UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE QUIMICA

87
29

CONSTRUCCION AUTOMATICA DE SIMULADORES A PARTIR DE SU ESPECIFICACION MATEMATICA

T E S I S

CARLOS ROJAS GUZMAN

INGENIERO QUIMICO

1989

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

1 INTRODUCCION

El objetivo de este trabajo es implementar un sistema computacional capaz de construir simuladores de proceso en forma automática a partir de una descripción formal del modelo, es decir, a partir de un conjunto de ecuaciones que describen el comportamiento del proceso.

El análisis del modelo, el diseño automático de algoritmos para la simulación y la codificación del simulador, son las actividades que realiza este sistema computacional.

Este trabajo es parte de un proyecto más ambicioso que contempla la síntesis automática de procesos químicos. El proyecto global del grupo DIPAC (Diseño de Procesos Asistido por Computadora), dirigido por el Dr. Ignacio Ania y por el Dr.René Bañares incluye una interfaz gráfica para interacción con el usuario, la modelación automática del proceso, el diseño automático de algoritmos para la simulación de procesos y la construcción automática de simuladores. En esta tesis se presentan los primeros resultados de este proyecto.

Para lograr esto se utilizan técnicas de Inteligencia Artificial combinando el uso de matemáticas, teoría de

grafos, reglas heuristicas y teoria de conjuntos. Se usan dos lenguajes de programaciòn: COMMON LISP para el sistema que genera programas, y PASCAL, el lenguaje en el que se escriben automàticamente estos simuladores.

Es conveniente hacer notar la utilidad y el potencial que tiene la automatizaciòn de la construcciòn de simuladores de proceso para el Ingeniero Quimico. Una razòn es que el uso de simuladores facilita el anàlisis de los procesos. Algunas de las actividades en las que se aplica son el diseño de procesos quimicos, su control y optimizaciòn.

Dada la amplitud de los temas con los que tiene relaciòn este proyecto, se ha puesto màs atenciòn a la descripciòn de contribuciones originales que a las ideas presentadas en la literatura y sobre las que este trabajo se apoya. Aunque se supone que el lector tiene las bases necesarias, se presentan referencias en las que éstas se encuentran.

2.1 LA SIMULACION DE PROCESOS

La elaboración de modelos que traten de reproducir el comportamiento de fenómenos reales, que ayuden a comprenderlos, a explicarlos y a predecirlos, ha constituido una actividad fundamental de la ciencia.

La aplicación de principios científicos para resolver problemas prácticos es parte de lo que genéricamente se ha llamado Ingeniería. En el caso particular de la Ingeniería Química, son de interés procesos que involucran transformaciones físicas y químicas.

El comportamiento de estos procesos puede representarse utilizando modelos matemáticos que cuantifican las relaciones entre sus elementos y por eso es conveniente disponer de técnicas que sean capaces de resolver en forma eficiente estos modelos. Elaborar y resolver estos modelos se conoce como Simulación de Procesos.

La simulación de procesos facilita el diseño, operación, control y modificación de un proceso, así como el entrenamiento de operadores. Resultan evidentes las ventajas que tiene poder trabajar y experimentar con un modelo y no con un proceso real.

La complejidad de los modelos y la cantidad de cálculos necesarios para resolverlos hacen indispensable el uso de computadoras. Esta complejidad puede estar limitada por el nivel de comprensión de los fenómenos que ahí suceden o por la capacidad de almacenamiento y/o la velocidad de las computadoras en las que se implementen los programas encargados de resolver los modelos.

Un simulador es un programa codificado en algún lenguaje que pueda ser interpretado por alguna computadora. Para construirlo se necesita analizar el problema matemático a fin de desarrollar un procedimiento de cálculo y después escribir el programa. Estas dos etapas no son triviales. Debe notarse que el procedimiento de cálculo no es único y que simuladores con procedimientos distintos variarán en cuanto a flexibilidad, velocidad, y complejidad matemática. Estas diferencias se discutirán con más detalle al describir formas de simular un proceso.

2.2 ANTECEDENTES

Como consecuencia de un esfuerzo continuo por desarrollar herramientas más potentes, que permitan simular en forma más eficiente procesos industriales y al mismo tiempo aprovechar los avances tecnológicos de la computación, se hace cada vez más investigación con miras a automatizar lo más posible la tarea de implementar simuladores.

Una muestra de esta línea de investigación se lleva a cabo en el Instituto de Investigaciones Eléctricas, en Cuernavaca, Morelos, donde en el Departamento de Simulación se trabaja en el Proyecto Sistema de Desarrollo de Simuladores Industriales Asistido por Computadora, donde para apoyar la actividad de modelado se construye el conjunto de herramientas llamado "Construcción Automática de Simuladores Industriales" (CASI).

Los primeros resultados correspondientes a un sistema para la programación de la lógica de un diagrama de flujo fueron presentados como una herramienta para la programación automática por el Dr. René Bañares, [Bañares 88].

Entre los trabajos que se están realizando como parte de

este proyecto, deben mencionarse el de la M.en C. Nazira Guerrero, del ITESM-Morelos y el del Ing. José Salvador Villareal del IIMAS, UNAM, cuyas tesis de Maestría en Ciencias de la Computación, asesoradas por el Dr. Bañares se dirigen a la elaboración de sistemas fundamentados en la programación automática. El objetivo de los sistemas desarrollados en estos trabajos consiste en construir programas que serán parte de un sistema computacional del IIE y que simulan plantas relacionadas con la generación de energía.

Es importante también mencionar algunos lugares donde se hace investigación relacionada con Programación Automática e Inteligencia Artificial, como los Centros de Diseño Asistido por Computadora de la Universidad de Carnegie Mellon en E.U.A. y del Imperial College en Inglaterra así como en el Massachusetts Institute of Technology en E.U.A.

2.3 LA PROGRAMACION AUTOMATICA

Cuando se construye una herramienta novedosa, de cualquier tipo, que demuestra tener alguna utilidad práctica, no sólo es costumbre intentar predecir su alcance y sus limitaciones sino también tratar de ubicarla dentro del contexto del campo en el que se ha desarrollado.

Esta descripción comprende una clasificación conceptual de las técnicas y metodologías usadas para su desarrollo así como del área del conocimiento a la que pertenece. Es claro que dados los múltiples enfoques de expertos en diferentes campos, no es de esperarse que esta clasificación sea única.

Aunque parece que una ubicación de este tipo difícilmente contribuye a aumentar la utilidad práctica de la herramienta hipotética, objeto de esta discusión, la organización del conocimiento involucrado resulta una tarea conveniente. Con ella se favorece una mejor comprensión de las ideas y técnicas (usadas o generadas) y se facilita su aplicación en la resolución de otros problemas.

Son éstas las razones que motivan una breve revisión de algunos conceptos relacionados con este proyecto. Las ideas que

siguen se basan en trabajos de investigadores de este campo, recopiladas en [Wos 1985].

En forma general, el término 'programación automática' se usa para referirse al estudio e implementación de métodos cuyo objeto es automatizar una parte significativa del proceso de crear y mejorar programas computacionales. Una meta general de este campo consiste en facilitar el uso de las computadoras por medio de la automatización de este proceso. Objetivos más específicos incluyen

1. Aumentar la eficiencia en la producción de programas,
2. Disminuir sus costos,
3. Aumentar su confiabilidad,
4. Facilitar el manejo de sistemas complejos y
5. Permitir a los usuarios poner más atención a las soluciones de los problemas que a los detalles de la implementación.

Los términos 'síntesis de programas' y 'construcción de programas' se refieren a la construcción de un programa a partir de una especificación. Generalmente tiene una connotación de automatización parcial o total del proceso. El término 'programación automática' generalmente incluye otros aspectos del proceso de programación además de la síntesis del programa (como la administración del proyecto o la documentación del trabajo realizado).

Por último, el término 'herramientas de programación basadas en conocimientos' denota una actividad mucho más amplia que la

síntesis de programas y se refiere específicamente a herramientas computacionales que usan bases de conocimientos y técnicas de razonamiento automático para alcanzar sus metas.

Teóricamente el interés de la investigación en la síntesis de programas se dirige hacia el descubrimiento y articulación de los principios subyacentes a la creación de programas computacionales. El interés práctico se concentra en la forma de implementar sistemas que contengan este conocimiento y que lo apliquen para asistir al programador o al usuario final.

Debe mencionarse que existen diferentes grados de síntesis de programas, desde casos simples hasta el descubrimiento de algoritmos nuevos e interesantes.

En un principio, los lenguajes de computación eran lenguajes de máquina, y al trabajo que realizaban los compiladores se le llamó programación automática pues éstos automatizaban la producción de lenguaje de máquina o ensamblador, dado un programa en un lenguaje de alto nivel como especificación del resultado deseado. Actualmente el punto de partida para la programación automática es generalmente la especificación de un problema a un nivel formal cada vez más abstracto.

El sistema desarrollado en esta tesis está estrechamente relacionado con la programación automática, pues automatiza parte del proceso de creación de programas a partir de una especificación constituida por la descripción formal del modelo. En particular ésto es lo que se denomina síntesis de programas.

Una de las estrategias para usar computadoras para el razonamiento automàtico (en particular para la programaciòn automàtica) consiste en aprovechar la sintàxis y las reglas de la lògica deductiva como la describen Aristòteles, Russell y Whitehead o Church. Se extiende el formalismo cuando es necesario para representar conceptos ùtiles que no se expresan con facilidad. Sin embargo, se hace un esfuerzo por retener la consistencia lògica que estos formalismos proporcionan.

Existe otra lìnea de investigaciòn conocida como heurìstica: se usan los conocimientos disponibles para hacer y guiar inferencias probables sin importar si esos conocimientos son consistentes lògicamente o las inferencias son rigurosamente vàlidas. Este es el fundamento que se ha usado en muchas aplicaciones de la Inteligencia Artificial y que ha motivado todo el trabajo reciente en sistemas expertos. [Wos 1985].

En varias secciones del sistema desarrollado se utiliza el enfoque heurìstico, ya que no se dispone de algoritmos formales para resolver el problema.

2.4 LA INTELIGENCIA ARTIFICIAL

El sistema desarrollado en este trabajo utiliza técnicas de Inteligencia Artificial y expresiones simbólicas para representar modelos matemáticos de procesos químicos, su estructura y su análisis. Las usa también para diseñar algoritmos que encuentren soluciones al problema y para generar código computacional, es decir , escribir un simulador de procesos.

Es por eso que resulta conveniente presentar aunque sea en forma breve algunos conceptos relacionados con estas herramientas.

La Inteligencia Artificial se ha definido como el estudio de las facultades mentales por medio del uso de modelos computacionales [Charniak 1985].

Entre las areas de investigación de la Inteligencia Artificial se encuentran la creación de sistemas expertos para la resolución de problemas, la representación de conocimiento para simular el aprendizaje y el razonamiento, la construcción de programas que interactúen con las personas en lenguajes naturales o humanos, la creación de sistemas que sean capaces de construir a otros sistemas, la demostración automática de teoremas, la visión y el entendimiento del lenguaje hablado, y existen muchas

otras áreas más.

Para construir herramientas computacionales que sean capaces de realizar actividades similares a lo que consideramos un comportamiento inteligente, es necesario poder manejar información, no solamente en forma numérica , sino también simbólica.

Con un lenguaje simbólico, los datos y las actividades a realizar se representan con símbolos que pueden ser analizados para organizar, modificar o crear otros símbolos que a su vez representen datos o procedimientos.

El sistema desarrollado en este trabajo está escrito en LISP, el lenguaje más utilizado para investigación en Inteligencia Artificial [Charniak 1985]. Con objeto de que este sistema pueda ser utilizado en una gran variedad de computadoras se usa COMMON LISP, un dialecto reciente y ampliamente difundido del lenguaje de programación LISP, que constituye la base sobre la cual otros dialectos hacen extensiones particulares.

LISP es un lenguaje cuyos programas están estructurados en funciones que constituyen bloques independientes que pueden usarse unos a otros. Normalmente una función cumple su objetivo al satisfacer metas parciales, usando recursión y/o llamadas a varios niveles. Una de las ventajas de este lenguaje es su gran flexibilidad y la facilidad con que su sintaxis puede modificarse para adaptarse mejor a las necesidades y gustos del programador.

Además de estar orientado hacia la manipulación simbólica, LISP facilita la asociación de información con símbolos. La construcción de nuevas estructuras de datos es sencilla y para el tipo de trabajo necesario en este proyecto resulta muy adecuado.

2.5 EL PROBLEMA MATEMATICO: SISTEMAS GRANDES DE ECUACIONES NO LINEALES, DISPERSOS Y CON GRADOS DE LIBERTAD

Los sistemas de ecuaciones que resultan al construir modelos para procesos químicos se caracterizan por ser grandes y dispersos, por estar compuestos por ecuaciones frecuentemente no lineales y por tener un número mayor de variables que de ecuaciones.

Depende entonces de cada problema en particular el determinar cuáles de esas variables se conocen. Cuando el número de variables desconocidas es mayor que el número de ecuaciones el sistema no está completamente especificado, se tienen grados de libertad y es necesario fijar tantas variables como grados de libertad existan. Las variables fijadas se llaman variables de decisión y su selección tiene un efecto importante en la complejidad del procedimiento de cálculo a desarrollar. En el Apéndice II se ilustra este efecto con un ejemplo. Si el usuario no tiene ninguna preferencia pueden seleccionarse aquéllas que más simplifiquen el problema, sin embargo, es posible que no se disponga de suficiente información para dar una estimación o que sean precisamente las que se desea calcular. Se establece entonces un compromiso.

Encontrar el mejor conjunto de variables de decisión por búsqueda exhaustiva no es práctico dada la gran cantidad de combinaciones que se pueden formar, es decir, se trata de un problema de explosión combinatórica. Sin embargo, se han propuesto algoritmos que encuentran este conjunto, aunque no garantizan que sea el óptimo.

Después de esta selección se tiene ya un sistema cuadrado, es decir, un sistema en el que el número de ecuaciones es igual al de incógnitas. Se podría intentar resolver el sistema en forma simultánea iterando sobre todas las variables, o tratar de descomponer el problema en otros menores (encontrar una partición) y después resolver secuencialmente los bloques resultantes (cada uno es un sistema de ecuaciones no lineales menor que el original), o tal vez para cada bloque buscar variables de rompimiento para evitar la resolución de sistemas simultáneos.

En el sistema implementado en este proyecto, cuando el sistema es cuadrado se busca una partición y se establece un orden de solución para los bloques (orden de precedencia).
El código computacional que se genera resuelve numéricamente cada uno de los subsistemas propuestos, en el orden encontrado.

2.6 ALTERNATIVAS EN LA SIMULACION DE PROCESOS

Los simuladores de procesos pueden agruparse en tres grandes grupos de acuerdo con la forma en la que resuelven las ecuaciones que componen el modelo.

Una de èstas consiste en agrupar las ecuaciones en conjuntos tales que cada uno corresponda a una unidad del proceso. Se escribe una subrutina independiente para cada unidad y asi un proceso puede simularse al unir varios bloques en un orden apropiado. A esta técnica se le conoce como simulación secuencial modular y frecuentemente se tiene otra subrutina que coordina a las demàs. Cada mòdulo puede calcular las propiedades de las corrientes de salida de un equipo dadas las propiedades de las de entrada, las condiciones de operación y los parámetros del equipo.

