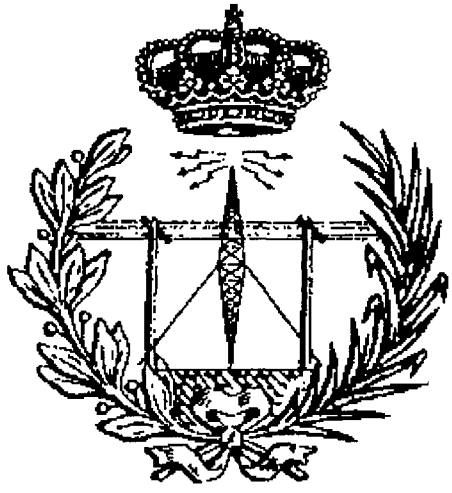
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA UNIVERSITARIA DE

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN



PROYECTO FIN DE CARRERA

SOFTWARE PARA LA SIMULACIÓN TÉRMICA

DE PLACAS DE CIRCUITO IMPRESO

TERMÍN 1.0

ESPECIALIDAD: EQUIPOS ELECTRÓNICOS

AUTOR: LUIS FELICIANO GUERRA GARCÍA

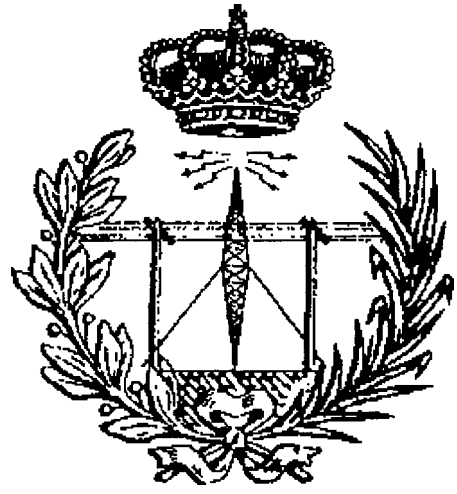
TUTOR: MANUEL FRANCISCO ENRÍQUEZ CHAVES

FECHA: JULIO 1999

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA UNIVERSITARIA DE

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN



PROYECTO FIN DE CARRERA

SOFTWARE PARA LA SIMULACIÓN TÉRMICA

DE PLACAS DE CIRCUITO IMPRESO

TERMÍN 1.0

AUTOR TUTOR

PRESIDENTE SECRETARIO VOCAL

CALIFICACIÓN

Matrícula de Honor - 10.

Termin 1.0
Simulador Térmico

Indice

INDICE

Termin 1.0
Simulador Térmico

Introducción

Introducción

I.1 PRESENTACIÓN

El diseño de circuitos electrónicos, tras la fase de cálculo, pasa por la fase de diseño en PCB (placa de circuitos impreso). Para ello se dispone de numerosas herramientas asistidas por ordenador que intentan realizar de una forma cómoda y rápida las diferentes fases por las que debe pasar hasta obtener el diseño definitivo: Captura de Netlist, emplazamiento de los componentes, implementación de las reglas de diseño, trazado de las pistas de interconexión. También es cierto que la mayoría de estos programas no tienen en cuenta la temperatura que disipará la distribución de todos los componentes en la placa. La mala distribución de estos componentes en la placa puede llevar al circuito diseñado a un mal funcionamiento e incluso a fallar. Es aquí donde verdaderamente cobran valor los programas de simulación térmica.

En función de lo dicho, nace la idea de realizar un programa simulador de temperatura de los componentes en la placa. La idea es crear un programa propio dentro de las limitaciones del calculo matemático, se tenga acceso al código fuente para poderlo adaptar y cambiar.

I.2 PETICIONARIO

Este proyecto ha sido realizado para la Universidad de Las Palmas de Gran Canaria presentado en la Escuela de Ingeniería de Telecomunicación en la especialidad de Equipos Electrónicos.

I.3 OBJETIVO GENERAL

El objetivo principal del proyecto es proporcionar a los alumnos de la Escuela Universitaria de Ingeniería Técnica de Telecomunicación, la posibilidad de comprobar la temperatura de las PCBs y de los componentes que sobre ella estén situados.

El usuario que utilice el programa tendrá varias opciones para poder desarrollar la simulación de la placa. Se requiere por tanto que la persona que utilice esta herramienta, tenga los conocimientos básicos sobre la disipación de calor de los componentes electrónicos.

La base principal en la que se respalda el programa para realizar los cálculos es el desarrollo de las ecuaciones diferenciales aplicadas al método RC.

I.4 DIVISIÓN DE LA MEMORIA

La estructura de la memoria está bien diferenciada en cuatro grandes apartados:

- 1º Aspectos generales en la disipación térmica en una placa de circuito impreso PCB (Capítulo 1º).
  - Aspectos históricos en la disipación de calor de los componentes en una placa de circuito impreso PCB.
  - Convección.
  - Radiación.
  - Radiación.
- 2º Desarrollo matemático de las ecuaciones de calor.
  - Ecuaciones de transferencia de calor (Capítulo 2º).
  - Coeficiente de transferencia (Capítulo 3º).
  - Método capacitancia resistiva RC (Capítulo 4º).
  - Potencia suministrada a la placa (Capítulo 5º).
- 3º Explicación detallada del funcionamiento del programa (Capítulo 6º).
  - Flujo gramas.
  - Matrices dinámicas.
  - Formatos de archivos utilizados.
  - Gradientes de color.
  - Ayuda del programa.
- 4º Aspectos formales del proyecto.
  - Pliego de condiciones (Capítulo 7º).
  - Instalación de Termín 1.0 (Capítulo 8º).
  - Presupuesto (Capítulo 9º).
  - Protección legal (Capítulo 10º).
  - Aspectos a mejorar (Capítulo 11º).

Termin 1.0
Simulador Térmico

Primera Parte

Breve historia en la disipación térmica en las PCB, formas de transmisión del calor y aspectos generales a tener en cuenta.

Termin 1.0
Simulador Térmico

Capítulo 1

Aspectos generales en la disipación térmica en una pcb

1 Aspectos generales en la disipación térmica en una PCB

1.1 ASPECTOS HISTÓRICOS EN CUANTO A LA DISIPACIÓN TÉRMICA DE LOS COMPONENTES

Los problemas térmicos que prevalecieron durante la era de las válvulas fueron eliminados durante un tiempo, con la llegada de los componentes de estado sólido debido a su baja potencia y su pequeña densidad de integración. Durante esa época la generación de calor era debida a los resistores de potencia y los problemas ocasionados por estos se solucionaba separando estos últimos una distancia prudente de los componentes semiconductores y colocando una adecuada refrigeración.

Con la llegada de los componentes microencapsulados, hizo que la densidad de integración fuese mucho mayor. Por otro lado surgen los circuitos integrados de alta velocidad, componentes que desarrollan cada vez más potencia tiristores, etc. Debido a todo lo anterior, a la hora del diseño de un proyecto es necesario un estudio térmico del conjunto placa, componentes y medio donde van a estar rodeado los componentes.

1.2 OBJETIVO PRINCIPAL DEL CONTROL TÉRMICO DE LOS COMPONENTES EN UNA PCB

El objetivo principal del estudio térmico de los componentes en una placa de circuito impreso PCB es mantener la temperatura por debajo del límite máximo establecido por el fabricante, normalmente se intenta mantener la temperatura máxima en la unión un porcentaje por debajo de su temperatura máxima.

1.3 FORMAS DE TRANSMISIÓN DEL CALOR EN LA PLACA

El calor se transmite mediante tres mecanismos en una placa: conducción, convección y radiación. A continuación se explican los aspectos generales de cada uno de ellos.

1.3.1 Conducción

Conducción calorífica es el término que se aplica al mecanismo de intercambio de energía interna de un cuerpo a otro, mediante el intercambio de la energía cinética debida al movimiento de las moléculas por comunicación directa o por el flujo de electrones libres. Este flujo de energía o de calor se

dirige desde las moléculas de energía más elevadas a las que tienen una cantidad menor, es decir, de la región de mayor a menor temperatura. La característica específica de la conducción es que tiene lugar dentro de los límites del cuerpo, o entre dos cuerpos que están en contacto, sin que se registre un desplazamiento de la materia que lo constituye.

Mediante un ejemplo simple de flujo de calor unidimensional, se podrá ver mejor:

Supóngase una muestra de superficie A de una pared de conductividad k y de longitud b, con la cara $x=0$ mantenida a la temperatura de T_f, y la cara $x=b$ mantenida a T_s, (ver figura 1.3.1). El flujo de calor por unidad de tiempo q a través de la pared es en la dirección de decrecimiento de temperatura, si $T_f > T_s$ es en la dirección positiva de x. La ley de Fourier de conducción de calor, que es como se denomina, para un cuerpo homogéneo, dice: Que el flujo de calor es proporcional al negativo de gradiente local de temperatura:

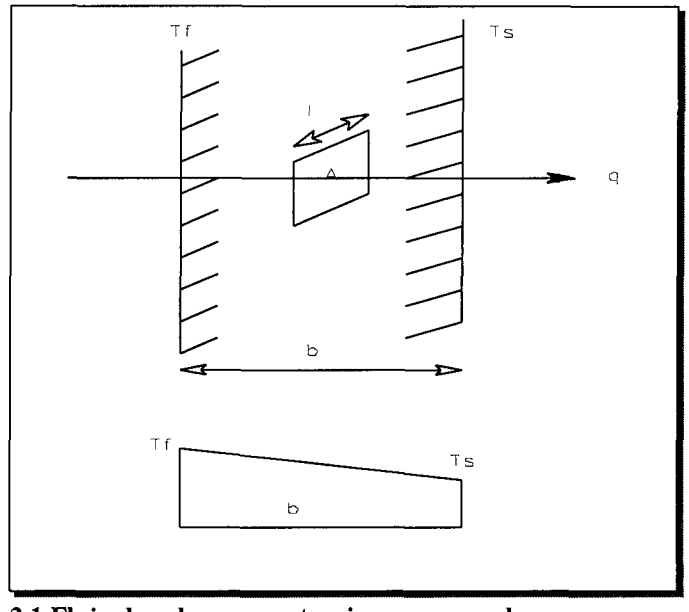


Fig. 1.3.1 Flujo de calor q que atraviesa una pared.

La cantidad de calor perdido por unidad de longitud en A, debido al flujo q de calor, ocasionado por la diferencia de temperatura es lo que llamamos potencia:

$$Q = \frac{\kappa}{b} A \left(T_f - T_s\right) \quad W \qquad (1.1)$$

El volumen de fluido que atraviesa la superficie en este caso unidad de área A, por unidad de tiempo o cantidad de calor que penetra en un área unidad (A) de superficie por unidad de tiempo, se conoce como flujo de calor:

$$\frac{Q}{A} = q = \frac{\kappa}{b}(T_f - T_i) \quad \frac{W}{m^2} \qquad (1.2)$$

Que de forma diferencial se queda:

$$qx = -k\frac{\partial T}{\partial x} \qquad (1.3)$$

Como es evidente, la conducción mejora con el parámetro *k*. Este parámetro depende del material a través del cual se transmite el calor. Utilizar materiales con mejor conductividad térmica es un recurso muy generalizado, aunque es difícil de conseguir.

1.3.2 Convección

Por convección designamos al mecanismo de transmisión calorífica que se produce en un fluido cuando una parte de éste se mezcla con otra a causa de los movimientos de la masa del mismo. En el capítulo 3 se entrará en mayor profundidad sobre la convección.

1.3.3 Radiación

La radiación térmica es el término que se emplea para describir la radiación electromagnética emitida por la superficie de un cuerpo excitado térmicamente. Esta radiación electromagnética se emite en todas las direcciones y, cuando incide sobre otro cuerpo, una parte de la misma puede ser reflejada, otras transmitida y otras absorbidas. Si la radiación incidente es de tipo térmico (es decir, si es de longitud de onda apropiada), la radiación absorbida aparecerá como calor en el cuerpo que la ha absorbido. Este mecanismo no se tiene en cuenta en el programa. La ecuación que rige este fenómeno es:

$$\frac{dQ}{dt} = E \cdot \sigma \cdot S \cdot (T_s^4 - T_{amb}^4) \quad (\mathrm{W}) \qquad (1.4)$$

- E es la potencia emisiva de un cuerpo negro, en W/m^2.
- σ es la constante de Stefan-Boltzmann ($5{,}67\ 10^{-8}\ W/m^2\ ^oC^4$).
- S es la superficie del objeto.

