo 6, que la versión del algoritmo que utiliza el Árbol de Mínimo Coste ahorra en tiempo de ejecución respecto a la que no lo utiliza. Sin embargo, el tiempo de obtención del árbol hace que dicha versión no sea útil en muchos casos salvo que se disponga previamente del árbol construido. Una línea de trabajo futura consiste en la mejora del algoritmo heurístico que obtiene dicho árbol. Como se ha visto en los experimentos del capítulo 6, el número de nodos que se añaden a un árbol es pequeño en comparación con el número de nodos que se rechazan. Por lo tanto, si se consiguieran obtener ciertas reglas o condiciones para reducir el número de nodos que se analizan, se reduciria considerablemente el tiempo de ejecución.

- Metaheurísticas. Se han desarrollado un algoritmo genético y un algoritmo híbrido en la obtención de M.E.S. a partir de los datos de las variables. Un siguiente paso sería desarrollar algoritmos utilizando otras metaheurísticas (Scatter Search, GRASP, Búsqueda tabú, etc.) para su comparación tanto en el tiempo de ejecución como en la bondad de la solución encontrada.

- Búsqueda exhaustiva. Se ha comprobado en el capítulo de aplicaciones (concretamente en la correspondiente a la preeclampsia) que el modelo obtenido por la aplicación puede ser mejorable. De hecho se ha obtenido mejor AIC con un algoritmo de dos ecuaciones que con otro de seis. Como trabajo futuro se podría estudiar la posibilidad de desarrollar algoritmos de búsqueda exhaustiva encontrando propiedades que reduzcan el espacio de búsqueda, o añadiendo restricciones dadas por los expertos.

- Modelos de Ecuaciones Simultáneas No Lineales. Actualmente se está avanzando mucho en la teoría de M.E.S. no lineales y, aún siendo una teoría relativamente reciente, ya se ve claramente la gran necesidad computacional que tiene. Como trabajo futuro a largo plazo se plantea el desarrollo y estudio de

algoritmos paralalelos para este tipo de modelos de la misma forma que se ha hecho para los M.E.S. lineales.

- Otras técnicas econométricas. También se plantea como trabajo a largo plazo el desarrollo y estudio de algoritmos paralelos para otras técnicas econométricas como, por ejemplo, los modelos VAR, Modelos de Datos Panel, etc.

Bibliografía

[1] Barcelona Supercomputing Center. www.bsc.es.

[2] Basic linear algebra subprograms. www.netlib.org/blas.

[3] Centro de Predicción Económica. www.ceprede.com.

[4] Instituto Universitario de Predicción Económica "Lawrece R. Klein". web.uam.es/otroscentros/klein.

[5] Jurgen a doornik. Oxmetrics. www.doornik.com.

[6] Linear algebra package. www.netlib.org/lapack.

[7] Numerical algorithms group. www.nag.co.uk.

[8] Parallel basic linear algebra subprograms. www.netlib.org/scalapack/pblas_qref.html.

[9] Project LINK Research Centre. www.chass.utoronto.ca/link.

[10] Quantitative micro software. www.eviews.com.

[11] Scalable linear algebra package. www.netlib.org/scalapack.

[12] Statistical package for the social sciences. www.spss.com.

[13] Top500. listado y estadísticas de las 500 supercomputadoras más potentes del mundo. www.top500.com.

[14] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Proceedings of the 2nd International Symposium on Information Theory*, pages 267–281. B. N. Petrov and F. Csaki Editions, 1973.

[15] Francisco Almeida, Domingo Giménez, José Miguel Mantas, and Antonio M. Vidal. *Introducción a la programación paralela*. Paraninfo Cengage Learning, 2008.

[16] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. J. Dongarra, J. Du Croz, A. Grenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995.

[17] E. J. Bedrick and C. L. Tsai. Model selection for multivariate regression in small samples. *Biometricsg*, 50:226–231, 1994.

[18] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

[19] P. M. Bosio, P. J. McKenna, R. Conroy, and C. O'Herlihy. Maternal central hemodynamics in hypertensive disorders of pregnancy. *Obstet Gynecol*, 94:978–984, 1999.

[20] H. Bozdogan and D. Haughton. Informational complexity criteria for regression models. *Computational Statistics and Data Analysis*, 28:51–76, 1998.

[21] Rohit Chandra, Ramesh Menon, Leo Dagum, David Kohr, Dror Maydan, and Jeff McDonald. *Parallel Programming in OpenMP*. Morgan Kauffman, 2001.

