16
2 Ejem

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

IMPLANTACION DE UN SISTEMA PARA ANALIZAR ESCENARIOS EN UN MODELO MATEMATICO DE ECUACIONES ALGEBRAICAS

T E S I S

QUE PARA OBTENER EL TITULO DE

A C T U A R I O

P R E S E N T A :

NORMA JULIANA CASTRO ROA

MEXICO D.F. AGOSTO 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

UNIVERSIDAD NACIONAL AVENTMA DE MEXICO

M. EN C. VIRGINIA ABRIN BATULE
Jefe de la División de Estudios Profesionales
Facultad de Ciencias
P r e s e n t e

Los abajo firmantes, comunicamos a Usted, que habiendo revisado el trabajo de Tesis que realiz(ó)ron la pasante(s) Norma Juliana Castro Roa

con número de cuenta 8752725-8 con el Título:

"IMPLANTACION DE UN SISTEMA PARA ANALIZAR ESCENARIOS EN UN MODELO MATEMATICO DE ECUACIONES ALGEBRAICAS".

Otorgamos nuestro **Voto Aprobatorio** y consideramos que a la brevedad deberá presentar su Examen Profesional para obtener el título de Actuario

GRADO	NOMBRE(S)	APELLIDOS COMPLETOS	FIRMA
Dr.	Gilberto	Calvillo Vives	
Director de Tesis M. en C.	Virginia	Abrin Batule	
Act.	Gerardo	Loredo Fuentes	
M. en C.	Beatriz Eugenia	Rodríguez Fernández	
Suplente Dr.	Raymundo Eugenio	Peralta Herrera	
Suplente			

A mis padres por enseñarme a enfrentar esta vida con trabajo, honestidad y amor; por siempre estar conmigo. Porque creen en mi, les dedico este trabajo de tesis, con todo el esfuerzo que representa.

A mis hermanos Rocio, José, Jesús y Yanelli por su increíble amistad, apoyo, respeto e inmenso amor que me dan todos los días, también para ustedes es esta tesis

Además dedico este trabajo a quienes también forman parte de mi vida:

A Josefina y Paty por entenderme y nunca dejarme sola, porque siempre están allí. Gracias por todos estos años de amistad y cariño.

A José Manuel por darme todo tu apoyo y amor, porque estas ahí cuando te necesito.

A todos mis amigos de la universidad, en especial a Carlos y Tere, porque esta tesis es la conclusión de toda una carrera que llevé con ustedes de una forma increíblemente padre.

AGRADECIMIENTOS

Un agradecimiento a todas aquellas personas que creyeron en mi y fueron pacientes con migo. A toda la gente que de una u otra manera mostró interés en mi trabajo, siempre con una palabra de ánimo que me ayudó a que esta tesis llegara a su fin.

Al Dr. Gilberto Calvillo por su gran paciencia durante todo este tiempo, su acertada dirección y su disposición para la realización de este trabajo.

Al Dr. Pablo Barrera por su apoyo incondicional al brindarme el material necesario para poder realizar la programación que se incluye en este trabajo.

Al Dr. Raymundo Peralta por el apoyoa en la programación del sistema y la revisión de esta tesis.

A los profesores Virginia Abrin, Gerardo Loredo y Bety Rodriguez por el tiempo dedicado a este trabajo; por sus observaciones y correcciones hechas al mismo.

Un especial agradecimiento al Dr. Manuel Galán por su siempre interés en este trabajo, su tiempo y gran apoyo brindados para la realización del mismo.

A las personas del Banco de México, por su apoyo y su tiempo.

IMPLANTACION DE UN SISTEMA PARA ANALIZAR ESCENARIOS EN UN MODELO MATEMATICO DE ECUACIONES ALGEBRAICAS.

INDICE

INDICE

INDICE

CAPITULO I

INTRODUCCION.

1.1 GENERALIDADES.

El objetivo de este trabajo de tesis, es crear un sistema de simulación determinística (SIDEM), para resolver diferentes modelos matemáticos, basados en sistemas de ecuaciones algebraicas. Este sistema nos ayudará a analizar diferentes escenarios que el usuario pueda crear, sujeto obviamente, a las restricciones mismas del problema.

El SIDEM esta enfocado a diferentes tipos de usuarios sean estos economistas, matemáticos, etc.; la manera de interactuar con el sistema es muy amigable brindando al usuario la facilidad de crear diferentes escenarios. Los modelos se especifican usando archivos de datos.

El SIDEM propuesto posee dos propiedades que debemos mencionar ya que facilitan el trabajo de especificación y de resolución del problema, estas son:

- Desde el punto de vista de la especificación del modelo, el sistema permite definir a este, mediante un sistema de ecuaciones de la forma

$$y=f(y,x)$$

sin importar el orden en el que dichas ecuaciones se introducen.

- Desde el punto de vista del análisis, el sistema permite especificar valores predeterminados o "metas", ajustando el valor de otras variables para llegar a la meta propuesta.

Para hacer su trabajo este sistema usa teoría de gráficas, lo cual es muy ventajoso porque la creación de la gráfica del modelo ayuda en el proceso de solución ya que a esta le podemos aplicar diferentes procedimientos, para manejar los datos del problema y de esta manera solucionar y analizar sus resultados.

Para dar solución al modelo, tomamos en cuenta las ecuaciones mencionadas, que en general constan de variables exógenas, endógenas y de control, mismas que forman las relaciones funcionales del sistema. Una vez que el sistema ha definido estos datos, tomando en cuenta si las variables endógenas dependen de otras variables endógenas, creamos la gráfica del modelo representando con nodos cualquiera de los tipos de variables y con arcos la dependencia entre cada una de estas variables.

Ya elaborada la gráfica, podemos aplicar el método de búsqueda primero a profundidad para encontrar un ordenamiento parcial sobre los nodos de la misma. Con este ordenamiento es posible recorrer la gráfica y buscar los ciclos formados dentro de esta, y así definir cuales y cuantas componentes fuertemente conexas existen en la gráfica y el orden que estas tienen.

Con la información obtenida de la gráfica, es posible plantear la solución al sistema de ecuaciones del problema de la siguiente manera:

1. Tomamos las componentes fuertemente conexas del sistema y el orden parcial de la gráfica.

2. De acuerdo a esto se ordenan las ecuaciones de manera que: en primer lugar se encuentran las ecuaciones de variables endógenas que únicamente estén formadas por variables exógenas; en segundo lugar a las ecuaciones de variables endógenas que se encuentran en la primera componente fuertemente conexa; en seguida las ecuaciones de la segunda componente conexa y así sucesivamente.

3. Se da una primera solución o "solución parcial" para las variables que dependen únicamente de variables exógenas y para las variables en la primera componente fuertemente conexa.

4. Se sustituyen los valores obtenidos en la primera solución parcial del sistema general, se resuelve nuevamente y se obtiene una segunda "solución parcial", incluyendo ahora en estos resultados a las variables de la segunda componente fuertemente conexa y a las variables que estando fuera de componentes dependen de variables en la primera componente. Este procedimiento se repite hasta agotar las ecuaciones del sistema.

5. Cuando se ha dado solución a todo el sistema, se da por terminado el proceso.

Ya concluida la primera parte del sistema, se da entrada a la consulta de cuadros y al análisis de escenarios nuevos, los cuales son creados por el usuario según sus necesidades y objetivos.

Esta segunda parte corresponde a un análisis de sensibilidad, ya que podemos observar el cambio registrado en una variable v1 ante un cambio en la variable v2, cuando v1 depende de v2.

1.2 EL SISTEMA

La herramienta desarrollada para este trabajo, es un programa de computadora, el cual resulta muy sencillo de manejar. Este contiene los procedimientos para crear, en base a los datos del problema, su gráfica respectiva y encontrar las componentes fuertemente conexas de ésta.

Se resuelve el sistema de la forma ya indicada, mostrando la solución en cuadros con además la posibilidad de analizar los datos del problema por medio de los diferentes escenarios que se pueden crear, fijando metas y dejando libres otras variables siendo esto de gran utilidad.

Este sistema representa ventajas: en problemas en donde el número de variables involucradas es grande, el elaborar la gráfica a mano llevaría bastante tiempo, y aún mas el tratar de aplicar cualquier procedimiento a la misma. Con esto el proceso de analizar diferentes escenarios se torna sencillo.

Este sistema debe ser visto como un sistema de ayuda para la resolución de problemas algebraicos grandes.

Los lenguajes usados en la programación del sistema fueron PASCAL y FORTRAN. PASCAL se usó para el manejo de datos y creación de la gráfica del problema y FORTRAN se usó ya que el algoritmo de solución que se usa es el del paquete GRG2.

1.3 ESTRUCTURA DEL TRABAJO.

Este trabajo está estructurado de la siguiente manera.

En el capítulo II describimos en general, los tipos de modelos y sus partes. Definimos a los modelos recursivos y los modelos simultáneos, la representación gráfica de modelos algebraicos y el análisis de éstas.

En el capítulo III definimos el orden topológico y mencionamos el algoritmo para sumergir un orden parcial en un orden lineal. Se da la interpretación de la cerradura transitiva en gráficas. Se explica que son las componentes fuertemente conexas y como encontramos éstas en una gráfica. Por último se menciona el cálculo de derivadas en un modelo recursivo.

En el capítulo IV se da la descripción del sistema. Aquí vemos la relación de los modelos y los algoritmos que se que se mencionaron en los capítulos anteriores y la manera en que estos se relacionan con el sistema creado para esta tesis. También se da una referencia del optimizador utilizado.

La descripción del programa se da en el capítulo V, ahí se menciona, a grandes rasgos, el funcionamiento de subrutinas, funciones y procedimientos usados en la programación, tanto de la primera parte programada en PASCAL, como de la segunda parte programada en FORTRAN. El código de máquina se puede consultar en el apéndice A de este trabajo.

Un ejemplo resuelto con este sistema se da en el capítulo VI. Dando un ejemplo de los archivos de datos, con solución y diferentes escenarios.

En el apéndice B se dan algunas descripciones básicas de redes.

En conclusión, el objetivo de esta tesis es el de proporcionar una herramienta para poder analizar modelos matemáticos bajo diferentes escenarios, la manera en que lo hacemos es usando teoría de gráficas, sin profundizar en esta.

CAPITULO II

SISTEMAS DE ECUACIONES ALGEBRAICAS COMO MODELOS MATEMATICOS Y SU REPRESENTACION GRAFICA.

2.1 GENERALIDADES.

Cuando queremos dar la definición de modelo, encontramos que ésta podría variar dependiendo del campo en el que estamos trabajando. En general podemos decir que un modelo es la representación de un fenómeno dado.

Tomemos aquellos fenómenos llamados fenómenos de sistemas reales. El fenómeno de un sistema real es representado por un modelo de tal manera que éste puede explicar al fenómeno, predecirlo y controlarlo. [7].

Dos características que un modelo debe tener son: 1) que sea realista y 2) que sea manejable. Que sea realista significa que el modelo debe de representar razonablemente al sistema que está modelando, esto es, debe de incorporar los principales elementos del fenómeno en representación; cuando hablamos de manejabilidad, generalmente está involucrado el proceso de idealización, incluyendo la eliminación de influencias que nos parezcan despreciables; se trata de poder obtener resultados del modelo. Estas dos características importantes se deben de aplicar cuidadosamente, pues el proceso de idealización podría hacer al modelo tan poco realista que no tuviera poder de explicación o de predicción, y por otra parte el nivel de complejidad de un modelo muy realista podría significar la inutilidad del mismo, debido a su poca manejabilidad.

2.2 MODELOS ALGEBRAICOS.

Los modelos se pueden clasificar de acuerdo a su área de aplicación o bien de acuerdo a su estructura formal. Así podemos hablar de modelos Económicos, Físicos, Biológicos, etc., pero también de modelos de optimización, de ecuaciones diferenciales, de ecuaciones algebraicas, etc..

En esta tesis estamos interesados en modelos basados en ecuaciones algebraicas para cualquier área del conocimiento, aunque cabe mencionar que la principal motivación se deriva de modelos económicos y econométricos.

Debido a esto, parte de la nomenclatura usada en este trabajo proviene del argot económico.

Un modelo algebraico consta de un conjunto de variables y un conjunto de ecuaciones que indican la relación existente entre las variables; cada ecuación tiene un significado y función determinada en el modelo. Las ecuaciones pueden ser lineales o no lineales.

Durante largo tiempo, se usaron en economía modelos geométricos. Estos usan diagramas para indicar la relación existente entre las variables de determinada situación económica.

En general un sistema algebraico tiene ventajas sobre un modelo geométrico, algunas de éstas son: la facilidad de manejo (por ejemplo obtener derivadas), la capacidad para aumentar el número de variables y el número de ecuaciones, la libertad de actuar sin tener restricción a cierto número de dimensiones, la posibilidad de variar en tamaño y la capacidad para generalizar el modelo.

Por conveniencia adoptaremos aquí la clasificación de las variables usada por los economistas, esta es en variables exógenas, variables endógenas y variables de control. A continuación definiremos cada uno de estos elementos.

El modelo determina los valores de ciertas variables llamadas variables endógenas, estas son en conjunto las variables dependientes del modelo, las cuales se determinan por medio de las relaciones del mismo.

Los valores de las variables exógenas, se determinan fuera del sistema y no dependen del usuario del modelo. Ellas afectan al sistema mas no son afectadas por el sistema.

Las variables de control son desde el punto de vista matemático, parecidas a las exógenas ya que su valor no es determinado por medio de las relaciones del modelo. La diferencia con las exógenas, es que el valor de estas puede ser modificado por el usuario del modelo para controlar el comportamiento del sistema.

Por último los modelos contienen ciertas funciones, las cuales están previamente determinadas y relacionan las variables exógenas y de control con las variables endógenas. En modelos algebraicos podemos encontrar una gran variedad de formas funcionales.

Los modelos que se usarán en este trabajo son de la forma:

(1) $y_i = f_i(X,Y)$ $\quad i=1,\ldots,m.$

con $X = (x_1, \ldots, x_n) \in R^n$ y

$Y = (y_1, \ldots, y_m) \in R^m$

en donde:

X es un vector de n variables exógenas y de control

Y es un vector de m variables endógenas y

f_i son funciones que definen a las variables endógenas.

Tomando la estructura general del modelo algebraico y de acuerdo a la definición de cada una las variables, podemos clasificar los modelos en 2 grupos:

- Modelos recursivos
- Modelos simultáneos

Definiremos ambos modelos.

2.3 MODELOS RECURSIVOS.

Definimos a los modelos recursivos como aquellos de la forma:

(2) $\quad y_i = f_i(X, y_1, \ldots, y_{i-1}) \qquad i = 1, \ldots, m.$

En este modelo para poder llegar a una solución, la variable endógena **i** está en función de las variables exógenas y de las **(i-1)** variables endógenas anteriores.

Es decir, en un modelo recursivo, cada variable endógena puede ser determinada en términos de una función, de variables exógenas, de variables de control y de variables endógenas previamente determinadas.

Dicho de otra manera, en un modelo recursivo las ecuaciones se pueden ordenar de tal forma (he aquí la importancia del orden) que podemos resolver el sistema por sustitución a partir de los valores conocidos, es decir, no se podrá llegar a y_k si no se tiene el valor de y_{k-1}.

Cabe mencionar que en la práctica las variables pueden tener nombres arbitrarios y las relaciones del modelo pueden aparecer en un orden también arbitrario . Esto hace que no sea obvio ver si un modelo es recursivo o no. Este problema se analizará en detalle en el próximo capítulo.

Si desarrollamos (2) para cada una de las y_i obtenemos el sistema (3) con la siguiente estructura:

$y_1 = f_1(X) = F_1(X)$

$y_2 = f_2(X, y_1) = f_2(X,F_1(X)) = F_2(X)$

.

.

$y_m = f_m(X,y_1,y_2,...y_{m-1})=f_m(X,F_1(X),F_2(X),...F_{m-1}(X))=F_m(X)$

Al desarrollar las ecuaciones del modelo recursivo de esta manera podemos llegar a expresar el modelo en la forma reducida:

$$Y_i = F_i(X) \qquad i = 1,...,m.$$

La forma reducida de un modelo recursivo es importante, porque en él se muestra explícitamente la relación de cada variable endógena con las variables exógenas. En particular es posible calcular la variación de la i-ésima variable endógena al variar la j-ésima variable exógena mediante la derivada $\frac{\partial Fi}{\partial xj}$. Este tema se trata brevemente en el capítulo siguiente.

2.4 MODELOS SIMULTANEOS.

Definimos un modelo simultáneo como:

(4) $y_i = f(X, y_1, \ldots, y_m)$

es decir la variable endógena **i** está en función de variables exógenas y cualquiera de las **m** variables endógenas, la posible dependencia de y_i con y_k y al mismo tiempo de y_k con y_i, hacen del modelo un modelo simultáneo.

En este caso no es posible encontrar un orden de las ecuaciones en (4) que permita calcular secuencialmente el valor de cada una de las variables endógenas. Así mismo, tampoco es posible mediante el procedimiento de la sección anterior, obtener la forma reducida del modelo.

Un problema interesante en un modelo simultáneo es encontrar que tan localizada o generalizada es la simultaneidad. Este problema también se trata en el capítulo III.

2.5 REPRESENTACION GRAFICA.

Dado un modelo algebraico en la forma descrita en este capítulo, se le asocia una gráfica **G = (V,A)** donde **V={v1,...,vm}** es el conjunto de nodos o variables de todo tipo y **A={a1,...,ap}** es el conjunto de arcos, formado por aquellas parejas de vértices (u,v) donde **v** depende de **u**; es decir, existe una relación de la forma **v=f(...,u,...)**.

A esta gráfica la llamaremos la gráfica de relaciones funcionales del modelo, o simplemente gráfica funcional.