Esto tiene algunas ventajas: es relativamente fácil escribir un mòdulo, pues el problema está claramente definido, y al escribir las unidades se puede asegurar que sean robustas (que la mayor parte de las veces sean capaces de encontrar la solución del problema). Esto es posible pues siempre se resuelve un problema con la misma estructura.

Sin embargo, la principal desventaja es que es poco flexible, la distribución entre variables dependientes (desconocidas) e independientes (especificadas) se ha fijado de antemano, y es obvio que estos conjuntos no siempre coincidirán con los del problema planteado, es decir, las variables conocidas no siempre serán las mismas. Cuando las variables conocidas no son las variables de entrada de algún equipo, (sino que por ejemplo se especifica una variable de salida), es necesario resolver todo el sistema de ecuaciones varias veces modificando los valores de las variables de entrada hasta que el valor calculado para esa variable de salida sea suficientemente cercano al especificado. Otra desventaja es que deben suponerse las propiedades de una corriente para cada ciclo físico (recirculación) del proceso e iterar sobre los valores de esas propiedades.

Otra alternativa conocida como simulación modular simultánea [Westerberg 1979] consiste en escribir los módulos como en la simulación secuencial modular. La diferencia fundamental es que se debe escribir un segundo módulo para cada unidad el cual debe ser capaz de aproximar el comportamiento del equipo por medio de una combinación lineal de las variables de entrada. Este módulo es una aproximación lineal del comportamiento del equipo en las vecindades del vector que se acerca iterativamente a la solución. Para su construcción se utiliza al módulo original que describe el comportamiento (frecuentemente no lineal) del equipo. Resolver sistemas lineales es más sencillo y éstos se van mejorando iterativamente.

Las limitaciones de la simulación modular secuencial se pueden superar al resolver simultáneamente todas las ecuaciones del modelo. Esta técnica, 'orientada a la resolución simultánea de ecuaciones' tiene una flexibilidad mucho mayor pero representa un problema más complicado desde el punto de vista matemático y computacional, ya que implica resolver un sistema grande de ecuaciones no lineales simultáneamente.

2.7 ESTRATEGIAS EN LA RESOLUCION SIMULTANEA DE LAS ECUACIONES NO LINEALES

Para resolver el sistema en forma simultánea se pueden utilizar distintas estrategias. Una de ellas consiste en subdvidir el problema en partes menores y de menor complejidad. Se trata de encontrar subsistemas o bloques que puedan resolverse en forma independiente en algún orden especial.

Si algunas de las incógnitas del problema se conocieran sería posible reducir considerablemente el esfuerzo necesario para encontrar la solución. Este es el fundamento de otra estrategia que puede combinarse con la primera. Estas incógnitas aparecen en por lo menos dos ecuaciones que consecuentemente deben resolverse en forma simultánea (esto puede ser pero no necesariamente es consecuencia de la existencia de una corriente que recircula en el proceso formando un ciclo físico).

Es posible resolver esas ecuaciones individualmente si al suponer el valor de una de las variables desconocidas existe y se encuentra un orden tal que permita resolver cada variable a partir de una ecuación en la cual sea incógnita única. La última ecuación que se resuelve es aquélla en la que la única variable cuyo valor se desconoce es la que se supuso al

principio. Cuando el valor calculado sea suficientemente cercano al supuesto, termina la secuencia cíclica de cálculos.

De esta forma se itera sobre una sola variable en vez de hacerlo sobre todo el conjunto del sistema y se necesita dar una estimación para una sola variable en vez de darla para todo el vector del sistema. Estas variables se conocen como variables de rompimiento ya que descomponen o rompen ciclos de cálculos. Encontrar el conjunto mínimo de variables que permita resolver individualmente en algún orden todas las ecuaciones no es trivial. En el sistema aquí implementado no se utiliza esta técnica ya que no es claro si la cantidad de trabajo necesaria para encontrar un conjunto de variables de rompimiento y resolver iterativamente los ciclos encontrados es menor que la necesaria para resolver simultáneamente los subsistemas encontrados al descomponer el sistema. Sin embargo, como una posible ampliación de este trabajo, pueden implementarse la búsqueda de variables de rompimiento para los subsistemas y comparar la eficiencia de los dos métodos.

3 EL SISTEMA GENERADOR DE SIMULADORES

3.1 GENERALIDADES

Tanto la construcción de procedimientos para ser utilizados por simuladores como la codificación de los mismos se hace en forma automática en este trabajo. Se describirán primero en forma general las etapas del sistema desarrollado y posteriormente se desglosará cada una con más detalle.

El sistema computacional desarrollado parte del modelo que representa al proceso para el cual construirá un simulador. Supone que el modelo está formado por ecuaciones algebraicas no lineales, consistentes y no redundantes. El hecho de que formen un sistema disperso no es indispensable pero favorece la descomposición en subsistemas de ecuaciones. Como una gran parte de los modelos de procesos químicos están constituidos por sistemas dispersos, se usan técnicas que aprovechan esta distribución.

La primera etapa consiste en interactuar con el usuario para definir completamente el problema, es decir, saber cuáles son las variables que en ese caso particular se conocen, lo que es posible gracias a la flexibilidad que permite la resolución simultánea de ecuaciones.

Como siguiente paso se analiza la estructura del sistema de ecuaciones resultante y si el número de ecuaciones no es igual al de incógnitas el sistema selecciona un conjunto de variables a las que llamaremos "variables de decisión",con tantos elementos como grados de libertad existan.

Este conjunto de variables es sugerido al usuario del sistema y éste tiene la opción de hacer otra selección si lo cree conveniente.

Ya con un sistema cuadrado, un requisito para encontrar una partición y subdividir el problema original en bloques menores consiste en encontrar una relación biunívoca entre los conjuntos de variables y funciones. A esta relación la llamaremos "asignación de variables de salida".

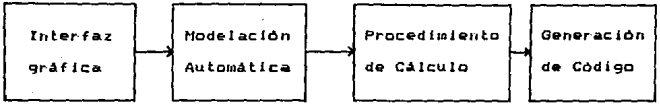
Teniendo una asignación de variables de salida se busca una subdivisión del problema en sistemas menores, a la que llamaremos "partición", y al mismo tiempo se establece una secuencia de cálculo para los bloques a la que llamaremos "orden de precedencia".

Finalmente, usando el procedimiento de cálculo desarrollado se genera automáticamente el simulador, es decir, un programa en PASCAL, específico al problema planteado, con comentarios útiles al usuario.

3.2 ARQUITECTURA DEL SISTEMA

Como se dijo, el sistema desarrollado está escrito en COMMON LISP [Steele 1984] y se ha construido en bloques para facilitar su manejo.

Conviene recordar que aunque este sistema puede funcionar en forma autónoma al proporcionarle el modelo de algún proceso, se planea extenderlo para que, contando con un módulo que efectúe la modelación automática, la información de salida de este sistema sea la de entrada del sistema aquí propuesto.



El programa principal es una función cuyo objetivo es codificar un simulador en PASCAL [Jensen 1974] a partir del modelo que ha sido codificado como otra función. Los únicos parámetros de esta función son el nombre del modelo y el del archivo en el que se encuentra.

Para cumplir con este objetivo, se subdivide el trabajo en las siguientes etapas:

1. Definición del problema y construcción de una estructura de

datos utilizando listas organizadas en varios niveles.

2. Selección de variables de decisión.
3. Búsqueda de una asignación de variables de salida.
4. Construcción de una partición y un orden de precedencia.
5. Generación de código computacional usando el algoritmo desarrollado en las etapas anteriores.

Cada bloque cumple con uno de estos objetivos al subdividir el trabajo en metas parciales que, a su vez, son llevadas a cabo por otros bloques menores.

La función principal lee del disco los bloques de funciones y una vez que han quedado definidas coordina las siguientes actividades:

(a) El análisis del modelo,
(b) El diseño de un procedimiento de cálculo para resolverlo,
(c) La creación del simulador y
(d) Su almacenamiento como otro archivo listo para ser compilado y usado.

Se tiene también un módulo con funciones auxiliares que pueden ser utilizadas por cualquier otra función. Estas incluyen, entre otras, funciones para hacer operaciones con conjuntos, para manejar estructuras de datos y para rastrear funciones por bloques.

3.2.1 REPRESENTACION DEL MODELO Y DEL ANALISIS

Aprovechando la orientación simbólica de LISP, se almacena la información relacionada con el modelo clasificada en conjuntos identificados con una propiedad del modelo (algunas de las propiedades del modelo son el número de funciones que tiene, el conjunto de variables de decisión sugerido o la ruta seguida durante un algoritmo al recorrer una representación gráfica de las relaciones entre funciones). Entre las propiedades que le asignamos a este simbolo está una estructura a la que llamaremos marco.

La implementación de estas estructuras usa listas asociadas a diferentes niveles. Para este trabajo se extiende el conjunto de funciones para manejo de marcos que se propone en [Winston 1984] y se usa esta organización como base para el almacenamiento de información en todo el sistema.

Aunque no todas las ventajas que ofrece un marco se utilizan, al usar esta estructura se facilita la extensión de sus posibilidades. El uso convencional de los marcos se modifica para adecuarlo a las necesidades particulares de este trabajo, en especial para el manejo de los algoritmos para este trabajo.

En la sección 4 y en el Apéndice III se muestran representaciones de modelos de proceso que consisten en el conjunto de ecuaciones que lo forman y la lista de variables que aparecen en cada ecuación.

Como otras propiedades se almacenan listas ya organizadas cuya principal función es conservar un registro del progreso del análisis y de las opciones intermedias elegidas. Los elementos de estas listas tienen un manejo mínimo. Entre ellas están los conjuntos de incógnitas y funciones que se reducen al definir el problema y al seleccionar variables de decisión. También así se almacena un registro de rutas tomadas por algoritmos que usan representaciones con redes y búsquedas en grafos.

3.2.2 ANALISIS DE ESTRUCTURA

En esta etapa el sistema interactúa con el usuario para definir el problema particular dando las variables que conoce o desea determinar inicialmente. Estas variables y las funciones que no tienen variables desconocidas, son eliminadas de los conjuntos que serán analizados.

La representación de estos dos conjuntos y sus relaciones se ha hecho por medio de gráficas bipartitas, matrices de adyacencia y matrices estructurales o de incidencia según la literatura [Ramirez 1972].

En este sistema se representa información equivalente a la de una matriz de incidencia usando listas dentro de un marco (la estructura de un marco consiste en listas dentro de listas, los niveles tienen los siguientes nombres : marco, sección, faceta y valor). Inicialmente se tiene una lista de variables asociada a cada función (valores de una faceta del marco) y para hacer más eficiente la representación y el manejo de redes al usar teoría de grafos, se construye una lista para cada función (que contiene sus variables), y una para cada variable (donde se encuentran las funciones en las que aparece).

3.2.3 SELECCION DE VARIABLES DE DECISION

Con objeto de tener un sistema cuadrado cuando el modelo tiene grados de libertad, se selecciona un conjunto de variables de decisión que posteriormente reduzca el tamaño de los bloques en la partición. Se usa un algoritmo inicialmente propuesto como parte de una estrategia para escoger variables de rompimiento [Ramirez 1972].

Frecuentemente es necesario tomar ciertas decisiones a lo largo del proceso de selección, entre varias opciones escogidas como las mejores. Aunque bajo los criterios usados parecen equivalentes, se cree que estas reglas se pueden ampliar para poder comparar estas opciones. Es por eso que se ha construido en forma independiente esta selección de opciones, hasta ahora equivalentes, para poder implementar con facilidad otros criterios.

Originalmente durante el avance del algoritmo se eliminaban funciones y variables de la matriz de incidencia; en esta implementación particular se van sacando de dos listas construidas para este fin. Cada vez que se elimina un elemento de alguna de ellas, se añade éste a una lista y así se va construyendo la "historia de la selección". Cuando la selección

termina, esta historia se convierte en una propiedad del simbolo que representa al modelo.

En otra lista, parte del marco, se van registrando las variables de decisión sugeridas y el nivel de jerarquia en el que fueron encontradas.

Este conjunto de variables es sugerido al usuario del sistema quien tiene la opción de hacer la selección final si lo cree conveniente.

Debe notarse que la selección de las variables de decisión tiene un efecto importante sobre el procedimiento de cálculo que se propone en forma automática y sobre el simulador generado. La descomposición del sistema de ecuaciones original en subsistemas menores es distinta por lo que la eficiencia del simulador varia. Este efecto puede apreciarse en el Apéndice II.

3.2.4 ASIGNACION DE VARIABLES DE SALIDA

Antes de poder encontrar una partición para el sistema de ecuaciones, es necesario asignar a cada ecuación, una variable que aparezca en ella. Esta relación entre el conjunto de variables y el de ecuaciones debe ser biunívoca.

Steward demostró que cuando esta asignación de variables de salida no existe las ecuaciones son "estructuralmente singulares", es decir, existe un subconjunto de [q] ecuaciones que involucran [p] variables (p menor que q) que hace [q-p] ecuaciones redundantes o inconsistentes con las [p] ecuaciones restantes, [Westerberg 1979 APUD EN Steward 1962]. Debe notarse además que esta asignación no es única.

Para realizar una asignación como la descrita se modifica un algoritmo descrito en [Westerberg 1979] en el que se representan las relaciones entre funciones , variables una matriz de incidencia.

El algoritmo original propone seleccionar la variable que aparece en menos ecuaciones , de esas ecuaciones en las que aparece, escoger la que tiene menos variables. La modificación

que se le hace en este trabajo al algoritmo se basa en dos observaciones:

Una es que con los criterios usados las selecciones descritas no son únicas, es decir, cuando más de una ecuación o variable cumple con tener el mínimo número de incidencias, se tiene que tomar alguna sin tener justificación para no tomar otra que cumpla con el requisito. De la descripción de los criterios se desprende que conviene asignar una par cuyos dos elementos tengan un mínimo de incidencias. Esto es lógico si se trata de que cada par elegido sea el que más contribuya a aumentar la probabilidad de que todas queden asignadas.

La segunda observación, consecuencia de la primera, es que un criterio como el descrito no siempre encontrará el par que tiene menos incidencias por hacerlo en dos pasos. Es por eso que aquí se escoge uno de los pares que tenga la "mínima frecuencia combinada", definida ésta como la suma de las incidencias de la variable y de la función.

Una ventaja es que el algoritmo modificado podrá encontrar una asignación completa con mayor frecuencia. Como es necesario calcular la frecuencia combinada para todas las posibles combinaciones, se organiza la información correspondiente al grafo en el marco en forma tal que el acceso y la eliminación de elementos de las listas sea rápido y eficiente. Se construye para cada función (faceta del marco) una lista con las variables que aparecen en ella (valores de esa faceta). Y para cada variable

(faceta) se construye una lista con sus funciones (valores).

Debe notarse que aunque no se encuentra UN mejor par en cada asignación, el número de opciones que queda es menor y la probabilidad de que la asignación sea completa aumenta.

El criterio de selección de opciones equivalentes se ha implementado como una función independiente para poder extender o cambiar el criterio con facilidad.

