CAPITULO 4.

SIMULADOR NUMERICO GEO

4.1.- Estructura del simulador numérico GEO.

El simulador numérico GEO está compuesto de nueve módulos integrados como se muestra en la figura 4-1. Las características de cada una de las partes del simulador se definen a continuación.

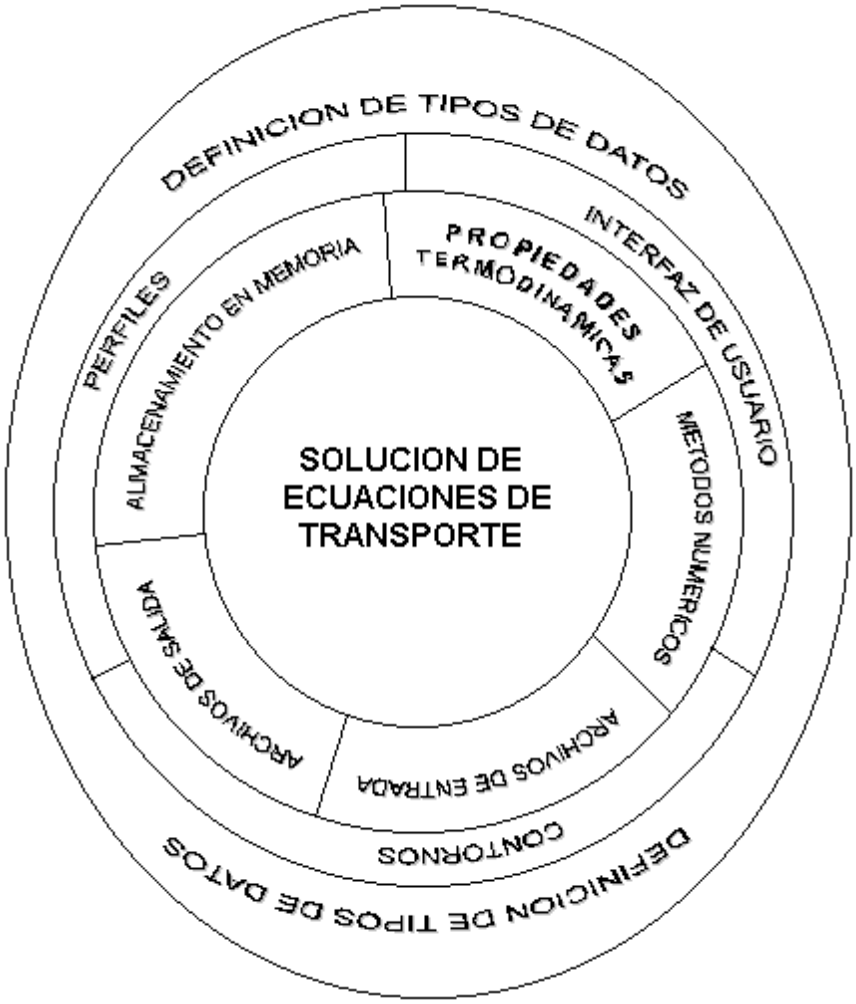

Figura 4-1. Módulos que componen al simulador numérico GEO.

4.1.1.- Módulos que conforman a GEO

1.SOLUCION DE LAS ECUACIONES DE TRANSPORTE. Resuelve simultáneamente el sistema de ecuaciones, no lineales y acopladas del transporte de masa y energía en los yacimientos geotérmicos. Por ser el módulo más importante de GEO se anexa el código fuente en el apéndice A y explicado con mayor detenimiento. Las principales funciones que lo componen son las siguientes:

void tiempo_cpu(void);
Función que calcula el tiempo transcurrido (CPU) por cada iteración generada en cada etapa de tiempo.
void inicia_matriz(void);
Inicializa en ceros en cada etapa de tiempo las matrices que almacenarán al Jacobiano.
void residuo_func(double *,double *,int *);
El tercer argumento encuentra el elemento con mayor residuo del sistema de ecuaciones, regresando en el segundo argumento su valor. El primer argumento es el vector de residuos de todos los elementos de la malla numérica.
int propfis_elem(lista_elemento **,double,double,int);
La energía interna específica y densidad de mezcla en el segundo y tercer argumentos, son transformadas a las propiedades secundarias e introducidas en la base de datos, para el elemento indicado en el cuarto argumento, con la dirección de ubicación en memoria que se indica en el primer argumento.
void prop_elem_pert(lista_elemento **,double,double,int);
Calcula las propiedades perturbadas en un elemento y las introduce a la base de datos. También calcula las propiedades sin perturbar del mismo elemento y las retorna nuevamente a la base de datos. El primero y cuarto argumentos indican la dirección y el número del elemento. El segundo y tercer argumento son la densidad y energía interna específica de mezcla, con o sin la perturbación.

void condini_def_elemts(int,int *,double *,double *,int);
Cuando algún elemento no tiene condiciones iniciales, ésta función se encarga de asignárselas y almacenarlas en la base de datos. El primer argumento indica la saturación de vapor para designar si es una o dos fases, el segundo y quinto argumentos darán el intervalo inicial y final de los elementos sin condiciones, el tercero y cuarto argumento, son los vectores que almacenan la energía interna específica y la densidad de mezcla.
void tolerancia(double *,double *,int *,double *,double *);
Da las tolerancias permitidas como son el valor de la perturbación en la ecuación de densidad y energía, máximo número de iteraciones por etapa de tiempo, error máximo permitido en los residuos de las ecuaciones de energía y masa, siendo en ese orden la aplicación de los argumentos.
int transf_Intr_propSint(lista_elemento **,double,double,int);
int transf_Intr_propSuno(lista_elemento **,double,double,int);
int transf_Intr_propScero(lista_elemento **,double,double,int);
Hace la transformación de las variables de estado e introduce en la base de datos para el elemento asignado, las variables secundarias. La función utilizada es según el valor de la saturación de vapor. Siendo el primero y cuarto argumento, la dirección de

ubicación en memoria del elemento y el número del mismo, el segundo y tercer argumento son las variables de estado energía interna específica y densidad de mezcla.