1.4 ASPECTOS AMBIENTALES QUE INFLUYEN EN EL MAPA TÉRMICO

Los aspectos que influyen en el análisis térmico de una placa son muchos, de los cuales aquí se van a mencionar unos cuantos:

- Dimensiones del contenedor y zonas en que está dividido.
- Material del que están compuesto las paredes del contenedor.
- Radiación solar incidente.
- Temperatura, dirección y velocidad del aire o fluido que rodea al contenedor.
- Naturaleza del aire que rodea la placa, en cuanto a su capacidad térmica y densidad.
- Naturaleza del material base de las PCBs y de las pistas de circuito impreso, así como su proporción en volumen (aspecto este que también se tiene en cuenta en el programa).
- Distribución y tipo de componentes a situar en las PCBs y la distribución de potencia en la placa (aspecto este que también se tiene en cuenta en la placa).

1.5 LOS FACTORES QUE INTERVIENEN EN LA OBTENCIÓN DE LA RESISTENCIA TÉRMICA DEL COMPONENTE

Los factores que intervienen en la obtención de la resistencia térmica de un componente son también bastantes, pasamos a definir algunos de ellos:

- Resistencia térmica en función de la potencia disipada. Cuanto mayor es la potencia disipada menor es la resistencia unión ambiente del componente
- Cuanto mayor es la superficie de la PCB en la que se encuentra el componente menor es la resistencia unión-ambiente de este.
- Cuanto mayor es la distancia entre el componente y la placa mayor es la resistencia unión-ambiente.
- La longitud de las pistas y el grosor influyen en la obtención de la resistencia del componente. Se puede decir que cuanto mas largas y más gruesas sean las pistas menor será la resistencia unión ambiente.
- Cuanto mayor es la superficie del encapsulado menor es la resistencia térmica unión ambiente del componente.
- El número de terminales del componente es un factor que interviene a la hora de obtener la resistencia térmica, actúa de forma inversamente proporcional al incremento de números de terminales, en cuanto a la resistencia térmica se refiere.
- El aire forzado es unos de los factores que más influye en la resistencia térmica unión-ambiente del componente, es evidente que a mayor velocidad del aire, menor será la resistencia.

  Además de lo anterior hay que añadir el posible radiador que puede tener el componente, de este se puede tener en cuenta los siguientes aspectos:

- Forma
- Material
- Acabado
- Color
- Ventilación

Termin 1.0
Simulador Térmico

Segunda Parte

Fundamentos matemáticos

Termin 1.0
Simulador Térmico

Capítulo 2

Ecuaciones de transferencia de calor

2 Ecuaciones de transferencia de calor

2.1 ECUACIONES QUE INTERVIENEN EN LA TRANSFERENCIA DE CALOR

2.1.1 Ecuaciones de conducción

La distribución de la temperatura en un medio cuya conductividad térmica es *k*, se satisface con la ecuación de Laplace para la conducción de calor estacionario.

$$\frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\kappa \frac{\partial T}{\partial y}\right)=0 \qquad (2.1)$$

Si hay fuentes de calor internas, hay que recurrir a la ecuación de Poisson:

$$\frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\kappa \frac{\partial T}{\partial y}\right)+q''=0 \qquad (2.2)$$

La variación de la temperatura con el tiempo en un medio cuya calor específico es *c* y densidad volumétrica ρ se satisface con la ecuación de Fourier:

$$\frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\kappa \frac{\partial T}{y}\right)+Q-\rho c \frac{\partial T}{\partial t}=0$$

$$\nabla \cdot \kappa \nabla T+Q=\rho c \frac{\partial T}{\partial t} \qquad (2.3)$$

2.1.2 Ecuaciones de conducción y convección

En presencia de un flujo de calor con las componentes de velocidad *u* y v en las direcciones *x* e *y* respectivamente, la ecuación de transferencia de calor puede ser descrito por la forma:

$$\rho c\left(u \frac{\partial T}{\partial x}+v \frac{\partial T}{\partial y}\right)-\left[\frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\kappa \frac{\partial T}{y}\right)\right]=0 \qquad (2.4)$$

Si las fuentes internas por unidad de volumen q''' son diferentes de cero, la ecuación de transferencia de calor puede escribirse de la forma:

$$\rho c \frac{\partial T}{\partial t} + \rho c\left(u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} \right) - \left[\frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y}\left(\kappa \frac{\partial T}{y} \right) \right] - q''' = 0 \quad (2.5)$$

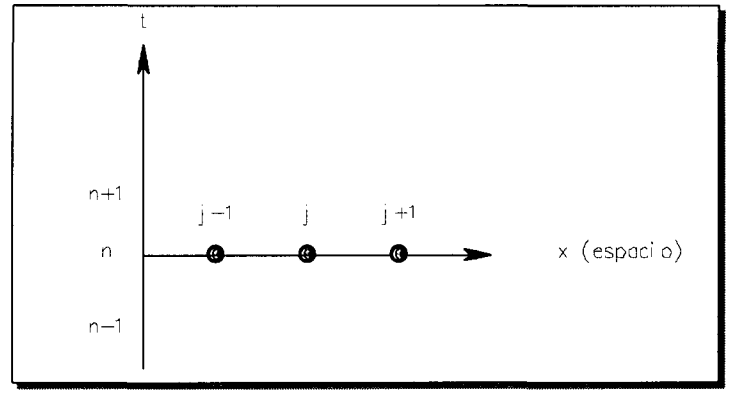
2.2 REPRESENTACIÓN DE LAS DERIVADAS POR DIFERENCIALES FINITAS

La sustitución de las derivadas de espacio por diferencias finitas se consigue mediante un desarrollo truncado de serie de Taylor. Por lo que la temperatura T alrededor de un punto x, en una dimensión, viene dada por:

$$T(x + h) = Tx + h\left(\frac{\partial T}{\partial x} \right) + \frac{h^2}{2}\left(\frac{\partial T^2}{\partial x^2} \right) + \frac{h^3}{6}\left(\frac{\partial T^3}{\partial x^3} \right) + \ldots \quad (2.6)$$

$$T(x - h) = Tx - h\left(\frac{\partial T}{\partial x} \right) + \frac{h^2}{2}\left(\frac{\partial T^2}{\partial x^2} \right) - \frac{h^3}{6}\left(\frac{\partial T^3}{\partial x^3} \right) + \ldots \quad (2.7)$$

Para tres puntos igualmente espaciados en que se designa por el punto j al del centro, es decir:



Si ahora se suman las dos ecuaciones anteriores en los términos que contienen h^3 se obtiene:

- ***Diferencial central para la segunda derivada***

$$\frac{\partial^2 u}{\partial x^2}\Big|_{j,n} = \frac{u^n_{j+1} + u^n_{j-1} - 2u^n_j}{\Delta x^2} + O(\Delta x^2) \quad (2.8)$$

El último término indica que el error que se produce es del orden de Δx^2.

Truncando la ecuación 2.6 después del termino h, se obtiene:

- ***Diferencial hacia delante para la primera derivada***

$$\frac{\partial u}{\partial x}\Big|_{j,n} = \frac{u_{j+1} - u_j^n}{\Delta x} + O(\Delta x) \qquad (2.9)$$

El último término indica que el error que se produce es del orden de Δx.

Truncando la ecuación 2.7 después del termino h, se obtiene:

- ***Diferencial hacia a tras para la primera derivada***

$$\frac{\partial u}{\partial x}\Big|_{j,n} = \frac{u_j - u_{j-1}^n}{\Delta x} + O(\Delta x) \qquad (2.10)$$

El último termino indica que el error que se produce es del orden de Δx.

Restando la ecuación 2.6 de la 2.7 y despreciado los términos h^3 y de orden mayor. Se obtiene:

- ***Diferencial centrada de la primera derivada***

$$\frac{\partial u}{\partial x}\Big|_{j,n} = \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + O(\Delta x^2) \qquad (2.11)$$

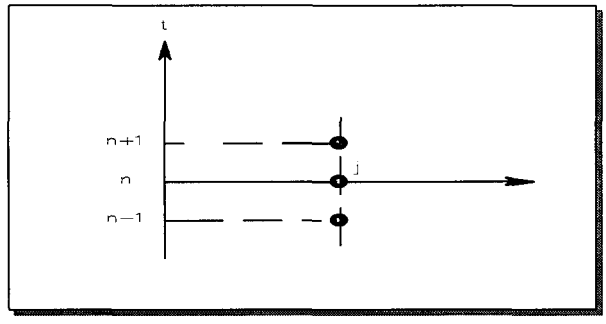
De la misma manera se puede obtener las diferenciales finitas en el tiempo.

- ***Diferencial hacia delante***

$$\frac{\partial u}{\partial t}\Big|_{j,n} = \frac{u_j^{n+1} - u_j^n}{\Delta t} = u_n' + \frac{\Delta t}{2} u_n'' + O(\Delta t^2) \qquad (2.12)$$

- ***Diferencial centrada***

$$\frac{\partial u}{\partial t}\Big|_{j,n} = \frac{u_j^{n+1} - u_j^{n-1}}{2\Delta t} \qquad (2.13)$$



2.3 APLICACIÓN DE LAS DIFERENCIALES FINITAS A LAS ECUACIONES DE TRANSFERENCIA DE CALOR

Si se aplica a la ecuación 2.3 la diferencial centrada anteriormente hablada, se obtiene:

$$\alpha = \frac{1}{cp}$$

$$\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} + \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{\Delta y^2} + \frac{Q}{\kappa} = \frac{\partial T_{i,j}}{\alpha \cdot \partial t} \quad (2.14)$$

De forma gráfica puede verse de la siguiente forma:

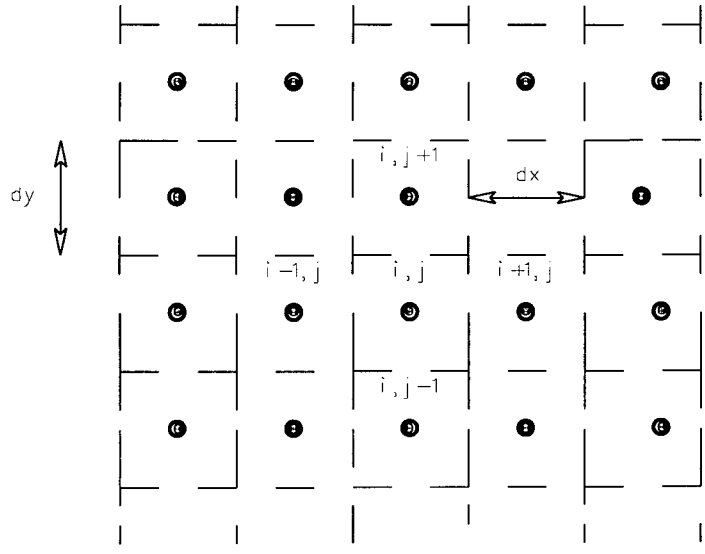


Fig. 2.3.1 Representación de los nodos de una malla.

Las áreas rectangulares mostradas comprenden una malla en la cual se subdivide el medio conductor total, caracterizado por propiedades tales como: $T_{i,j}$, $\rho_{i,j}$ y $c_{i,j}$. Los puntos centrales de los rectángulos, llamados nodos, también forman una malla.

En caso de considerar las divisiones de la malla iguales (dx=dy) la ecuación 2.14 se reduce a:

$$T_{i-1,j} - 2T_{i,j} + T_{i+1,j} + T_{i,j-1} - 2T_{i,j} + T_{i,j+1} + \frac{Q}{\kappa}\Delta x^2 = \frac{\Delta x^2 \cdot \partial T_{i,j}}{\alpha \cdot \partial t} \quad (2.15)$$

2.4 MÉTODO IMPLÍCITO Y EXPLÍCITO

2.4.1 Método explícito

Un sistema numérico explícito es aquel mediante el cual se puede obtener directamente la cantidad desconocida.

Aplicando a la ecuación 2.15 la primera diferencial hacia adelante se obtiene:

$$T^{n}_{i-1,j} - 2T^{n}_{i,j} + T^{n}_{i+1,j} + T^{n}_{i,j-1} - 2T^{n}_{i,j} + T^{n}_{i,j+1} + \frac{q}{\kappa}\Delta x^{2} = \frac{\Delta x^{2}}{\alpha \cdot \Delta t}\left(T^{n+1}_{i,j} - T^{n}_{i,j}\right) \quad (2.16)$$

Esta es la representación típica de la expresión bidimensional de la conducción transitoria con generación interna de calor.