Ejemplo:

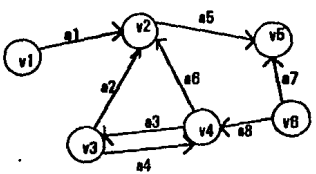
Sean **v1** y **v6** las variables exógenas y sean **v2,...,v5** las variables endógenas. La forma funcional del modelo algebraico así como la representación gráfica, se dan a continuación:

$v2 = f_2(v1,v3,v4)$

$v3 = f_3(v4)$

$v4 = f_4(v3,v6)$

$v5 = f_5(v2,v6)$



2.6 ANALISIS DE LAS GRAFICAS

La gráfica funcional de un modelo captura la estructura del mismo pero no su forma funcional. Es decir, refleja cuales son las relaciones de dependencia entre las variables, pero no la forma específica de la misma. Sin embargo esta estructura será suficiente para distinguir entre modelos recursivos y modelos simultáneos.

Si la gráfica es acíclica, es decir no tiene ciclos dirigidos, el modelo es recursivo [7]. El ordenamiento de las ecuaciones mencionado en la sección 2.3 corresponde en la gráfica a ordenar topológicamente sus nodos. La descripción de este orden se describe en el siguiente capítulo.

La simultaneidad de un modelo corresponde a la existencia de ciclos dirigidos en su gráfica funcional.

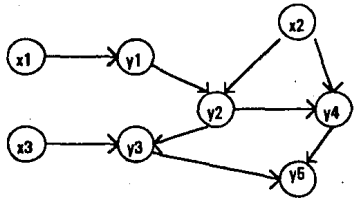
Si en la gráfica existen ciclos dirigidos, se puede llegar a otra forma canónica obteniendo las componentes fuertemente conexas del sistema y además ordenando dichas componentes.

Dos ejemplos de gráficas, uno de modelos recursivos y uno de modelos simultáneos, se muestran a continuación.

Considérese el modelo recursivo siguiente:

$y_1 = f_1(x_1)$
$y_2 = f_2(x_2, y_1)$
$y_3 = f_3(x_3, y_2)$
$y_4 = f_4(x_2, y_2)$
$y_5 = f_5(y_3, y_4)$

De acuerdo a este sistema podemos dibujar la gráfica, que resulta acíclica, correspondiente como se muestra a continuación:



Por otra parte considérese el sistema:

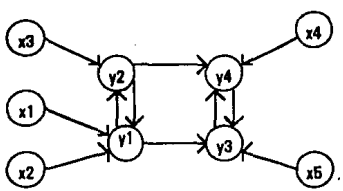
$y_1 = f_1(x_1, x_2, y_2)$

$y_2 = f_2(x_3, y_1)$

$y_3 = f_3(x_5, y_1, y_4)$

$y_4 = f_4(x_4, y_2, y_3)$

Siguiendo este sistema podemos dibujar la gráfica correspondiente; esta tiene dos ciclos como se muestra en la siguiente figura:



En este caso **y1** y **y2** forman una componente fuertemente conexa y **y3** y **y4** forman otra.

CAPITULO III

ANALISIS CUALITATIVO DE MODELOS USANDO TEORIA DE GRAFICAS

3.1 INTRODUCCION.

En el capítulo anterior se estableció la representación de la estructura de un modelo algebraico mediante su gráfica funcional. En este capítulo se presentan diversos algoritmos para gráficas que son útiles para analizar y explotar los modelos algebraicos mediante la mencionada representación.

En la primera parte del capítulo se presentan los algoritmos sin ninguna mención a los modelos. En la segunda parte se explica la utilidad de cada uno de ellos en la explotación de los modelos algebráicos.

3.2 RELACIONES, ORDENES PARCIALES Y SUS GRAFICAS.

Para facilitar la exposición de algunos conceptos se usará el lenguaje de relaciones y ordenes parciales.

Una relación **R** sobre un conjunto **V** es un subconjunto del producto cartesiano **VxV**.

A cada relación **R** sobre **V** le corresponde una gráfica G=(V,A) donde los arcos de G son precisamente los elementos de **R**.

La cerradura transitiva de una relación **R** es la relación $\Gamma(\mathbf{R})$ tal que $(\mathbf{u},\mathbf{v}) \in \Gamma(\mathbf{R})$ si y solo si, para todo $\mathbf{u} \neq \mathbf{v}$ existe en la gráfica asociada, una trayectoria de **u** a **v**.

3.3 ORDEN PARCIAL Y ORDEN LINEAL.

Definimos un orden parcial $P=(V,\langle_P)$ sobre un conjunto **V** como una relación $\langle_P$ entre los elementos de **V**, que satisface las siguientes propiedades para **x, y, z** que pertenecen a **V**:

1) Si $x \langle_P y$ y $y \langle_P z$ entonces $x \langle_P z$ (transitividad)

2) Si $x \langle_P y$ entonces no $y \langle_P x$ (asimetría)

3) **No** $x \langle_P x$ (antiflexividad)

.[11].

Debido a la transitividad y asimetría, la digráfica asociada a un orden parcial, es una gráfica acíclica.

Hemos visto como asociar a una relación una digráfica. Es claro que el proceso inverso también es posible. En efecto, a cualquier digráfica que no contenga arcos paralelos se le asocia una relación en la forma natural.

Una relación **R** cuya gráfica asociada es acíclica, puede extenderse a un orden parcial tomando su cerradura transitiva. Dicho orden será llamado el orden inducido por **R**.

Definimos un orden lineal $L=(S,\langle_L)$ como un orden parcial en el cual cualesquiera dos elementos **u** y **v** diferentes entre si son comparables, es decir, $u \langle v$ o $v \langle u$.

Inmersion de un orden parcial en uno lineal.

Decimos que el orden parcial P está inmerso en el orden lineal L si todos los elementos de P están en L y además $u \langle_P v$ implica $u \langle_L v$.

3.4 ALGORITMO DE ORDEN

Definimos a $v \in V$ como un elemento minimal del orden parcial P, si no existe $u \in V$ tal que $u \langle_P v$.

Los elementos del conjunto **V** se pueden ordenar linealmente de diferentes maneras. Para obtener uno de los posibles ordenamientos, se toma un elemento minimal **v** el cual no es precedido por otro elemento de **V** y lo ponemos a la cabeza de la lista (siempre existe uno con dichas características, de no ser así existiría un ciclo dirigido en la gráfica), borramos el elemento **v** del conjunto **V**, así como los arcos que salgan de él. El nuevo conjunto **V** sigue parcialmente ordenado, por lo tanto usamos el mismo algoritmo hasta dejar **V** vacío. De esta manera obtenemos un orden lineal, sobre los elementos de **V**. [11].

Para encontrar un elemento minimal de P se usará la gráfica asociada al orden.

Definimos la vecindad exterior de un vértice $v \in V$ como:

$$\delta^+(v) = \{(v,w) \in A: w \in V\}$$

es decir la vecindad está formada por el conjunto de arcos (v,w) tales que $v \langle w$.

Definimos la vecindad interior de un nodo **v** como:

$$\delta^{-}(v) = \{(u,v) \in A: u \in V\}$$

es decir la vecindad está formada por el conjunto de arcos **(u,v)** tales que **u ⟨ v.**

Dados estos conceptos podemos ahora definir *fuente* como un nodo **v** tal que su vecindad exterior es ≠ ∅ y su vecindad interior = ∅.

Es decir una fuente es un nodo del cual salen arcos y además ningún arco entra a este nodo.

3.4.1 ALGORITMO NUMERA

Con las definiciones dadas, podemos ahora describir un algoritmo, el cual numera los n nodos de la gráfica G de la siguiente manera: se le asigna el nivel 1 a un nodo **v** que sea fuente, se elimina este nodo de la gráfica y se busca un nodo que ahora sea fuente, a este se le asigna el nivel 2 y así sucesivamente hasta asignar el nivel n al último nodo del cual que no salgan mas arcos.

El código es el siguiente:

```
PROCEDIMIENTO NUMERA

i ← 1
K ←V
while K ≠ ∅ do
begin
        take one  v ∈ K
```

```
        while δ⁻(v) ≠ ∅ do
        begin
                find (u,v) ∈ A
                v ← u
        end
        l(v) = i
        i ← i + 1
        K ← K - {v}
End
```

3.5 COMPONENTES FUERTEMENTE CONEXAS

Si la gráfica de una relación **R** no es acíclica, dicha relación no se puede extender a un orden parcial; sin embargo se pueden ubicar las regiones de la relación en las que falta la transitividad y/o la asimetría.

Para hacer esto, se toma nuevamente la cerradura transitiva **Γ(R)** de la relación **R**. Esta relación **Γ(R)**, puede partirse en dos relaciones **S** y **T**. La relación **S** es la parte simétrica de **Γ(R)** y **T** es el complemento que llamaremos la parte transitiva.

La relación **S** es una relación de equivalencia sobre **V** si se postula la reflexividad de la misma. Las clases de equivalencia de **S** corresponden a las Componentes Fuertemente Conexas de la gráfica G de **R** y los elementos de **T** corresponden a los arcos de G que unen dichas componentes.

La digráfica cociente **G/S** tiene como vértices las clases de equivalencia de **S** y como arcos los elementos de **T**. Esta gráfica es acíclica y por tanto se pude ordenar linealmente como se indicó en la sección anterior.

3.6 LA CERRADURA TRANSITIVA EN LA GRAFICA DE RELACIONES FUNCIONALES.

Por definición de cerradura transitiva tenemos que si **u** depende de **v** y **v** depende de **w** entonces **u** depende de **w**, cumpliéndose la transitividad del orden parcial. Es decir, el arco **(w,u)** cumple la función de la trayectoria T_{WU} en **G** uniendo al nodo **w** con el nodo **u**; con esto vemos que la dependencia funcional define un orden parcial.

Observación. Si el modelo no es recursivo, no se cumple la transitividad.

3.7 MODELOS SIMULTANEOS.

Como mencionamos en el capítulo anterior, podemos dividir los modelos en recursivo y simultáneos, en estos últimos nos enfocaremos en esta sección.

Veamos ahora como se define una componente fuertemente conexa en términos de gráficas. Sea **G =(V,A)** una gráfica dirigida. Podemos particionar a **V**, los vértices de **G**, en clases de equivalencia **Vi**, **i=1,..,r**, tal que los vértices **vi** y **vj** están en la misma clase de equivalencia, si existe un camino de **vi** a **vj** y un camino de **vj** a **vi**.

A las clases obtenidas **V1,...,Vr**, se les llama componentes fuertemente conexas de la gráfica **G**, o equivalentemente, las **r** gráficas inducidas denotadas por **Gi=(Vi,Ai)** son llamadas componentes fuertemente conexas de **G** [4].

También podemos definir a las componentes fuertemente conexas por relación de equivalencia. Es posible establecer un orden para todos dos elementos **u** y **v** ε **V**;

con este orden se pueden determinar tres casos diferentes (el símbolo ≡ representa "equivalencia"):

1) **u<v**

2) **v<u**

3) **u≡v**

En el caso 1) tenemos que **u<v** si existe un trayecto T_{uv} pero no existe trayectoria T_{vu}.

En el caso 2) es similar existe una trayectoria T_{vu} pero no una trayectoria T_{uv}.

En el caso 3) tenemos que **u≡v** si existe una trayectoria T_{uv} y además una trayectoria T_{vu}. Para afirmar 3) demostraremos la siguiente propuesta:

Propuesta:
u y **v** forman una relación de equivalencia.

Demostración:
Sean **u** y **v** elemento distintos de **V**. Como **u≠v** entonces podemos unir estos elementos con un arco **(u,v)** que es equivalente a una trayectoria T_{uv}; de la misma manera unimos y por reflexividad, **v** y **u** con un arco **(v,u)** equivalente a una trayectoria T_{vu}.

Vemos que existen dos arcos **(u,v)≠(v,u)**, por lo que existen dos trayectorias $T_{uv}≠T_{vu}$, lo que implica un orden definido como **u<v** y **v<u**.

Ya que existen dos trayectoria y ambas condiciones de orden se cumplen entonces afirmamos que existe la relación de equivalencia entre los elementos u y $v \varepsilon V$.

A los elementos de la partición de V inducida por la relación de equivalencia, se les llaman componentes fuertemente conexas. Cada componente fuertemente conexa representa un subsistema en el cual cada variable depende de todas las otras variables del subsistema.

El sistema puede resolverse por partes, dando solución a los sistemas dados por las componentes en el orden obtenido por el algoritmo.

Se dice que una gráfica es fuertemente conexa, si todos los vértices y aristas de la gráfica, forman parte de una y solo una componente fuertemente conexa.

3.8 RECONOCIMIENTO DE COMPONENTES FUERTEMENTE CONEXAS

Un método para encontrar componentes fuertemente conexas, es la "búsqueda primero a profundidad", el cual es un método de visita a todos los vértices de una gráfica.

Búsqueda primero a profundidad en digráficas.

La búsqueda primero a profundidad se realiza de la siguiente manera: seleccionamos el vértice v y lo visitamos, seleccionamos cualquier arista (v,w) que salga de v y visitamos w, supongamos en general que x es el vértice visitado más recientemente, la búsqueda continúa seleccionando y explorando aristas (x,y) que salgan de x. Después de acabar con las aristas que salen de y,

regresamos a **x** aunque queden aristas dirigidas hacia **y** sin haber entrado a la búsqueda. El proceso de seleccionar arcos inexplorados continúa hasta que la lista de arcos se acaba. El método de visitar vértices es llamado búsqueda primero a profundidad, ya que sigue buscando hacia adelante (mas profundo) tanto como sea posible. [1].

Si la búsqueda primero a profundidad se realiza en una gráfica conexa, cada vértice será visitado y cada arista examinada.

Si la gráfica no es conexa, es decir, no está conectada entonces cada componente de la gráfica será analizada por separado. En este caso una vez terminada la búsqueda de una componente conexa, se selecciona cualquier otro vértice no visitado y se empieza nuevamente la búsqueda.

3.9 ORDEN PARCIAL EN COMPONENTES FUERTEMENTE CONEXAS

Ya descrito el orden parcial, la importancia que éste tiene en los modelos y una vez definidas las componentes fuertemente conexas, definiremos el orden parcial en componentes fuertemente conexas de la siguiente manera:

Sean **C1,C2,....,Ck** las componentes fuertemente conexas de **G**. Se toma la componente **Ci** de donde no salgan arcos de ninguno de sus vértices a ninguno de los vértices de las componentes restantes. Renumeramos a ésta componente con **C1**, repetimos el mismo procedimiento de tal manera que no haya arcos de ningún vértice de **C2** a ningún vértice de las restantes **k-2** componentes y así sucesivamente [4]. Es decir **C1** será la ultima componente a resolver en un sistema previamente ordenado en componentes fuertemente conexas.

3.10 DERIVADAS

En esta sección nos enfocaremos en el modelo recursivo descrito en el capítulo anterior.

Tomemos el sistema (3) que define un modelo recursivo. Expresamos la gráfica G de relaciones funcionales del sistema definida con las siguientes características:

$(y_i,y_j) \in A$

$y_i = f_i(...y_j...)$.

Definimos ahora el "peso" sobre el arco (y_i,y_j) como $\partial f_i/\partial y_j$, evaluada en un punto $x^0 = [x_1{}^0, x_2{}^0, ..., x_n{}^0]$.

A la red formada por la gráfica G se le conoce como "red de derivadas parciales del sistema" [10].

Digamos ahora que nos interesa conocer la variación del valor de la función, al variar el valor de una variable exógena, mas aún queremos conocer la variación del modelo al variar el conjunto de variables exógenas. Esto es, queremos conocer la derivada del modelo.

Tomemos la forma reducida del sistema, $Ym = F(X)$ y obtenemos su derivada $\partial F/\partial X$, la cual definimos como una expresión de la regla de la cadena de la siguiente manera:

$$\frac{\partial F}{\partial X} = \sum_{T \in \mathbf{T}} \prod_{(i,j) \in T} \frac{\partial fi}{\partial xj}$$

En donde **T** es la trayectoria de *i* a *j* y **T** es el conjunto de trayectorias del sistema.

Para calcular la parcial de F con respecto a la variable **X** se multiplican las parciales asociadas a los arcos de cada trayectoria del nodo x_i con el nodo y_j y se suman los productos sobre todas las trayectorias que unen estos nodos. Si no existe una trayectoria del nodo x_i al nodo y_j, entonces $\partial f_i/\partial x_j=0$ [10].

Esto es si el arco (x_j,y_i) no existe entonces tampoco existirá el peso que se le asignó, lo que es lo mismo el peso es 0. Así si multiplicamos los pesos sobre la trayectoria que une x_j con y_i este producto es igual a cero, pues no habrá dependencia sobre la variable x_j. Al hacer la suma sobre todas las trayectorias esta nos dará la variación total del modelo.

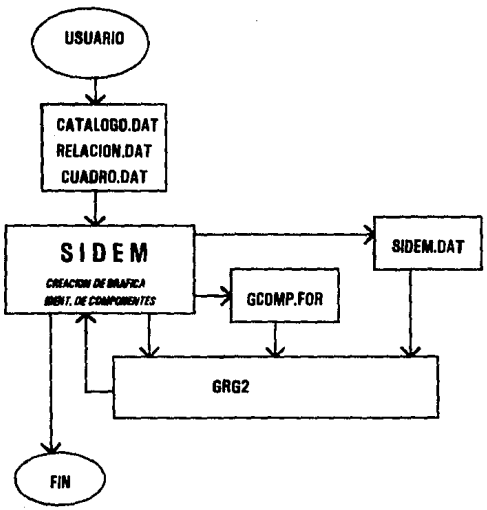
CAPITULO IV

DESCRIPCION DEL SISTEMA

4.1 GENERALIDADES

En los primeros capítulos presentamos diferentes tipos de modelos y sus características; también se mencionó brevemente a SIDEM como un sistema para analizar diferentes situaciones sobre un modelo dado. En este capítulo se presenta detalladamente a SIDEM, describiendo el funcionamiento de este para abordar diferentes tipos de modelos.

La estructura funcional de SIDEM se muestra en el siguiente diagrama de flujo.