void flux(int,lista_elemento *[],double,double *,double *,double *,double *,double *,int);

Da solución al sistema de ecuaciones de masa y energía, según se indique en el argumento nueve, encontrando como resultado a los residuos de cada ecuación del sistema. También sirve para calcular los residuos de las ecuaciones perturbadas, el cual se indica en el segundo argumento. Los vectores que llevan la información para cada elemento de energía interna específica y de densidad de mezcla, calculados en una etapa anterior de tiempo y también los actuales, son introducidos a la función a través de los argumentos cuarto, quinto, sexto y séptimo. El incremento de tiempo en cada etapa lo maneja el tercer argumento.

void solve_part(double *,double *,double (*)[TA],double (*)[TA],double (*)[TA], double(*)[TA]);

Resuelve el sistema de ecuaciones lineales encontradas por partición de matrices. La matriz a resolver está dividida en cuatro partes indicadas del tercer al sexto argumentos. El vector de términos independientes entra dividido a través del primer y segundo argumentos.

void profgeo_elem(lista_elemento *,double *,double *,double *,double *,double *);

Encuentra las propiedades, volumen, presión, temperatura, densidad y calor específico de la roca, indicados por los argumentos del segundo al sexto. El primer argumento indica cual es el elemento en el cual se calculan las propiedades.

int Ecflujo(double,int,int,double,double,double,double,double,double *,double *,double*,double*);

Se obtienen como resultado los vectores de las variables de estado para los elementos del sistema, en la etapa de tiempo calculada. La función tiene un valor entero que regresa el valor de uno o cero si se logra o no la convergencia.

void entrada(double *,double *,double *,double *,int *,double *,double *,double *,double*, double *,int *);

Da los datos de entrada que son utilizados a lo largo del programa: incremento por etapa de tiempo, máximo tiempo de cálculo, perturbación en la ecuación de la energía y en la de masa, número máximo de iteraciones por cada etapa de tiempo, errores máximos en las ecuaciones de energía y de masa y los vectores de las variables de estado, para todos los elementos, en un tiempo anterior y en el actual. Al inicio de cada etapa de tiempo los valores de las variables termodinámicas se toman iguales en valor en los tiempos k y k+1, conforme avanza la convergencia estos se van ajustando al tiempo k+1.

void intr_prop_lista(lista_elemento **,double,double, double, double, double,double,double, double , double,double,double,double,int);

Introduce las propiedades físicas calculadas en el nodo de cada elemento. El último argumento indica el número del elemento. Las propiedades son presión, temperatura, densidad, permeabilidad, viscosidad y entalpia, para cada fase (liquido, vapor) y, finalmente, saturaciones de vapor y líquido.

void jac(double (*)[TA],double (*)[TA],double (*)[TA],double (*)[TA],double *, double *, double *, double *,double *,double *,double,double,double, double *,double *);

Forma la matriz Jacobiana. Los primeros cuatro argumentos almacenan la matriz y los últimos dos almacenan el vector de términos independientes. El quinto y el sexto argumento son los vectores de los residuos de las ecuaciones para cada elemento. Los

vectores de energía interna y de densidad de mezcla para todos los elementos en el tiempo anterior y actual corresponden al séptimo, octavo, noveno y décimo. El incremento del tiempo y las perturbaciones para la energía interna específica y para la densidad de mezcla se encuentran en el onceavo, doceavo y treceavo argumentos.

void icara(lista_elemento *,lista_conex *,double *,double *,double *,double *,double *, double *, double *,double *,double *,double *,double *,double *,double *,double *,double *);

Calcula las propiedades termodinámicas en la intercara del elemento indicado en el segundo argumento y regresa estas para ser utilizadas.

2. METODOS NUMERICOS. Contiene las funciones de las técnicas numéricas empleadas para la solución de las ecuaciones gobernantes discretizadas (newton, partición de matrices, método de Gauss), para el cálculo de las propiedades en las intercaras (interpolación de Lagrange) y para el cálculo de las variables termodinámicas (Newton, Gauss).

3. ARCHIVOS DE ENTRADA. Contiene las funciones que permiten la entrada de datos al programa como: las propiedades termodinámicas y termofísicas, condiciones iniciales, parámetros de la simulación transitoria, fuentes/sumideros, tolerancias y perturbaciones en el sistema de ecuaciones, la malla numérica, etc.

4. ARCHIVOS DE SALIDA. Maneja todas las salidas del programa como: archivos de las variables termodinámicas por etapa de tiempo para los perfiles y contornos gráficos por etapa de tiempo en pantalla. Muestra el avance de convergencia por etapa de tiempo.

5. PROPIEDADES TERMODINÁMICAS. Transforma las variables independientes (energía interna específica y densidad de mezcla) a las variables dependientes (temperatura y presión) en función de las cuales se calculan los parámetros físicos restantes.