Aunque se reduce la frecuencia con la que la asignación no es completa, se presentan casos en los que es necesario recurrir a un algoritmo auxiliar para que no queden elementos sin asignar por no tener incidencias comunes. Su objetivo es cambiar algunas asignaciones para que a cada ecuación le corresponda una variable. A la ruta seguida para hacer este cambio se le conoce como "Camino de Steward" y se ha implementado como se describe en [Westerberg 1979].

3.2.5 REARREGLO DE LA ASIGNACION

En muchos problemas estudiados en Inteligencia Artificial es necesario tomar decisiones entre opciones que no pueden compararse con precisión para determinar cuál es la mejor. En esos casos se pueden hacer estimaciones que aunque no garantizan una selección óptima, resultan mejores que una selección aleatoria. La teoría de estas estimaciones es la teoría de búsquedas. [Charniak 1985].

La búsqueda de un Camino de Steward ilustra este tipo de actividad y es por eso que se describirá con más detalle. Cuando el algoritmo que sigue reglas heurísticas falla, se recurre a una búsqueda sistemática.

Como todo problema de búsqueda, éste se caracteriza por un estado inicial y la descripción de una meta. El estado inicial consiste en una asignación incompleta (registrada en el marco como la faceta "asignación" y cuyos valores son los pares asignados), y dos listas (variables y ecuaciones no asignadas). Se tienen operadores que transforman un estado en otro, que se espera esté más cerca de la meta que el anterior. En este caso no es necesario encontrar la solución óptima, pues cualquier solución (los recorridos exitosos no son únicos) permite cambiar

y completar la asignación.

La distinción entre buscar un estado o una trayectoria conceptualmente tiene poca importancia ya que la definición del estado puede incluir al camino recorrido para alcanzarlo. [Charniak 1985].

El conjunto de todos los estados que pueden encontrarse aplicando operadores a partir del estado inicial se llama espacio de búsqueda y puede representarse como un árbol. Disponer de una función capaz de estimar la distancia entre cualquier estado y la meta más cercana es de gran utilidad; sin embargo, en este problema se sabe de la existencia de alguna solución hasta que se encuentra. Por eso se recurre a una búsqueda sistemática que puede ser horizontal o vertical [Charniak 1985]; en la primera se prueban primero todos los sucesores de un nodo antes de pasar al siguiente nivel mientras que en la vertical o con "prioridad a profundidad" se sigue la siguiente regla: al escoger el siguiente paso se selecciona un camino que se origine en el último nodo visitado que todavía tenga salidas sin explorar. [Tarjan 1972]. Es este criterio el que se sigue al buscar un camino de Steward.

Cuando llega a un nodo que no tiene sucesores útiles (por no existir éstos o por ser ya parte de la trayectoria seguida) la representación de la gráfica del árbol se va modificando cerrándose caminos no viables.

Como la búsqueda es sistemática, termina hasta que encuentra

una solución o hasta que ha recorrido todo el espacio de búsqueda sin encontrarla concluyendo que no existe tal solución.

En el Apéndice I se muestra la implementación de esta búsqueda y posteriormente se presenta un caso particular.

3.2.6 PARTICION Y ORDEN DE PRECEDENCIA

Como el número de operaciones necesarias para resolver simultáneamente un sistema de ecuaciones aumenta mucho más rápido que el número de ecuaciones, se puede evitar una gran parte del trabajo si se aprovecha el que el sistema sea disperso. Esto puede lograrse al resolver subsistemas de las ecuaciones. Cada bloque en una partición es el mínimo conjunto de ecuaciones que incluye todas las ecuaciones que necesariamente deben resolverse simultáneamente. Esta asociación en bloques puede verse como la construcción de una clase de equivalencia.

La partición es independiente de la asignación de variables de salida ya que cualquier otra relación podría obtenerse al intercambiar las asignaciones alrededor de los ciclos formados, y los ciclos (formados en el grafo que representa relaciones funcionales) están confinados a los bloques.[Steward 1965 APUD EN Steward 1962]. La partición es única pero el orden de precedencia no. [Westerberg 1979].

Se ha implementado el algoritmo descrito en [Christensen 1969], el cual aunque propuesto para organizar cálculos en sistemas de simulación modular secuencial, tiene la misma estructura que la de este problema. En el grafo original, los

nodos representan equipos o unidades y los arcos que los unen representan corrientes de material entre equipos. En esta implementación los nodos representan funciones y los arcos relaciones entre funciones, cada una está unida con aquéllas en las que aparece su variable asignada.

El grafo descrito se representa como la propiedad GRAFICA del modelo y consiste en una lista cuyo primer elemento es otra lista que contiene a todas las funciones que forman el nodo. Los demás elementos son las funciones hacia las que tiene arcos dirigidos el nodo. Esta organización permite el que al progresar la búsqueda, el grafo sobre el que ésta se hace pueda modificarse. La propiedad TRAYECTORIA registra el recorrido del grafo hasta que se encuentra un elemento repetido (detectando un ciclo) o un nodo sin salidas o sin entradas (que puede sacarse del grafo).

Al avanzar el algoritmo y encontrarse ciclos, todos los nodos que forman parte del ciclo se funden en un nuevo nodo. Los ciclos sin entradas o sin salidas simplemente se sacan del conjunto y se añaden a otras listas, INICIO y FINAL, las que al unirse formarán la lista cuyos elementos son los bloques encontrados, en un orden adecuado para calcularlos secuencialmente.

Este procedimiento de cálculo propuesto es codificado en forma automática en la siguiente etapa.

3.2.7 GENERACION DE CODIGO

Al encontrar una partición y un orden de precedencia se ha desarrollado un procedimiento de cálculo que debe ser implementado en un simulador. Lo que el simulador debe hacer es resolver numéricamente en el orden propuesto una serie de sistemas algebraicos de ecuaciones no lineales.

La función GENERA-CODIGO del sistema escribe un simulador ad hoc para el modelo y para la definición particular del problema. Además, los simuladores escritos por el sistema generador contienen comentarios útiles al usuario.

Su interacción con el usuario consiste en pedirle los valores que tienen las variables conocidas (el sistema generador de código no pide estos valores para que el simulador sea más flexible y general), valores para las variables de decisión y el usuario tiene la opción de suministrar una primera estimación al resto de las variables para iniciar los cálculos iterativos en cada bloque. Se piensa extender esta parte para que pueda hacerse esta estimación automática dependiendo del tipo de variable (ya que las fracciones mol toman valores órdenes de magnitud menores que las temperaturas absolutas), lo que es fácil de implementar usando marcos, al asociar a cada variable, el tipo de variable a

la que pertenece (por ejemplo fracción mol).

Los simuladores se escriben en Pascal, porque ofrece las siguientes ventajas: es un lenguaje estructurado y tipificado, existe un control sobre la integridad semántica y además es recursivo. Para que sea compatible con la mayor parte de las versiones del lenguaje, se usan las bases establecidas en [Jensen 1974], sobre las que implementaciones comerciales hacen extensiones particulares. Aunque se espera que éstas cumplan con un mínimo de posibilidades, no todas lo hacen por lo que se ha evitado usar 'funciones' como parámetros de otras 'funciones', porque no están implementadas en las versiones del lenguaje ampliamente distribuidas, a pesar de que su uso mejoraría la arquitectura de los simuladores.

El método numérico usado se ha escrito como una función independiente para facilitar su cambio o modificación. Se planea ampliar el sistema para que seleccione el método más adecuado para cada caso particular. El método usado ahora es el de Newton Raphson en N dimensiones, el sistema de ecuaciones lineal cuya solución es el conjunto de incrementos a cada variable se resuelve por descomposición en dos matrices triangulares inferior y superior usando el algoritmo de Crout con pivotes implícitos [Press 1986].

Para reducir al mínimo el uso de memoria del simulador y aprovechando que los bloques se resuelven simultáneamente, se definen las matrices y vectores para el tamaño del mayor

subsistema y se usa sòlo la parte necesaria en los demàs. Es por eso que la parte principal del programa antes de usar la funciòn RESUELVE; establece a las estimaciones iniciales como los primeros valores del vector soluciòn y una vez resuelto el bloque, asigna la soluciòn al nombre de la variable.

El procedimiento RESUELVE utiliza al procedimiento EVALUA, encargado de proporcionar el valor aproximado de las derivadas y el de las funciones evaluadas en el màs reciente vector soluciòn. Este último selecciona a la secciòn de còdigo que representa al sistema que se està resolviendo en ese momento.

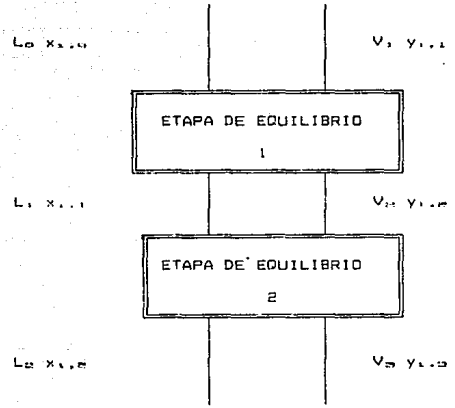
Si el usuario lo desea, el sistema puede escribir una secciòn de "Historia del Anàlisis y de la Elaboraciòn del Simulador", con resultados intermedios y decisiones tomadas en el curso de su construcciòn. Estos incluyen las trayectorias seguidas en los algoritmos que usan redes y grafos, las variables de decisiòn sugeridas por el programa y las seleccionadas por el usuario.

4 Un ejemplo completo

Con objeto de ilustrar la construcción automática de un simulador, se presentan algunas etapas intermedias seguidas por el sistema computacional MP (Meta Programa) durante el análisis de un problema y la generación del simulador correspondiente.

El ejemplo que sigue consiste en un equipo de absorción que opera a contracorriente, cuyo objetivo es separar mezclas, aprovechando la distinta afinidad química que los componentes de la mezcla tienen con solventes diferentes. Este equipo tiene dos etapas de equilibrio y las mezclas que se separan tienen tres componentes. Un esquema del equipo se muestra en la figura 1.

El modelo está formado por 24 ecuaciones que contienen 31 incógnitas y se muestra en la figura 2.



EQUIPO PARA ABSORCION MULTICOMPONENTE
CON DOS ETAPAS DE EQUILIBRIO

Figura 1

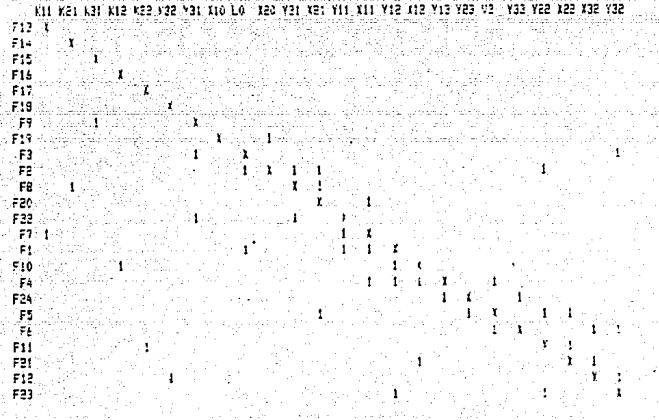
```
;DESCRIPCION FORMAL DEL MODELO ABSORBEDOR

                                        ;La estructura de un modelo similar
                                        ;es analizada en [Westerberg 1979]

                                        ;Para ilustrar ADEMAS del ANALISIS de
                                        ;estructura, la SOLUCION NUMERICA del
                                        ;modelo, se han suprimido algunas
                                        ;simplificaciones

(DEFUN ABSORBEDOR ()                    ;ABSORBEDOR de 2 ETAPAS y 3 COMPONENTES
  (SETF (GET 'ABSORBEDOR 'FRAME)
    '(ABSORBEDOR (F1 (VARIABLES I0 V2 L1 V1 X10 Y12 X11 Y11 )
                     (FORMULA (X10XL0+Y12XV2-X11XL1-Y11XV1) ) )
                 (F2 (VARIABLES I0 V2 L1 V1 X20 Y22 X21 Y21)
                     (FORMULA (X20XL0+Y22XV2-X21XL1-Y21XV1) ) )
                 (F3 (VARIABLES I0 V2 L1 V1 X30 Y32 X31 Y31)
                     (FORMULA (X30XL0+Y32XV2-X31XL1-Y31XV1)))
                 (F4 (VARIABLES L1 V3 L2 V2 X11 Y13 X12 Y12 )
                     (FORMULA (X11XL1+Y13XV3-X12XL2-Y12XV2) ) )
                 (F5 (VARIABLES L1 V3 L2 V2 X21 Y23 X22 Y22 )
                     (FORMULA (X21XL1+Y23XV3-X22XL2-Y22XV2) ) )
                 (F6 (VARIABLES L1 V3 L2 V2 X31 Y33 X32 Y32)
                     (FORMULA (X31XL1+Y33XV3-X32XL2-Y32XV2)))
                 (F7 (VARIABLES Y11 K11 X11)
                     (FORMULA (Y11-K11*X11)))
                 (F8 (VARIABLES Y21 K21 X21)
                     (FORMULA (Y21-K21*X21)))
                 (F9 (VARIABLES Y31 K31 X31)
                     (FORMULA (Y31-K31*X31)))
                 (F10 (VARIABLES Y12 K12 X12)
                     (FORMULA (Y12-K12*X12)))
                 (F11 (VARIABLES Y22 K22 X22)
                     (FORMULA (Y22-K22*X22)))
                 (F12 (VARIABLES Y32 K32 X32)
                     (FORMULA (Y32-K32*X32)))
               ;F13 a F18 son Keq(T,P=250 psia)
               (F13 (VARIABLES T  K11) C6
                    (FORMULA (K11-0.093+15.39E-4*T-10.37E-6*T*T+0.159E-8*T*T*T)))
               (F14 (VARIABLES T  K21) C3
                    (FORMULA (K21-0.84+46.6E-4*T-49.4E-6*T*T+3.033E-8*T*T*T)))
               (F15 (VARIABLES T  K31) C4
                    (FORMULA (K31+0.177-49.5E-4*T+4.15E-6*T*T-2.22E-8*T*T*T)))
               (F16 (VARIABLES T  K12) C6
                    (FORMULA (K12-0.093+15.39E-4*T-10.37E-6*T*T+0.159E-8*T*T*T)))
               (F17 (VARIABLES T  K22) C3
                    (FORMULA (K22-0.84+46.6E-4*T-49.4E-6*T*T+3.033E-8*T*T*T)))
               (F18 (VARIABLES T  K32) C4
                    (FORMULA (K32+0.177-49.5E-4*T+4.15E-6*T*T-2.22E-8*T*T*T)))
                    (F19 (VARIABLES X10 X20 X30)
                         (FORMULA (X10+X20+X30-1)))
                    (F20 (VARIABLES X11 X21 X31)
                         (FORMULA (X11+X21+X31-1)))
                    (F21 (VARIABLES X12 X22 X32)
                         (FORMULA (X12+X22+X32-1)))          Figura  2
                    (F22 (VARIABLES Y11 Y21 Y31)
                         (FORMULA (Y11+Y21+Y31-1)))
                    (F23 (VARIABLES Y12 Y22 Y32)                43
                         (FORMULA (Y12+Y22+Y32-1)))
                    (F24 (VARIABLES Y13 Y23 Y33)
                         (FORMULA (Y13+Y23+Y33-1)))  )))
```

Para este problema solamente se conoce el valor de una variable, T. Con objeto de tener un sistema cuadrado, es necesario fijar el valor de algunas incógnitas. Para seleccionarlas se usa un algoritmo inicialmente propuesto como parte de una estrategia distinta por [Ramirez 1972].