Para ejecutar SIDEM el usuario tiene que proveer 3 archivos de datos: CATALOGO.DAT, RELACION.DAT Y CUADROS.DAT. El primero contiene como su nombre lo indica el catálogo de los nombres de las variables incluidas en el modelo, además incluye en el caso de variables exógenas el valor numérico de estas. En el segundo archivo, RELACION.DAT se describen las ecuaciones que definen al modelo y por último en el archivo CUADROS.DAT se incluyen los títulos de los cuadros de resultados indicando adicionalmente que variables van a ser presentadas en cada cuadro. El formato detallado de cada uno de los archivos se describe en el siguiente capítulo.[1]

Una vez creados estos tres archivos, se procede a ejecutar SIDEM. Lógicamente SIDEM realiza los siguientes tres pasos; primero crea la gráfica asociada al modelo, segundo determina la estructura de las componentes fuertemente conexas y por último resuelve el sistema. En las siguientes secciones se describen estos tres pasos.

4.2 SIDEM

SIDEM, es un paquete que permite realizar "simulaciones deterministas", de una situación que ha sido modelada mediante un sistema de ecuaciones (en general no lineales).

En SIDEM los sistemas de ecuaciones pueden ser tanto para modelos recursivos como para no recursivos. Precisamente una de las características de SIDEM es identificar si el modelo en estudio es recursivo o simultáneo, con la finalidad de obtener una solución mediante un tratamiento adecuado.

[1] Un ejemplo de estos archivos se muestra en el capítulo VI.

La estructura de las ecuaciones y las variables de los sistemas que usa SIDEM son las mismas que se describieron en el capítulo II, tanto para problemas recursivos como para problemas simultáneos. Esto es, se emplean ecuaciones de la forma $y_i = f_i(x,y)$. Adicionalmente se supondrá que las funciones f_i son diferenciables.

SIDEM es un sistema capaz de analizar un modelo algebraico, procesar los datos y resolver el sistema; además facilita el análisis de diferentes escenarios que el usuario quiera estudiar.

Para facilitar la exposición introduciremos algunas definiciones y notación.

Cabe hacer la observación que cada una de las funciones del sistema (2) presentado en el capítulo II tienen por lo general menos argumentos de los indicados en dicho sistema. Tomando en cuenta lo anterior se puede representar dicho sistema (2) de la siguiente manera:

$$4) \qquad z_i = f_i (Z_{p(i)}) \qquad\qquad i \in \{1,...N+M\}$$

donde:

x_n : son las variables exógenas del sistema, y n es un índice que varía de 1 hasta en número total de variables exógenas N.

y_m : son las variables endógenas del sistema, y m es un índice que varía de 1 hasta el número total de variables endógenas M.

$Z = (x_1,...,x_N,y_1,...,y_M)$ es decir Z engloba tanto a las variable exógenas como a las endógenas. Cada elemento de Z define un nodo y se denotará por z_i donde $i \in \{1,...N+M\}$

$Z_{p(i)}$: es el conjunto de variables de las que depende z_i. El índice p_i define al "conjunto de predecesores" de la variable z_i, es decir al conjunto de índices que define al nodo i,

4.3 CREACION DE LA GRAFICA ASOCIADA AL MODELO

El primer paso que realiza SIDEM es el de transformar las ecuaciones del modelo incluidas en los archivos de datos[2], en una gráfica.

Para crear la gráfica SIDEM lee del archivo CATALOGO.DAT el número de variables y las variables mismas, manteniendo a estas dentro de su clasificación de exógenas y endógenas. La gráfica tendrá tantos nodos como variables tenga el modelo.

Con los datos del catálogo y de la lectura del archivo RELACION.DAT se "dibujan" los arcos de la gráfica de la siguiente manera:

- Se realiza la lectura de la ecuación y a la variable que se encuentra antes del signo de igual "=" , si coincide con una variable endógena registrada, se le asigna un nodo por ejemplo v1.
- Se "dibuja" un nodo por cada una de las variables que se encuentra después del igual "=" en la ecuación, digamos v2,...vn.
- Se "dibujan" arcos dirigidos de cada nodo v2,...vn al primer nodo v1.

[2]CATALOGO.DAT Y RELACION.DAT.

- Se repite lo mismo para cada ecuación del modelo. Si alguna de las variables apareció con anterioridad, no se repite, se toma el mismo nodo definido anteriormente y se dibujan los arcos correspondientes.

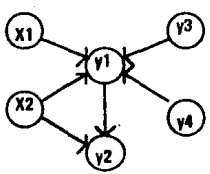
De esta manera se crea de acuerdo al modelo una gráfica dirigida. Un ejemplo de esto se muestra a continuación:

las ecuaciones

y1 = x1-x2+y3*2y4

y2 = x2+y1

las podemos representar con la siguiente gráfica:



Una vez asignada la gráfica, SIDEM utiliza esta para en base a los métodos determinados anteriormente, determinar las componentes fuertemente conexas en el sistema, esta etapa se describe en la siguiente sección.

4.4 LAS COMPONENTES FUERTEMENTE CONEXAS DEL SISTEMA

Elaborar la gráfica es un proceso detallado, pues es preciso llevar la relación de datos tales como el número de nodos e identificación de cada uno de estos, así como las adyacencias de los nodos del modelo. Esta información es importante

ya que el siguiente paso será encontrar los ciclos que nos darán las componentes fuertemente conexas de la gráfica.

El algoritmo usado es el siguiente: se toma uno de los nodos y se busca un camino en la gráfica hasta encontrar un ciclo o un nodo del cual no salgan arcos. Si se encuentra un ciclo se contraen los nodos de este en un pseudo nodo; los nodos dentro del ciclo se ignoran y los que entran y/o salen de los pseudo nodos se consideran como arcos.

La búsqueda "primero a profundidad" descrita en el capitulo III, se aplica en este algoritmo cuando se busca un camino en la gráfica. El algoritmo va visitando nodos, buscando cada vez mas profundamente, hasta encontrar o bien un ciclo que será una componente conexa o un nodo.

Una vez que se cuenta con la gráfica del modelo y se tienen identificadas las componentes fuertemente conexas de esta, se crean dos archivos de datos; GCOMP.FOR y SIDEM.DAT.

El primero contiene la subrutina GCOMP escrita en lenguaje FORTRAN, que indica al optimizador GRG2 cuales son las variables del modelo, esto se hace relacionando cada una de las variables con cada uno de los elementos de un vector $\mathcal{I}$, es decir se asigna una $\mathcal{I}(i)$, i=1,...N a cada una de las variables exógenas y una $\mathcal{I}(i)$, i=N+1,...N+P a cada una de las variables endógenas en componentes fuertemente conexas.

En vista de que SIDEM conoce el conjunto de variables exógenas, en el archivo GCOMP, se escribirán únicamente las $\mathcal{I}$ asociadas a las variables endógenas en

componentes fuertemente conexas. GRG2 mantiene esta asignación y en adelante se referirá a la variable del modelo por la $\tau(i)$ asociada.

GCOMP también le indica al optimizador cuales son las ecuaciones del modelo, esto lo hace asociando una variable G(m) a cada ecuación del modelo de la siguiente manera:

a) para aquellas variables endógenas que únicamente contienen en su relación a variables exógenas, asignará después de la escritura completa de su relación una G(m) m=1,..M, igualada únicamente a la variable endógena correspondiente.

Ejemplo: A=B+C

G(1)=A

b)para aquellas variable endógenas formadas únicamente por variables endógenas o por un conjunto de variables exógenas y endógenas, asigna una G(m) m=1,..M a la ecuación igualada a cero.

Ejemplo: A=B+C

G(1)=-A+B+C

La escritura de las variables G(m) en la subrutina GCOMP se lleva a cabo en grupos de variables, que definen cada componentes del sistema.

La escritura de los grupos se realiza bajo un orden estricto de independencia. Esto es, si el grupo o componente solo depende de variables exógenas se escribirá primero en la lista, posteriormente se incluirán los grupos de variables endógenas que dependen de las variables exógenas escritas anteriormente y así sucesivamente. De esta manera las ecuaciones en la subrutina quedan

ordenadas de tal suerte que no se puede resolver un grupo sin resolver el grupo anterior.

Cabe señalar que la subrutina GCOMP es escrita una sola vez y leída tantas veces como componentes fuertemente conexas tenga el modelo. En cada una de estas lecturas, SIDEM resuelve una componente, el control que realiza SIDEM para conocer que componente está resuelta o no, se lleva a cabo mediante la condición llamada EVALUA. Es por esta razón que la condición se introduce al final de cada componente en la subrutina GCOMP.

Adicionalmente, SIDEM con el propósito de proveer una solución factible al modelo, incorpora una función objetivo a la subrutina GCOMP. Esta función objetivo busca minimizar el cambio porcentual de las variables exógenas. Cabe señalar que si el modelo tiene solución factible dadas las variables exógenas iniciales esta función será cero. Sin embargo, si el modelo no tiene solución dadas las variables exógenas, SIDEM permitirá que estas varíen minimizándose la suma de los cambios porcentuales al cuadrado de las variables exógenas.

El segundo archivo SIDEM.DAT proporciona al optimizador los datos sobre las variables y las ecuaciones. Este archivo está formado por tres secciones; "BOUNDS", "INITIAL" y "ROWS". Cada una tiene una función especifica: "BOUNDS" indica si la variable tiene un valor fijo, si está comprendida entre un rango de valores o si la variable no tiene restricciones. En la sección "INITIAL" se encuentran los valores iniciales de las variables exógenas y finalmente en la sección de "ROWS" se indican los valores de las ecuaciones descritas en GCOMP.FOR.

A diferencia de GCOMP.FOR, SIDEM.DAT se escribe y se lee tantas veces como componentes tenga el modelo. Las especificaciones de SIDEM.DAT son las siguientes:

Si conocemos el valor de una variable porque es exógena o bien porque ya encontramos su valor, marcamos en la sección de "BOUNDS" a dicha variable con "E" y su valor encontrado. Si dicha variable queremos que sea mayor o menor a determinado valor la marcamos con "G" o "L" y el valor respectivo, si queremos que fluctúe en un rango la marcamos con "R" y los dos valores de fluctuación. Si no tiene restricción la variable la marcamos con "N". Inicialmente las variables exógenas tendrán "E" y las variables endógenas "N". Cada vez que conocemos el valor de las variables de una componente cambiamos el marcaje de la variable de "N" a "E"

En la sección de "INITIAL" únicamente damos los valores iniciales de las variables exógenas, por lo que aquí no necesitamos la relación de las componentes fuertemente conexas.

En la sección de "ROWS" la relación de las componentes también es útil, ya que iguala las ecuaciones de las variables en determinada componente a 0 cuando intenta resolver ésta e iguala la variable a su valor después de haber resulto la componente.

Ejemplos de ambos archivos GCOMP.FOR y SIDEM.DAT se muestran en el capítulo VI.

4.5 INTERACCION ENTRE GRG2 Y SIDEM

Cualquier solución obtenida se logra resolviendo iterativamente el sistema de ecuaciones, componente por componente. Esto es, una vez creado GCOMP.FOR, SIDEM crea en la primera iteración SIDEM.DAT, en esta iteración GRG2 resuelve para el primer grupo de variables que usualmente comprenden variables endógenas que dependen de solamente de variables exógenas y resuelve además el siguiente grupo, es decir variables endógenas que dependen de variable endógenas incluidas en el primer grupo. Posteriormente SIDEM reescribe SIDEM.DAT con los valores obtenidos por GRG2 para las variables resueltas previamente. SIDEM y GRG2 iteran de esta manera hasta encontrar solución a todas las componentes.

Una vez hecho esto SIDEM está listo para mostrar al usuario los resultados obtenidos por medio de los cuadros que el usuario proporcionó. Antes de esto, en la siguiente sección se describen las características del optimizador GRG2.

4.6 EL OPTIMIZADOR

El optimizador llamado GRG2, es un programa en lenguaje FORTRAN de programación no-lineal con restricciones, basado en una versión del algoritmo del gradiente reducido generalizado (Generalized Reduced Gradient). GRG2 intenta resolver un problema de la forma $y_i = f_i(x,y)$ con i=1,..,M por el método del gradiente reducido.

El usuario deberá proporcionar la subrutina GCOMP, así como también proporcionar por medio de un archivo de datos, SIDEM.DAT, los límites superiores e inferiores de las variables, y los valores iniciales de las mismas.

GRG2 usa la primera derivada parcial de cada función f_i con respecto a cada variable x_i con i=1,..,M. Estas se calculan automáticamente por "aproximación en diferencias finitas".

Después del segmento de entrada inicial, el programa opera en dos fases. Si el punto inicial dado por el usuario no satisface todas ecuaciones se inicia la fase I. La función objetivo de la fase I es la suma de las restricciones violadas, mas una fracción de la función objetivo verdadera. La optimización termina, ya sea con un mensaje de que el problema no tiene solución factible o con una solución factible.

La fase II comienza con una solución factible, ya sea encontrada por la fase I o proporcionada por un punto inicial dada por el usuario y trata de optimizar la función objetivo. Al finalizar la fase II un ciclo completo de optimización ha sido concluido y los resultados son proporcionados.

4.7 ANALISIS DE ESCENARIOS

Una herramienta interesante del sistema se presenta en esta sección, ya que aquí se modelan y analizan los diferentes escenarios del usuario.

Para esto y debido al manejo que se les da a las variables exógenas y endógenas en SIDEM, podemos definir 3 conjuntos en donde se clasifican a estas de acuerdo a la función que en determinado momento ejecutan.

Definimos a las variables como metas, si es que se solicita que el valor de esta variable alcance un valor determinado, denotaremos al conjunto de dichas variables como Z_T . Definimos a las variables como libres si es que decidimos dejar la variable fluctuar tanto como sea necesario, el conjunto de las variables

libres lo denotamos por Z_L. Finalmente se define al resto de las variables como fijas, teniendo estas variables su valor predeterminado, el conjunto de estas variables lo denotaremos por Z_F.

Una vez clasificadas las variables en los conjuntos Z_T, Z_L y Z_F, se define un escenario inicial Z^0 asignando valores a las variables de Z_T y Z_F para resolver Z_F..

SIDEM calcula entonces una solución para el sistema de ecuaciones:

$$Z_i = f_i\,(Z_{p(i)}) \qquad i\varepsilon\,\{1,\ldots N+M\}$$

5) $$Z_j = Z^0{}_j \qquad \text{para } Z_j \in\{Z_T \cup Z_F\}$$

SIDEM determina que valores deben adquirir las variables en Z_L para que el sistema (5) se satisfaga y la asociación $Z^0{}_T$ sea posible.

Mas de una solución es posible ya que los conjuntos Z_T y Z_L son arbitrarios. SIDEM permite al usuario construir una serie de escenarios en los cuales los conjuntos de variables Z_T y Z_L permanecen inalterados pero los valores de las variables metas van cambiando. Esto es de gran utilidad para estudiar el comportamiento de algunas variables respecto a otras. [3].

En una última etapa el usuario puede regresar una y otra vez al escenario planteado originalmente o cambiar de conjuntos Z_T y Z_L para así poder crear escenarios completamente nuevos.

CAPITULO V

DESCRIPCION DEL PROGRAMA

5.1 ESTRUCTURA DE LA PROGRAMACIÓN

Como vimos en el capítulo anterior, tenemos que en general SIDEM opera en el modo de consulta y en el modo de simulación.

En el modo de consulta el usuario puede precisamente consultar las tablas con los valores que cada variable obtuvo en el escenario inicial o bien en el último escenario obtenido.

El modo de simulación permite al usuario generar escenarios alternativos a partir del escenario inicial. Para esto se le pide al usuario proporcione la variable o variables del modelo con los valores meta que se requiere y la variable o variables que se quieren dejar como "variables libres" para así alcanzar las metas especificadas.

Una de las características de SIDEM, es que traduce la información proporcionada por el usuario al formato de GRG2 y de esta manera evita al usuario la escritura manual de los archivos de datos, cada vez que se ejecuta el optimizador con modificación en los datos del modelo.

5.2 FUNCIONAMIENTO

Debido a las necesidades, facilidades y limitaciones que se tuvieron para utilizar el paquete GRG2, fue necesario dividir la programación del sistema en dos etapas o partes principales:

La primera etapa llamada GENERA, esta programada en lenguaje PASCAL y se encarga de realizar el análisis de los datos del problema. Aquí es en donde se crea la gráfica y se reconocen las componentes fuertemente conexas.

Con estos datos, crea o genera el archivo GCOMP.FOR, este contiene a la subrutina GCOMP en lenguaje FORTRAN, necesaria para informar a GRG2 de los datos específicos del problema.

La segunda etapa llamada SIDEM, está programada en lenguaje FORTRAN y se encarga de la elaboración del archivo SIDEM.DAT, el cual contiene valores específicos de inicio y rangos de operación para las variables. Una vez creado el archivo hace el llamado al optimizador GRG2, traduce y despliega los resultados obtenidos, para después permitir al usuario analizar diferentes escenarios por medio de la simulación.

5.3 DATOS DE ENTRADA.

El formato de los datos de entrada a SIDEM, es muy sencilla ya que el usuario debe de proporcionar únicamente 3 archivos de datos, los cuales se describen a continuación:

1) CATALOGO.DAT.- Archivo en el que se proporciona el número de variables exógenas, número de variables endógenas, nombre, descripción y valor inicial de las variables exógenas así como nombre y descripción de las variables endógenas. Los nombres de las variables deben tener a lo mas 6 caracteres, y deben de cumplir con las características de variables en lenguaje FORTRAN, en donde el primer caracter debe de ser una letra.

2) RELACIONES.DAT.- Archivo que contiene las funciones que definen al sistema. Por cada relación se escribe primero el nombre de la variable endógena, el caracter "=" y después la función correspondiente, teniendo cuidado de usar símbolos matemáticos permitidos en lenguaje FORTRAN. Se puede consultar cualquier libro de FORTRAN para consultar mayor detalle.

3) CUADROS.DAT.- Contiene información sobre las variables que deben aparecer en los cuadros que forman los escenarios de consulta del sistema. Este archivo especifica el número de cuadros que se quieren desplegar, así como el título y las variables que se van a incluir en cada uno de ellos.

5.4 DESCRIPCION DE LA PROGRAMACION

En esta sección se describirán los procesos y funciones programados en cada una de las etapas. El código completo se encuentra en el apéndice B de esta tesis.