Una solución para T_{ij} para el n+1 paso, requiere inicialmente de una distribución conocida de temperatura, donde n=0. Esta puede ser conocida, o se puede resolver la distribución de estado estable con condiciones de frontera, haciendo igual a cero la derivada del tiempo en la ecuación 2.15. Esta operación, da la forma de la diferencia finita bidimensional de la ecuación de Poisson, es decir la ecuación 2.2, que aplicando las diferencias finitas se transforma en:

No obstante para la mayoría de los problemas se conocen las condiciones

$$T_{i-1,j} - 2T_{i,j} + T_{i+1,j} + T_{i,j-1} - 2T_{i,j} + T_{i,j+1} + \frac{q}{\kappa}\Delta x^{2} = 0 \qquad (2.17)$$

iniciales. Una técnica para resolver un problema de estado estable es suponer una distribución inicial arbitraria de temperatura y desarrollar una solución transitoria hasta que T_{ij} en el paso n+1 sea igual a $T_{i,j}$ en el paso n, pero con una cierta tolerancia de temperatura.

2.4.1.1 Limitaciones de estabilidad en cuanto al método explícito

Si se intenta resolver la ecuación 2.16 para obtener la solución puede verse que la solución de T_{ij} para en el paso $n+1$ es en realidad, el valor anterior para el paso n, más una cierta cantidad, es decir:

$$T^{n+1}_{i,j} = T^{n}_{i,j} + \frac{\alpha \cdot \Delta t}{\Delta x^{2}}\left(T^{n}_{i-1,j} - 2T^{n}_{i,j} + T^{n}_{i+1,j} + T^{n}_{i,j-1} - 2T^{n}_{i,j} + T^{n}_{i,j+1}\right) + \frac{\alpha \cdot q \cdot \Delta t}{\kappa} \quad (2.18)$$

Lo que se comporta como una serie. Debido a eso se dice que la solución debe ser convergente, y al estar descompuestas en diferenciales finitas se dice que debe ser estable. De hecho la estabilidad en una serie garantiza la convergencia de la serie, porque la estabilidad es una condición más fuerte. Para calcular la estabilidad se usa el análisis de estabilidad de Von Neumann, el cual viene explicado en el anexo.

Si se aplica el criterio de estabilidad de Von Neumann a la ecuación 2.18 se llega a la expresión:

$$\frac{4\alpha \cdot \Delta t}{\Delta x^2} \leq 1$$

$$\Delta t \leq \frac{\Delta x^2}{4\alpha}$$

el limite sería :

$$\Delta t = \frac{\Delta x^2}{4\alpha} \qquad (2.19)$$

La interpretación física de la restricción anterior es que el máximo tiempo permitido entre en paso n y el n+1 está limitada a la ecuación 2.19.

Otro forma de hallar la estabilidad (la que se usa normalmente) es saber que la ecuación 2.18 nunca puede ser negativa, de esta forma sale una restricción un poco más fuerte, pero mucho más fácil de hallar que la de Von Neumann, que en este caso es:

$$T_{i,j}^{n+1} = T_{i,j}^{n} + \frac{\alpha \cdot \Delta t}{\Delta x^2}\left(T_{i-1,j}^{n} + T_{i+1,j}^{n} + T_{i,j-1}^{n} + T_{i,j+1}^{n} - 4T_{i,j}^{n}\right) + \frac{\alpha \cdot q \cdot \Delta t}{\kappa} \Rightarrow$$

$$T_{i,j}^{n+1} = T_{i,j}^{n}\left(1 - 4\frac{\alpha \cdot \Delta t}{\Delta x^2}\right) + \frac{\alpha \cdot \Delta t}{\Delta x^2}\left(T_{i-1,j}^{n} + T_{i+1,j}^{n} + T_{i,j-1}^{n} + T_{i,j+1}^{n}\right) + \frac{\alpha \cdot q \cdot \Delta t}{\kappa}$$

La única restricción para que el término anterior no sea negativo es que:

$$1 - 4\frac{\alpha \cdot \Delta t}{\Delta x^2} \geq 0$$

$$\Delta t \leq \frac{\Delta x^2}{4\alpha}$$

en el limite sería :

$$\Delta t = \frac{\Delta x^2}{4\alpha}$$

Dando igual que antes, pero normalmente no lo dará.

Si se aplica la ecuación 2.5 de las diferencias finitas anteriormente estudiadas llegamos a:

$$T_{i,j}^{n+1} = \left(\frac{u_{i,j}\Delta t}{2\Delta x} + \frac{\alpha\Delta t}{\Delta x^2}\right)T_{i-1,j}^{n} + \left(\frac{v_{i,j}\Delta t}{2\Delta x} + \frac{\alpha\Delta t}{\Delta x^2}\right)T_{i,j-1}^{n}$$
$$+ \left(1 - \frac{2\alpha\Delta t}{\Delta x^2} - \frac{2\alpha\Delta t}{\Delta x^2}\right)T_{i,j}^{n} + \left(-\frac{u_{i,j}\Delta t}{2\Delta x} + \frac{\alpha\Delta t}{\Delta x^2}\right)T_{i+1,j} +$$
$$\left(-\frac{v_{i,j}\Delta t}{2\Delta x} + \frac{\alpha\Delta t}{\Delta x^2}\right)T_{i,j+1} \quad (2.20)$$

Donde $u_{i,j}$ y $v_{i,j}$ denotan las componentes de velocidad en el punto (i,j). Para que la ecuación 2.20 sea estable debe cumplir el criterio de estabilidad anterior, es decir, que sólo aparezcan coeficientes positivos, pero ahora la restricción es en cuanto al tiempo y a las dimensiones de la malla. Dichas restricciones son las siguientes:

$$\Delta t \le \left[\frac{4\alpha}{\Delta x^2}\right]^{-1} \quad (2.21)$$

$$\frac{|u_{i,j}|\Delta x}{\alpha} \le 2 \qquad \frac{|v_{i,j}|\Delta x}{\alpha} \le 2 \quad (2.22)$$

Si se supone que los módulos de las velocidades son iguales, puede verse con un ejemplo como serán las restricciones de tiempo y tamaño de malla para unas constantes dadas, y como la placa se comporta en cierto modo como un aislante, las constantes se ponen en relación aun aislante, a saber:

$c = 1{,}59$	$\frac{\text{KJ}}{\text{Kg} \cdot °\text{C}}$	➢ Calor específico
$p = 1273$		➢ Densidad volumétrica
$\kappa = 0{,}232$	$\frac{W}{m \cdot °C}$	➢ Conductividad
$\alpha = \frac{k}{cp} = 1{,}14 \times 10^{-4}$	$\frac{W \cdot s}{m^3 \cdot °C}$	
$V = 40$	$\frac{m}{s}$	➢ Velocidad del viento

Sustituyendo los datos anteriores en la ecuación 2.22, da una división de malla de:

$$\Delta x \le 0.0057 \quad mm$$
$$\Delta t \le 71.25 \times 10^{-9}\, s$$

De las cantidades anteriores se puede ver que las restricciones para la forma explícita son muy severas, imposibles de alcanzar de esta forma, siendo a aun peor si se fija un cuadricula Δx de malla, de orden de milímetros, no obstante se verá más adelante, que para la forma implícita, sólo tiene restricción respecto a la división de la malla y no en cuanto al tiempo. También en el apartado 3.2.1 se explicará como poder introducir ese flujo exterior debido a una ventilación forzada o no forzada a través del llamado *coeficiente de transferencia,* para poder utilizar la forma explícita sin la restricción de la división de la malla.

2.4.2 Método implícito

De forma implícita la incógnita aparece en una serie de ecuaciones que se deben resolver simultáneamente. Aplicando a la ecuación 2.15 la primera diferencial hacia a delante se obtiene:

$$T_{i-1,j}^{n+1} - 2T_{i,j}^{n+1} + T_{i+1,j}^{n+1} + T_{i,j-1}^{n+1} - 2T_{i,j}^{n+1} + T_{i,j+1}^{n+1} + \frac{q}{\kappa}\Delta x^2$$

$$= \frac{\Delta x^2}{\alpha \cdot \Delta t}\left(T_{i,j}^{n+1} - T_{i,j}^{n}\right) \quad (2.23)$$

No se puede despejar directamente la incógnita $T_{i,j}$ en el $n+1$ paso, de ahí le viene el nombre de implícito.

La solución se obtiene resolviendo simultáneamente el conjunto de ecuaciones algebraicas que se producen en un incremento $n+1$ de tiempo. Hay que recordar que la ecuación no es la única en este proceso, ya que los nodos de los bordes y los nodos de las esquinas tienen ecuaciones diferentes.

Veamos como es la matriz que sale de la ecuación anterior, sin incluir nodos de bordes y esquinas, suponiendo la siguiente distribución de nodos:

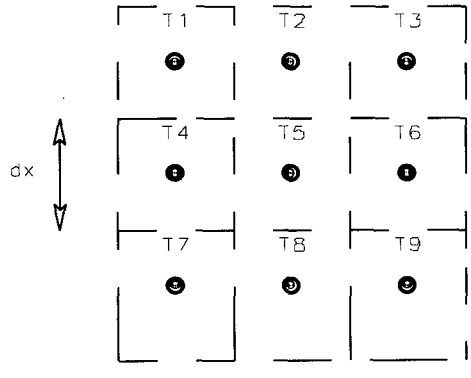


Fig 2.4.2.1 Malla de nodos utilizada para el ejemplo del método implícito.

Según la matriz de nodo de la figura 2.4.2.1 y aplicando la ecuación (2.23), se llega a la siguiente matriz:

	1	2	3	4	5	6	7	8	9
1	− 4	1	0	1	0	0	0	0	0
2	1	− 4	1	0	1	0	0	0	0
3	0	1	− 4	0	0	1	0	0	0
4	1	0	0	− 4	1	0	1	0	0
5	0	1	0	1	− 4	1	0	1	0
6	0	0	1	0	1	− 4	0	0	1
7	0	0	0	1	0	0	− 4	1	0
8	0	0	0	0	1	0	1	− 4	1
9	0	0	0	0	0	1	0	1	− 4

Fig. 4.2.2.1 Matriz de los nodos centrales de la ecuación 2.23.

2.4.2.1 Limitaciones de la matriz característica del método implícito

Si intentamos hallar la matriz característica pero esta vez incluyendo las ecuaciones de los nodos laterales y extremos, ecuaciones que explicadas en el capítulo 4, sale una matriz que no es triangular, lo que no se puede utilizar la descomposición LU, es más, ni siquiera es simétrica. Para poder realizar los cálculos de esta forma implícita en dos dimensiones hay que recurrir a algoritmos de resolución de sistemas de ecuaciones muy complejos y difíciles de conseguir, no obstante usando esta forma ya no hay restricción de tiempo, debido a que todas las incógnitas se realizan a vez. La restricción de la división de la malla (dx) aún persiste, es decir que hay que resolver el sistema con divisiones de malla muy chicas del orden de centésimas de milímetros, el método que se utiliza para salvar esta dura restricción es el *método de los elementos finitos* aplicados a la transferencia de calor.

Este método es muy complejo, básicamente está basado en la elección de unas funciones de interpolación y funciones de forma. Los primeros son polinomios que se usan para obtener una solución aproximada, para después obtener mediante el método de los residuos un conjunto de funciones arbitrarias, que con junto con las funciones de forma(normalmente triángulos) se obtiene la matriz de conductividad y la matriz de carga, la explicación de este método esta más detallada en el anexo.