El conjunto de variables sugerido al usuario ha sido el siguiente:
X21, X31, X20, X30, L1, V2, L0, V1, V3 y L2
y de éste se seleccionaron L2, X30, L1, V2, V1 y X31.

Debe notarse que la selección de estas variables conocidas como variables de decisión tiene un efecto importante sobre el procedimiento de cálculo que en forma automática se propone para el simulador.

Antes de poder encontrar una partición para el sistema de ecuaciones, es necesario asignar a cada ecuación, una variable que aparezca en ella. Para realizar una asignación como la descrita se modifica un algoritmo propuesto por [Westerberg 1979] en el cual se representan las relaciones entre funciones y variables con una matriz de incidencia. En la figura 3 se muestra la matriz de incidencia una vez que se han eliminado las variables de decisión.



MATRIZ DE INCIDENCIA PARA EL MODELO ABSORBEDOR

Figura 3

La asignación se lleva a cabo por medio de una búsqueda heurística y en los casos en los que no tiene éxito, se recurre a una búsqueda sistemática. Este tipo de técnicas son frecuentes en Inteligencia Artificial y se describen en [Charniak 1985]. La búsqueda sistemática se realiza sobre el espacio de búsqueda ya reducido, se conoce como Camino de Steward y en este ejemplo no fue necesaria. [Steward 1965] y [Steward 1962].

Los pares (variable,función) formados en esta asignación son los siguientes:

(K11,F13), (K21,F14), (K31,F15), (K12,F16),
(K22,F17), (K32,F18), (Y31,F9), (X10,F19),
(L0,F3), (X20,F2), (Y21,F8), (X21,F20),
(Y11,F22), (X11,F7), (Y12,F1), (X12,F10),
(Y13,F4), (Y23,F24), (V3,F5), (Y33,F6),
(Y22,F11), (X22,F21), (X32,F12) y (Y32,F23).

La representación gráfica de esta asignación se muestra en la figura 3, marcando con una equis los pares asignados.

Para encontrar una partición, se ha implementado el algoritmo descrito en [Christensen 1969], el cual aunque propuesto para organizar cálculos de sistemas de simulación modular secuencial, tiene la misma estructura que la de este tipo de problemas. En el grafo original que se muestra en la figura 4, los nodos representan ecuaciones y los arcos que los unen relaciones entre ecuaciones. Cada ecuación está unida con aquéllas en las que aparece su variable asignada.

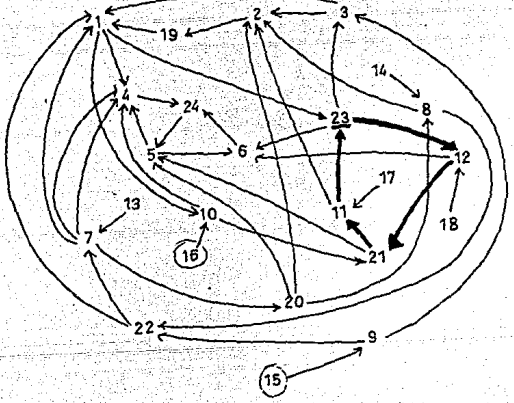


Figura 4

Algunas veces, al recorrer el grafo se llega a un nodo por segunda vez, y se forma un ciclo. Esto indica la existencia de un conjunto de ecuaciones que debe resolverse simultáneamente. Los nodos que forman parte el ciclo se unen para formar otro nuevo. En la figura 4 se presenta el grafo original y la detección del primer ciclo del recorrido formado por las funciones F23, F12, F21 y F11. En esta figura se muestran también nodos como F15 ó F16, que por no pertenecer a ningún ciclo en el grafo, forman cada uno un nodo independiente y además, por no tener entradas, pueden calcularse antes que los demás. Este tipo de búsqueda 'con prioridad a profundidad' es descrito en [Tarjan 1972].

Finalmente, se propone una partición y un orden de precedencia, con el que se construye el simulador. Esta partición está formada por los siguientes 10 bloques:

(F15), (F14), (F13), (F9), (F16), (F8, F20, F7, F22),
(F17), (F18), (F3, F10, F1, F19, F2, F11, F21, F12, F23)
y (F6, F5, F24, F4)

Cada bloque es un subsistema de ecuaciones que se resuelve en el simulador numéricamente.

Finalmente, se muestra el código generado para este modelo.

```
type abs2.pas
PROGRAM SIMULADOR;                    (*  ABS2.PAS  *)


(*      Este  programa  fue      CODIFICADO  EN  FORMA  AUTOMATICA          *)
(*      por  el     SISTEMA    MP       escrito     en      COMMON LISP      *)
(*                                                                          *)
(*      El  SISTEMA  MP    fue  escrito  por    CARLOS ROJAS GUZMAN         *)
(*                                                                          *)
(*      con el apoyo del Dr.Ignacio Avila  y del   Dr.  Rene Banares        *)
(*      Directores del grupo de   Diseno  de  Procesos  Asistido            *)
(*      por Computadora     de la   FACULTAD  de   QUIMICA,     de la        *)
(*              Universidad Nacional Autonoma de Mexico                     *)
(*                          M E X I C O                                     *)



CONST
    nmax = 9 ;          (*  DIMENSIONES DEL MAYOR SISTEMA DE ECUACIONES  *)
    nblo = 10 ;         (*  NUMERO DE BLOQUES   *)
    inc  = 0.001 ;      (* INCREMENTO PARA APROXIMAR DERIVADAS *)

TYPE
    ncuadrada = array [1..nmax,1..nmax] of real;
    matlineal = array [1..nmax] of real;
    matindices = array [1..nmax] of integer;

VAR
    : matlineal;
    bloque, orden : integer;
                                        (*CONJUNTO DE VARIABLES DEL MODELO *)
    U0  : real ;
    U0  : real ;
    U2  : real ;
    U3  : real ;
    U3  : real ;
    U11 : real ;
    X21 : real ;
    X31 : real ;
    U12 : real ;
    X32 : real ;
    X33 : real ;
    P   : real ;
    T   : real ;
    X30 : real ;
    X20 : real ;
    X10 : real ;
    X31 : real ;
    X21 : real ;
    X11 : real ;
    X32 : real ;
    X22 : real ;
    X12 : real ;
    Y31 : real ;
    Y21 : real ;
    Y11 : real ;
    Y32 : real ;
    Y22 : real ;
    Y12 : real ;
    Y33 : real ;
    Y23 : real ;
    Y13 : real ;


(*SUBSISTEMA DE ECUACIONES NUMERO 1  *)
PROCEDURE  SIST1  (var: matlineal; var alfa: ncuadrada; var beta: matlineal);
begin
    X31 := a[1];        (* FUNCION  F15  *)
    beta[1] :=   (X31)*0.177-49.5E-4*T+4.15E-6*T*T-2.22E-8*(T*T*T) ;
end;
```

```
{*SUBSISTEMA DE ECUACIONES NUMERO 2  *}
PROCEDURE  SIST2  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    K21 := xd[1];
                        {* FUNCION  F14  *}
    beta[1] := - (K21-0.94+46.6E-4*T-49.4E-6*T*T+3.033E-8*T*T*T) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 3  *}
PROCEDURE  SIST3  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    K11 := xd[1];
                        {* FUNCION  F13  *}
    beta[1] := - (K11-0.093+15.39E-4*T-10.37E-6*T*T+0.159E-8*T*T*T) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 4  *}
PROCEDURE  SIST4  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    Y31 := xd[1];
                        {* FUNCION  F9  *}
    beta[1] := - (Y31-K31*X31) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 5  *}
PROCEDURE  SIST5  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    K12 := xd[1];
                        {* FUNCION  F16  *}
    beta[1] := - (K12-0.093+15.39E-4*T-10.37E-6*T*T+0.159E-8*T*T*T) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 6  *}
PROCEDURE  SIST6  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    Y11 := xd[1];
    X11 := xd[2];
    X21 := xd[3];
    Y21 := xd[4];
                        {* FUNCION  F22  *}
    beta[1] := - (Y11+Y21+Y31-1) ;
                        {* FUNCION  F7  *}
    beta[2] := - (Y11-K11*X11) ;
                        {* FUNCION  F20  *}
    beta[3] := - (X11+X21+X31-1) ;
                        {* FUNCION  F8  *}
    beta[4] := - (Y21-K21*X21) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 7  *}
PROCEDURE  SIST7  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    K22 := xd[1];
                        {* FUNCION  F17  *}
    beta[1] := - (K22-0.94+46.6E-4*T-49.4E-6*T*T+3.033E-8*T*T*T) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 8  *}
PROCEDURE  SIST8  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    K32 := xd[1];
                        {* FUNCION  F18  *}
    beta[1] := - (K32+0.127-49.5E-4*T+4.15E-6*T*T-2.22E-8*T*T*T) ;
end;
```

```
{*SUBSISTEMA DE ECUACIONES NUMERO 9  *}
PROCEDURE  SIST9  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);

begin
     Y32 := xd[1];
     X32 := xd[2];
     X22 := xd[3];
     Y22 := xd[4];
     X20 := xd[5];
     X10 := xd[6];
     Y12 := xd[7];
     X12 := xd[8];
     L0 := xd[9];
                         {* FUNCION  F23  *}
     beta[1] := -  (Y12*Y22+Y32-1) ;
                         {* FUNCION  F12  *}
     beta[2] := -  (Y32-L*32*X32) ;
                         {* FUNCION  F21  *}
     beta[3] := -  (X12*X22+X32-1) ;
                         {* FUNCION  F11  *}
     beta[4] := -  (Y22-L*22*X22) ;
                         {* FUNCION  F2  *}
     beta[5] := -  (X20*L0+Y22*V2-X21*L1-Y21*V1) ;
                         {* FUNCION  F19  *}
     beta[6] := -  (X10+X20+X30-1) ;
                         {* FUNCION  F1  *}
     beta[7] := -  (X10*L0+Y12*V2-X11*L1-Y11*V1) ;
                         {* FUNCION  F10  *}
     beta[8] := -  (Y12-K12*X12) ;
                         {* FUNCION  F3  *}
     beta[9] := -  (X30*L0+Y32*V2-X31*L1-Y31*V1) ;
end;



{*SUBSISTEMA DE ECUACIONES NUMERO 10  *}
PROCEDURE  SIST10  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
     Y13 := xd[1];
     Y23 := xd[2];
     V3 := xd[3];
     Y33 := xd[4];
                         {* FUNCION  F4  *}
     beta[1] := -  (X11*L1+Y13*V3-X12*L2-Y12*V2) ;
                         {* FUNCION  F24  *}
     beta[2] := -  (Y13+Y23+Y33-1) ;
                         {* FUNCION  F5  *}
     beta[3] := -  (X21*L1+Y23*V3-X22*L2-Y22*V2) ;
                         {* FUNCION  F6  *}
     beta[4] := -  (X31*L1+Y33*V3-X32*L2-Y32*V2) ;
end;



PROCEDURE EVALUA (bloque: integer; xd: matlineal; var alfa: mcuadrada;
                  var beta: matlineal);
begin
     case bloque of
     1 : SIST1 (xd,alfa,beta) ;
     2 : SIST2 (xd,alfa,beta) ;
     3 : SIST3 (xd,alfa,beta) ;
     4 : SIST4 (xd,alfa,beta) ;
     5 : SIST5 (xd,alfa,beta) ;
     6 : SIST6 (xd,alfa,beta) ;
     7 : SIST7 (xd,alfa,beta) ;
     8 : SIST8 (xd,alfa,beta) ;
     9 : SIST9 (xd,alfa,beta) ;
     10 : SIST10 (xd,alfa,beta)

     end;

end;
```

```
B:type abs2.pas



PROCEDURE DERIVVFUN  (bloque :integer; x :matlineal; n :integer;
                      var alfa :mcuadrada; var beta :matlineal);

var xd :matlineal; a, b, c :integer; betabase :matlineal;
begin
    evalua (bloque, x, alfa, beta);
    for c := 1 to n do betabase[c] := beta[c];    {X F  valuado en  Xo X}
    for a := 1 to n do                            {F para cada variable X}
    begin
        xd := x ; xd[a] := x[a] +inc;
        evalua (bloque, xd, alfa, beta);
        for b := 1 to n do                        {F para cada FUNCION  X}
            begin
            alfa[b,a] := (betabase[b] - beta[b]) / inc
            end;
    end;
end;




PROCEDURE LUTRIANG (bloque :integer; var a :mcuadrada; n :integer;
                    var indice :matindices; var d :real);

Const
    epsilon = 1.0e-20;

Var
    k, i, imax, j :integer; suma, val, cota :real; vv :matlineal;

begin
    d := 1.0;
    for i := 1 to n do begin
        cota := 0.0;
        for j := 1 to n do if ( abs(a[i,j]) > cota) then cota := abs(a[i,j]);
        if (cota = 0.0) then begin
            writeln ('aviso desde LUTRIANG - la matriz del sistema ',
            bloque, ' es singular'); readln; readln;
        end;
        vv[i] := 1.0 / cota  end;
    for j := 1 to n do begin
        for i := 1 to j-1 do begin
            suma := a[i,j];
            for k := 1 to i-1 do begin
                suma := suma - a[i,k]*a[k,j]  end;
            a[i,j] := suma end;
        cota := 0.0;
        for i := j to n do begin
            suma := a[i,j];
            for k := 1 to j-1 do begin
                suma := suma - a[i,k]*a[k,j]  end;
            a[i,j] := suma;
            val := vv[i]*abs(suma);
            if ( val >= cota ) then begin
                cota := val; imax := i end end;
        if (j <> imax) then begin
            for k := 1 to n do begin
                val := a[imax,k]; a[imax,k] := a[j,k]; a[j,k] := val  end;
            d := -d; vv[imax] := vv[j] end;
        indice[j] := imax;
        if (a[j,j] = 0.0) then a[j,j] := epsilon;
        if ( j <> n ) then begin
            val := 1.0 / a[j,j];
            for i := j+1 to n do begin
                a[i,j] := a[i,j]*val end end end;

end;
```