5.4.1 PRIMERA PARTE: PROGRAMACION EN PASCAL.

Como ya se mencionó, aquí se analizan los datos y se elabora el archivo GCOMP.FOR. Para tal efecto se diseñó el sistema GENERA, el cual tiene la siguiente estructura:

ESTRUCTURA DEL PROGRAMA EN PASCAL

ULTIMO

UNIDAD VARGLOBA

UNIDAD LEE-RELACION

UNIDAD CICLOS

PROC. CFC1

PROC. QUITA_NODO

PROC. BLOQUES

UNIDAD GCOMP.

PROC. LETRA I_N

PROC. CUENTA

PROC. CUENTACOM

PROC. CUENTASOLA

PROC. VARSOLA

PROC. SINIGUAL

PROC. VARENCOM

UNIDAD PROCEMOD

PROC. CATALOGO

PROC. LECTCATAL

FUNC. ELIMINA_BLANCOS

FUNC. LEE_VAR

PROC. BUSCA_LETRA

FUNC. ENCONTRO

FUNC. MAYUSCULA

FUNC. BLANCOS

Ahora describiremos el funcionamiento de cada una de las unidades de GENERA.

UNIDAD 1. VARGLOBAL. Contiene a todas las variables del programa usadas en forma común tales como arreglos de variables y arreglos de valores. Esta unidad está en interface con el resto de las unidades.

UNIDAD 2. LEE_RELACIONES. En esta unidad se ejecuta la lectura de los archivos de entrada para identificar las variables y sus características así como las relaciones.

Se hace el llamado a la subrutina CATALOGO, la cual ejecuta la lectura del archivo CATALOGO.DAT, identificando nombre descripción y valor de las variables exógenas, y nombre y descripción de las variables endógenas; aquí se revisa que sea una variable permitida en lenguaje FORTRAN.

Ya identificadas las variables del problema, se ejecuta la lectura del archivo RELACION.DAT. Cada una de estas relaciones es identificada cuidando que la variable leída antes del igual, sea una variable endógena descrita en el catálogo.

Se revisa la correcta escritura de la relación y se coloca a cada una de estas en un campo del arreglo asignado. En cada una de las relaciones identifica las variables que involucra, de manera que elabora con estas y las relaciones funcionales, una gráfica en donde los nodos son las variables y los arcos son las relaciones.

Cada vez que lee una relación mas, se van uniendo las variables a la gráfica de la misma manera, teniendo cuidado de no repetir las variables en la gráfica y llevando el conteo de las adyacencias de cada una de las variables encontradas.

Toda la información es guardada en arreglos y apuntadores que se usarán mas adelante para encontrar las componentes del problema.

Algunas de las funciones y procedimientos utilizadas en esta unidad, se encuentran en la unidad PROCEMOD.

UNIDAD 3 . CICLOS. En esta unidad se usa el procedimiento CFC (componentes fuertemente conexas), tomado del trabajo de tesis de Rogelio Garduño. [4]. Este procedimiento toma como datos de entrada, los arreglos obtenidos en la unidad LEE-RELACIONES, correspondientes a la gráfica del problema, en estos arreglos encuentra el número de adyacencias de cada nodo, cuales son éstas y un apuntador que nos indica en donde empieza el siguiente nodo. Como salida del procedimiento CFC obtenemos el número de componentes fuertemente conexas, el número total de nodos involucrados en las componentes, los nodos contenidos en estas componentes y un apuntador que separa los nodos de cada componente.

El procedimiento CFC trabaja por medio de pilas y apuntadores, agregando y quitando nodos de tal manera que realiza una búsqueda primero a profundidad sobre la gráfica, asegurándose de recorrer y ordenar todos los nodos involucrados.

UNIDAD 4.- GCOM. Esta unidad escribe los archivos GCOMP.FOR y VACFC.DAT con los datos del problema.

La subrutina GCOMP contenida en el archivo GCOMP.FOR, debe de cumplir con cierto formato cumpliendo además los requerimientos de escritura de una subrutina en lenguaje FORTRAN.

Se mandan a impresión las instrucciones iniciales en FORTRAN necesarias para declarar nombre y tamaño de las variables, ajustadas a los datos del problema, así como instrucciones específicas para GRG2.

La forma en que se ubican los datos del problema en el archivo y como se construye esta parte de la subrutina GCOMP, se describió en el capítulo anterior.

Para completar la subrutina, el sistema escribe en el archivo, las instrucciones generales necesarias en lenguaje FORTRAN para finalizar la subrutina.

La escritura del archivo VACFC.DAT se realiza al final de la unidad y tiene el siguiente orden: el número total de componentes fuertemente conexas, el número de total de nodos involucrados en las componentes conexas, el arreglo del apuntador para separar cada componente conexa y finalmente el arreglo que contiene las variables endógenas que están en las componentes fuertemente conexas.

Algunas de las funciones y procedimientos usadas en esta unidad se encuentran en la unidad PROCEMOD.

UNIDAD 5. PROCEMOD. En esta unidad se encuentran los procedimientos y funciones que se usan en las unidades LEE-RELA y GCOMP. Los mencionamos únicamente ya que su descripción se hizo en las unidades anteriores.

Procedure CATALOGO.
Procedure LECTCATALOGO.
Función ELIMINABLANCOS.
Function BLANCOS.
Function LEE_VAR.
Procedure BUSCA_LETRA.
Función ENCONTRO.
Función MAYUSCULA.

5.4.2 SEGUNDA PARTE: PROGRAMACION EN FORTRAN.

En la primera parte obtuvimos el archivo GCOMP.FOR en lenguaje FORTRAN, el cual contiene la subrutina GCOMP. En esta segunda parte, se realiza la escritura del archivo de datos SIDEM.DAT. Con estos dos archivos conteniendo los datos específicos del problema se hace el llamado a GRG2 y se obtienen resultados. La programación en fortran tiene la siguiente estructura:

ESTRUCTURA DE LA PROGRAMACION EN FORTRAN

PROGRAMA PRINCIPAL
- SUBR. LECACU
- SUBR. LEVATA
  - SUBR. LEVACF
  - SUBR. RANGO
  - SUBR. ENTRADA
- LLAMADO A GRG
- SUBR. ESCENA
  - SUBR.IMPRESION
- SUBR. LEEMETAS
- SUBR. LEE_VARIABLES
  - SUBR. ESCRIBE_Y_LEE
    - SUBR. IMPRIMIR
  - SUBR. RESTO

RUTINAS
- SUBR. BUSCAVAR
- SUBR. BANDERA

Ahora describiremos el funcionamiento de las subrutinas de SIDEM. El detalle se puede consultar en el código del apéndice A.

Subrutina LECACU.- Ejecuta la lectura del archivo CATALOGO.DAT. Aquí se registra el número de variables exógenas y endógenas, así como el nombre de la variable exógena, descripción y valor, y nombre de la variable endógena y descripción.

Subrutina LEVATA.- Ejecuta la lectura del archivo CUADROS.DAT, aquí se registra el número de cuadros y el título de cada uno de ellos. Realiza la lectura de las variables que se incluyen en cada cuadro, las busca en el catálogo e identifica si estas son variables exógenas o endógenas.

Registra y almacena a las variables en una matriz llamada VEC (variables en cuadros), la primera entrada de la matriz corresponde al número de cuadro, la segunda al número de variable en dicho cuadro.

Si alguna de las variables registradas en el cuadro no se encuentra en el catálogo, se despliega un letrero en pantalla notificando al usuario de esto.

Subrutina LEVACF.- Ejecuta la lectura del archivo VACFC.DAT, el cual creamos en GENERA y tiene la información de las variables en componentes fuertemente conexas.

Con esta información inicializa los conjuntos, fijos, metas, ceros, mayor y menor

de la siguiente manera:

Fijos (i) = 1 i= número de variable exógena
Metas (i) = 0 i= número de variable endógena
Ceros (i) = 0 i= número de variable endógena en cfc
Menor (i) = 0 i= número de variable endógena en cfc
Mayor (i) = 0 i= número de variable endógena en cfc

Subrutina RANGO.- Debido a los límites que los valores de las variables del problema pueden tener, es necesario para aquellas variables endógena en componentes con restricciones, especificar el rango en que se debe de operar para evitar que el sistema no tenga solución.

Con tal fin se le pide al usuario proporcione el rango de valores permitidos para estas variables, es decir un valor mínimo y un valor máximo. Este campo no es un requisito para todas las variables.

Subrutina ENTRADA.- Esta subrutina crea e imprime el archivo SIDEM.DAT que contiene los datos de las variables en un formato específico de GRG2, el cual se indica a continuación:

Formato de SIDEM.DAT.

BOUNS
:
E num. variable VALOR DE LA VARIABLE o
N num. variable o
G num. variable VALOR DE LA VARIABLE o

```
 L  num. variable VALOR DE LA VARIABLE o
 :
END
INITIAL
SEPARATE
  :
  num. variable  VALOR DE LA VARIABLE
  :
END
ROWS
 :
 E  num. variable VALOR DE LA VARIABLE
 E  num. variable  0.0
 N  num. variable
END
PRINT
IPR  -1
END
GO
STOP
```

En BOUNDS se indica que valor tomará dicha variable; se precede el numero de la variable con E cuando igualamos a la variable a un valor específico, con N cuando dejamos que el valor de la variable sea libre, R si el valor se restringe a un rango, G si el valor de la variable es mayor a un número y L si es menor.

En INITIAL se indica el valor inicial que tendrán las variables. A todas las variables que no se inicializan explícitamente, se les asignará por default un valor de 0.0.

En ROWS indicamos los valores de las ecuaciones. Precedemos al número de la ecuación (o variable endógena) con E cuando queremos igualar dicha ecuación a un valor específico, incluyendo 0.0 si fuera necesario y usamos N cuando la ecuación no tiene restricción.

Finalmente se dan los comandos en lenguaje FORTRAN para dar la orden de imprimir resultados y regresar al sistema.

El formato específico de este archivo de datos, está en el manual de operación de GRG2.

Una vez elaborados los archivos necesarios se hace el llamado a GRG2, quien toma los datos, resuelve el modelo y da resultados iniciales, que se pueden consultar en los cuadros proporcionados por el mismo usuario.

Aquí es en donde empieza la interacción con el usuario. Si es la primera vez que corre GRG2, sigue con subrutina ESCENA, si es la segunda vez o mas, se va a la subrutina ESCRIBE Y LEE.

Subrutina ESCENA.- Esta subrutina permite al usuario consultar los diferentes cuadros que se especificaron en el archivo CUADROS.DAT, dando los valores que se obtuvieron en la corrida para cada variable.

Dentro de esta subrutina está la opción a impresión en papel de los cuadros consultados.

Si es la primera vez que pasa por GRG2, sigue a subrutina LEEMETAS, si no, sigue a subrutina RESTO.

Subrutina LEEMETAS.- En esta subrutina se calculan los nuevos escenarios proporcionados por el usuario. Se le pide al usuario proporcione las variables meta así como el valor a alcanzar en estas, identifica a las variables y si es válida, es decir, si se identificó a esta en el catálogo, la guarda en el arreglo VARCOMA.

Si encontró una variable exógena, el conjunto de Fijos cambiará en el lugar de la variable de 0 a 1 y los valores iniciales cambiarán por aquellos valores nuevos que sea necesario.

Si encontró a una variable endógena, el conjunto de metas en el lugar de la variable cambiará de 0 a 1 y también los valores cambiarán por los nuevos valores solicitados.

Subrutina LEEVARIABLES.- En esta subrutina se le pide al usuario que indique aquellas variables que se dejarán libres, es decir, aquellas que puedan tomar un valor distinto al inicial.

Si la variable que se pide es exógena, se modifica el conjunto de Fijos, en el lugar de la variable cambiará de 1 a 0, es decir, dejamos que el valor varíe tanto como sea necesario.

En cambio si la variable es endógena, en el conjunto de Metas en el lugar de la variable, cambia de 1 a 0 para que su valor pueda variar tanto como sea necesario.

Con la nueva información regresa a la subrutina ENTRADA para escribir el archivo SIDEM.DAT con las modificaciones y volver a entrar a GRG2, calculando así nuevos resultados. Cuando ya pasó mas de una vez por GRG2 se va a la subrutina ESCRIBE Y LEE.

Subrutina ESCRIBE Y LEE .- Esta subrutina despliega los nuevos resultados en forma de tabla con el siguiente formato: debajo de las variables solicitadas en metas, y de las variables solicitadas como variables libres, se despliega en el primer renglón su valor inicial y en los renglones siguientes los valores obtenidos en las nuevas corridas.

Si se requieren nuevas metas, se dan los valores para estas debajo de cada variable que aquí se haya declarado, hace la lectura y se procesa la información de la misma manera que la primera vez crea un nuevo archivo de datos, llama a GRG y da nuevos resultados.

Esta subrutina también tiene la opción a impresión de las tablas creadas.

Subrutina RESTO.- Esta permite al usuario regresar a alguna de las subrutinas anteriores. Si se requieren nuevos escenarios se direcciona a ESCRIBE Y LEE; Si quiere cambiar variables metas y variables libres, se direcciona a LEEMETAS; Si no se requieren nuevas metas y ya no se quiere consultar ni modificar ningún escenario, se da por terminado el sistema.

CAPITULO VI

APLICACION DEL SISTEMA

6.1 EL MODELO

En este capítulo resolveremos un modelo con el sistema SIDEM. Empezaremos por dar los datos del problema.

El objetivo es resolver un problema en donde se involucran 4 variables a través del tiempo, supongamos 5 periodos. La variable S_t representa el saldo en el período t, el modelo proporciona una S_0 inicial de donde partiremos para calcular las S_t sucesivas. La variable P_t es el pago a determinar en el periodo t de acuerdo al saldo anterior, a una cierta tasa y a un factor K_t, esta última variable también se determina para cada periodo y la obtenemos de los saldos S_t y S_{t-1}.

La idea es determinar los saldos, pagos y factores en base a un saldo inicial S_0 y a t tasas de interés R_t, las cuales también se darán inicialmente. Por la forma en que se estructuró el modelo tenemos que las variables exógenas serán S_0 y R_t con t=1,...,5. Las variables endógenas serán las S_t restantes, las P_t y las K_t con t=1,...,5.

Los datos del problema los podemos ver en los archivos que le proporcionamos a SIDEM, es decir catalogo.dat, relaciones.dat y cuadro.dat. A continuación se muestran cada uno de los archivos de datos.

CATALOGO.DAT

6 15

S0	100
R1	0.10
R2	0.10
R3	0.10
R4	0.10
R5	0.10
R6	
S1	
P1	
K1	
S2	
P2	
K2	
S3	
P3	
K3	
S4	
P4	
K4	
S5	
P5	
K5	
S6	
P6	
K6	

RELACION .DAT

S1 = S0*(1+R1)-P1
P1 = S0*(1+R1)/K1
K1 = 6*(S0-S1)/S0
S2 = S1*(1+R2)-P2
P2 = S1*(1+R2)/K2
K2 = 6*(S1-S2)/S1
S3 = S2*(1+R3)-P3
P3 = S2*(1+R3)/K3
K3 = 6*(S2-S3)/S2
S4 = S3*(1+R4)-P4
P4 = S3*(1+R4)/K4
K4 = 6*(S3-S4)/S3
S5 = S4*(1+R5)-P5
P5 = S4*(1+R5)/K5
K5 = 6*(S4-S5)/S4

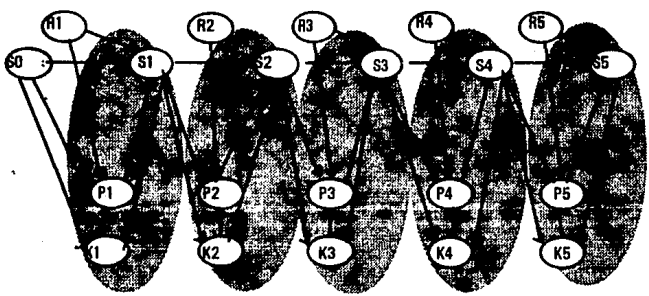
CUADRO.DAT

3
PERIODOS 1 Y 2
S1, P1, K1, S2, P2, K2
PERIODOS 3 Y 4
S3, P3, K3, S4, P4, K4
PERIODO 5
S5, P5, K5

De esta manera los datos son introducidos al sistema para procesarlos, crear su gráfica, encontrar los ciclos y escribir el archivo GCOMP.FOR, como se describió en los capítulos anteriores.

6.2 LA GRAFICA

Podemos dibujar la gráfica del modelo de la siguiente manera:



Las componentes fuertemente conexas se muestran en la zona sombreada.

6.3 ARCHIVOS PARA SIDEM

El sistema trabaja con los datos de tal manera que hace posible la escritura de los archivos de datos GCOMP.FOR y SIDEM.DAT. Los archivos obtenidos para este ejemplo por el sistema se muestran a continuación. Nótese que en GCOMP.FOR se señalan cada una de las componentes fuertemente conexas; la función EVALUA también se incluye para detener el proceso al final de cada componente y hacer correr el sistema tantas veces como componentes tenga.

En SIDEM.DAT se muestra el último de los archivos generados, ya que este se modifica cada vez que se hace una nueva corrida del optimizador, agregando y modificando datos que hacen posible la solución componente por componente.

La figura 1 muestra el archivo GCOMP.FOR y la figura 2 el archivo SIDEM.DAT.