[22] Rohit Chandra, Ramesh Menon, Leo Dagum, David Kohr, Dror Maydan, and Jeff McDonald. *OpenMP C and C++ Application Program Interface*. OpenMP Architecture Review Board. `http://www.openmp.org/drupal/mp-documents/cspec20.pdf`, 2002.

[23] J. Choi, J. J. Dongarra, S. Ostrouchov, A. Petitet, D. Walker, and R. Clint Whaley. A proposal for a set of parallel basic linear algebra subprograms. Technical Report CS-95-292, Computer Science Dept. University of Tennesee, May. 1995.

[24] A. C. Darnell and J. L. Evans. *The limits of econometrics*. Gower, 1990.

[25] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. An extended set of fortran basic linear algebra subroutines. *ACM Transactions on Mathematical Software*, 14:1–17, 1988.

[26] J. Duvecot and L. Peeters. Very early changes in cardiovascular physiology. *Blackwell Science*, pages 3–32, 1998.

[27] R. J. Epstein. *A history of econometrics*. North-Holland, 1987.

[28] W. Feller. *Introducción a la teoría de probabilidades y sus aplicaciones*. Limusa, 1989.

[29] P. Foschi and E. J. Kontoghiorghes. Estimation of VAR Models: Computational aspects. *Computational Economics*, 21(1-2):3–22, 2003.

[30] Y. Fujikoshi and K. Satoh. Modified AIC and Cp in multivariate linear regression. *Biometrika*, 84(3):707–716, 1997.

[31] C. Gatu and E. J. Kontoghiorghes. Parallel algorithms for computing all possible subset regression models using the QR decomposition. *Parallel Computing*, 29(4):505–521, 2003.

[32] C. Gatu and E. J. Kontoghiorghes. A branch and bound algorithm for computing the best subset regression models. *Comput. Graph Statistic*, 15:1–19, 2006.

[33] F. Glover and G. Kochenberger. *Handbook of Metaheuristics*. Springer, 2003.

[34] G. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[35] A. Gonschorek, I. Lu, J. Halliwill, J. A. Beightol, J. A. Taylor, H. Painter, and D. Eckberg. Influence of respiratory motor neurone activity on human autonomic and haemodynamic rhythms. *Clinical Physiology*, 21(3):323–334, 2001.

[36] A. Gorobets. The optimal prediction simultaneous equations selection. *Economics Bulletin*, 36(3):1–8, 2005.

[37] W. Greene. *Econometric Analysis*. Prentice Hall, third edition, 1998.

[38] D. Gujarati. *Basic Econometrics*. McGraw Hill, 1995.

[39] M. Harchol-Balter and P. E. Black. Queueing analysis of oblivious packet-routing networks. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, pages 583–592, 1994.

[40] R. Henry, I. Lu, L. Beightol, and D. Eckberg. Interactions between CO2 Chemoreflexes and Arterial Baroreflexes. *Am. Journal of Physiology*, 274(43):H2177–H2187, 1998.

[41] M. Hofmann, C. Gatu, and E. J. Kontoghiorghes. Efficient algorithms for computing the best subset regression model for large-scale problems. *Computational Statistics and Data Analysis*, 52:16–29, 2007.

[42] Kai Hwang. *Advanced Computer Architecture: Parallelism, Scalability, Programmability, 1st edition*. McGraw-Hill, 1992.

[43] Kai Hwang and Faye A. Briggs. *Computer Architecture and parallel processing*. McGraw-Hill, 1984.

[44] Kai Hwang and Zhiwei Xu. *Scalable parallel computing: technology, architecture, programming*. McGraw-Hill, 1998.

[45] N. Kametas, F. McAuliffe, B. Cook, K. Nicolaides, and J. Chambers. Maternal left ventricular transverse and long-axis systolic function during pregnancy. *Ultrasound Obstet Gynecol*, 18:467–474, 2001.

[46] G. Kapetanios. Choosing the optimal set of instruments from large instrument sets. *Computational Statistics and Data Analysis*, 51(2):612–620, 2006.

[47] G. Kapetanios. Variable selection in regression models using nonstandard optimisation of information criteria. *Computational Statistics and Data Analysis*, 52(1):4–15, 2007.

[48] E. J. Kontoghiorghes. *Parallel algorithms for linear models: Numerical methods and estimation problems*. Advances in Computational Economics, 2000.