Termin 1.0
Simulador Térmico

Capítulo 3

Coeficiente de transferencia

3 Coeficiente de transferencia

3.1 EL COEFICIENTE DE TRANSFERENCIA Y SU SIGNIFICADO

Normalmente la convección ocurre por un cambio de calor entre dos masas que están a diferente temperatura y fue Newton quien observó por primera vez que el calor transferido por convección era un una función, que comprendía la diferencia de temperatura de las dos masas multiplicado por un coeficiente llamado: *coeficiente de transferencia,* es decir:

$$\text{calor perdido} = h\left(T_s - T_f\right) \qquad (3.1)$$

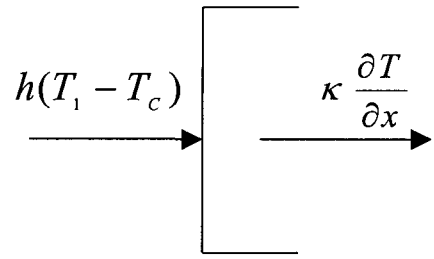
Por lo que el coeficiente de transferencia esta muy relacionado con la resistencia térmica de esas dos masas (aire y placa) a diferente temperatura, de hecho la resistencia térmica entre el aire que rodea la placa y esta es:

$$Rth = \frac{1}{hA} \quad \frac{°C}{W} \qquad (3.2)$$

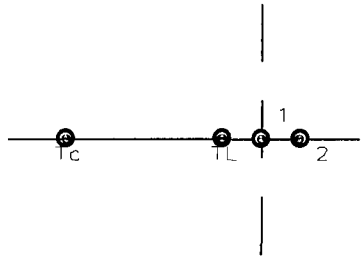
En principio este cambio/perdida de calor se produce en el contorno de la superficie, pero se verá más adelante que al hablar de una velocidad de viento, tanto forzado como natural existe un coeficiente de transferencia en toda la placa no solo en sus bordes. Para relacionar la ecuación anterior con la perdida de calor se tiene en cuenta las ecuaciones del capitulo 1, sobre el flujo de calor, es decir:

$$qx = -k\frac{\partial T}{\partial x} \qquad (3.3)$$

Como q_x es el flujo de calor que atraviesa la superficie, debe coincidir con el calor perdido, es decir, que la cantidad de calor que deja la superficie debe ser igual a flujo de calor en la superficie, y por lo tanto:



$$-\kappa \frac{\partial T}{\partial x} = -h(T_1 - T_C) \qquad (3.4)$$



El signo negativo es porque el flujo va en dirección contraria al gradiente, en el lado derecho es negativo porque el calor se pierde en sentido contrario del eje de la x.

Aplicando la primera diferencial centrada del capítulo 2, queda:

$$-\kappa \frac{T_2 - T_L}{2\Delta x} = -h(T_1 - T_C)$$

$$T_L = \frac{2h\Delta x}{\kappa} T_1 - \frac{2h\Delta x}{\kappa} T_C - T_2 \qquad (3.5)$$

T_L = temperatura del aire al borde derecho de la placa
T_C = temperatura del aire constante lejos de la placa

Que al aplicarla en la ecuación 2.18 en los términos $T_{l,n}$ se queda como:

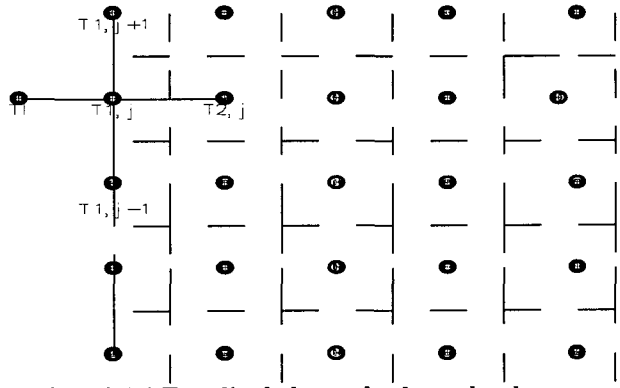


Fig. 3.1.1 Estudio de los nodos laterales de una malla.

$$T_{1,j}^{n+1} = T_{1,j}^{n}\left(1 - \frac{\alpha \cdot \Delta t}{\Delta x^2}\left(4 + \frac{2h\Delta x}{k}\right)\right) + \frac{\alpha \cdot \Delta t}{\Delta x^2}$$
$$\left(2T_{2,j}^{n} + \frac{2h\Delta x}{k}T_C^{n} + T_{1,j-1}^{n} + T_{1,j+1}^{n}\right) + \frac{\alpha \cdot q \cdot \Delta t}{\kappa} \quad (3.6)$$

También hay que hacerlo al extremo derecho, superior y inferior, sin olvidar las esquinas izquierdas y derechas; dando otras ecuaciones parecidas a la (3.6) Dichas ecuaciones se obtienen mediante el método de resistencia capacitancia del capitulo 4. No obstante mirando la ecuación (3.6) se observa que la restricción de tiempo ya no es la misma, sino que ha empeorado, de tal forma que la restricción para los puntos del extremo derecho ahora es:

$$1 - \frac{\alpha \cdot \Delta t}{\Delta x^2}\left(4 + \frac{2h\Delta x}{k}\right) \geq 0$$
$$\Delta t \leq \frac{pc}{\left(\frac{4\kappa}{\Delta x^2} + \frac{2h}{\Delta x}\right)}$$

En el limite sería:

$$\Delta t = \frac{pc}{\left(\frac{4\kappa}{\Delta x^2} + \frac{2h}{\Delta x}\right)} \quad (3.7)$$

Lo que el incremento del tiempo en cada iteración es aun menor que la restricción de la ecuación 2.19.

3.2 COMO OBTENER EL COEFICIENTE DE TRANSFERENCIA

La obtención del coeficiente de transferencia es muy variada y complicada, hay infinidad de estudios para conseguir obtener este valor para diferentes materiales y condiciones ambientales (viento, dimensiones del material...). En el anexo se encuentra el estudio resumido del libro *transferencia de calor* de *Alan J. Chapman*, sobre como obtener el coeficiente de transferencia.

Coeficiente de transferencia (h) depende en primer lugar si el viento circundante que rodea el material es forzado o no, es decir, si es una convección forzada o natural.

Si se aplica un flujo laminar (ver a apartado 3.2.1) a lo largo de un sólido puede demostrarse que *h* está relacionado con el número de Nusselt (Nn), que a su vez es función del número de Prandtl (Pr) y del número de Grashof (Gr), es decir:

$$h_L = \frac{\kappa Nu_L}{L} \quad \text{c.t. medio} \qquad \frac{W}{m^2\,^oC} \qquad (3.8)$$

$$h_x = \frac{\kappa Nu_x}{x} \quad \text{c.t. local} \qquad \frac{W}{m^2\,^oC} \qquad (3.9)$$

Donde L es la longitud de la placa y x es la coordenada x de la placa.

$$Nu_L = \frac{4}{3}\cdot\frac{1}{\sqrt{2}} Gr_L^{1/4} f(Pr)$$

$$Nu_x = \frac{1}{\sqrt{2}} Gr_x^{1/4} f(Pr)$$

$$Gr = \frac{g\beta(T_f - T_s)L^3}{\nu^2}$$

$$Pr = \nu\frac{\rho c_p}{\kappa} = \frac{\mu c_p}{\kappa} \qquad (3.10)$$

κ	$\frac{W}{m\,^oC}$	➢ Conductividad
β	^oC	➢ Coeficiente volumétrico
g	m/s^2	➢ Gravedad
	$\frac{m^2}{s}$	➢ Viscosidad cinemática
c	$\frac{KJ}{Kg\cdot\,^oC}$	➢ Calor específico
μ	$\frac{Kg}{m\cdot s}$	➢ Viscosidad dinámica
L	m	➢ Longitud de la placa

Se puede ver lo complicado que resulta obtener el coeficiente de transferencia de un material. Pero es aquí donde entra en función los estudios y experimentos que se han hecho para obtener ciertas relaciones entre los números anteriores llamadas *Correlaciones.*

3.2.1 El coeficiente de transferencia en convección forzada

El flujo puede ser forzado, como en el caso de un líquido bombeado en un tubo o aire en el vehículo propulsado en la atmósfera. El flujo natural o libre se produce por la fuerza ascendente resultante de la diferencial de la densidad, como en el caso de una torre de refrigeración de intercambio normal. Tanto el flujo forzado como el natural puede ser laminar o turbulento. El flujo laminar predomina en las bajas velocidades, para pequeños tamaños y para los fluidos más viscosos. La Transferencia de calor tiende a ser mucho más grande en flujos turbulentos que en flujos laminares, debido a la mezcla vigorosa del fluido.

La relación de transferencia de calor por convección es generalmente una función compleja de la geometría de la superficie y temperatura, la temperatura y velocidad del fluido, y propiedades termofísicas del fluido.

Dependiendo de que sea un flujo laminar o turbulento se tendrá unas correlaciones o tras.

En el programa sólo se aplica las correlaciones que a continuación se van a dar, recordando que hay infinidades de correlaciones para diferentes situaciones de contorno y ambientales. Las correlaciones están sacadas del libro *Transmisión del calor* de *Alan J. Chapman.*

Por lo que respecta al flujo laminar a lo largo de una superficie plana, se ha comprobado experimentalmente que las correlaciones siguientes dan muy buen resultado:

$$Nu_x = 0.332 \cdot \mathrm{Re}_x^{\frac{1}{2}} \cdot \mathrm{Pr}^{\frac{1}{3}} \qquad (3.11)$$

$$Nu_L = 0.664 \cdot \mathrm{Re}_L^{\frac{1}{2}} \cdot \mathrm{Pr}^{\frac{1}{3}} \qquad (3.12)$$

$$0.6 \leq \mathrm{Pr} \leq 50$$

$$\mathrm{Re} < \mathrm{Re}_x \approx 5 \times 10^5$$

Propiedades a temperatura media (T_m)

Re_x y Re_L representan respectivamente el número de Reynol local y medio, el cual viene definido de la forma:

$$\mathrm{Re}_x = \frac{V_x \cdot l_x \cdot \rho_x}{\mu_x} = \frac{V_x \cdot l_x}{\nu_x} \qquad (3.13)$$

$$\mathrm{Re}_L = \frac{V \cdot l \cdot \rho}{\mu} = \frac{V \cdot l}{\nu} \qquad (3.14)$$

$$T_m = \frac{Tamb. + T_x}{2}$$

T_x	C°	➢	Temperatura de la placa en la coordenada x
ρ	$\frac{Kg}{m^3}$	➢	Densidad volumétrica
$c = c_p$	$\frac{KJ}{Kg \cdot °C}$	➢	Calor específico
μ	$\frac{Kg}{m \cdot s}$	➢	Viscosidad dinámica
	$\frac{m^2}{s}$	➢	Viscosidad cinemática
V	$\frac{m}{s}$	➢	Velocidad viento

Los valores de ρ, c, etc, se obtienen de las características del aire seco. En el programa estas tablas están en un archivo llamado *aire.txt* que se leen y se pasa a una lista para mayor rapidez.

Tras haber obtenido el valor de Nusselt este se aplica a la ecuación 3.8 o 3.9 según se halla obtenido la correlación para valor medio o local. A partir de aquí tenemos el valor de h (coeficiente de transferencia), que habrá que obtenerlo de nuevo para cada estado n que se haga en el tiempo.

Para un flujo turbulento a lo largo de una superficie plana, se ha comprobado experimentalmente que las correlaciones siguientes dan muy buen resultado:

$$Nu_x = 0.0296\ \mathrm{Re}_x^{0.8}\ \mathrm{Pr}^{1/3} \qquad 5\times10^5 < \mathrm{Re}_x < 10^7 \qquad (3.15)$$

$$Nu_x = 0.185\ \mathrm{Re}_x\left(\log_{10}\mathrm{Re}_x\right)^{-2.584}\ \mathrm{Pr}^{1/3} \qquad \mathrm{Re}_x > 10^7 \qquad (3.16)$$

$$0.6 \le \mathrm{Pr} \le 60$$

- Propiedades a Tm (temperatura media)

Se sustituye este valor en las ecuaciones 3.13 y 3.14 para obtener el valor de Nusselt, y este aplicarlo de nuevo en las ecuaciones 3.8 y 3.9 para obtener el coeficiente de transferencia.

3.2.2 El coeficiente de transferencia en convección natural

La convección libre alrededor de placas horizontales es muy distinta al anterior. En la convección forzada la fuerza principal de empuje era la fuente de viento exterior (ventiladoe o algún otro elemento de refrigereación de inyección de aire por canalización o no). Ahora, es la gravedad la fuerza principal y la dirección del cuerpo, en este caso la placa es normal a la dirección de esta última fuerza.

A lo largo de una superficie plana, y con convección natural se ha comprobado experimentalmente mediente estudios en laboratorios con diferentes situaciones de temperatura y condiciones medio ambientales que las correlaciones siguientes dan muy buen resultado:

$$NL_c = 0.27 \cdot RaL^{\frac{1}{4}} \qquad 3\cdot10^5 < RaL < 3\cdot10^{10} \qquad (3.8)$$

$$L_C = \frac{area\ \ de\ \ la\ \ placa}{perímetro\ \ \ de\ \ la\ \ placa}$$

$$RaL = Gr \cdot \mathrm{Pr} = \frac{L_c^3 \cdot g \cdot \beta \cdot \Delta T}{\nu^2}$$

- ΔT ➢ Diferencia de temperatura entre placa y aire
- β ➢ Coeficiente de dilatación térmica para el gas (aire) 1/293.15
- RaL ➢ Valor de Reynol modificado

Cuando la correlación anterior se salga fuera de los márgenes de *RaL* se usa la aproximación de:

$$h = \left(\frac{\Delta T}{l} \right)^{\frac{1}{4}} \quad \frac{\mathrm{W}}{\mathrm{m}^2 \, ^\circ C}$$

- ➢ Donde l es la longitud de la placa.