6.4 SOLUCIONES Y ESCENARIOS

La solución inicial se muestra en los cuadros 1, 2 y 3, esta impresión es dada por el sistema.

```
  PERIODO 1 Y 2
S1                                  61.8916
P1                                  48.1084
K1                                   2.2865
S2                                  38.3057
P2                                  29.7750
K2                                   2.2865
  PERIODO 3 Y 4
S3                                  23.7080
P3                                  18.4283
K3                                   2.2865
S4                                  14.6732
P4                                  11.4056
K4                                   2.2865
  PERIODO 5
S5                                   9.0815
P5                                   7.0590
K5                                   2.2865
```

FIGURA 1

```
      SUBROUTINE GCOMP(G,X)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION K1,K2,K3,K4,K5
      DIMENSION G(16), X(21)
      INTEGER EVALUA
      COMMON/EVAL/EVALUA(256)
      S1 = X(7)
      K1 = X(8)
      P1 = X(9)
      S2 = X(10)
      K2 = X(11)
      P2 = X(12)
      S3 = X(13)
      K3 = X(14)
      P3 = X(15)
      S4 = X(16)
      K4 = X(17)
      P4 = X(18)
      S5 = X(19)
      K5 = X(20)
      P5 = X(21)
C      variables en  componente conexa 1
      G(3)= -S1+X(1)*(1+X(2))-P1
      G(2)= -K1+6*(X(1)-S1)/X(1)
      G(1)= -P1+X(1)*(1+X(2))/K1
C      fin de la componente conexa 1
      IF (EVALUA(1).EQ.1) THEN
C      variables en  componente conexa 2
      G(6)= -S2+S1*(1+X(3))-P2
      G(5)= -K2+6*(S1-S2)/S1
      G(4)= -P2+S1*(1+X(3))/K2
C      fin de la componente conexa 2
      IF (EVALUA(2).EQ.1) THEN
C      variables en  componente conexa 3
      G(9)= -S3+S2*(1+X(4))-P3
      G(8)= -K3+6*(S2-S3)/S2
      G(7)= -P3+S2*(1+X(4))/K3
C      fin de la componente conexa 3
      IF (EVALUA(3).EQ.1) THEN
C      variables en  componente conexa 4
      G(12)= -S4+S3*(1+X(5))-P4
      G(11)= -K4+6*(S3-S4)/S3
      G(10)= -P4+S3*(1+X(5))/K4
C      fin de la componente conexa 4
      IF (EVALUA(4).EQ.1) THEN
C      variables en  componente conexa 5
      G(15)= -S5+S4*(1+X(6))-P5
      G(14)= -K5+6*(S4-S5)/S4
      G(13)= -P5+S4*(1+X(6))/K5
C      fin de la componente conexa 5
      ENDIF
      ENDIF
      ENDIF
      ENDIF
C      funcion objetivo
      G(16) = ((X(1)- 1.0000000000E+02)/ 1.0000000000E+02)**2
     1 + ((X(2) - 1.0000000000E-01)/ 1.0000000000E-01)**2
     1 + ((X(3) - 1.0000000000E-01)/ 1.0000000000E-01)**2
     1 + ((X(4) - 1.0000000000E-01)/ 1.0000000000E-01)**2
     1 + ((X(5) - 1.0000000000E-01)/ 1.0000000000E-01)**2
     1 + ((X(6) - 1.0000000000E-01)/ 1.0000000000E-01)**2
      RETURN
      END
```

FIGURA 2

```
   21   16
BOUNDS
 E    1                100.0000
 E    2                  0.1000
 E    3                  0.1000
 E    4                  0.1000
 E    5                  0.1000
 E    6                  0.1000
 E    7                 61.8916
 E    8                  2.2865
 E    9                 48.1084
 E   10                 38.3057
 E   11                  2.2865
 E   12                 29.7750
 E   13                 23.7080
 E   14                  2.2865
 E   15                 18.4283
 E   16                 14.6732
 E   17                  2.2865
 E   18                 11.4056
 G   19                  0.0010
 G   20                  0.0010
 N   21
END
INITIAL
SEPARATE
      1                 100.0000
      2                   0.1000
      3                   0.1000
      4                   0.1000
      5                   0.1000
      6                   0.1000
END
ROWS
 E    1                 61.8916
 E    2                  2.2865
 E    3                 48.1084
 E    4                 38.3057
 E    5                  2.2865
 E    6                 29.7750
 E    7                 23.7080
 E    8                  2.2865
 E    9                 18.4283
 E   10                 14.6732
 E   11                  2.2865
 E   12                 11.4056
 E   13      0.0
 E   14      0.0
 E   15      0.0
 N   16
END
PRINT
IPR     -1
END
GO
STOP
```

Ahora jugaremos un poco con los escenarios, es decir supongamos que queremos fijar una de nuestras variables exógenas a determinado valor. Digamos que queremos hacer el pago P2=30.0 , dejando que el saldo S2 varíe tanto como sea necesario. Después de la misma manera P2=32.0 y P2=34.0. Las resultados de las diferentes propuestas se muestran a continuación en las tablas dadas por el sistema.

Entra a la

SECCION DE CALCULO DE ESCENARIOS NUEVOS

PROPORCIONE LAS NUEVAS METAS EN FORMATO:

NOMBRE DE LA VARIABLE = VALOR, ... ,

EJ. CAP=1000, VAR=50

P2=32.0

PROPORCIONE LAS VARIABLES QUE QUIERE DEJAR LIBRES

CON FORMATO: NOMBRE, NOMBRE

EJ. VAR1, VAR5

La tabla desplegada después de proporcionar los diferentes valores de P2, uno por uno son:

	P2	S2
1	29.7750	38.3057
2	30.0000	8.0809
3	32.0000	4.0809
4	34.0000	0.0808

Una sección mas se despliega al usuario, esta es:

DESEA RECUPERAR SU TABLA DE ESCENARIOS (S/N)?

DESEA RECUPERAR SUS VARIABLES Y METAS (S/N)?

Esta nos permite regresar a la tabla con la que estábamos trabajando y seguir calculando nuevos escenarios o crear una nueva tabla con una combinación de variable diferentes.

Después de cada una de las secciones se le da al usuario la opción de impresión del cuadro.

Cuando se le da fin al sistema un letrero de "FIN DE SIDEM" se despliega en la pantalla.

CONCLUSIONES

Cuando nos encontramos ante un problema al cual queremos darle una solución, la manera de atacarlo es modelarlo mediante un sistema de ecuaciones, mismo que tendremos que plantear y resolver.

En este trabajo se implementa un sistema el cual expone una forma específica de organizar un conjunto de ecuaciones, combinando la teoría de gráficas para el planteamiento del problema con el optimizador GRG2 para la solución del mismo.

Esta combinación resultó además de interesante, por lo que la teoría de gráficas implica, eficiente por la simplificación del arduo trabajo que representa el solucionar un sistema de ecuaciones de esta naturaleza; especialmente si trabajamos con sistemas grandes en número de variables y restricciones.

El resolver un sistema como lo hace SIDEM, es decir componente por componente, es equivalente a desglosar el sistema de ecuaciones en "pequeños sistemas" para que GRG2 resuelva cada una de ellos en forma individual. De tal manera que cuando resuelve uno de estos, SIDEM proporciona resultados para las variables involucradas en éste, y los sustituye en el resto del sistema como puntos fijos. Las variables de este pequeño sistema resuelto, simulan ahora variables exógenas en el sistema general.

Los problemas con los que nos enfrentamos en la programación del sistema, fueron principalmente dos:

- El primero y mas importante fue saber en que orden se agruparon las variables originales del problema y sobre todo seguir el comportamiento de este orden y la

relación que guardaban entre sí. Esto representó trabajar con el menor número de apuntadores y variables, para evitar el desorden y confusión en dicho problema o modelo.

- El segundo problema fue después de haber identificado a las componentes del modelo, el saber como resolverlo una por una. Para esto fue necesario estudiar GRG2 y entender el funcionamiento de éste e implementar una función EVALUA a la subrutina GCOMP, lo cual nos ayuda a correr el sistema componente por componente para que así cuando queramos resolver una parte del sistema le demos el tratamiento necesario a los archivos creados.

El optimizador utilizado GRG2, es un paquete que se utiliza desde hace tiempo con buenos resultados y un alto grado de confiabilidad. Implementar SIDEM con el optimizador GRG2 nos resultó eficiente, después de su instalación y compilación con una versión F77L de FORTRAN.

La forma de hacer correr el optimizador para un problema específico sin SIDEM, es crear "a mano", los archivos de datos GCOMP.FOR y SIDEM.DAT y hacer el llamado a GRG2. Tomando en cuenta la manera en que estamos resolviendo el problema, es decir componente por componente, se tendría que crear un archivo de datos diferente cada vez que se resuelve el sistema formado por una de estas, con esto SIDEM resulta bastante efectivo considerando que es mínimo el tiempo que se necesita el proporcionar los datos iniciales del modelo