6. ALMACENAMIENTO EN MEMORIA. Genera el espacio de memoria requerido por las submatrices que almacenan los datos para el Jacobiano y la matriz particionada. La dimensión máxima para cada submatriz es de 51 x 51 de tipo doble precisión (20,808 Bytes). El programa soporta un tamaño máximo de 102 x 102 (83,232 Bytes). La construcción se muestra en la figura 4-2.

$$Jacobiano = \begin{bmatrix} [SUBMATRIZ\,1] & [SUBMATRIZ\,2] \\ [SUBMATRIZ\,3] & [SUBMATRIZ\,4] \end{bmatrix}$$

Figura 4-2. El Jacobiano está formado por cuatro submatrices.

7. INTERFAZ DE USUARIO. Pantalla de interacción con el simulador GEO, la cual tendrá la función de interactuar con el usuario. Aquí se encuentran las funciones de usuario como: presentación de tiempo, fecha, entrada de datos, marcos de presentación y estilos de fuentes.

8. CONTORNOS. Genera los contornos gráficos para todas las propiedades termodinámicas obtenidas por GEO.

9. PERFILES. Genera los perfiles gráficos para la energía interna de mezcla, densidad de mezcla, temperatura y presión.

10. DEFINICION DE TIPOS DE DATOS. Define las variables globales, constantes, nuevos tipos de datos, funciones locales y externas ocupadas en los módulos que conforman al simulador GEO. También contiene a las funciones que manipulan la geometría del sistema (las estructuras que manejan la malla nodal son explicadas en la sección 4.2.3).

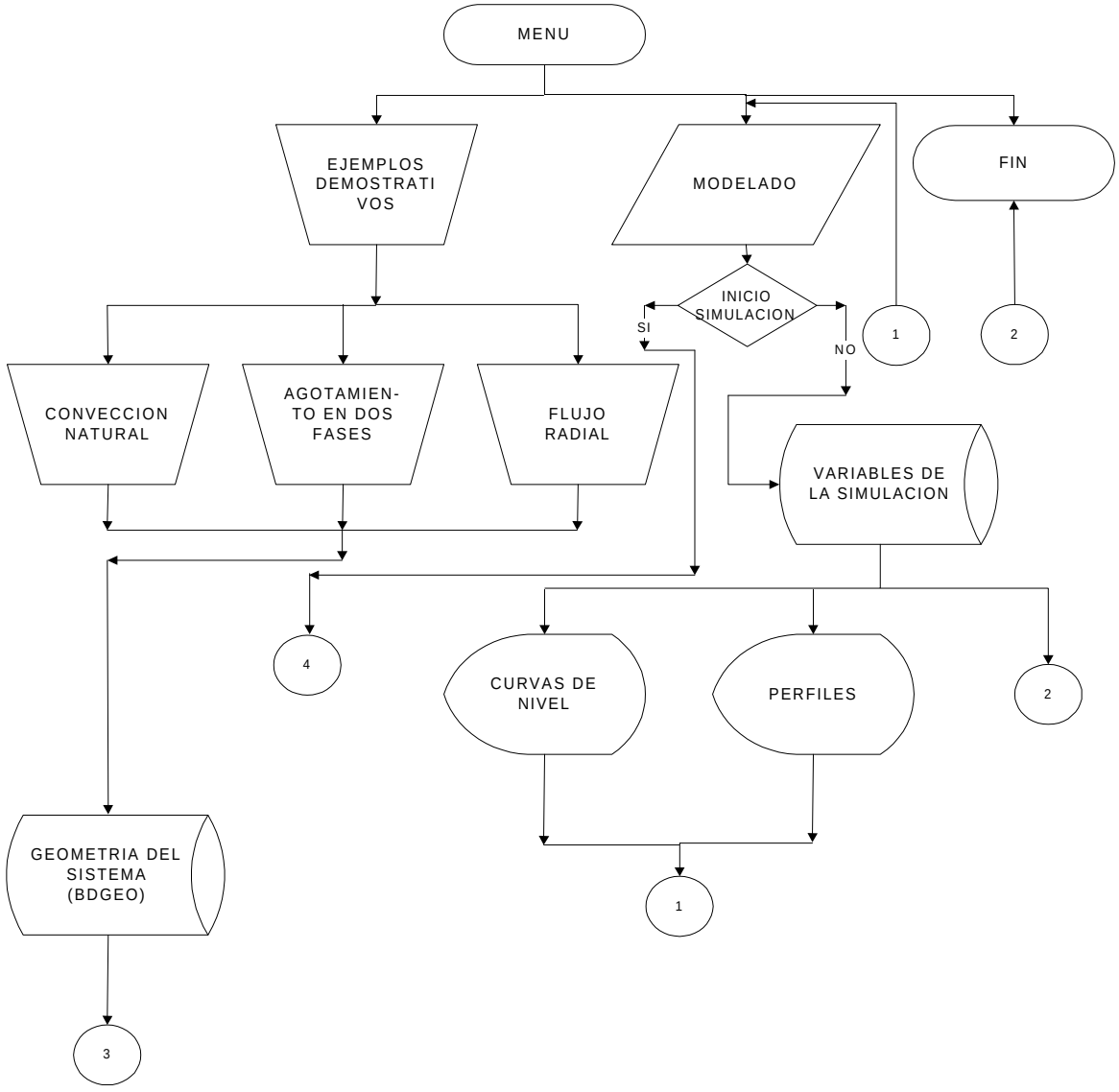

Figura 4-3(a). Diagrama de flujo general del simulador GEO.