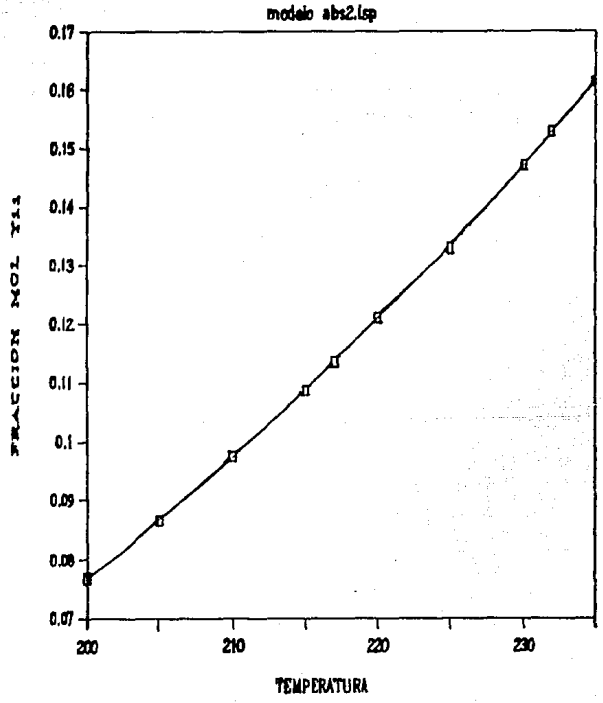
```
PROCEDURE LURESOL (a :mcuadrada; n :integer; indice :matindices;
                   var b: matlineal);

Var
    i, ip, ii, j :integer; suma :real;
begin
    ii := 0;
    for i := 1 to n do begin
        ip := indice[i]; suma := b[ip]; b[ip] := b[i];
        if ( ii<>0 ) then begin
            for j := ii to i-1 do begin
                suma := suma-a[i,j]*b[j] end end
        else if ( suma <> 0 ) then begin
            ii := i end;
        b[i] := suma end;
    for i := n downto 1 do begin
        suma := b[i];
        if ( i<n ) then begin
            for j := i+1 to n do begin
                suma := suma - a[i,j]*b[j] end end;
        b[i] := suma / a[i,i] end
end;


PROCEDURE RESUELVE (bloque, iteramax :integer; var x :matlineal; n :integer;
                    tolx, tolf :real);
label 99;
Var
    k, i :integer; errx, errf, d :real; beta :matlineal; alfa :mcuadrada;
    indice :matindices;

begin
    for k := 1 to iteramax do begin
    derivfun (bloque, x, n, alfa, beta); errf := 0.0;
    for i := 1 to n do errf := errf + abs (beta[i]);
    if (errf <= tolf) then goto 99;
    lutriang(bloque,alfa,n,indice,d);
    luresol(alfa,n,indice,beta);
    errx := 0.0;
    for i := 1 to n do begin
            errx := errx + abs(beta[i]); x[i] := x[i] + beta[i];
            end;
        if (errx <= tolx) then goto 99
    end;
99: end;


PROCEDURE PRESENTACION; (* presentacion inicial en pantalla *)
Var q :integer;
begin
    for q := 1 to 25 do writeln;
    writeln ('Este programa fue CODIFICADO AUTOMATICAMENTE ... etc.');
    for q := 1 to 10 do writeln;
    writeln('                    SIMULADOR DE PROCESOS  ABSORBEDOR' );
    for q:= 1 to 10 do writeln;
end;

PROCEDURE VCONOCIDAS;(* pide al usuario el valor de variables conocidas *)
begin
    writeln ('Necesito el valor de las siguientes variables que conoces');
    write(' F = '); read ( F );
    write(' P = '); read ( P );
end;


PROCEDURE VDECISION; (* pide el valor de las variables de decision *)
begin
    writeln ('Necesito ahora el valor de las variables de decision');
    write (' X31 = '); read ( X31 );
    write (' V1 = '); read ( V1 );
    write (' V2 = '); read ( V2 );
    write (' L1 = '); read ( L1 );
    write (' X30 = '); read ( X30 );
    write (' L2 = '); read ( L2 );
end;
```

```
PROCEDURE ESTIMACION; (X El usuario da estimacion inicial a incognitas X)
begin
    write (' L0  = '); read( L0 );
    write (' V3  = '); read( V3 );
    write (' K11 = '); read( K11 );
    write (' K21 = '); read( K21 );
    write (' K31 = '); read( K31 );
    write (' K12 = '); read( K12 );
    write (' K22 = '); read( K22 );
    write (' K32 = '); read( K32 );
    write (' X20 = '); read( X20 );
    write (' X10 = '); read( X10 );
    write (' X21 = '); read( X21 );
    write (' X11 = '); read( X11 );
    write (' X32 = '); read( X32 );
    write (' X22 = '); read( X22 );
    write (' X12 = '); read( X12 );
    write (' Y31 = '); read( Y31 );
    write (' Y21 = '); read( Y21 );
    write (' Y11 = '); read( Y11 );
    write (' Y32 = '); read( Y32 );
    write (' Y22 = '); read( Y22 );
    write (' Y12 = '); read( Y12 );
    write (' Y33 = '); read( Y33 );
    write (' Y23 = '); read( Y23 );
    write (' Y13 = '); read( Y13 );
end;

PROCEDURE AUTOGENERADOS; (X Se da un valor inicial para cada variable X)
begin
    L0 := 0.5;
    V3 := 0.5;
    K11 := 0.5;
    K21 := 0.5;
    K31 := 0.5;
    K12 := 0.5;
    K22 := 0.5;
    K32 := 0.5;
    X20 := 0.5;
    X10 := 0.5;
    X21 := 0.5;
    X11 := 0.5;
    X32 := 0.5;
    X22 := 0.5;
    X12 := 0.5;
    Y31 := 0.5;
    Y21 := 0.5;
    Y11 := 0.5;
    Y32 := 0.5;
    Y22 := 0.5;
    Y12 := 0.5;
    Y33 := 0.5;
    Y23 := 0.5;
    Y13 := 0.5;
end;


PROCEDURE INICIA;  (X pide al usuario valor inicial de incognitas, OPCIONAL X)
Var opinicia :integer;
begin
    writeln('Quieres dar una primera estimacion para las incognitas(1/0)');
    read (opinicia);
    if opinicia = 1 then ESTIMACION else AUTOGENERADOS;
end;
```

```
PROCEDURE RESULTADOS;  (* muestra el conjunto final de variables y valores K)
begin
    writeln('Se ha encontrado la siguiente solucion al sistema ');
    writeln('Los valores de las variables conocidas son');
    writeln (' T = ', T);
    writeln (' P = ', P);
    writeln ('Los valores dados a las variables de decision son');
    writeln (' X31 = ', X31);
    writeln (' V1 = ', V1);
    writeln (' V2 = ', V2);
    writeln (' L1 = ', L1);
    writeln (' X30 = ', X30);
    writeln (' L2 = ', L2);
    writeln ('Y los valores encontrados para las incognitas');
    writeln (' L0 = ', L0);
    writeln (' V3 = ', V3);
    writeln (' K11 = ', K11);
    writeln (' K21 = ', K21);
    writeln (' K31 = ', K31);
    writeln (' K12 = ', K12);
    writeln (' K22 = ', K22);
    writeln (' K32 = ', K32);
    writeln (' X20 = ', X20);
    writeln (' X10 = ', X10);
    writeln (' X21 = ', X21);
    writeln (' X11 = ', X11);
    writeln (' X32 = ', X32);
    writeln (' X22 = ', X22);
    writeln (' X12 = ', X12);
    writeln (' Y31 = ', Y31);
    writeln (' Y21 = ', Y21);
    writeln (' Y11 = ', Y11);
    writeln (' Y32 = ', Y32);
    writeln (' Y22 = ', Y22);
    writeln (' Y12 = ', Y12);
    writeln (' Y33 = ', Y33);
    writeln (' Y23 = ', Y23);
    writeln (' Y13 = ', Y13);
    readln;
end;

(*.....................PROGRAMA PRINCIPAL.....................*)
begin
    PRESENTACION;
    VCONOCIDAS;
    VDECISION;
    INICIA;

    x[1] := K3;
    RESUELVE (1,50,x,1.1e-9,1e-9);
    K31 := x[1];      writeln (' K31 = ', x[1] );

    x[1] := K2;
    RESUELVE (2,50,x,1.1e-9,1e-9);
    K21 := x[1];      writeln (' K21 = ', x[1] );

    x[1] := K1;
    RESUELVE (3,50,x,1.1e-9,1e-9);
    K11 := x[1];      writeln (' K11 = ', x[1] );

    x[1] := Y31;
    RESUELVE (4,50,x,1.1e-9,1e-9);
    Y31 := x[1];      writeln (' Y31 = ', x[1] );

    x[1] := K12;
    RESUELVE (5,50,x,1.1e-9,1e-9);
    K12 := x[1];      writeln (' K12 = ', x[1] );
```

```
    x[1] := Y11;
    x[2] := X11;
    x[3] := X21;
    x[4] := Y21;
    RESOLVE (6,50,x,0.1e-9,1e-9);
    Y11 := x[1];        writeln (' Y11 = ', x[1] );
    X11 := x[2];        writeln (' X11 = ', x[2] );
    X21 := x[3];        writeln (' X21 = ', x[3] );
    Y21 := x[4];        writeln (' Y21 = ', x[4] );


    x[1] := K22;
    RESOLVE (7,50,x,1.1e-9,1e-9);
    K22 := x[1];        writeln (' K22 = ', x[1] );


    x[1] := K32;
    RESOLVE (8,50,x,1.1e-9,1e-9);
    K32 := x[1];        writeln (' K32 = ', x[1] );


    x[1] := Y32;
    x[2] := X32;
    x[3] := X22;
    x[4] := Y22;
    x[5] := X20;
    x[6] := X10;
    x[7] := Y12;
    x[8] := X12;
    x[9] := L0;
    RESOLVE (9,50,x,9.1e-9,1e-9);
    Y32 := x[1];        writeln (' Y32 = ', x[1] );
    X32 := x[2];        writeln (' X32 = ', x[2] );
    X22 := x[3];        writeln (' X22 = ', x[3] );
    Y22 := x[4];        writeln (' Y22 = ', x[4] );
    X20 := x[5];        writeln (' X20 = ', x[5] );
    X10 := x[6];        writeln (' X10 = ', x[6] );
    Y12 := x[7];        writeln (' Y12 = ', x[7] );
    X12 := x[8];        writeln (' X12 = ', x[8] );
    L0 := x[9];         writeln (' L0 = ', x[9] );


    x[1] := Y13;
    x[2] := Y23;
    x[3] := Y31;
    x[4] := Y33;
    RESOLVE (10,50,x,9.1e-9,1e-9);
    Y13 := x[1];        writeln (' Y13 = ', x[1] );
    Y23 := x[2];        writeln (' Y23 = ', x[2] );
    Y31 := x[3];        writeln (' Y31 = ', x[3] );
    Y33 := x[4];        writeln (' Y33 = ', x[4] );

readln;

RESULTADOS;

end.
```

SIMULADOR "ABSORBEDOR 1"



5 Conclusiones

Se ha cumplido con el objetivo planteado inicialmente al construir un sistema capaz de analizar un modelo matemàtico, desarrollar un procedimiento de cálculo y construir automàticamente un simulador especifico para el modelo y el problema dados. También se ha mostrado la utilidad de un sistema como el desarrollado.

Las aplicaciones del sistema MP son muy amplias, ya que permite construir simuladores para cualquier sistema que pueda describirse formalmente por medio de un sistema de ecuaciones algebraicas, lineales o no lineales, consistente y completo.

Dado que los simuladores de proceso constituyen una herramienta valiosa para quien diseña procesos quimicos, la investigación dirigida hacia la automatización de la construcción de simuladores en general, y los resultados de este proyecto en particular, representan una contribución importante tanto en el plano práctico como en el teórico por varias razones.

La automatización descrita permite a quien diseña un proceso concentrar su atención y dedicar su tiempo a la solución del problema de diseño en forma global sin preocuparse tanto por la

implementación de los programas computacionales y del desarrollo de los procedimientos de cálculo.

Además, como el tiempo que toma generar en forma automática un simulador es extraordinariamente corto al compararse con el tiempo que necesitaría un experto, resulta claro que el diseñador puede aprovechar más su capacidad creativa al construir una gran cantidad de simuladores, uno para cada proceso o para cada conjunto de incógnitas y especificaciones. Tiene entonces la posibilidad de comparar un número mucho mayor de opciones lo que le facilita encontrar aquélla que, de acuerdo con su criterio, apoyado en resultados numéricos obtenidos con los simuladores, sea la mejor.

Se ha modificado el algoritmo de asignación de variables de salida propuesto por [Westerberg 1979] para que la probabilidad de éxito de la búsqueda heurística aumente y así se recurra con menos frecuencia a la búsqueda sistemática. La ventaja que para este sistema representa y las razones por la que esto sucede se describen en la sección 3.,2.4, sin embargo, para cuantificar con precisión esta diferencia de probabilidades es necesario implementar también el algoritmo original y hacer un estudio estadístico con suficientes ejemplos.

La investigación realizada puede servir como base para el desarrollo de un sistema experto para la solución de sistemas de ecuaciones no lineales. Con objeto de aumentar las aplicaciones del sistema MP y hacerlo más general se sugiere extenderlo para

que pueda trabajar con ecuaciones diferenciales. Es también posible mejorar los algoritmos usados a lo largo del desarrollo del procedimiento de cálculo (variables de decisión y asignación de variables de salida) para poder elegir entre opciones que hasta ahora parecen equivalentes. Para facilitar estas mejoras, los criterios de selección se han implementado en forma modular.

Otras mejoras incluyen la selección automática del método numérico más adecuado para cada problema y la implementación de un algoritmo para buscar variables de rompimiento, para determinar si el esfuerzo necesario para encontrar estas variables e iterar sobre ellas es menor que el necesario para resolver simultáneamente los subsistemas propuestos.

Finalmente, es importante señalar que aunque esta investigación tuvo como objetivo resolver un problema típico de la Ingeniería Química, el sistema computacional desarrollado tiene un alcance mucho más amplio. Pueden construirse programas computacionales para reproducir el comportamiento de cualquier fenómeno que pueda representarse con un modelo como el descrito y ésto lo hace útil en muchos otros campos del conocimiento.

NOTA: El código computacional desarrollado, escrito en COMMON LISP (50 K) se encuentra a disposición de los interesados en el Departamento de Ingeniería Química de la Facultad de Química de la Universidad Nacional Autónoma de México.

Apèndice I

Un Algoritmo de Búsqueda

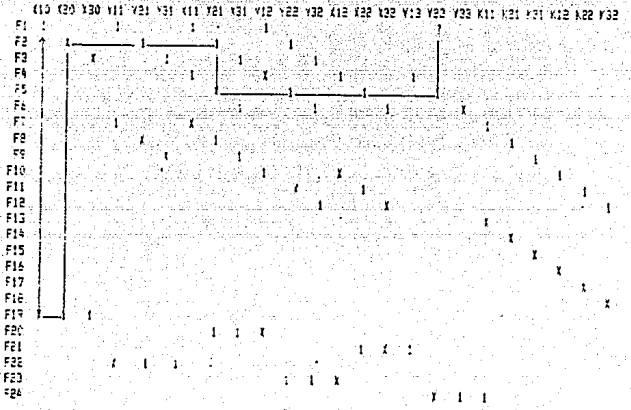
La asignación de variables de salida es un problema de búsqueda con un número de combinaciones demasiado grande como para llevarla a cabo en forma sistemática, es decir, es un problema de explosión combinatórica. Por eso la búsqueda es heurística, sin embargo, como es común en este tipo de búsquedas, existen casos en los que así no se encuentra una solución y es necesario recurrir a una búsqueda sistemática sobre el espacio de búsqueda ya reducido.