A P E N D I C E A

```
PROGRAM GENERA;
    { Programa principal: ejecuta la lectura de los archivos
      de catalogo.dat y relaciones.dat. Con los datos obtenidos
      hace una grafica, en donde las variables son los nodos y
      y las operaciones en las relaciones son los arcos. Aquí se
      encuentran las componentes fuertemente conexas del problema,
      obteniendo el archivo de salida "GCOMP.FOR", el cual se
      utiliza en 'GRG2'. }

uses ciclos,gcom,varglobal,lee_relaciones,crt ;
var iw:integer;


BEGIN
    clrscr;
    writeln;
    lee_relacion;
    CFC(icfc,ICN,LICN,IP,LENR,IOR,IB,NUM,IW);
    gcom1;
    writeln;
    writeln('                HE TERMINADO        ');
END.
```

```
UNIT LEE_RELACIONES;
    { Esta unidad es utilizada por el programa 'GENERA';
      ejecuta la lectura del catalogo y de las relaciones,
      revisa que este correctamente escrito y crea los
      arreglos 'lenr', 'ip','icn' y la variable 'licn'.
      Utiliza la unidad PROCEMOD                          }

INTERFACE
uses  crt,varglobal,procemod;
procedure lee_relacion;

IMPLEMENTATION
var
  salida, archrel    : text;
  varia, nombre,
  relacion           : string[72];
  aux                : string;
  jjj, kkk           : byte;
  k, i2, i, j,
  nod, ayu, lrenodo  : integer;
  nuevo              : boolean;
  conj               : set of byte;
  sig                : char;
  apreno,renodo      :arr1;


PROCEDURE LEE_RELACION;

var
    endogena      : string[6];
    jjj, kkk, k   : integer;
    relanum, car  : string;
    new           : boolean;

begin
   naisladas := 0;
   catalogo(numexo,numendo,variable,apuntador);     {procedimiento catalogo en pr
   {inicializa arreglos y variables}
   for i:=1 to numendo do
   begin
      lenr[i]:=0;
      ip[i]:=1;
      ayuda[i] := 0;
   end;
   licn := 0;
   i2 := 1;
   icfc := 0;
   concfc := [];
   lrenodo := 0;
   ayu := 0;
   nombre:=directorio+'relacion.dat';
   assign(archrel,nombre);
   reset(archrel);
    {Empieza la lectura de cada una de las relaciones, revizando
     que cada una de las variables leidas se encuentre en el catalogo}
   for i:=1 to numendo do
   begin
     readln(archrel,aux);
     aux:=blancos(aux);
     relacion:=aux;
```

```
relaciones[i]:=relacion;
relanum := '';
j := 1;
endogena:=lee_var(j,relacion,relanum,sig);     {busca la primera variable de
endogena := mayuscula(endogena);
kkk := buscavar(endogena);
if kkk = -1 then
begin
   writeln('La variable ',endogena,' no esta catalogada');
   writeln('pero aparece en la relacion ',relacion);
   halt;
end
else
if kkk < apendo then
begin
   writeln('La variable ',endogena,'escrita en la relacion',relacion;;
   writeln('está catalogada como variable exogena');
   halt;
end;
if (endogena = '') or (sig <> '=') then
begin
   writeln('La relacion ',relacion,' esta mal escrita ');
   halt;
end;

{Hasta aquí ha revisado la primera parte de la igualdad de la
 relación, si está correctamente escrita, continua revisando
 el resto de la expresión}

new := true;
while (j < length(relacion))  do                {toma a cada una de las}
begin                                           {variables de la relacion}
   varia:= lee_var(j,relacion,relanum,sig);
   varia := mayuscula(varia);
   if (varia <> '') and (sig <> '(') then
   begin
      jjj := buscavar(varia);
      if jjj = -1 then
         begin
            writeln('La variable ',varia,' no está catalogada');
            writeln('pero aparece en la relacion ',relacion);
            halt;
         end;

      {Pone en el arreglo RENODO, a las todas las variables que se
       encuentran en las relaciones de las endogenas; LRENODO nos
       dara la longitud del arreglo RENODO. El arreglo de apuntadores
       APRENO, nos indica el lugar en donde empieza la iésima relacion.
       El arreglo AYUDA nos dice el numero de aquellas variables
       endogenas que contienen en su relacion variables endógenas.
       La variable NEW nos dice si ya pusimos a la variable en el
       arreglo.}

      inc(lrenodo);
      renodo[lrenodo] := jjj;
      if new then
      begin
         apreno[i] := lrenodo;
         new := false;
      end;
```

```
            if jjj >= apendo then
            begin
               if not (kkk in concfc) then
               begin
                  concfc := concfc + [kkk];
                  inc (ayu);
                  ayuda [ayu] := kkk;
               end;
            end;
           {Continua construyendo las relaciones con numeros}

           STR(jjj,car);
           if jjj < apendo then
              relanum := relanum + 'X(' + car + ')'
           else
              relanum := relanum + varia;
        end;
     end;
     arrelnum[kkk] := relanum;
     writeln(kkk,'-->',arrelnum[kkk]);}
   end;   {for i = ..... }
   apreno [numendo + 1] := lrenodo + 1;

   {Recorre nuevamente las relaciones de las variables endogenas para
    crear los arreglos ICN, IP, LENR solamente para aquellas variables
    endógenas que contengan en sus relaciones variables endógenas}

   for i := 1 to numendo do
   begin
        nuevo := true;
        conj := [];
        for j := apreno[i] to (apreno[i+1]-1) do
        begin
           jjj := renodo[j];
           if jjj >= apendo then
           begin
              if not( jjj in conj) then          {encuentra una nueva variable}
              begin
                 conj:=conj +[jjj];                          {y la mete a la grafi
                 for nod := 1 to ayu do
                    if (ayuda[nod] = jjj) then
                    begin
                       inc(licn);                  {icn guarda a la variable k co
                       icn[licn]:= nod;    {adyacencia de la variable que }
                       if nuevo then               {estamos analizando}
                          begin
                             inc(icfc);
                             ip[icfc]:=licn;          {ip nos indica en que lugar
                             nuevo:=false;           {estan las variables endogenas
                          end;                       {la variable i}
                       inc(lenr[icfc]);                {lenr es el numero de adyac
                    end;                              {de la variable i}
              end;
           end;
        end;
   end;    { segundo for i := .....}
   close(archrel);
end;

end.  {unit lee_relaciones}
```

```
UNIT CICLOS;
{Esta unidad es llamada por el programa ULTIMO; esta unidad contiene a la
 subrutina CFC (componentes fuertemente conexas), necesitamos como entrada
 el numero de nodos (variables), adyacencias de cada nodo, cuantas
 adyacencias por nodo, total de adyacencias y un arreglo apuntador que nos
 dice en donde empiezan las adyacencias de cada nodo. La salida es el
 numero de bloques encontrados y cuales son estos}
interface
uses varglobal;

PROCEDURE CFC(N:integer;ICN:ARR5;LICN:integer;IP:ARR1;LENR:ARR3;VAR IOR:ARR3;VAR

implementation
VAR
  IW,N:integer;
  MATRIZ:ARRAY[1..MAX,1..MAX] OF integer;
  I,J:integer;
  ARCH:TEXT;

PROCEDURE CFC(N:integer;ICN:ARR5;LICN:integer;IP:ARR1;LENR:ARR3;VAR IOR:ARR3;VAR
   VAR
     LOWL,NUMB:ARR1;

   PROCEDURE CFC1(N:integer;ICN:ARR5;LICN:integer;IP:ARR1;LENR:ARR3;
            VAR ARP:ARR3;VAR IB:ARR3;VAR NUM:integer;VAR LOWL,NUMB:ARR1);

     VAR
        J,ISN,I2,II,IV:integer;
        DUMMY,ROOT,ROOTAX:integer;
        I:integer;
        ICNT,NNM1,LICNT:integer;
        I1,IST,IST1:INTEGER;
        SALE,SIGUE,SIGUEPROC:BOOLEAN;

     PROCEDURE PONE_NODO;
     {pone un nuevo nodo en la pila}
     VAR
        K:integer;

     BEGIN    {PONE_NODO}
       NUMB[IV]:=I2-I1-1;
       IV:=IW;
       IST:=IST+1;
       K:=N+1-IST;
       IB[k]:=IV;
       ARP[k]:=ROOT;
       ROOT:=K;
       LOWL[iv]:=ROOT;
       SIGUEPROC:=FALSE;
     END;     {PONE_NODO}

     PROCEDURE QUITA_NODO;
     {despues de hacer busqueda primero a profundidad y analizar cierto
      nodo, lo quita de la pila, regresando a el nodo que fue visitado
      previamente}
       BEGIN          {QUITA_NODO}
         ROOT:=ARP[ROOT];
         IW:=IV;
         IV:=IB[root];
         IF LOWL[iw] > LOWL[iv] THEN
```

```
    LOWL[iv]:=LOWL[iw]
  END;          {QUITA_NODO}

PROCEDURE BLOQUES;
VAR
STP:integer;

  BEGIN     {BLOQUES}
    IF (LOWL[IV]>ROOT) THEN
       QUITA_NODO
    ELSE
      BEGIN
        NUM := NUM +1;                    {ordena los nodos de un bloque y}
        LICNT := ICNT +1;                 {regresa a el nodo previamente}
        IST1 := N + 1 - IST;              {visitado}
        ROOTAX := ARP[ROOT];
        FOR STP:=IST1 TO ROOT DO          {a los nodos de ese bloque los pone}
          BEGIN                           {en orden correcto, trabajando de}
            IW:= IB[stp];                 {arriba hacia la raiz del bloque}
            LOWL[iw]:=0;
            ICNT:=ICNT+1;
            ARP[ICNT]:=IW;
          END;
        IST:=N-ROOT;
        IB[num]:=LICNT;
        ROOT:=ROOTAX;
        IV:=IB[root];
        IF (IST=0) THEN                   {busca nodos en la pila}
          SIGUE:=FALSE;
        IF (ICNT>=N) THEN                 {pregunta si todos los nodos han}
          SALE:=TRUE;                     {ordenados}
      END;     {ELSE DE IF(LOWL...)}
  END;  {BLOQUES}

  BEGIN     {CFC1}
       ICNT:=0;                           {ICNT es el numero de nodos que se}
       NUM:=0;                            {han encontrado en posicion final}
       NNM1:=N + N - 1;                   {NUM numero de bloques encontrados}
       ARP[n]:=N;
       FOR J := 1 TO N DO                 {inicializacion de arreglos}
        BEGIN
              NUMB[j]:= LENR[j]-1;
              LOWL[j]:= -1;
              IB[j]:= 0;
        END;
       ISN:=0;
       SALE:=FALSE;
       WHILE (ISN<=N) AND (NOT SALE) DO       {busca un nodo para empezar}
          BEGIN
            INC(ISN);
            IF LOWL[isn] < 0 THEN
              BEGIN
                  IV:=ISN;
                  IST:= 1 ;           {IST es el numero de nodos en la pila,
                                       es la punta de la pila}
                  ROOT:= N;           {ROOT indica la posicion de la pila}
                  LOWL[iv]:= N;       {pone el nodo IV en el principio de la pil
                  IB[n]:= IV;
                  DUMMY:=0;
                  SIGUE:=TRUE;
```

```
                    WHILE (DUMMY<=NNM1) AND SIGUE DO
                        BEGIN                       {revisa si tiene que poner un}
                           INC(DUMMY);              {nuevo nodo en la pila o quitar
                           I1:=NUMB[iv];
                           IF I1 < 0  THEN
                              BLOQUES
                           ELSE
                             BEGIN
                                 I2:=IP[iv] + LENR[iv] - 1;
                                 I1:=I2-I1;
                                 SIGUEPROC:=TRUE;
                                 II:=I1-1;
                                 WHILE (II<I2) AND SIGUEPROC DO
                                   BEGIN
                                     INC(II);
                                     IW:=ICN[ii];
                                     IF LOWL[iw] < 0 THEN
                                       PONE_NODO
                                     ELSE
                                       IF LOWL[iw] > LOWL[iv] THEN
                                         LOWL[iv]:=LOWL[iw];
                                   END;
                                 IF SIGUEPROC THEN
                                   BLOQUES;
                             END; {ELSE}
                        END; {WHILE DUMMY}
                  END;    {IF LOWL}
            END;  {WHILE ISN}
      END;  {CFC1}
BEGIN      {CFC}
    CFC1(N, ICN, LICN, IP, LENR, IOR, IB, NUM, LOWL, NUMB);
    IB[NUM+1]:=icfc+1;
END;   {CFC}

END.
{ N       = numero de nodos; orden de la matriz
 ICN[j]   = arreglo que contiene las adyacencias de cada nodo, i.e. en la
            grafica, son los nodos que apuntan a ese vertice
 LICN     = longitud del arreglo ICN
 IP[i]    = arreglo que contiene la posicion en el arreglo ICN, nos indica
            en donde se encuentra el primer elemento del nodo i.
 LENR[i]  = numero de adyacencias del nodo i.
 IOR      = arreglo que contiene a los nodos ordenados por bloques; es la
            permutacion de ICN.
 IB[i]    = arreglo que contiene la posicion en el arreglo IOR, nos indica
            en donde se encuentra el primer elemento del nodo i
 NUM      = numero de bloques encontrados}
```

```
                                        UNIT GCOM ;
    { Esta unidad crea el archivo GCOMP.FOR en lenguaje fortran.
      Lo escribirá en el mismo subdirectorio de donde se tomaron
      los datos. Utiliza los datos obtenidos en las unidades
      lee_rela y ciclos}

INTERFACE
uses varglobal,procemod,ciclos;
procedure gcom1;

IMPLEMENTATION
var
   i, j, n, coev : integer;
   relacion      : string[72];
   conjx         : set of byte;

PROCEDURE GCOM1;
var
  jj,ii    : byte;
  k, i, i2 : integer;
  salida   : text;

function sin_igual(relacion:string):string;
 var
    k:integer;
 begin
      k:= 1;
      while relacion[k] <> '=' do
          inc(k);
       sin_igual :=copy (relacion,k+1,200);
 end;

PROCEDURE VARSOLA(a,k:integer);
var
   kk : integer;

begin
   kk := ior[a];
       writeln(salida,'':6,variable[ayuda[kk]],' = ',arrelnum[ayuda[kk]]);
       writeln(salida,'':6,'G(',(kk-numexo),') = ',variable[kk]);
{       writeln(salida,'':6,'G(',ayuda[kk]-numexo,')= -',
          variable[ayuda[kk]],'+', arrelnum[ayuda[kk]]);}
end;

PROCEDURE VARENCOM(c,d,k:integer);
var
   kk : integer;

begin
   writeln(salida,'C','':6,'variables en  componente conexa ',coev + 1);
   for j := c to (d - 1) do
   begin
      kk := ior[j] ;
      writeln(salida,'':6,'G(',ayuda[kk]-numexo,')= -',
                variable[ayuda[kk]],'+',arrelnum[ayuda[kk]]);
      inc(i);
   end;
   writeln(salida,'C','':6,'fin de la componente conexa ',coev + 1);
   if k < num then
   begin
```

```
      inc(coev);
      writeln(salida,'':6,'IF (EVALUA(',coev,').EQ.1) THEN ');
    end;
end;

procedure cuentacom(c,d:integer;var numaux:integer);
    begin
      for j:= c to d-1 do
          begin
            conjx := conjx + [ior[j]];
            inc(i);
            inc(numaux);
          end;
    end;


procedure cuentasola(c:integer; var numaux:integer);
var
  nombre:string[6];
  i:integer;
  sig:char;
  nula:string;

    begin
      i:=1;
      nombre:=lee_var(i,relaciones[ior[c]+naisladas],nula,sig);
      while i<length (relaciones[ior[c]+naisladas]) do
       begin
          nombre:=lee_var(i,relaciones[ior[c]+naisladas],nula,sig);
          if nombre=endo[ior[c]+naisladas] then
              begin
               inc(numaux);
               conjx:= conjx + [ior[c]];
               i:=200;
              end;
       end;
    end;

procedure cuenta(numexo:integer;var numaux:integer);
var
   k: integer;

begin
 i := 1;
 numaux:=numexo;
 conjx := [];
 for k := 1 to num-1 do
    if (ib[k+1]-ib[k]) > 1 then
        cuentacom(ib[k],ib[k+1],numaux)
    else
        cuentasola(ib[k],numaux);
end;

PROCEDURE LETRASI_N(tipo:list;n:integer);
var
   letrain:string[66];
   i:integer;
   temp:string[6];

begin
```

```
   i:=1;
   repeat
      letrain:='';
      repeat
         temp:=tipo[i];
         if temp[1] in ['i'..'n','I'..'N'] then
            letrain:=letrain + ','+tipo[i];
         inc(i);
      until (length(letrain) >=60) or (i-1=n);
      if letrain<> '' then
      begin
         letrain:=copy(letrain,2,100);
         writeln(salida,'':6,'DOUBLE PRECISION ',letrain);
      end;
   until i-1=n;
end;

PROCEDURE ARCHEXTRA;
var
   i     : integer;
   extra : text;

begin
   assign(extra,directorio+'vacfc.dat');
   rewrite(extra);
   writeln(extra,NUM);
   writeln(extra,coev+1);
   writeln(extra,icfc);
   for i := 1 to NUM + 1 do
      writeln(extra,ib[i]);
   for i := 1 to icfc do
      writeln(extra,ayuda[ior[i]]);
   close (extra);
end;


BEGIN
{ De no haber encontrado componentes fuertemente conexas, inicializa
  los arreglos para fines de escritura de archivos de salida}
   if num = 0 then
   begin
      icfc := 0;
      concfc := [];
   end;
   assign(salida,directorio+'gcomp.for');
   rewrite(salida);
   coev := 0;
   writeln(salida,'':6,'SUBROUTINE GCOMP(G,X)');
   WRITELN(salida,'':6,'IMPLICIT DOUBLE PRECISION (A-H,O-Z)');
   letrasI_N(variable,M);
{    cuenta(numexo,numaux);}
   writeln(salida,'':6,'DIMENSION G(',NUMENDO+1,'), X(',NUMEXO+ICFC,')');
   writeln(salida,'':6,'INTEGER EVALUA');
   writeln(salida,'':6,'COMMON/EVAL/EVALUA(256) ');
   {Asigna a las variable endógenas que se encuentran en componentes
    fuertemente conexas una X }
   for ii := 1 to icfc do
   begin
      writeln(salida,'':6,variable[ayuda[ior[ii]]],' = X(',numexo+ii,')');
   end;
```

```
    {Escribe las relaciones con las G( ) corespondientes}
   for jj := apendo to M do
      begin
         if not (jj in concfc) then
         begin
            writeln(salida,'':6,variable[jj],' = ',arrelnum[jj]);
            writeln(salida,'':6,'G(',(jj-numexo),') = ',variable[jj]);
         end;
      end;
   for k := 1 to num do
      if (ib[k + 1] - ib[k]) = 1 then
         varsola(ib[k], k)
      else
      if ((ib[k + 1] - ib[k]) > 1) then
         varencom(ib[k], ib[k + 1], k);
   for k := 1 to coev do
      writeln(salida,'':6,'ENDIF');

    {Escribe la funcion objetivo en la ultima G}
   writeln(salida,'C','':6,'funcion objetivo');
   writeln(salida,'':6,'G(',numendo+1,') = ((X[1]-',
                  valor[1],')/',valor[1],')**2');
   for i := 2 to (numexo) do
      writeln(salida,'':5,'+',' ((X(',i,') -',
                  valor[i],')/',valor[i],')**2 ');
   writeln(salida,'':6,'RETURN');
   writeln(salida,'':6,'END');
   close(salida);
   archextra;
END;

END.
```

```
   unit procemod;
{esta unidad es utilizada por la unidad LEE_RELACION para hacer la lectura
 del catalogo,quitar los espacios en blanco, revisar que los nombres de
 las variables esten permitidos y que estas esten declaradas}

interface
uses varglobal;
procedure lectcatalogo ;
function eliminablancos(s:string):string;
procedure catalogo(var numexo,numendo:integer;var variable:list;var apuntador:ar
function blancos(s:string):string;
function lee_var(var i:integer;s:string;var relanum:string;var sig:char):string;
function mayuscula(cadena:string):string;
function encontro(numcat:integer;cat:list;v:string;var lugar:integer):boolean;
function existe_arch(nombre:string):boolean;
procedure ordenacion(var conjunto:list;var apuntord:arr3);
function buscavar(busvar:string) : integer;

implementation
  var
  i,j:integer;

FUNCTION EXISTE_ARCH(nombre:string):boolean;
  var
  f:text;

begin
   {$I-}
   assign(f,nombre);
   reset(f);
   close(f);
   {$I+}
   existe_arch :=(IOresult = 0) ;
end;

PROCEDURE LECTCATALOGO;
    {Procedimiento que realiza la lectura del archivo catalogo.dat,
     forma el arreglo llamado variable el cual contiene a las
     variables exogenas, seguidas de las variables endógenas; forma
     el arreglo llamado desc el cual contiene la descripcion de las
     variables y finalmente forma el arreglo llamado valor, el cual
     contiene el valor de las variable exógenas}
var
  i:integer;
  datos:text;
  catalogo:string[50];
  regresa:boolean;

begin
   repeat
   writeln;
   writeln('  Proporciona el subdirectorio en donde se encuentran los archivos
   writeln('  "CATALOGO.DAT" y "RELACION.DAT" : (disco:\subdirectorio\) ');
   readln(directorio);
{   directorio := 'c:\norma\pascal\datos2\';}
   catalogo:=directorio+'catalogo.dat';
   regresa:=false;
   if (existe_arch(catalogo)) then
      begin
         assign(datos, catalogo);
```

```
         reset(datos);
         readln(datos,numexo,numendo);
         M := numexo + numendo;
         for i:=1 to numexo do
         begin
            read(datos,variable[i]);
            read(datos,desc[i]);
            readln(datos,valor[i]);
         end;
         for i:= (numexo+1) to M do
         begin
            read(datos,variable[i]);
            readln(datos,desc[i]);
         end;
         apexo := 1;
         apendo := numexo + 1;
         fin := M + 1;
         close(datos);
      end
   else
      begin
         regresa:=true;
         writeln('  El archivo CATALOGO.DAT no esta en :  ',directorio);
      end;
   until (not regresa);
end;  {lectcatalogo}

FUNCTION ELIMINABLANCOS(s:string):string;
    {Esta función quita los espacios en blancos y cualquier otro
     caracter diferente a letras, numeros o guion en una palabra
     dada, sin permitir que el primer caracter sea un numero }

var
s1:string;
j,l:integer;

begin
   l:=length(s);
   s1:='';
   for j:=1 to l do
      if s[j] in ['A'..'z','0'..'9','_'] then
         s1:=s1+s[j];
   if not( s[1]  in ['A'..'z']) then
      begin
         writeln(s1, '  no es admisible como variable');
         halt;
      end;
   eliminablancos:=s1;
end;    { function eliminablancos }

PROCEDURE CATALOGO(var numexo,numendo:integer;var variable:list;var apuntador:ar
    {Procedimiento que llama a la lectura del catalogo y a la
     funcion eliminablancos. Se obtiene el arreglo de todas las
     variables catalogadas }

begin
   lectcatalogo;
   for i:=1 to M do
      variable[i]:=eliminablancos(variable[i]);
   ordenacion(variable,apuntador);
```

```
end;

function blancos(s:string):string;
{con esta funcion se quitan unicamente espacios en blanco}
  var
  i,l:integer;
  s1:string;
    begin
         l:=length(s);
         s1:='';
         for i:=1 to l do
             if (s[i] <> ' ') then
                 s1:=s1+s[i];
         blancos:=s1;
    end;

FUNCTION LEE_VAR(var i:integer;s:string;var relanum:string;var sig:char):string;
    {Esta funcion busca a las variables en una relacion dada; identifica
     que es una variable cuando empieza con alguna letra del abecedario}
var
  s1:string;

procedure busca_letra(var i:integer;s:string;var relanum:string);

    begin
      while not (s[i] in ['A'..'Z']+['a'..'z']) and (i<=length(s)) do
          begin
             if s[i] <> '=' then
                relanum:= relanum + s[i];
             inc(i);
          end;
    end;

begin
   s1:='';
   busca_letra(i,s,relanum);
   while (s[i] in ['A'..'Z','a'..'z','_','0'..'9']) and (i<=length(s)) do
   begin
      s1:=s1+s[i];
      inc(i);
   end;
   lee_var:=s1;
   sig:=s[i];
end;

FUNCTION MAYUSCULA(cadena:string):string;
    {Esta funcion escribe en mayusculas una cadena dada}

var
  cont:integer;
  aux:string;

begin
   aux:='';
   for cont:=1 to length(cadena) do
     aux:=aux+upcase(cadena[cont]);
   mayuscula:=aux;
end; {function mayuscula}

 function encontro(numcat:integer;cat:list;v:string; var lugar:integer):boolean;
```

```
{busca a la variable 'v' en el arreglo 'cat', si la encuentro entonces
 encuentra sera verdadero de lo contrario falso}

 begin
   lugar:=1;
   while (mayuscula(v)<>mayuscula(cat[lugar])) and (lugar<=numcat) do
      inc(lugar);
   if (lugar > numcat) then
      encontro:=false
   else
      encontro:=true;
 end; {function encontro}

PROCEDURE ORDENACION(var conjunto:list;var apuntord:arr3);
    {Este procedimiento ordena alfabeticamente un arreglo
     de variables dado, dando como resultado un arreglo
     apuntador}

var
  k,i,j,menor:integer;
  sal:text;

begin
   for i := 1 to M do
      apuntord[i] := i;
   for i := 1 to M-1 do
   begin
      k := i;
      menor := apuntord[i];
      for j := i + 1 to M do
         if conjunto[apuntord[j]] < conjunto[menor] then
         begin
            k := j;
            menor := apuntord[j];
         end;
      apuntord[k] := apuntord[i];
      apuntord[i] := menor;
   end;
   assign(sal,directorio+'sal');
   rewrite(sal);
   writeln(sal,'Vars. leidas   Vars. ordenadas    Apuntador');
   for i:=1 to M do
      writeln(sal,i,'':4,conjunto[i],'':12,
                        conjunto[apuntord[i]],'':16,apuntord[i]);
   close(sal);
end;

FUNCTION BUSCAVAR(busvar:string) : integer;
var
  low, high, j : integer;

begin
   low  := 0;
   high := M;