[49] J. J. López and D. Giménez. Message-passing two steps least square algorithms for simultaneous equations models. In *Proceedings International Symposium on Parallel Processing and Applied Mathematics*, pages 127–136. Springer, 2007.

[50] J. J. López and D. Giménez. Message-passing two steps least square algorithms for simultaneous equations models. In *Advance in Soft Computing*, pages 127–136. Springer, 2008.

[51] Jose J. López-Espín and Domingo Giménez. Solution of simultaneous equations systems by high performance methods. In *Computational Statistics and Data Analysis*, 2005.

[52] Jose J. López-Espín and Domingo Giménez. Solution of simultaneous equations models in high performance systems. In *XXIX Congreso Nacional de Estadística e Investigación Operativa*, 2006.

[53] Jose J. López-Espín and Domingo Giménez. Solution of simultaneous equations models in high performance systems. In *Workshop On State of the art in scientific and parallel computing*, 2006.

[54] Jose J. López-Espín and Domingo Giménez. Algoritmos de paso de mensajes para modelos de ecuaciones simultáneas. In *Meeting on Optimization of Parallel Routines and Applications*, 2007.

[55] Jose J. López-Espín and Domingo Giménez. Birds of a feather about use and management of clusters of processors. In *Meeting on Optimization of Parallel Routines and Applications*, 2007.

[56] Jose J. López-Espín and Domingo Giménez. A genetic algorithm for estimation of simultaneous equation models. In *2nd International Workshop on Computational and Financial Econometrics*, 2008.

[57] Jose J. López-Espín and Domingo Giménez. Obtención de modelos de ecuaciones simultáneas mediante algoritmos genéticos. In *XXXI Congreso Nacional de Estadística e Investigación Operativa*, 2009.

[58] I. Lu. *Applications of structural equation models: Case studies in biomedical and aerospace engineering research.* Joint Statistical Meeting, 2006.

[59] I. Lu and S. P. Jones. An investigation of the effect of cabin pressure altitude and level of SaO2 on passenger comfort - a simultaneous equation model. Technical Report M&CT-TECH-04-019, Boeing Technical Report Series, 2004.

[60] I. Lu, J. Peixoto, and W. A. Taam. Simultaneous equation model for air traffic in the New York area. In *The Seventh Air Transport Research Society World Conference Journal*, 2003.

[61] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate analysis.* Academic Press, 1979.

[62] M. Mitchell. *An Introduction to genetic algorithm.* MIT Press, 1998.

[63] M. S. Morgan. *The history of econometric ideas.* Cambridge University Press, 1990.

[64] Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP.* McGraw Hill, 2004.

[65] M. A. Quinones, C. M. Otto, M. Stoddard, A. Waggoner, and W. A. Zoghbi. Recommendations for quantification of doppler echocardiography: a report from the doppler quantification task force of the nomenclature and standards committee of the american society of echocardiography. *J Am Soc Echocardiogr*, 15:167–184, 2002.

[66] R. Ramanathan. *Introductory Econometrics with applications.* South westers, 2002.

[67] W. R. Ressler and M. S. Waters. Female earnings and the divorce rate: a simultaneous equation model. *Applied Economics*, 32:1889–1898, 2000.

[68] S. C. Robson, S. Hunter, R. J. Boys, and W. Dunlop. Serial study of factors influencing changes in cardiac output during human pregnancy. *Am J Physiol*, 256:1060–1065, 1989.

[69] L. Ruíz-Maya and F. J. Martín Pliego. *Estadística II: Inferencia.* AC, 1995.

[70] P. Shi and C. L. Tsai. A note on the unification of the Akaike information criterion. *J.R. Statist. Soc. B*, 60(3):551–558, 1998.

[71] Marc Snir and William Gropp. *MPI. The Complete Reference. 2nd edition.* The MIT Press, 1998.

[72] P. Winker and M. Gilli. Applications of optimization heuristics to estimation and modelling problems. *Computational Statistics and Data Analysis*, 47(2):211–223, 2004.

[73] P. Yanev, P. Foschi, and E. J. Kontoghiorghes. Algorithm for computing the QR decomposition of a set of matrices with common columns. *Algorithmica*, 39:83–93, 2004.

[74] P. Yanev and E. J. Kontoghiorghes. Efficient algorithms for estimating the general linear model. *Parallel Computing*, 32:195–204, 2006.