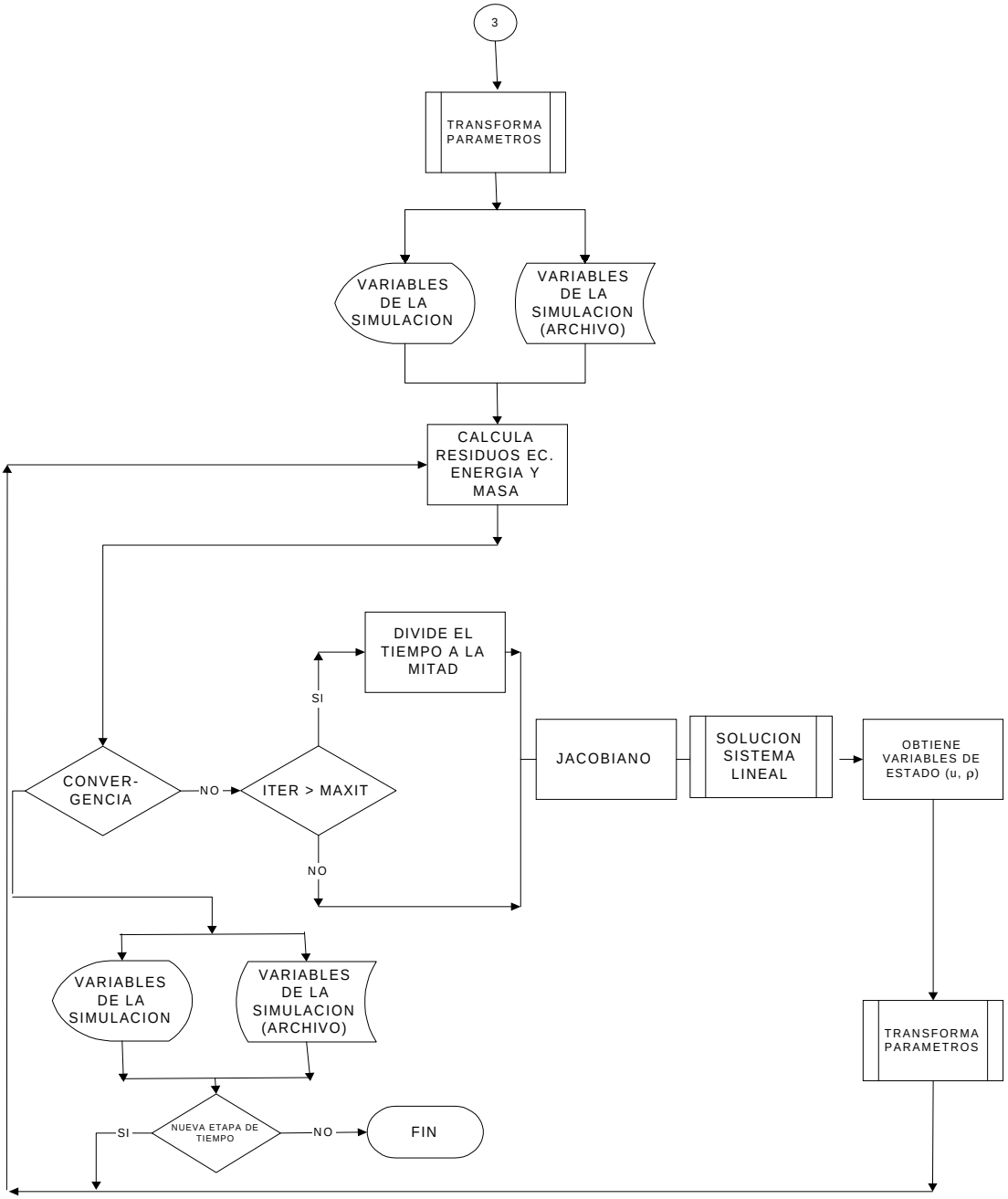

Figura 4-3(b). Diagrama de flujo general del simulador GEO. Solución numérica.

4.2.- Descripción del funcionamiento del simulador GEO

El funcionamiento de GEO se muestra en el diagrama de flujo de la figura 4-3. El simulador se inicializa con un menú de usuario, integrado con una serie de ejemplos demostrativos tomados del manual de usuario del simulador SHAFT79 **[Pruess y Schroeder, 1980]**, dos de los cuales se explican en el capítulo cinco para efectos de

validación del simulador GEO. También al mismo nivel se tiene una sección de modelado y una condición de término del programa.

BDGEO define las características geométricas de cualquier medio en estudio (explicado en la sección 4.2.3) y se encuentra en el módulo de **"ARCHIVOS DE ENTRADA"**. Asimismo en este módulo se presenta la estructura de la entrada de datos que se esquematiza en la figura 4-3(c). Los pasos para encontrar la solución numérica de las ecuaciones gobernantes se esquematizan en el diagrama de flujo de la figura 4-3(b) y pertenecen al módulo **"SOLUCION DE ECUACIONES DE TRANSPORTE"**.

Para el modelado se tiene la opción (figura 4-3(a)) de comenzar un nuevo problema o bien mostrar los resultados almacenados en archivos (módulo de **"ARCHIVOS DE SALIDA"**), de una simulación previa. GEO permite un estudio más fácil del comportamiento de los sistemas físicos al analizar los campos de las variables encontradas a través de salidas gráficas que se presentan en pantalla; estas variables físicas son graficadas por etapa de tiempo, teniendo el usuario la opción de elegir el tipo de gráfica ya sea en forma de perfiles o de curvas de nivel.