A continuación se ilustra la búsqueda sistemática descrita en la sección 3.2.5 usada para encontrar un Rearreglo de la Asignación de variables de salida.

En la figura I se muestra el resultado de la búsqueda heurística y puede verse que la asignación no es completa, (un par asignado se indica con un círculo en la intersección del renglón la columna correspondientes a la función y variable que lo forman). Quedan una función, F1 y una variable, Y23. Como Y23 no aparece en F1, entonces no puede ser asignada. Se lleva a cabo una búsqueda sistemática con prioridad a profundidad, (Tarjan

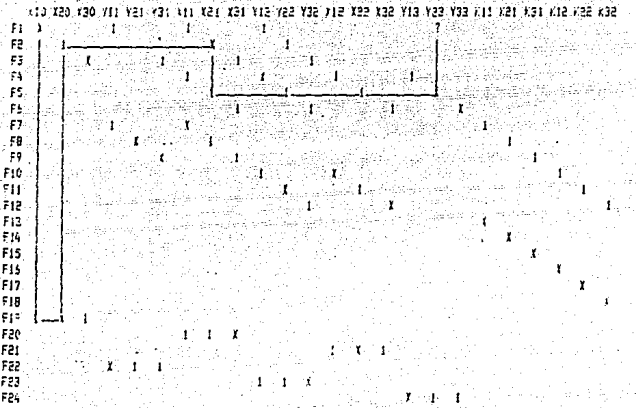
1972] y [Charniak 1985] (para encontrar un "Camino de Steward"). También en la figura 6 puede verse la trayectoria seguida hasta encontrar dicho camino, desde Y23 hasta F1. Una vez encontrado, se hace un rearreglo de los pares asignados; los pares no asignados en la ruta quedan asignados y viceversa. La nueva asignación, ahora completa, se muestra en la figura 7.

Finalmente, se muestra la parte del código en LISP que realiza la búsqueda descrita.



BUSQUEDA PARA COMPLETAR LA ASIGNACION

Figura 6



REARREGLO DE LA ASIGNACION

Figura 7

FIGURA 8

```
(DEFUN FIN-ASIG (MODELO) ;| Muestra y guarda una copia de la ASIGNACION
      (SETF (GET MODELO 'LIST-ASIG) LIST-ASIG)
      (ESCRIBE'(ESTA ES LA ASIGNACION DE VARIABLES DE SALIDA))
      (ESCRIBE (GET MODELO 'LIST-ASIG)))


(DEFUN PSOL (FUNCION)  ;|Prueba para saber si una FUNCION es SOLUCION
  (MEMBER FUNCION (FGET MODELO 'PARTICION 'FUNFINALES)))


(DEFUN PFIN (SIMBOLO) ;|Prueba para saber si SIMBOLO es FUNCION (y no VARIABLE)
  (MEMBER SIMBOLO (GET MODELO 'FUNFINALES)))


(DEFUN PVAR (SIMBOLO);|Prueba para saber si SIMBOLO es VARIABLE (y no FUNCION)
  (MEMBER SIMBOLO (GET MODELO 'INCFINALES)))


;|Dada la GRAFICA que relaciona funciones por las variables que contienen,
encuentra el conjunto de SALIDAS (funciones) de una variable, NOTESE que la
;|FUNCION asignada no es parte del conjunto. Finalmente construye el nodo.
(DEFUN SALIDAS-VAR (VARIABLE)
  (LET ((SALIDAS NIL)(F-A (FUN-ASIGNADA VARIABLE)))
    (MAPCAR '(LAMBDA (X)
               (COND ((AND (MEMBER VARIABLE (FGET MODELO X 'VARIABLES))
                           (NOT (EQUAL  F-A  X )))
                      (SETQ SALIDAS (CONS X SALIDAS)))
                     ( T NIL )))
            (GET MODELO 'FUNFINALES))
     (CONS (LIST VARIABLE) SALIDAS)))


;|Construye el NODO cuya UNICA salida es la VARIABLE ASIGNADA
(DEFUN SALIDA-FUN (FUNCION)
  (LET ((NODOF NIL))
    (MAPCAR '(LAMBDA (X)
               (COND ((MEMBER FUNCION X)
                      (SETQ NODOF (LIST (CDR X) (CAR X))))
                     ( T NIL)))
            LIST-ASIG)
     NODOF))
```

```
(Encuentra CUAL es la FUNCION asignada a una variable
(DEFUN FUN-ASIGNADA (VARIABLE)
  (LET ((FUN-ASIG NIL))
    (MAPCAR '(LAMBDA (X)
               (COND ((MEMBER VARIABLE X) (SETQ FUN-ASIG (CADR X)))
                     ( T NIL )))
            HIST-ASIG)
    FUN-ASIG))



(DEFUN PVISITADA (FUNCION) ;Determina si la RUTA ya PASO por esa FUNCION
  (LET ((L NIL))
    (MAPCAR '(LAMBDA (X)
               (COND ((EQUAL FUNCION (CAAR X))(SETQ L (CONS X L)))
                     ( T NIL )))
            (FGET MODELO 'RUTA 'VALUE))
    L))     ;El resultado es NIL si no ha pasado por AHI, si no, NON-NIL



(DEFUN CAMINO-INEXISTENTE ()
 (FORMAT T "~%")
 (FORMAT T "~%Para encontrar una PARTICION es necesario ASIGNAR una VARIABLE")
 (FORMAT T "~%a cada FUNCION.  Para encontrar una RELACION  (no es UNICA) se")
 (FORMAT T "~%use un algoritmo basado en  REGLAS HEURISTICAS (pues el NUMERO")
(FORMAT T "~%de  COMBINACIONES  es demasiado  grande para probarlas todas).")
 (FORMAT T "~%Sin embargo, este  tipo de METODOS  no garantiza encontrar una")
 (FORMAT T "~%SOLUCION al problema. Cuando falla, se recurre a una  BUSQUEDA")
 (FORMAT T "~%SISTEMATICA de un  espacio reducido, se trata de  encontrar un")
 (FORMAT T "~%Camino de Steward.")
 (FORMAT T "~%~%La BUSQUEDA SISTEMATICA, para este problema, se ha realizado")
 (FORMAT T "~%y  como  el camino  buscado  no se ha encontrado,  se  puede ")
 (FORMAT T "~%CONCLUIR que este camino  NO EXISTE ")
 (FORMAT T"~%~%Se recomienda intentar con otras VARIABLES DE DECISION")
 (READ)(READ) )



(DEFUN CLAUSURA-SALIDA (SALIDA NODO)
  (LET ((NUEVO-NODO NIL))
    (cond ((equal salida 'origen) (camino-inexistente))
          ( t nil ))
    (REMPROP MODELO 'RUTA 'VALUE NODO)  ;lo saca del final de la lista
    (MAPCAR '(LAMBDA (X)  ;elimina la salida del nodo original
               (COND ((EQUAL X SALIDA) NIL)
                     ( T (SETQ NUEVO-NODO (CONS X NUEVO-NODO)))))
            (CDR NODO))
    (SETQ NUEVO-NODO (CONS (CAR NODO) NUEVO-NODO));define al nuevo nodo
    (PUTP MODELO 'RUTA 'VALUE NUEVO-NODO)) ;lo coloca al final de la lista
```

```
;Cuando llega a un nodo SIN SALIDAS VIABLES
(DEFUN CAMINO-CERRADO (NODOC)     ;NODOC es un nodo del tipo ( (X1) __  .  )
  (LET ((OPCION-NO-VIABLE NIL))
    (COND ( (EQUAL NODOC '(nil) )  (CAMINO-INEXISTENTE) )
          ( T
    (FREMOVE MODELO 'RUTA 'VALUE NODOC)
    (SETQ OPCION-NO-VIABLE (CAAAR(LAST (FGET MODELO 'RUTA 'VALUE))))
    (FREMOVE MODELO 'RUTA 'VALUE (CAR(LAST(FGET MODELO 'RUTA 'VALUE))))
    (CLAUSURA-SALIDA OPCION-NO-VIABLE (CAR(LAST(FGET MODELO 'RUTA 'VALUE))))
))))


(DEFUN SACA-ASIG (NODO)
  (MAPCAR '(LAMBDA (X)
             (COND ( (EQUAL X NODO) (FREMOVE MODELO 'ASIGNACION 'VALUE X))
                   ( T NIL )))
          (FGET MODELO 'ASIGNACION 'VALUE )))


;Una vez encontrada una TRAYECTORIA EXITOSA, establece una NUEVA ASIGNACION
(DEFUN CAMBIA-ASIG (NODO-META)
  (LET ((SOMBRA (CADR NODO-META)))
    (SETF (CET MODELO 'RUTA) (FGET MODELO 'RUTA 'VALUE))
    (FREMOVE MODELO 'PARTICION 'PUNTFINALES (CADR NODO-META))
    (FREMOVE MODELO 'PARTICION 'INCFINALES PRIMERAV)
    (DO ((PENULTIMO NIL))
        ((EQUAL PENULTIMO 'ORIGEN)
           (SETQ LIST-ASIG (FGET MODELO 'ASIGNACION 'VALUE)))
      (SETQ ULTIMO (CAAAR(LAST(FGET MODELO 'RUTA 'VALUE))))
      (FREMOVE MODELO 'RUTA 'VALUE (CAR(LAST (FGET MODELO 'RUTA 'VALUE))))
      (SETQ PENULTIMO (CAAAR (LAST (FGET MODELO 'RUTA 'VALUE))))
      (FPUT MODELO 'ASIGNACION 'VALUE (LIST ULTIMO SOMBRA))
      (SACA-ASIG (LIST ULTIMO PENULTIMO))
      (FREMOVE MODELO 'RUTA 'VALUE (CAR(LAST (FGET MODELO 'RUTA 'VALUE))))
      (SETQ SOMBRA PENULTIMO))))


(DEFUN REARREGLA (MODELO) ;Algoritmo para construir un "CAMINO de STEWARD"
 (COLOCAF LIST-ASIG  MODELO 'ASIGNACION);Guarda el avance alcanzado
 (FPUT MODELO 'RUTA 'VALUE(CONS '(ORIGEN) (FGET MODELO 'PARTICION 'INCFINALES)))
   (DO ((U-NODO (CAR(FGET MODELO 'RUTA 'VALUE));El ultimo NODO de la ruta
                (CAR(LAST(FGET MODELO'RUTA'VALUE))))
       (PRIMERAV (CADAR (FGET MODELO 'RUTA 'VALUE)));la 1a var del 1er nodo
       (PUNTA (CADR U-NODO) (CADR U-NODO)));Es la primera salida del U-NODO
     ((AND PUNTA (PFUN PUNTA) (PSOL PUNTA));Si EXISTE, es FUNCION, y SOLUCION
        (CAMBIA-ASIG U-NODO)                ;entonces ¡¡ENCONTRO un camino
        (SETF (GET MODELO 'LIST-ASIG ORIG) LIST-ASIG)
        (FIN-ASIG MODELO))
     (COND ((EQUAL PUNTA NIL) (CAMINO-CERRADO U-NODO))
           ((AND (PFUN PUNTA)(VISITADA PUNTA)(CLAUSURA SALIDA PUNTA U NODO))
           ((AND (PFUN PUNTA)(NOT (VISITADA PUNTA)))
            (FPUT MODELO 'RUTA 'VALUE (SALIDA-FUN PUNTA)))
           ((VAR PUNTA) (FPUT MODELO 'RUTA 'VALUE (SALIDAS-VAR PUNTA)))))))
```

Apéndice II

Efecto de la Selección de Variables de Decisión sobre el Procedimiento de Cálculo Generado

Se muestra la partición encontrada para distintas selecciones de variables de decisión. Conviene encontrar el mayor número de subsistemas, cada uno con el menor tamaño posible para simplificar los cálculos numéricos.

Usando el modelo del absorbedor, con el que se ilustra la construcción de un simulador, se ilustra el efecto de la selección de variables de decisión mostrando la partición encontrada en cada caso.

En el primer caso se seleccionaron T, V3, L2, X20, X30, K22, y Y11, y el sistema original se descompuso en 7 sistemas, el mayor de 18 ecuaciones. Una representación gráfica que facilita ver la reducción se muestra en la figura 9a.

Cuando se seleccionaron T, V1, L0, V2, L2, V3, L1, K11 (siguiendo la sugerencia del sistema generador de simuladores), la partición quedó constituida por 10 bloques, el mayor de 9 ecuaciones, se muestra en la figura 9b.

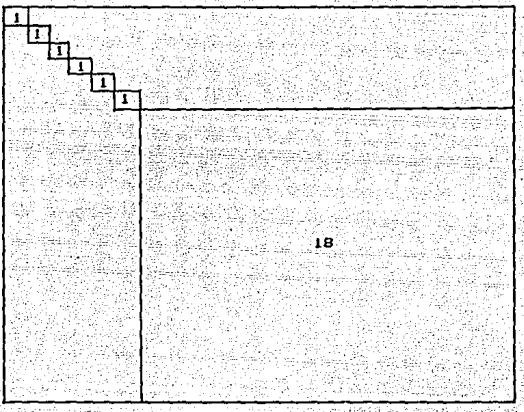


FIGURA 9a

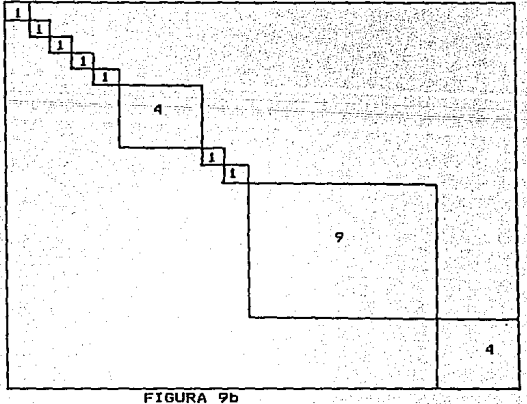


FIGURA 9b

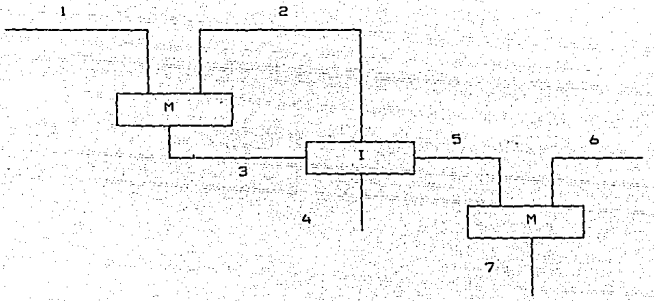
APENDICE III

Se muestra con un ejemplo, lo que el sistema computacional desarrollado en esta tesis realiza.

El modelo que en este caso particular se usa es el de un proceso que consiste en dos mezcladores y un intercambiador de calor, 4 componentes, y 6 corrientes, una de las cuales recircula. Se tomó como base el modelo que se propone en [Ramirez 1972].

Después del diagrama de proceso, se muestra la descripción formal del modelo (el conjunto de ecuaciones que reproduce el comportamiento del proceso) y finalmente se presenta el simulador generado en forma automática.

Como puede verse, en este caso el sistema de ecuaciones inicial se reduce a 25 subsistemas de 1 ecuación cada uno.