   while high - low > 1 do
   begin
      j := (high + low) div 2;
      if busvar <= variable[apuntador[j]] then
         high := j
      else
```

```
        low  := j
    end;
    if busvar = variable[apuntador[high]] then
       buscavar := apuntador[high]
    else
       buscavar := -1;
end;

end.      {fin de la unidad procemod}
```

```
UNIT VARGLOBAL;
interface
const
     max = 150;
     MAXARC =500;
type
     list = array[1..max] of string[6];
      val = array[1..max] of real;
     arr1 = array[1..max] of integer;
     arr2 = array[1..max] of string[72];
     arr3 = array[1..max] of integer;
     arr4 = array[1..max] of string[59];
     ARR5 = ARRAY[1..MAXARC] OF INTEGER;
var
     exo,endo,variable,conjunto:list;
     naisladas,numexo,numendo,numaux:integer;
     apexo,apendo,fin,M:integer;
     icfc,licn,num,n:integer;
     icn       :ARR5;
     ip        :arr1;
     ayuda     :arr1;
     lenr,ior,ib:arr3;
     relaciones,arrelnum:arr2;
     valor:val;
     desc:arr4;
     directorio:string[30];
     apuntador,apuntord:arr3;
     relanum:string;
     concfc : set of byte;

implementation
end.
```

```
CC    ------------------------------------------------------------
CC    PROGRAMA PRINCIPAL
CC    SIDEM  (SIMULADOR DETERMINISTICO 1994)
CC    ------------------------------------------------------------
      CHARACTER*30 DESC(120)
      CHARACTER*50 TITULO(20)
      CHARACTER*20 NUMERO,LEENUM(20),AUX
      CHARACTER*6 VARIAB(120),VARSEP,NUEVA,NOMCOM,AVAR
      CHARACTER*1 VAREN(72),RESP,NUEMET(72),RENGLO(72)
      INTEGER NUMCUA,VEC,NV(20),METAS(120),FIJOS(120),CEROS(120),
     /        VARCFC(60),CUADRO,MENOR(120),MAYOR(120)
      DOUBLE PRECISION VALIN(60)
      INTEGER NEXO,NENDO,TOT,VARCOM(20),NUM,TOTCOM
      INTEGER NCORE
      INTEGER EVALUA,IB(120)
      DIMENSION VEC(10,20),AVAR(10,10),AVAL(10,10)
      DIMENSION VALOR(120),VMINIM(120),VMAXIM(120)
      DOUBLE PRECISION  Z(9000)
      COMMON/EVAL/EVALUA(256)
      COMMON/VARS/VARIAB, NEXO, NENDO
      COMMON/VAL/VALOR,VMAXIM,VMINIM
      COMMON/NYV/NUMERO
      COMMON/PROBL/NORMA
      COMMON/DYNAM1/KX,KG,KALB,KUB,KICAND,KISTAT,KIFIX
      DATA NCORE /9000/
CC    ------------------------------------------------------------
      write(*,*)'entrando a sidem.....'
      OPEN(UNIT = 10, FILE = 'lpt1')
      KBAND=0
      JJJ=0
C
CC    *****       SUBROUTINE LECACU
CC    SE EJECUTA LA LECTURA DEL ARCHIVO CATALOGO.DAT
16    WRITE(*,15)
15    FORMAT(' TENGA CUIDADO EN ESTAR EN EL SUBDIRECTORIO EN EL CUAL',/
     *       ,' TRABAJO CON EL PROGRAMA GENERA, EN ESTE DEBEN ESTAR ',/
     *       ,' LOS ARCHIVOS CATALOGO.DAT, CUADRO.DAT, VACFC.DAT, ',/ )
      open(unit=2,file='CATALOGO.DAT')
      READ(2,1)NEXO,NENDO
1     FORMAT(I2,1X,I2)
      DO 10 I=1,NEXO
          READ(2,2)VARIAB(I),DESC(I),VALIN(I)
2         FORMAT(A6,A60,F20.4)
10    CONTINUE
      DO 20 J=1,NENDO
          READ (2,3)VARIAB(J+NEXO),DESC(J+NEXO)
3         FORMAT(A6,A60)
20    CONTINUE
      TOT = NEXO + NENDO
      close(unit=2)
      write(*,*)'numexo,numendo ----->'nexo,nendo
CC    *****       fin de lecacu
CC    *****       SUBROUTINE LEVATA(K)
CC    IDENTIFICA LAS VARIABLES ESPECIFICADAS EN TABLAS.
CC    LAS VARIABLES SERAN ALMACENADAS EN LA MATRIZ VEC (VARIABLES EN
CC    CUAROS), LA PRIMERA ENTRADA DE LA MATRIZ CORRESPONDE A EL NUMERO
CC    DE TABLA, LA SEGUNDA ENTRADA CORRESPONDE A EL NUMERO DE VARIABLE
CC    EN ESA TABLA.
CC    SE INICIALIZAN ARREGLOS Y CONTADORES.
```

```
CC
      open(unit=4,file='CUADRO.DAT')
      READ(4,11)NUMCUA
11    FORMAT(I2)
      DO 30 K=1,NUMCUA
         READ(4,4)TITULO(K)
4        FORMAT(A50)
         VARSEP='      '
         NV(K)=0
         READ(4,110)VAREN
110      FORMAT(72A1)
         JJ=1
         DO 100 J=1,72
C        EJECUTA LA LECTURA DEL RENGLON E IDENTIFICA CADA VARIABLE
C        POR LAS COMAS,IGNORA LOS BLANCOS ENTRE VARIABLES.  INDICA FIN
C        DE LAS VARIABLES DE ESA TABLA.
            IF (VAREN(J).NE.' '.OR.J.EQ.72) THEN
               IF (VAREN(J).NE.','.AND.VAREN(J).NE.' ') THEN
                  VARSEP(JJ:JJ)=VAREN(J)
                  JJ=JJ+1
               ELSE
                  JJ=1
                  CALL BUSCAR(VARSEP)
                  NVAR=NORMA
                  IF (NVAR.NE.0) THEN
                     NV(K)=NV(K)+1
                     VEC(K,NV(K))=NVAR
                     VARSEP='      '
                  ELSE
                     WRITE(*,130)VARSEP,K
                     STOP
                  ENDIF
               ENDIF
            ENDIF
100      CONTINUE
30    CONTINUE
      close(unit=4)
130   FORMAT('  LA VARIABLE  ',A6,' NO ESTA EN EL CATALOGO'
     1       ,/,'  PERO APARECE EN EL CUADRO ',I3)
CC    *****      fin de levata

C     *****       SUBROUTINE LEVACFC
C     ESTA SUBRUTINA LEE DEL ARCHIVO VACFC.DAT LOS DATOS DE LAS
C     COMPONENTES FUERTEMENTE CONEXAS.
C     TOTCOM      =  NUMERO DE COMPONENTES CONEXAS SEGUN CICLOS
C     COEV        =  NUMERO REAL DE COMPONENTES CONEXAS(SIN VARS. SUELTAS)
C     NVACFC      =  NUMERO TOTAL DE NODOS INVOLUCRADOS EN LAS
C                    TOTCOM COMPONENTES
C     IB          =  APUNTADOR DE CFC
C     VARCFC      =  VARIABLES EN COMPONENTE CONEXA, MANTENIENDO
C                    EL ORDEN DE ENTRADA
C
      open(unit=7,file='VACFC.DAT')
      READ(7,*)TOTCOM
      READ(7,*)COEV
      READ(7,*)NVACFC
      DO 4010  J = 1, (TOTCOM + 1)
         READ(7,*)IB(J)
4010  CONTINUE
      DO 4020  K = 1,NVACFC
```

```
      READ(7,*)VARCFC(K)
4020  CONTINUE
      close(unit=7)
C
C     UNA VEZ LEIDA LA INFORMACION NECESARIA, FORMA LOS CONJUNTOS
C     FIJOS, CEROS Y METAS, LOS CUALES INDICAN EL ESTADO DE
C     DETERMINADA VARIABLE. SI EL CONJUNTO ESTA DESACTIVADO EN
C     EL LUGAR DE LA VARIABLE, LE CORRESPONDE 0, SI ESTA ACTIVADO
C     LE CORRESPONDE 1.
C     FIJOS  =  FIJA A LA VARIABLE CON UN VALOR ESPECIFICO
C     CEROS  =  IGUALA A CERO DETERMINADA RESTRICCION
C     METAS  =  SE PRETENDE OBTENER DETERMINADO VALOR
C
      DO 205 I=1,NEXO
205      FIJOS(I) = 1
      DO 206 I=1,NENDO
         CEROS(I + NEXO) = 0
         METAS(I + NEXO) = 0
206   CONTINUE
      DO 208 I = 1,NVACFC
         FIJOS(I + NEXO) = 0
         MENOR(I + NEXO) = 0
         MAYOR(I + NEXO) = 0
208   CONTINUE
C     INICIALIZA EL ARREGLO DE VALORES DE ENDOGENAS Y EXOGENAS
      DO 215 I= 1,NEXO
215      VALOR(I) = VALIN(I)
      DO 216 I= 1,NENDO
216      VALOR(I+NEXO) = 0
CC    *****       fin de levacf
CC    *****       SUBRUTINA RANGO
CC    SE CREAN LOS RANGOS EN LOS QUE LAS VARIABLES PUEDEN TOMAR UN
CC    VALOR INICIAL, PARA EVITAR INDEFINICIONES.
      IF (TOTCOM.GT.0) THEN
225      WRITE(*,*)' TIENE RESTRICCION EN LOS VALORES? S/N> '
         READ(*,227)RESP
227      FORMAT(A1)
         IF(RESP.NE.'S'.AND.RESP.NE.'N'.
     *      AND.RESP.NE.'s'.AND.RESP.NE.'n') GOTO 225
         IF (RESP.EQ.'S'.OR.RESP.EQ.'s') THEN
275         DO 235 I = 1,NVACFC
               NUM = VARCFC(I)
               WRITE(*,*)'ASIGNACION NUM Y VARCFC(I)--->'NUM,VARCFC(I)
               NUMERO = '                                '
               WRITE(*,240)VARIAB(NUM)
               WRITE(*,245)
               READ(*,250)NUMERO
               IF (NUMERO.NE.'                          ') THEN
                  CALL BANDER(VMINIM,NUM)
                  MAYOR(NUM) = 1
               ENDIF
               WRITE(*,255)
               READ(*,250)NUMERO
               IF (NUMERO.NE.'                          ') THEN
                  CALL BANDER(VMAXIM,NUM)
                  MENOR(NUM) = 1
               ENDIF
235         CONTINUE
         ENDIF
240   FORMAT(10X,A6)
```

```
245   FORMAT('      VALOR MINIMO ?')
250   FORMAT(A20)
255   FORMAT('      VALOR MAXIMO ?')
      ENDIF
CC    *****       fin de subrutina rango
C
C     A PARTIR DE AQUI EL PROCESO SE REPETIRA  TOTCOM  VECES. EN CADA
C     VUELTA SE DEJA CORRER UNA COMPONENTE CONEXA MAS, AGREGANDO LOS
C     RESULTADOS DE LA CORRIDA ANTERIOR.
C
1000  CONTINUE
      DO 13 i = 1, 256
C         EVALUA(i) = 0
          EVALUA(I) = 1
13    CONTINUE
      DO 4100 GLOB = 1, TOTCOM
C      GLOB=TOTCOM
      write(*,*)'el numero de vueltas qu va a dar es 'TOTCOM
      write(*,*)'y se encuentra en la vuelta  'GLOB
      IF (GLOB.LE.(TOTCOM)) THEN
         DO 207 I = 1, (IB(GLOB + 1) - 1)
207         CEROS(VARCFC(I))=1
      ENDIF
CC    *****       SUBROUTINE ENTRADA
CC    ESCRIBE EL ARCHIVO SIDEM.DAT CON LOS DATOS ESPECIFICOS DEL PROBLEMA
      N=NEXO + NVACFC
      M=NENDO+1
      write(*,*)'entrando a escribir sidem.dat'
      open(unit=1,file='SIDEM.DAT')
      write(1,310)N,M
310   format(I5,I5)
      write(1,312)
312   format('BOUNDS')
      do 300 I=1,N
         if (FIJOS(I).EQ.1) then
            write(1,390)I,VALOR(I)
390         format(' E ',I3,4X,F20.4)
         else
           if (MAYOR(VARCFC(I-NEXO)).EQ.1.
     *          and.MENOR(VARCFC(I-NEXO)).EQ.1) then
     *          write(1,320)I,VMINIM(VARCFC(I-NEXO)),
     *                        VMAXIM(VARCFC(I-NEXO))
320           format(' R ',I3,4X,F20.4,F20.4)
            else
              if (MAYOR(VARCFC(I-NEXO)).EQ.1) THEN
                 write(1,325)I,VMINIM(VARCFC(I-NEXO))
325              format(' G ',I3,4X,F20.4)
              else
                if (MENOR(VARCFC(I-NEXO)).EQ.1) THEN
                   write(1,325)I,VMAXIM(VARCFC(I-NEXO))
335                format(' L ',I3,4X,F20.4)
                else
                  write(1,395)I
395               format(' N ',I3)
                endif
              endif
            endif
         endif
300   continue
      write(1,318)
```

```
      write(1,314)
314   format('INITIAL',/,'SEPARATE')
      do 358 J=1,NEXO
         write(1,316)J,VALOR(J)
316      format(3x,I3,4X,F20.4)
358   continue
      write(1,318)
C     escribe la seccion de "ROWS"
      write(1,313)
313   format('ROWS')
      do 350 K=1,M
         if (METAS(K + NEXO).EQ.1) then
            write(1,360)K,VALOR(K + NEXO)
360         format(' E ',I3,4X,F20.4)
         else
            if (CEROS(K + NEXO).EQ.1) then
               write(1,370)K
370            format(' E ',I3,4X,' 0.0 ')
            else
               write(1,380)K
380            format(' N ',I3)
            endif
         endif
350   continue
      write(1,318)
C     escribe la seccion de impresion.
      write(1,399)
399   format('PRINT')
      write(1,365)
365   format('IPR',3X,' -1 ')
      write(1,318)
318   format('END')
      write(1,330)
330   format('GO '/ 'STOP')
      close(unit=1)
CC    *****      fin de entrada
CC    ABRE EL ARCHIVO DE SIDEM.DAT
CC    EJECUTA LA LLAMADA A GRG2, UTILIZANDO LA SUBRUTINA DE GCOMP.FOR.
CC    ASIGNA LOS VALORES OBTENIDOS A LAS VARIABLES DEL PROGRAMA.
C
      OPEN(UNIT=5,FILE='SIDEM.DAT')
      write(*,*) 'Lamamos a grg'
      read(*,*)
      CALL GRG(Z,NCORE)
      close (unit=5)
      WRITE(*,*)'salida de grg2'
C     SE ASIGNAN LOS VALORES OBTENIDOS EN GRG
C     A LAS VARIABLES DE SALIDA DEL PROGRAMA SIDEM
      DO 500 I=1,NEXO
500      VALOR(I)=Z(I)
      DO 510 I=1,NENDO
510      VALOR(I + NEXO)=Z(KG+I-1)
      DO 520 I=1,NVACFC
520      VALOR(VARCFC(I))=Z(NEXO+I)
C
      IF (GLOB.LE.TOTCOM) THEN
         DO 4150  I = 1, (IB(GLOB + 1) - 1)
           METAS(VARCFC(I)) = 1
           CEROS(VARCFC(I)) = 0
4150       FIJOS(VARCFC(I)) = 1
```

```
      ENDIF
      EVALUA(GLOB) = 1
4100  CONTINUE
      IF (KBAND.NE.0) GOTO 2000
C     TERMINA ASIGNACION DE VALORES.
C     SI ES LA PRIMERA VEZ QUE EL USUARIO EJECUTA LA CORRIDA, CONTINUA
C     EN SUBRUTINA ESCENA (MUESTRA LOS DIFERENTES CUADROS QUE EL
C     MISMO USUARIO PROPORCIONO), SI ES LA SEGUNDA EJECUCION O MAS,
C     MUESTRA LOS RESULTADOS EN TABLAS Y CONTINUA EN LINEA 2000.
C
C     - - - - - - - - - - - - - - - - - - - - - - - - -
C     EMPIEZA ITERACION DE SIDEM CON EL USUARIO.
C     PRESENTA CUADROS DE RESULTADOS INICIALES.
C     SE DA LA OPCION A NUEVAS METAS, YA SEA PARA
C     LAS VARIABLE EXOGENAS O PARA LAS ENDOGENAS,
C     ASI COMO AQUELLAS VARIABLES QUE SE QUIEREN
C     DEJAR LIBRES.
C     - - - - - - - - - - - - - - - - - - - - - - - - -
CC    *****       SUBROUTINE ESCENA
C     ESTA SUBRUTINA MUESTRA LOS CUADROS SOLICITADOS
C     POR EL USUARIO EN PANTALLA O EN IMPRESORA.
410   WRITE(*,420)
420       FORMAT(' ------------------------------------------',
     *         /,'          SECCION DE CONSULTA DEL ESCENARIO',
     *         /,'  ESCRIBA EL NUMERO DE CUADRO QUE QUIERE VER',
     *         /,' ------------------------------------------',
     *         /,'                                          ',/)
          WRITE(*,565)NUMCUA
565       FORMAT(' SE TIENEN ',I2,' CUADROS QUE PUEDE CONSULTAR',/)
445       WRITE(*,440)
440       FORMAT(' ESCRIBA  0 (CERO) PARA TERMINAR ',/)
          READ(*,422,ERR=445)CUADRO
422       FORMAT(I2)
          IF (CUADRO.EQ.0) GOTO 600
          IF (CUADRO.GT.NUMCUA) THEN
             WRITE(*,423)NUMCUA
             GOTO 445
          ENDIF
423       FORMAT(' RECUERDE QUE SOLO HAY  ',I2,'  CUADROS ')
C     DESPLIEGA EL CUADRO SOLICITADO CON LOS RESULTADOS OBTENIDOS
430   WRITE(*,424) TITULO(CUADRO)
      DO 435 I=1,NV(CUADRO)
        WRITE(*,425)VARIAB(VEC(CUADRO,I)),DESC(VEC(CUADRO,I)),
     *              VALOR(VEC(CUADRO,I))
435   CONTINUE
C
C     *****       SUBROUTINE IMPRESION
436   CONTINUE
      WRITE(*,426)
426   FORMAT('  QUIERES IMPRIMIR ESTE CUADRO (S/N)? ',/,2X,'>')
      READ(*,427)RESP
427   FORMAT(A1)
      IF(RESP.NE.'S'.AND.RESP.NE.'N'.
     *   AND.RESP.NE.'s'.AND.RESP.NE.'n') GOTO 436
      IF (RESP.EQ.'S'.OR.RESP.EQ.'s') THEN
          WRITE (10,424) TITULO(CUADRO)
424       FORMAT(5X,A50)
          DO 437 I=1,NV(CUADRO)
             WRITE(10,425)VARIAB(VEC(CUADRO,I)),DESC(VEC(CUADRO,I)
     *                    VALOR(VEC(CUADRO,I))
```

```
425          FORMAT(3X,A6,A20,F20.4)
437        CONTINUE
     ENDIF
C    *****      fin de impresion
     GOTO 445
C    SE REPITE HASTA QUE NO SE PIDAN MAS CUADROS
C    ***** fin de escena
600  CONTINUE
     IF (KBAND.NE.0) GOTO 910
C    ***** SUBROUTINE LEEMETAS
C    SE LEEN LAS NUEVAS METAS DADAS POR EL USUARIO PARA
C    CORRER NUEVAMENTE GRG2 Y OBTENER NUEVOS RESULTADOS
     WRITE(*,111)
111  FORMAT(' DESEAS CALCULAR NUEVOS ESCENARIOS ?  (S/N) ',/)
     READ(*,112)RESP
112  FORMAT(A1)
     IF (RESP.EQ.'N'.OR.RESP.EQ.'n') GOTO 1001
635  WRITE(*,610)
610  FORMAT('  ------------------------------------------',
    *       /,'   SECCION DE CALCULO DE ESCENARIOS NUEVOS ',
    *       /,' ------------------------------------------')
605  CONTINUE
C    SE INICIALIZAN CONTADORES
     JJ=0
     KKK=0
     KKMET=0
     JJJ=JJJ+1
645  WRITE(*,640)
640  FORMAT(5X,'PROPORCIONE LAS NUEVAS METAS EN FORMATO:',/,
    *       5X,'NOMBRE DE LA VARIABLE = VALOR, ..., ',/
    *       5X,'EJ. CAP=1000, VAR=50,....',/)
     DO 642 I=1,72
        NUEMET(I)=' '
642  CONTINUE
     READ(*,615)NUEMET
615  FORMAT(72A1)
C    QUITA ESPACIOS EN BLANCO
     II=0
     DO 620 I=1,72
           IF (NUEMET(I).NE.' ') THEN
              II=II+1
              NUEMET(II)=NUEMET(I)
           ENDIF
620  CONTINUE
     I=0
C    BUSCA EL NOMBRE DE LA VARIABLE
649  NUEVA='      '
     II=0
650  I=I+1
     II=II+1
     IF (NUEMET(I).NE.'='.AND.II.LE.6) THEN
         NUEVA(II:II)=NUEMET(I)
         GOTO 650
     ENDIF
      IF (NUEVA.NE.'      ') THEN
C       IDENTIFICA EL VALOR ASIGNADO A LA VARIABLE
        NUMERO='          '
        II=0
651     I=I+1
        IF ((NUEMET(I).EQ.'+'.OR.NUEMET(I).EQ.'-'.OR.NUEMET(I).EQ.'.')
```

```
     *       .OR.(NUEMET(I).GE.'0'.AND.NUEMET(I).LE.'9')) THEN
             II=II+1
             NUMERO(II:II)=NUEMET(I)
             GOTO 651
          ENDIF
C         REGISTRA EL NOMBRE DE UNA VARIABLE Y LO BUSCA EN EL CATALOGO;
C         SI LO ENCUENTRA, GUARDA EL NUMERO DE LA VARIABLE EN EL ARREGLO
C         VARCOM, EL NUMERO DE VARIABLES EN ESTE ARREGLO ESTA DADO POR
C         EL CONTADOR KKK.
CC        PONE UN 1 EN EL CONJUNTO FIJOS EN EL LUGAR "NUM" SI ES EXOGENA
CC        PONE UN 1 EN EL CONJUNTO METAS EN EL LUGAR "NUM" SI ES ENDOGENA
          CALL BUSCAR(NUEVA)
          NUM=NORMA
          WRITE(*,*)'LE CORRESPONDE EL NUMERO  'NUM
          IF (NUM.NE.0) THEN
             IF (NUM.LE.NEXO) THEN
                FIJOS(NUM)=1
             ELSE
                METAS(NUM)=1
             ENDIF
             WRITE(*,*)' EL VALOR REGISTRADO ES  'NUMERO
             KKK=KKK+1
             KKMET=KKMET+1
             AVAL(JJJ,KKK)=VALOR(NUM)
             CALL BANDER(VALOR,NUM)
             VARCOM(KKK)=NUM
          ELSE
C            NO ENCONTRO DECLARADA LA VARIABLE Y MANDA MENSAJE DE ERRCR
             WRITE(*,625)NUEVA
625          FORMAT('  LA VARIABLE ',A6,' NO ESTA EN CATALOGO',/)
             GOTO 605
          ENDIF
       ENDIF
       IF (NUEMET(I).EQ.',') THEN
          GOTO 649
       ELSE
          I=I+1
          IF (NUEMET(I).NE.' ') THEN
             WRITE(*,655)
655          FORMAT('  ERROR EN LA SINTAXIS DE LAS METAS ',/)
             GOTO 605
          ENDIF
       ENDIF
CC     ***** fin de leemetas
C      ***** SUBROUTINE LEEVARIABLES
C      ESTA SUBRUTINA DA LECTURA A LAS VARIABLES LAS CUALES
C      SE QUIERE QUEDEN CON FORMATO LIBRE, ES DECIR AQUELLAS
C      QUE PUEDEN TOMAR CUALQUIER VALOR.
C      ESTAS VARIABLES JUNTO CON LAS VARIABLES DADAS EN METAS
C      SERAN LAS QUE SE DESPLIEGEN EN LAS TABLAS.
C      SE INICIALIZAN ARREGLOS Y CONTADORES
       DO 844 K=1,72
          RENGLO(K)=' '
844    CONTINUE
C      SE DESPLIEGAN EN LA PANTALLA INSTRUCCIONES
835    WRITE(*,800)
800    FORMAT(' PROPORCIONE LAS VARIABLES QUE QUIERAN DEJAR LIBRES',/
     *       6X,'CON FORMATO:   NOMBRE, NOMBRE ,... ',/
     *       6X,'EJ. VAR1, VAR5, ...',/,'>')
       DO 803 I=1,72
```

```
         RENGLO(I)=' '
803   CONTINUE
      READ (*,810)RENGLO
810   FORMAT(72A1)
C     QUITAMOS LOS BLANCOS
      II=0
      DO 805 I=1,72
            IF (RENGLO(I).NE.' ') THEN
               II=II+1
               RENGLO(II)=RENGLO(I)
            ENDIF
805   CONTINUE
      I=0
C     BUSCA EL NOMRE DE LA VARIABLE
840   NOMCOM='          '
      II=0
815   I=I+1
      II=II+1
      IF (RENGLO(I).NE.','.AND.II.LE.6) THEN
         NOMCOM(II:II)=RENGLO(I)
         GOTO 815
      ENDIF
C     BUSCA LA VARIABLE EN EL CATALOGO. SI LA ENCUENTRA LA PONE EN
C     EL ARREGLO VARCOM;
C     EL CONTADOR KKK, SE SIGUE INCREMENTANDO.
      CALL BUSCAR(NOMCOM)
      NUM=NORMA
      IF (NUM.NE.0) THEN
         IF (NUM.LE.NEXO) THEN
            PIJOS(NUM) = 0
         ELSE
            METAS(NUM) = 0
         ENDIF
         KKK=KKK+1
         VARCOM(KKK)=NUM
         AVAL(JJJ,KKK)=VALOR(NUM)
      ELSE
         WRITE(*,825)NOMCOM
825      FORMAT(' LA VARIABLE ',A6,' NO ESTA EN EL CATALOGO')
         GOTO 835
      ENDIF
      IF (RENGLO(I).EQ.',') GOTO 840
C     LO REPITE HASTA TENER EN EL ARREGLO, TODOS LAS VARIABLES
C     QUE SE VAN A DEJAR LIBRES.
C     ******* fin de leevariables
      KBAND=1
      GOTO 1000
2000  CONTINUE
      IF (KBAND.GE.100) GOTO 755
C     *****       SUBRUTINA ESCRIBE Y LEE
C     SE DESPLIEGAN LAS VARIABLES SOLICITADAS COMO METAS CON
C     LOS VALORES CORRESPONDIENTES, ASI COMO LAS VARIABLE
C     DADAS COMO FORMATO LIBRE, DESPUES DE PASAR POR GRG.
      DO 750 I=1,KKK
            AVAR(1,I)=VARIAB(VARCOM(I))
750   CONTINUE
755   CONTINUE
      JJJ=JJJ+1
C     ASIGNA A LA MATRIZ AVAL (ARREGLO DE VALORES), LOS VALORES
C     YA OBTENIDOS
```

```
      DO 760 I=1,KKK
         AVAL(JJJ,I)=VALOR(VARCOM(I))
760   CONTINUE
765   CONTINUE
C     DESPLIEGA EN LA PANTALLA LA TABLA, MOSTRANDO DEBAJO DE CADA VARIABLE
C     EL VALOR OBTENIDO
      WRITE(*,730)(AVAR(1,I),I=1,KKK)
730   FORMAT(2X,10(14X,A6))
      DO 780 K=1,JJJ
         WRITE(*,740)K,(AVAL(K,J),J=1,KKK)
740      FORMAT(I2,10(F20.4))
780   CONTINUE
712   WRITE(*,745)
745   FORMAT(/,2X,'> PUEDE PROPORCIONAR NUEVAS METAS PARA LA(S)',/
     *       ,2X,'> VARIABLE(S) YA ELEGIDA(S) CON ANTERIORIDAD.',/
     *       ,2X,'> PARA TERMINAR SOLO TRANSMITA SIN DAR VALOR(ES)',/
     *       ,2X,'>')
C     SE HACE LA LECTURA DE NUEVOS VALORES, PARA LAS VARIABLES METAS
      DO 782 I=1,10
         LEENUM(I)='
782   CONTINUE
      READ(*,710)(LEENUM(I),I=1,10)
710   FORMAT(10A20)
      DO 716 II=1,KKMET
         IF (LEENUM(II).EQ.'                    ') GOTO 3000
716   CONTINUE
      II=0
      DO 768 I=1,KKMET
         AUX = '          '
         NUMERO = '          '
         AUX=LEENUM(I)
         DO 775 K=1,20
           IF ((AUX(K:K).EQ.'+'.OR.AUX(K:K).EQ.'-'.OR.AUX(K:K).EQ.'.')
     *       .OR.(AUX(K:K).GE.'0'.AND.AUX(K:K).LE.'9')) THEN
              II=II+1
              NUMERO(II:II)=AUX(K:K)
           ENDIF
775      CONTINUE
         NUM=VARCOM(I)
         CALL BANDER(VALOR,NUM)
768   CONTINUE
C     TERMINA LA ASIGNACION DE VALORES A LAS NUEVAS VARIABLES.
C     ***** fin de escribe y lee
C     SE VA A LA ETIQUETA 1000 PARA VOLVER A CREAR EL SIDEM.DAT
C     Y CORRER NUEVAMENTE GRG2
      KBAND=100
      GOTO 1000
3000  CONTINUE
C     *****     SUBROUTINE IMPRIMIR
C     CUANDO YA NO SE DAN MAS VALORES PARA LAS METAS, SE PREGUNTA
C     POR LA IMPRESION DE LA TABLA CREADA.
      WRITE(*,920)
920   FORMAT('  DESEA IMPRIMIR SU TABLA EN PAPEL (S/N)? ',/)
      READ(*,925)RESP
925   FORMAT(A1)
         IF (RESP.EQ.'S'.OR.RESP.EQ.'s') THEN
C           ESCRIBE EN IMPRESORA LA TABLA
            WRITE(10,730)(AVAR(1,I),I=1,KKK)
            DO 785 K=1,JJJ
               WRITE(10,740)K,(AVAL(K,J),J=1,KKK)
```

```
785         CONTINUE
          ELSE
            IF (RESP.NE.'S'.AND.RESP.NE.'N'.
     *          AND.RESP.NE.'s'.AND.RESP.NE.'n') THEN
                GOTO 3000
            ENDIF
          ENDIF
C     ***** fin de imprimir
      GOTO 410
910   CONTINUE
C     *****      SUBRUTINA RESTO
C     SI SE QUIERE SE RECUPERA LA TABLA DE ESCENARIOS QUE TENIA
932   WRITE(*,930)
930   FORMAT(' DESEA RECUPERAR SU TABLA DE ESCENARIOS (S/N)? ',/,'>')
      READ(*,935)RESP
935   FORMAT(A1)
       IF (RESP.EQ.'S'.OR.RESP.EQ.'s') THEN
          GOTO 765
       ELSE
          IF (RESP.NE.'N'.AND.RESP.NE.'n') GOTO 932
       ENDIF
C     SI SE QUIERE SE PUEDEN PONER NUEVAS VARIABLES Y/O NUEVAS METAS
942   WRITE(*,940)
940   FORMAT(' DESEA CAMBIAR SUS VARIABLES Y METAS (S/N)?',/,'>')
      READ(*,945)RESP
945   FORMAT(A1)
       IF (RESP.EQ.'S'.OR.RESP.EQ.'s') THEN
C            SE LIMPIA JJJ PARA EMPEZAR EN EL RENGLON 1 NUEVAMENTE
             JJJ=0
             GOTO 635
          ELSE
             IF(RESP.NE.'N'.AND.RESP.NE.'n') GOTO 942
          ENDIF
1001   WRITE(*,1002)
1002   FORMAT(' FIN DE SIDEM  ')
C     ***** fin de resto
C     *****  F I N   D E   S I D E M
      STOP
      END
```

```
CCC   ----- subrutinas y funciones usadas -----
      SUBROUTINE BUSCAR(VARAUX)
C     ESTA SUBRUTINA BUSCA VARAUX EN EL ARREGLO DE VARIABLES
C     SI ENCUENTRA A LA VARIABLE PONE EN NVAR EL NUMERO QUE
C     LE CORRESPONDE EN LA LISTA. SI NO, HACE NVAR IGUAL A CERO.
      CHARACTER*6 VARAUX,VARIAB
      INTEGER NEXO,NENDO,TOT
      DIMENSION VARIAB(120)
      COMMON/PROBL/NORMA
      COMMON/VARS/VARIAB,NEXO,NENDO
C
      TOT = NEXO + NENDO
      DO 50 J=1,TOT
      IF (VARAUX.EQ.VARIAB(J)) THEN
          NORMA=J
          RETURN
      ENDIF
 50   CONTINUE
      NORMA=0
      RETURN
      END

      SUBROUTINE BANDER(VAL1,NUM)
C     ESTA SUBRUTINA CAMBIA EL VALOR DE LA VARIABLE DE TIPO
C     CARACTER A TIPO REAL O ENTERO
      CHARACTER*20 NUMERO
      DIMENSION VALOR(120),VAL1(60),VMAXIM(120),VMINIM(120)
      COMMON/VAL/VALOR,VMAXIM,VMINIM
      COMMON/NYV/NUMERO
      BA=0
      DO 711 NN=1,20
       IF(NUMERO(NN:NN).EQ.'.') THEN
         BA=1
       ENDIF
711   CONTINUE
      IF(BA.EQ.1) THEN
         READ(NUMERO,660)RVALOR
660      FORMAT(F20.4)
         VAL1(NUM)=RVALOR
      ELSE
         READ(NUMERO,664)IVALOR
664      FORMAT(I20)
         VAL1(NUM)=IVALOR
      ENDIF
c      WRITE(*,*)'ACABO BANDERA EL VALOR ES,EN EL LUGAR>>>> 'VAL1(NUM),NUM
c      WRITE(*,*)'ACABO BANDERA VMAXIM ES,EN EL LUGAR>>>> 'VMAXIM(NUM),NUM
c      WRITE(*,*)'ACABO BANDERA MINIMO ES,EN EL LUGAR>>>> 'VMINIM(NUM),NUM
      RETURN
      END
```