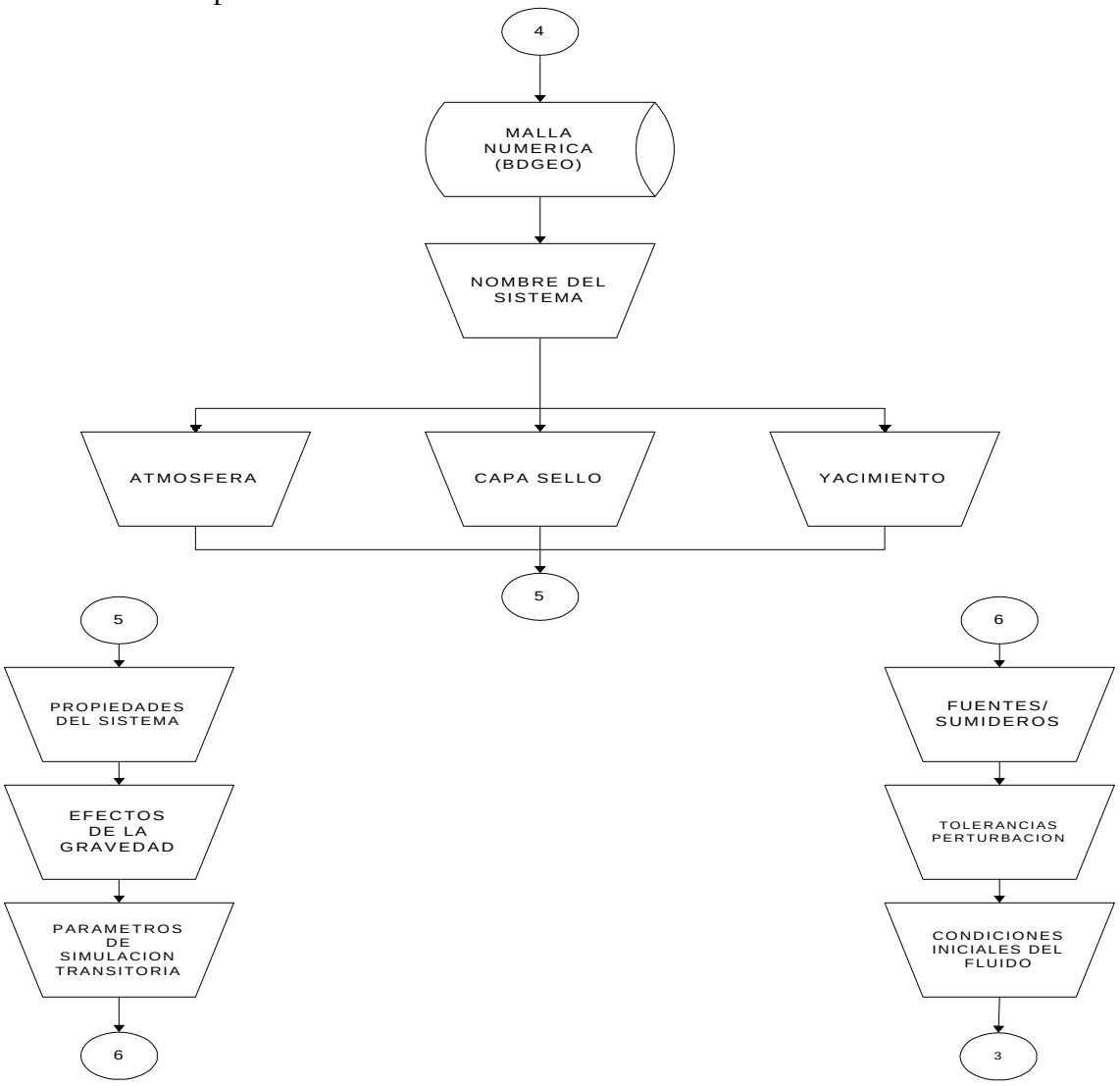

Figura 4-3(c). Diagrama de flujo general del simulador GEO. Entrada de datos.

4.2.1. - Entrada de datos.

Cuando se comienza una simulación, se requiere definir las propiedades termofísicas de la roca geotérmica y las condiciones iniciales del fluido geotérmico (figura 4-3c). Partiendo del conector cuatro lo primero que solicita el programa es el archivo de datos de la malla numérica generado por el módulo **"ARCHIVOS DE ENTRADA"**. Se pide un nombre para el sistema y se delimita la estructura del mismo. Esto es, si el sistema involucra condiciones ambientales y/o capa sello (roca poco permeable) además del yacimiento geotérmico. Se requieren las propiedades para cada una de las partes involucradas. Se establece si la gravedad es tomada en cuenta. La información anterior más la definición de los parámetros de la simulación transitoria, de la presencia de fuentes/sumideros y de las tolerancias permitidas en el manejo de las ecuaciones de masa y energía, deberá ser capturada manualmente por el usuario antes de comenzar una nueva simulación.

4.2.2. - Solución numérica.

El esquema de solución de las ecuaciones gobernantes se muestra en la figura 4-3(b). Una vez establecidas las características del medio físico en estudio, el programa generará resultados para un tiempo inicial igual a cero (condiciones de entrada) utilizando el módulo de **"PROPIEDADES TERMODINÁMICAS"**, mostrándolos en la pantalla y almacenándolos en un archivo en disco para su posterior utilización en el modelado. El nombre de los archivos por cada etapa de tiempo calculada será dado por el propio programa (sist0.dat, sist1.dat, sist2.dat,... ,sistn.dat). La simulación se inicia calculando los residuales del sistema de ecuaciones y se comparan las tolerancias de convergencia respectivas. Si no se logra la convergencia, la etapa de tiempo se divide a la mitad. Posteriormente se comienza a formar la matriz Jacobiana para darle solución como un sistema de ecuaciones lineales simultáneas encontrando el valor de las variables independientes densidad y energía interna específica a partir de las cuales se encontrarán las propiedades termodinámicas restantes. Utilizando los nuevos parámetros se procederá nuevamente a encontrar los residuales del sistema de ecuaciones hasta que la convergencia sea satisfecha. Entonces el programa generará un nuevo archivo de resultados y mostrará estos en pantalla para el tiempo calculado. El proceso continuará hasta el tiempo máximo solicitado o bien con una orden directa de cancelación por parte del usuario.