ESQUEMA DE PROCESO DE MEZCLADO Y CALENTAMIENTO

Figura 10

```
;DESCRIPCION FORMAL DEL MODELO MEZCLADOR-INTERCAMBIADOR-MEZCLADOR (MIM)
                                        ;Modelo basado en uno presentado
                                        ;por [Ramirez 1972].

(DEFUN MIM ()
  (SETF (GET 'MIM 'FRAME)
    '(MIM
        (E1 (VARIABLES X11 X21)
            (FORMULA (X11=X21-1)))
        (E2 (VARIABLES X12 X22)
            (FORMULA (X12=X22-1)))
        (E3 (VARIABLES X13 X23)
            (FORMULA (X13=X23-1)))
        (E4 (VARIABLES X14 X24)
            (FORMULA (X14=X24-1)))
        (E5 (VARIABLES X15 X25)
            (FORMULA (X15=X25-1)))
        (E6 (VARIABLES X16 X26 X36 X46)
            (FORMULA (X16)X26=X36=X46-1)))
        (E7 (VARIABLES X17 X27 X37 X47)
            (FORMULA (X17=X27=X37=X47-1)))
        (E8 (VARIABLES F1 F2 F3)
            (FORMULA (F1=F2 F3)))
        (E9 (VARIABLES F1 F2 F3 X11 X12 X13)
            (FORMULA (F1*X11+F2*X12-F3*X13)))
        (E10 (VARIABLES F1 T1 F2 T2 F3 T3)
             (FORMULA (0.81*F1*T1 + 0.82*F2*T2- 0.83*F3*T3)))
        (E11 (VARIABLES S1 TA01 F3)
             (FORMULA (S1-TA01*F3/1.3)))
        (E12 (VARIABLES X13 X15)
             (FORMULA (X13-X15)))
        (E13 (VARIABLES F3 F5)
             (FORMULA (F3-F5)))
        (E14 (VARIABLES X14 X12)
             (FORMULA (X14-X12)))
        (E15 (VARIABLES F4 F2)
             (FORMULA (F4-F2)))
        (E16 (VARIABLES F3 T3 F5 T5 Q2)
             (FORMULA (0.83*F3*T3-0.85*F5*T5 + Q2)))
        (E17 (VARIABLES F4 T4 F2 T2 Q2)
             (FORMULA (0.84*F4*T4   0.82*F2*T2 - Q2)))
        (E18 (VARIABLES DTL2 T2 T3 T4 T5 )
             (FORMULA (DTL2 -((T2-T3) - (T4-T5))/ LN( (T2-T3)/(T4-T5)) )))
        (E19 (VARIABLES Q2 A2 DTL2 )
             (FORMULA (Q2-1.0*A2*DTL2)))
        (E20 (VARIABLES F5 F6 F7)
             (FORMULA (F5=F6 F7)))
        (E21 (VARIABLES  F5 F6 F7  X15 X16 X17 )
             (FORMULA (F5*X15 + F6*X16 - F7*X17 )))
        (E22 (VARIABLES F5 F6 F7 X25 X26 X27)
             (FORMULA (F5*X25 + F6*X26 - F7*X27)))
        (E23 (VARIABLES  F5 F6 F7 X35 X36 X37)
             (FORMULA ( F5*X35 + F6*X36 - F7*X37 )))
        (E24 (VARIABLES S3 TA03 F7)
             (FORMULA (S3-TA03*F7/1.07)))
        (E25 (VARIABLES  F5 T5 F6 T6 F7 T7)
             (FORMULA (0.85*F5*T5 + 0.86*F6*T6 - 0.87*F7*T7))) )))
```

```
PROGRAM SIMULADOR;                    (*  A:MIM4.PAS  *)


(*      Este  programa  fue   CODIFICADO  EN  FORMA  AUTOMATICA        *)
(*      por  el   SISTEMA   MP    escrito    en    COMMON   LISP       *)
(*                                                                     *)
(*      El Sistema   MP   fue  escrito por   CARLOS ROJAS GUZMAN       *)
(*                                                                     *)
(*      con el apoyo del Dr.Ignacio Avila y del  Dr. Rene Banares      *)
(*      directores del grupo de   Diseno de  Procesos  Asistido        *)
(*      por Computadora    de la  FACULTAD  de  QUIMICA,   de la       *)
(*                 Universidad Nacional Autonoma de Mexico             *)
(*                              M E X I C O                            *)



CONST
    nmax = 1 ;          (*  DIMENSIONES DEL MAYOR SISTEMA DE ECUACIONES  *)
    nblo = 25 ;         (*  NUMERO DE BLOQUES  *)
    inc  = 0.001 ;      (* INCREMENTO PARA APROXIMAR DERIVADAS *)

TYPE
    mcuadrada = array [1..nmax,1..nmax] of real;
    matlineal = array [1..nmax] of real;
    matindices = array [1..nmax] of integer;

VAR
    x : matlineal;
    bloque, orden : integer;
                                  (*CONJUNTO DE VARIABLES DEL MODELO *)
    X23 : real ;
    X46 : real ;
    X47 : real ;
    X22 : real ;
    X12 : real ;
    T1  : real ;
    F1  : real ;
    TAO1 : real ;
    S1  : real ;
    X13 : real ;
    F3  : real ;
    F2  : real ;
    F4  : real ;
    T4  : real ;
    T3  : real ;
    T2  : real ;
    DTL2 : real ;
    A2  : real ;
    Q2  : real ;
    X17 : real ;
    X15 : real ;
    X27 : real ;
    X25 : real ;
    X37 : real ;
    X35 : real ;
    T7  : real ;
    T5  : real ;
    F6  : real ;
    F5  : real ;
    F7  : real ;
    TAO3 : real ;
    S3  : real ;


(*SUBSISTEMA DE ECUACIONES NUMERO 1  *)
PROCEDURE  SIST1  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    F1 := xd[1];
                    (* FUNCION  E5  *)
    beta[1] := - (F1+F2-F3) ;
end;
```

```
(*SUBSISTEMA DE ECUACIONES NUMERO 2  *)
PROCEDURE  SIST2  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    T1 := xd[1];
                    (* FUNCION  E7   *)
    beta[1] := - (0.81*F1*T1 + 0.82*F2*T2 - 0.83*F3*T3) ;
end;

(*SUBSISTEMA DE ECUACIONES NUMERO 3  *)
PROCEDURE  SIST3  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    S1 := xd[1];
                    (* FUNCION  E8   *)
    beta[1] := - (S1 - TA01*F3/1.3) ;
end;


(*SUBSISTEMA DE ECUACIONES NUMERO 4  *)
PROCEDURE  SIST4  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X46 := xd[1];
                    (* FUNCION  E3   *)
    beta[1] := - (X46*X36-1) ;
end;


(*SUBSISTEMA DE ECUACIONES NUMERO 5  *)
PROCEDURE  SIST5  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X27 := xd[1];
                    (* FUNCION  E16  *)
    beta[1] := - (F5*X25 - F7*X27) ;
end;


(*SUBSISTEMA DE ECUACIONES NUMERO 6  *)
PROCEDURE  SIST6  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    T4 := xd[1];
                    (* FUNCION  E13  *)
    beta[1] := - (0.84*F4*T4 - 0.82*F2*T2 - Q2) ;
end;

(*SUBSISTEMA DE ECUACIONES NUMERO 7  *)
PROCEDURE  SIST7  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    F4 := xd[1];
                    (* FUNCION  E11  *)
    beta[1] := - (F4-F2) ;
end;


(*SUBSISTEMA DE ECUACIONES NUMERO 8  *)
PROCEDURE  SIST8  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    Q2 := xd[1];
                    (* FUNCION  E12  *)
    beta[1] := - (0.83*F3*T3-0.85*F5*T5 + Q2) ;
end;


(*SUBSISTEMA DE ECUACIONES NUMERO 9  *)
PROCEDURE  SIST9  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    F5 := xd[1];
                    (* FUNCION  E10  *)
    beta[1] := - (F3-F5) ;
end;
```

```
{*SUBSISTEMA DE ECUACIONES NUMERO 10  *}
PROCEDURE  SIST10  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X15 := xd[1];
                        {* FUNCION  E9  *}
    beta[1] := - (X13-X15) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 11  *}
PROCEDURE  SIST11  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X13 := xd[1];
                        {* FUNCION  E6  *}
    beta[1] := - (F1+F2*X12-F3*X13) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 12  *}
PROCEDURE  SIST12  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    F7 := xd[1];
                        {* FUNCION  E16  *}
    beta[1] := - (F5+F6-F7) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 13  *}
PROCEDURE  SIST13  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X37 := xd[1];
                        {* FUNCION  E19  *}
    beta[1] := - (F6*X36 -F7*X37) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 14  *}
PROCEDURE  SIST14  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X23 := xd[1];
                        {* FUNCION  E1  *}
    beta[1] := - (X13+X23-1) ;
end;

{*SUBSISTEMA DE ECUACIONES NUMERO 15  *}
PROCEDURE  SIST15  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    S3 := xd[1];
                        {* FUNCION  E21  *}
    beta[1] := - (S3-TA03*F7/1.07) ;
end;

{*SUBSISTEMA DE ECUACIONES NUMERO 16  *}
PROCEDURE  SIST16  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X15 := xd[1];
                        {* FUNCION  E9  *}
    beta[1] := - (X13-X15) ;
end;


{*SUBSISTEMA DE ECUACIONES NUMERO 17  *}
PROCEDURE  SIST17  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X25 := xd[1];
                        {* FUNCION  E2  *}
    beta[1] := - (X15+X25-1) ;
end;
```

```
(*SUBSISTEMA DE ECUACIONES NUMERO 18  *)
PROCEDURE  SIST18  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X17 := xd[1];
                        (* FUNCION  E17  *)
    beta[1] := - (F5*X15 - F7*X17) ;
end;


(*SUBSISTEMA DE ECUACIONES NUMERO 19  *)
PROCEDURE  SIST19  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    Q2 := xd[1];
                        (* FUNCION  E12  *)
    beta[1] := - (0.83*F3*T3-0.85*F5*T5 + Q2) ;
end;


(*SUBSISTEMA DE ECUACIONES NUMERO 20  *)
PROCEDURE  SIST20  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    T4 := xd[1];
                        (* FUNCION  E13  *)
    beta[1] := - (0.84*F4*T4 - 0.82*F2*T2 - Q2) ;
end;


(*SUBSISTEMA DE ECUACIONES NUMERO 21  *)
PROCEDURE  SIST21  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    DTL2 := xd[1];
                        (* FUNCION  E14  *)
    beta[1] := - (DTL2 - ((T2-T3) - (T4-T5)) / LN ((T2-T3) / (T4-T5))) ;
end;

(*SUBSISTEMA DE ECUACIONES NUMERO 22  *)
PROCEDURE  SIST22  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    A2 := xd[1];
                        (* FUNCION  E15  *)
    beta[1] := - (Q2 1.0*A2*DTL2) ;
end;

(*SUBSISTEMA DE ECUACIONES NUMERO 23  *)
PROCEDURE  SIST23  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X27 := xd[1];
                        (* FUNCION  E18  *)
    beta[1] := - (F5*X25 - F7*X27) ;
end;


(*SUBSISTEMA DE ECUACIONES NUMERO 24  *)
PROCEDURE  SIST24  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    X47 := xd[1];
                        (* FUNCION  E4  *)
    beta[1] := - (X17*X27*X37*X47-1) ;
end;


(*SUBSISTEMA DE ECUACIONES NUMERO 25  *)
PROCEDURE  SIST25  (xd: matlineal; var alfa: mcuadrada; var beta: matlineal);
begin
    T6 := xd[1];
                        (* FUNCION  E20  *)
    beta[1] := - (0.85*F5*T5 + 0.85*F6*T6 - 0.87*F7*T7) ;
end;
```

```
PROCEDURE EVALUA (bloque: integer; xd: matlineal; var alfa: mcuadrada;
                  var beta: matlineal);
begin
    case bloque of
    1 : SIST1 (xd,alfa,beta) ;
    2 : SIST2 (xd,alfa,beta) ;
    3 : SIST3 (xd,alfa,beta) ;
    4 : SIST4 (xd,alfa,beta) ;
    5 : SIST5 (xd,alfa,beta) ;
    6 : SIST6 (xd,alfa,beta) ;
    7 : SIST7 (xd,alfa,beta) ;
    8 : SIST8 (xd,alfa,beta) ;
    9 : SIST9 (xd,alfa,beta) ;
    10 : SIST10 (xd,alfa,beta) ;
    11 : SIST11 (xd,alfa,beta) ;
    12 : SIST12 (xd,alfa,beta) ;
    13 : SIST13 (xd,alfa,beta) ;
    14 : SIST14 (xd,alfa,beta) ;
    15 : SIST15 (xd,alfa,beta) ;
    16 : SIST16 (xd,alfa,beta) ;
    17 : SIST17 (xd,alfa,beta) ;
    18 : SIST18 (xd,alfa,beta) ;
    19 : SIST19 (xd,alfa,beta) ;
    20 : SIST20 (xd,alfa,beta) ;
    21 : SIST21 (xd,alfa,beta) ;
    22 : SIST22 (xd,alfa,beta) ;
    23 : SIST23 (xd,alfa,beta) ;
    24 : SIST24 (xd,alfa,beta) ;
    25 : SIST25 (xd,alfa,beta)
    end;
end;


PROCEDURE DERIVYFIN  (bloque :integer; x :matlineal; n :integer;
                      var alfa :mcuadrada; var beta :matlineal);
var xd :matlineal; a, b, c :integer; betabase :matlineal;
begin
    evalua (bloque, x, alfa, beta);
    for c := 1 to n do betabase[c] := beta[c];    (* F  valuado  en  Xo *)
    for a := 1 to n do                            (* para cada variable *)
    begin
        xd := x ; xd[a] := x[a] +inc;
        evalua (bloque, xd, alfa, beta);
        for b := 1 to n do                        (* para cada FUNCION  *)
            begin
            alfa[b,a] := (betabase[b] - beta[b]) / inc
            end;
    end;
end;


PROCEDURE LUTRIANG (bloque :integer; var a :mcuadrada; n :integer;
                    var indice :matindices; var d :real);

Const
    epsilon = 1.0e-20;

Var
    k, i, imax, j :integer; suma, val, cota :real; vv :matlineal;

begin
    d := 1.0;
    for i := 1 to n do begin
        cota := 0.0;
        for j := 1 to n do if ( abs(a[i,j]) > cota) then cota := abs(a[i,j]);
        if (cota = 0.0) then begin
            writeln ('aviso desde LUTRIANG - la matriz del sistema ',
            bloque, ' es singular'); readln; readln;
        end;
        vv[i] := 1.0 / cota  end;
    for j := 1 to n do begin
        for i := 1 to j-1 do begin
            suma := a[i,j];
            for k := 1 to i-1 do begin
```