APENDICE B

Una gráfica G es una tercia ordenada (V(G), E(G), fi_G) la cual consistente en un conjunto no vacío de vértices V(G), un conjunto de arcos E(G) y una función de incidencia fi_G que asocia a cada arco de G un par ordenado de vértices (no necesariamente distintos).

Una digráfica o gráfica dirigida es una gráfica en donde los arcos son pares ordenados (v,w) de vértices; a v se le llama la cola y a w la cabeza del arco (v,w).

Se dice que el vértice w es adyacente al vértice v si existe $(v,w) \subset E$.

Un arco con extremos idénticos es llamado un "loop".

Un camino del nodo v al nodo u, es una sucesión de arcos de la forma (v,n_1), $(n_1,n_2),\ldots,(n_k,u)$. Si $v,n_1,n_2 \ldots n_k,u$ son nodos distintos se dice que el camino es simple o que es una trayectoria.

Un arco con extremos diferentes es llamado una cadena o camino.

Un ciclo es un camino que va de u a v y además u=v.

Una gráfica sin ciclos es una gráfica acíclica.

Un nodo i se dice que está conectado al nodo j, si existe un camino dirigido de i a j. Una digráfica G se dice que es fuertemente conexa si para todo par de nodos i y j, i está conectado a j y j está conectado a i. Es decir, una gráfica es fuertemente conexa si existe un camino entre cualquiera dos vértices de la gráfica.

Un árbol o arboresencia es una digráfica acíclica que satisface las siguientes propiedades:

a) hay exactamente un vértice llamado raíz, al cual no entran arcos.

b) cada vértice excepto la raíz tiene exactamente un arco entrando a él.

c) existe un camino (el cual se puede probar que es único) de la raíz a cada uno de los vértices.

BIBLIOGRAFIA

1. AHO ALFRED V., HOPCROFT JOHN E.y ULLMAN JEFFREY D. *The design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Company, 1974.

2. BONDY J.A., MURTY U.S.R. *Graph Theory with Applications*, The Mac Millan Presss LTD, 1976.

3. CALVILLO VIVES GILBERTO y A. RODRIGUEZ VERA. *Un sistema para simulación determinística*, Nota técnica No. 55, Unidad de investigación y desarrollo, Banco de México, junio 1981.

4. GARDUÑO VERGARA ROGELIO. *Teoría de gráficas en la programación lineal numérica*, Instituto Politécnico Nacional, México D.F., 1985.

5. HERBEST ROBERT TAYLOR. *Softer design using FORTRAN 77*, Prentice Hall, New Jersey, 1990.

6. HAMMOND ROBERT H., WILLIAM B. ROGERS y JOHN B. CRITTENDERN. *Introducción al FORTRAN 77 y a la PC*, Mc Graw-Hill, México, 1989.

7. INTRILIGATOR MICHAEL D. *Econometric Models, Techniques, & Applications*, Prentice-Hall, Inc., New Jersey, 1978.

8. KLEIN L.R. and GOLDBERG A.S. *An Econometric Model of the United States 1929-1952*, North-Holland Publishing Company, Amsterdam, 1955.

9. LASDON LEON S., ALLAN D. WAREN Y MARGERY W. RATNER. *GRG2 User's Guide*, Guia de Operacion No. 63, Banco de Mexico, septiembre 1980.

10. RODRIGUEZ SANCHEZ M. GUADALUPE. *La cerradura Transitiva Generalizada en Redes y sus Aplicaciones*, Instituto Politécnico Nacional, México, D.F. 1989.

11. WIRTH NIKLAUS, *Algorithms + Data Structure = Programs*, Prentice-hall, Inc., New Jersey, 1976.