4.2.3. - Malla numérica.

La importancia del manejo de una buena base de datos para atacar el problema de un sistema con geometría y dimensión cualesquiera llevó a la elaboración del programa BDGEO. Este programa maneja listas doblemente enlazadas bidimensionales (figura 4-4); la lista principal es un conjunto de nodos llamados **nodo_elemento**, en el que cada uno de estos, contiene información de un elemento: propiedades físicas, número del elemento, volumen y número de conectividades. Las listas que cuelgan en cada uno de estos nodos, contienen información de cada conexión a ese **nodo_elemento**. Estas listas están formadas

por nodos llamados **nodo_conexión** y cada uno contiene información del elemento al que están conectados: número del **nodo_elemento** al que se está conectado, número del elemento actuando como **nodo_conexión,** distancias de intercaras y área de la intercara de ese **nodo_conexión** con el **nodo_elemento**.

El utilizar listas bidimensionales da la ventaja de manejar cualquier tipo de geometría en un sistema y además hace un uso eficiente de la memoria disponible de la máquina PC. Esto es, sólo se utiliza la memoria que se requiere, a diferencia de los arreglos bidimensionales.

El programa BDGEO solamente captura la geometría del sistema en estudio y reserva espacio de memoria para almacenar las propiedades físicas de cada uno de los elementos que conforman el medio; genera un archivo de datos que contiene la descripción geométrica del sistema geotérmico en estudio, el cual será utilizado por el simulador GEO.

BDGEO consta de un menú de usuario principal con cuatro opciones:

1.- Elementos componentes del sistema.
2.- Conexiones a elementos del sistema.
3.- Lectura de la base de datos del disco.
4.- Almacenamiento de la base de datos en el disco.

La primera opción tiene un submenú con lo siguiente.
1.- Insertar un elemento.
2.- Eliminar un elemento.
3.- listar elementos individuales.

La segunda opción tiene un submenú con lo siguiente.
1.- Insertar conexiones.
2.- Eliminar conexiones.
3.- Listar conexiones entre elementos.

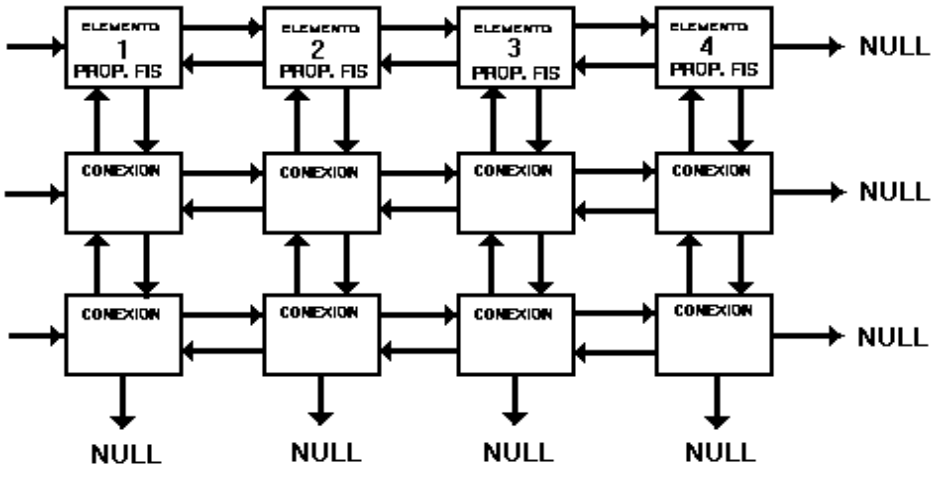

Figura 4-4. Esquema conceptual de las listas bidimensionales doblemente enlazadas para generar la geometría requerida.

De esta forma la geometría puede ser cambiada a conveniencia del usuario o generar diferentes archivos con características específicas para cada uno de estos y guardarlos para posteriores estudios.

La estructura para definir al **nodo_elemento** es

```
typedef struct elemento{
    int num_elem;     int num_conec;    double vol_elem;   double sat_liqn;
    double sat_vapn;  double den_liqn;  double den_vapn;   double k_absn;
    double k_liqreln; double k_vapreln; double visc_liqn;  double visc_vapn;
    double pn;        double tn;        double cond_termn; double entp_liqn;
    double entp_vapn; double den_roca;  double cp_roca;    double Q;
    double q;         double x;         double y;
    struct elemento *sig; struct elemento *ant; struct conexiones *abajo;
 }lista_elemento;
```

Donde en orden de aparición tenemos: número asignado al elemento, número de conexiones del elemento, volumen del elemento, propiedades termodinámicas en el nodo, coordenadas ‘x’, ‘y’ de ubicación del elemento en metros (utilizadas para las salidas gráficas), dos apuntadores (*sig, *ant) que generan la lista doblemente enlazada de elementos, y el último apuntador (*abajo), que comunica a la lista de conexiones.