Termin 1.0
Simulador Térmico

Capítulo 4

Método capacitancia resistiva RC

4 Método capacitancia resistiva (RC)

4.1 EL MÉTODO DE CAPACITANCIA RESITIVA Y SUS APLICACIONES

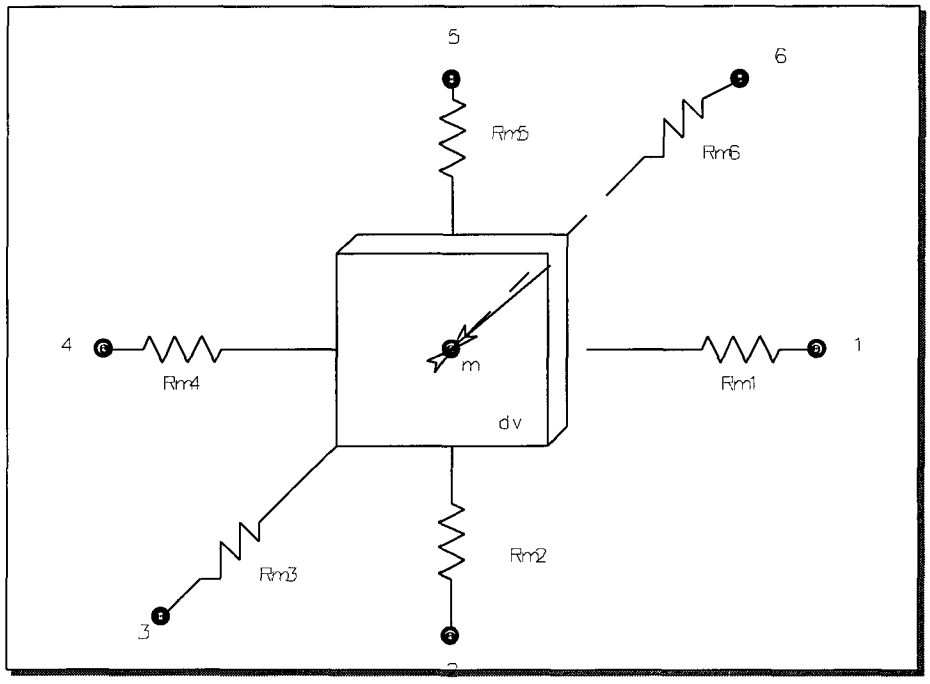


Fig. 4.1 Resistencia térmicas del nodo de una malla.

Una alternativa útil a las formulaciones para los métodos numéricos de diferencias finitas que se presentaron en el capítulo 2 es la formulación, en términos de un circuito, de la capacitancia resistiva (RC). Considerando la conducción no estacionaria multidimensional con generación de calor interna y propiedades térmicas variables. La figura 4.1 muestra el nodo de interés m, rodeado por el volumen ΔV_m de capacitancia térmica:

$$C_m = \rho_m c_m \Delta V_m$$

Los nodos que lo rodean se denotan por n y la conducción del volumen ΔV_n al volumen ΔV_m se expresa en función de una resistencia térmica R_{mn}. El balance de energía sobre el volumen ΔV_m durante el intervalo de tiempo Δt da:

$$C_m \frac{T_m^{i+1} - T_m^i}{\Delta t} = \sum_n \frac{T_n^i - T_m^i}{R_{mn}} + Q'''\Delta V_m + Q''\Delta S_m \qquad (4.1)$$

Donde el supraíndice se refiere al paso temporal, como antes. La ecuación 4.1 es una formulación explícita. Aunque está escrita para un nodo interior, esta ecuación también se puede aplicar a nodos de contorno con resistencias R_{mn} que deben evaluarse de manera adecuada. Obsérvese que cuando las propiedades dependen de la temperatura, la capacitancia C_m, las resistencias R_{mn} y la fuente de calor Qv''' y Qs'' deben evaluarse en el instante i para validar la forma explícita. Si se desea obtener una formulación implícita, los potenciales (T_n-T_m) deben evaluarse en el instante $i+1$. Sin embargo, para obtener una formulación totalmente implícita, C_m $R_{m,n}$ y Qv''' deben calcularse en el instante $i+1$, lo que puede hacer más lento el proceso de iteración. Así, en general la formulación explícita es preferible cuando la formulación se usa en función de circuitos RC.

Despejando T_m en el paso $i+1$ de la ecuación 4.1 se obtiene:

$$Cm = c \times p \times \Delta y \times \Delta x \times \Delta z$$

$$T_m^{i+1} = T_m^i \left(1 - \frac{\Delta t}{Cm} \sum_n \frac{1}{Rmn}\right) + \frac{\Delta t}{Cm}\left(Q_v'''\Delta Vm + Q_s''\Delta S_m + \sum_n \frac{T^i{}_n}{R_{nm}}\right) \qquad (4.2)$$

Una vez más una condición de estabilidad adecuada es que el coeficiente de T_m en el paso i no sea negativo:

$$1 - \frac{\Delta t}{C_m} \sum_n \frac{1}{Rmn} \geq 0 \qquad (4.3)$$

La principal ventaja de la formulación RC es la versatilidad con que se puede implementar mediante lenguajes de programación como el C++, Delphi...

4.2 OBTENCIÓN DE LOS DIFERENTES PUNTOS DE LA MALLA CON EL MÉTODO CAPACITANCIA RESISTIVA

4.2.1 Puntos dentro de la malla

El esquema de figura 4.2.1.1 representa los nodos de un punto central de la malla, cada letra corresponde a la temperatura de un nodo.

Las resistencias de *Raf* y *Rag* de la figura 4.2.1.1 son las resistencias de la placa con el medio ambiente. Las restantes resistencias térmicas son de conducción.

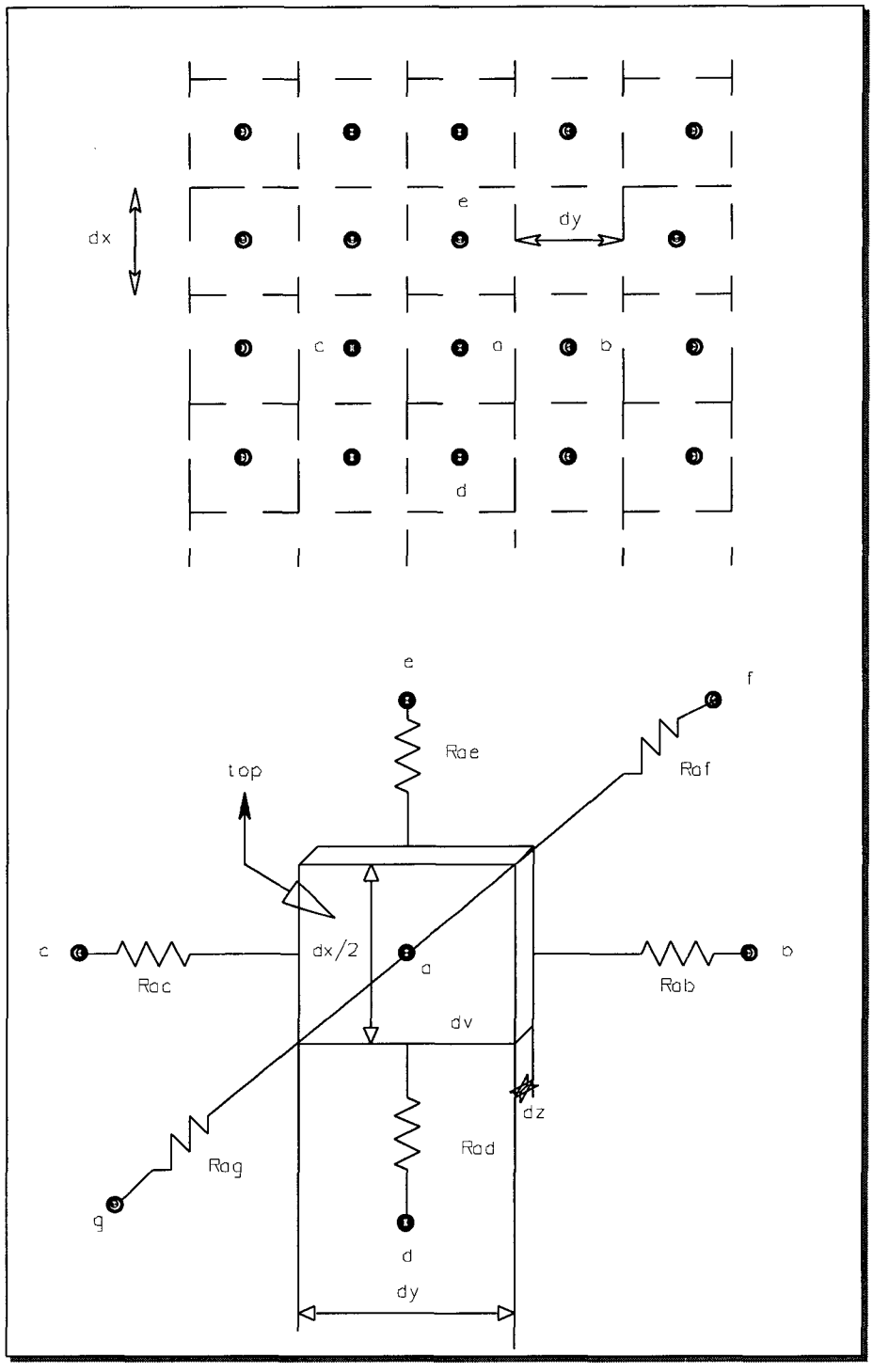


Fig. 4.2.1.1 Estudio de los nodos centrales de la malla.

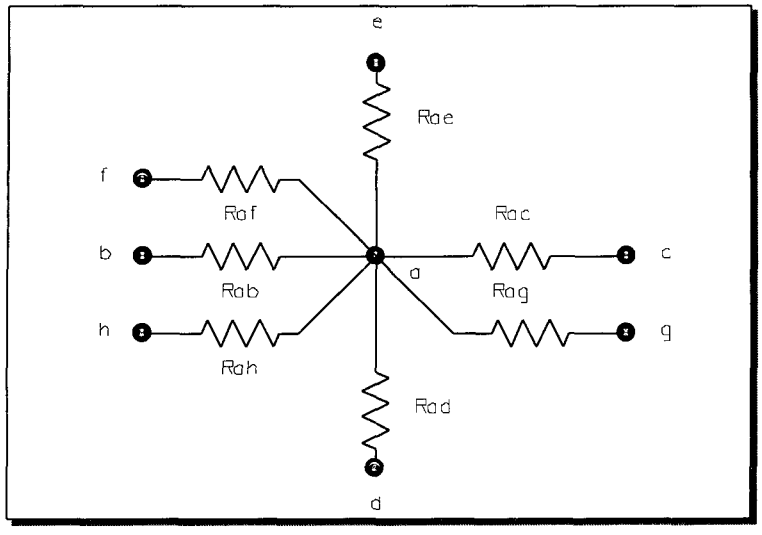


Fig. 4.2.1.2 esquema eléctrico de las resistencias térmicas de un nodo central.

La resistencia térmica de la figura 4.2.1.2 *Rah* es la resistencia térmica del componente con la placa.