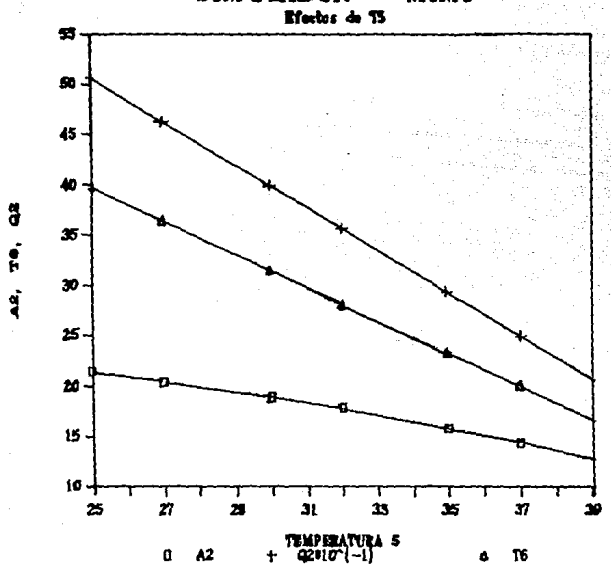
```
            suma := suma - a[i,j]*a[k,j]  end;
        a[i,j] := suma end;
    cota := 0.0;
    for i := j to n do begin
        suma := a[i,j];
        for k := 1 to j-1 do begin
            suma := suma - a[i,k]*a[k,j]  end;
        a[i,j] := suma;
        val := vv[i]*abs(suma);
        if ( val )= cota ) then begin
            cota := val; imax := i end end;
    if ( j <> imax) then begin
        for k := 1 to n do begin
            val := a[imax,k]; a[imax,k] := a[j,k]; a[j,k] := val  end;
        d := -d; vv[imax] := vv[j] end;
    indice[j] := imax;
    if (a[j,j] = 0.0) then a[j,j] := epsilon;
    if ( j <> n ) then begin
        val := 1.0 / a[j,j];
        for i := j+1 to n do begin
            a[i,j] := a[i,j]*val end end end;
end;

PROCEDURE LUIRESOL (a :mcuadrada; n :integer; indice :matindices;
                    var b: matlineal);

Var
    i, ip, ii, j :integer; suma :real;
begin
    ii := 0;
    for i := 1 to n do begin
        ip := indice[i]; suma := b[ip]; b[ip] := b[i];
        if ( ii<>0 ) then begin
            for j := ii to i-1 do begin
            suma := suma-a[i,j]*b[j] end end
        else if ( suma <> 0 ) then begin
            ii := i end;
        b[i] := suma end;
    for i := n downto 1 do begin
        suma := b[i];
        if ( i<n ) then begin
            for j := i+1 to n do begin
                suma := suma - a[i,j]*b[j] end end;
        b[i] := suma / a[i,i] end
end;

PROCEDURE RESUELVE (bloque ,iteramax :integer; var x :matlineal; n :integer;
                    tolx, tolf :real);
label 99;
Var
    k, i :integer; errx, errf, d :real; beta :matlineal; alfa :mcuadrada;
    indice :matindices;

begin
    for k := 1 to iteramax do begin
    derivyfun (bloque, x, n, alfa, beta) ; errf := 0.0;
    for i := 1 to n do errf := errf + abs (beta[i]);
    if (errf <= tolf) then goto 99;
    luirisanq(bloque,alfa,n,indice,d);
    luiresol(alfa,n,indice,beta);
    errx := 0.0;
    for i := 1 to n do begin
            errx := errx + abs(beta[i]); x[i] := x[i] + beta[i];
            end;
        if (errx <= tolx) then goto 99
    end;
99: end;
```

```
PROCEDURE PRESENTACION; (* presentacion inicial en pantalla *)
Var q :integer;
begin
    for q := 1 to 25 do writeln;
    writeln ('Este programa fue CODIFICADO AUTOMATICAMENTE ... etc.');
    for q := 1 to 10 do writeln;
    writeln('          SIMULADOR DE PROCESOS  MEN' );
    for q:= 1 to 10 do writeln;
end;

PROCEDURE VCONOCIDAS;(* pide al usuario el valor de variables conocidas *)
begin
    writeln ('Necesito el valor de las siguientes variables que conoces');
    write(' X12 = '); read ( X12 );
    write(' X36 = '); read ( X36 );
    write(' T5 = '); read ( T5 );
    write(' X22 = '); read ( X22 );
end;

PROCEDURE VDECISION; (* pide el valor de las variables de decision *)
begin
    writeln ('Necesito ahora el valor de las variables de decision');
    write (' F3 = '); read ( F3 );
    write (' F2 = '); read ( F2 );
    write (' F4 = '); read ( F4 );
    write (' T3 = '); read ( T3 );
    write (' T2 = '); read ( T2 );
    write (' TAO1 = '); read ( TAO1 );
    write (' T7 = '); read ( T7 );
    write (' TAO3 = '); read ( TAO3 );
end;

PROCEDURE ESTIMACION; (* El usuario da estimacion inicial a incognitas *)
begin
    write (' X23  = '); read( X23 );
    write (' X46  = '); read( X46 );
    write (' X47  = '); read( X47 );
    write (' T1   = '); read( T1 );
    write (' F1   = '); read( F1 );
    write (' S1   = '); read( S1 );
    write (' X13  = '); read( X13 );
    write (' F4   = '); read( F4 );
    write (' T4   = '); read( T4 );
    write (' DTL2 = '); read( DTL2 );
    write (' A2   = '); read( A2 );
    write (' B2   = '); read( B2 );
    write (' X17  = '); read( X17 );
    write (' X15  = '); read( X15 );
    write (' X27  = '); read( X27 );
    write (' X25  = '); read( X25 );
    write (' X37  = '); read( X37 );
    write (' T6   = '); read( T6 );
    write (' F5   = '); read( F5 );
    write (' F7   = '); read( F7 );
    write (' S3   = '); read( S3 );
end;

PROCEDURE AUTOGENERADOS;(* Se da un valor inicial para cada variable *)
begin
    X23 := 0.5;
    X46 := 0.5;
    X47 := 0.5;
    T1  := 0.5;
    F1  := 0.5;
    S1  := 0.5;
    X13 := 0.5;
    F4  := 0.5;
    T4  := 0.5;
    DTL2 := 0.5;
    A2  := 0.5;
    B2  := 0.5;
```

```
    X17 := 0.5;
    X15 := 0.5;
    X27 := 0.5;
    X25 := 0.5;
    X37 := 0.5;
    T6 := 0.5;
    F5 := 0.5;
    F7 := 0.5;
    S3 := 0.5;
end;


PROCEDURE INICIA;  (* pide al usuario valor inicial de incognitas, OPCIONAL *)
Var opinicia :integer;
begin
    writeln('Quieres dar una primera estimacion para las incognitas(1/0)');
    read (opinicia);
    if opinicia = 1 then ESTIMACION else AUTOGENERADOS;
end;


PROCEDURE RESULTADOS;  (* muestra el conjunto final de variables y valores *)
begin

    writeln('Se ha encontrado la siguiente solucion al sistema ');
    writeln('Los valores de las variables conocidas son');
    writeln (' X12 = ', X12);
    writeln (' X36 = ', X36);
    writeln (' T5 = ', T5);
    writeln (' X22 = ', X22);
    writeln ('Los valores dados a las variables de decision son');
    writeln (' F3 = ', F3);
    writeln (' F2 = ', F2);
    writeln (' F6 = ', F6);
    writeln (' T3 = ', T3);
    writeln (' T2 = ', T2);
    writeln (' TA01 = ', TA01);
    writeln (' T7 = ', T7);
    writeln (' TA03 = ', TA03);
    writeln ('Y los valores encontrados para las incognitas');
    writeln (' X23 = ', X23);
    writeln (' X46 = ', X46);
    writeln (' X47 = ', X47);
    writeln (' T1 = ', T1);
    writeln (' F1 = ', F1);
    writeln (' S1 = ', S1);
    writeln (' X13 = ', X13);
    writeln (' F4 = ', F4);
    writeln (' T4 = ', T4);
    writeln (' DIL2 = ', DIL2);
    writeln (' A2 = ', A2);
    writeln (' B2 = ', B2);
    writeln (' X17 = ', X17);
    writeln (' X15 = ', X15);
    writeln (' X27 = ', X27);
    writeln (' X25 = ', X25);
    writeln (' X37 = ', X37);
    writeln (' T6 = ', T6);
    writeln (' F5 = ', F5);
    writeln (' F7 = ', F7);
    writeln (' S3 = ', S3);
    readln;
end;


(*.....PROGRAMA PRINCIPAL.....  .....*)
begin
    PRESENTACION;
    VCONOCIDAS;
    VDECISION;
    INICIA;


    A[1] := F1;
    RESUELVE (1,50,x,1,1e-9,1e-9);
    F1 := x[1];        writeln (' F1 = ', x[1] );
```

```
x[1] := T1;
RESOLVE (2,50,   1,1e-9,1e-9);
T1 :=  x[1];        writeln (' T1 = ', x[1] );


x[1] := S1;
RESOLVE (3,50,x,1,1e-9,1e-9);
S1 := x[1];         writeln (' S1 = ', x[1] );


x[1] := X46;
RESOLVE (4,50,x,1,1e-9,1e-9);
X46 := x[1];        writeln (' X46 = ', x[1] );


x[1] := X27;
RESOLVE (5,50,x,1,1e-9,1e-9);
X27 := x[1];        writeln (' X27 = ', x[1] );


x[1] := T4;
RESOLVE (6,50,x,1,1e-9,1e-9);
T4 := x[1];         writeln (' T4 = ', x[1] );


x[1] := F4;
RESOLVE (7,50,x,1,1e-9,1e-9);
F4 := x[1];         writeln (' F4 = ', x[1] );


x[1] := Q2;
RESOLVE (8,50,x,1,1e-9,1e-9);
Q2 := x[1];         writeln (' Q2 = ', x[1] );


x[1] := F5;
RESOLVE (9,50,x,1,1e-9,1e-9);
F5 := x[1];         writeln (' F5 = ', x[1] );


x[1] := X15;
RESOLVE (10,50,x,1,1e-9,1e-9);
X15 := x[1];        writeln (' X15 = ', x[1] );


x[1] := X13;
RESOLVE (11,50,x,1,1e-9,1e-9);
X13 := x[1];        writeln (' X13 = ', x[1] );


x[1] := F7;
RESOLVE (12,50,x,1,1e-9,1e-9);
F7 := x[1];         writeln (' F7 = ', x[1] );


x[1] := X37;
RESOLVE (13,50,x,1,1e-9,1e-9);
X37 := x[1];        writeln (' X37 = ', x[1] );


x[1] := X23;
RESOLVE (14,50,x,1,1e-9,1e-9);
X23 := x[1];        writeln (' X23 = ', x[1] );


x[1] := S3;
RESOLVE (15,50,x,1,1e-9,1e-9);
S3 := x[1];         writeln (' S3 = ', x[1] );


x[1] := X15;
RESOLVE (16,50,x,1,1e-9,1e-9);
X15 := x[1];        writeln (' X15 = ', x[1] );
```

```
    x[1] := X25;
    RESUELVE (17,50,x,1,1e-9,1e-9);
    X25 := x[1];        writeln (' X25 = ', x[1] );

    x[1] := X17;
    RESUELVE (18,50,x,1,1e-9,1e-9);
    X17 := x[1];        writeln (' X17 = ', x[1] );

    x[1] := B2;
    RESUELVE (19,50,x,1,1e-9,1e-9);
    B2 := x[1];         writeln (' B2 = ', x[1] );

    x[1] := T4;
    RESUELVE (20,50,x,1,1e-9,1e-9);
    T4 := x[1];         writeln (' T4 = ', x[1] );

    x[1] := DTL2;
    RESUELVE (21,50,x,1,1e-9,1e-9);
    DTL2 := x[1];       writeln (' DTL2 = ', x[1] );

    x[1] := A2;
    RESUELVE (22,50,x,1,1e-9,1e-9);
    A2 := x[1];         writeln (' A2 = ', x[1] );

    x[1] := X27;
    RESUELVE (23,50,x,1,1e-9,1e-9);
    X27 := x[1];        writeln (' X27 = ', x[1] );

    x[1] := X47;
    RESUELVE (24,50,x,1,1e-9,1e-9);
    X47 := x[1];        writeln (' X47 = ', x[1] );

    x[1] := T6;
    RESUELVE (25,50,x,1,1e-9,1e-9);
    T6 := x[1];         writeln (' T6 = ', x[1] );

readln;

RESULTADOS;

end.
```


SIMULADOR MIM4
Efectos de T5
A2, T6, Q2
55
50
45
40
35
30
25
20
15
10
25
27
29
31
33
35
37
39
TEMPERATURA 5
□ A2 + Q2*10^(-1) △ T6

Bibliografía

Bañares Alcántara, René. AN AUTOMATIC PROGRAMMING TOOL FOR PROCESS SIMULATION, AAAI 88 Workshop on Artificial Intelligence in Process Engineering, St. Paul, Minnesota, (Ago. 1988).

Bañares Alcántara, René y Eduardo Morales M. DOS HERRAMIENTAS BASADAS EN TECNICAS DE INTELIGENCIA ARTIFICIAL PARA EL AUXILIO EN LA CONSTRUCCION DE SIMULADORES DE PROCESOS. V Reunión Nacional de Inteligencia Artificial, Mérida, Yucatán, Marzo 1988.

Charniak, Eugene y Drew McDermott. INTRODUCTION TO ARTIFICIAL INTELLIGENCE. E.U.A.: Addison-Wesley Publishing Company, (1985).

Christensen, J.H. y D.F.Rudd. "Structuring Design Computations",AICHE J. 15, 94-100,(1969).

Jensen, Kathleen y Niklaus Wirth. PASCAL. USER MANUAL AND REPORT. 2a ed. NY, EUA: Springer Verlag, (1974).

Minsky, Marvin. "A Framework for Representing Knowledge" en THE PSYCHOLOGY OF COMPUTER VISION, ed. P.H.Winston. NY, E.U.A.: McGraw Hill, (1975).

Press, William H., B.P.Flannery, S. A.Teukolsky y W.T.Vetterling, NUMERICAL RECIPES, THE ART OF SCIENTIFIC COMPUTING. NY, E.U.A.: Cambridge University Press, (1986).

Ramirez,W.F. y C.R.Vestal. "Algorithms for Structuring Design Calculations". CHEM. ENG. SCI. 27, 2243-54, (1972).

Steele, Guy L. Jr. ET AL, COMMON LISP: THE LANGUAGE, E.U.A.: Digital Press, (1984).

Steward, D.V. "On an Approach to Techniques for the Analysis of the Structure of Large Systems of Equations", SIAM REV., 4, pp. 321-42, (1962).

Steward, D.V. "Partitioning and Tearing Systems of Equations",SIAM J. NUMERICAL ANALYSIS, 2, 345-65,(1965).

Tarjan,R. "Depth-first Search and Linear Graph Algorithms",SIAM J.COMPUT. vol.1, no. 2, 146-60, (Junio 1972).

Westerberg, A.W., H.P.Hutchinson, R.L.Motard y P.Winter. PROCESS FLOWSHEETING. NY, E.U.A.: Cambridge University Press, (1979).

Winston, Patrick Henry y Berthold Klaus Paul Horn, LISP 2a ed., E.U.A.: Addison-Wesley Publishing Company, (1984).

Wos, L., B.G. Buchanan, C.Green, ET AL. "An Overview of Automated Reasoning and Related Fields", JOURNAL OF AUTOMATED REASONING, 1,5-48, (1985).