La estructura para definir al **nodo_conexión** es

```
typedef struct conexiones{
    int num_elem;       int elem_conect;    double dist_N_Int_M;
    double dist_M_Int_N; double area_interf; double cos_ang;
    struct conexiones *arriba; struct elemento *arriba0;
    struct conexiones *abajo;
 }lista_conex;
```

Donde en orden de aparición de los campos que componen la estructura, tenemos: el número del elemento que trabaja como conexión, el número del elemento al que está conectado, la distancia a las intercaras, el área de intercara, el ángulo creado con la línea formada por el nodo conexión al nodo elemento y el vector de gravedad, dos apuntadores (*arriba, *abajo) que forman la lista doblemente enlazada de conexiones y el último apuntador (*arriba0) que comunica la lista de conexiones con la lista principal de elementos.

4.2.4.- Algoritmo para la malla numérica.

De la figura 4-5 se puede ver la estructura del programa BDGEO que captura y ordena la geometría del sistema. El usuario puede generar a través del programa dos archivos para la lista principal de elementos y la lista secundaria de conexiones respectivamente. El contenido de estos puede ser cambiado en cualquier momento. Se tiene acceso a modificar las características de los elemento y conexiones; también muestra en la pantalla la información de cada elemento con sus conexiones respectivas para que el usuario pueda verificar los datos capturados.

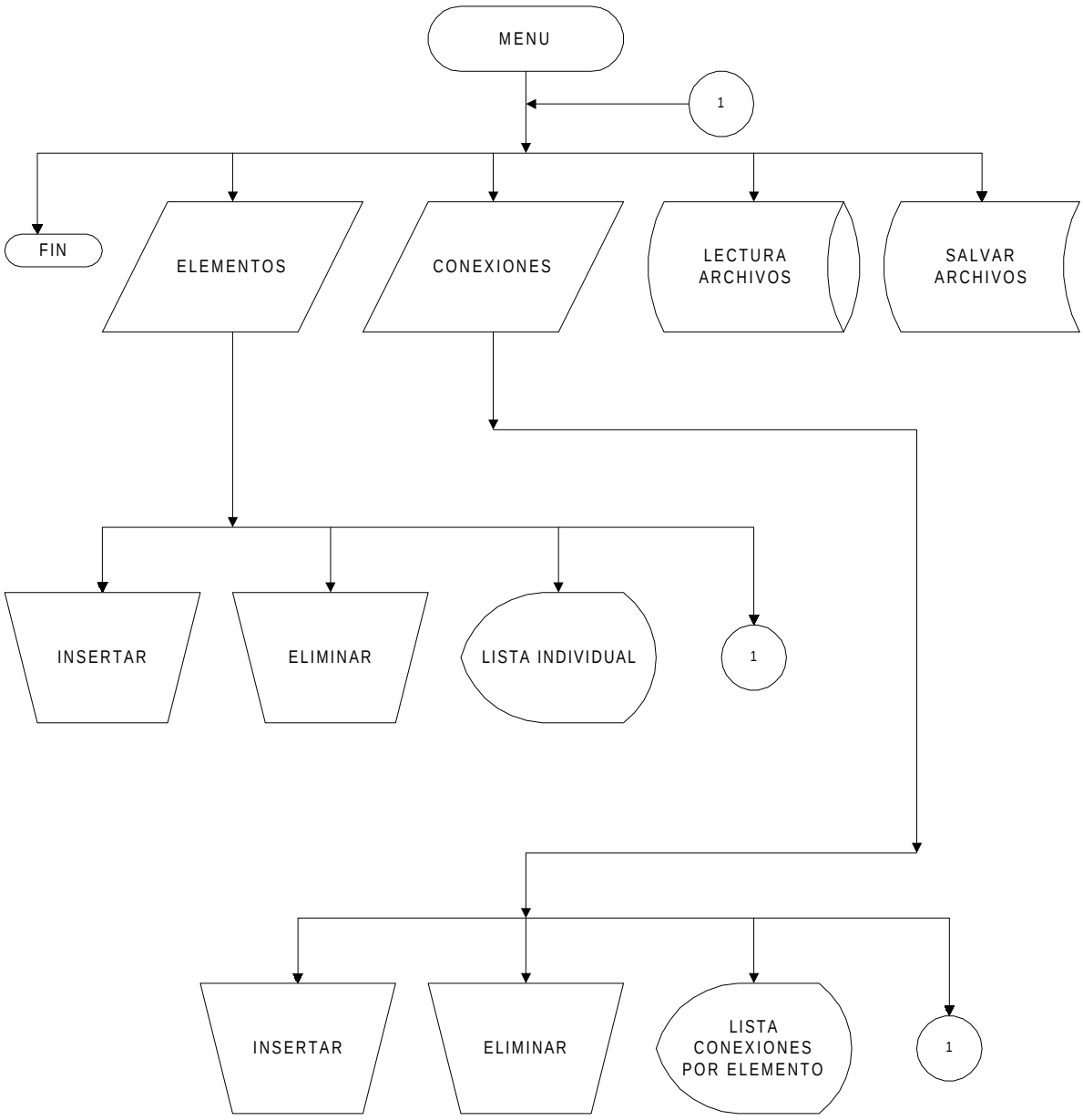

Figura 4-5. Diagrama de flujo para la obtención de la Malla Numérica.

4.2.5.- Algoritmo para la obtención de curvas de nivel

Una vez que se ha encontrado una malla de datos interpolados uniformemente espaciada como se explica en la sección anterior, se procede a localizar los contornos requeridos en cada celda. La figura 4-7 muestra los posibles contornos que pueden encontrarse en una celda. En base al planteamiento que se presenta en la figura 4-7, se puede establecer un algoritmo que pueda localizar y trazar curvas de nivel utilizando para este fin interpolación lineal.