Según las dos figuras anteriores las resistencias térmicas de los nodos son:

$$Rab = Rad = Rae = Rac \frac{dy}{\kappa(dxdz)}$$

$$Raf = Rag = \frac{1}{h_{abajo\ /\ arriba}(dydx)} \qquad (4.4)$$

Aplicando la ecuación 4.2 se llega a:

$$C_m = c\rho \cdot \Delta x \cdot \Delta y \cdot \Delta z$$

$$T_a^{i+1} = T_a^i \left(1 - \frac{\Delta t}{C_m}\left(\frac{1}{Rab} + \frac{1}{Rae} + \frac{1}{Rad} + \frac{1}{Rag} + \frac{1}{Rac} + \frac{1}{Raf}\right)\right) +$$

$$\frac{\Delta t}{C_m}\left(\begin{array}{l} Q''\Delta y \cdot \Delta x + T_b^i \frac{1}{Rab} + T_e^i \frac{1}{Rae} + T_d^i \frac{1}{Rad} + T_c^i \frac{1}{Rac} + \\ T_f^i \frac{1}{Raf} + T_g^i \frac{1}{Rag} \end{array}\right)$$

Aplicando los valores de las resistencias térmicas anteriores y el valor de la capacitancia *Cm* y despejando se llega a la ecuación:

$$T_a^{i+1} = T_a^i\left(1-\Delta t\left(2\frac{\kappa}{\Delta x^2 c\rho}+2\frac{\kappa}{\Delta x^2 c\rho}+\frac{h_{arriba}}{\Delta z\cdot c\rho}+\frac{h_{abajo}}{\Delta z\cdot c\rho}\right)\right)+$$

$$\Delta t\left(\begin{array}{l}\dfrac{Q''}{\Delta z\cdot c\rho}+T_b^i\dfrac{\kappa}{\Delta x^2\cdot c\rho}+T_e^i\dfrac{\kappa}{\Delta x^2\cdot c\rho}+T_d^i\dfrac{\kappa}{\Delta x^2\cdot c\rho}+\\ T_c^i\dfrac{\kappa}{\Delta x^2\cdot c\rho}+T_f^i\dfrac{h_{arriba}}{\Delta z\cdot c\rho}+T_g^i\dfrac{h_{abajo}}{\Delta z\cdot c\rho}\end{array}\right)$$

Agrupando términos se llega a:

Ecuación de la temperatura de los nodos centrales a la malla

$$T_a^{i+1} = T_a^{i+1}+\frac{\kappa\Delta t}{\Delta x^2 c\rho}\left(-4T_a^i+T_b^i+T_e^i+T_d^i+T_c^i\right)+$$

$$\frac{\Delta t}{\Delta z\cdot c\rho}\left(-T_a^i h_{arriba}-T_a^i h_{abajo}+T_g^i h_{arriba}+T_f^i h_{abajo}+Q_s''\right) \quad (4.5)$$

4.2.2 Puntos laterales de la malla

El esquema de figura 4.2.2.1 representa los nodos de un punto lateral de la malla, cada letra corresponde a la temperatura de un nodo.

Las resistencias de la figura 4.2.2.1 *Rac* y *Raf* Y *Rag* son resistencias de la placa con el medio ambiente las restantes resistencias térmicas son de conducción.

La superficie de la placa en estos sitios ahora es la mitad que el apartado anterior, lo que la densidad de volumen es la mitad.

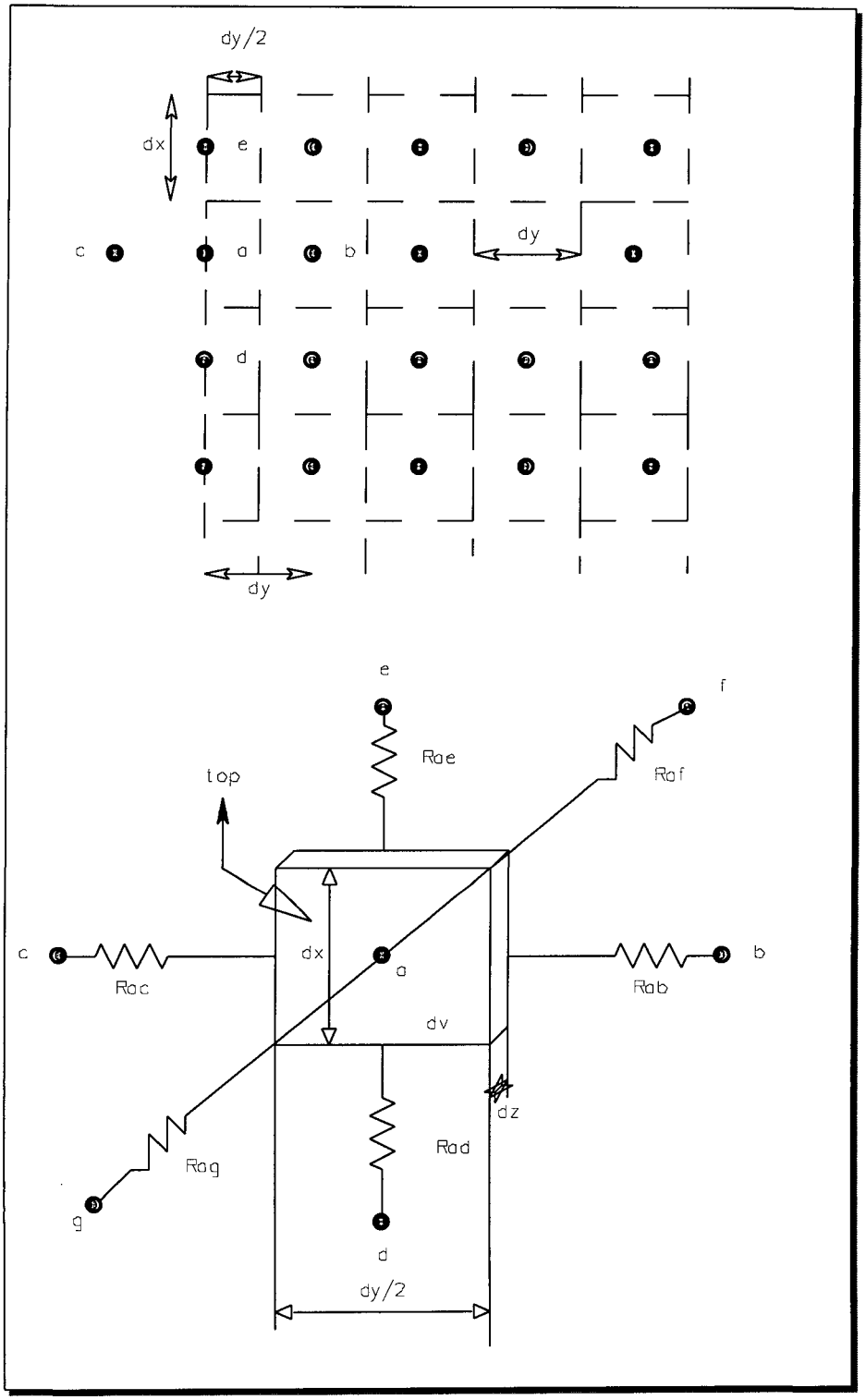


Fig. 4.2.2.1 Estudio de los nodos laterales de una malla.

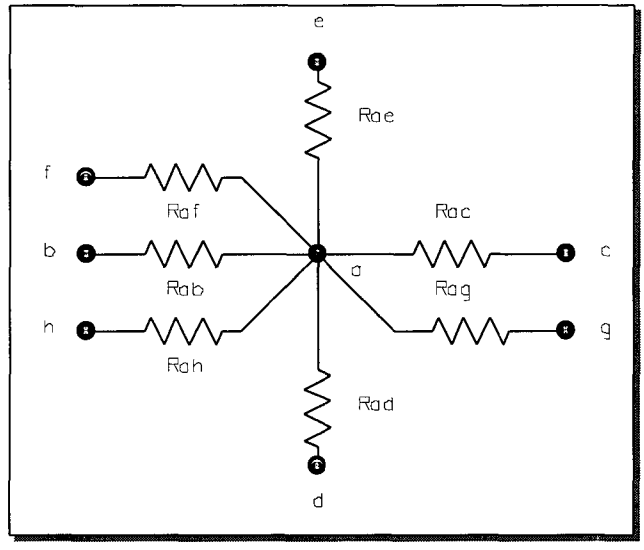


Fig. 4.2.2.2 Representación eléctrica de las resistencias térmicas de los nodos.

La resistencia térmica de la figura 4.2.2.2 *Rah* es la resistencia térmica del componente con la placa.

Según las dos figuras anteriores las resistencias térmicas de los nodos son:

$$Rab = \frac{dy}{\kappa(dxdz)}$$

$$Rae = Rad = \frac{dx}{\kappa\left(\frac{1}{2}dydz\right)}$$

$$Rac = \frac{1}{h_{borde}(dxdz)}$$

$$Raf = Rag = \frac{1}{h_{abajo\ /\ arriba}\left(\frac{1}{2}dydx\right)} \qquad (4.6)$$

Aplicando esotos valores de resistencias térmicas de conducción y convección a la ecuación 4.2 se llega a:

$$C_m = c\rho \cdot \Delta x \cdot \frac{\Delta y}{2} \cdot \Delta z$$

$$T_a^{i+1} = T_a^i \left(1 - \frac{\Delta t}{C_m}\left(\frac{1}{Rab} + \frac{1}{Rae} + \frac{1}{Rad} + \frac{1}{Rag} + \frac{1}{Rac} + \frac{1}{Raf}\right)\right) +$$

$$\frac{\Delta t}{C_m}\left(Q''\frac{\Delta y}{2}\Delta x + T_b^i \frac{1}{Rab} + T_e^i \frac{1}{Rae} + T_d^i \frac{1}{Rad} + T_c^i \frac{1}{Rac} + T_f^i \frac{1}{Raf} + T_g^i \frac{1}{Rag}\right)$$

Aplicando los valores de las resistencias térmicas anteriores y el valor de la capacitancia *Cm* y despejando se llega a la ecuación:

$$T_a^{i+1} = T_a^i \left(1 - \Delta t\left(2\frac{\kappa}{\Delta x^2 c\rho} + 2\frac{\kappa}{\Delta x^2 c\rho} + 2\frac{h_{borde-iz.}}{\Delta x \cdot c\rho} + \frac{h_{arriba}}{\Delta z \cdot c\rho} + \frac{h_{abajo}}{\Delta z \cdot c\rho}\right)\right) +$$

$$\Delta t\left(\frac{Q''}{\Delta z \cdot c\rho} + 2T_b^i \frac{\kappa}{\Delta x^2 \cdot c\rho} + T_e^i \frac{\kappa}{\Delta x^2 \cdot c\rho} + T_d^i \frac{\kappa}{\Delta x^2 \cdot c\rho} + T_c^i \frac{h_{borde-iz.}}{\Delta x \cdot c\rho} + T_f^i \frac{h_{arriba}}{\Delta z \cdot c\rho} + T_g \frac{h_{abajo}}{\Delta z \cdot c\rho}\right)$$

Agrupando términos se llega a:

Ecuación de la temperatura de los nodos laterales de la malla

$$T_a^{i+1} = T_a^{i+1} + \frac{\kappa \Delta t}{\Delta x^2 c\rho}\left(-4T_a^i + 2T_b^i + T_e^i + T_d^i\right) +$$

$$\frac{\Delta t}{\Delta z \cdot c\rho}\left(-T_a^i h_{arriba} - T_a^i h_{abajo} + T_g^i h_{arriba} + T_f^i h_{abajo} + Q_s''\right) +$$

$$\frac{\Delta t}{c\rho \cdot \Delta x}\left(-2T_a^i h_{borde\_iz.} + 2T_c^i h_{borde\_iz.}\right) \qquad (4.7)$$

4.2.3 Puntos en las esquinas de la malla

El esquema de figura 4.2.3.1 representa los nodos de un punto esquina de la malla, cada letra corresponde a la temperatura de un nodo.

Las resistencias de la figura 4.2.3.1 *Rac, Rae, Raf* y *Rag* son resistencias de la placa con el medio ambiente las restantes resistencias térmicas son de conducción.

La superficie de la placa que limita el nodo esquina es la cuarta parte que la del apartado 4.2.1, por lo tanto la densidad de volumen es también la cuarta parte.

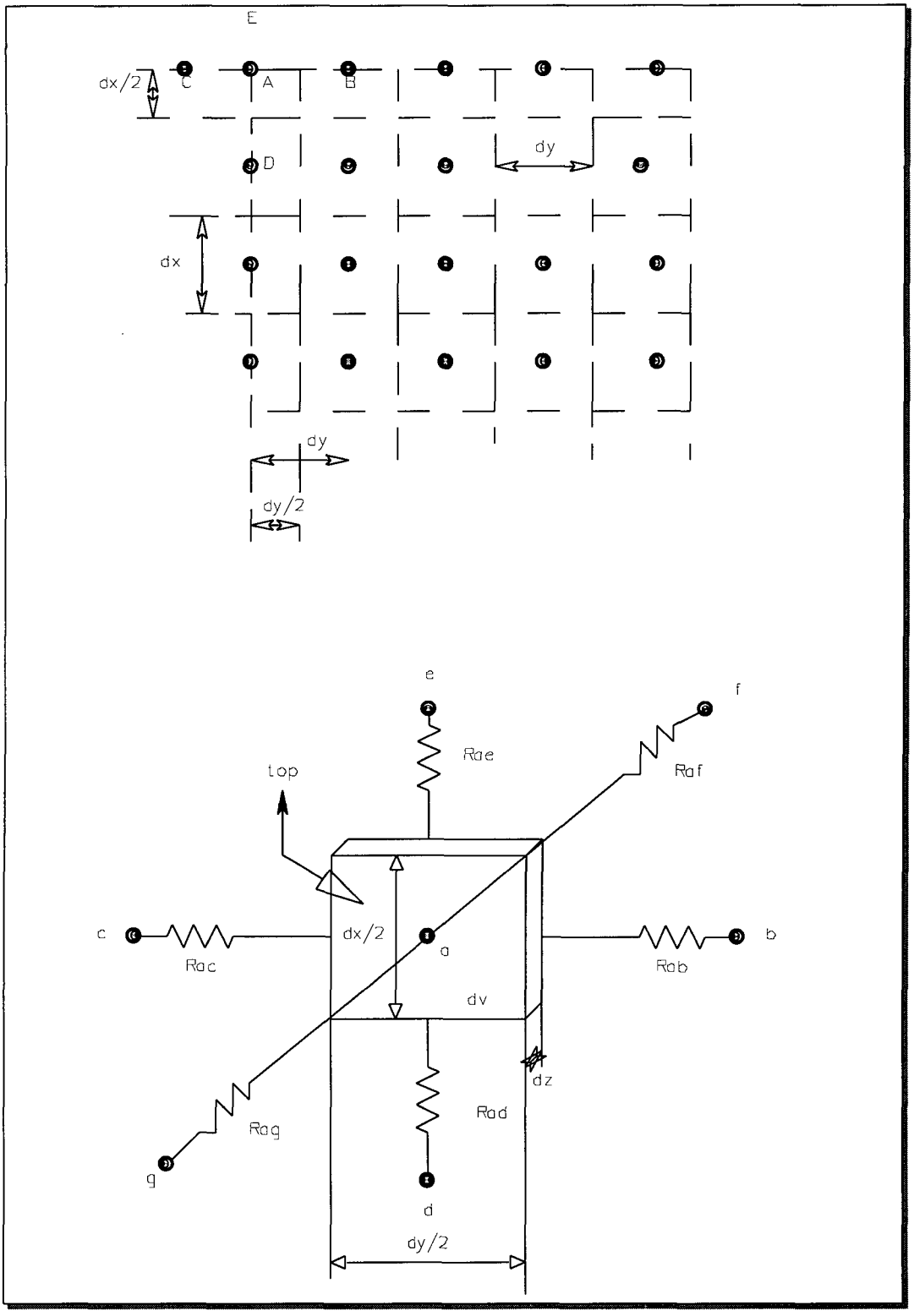


Fig. 4.2.3.1 Estudio de los nodos de las esquinas de la malla.

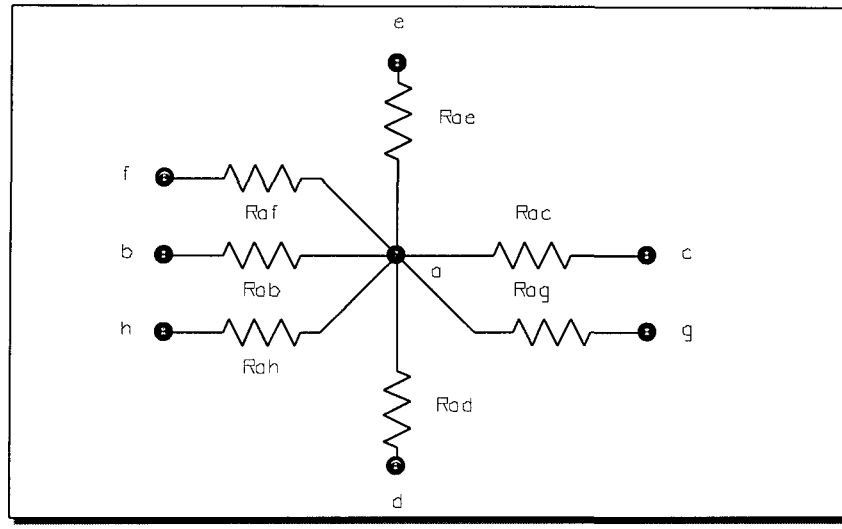


Fig. 4.2.3.2 Representación eléctrica de las resistencias térmicas de los nodos.

La resistencia térmica de la figura 4.2.3.2 *Rah* es la resistencia térmica del componente con la placa.

Según las dos figuras anteriores las resistencias térmicas de los nodos son:

$$Rad = Rab = \frac{dy}{\kappa\left(\frac{dx}{2}dz\right)} = \frac{dx}{\kappa\left(\frac{dy}{2}dz\right)}$$

$$Rae = \frac{1}{h_{borde-arriba}\left(\frac{dy}{2}dz\right)}$$

$$Rac = \frac{1}{h_{borde}\left(\frac{dx}{2}dz\right)}$$

$$Raf = Rag = \frac{1}{h_{abajo\ /\ arriba}\left(\frac{dx}{2}\frac{dy}{2}\right)} \qquad (4.8)$$

Aplicando la ecuación 4.2 se llega a:

$$C_m = c\rho \cdot \frac{\Delta x}{2} \cdot \frac{\Delta y}{2} \cdot \Delta z$$

$$T_a^{i+1} = T_a^i \left(1 - \frac{\Delta t}{C_m} \left(\frac{1}{Rab} + \frac{1}{Rae} + \frac{1}{Rad} + \frac{1}{Rag} + \frac{1}{Rac} + \frac{1}{Raf} \right) \right) +$$

$$\frac{\Delta t}{C_m} \left(\begin{array}{l} Q'' \frac{\Delta x}{2} \cdot \frac{\Delta y}{2} + T_b^i \frac{1}{Rab} + T_e^i \frac{1}{Rae} + T_d^i \frac{1}{Rad} + T_c^i \frac{1}{Rac} + \\ T_f^i \frac{1}{Raf} + T_g^i \frac{1}{Rag} \end{array} \right)$$

Aplicando los valores de las resistencias térmicas anteriores y el valor de la capacitancia *Cm*, despejando se llega a la ecuación:

$$T_a^{i+1} = T_a^i \left(1 - \Delta t \left(\begin{array}{l} 2\frac{\kappa}{\Delta x^2 c\rho} + +2\frac{h_{borde\_superior}}{\Delta x \cdot c\rho} + 2\frac{\kappa}{\Delta x^2 c\rho} + \\ \frac{h_{arriba}}{\Delta z \cdot c\rho} + 2\frac{h_{borde-iz.}}{\Delta x \cdot c\rho} + \frac{h_{abajo}}{\Delta z \cdot c\rho} \end{array} \right) \right) +$$

$$\Delta t \left(\begin{array}{l} \frac{Q''}{\Delta z \cdot c\rho} + 2T_b^i \frac{\kappa}{\Delta x^2 \cdot c\rho} + 2T_d^i \frac{\kappa}{\Delta x^2 \cdot c\rho} + 2T_c^i \frac{h_{borde-iz.}}{\Delta x \cdot c\rho} + \\ 2T_e^i \frac{h_{borde\_superior}}{\Delta x \cdot c\rho} + T_f^i \frac{h_{arriba}}{\Delta z \cdot c\rho} + T_g \frac{h_{abajo}}{\Delta z \cdot c\rho} \end{array} \right)$$

Agrupando términos iguales se llega a:

Ecuación de la temperatura de los nodos esquina de la malla

$$T_a^{i+1} = T_a^i + \frac{\kappa \Delta t}{\Delta x^2 c\rho}\left(-4T_a^i + 2T_b^i + 2T_d^i\right) + \qquad (4.9)$$

$$\frac{\Delta t}{\Delta z \cdot c\rho}\left(-T_a^i h_{arriba} - T_a^i h_{abajo} + T_g^i h_{arriba} + T_f^i h_{abajo} + Q_s''\right) +$$

$$\frac{\Delta t}{c\rho \cdot \Delta x}\left(-2T_a^i h_{borde\_sup.} - 2T_a^i h_{borde\_iz.} + 2T_e^i h_{borde\_sup.} + 2T_c^i h_{borde\_iz.}\right)$$

Las ecuaciones 4.5, 4.7 y 4.9 son las implementadas en el algoritmo desarrollado en el programa, y que se aplicarán por cada iteración al nodo correspondiente.

Termin 1.0
Simulador Térmico

Capítulo 5

Potencia suministrada a la placa

5 Potencia suministrada a la placa

5.1 LOS DIFERENTES TIPOS DE TRANSMISIÓN DE POTENCIA A LA PLACA

En esta aplicación, vamos a considerar dos tipos de potencia que suministra el componente a la placa:

- Proporcional al tamaño de la superficie del componente.
- Mediante los pines de los componentes (hasta 3 pines).

5.2 RESISTENCIA TÉRMICA DEL COMPONENTE CON LA PLACA

El flujo de calor en las tres direcciones se divide en un flujo en la dirección del gradiente de temperatura (x e y), y otro lateral. Es debido a este último que la resistencia térmica disminuye, puesto que la superficie "encerrada" por el flujo de calor es mayor.

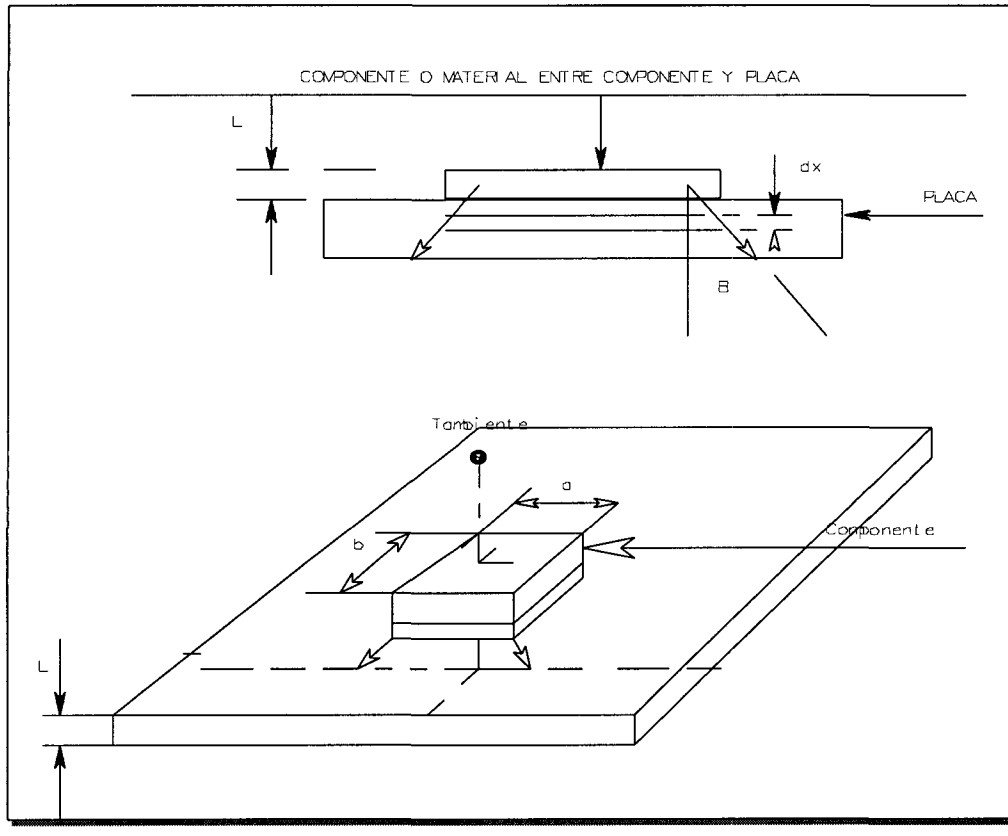


Fig. 5.2.1 Acoplamiento componente placa para obtener la resistencia de placa.

- *B*: ángulo de dispersión del flujo de calor.
- *Dx*: diferencial de longitud del camino seguido por e flujo de calor.
- *S=a*b* sección (superficial) del flujo de calor.

El ángulo de dispersión de *B* es el que forma el gradiente de temperatura en la dirección del flujo principal (que es el que une la parte más caliente, es decir el componente, con la más fría, es decir parte inferior del substrato que estará en contacto con el medio ambiente o en una "caja" de refrigeración) con el flujo lateral.