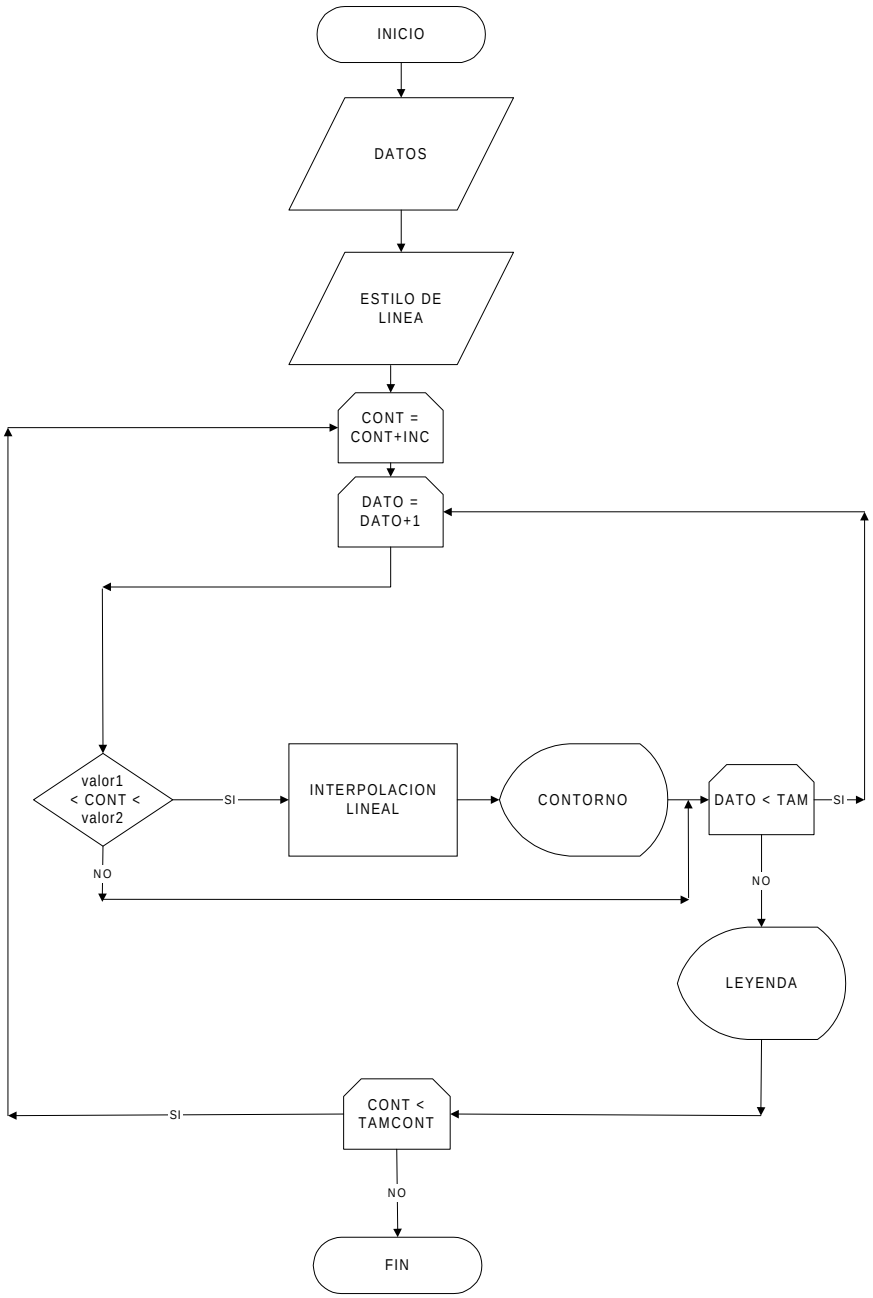

Figura 4-6. Diagrama de flujo para la obtención de curvas de nivel.

La figura 4-6 ejemplifica la solución en forma de diagrama de flujo; inicialmente se debe de tener una matriz de datos interpolados, cada nodo constará de tres valores, esto es, las distancias X, Y localizadas en cada columna-renglón de la malla, con su respectivo parámetro físico involucrado. Se define un estilo, color de línea, incremento de contorno y su valor de inicio. Se revisan todos los valores de la matriz utilizando una celda de cuatro nodos en cada análisis (ver figura 4-7) y se comprueba si existen contornos en dicha celda (A,B,C,D). Si este es el caso, se procede a interpolar linealmente para ir conformando la isolinea. Una vez realizada la búsqueda en toda la matriz de datos para el contorno definido, se comienza nuevamente con un nuevo contorno (cont = cont + inc), esta búsqueda no terminará mientras el valor de cont sea menor que cualquier dato contenido en la malla.

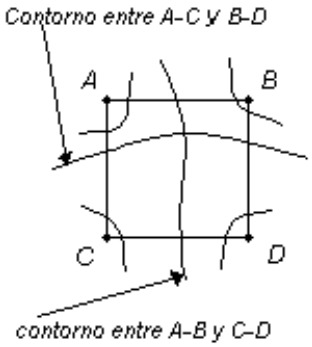

Figura 4-7. Localización de contornos en una celda con cuatro nodos.

La función que grafica las curvas de nivel es:

void contourplot(float (*)[], float, float,float,int);

La matriz de datos interpolados, el incremento, los valores iniciales y finales que tendrán los contornos son los argumentos del uno al cuatro, respectivamente. El último argumento da la opción de mostrar leyendas en cada contorno incluyendo el valor de este.