En la mayoría de los casos se toma este ángulo del valor de 45°. Por tanto para calcular la resistencia térmica del substrato se considera que el flujo de calor esta contenido en un cono de 45° de ángulo. El valor de esta resistencia esta descrita por:

$$Rth = \frac{1}{\kappa}\int_0^L \frac{dx}{S} \quad (5.1)$$

A continuación calculamos la resistencia térmica que presenta el substrato para distintas formas de los componentes.

- ***Componente (fuente calor) cuadrado:***

En este casa *a=b*. Suponiendo que el material entre componente y placa es isotérmico (su conductividad y demás aspectos físicos se mantienen constantes para toda la superficie), la resistencia térmica es:

$$RTh = \frac{1}{\kappa}\int_o^l \frac{dx}{S} = \frac{1}{\kappa}\int_0^l \frac{dx}{(a+2x)^2}$$

Realizando la integral:

$$Rth = \frac{1}{\kappa}\left|\frac{1}{2(a+2L)}\right|_0^l$$

$$Rth = \frac{L}{\kappa a(a+2L)} \quad (5.2)$$

- **Componente (fuente de calor) rectangular:**

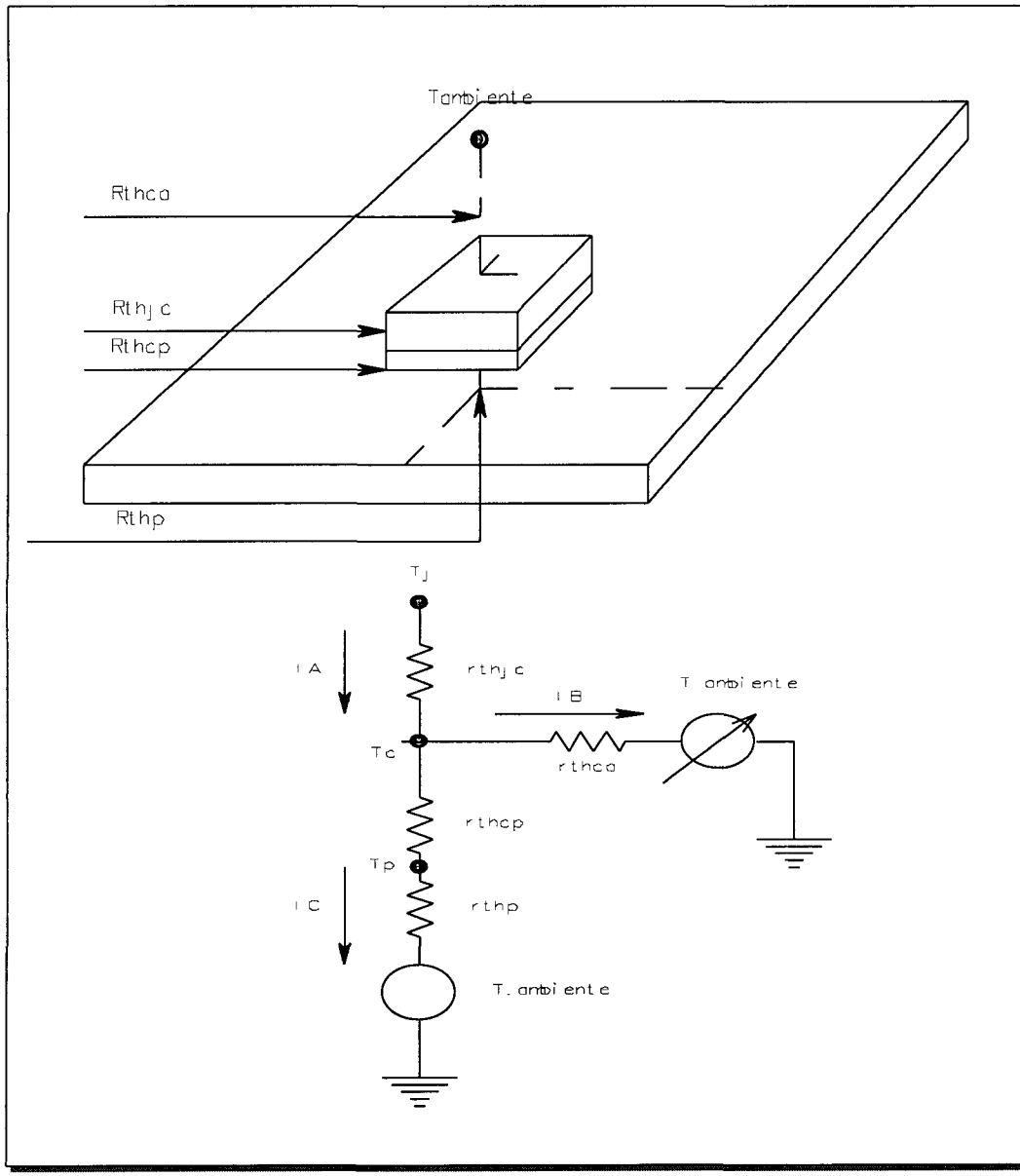
$$Rth = \frac{1}{\kappa}\int_0^L \frac{dx}{S} = \frac{1}{\kappa}\int_0^L \frac{dX}{(b+2x)(a+2x)}$$

Integrando se tiene:

$$Rth = \frac{1}{2\kappa(b-a)} \ln\left(\frac{b}{a}\left(\frac{a+2L}{b+2L}\right)\right) \qquad (5.3)$$

5.2.1 Potencia proporcional al tamaño de la superficie del componente

A continuación se realizará un estudio de la potencia suministrada a la placa cuando entre el componente y la placa hay un material que los separa.



5.2.1.1 Estudio de la potencia del componente a la placa en modo superficial.

Si se plantea las ecuaciones de la malla eléctrica de la figura 5.2.1.1, llegamos:

- Temperatura de la unión (Tj)
- Resistencia térmica unión–cápsula (Rthjc)
- Resistencia térmica cápsula-ambiente (Rthca)
- Resistencia térmica placa (Rthp)
- Resistencia térmica cápsula-placa (Rthcp)

$$Tj - Tambiente\_bajo\_placa = I(Rthjc + Rthcp + Rthp) - Ib(Rthcp + Rthp) \quad (5.4)$$
$$Tambiente\_bajo\_placa - Tambiente = Ib(Rthcp + Rthp + Rthca) - I(Rthcp + Rthp)$$

Si se hace la siguiente notación:

$$A = Rthjc + Rthcp + Rthp$$
$$B = Rthp + Rthcp$$
$$C = Rthca + Rthp + Rthcp$$

Sustituyendo la anterior expresión en la ecuación 5.4 se llega a:

$$Tj - Tambiente\_bajo\_placa = I \cdot A - Ib * B$$
$$Tambiente\_bajo\_placa - Tambiente = I \cdot B + Ib \cdot C$$

Si para el primer instante de tiempo la temparuta ambiente es la misma en todos los puntos, es decir:

Tambiene_bajo_placa-Tambiente=0

$$\text{Ib} = \frac{\text{I} \cdot \text{B}}{\text{C}} \Rightarrow Tj = Tambiente\_bajo\_placa + I \cdot A - B\left(\frac{\text{I} \cdot \text{B}}{\text{C}}\right) \Rightarrow$$
$$Tj = Tambiente\_bajo\_placa + I\left(A - \frac{B^2}{C}\right) \Rightarrow$$

Siguiendo desarrollando se llega a :

$$Tj = Tambiente\_bajo\_placa + I\left[Rthjc + Rthcp + Rthp - \frac{(Rthp + Rthcp)^2}{Rthca + Rthp + Rthcp}\right]$$

$$IB = IA \cdot \frac{B}{C}; \quad IC = IA \cdot \left(1 - \frac{B}{C}\right)$$
$$TP = IA \cdot rthp \cdot \left(1 - \frac{rthp + rthcp}{rthca + rthp + rthcp}\right) + T.ambiente$$
$$TC = IA \cdot \left(rthcp + rthp - \frac{(rthcp + rthp)^2}{rthcp + rthp + rthca}\right) + T.ambiente$$

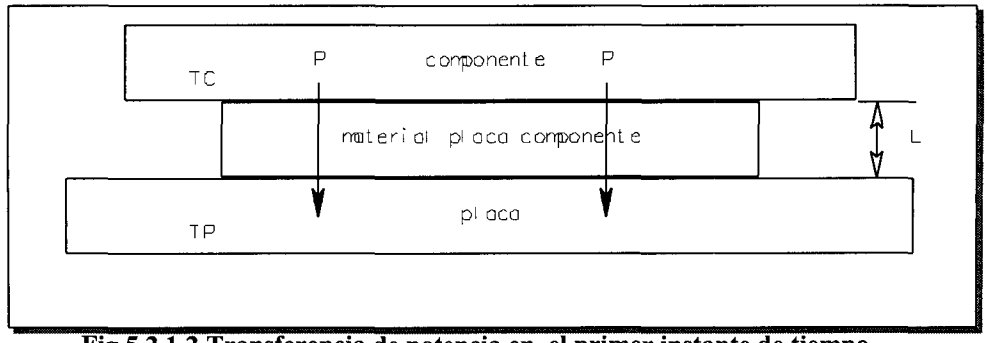


Fig 5.2.1.2 Transferencia de potencia en el primer instante de tiempo.

$$Rth = \frac{L}{K \cdot S} \Rightarrow \frac{k}{L} = \frac{1}{Rth \cdot S}$$

$$Q'' = \frac{k}{L}(TP - Tambiente) \Rightarrow Q'' = \frac{TP - Tambiente}{rthp \cdot A}$$

A=S Área del material entre la placa-componente.
k= Conductividad del material entre la placa-componente.
Q''=Potencia superficial que entrega el componente a la placa.
TP=Temperatura del material aislante de la placa.

Esta sería la potencia al inicio de la simulación, que cambiará por cada iteración, que se haga para calcular la temperatura final. Por eso en las restantes internaciones del algoritmo la resistencia de la placa no intervendrá y el nuevo esquema es el siguiente:

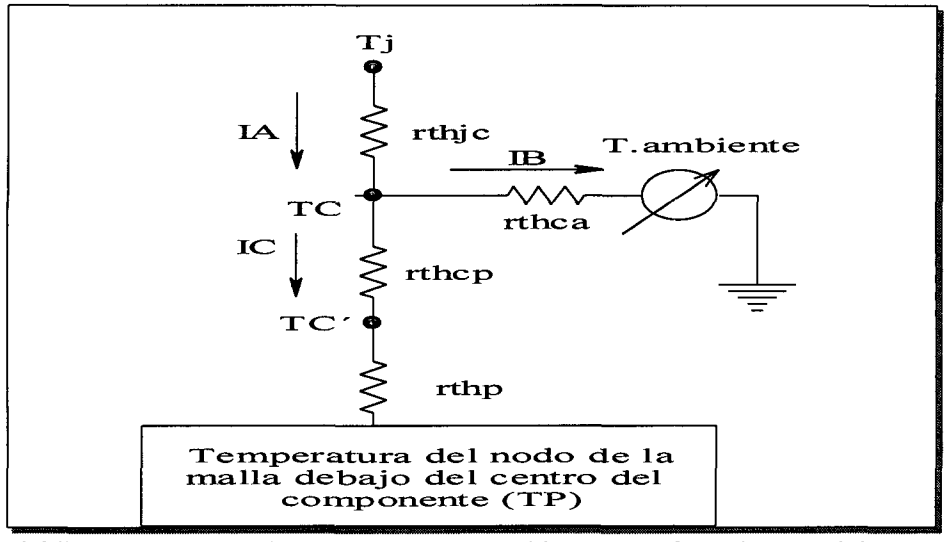


Fig 5.2.2 Reparto de potencia en placa y medio ambiente después del instante inicial.

Si se plantea las ecuaciones de las dos mallas del circuito anterior y se desarrollan:

$$IC = IA - IB$$

$$\left.\begin{aligned} 0 &= (Rcp + Rp)(IA - IB) - Tp + Rca \cdot IB + Tambiente \\ 0 &= -Rca \cdot IA + (Rcp + Rp) \cdot IB - Tp + Rca \cdot IB + Tambiente \end{aligned}\right\} \Rightarrow$$

$$\Rightarrow IB = \frac{Tp - Tambiente + (Rcp + Rp) \cdot IA}{Rcp + Rp + Rca}$$

$$TC' = Rp \cdot (IA - IB) + Tp \Rightarrow TC' = Rp \cdot \left(IA - \left(\frac{Tp - Tambiente + (Rcp + Rp) \cdot IA}{Rcp + Rp + Rca} \right) \right) + Tp$$

$$TC' = IA\left(\frac{Rca \cdot Rp}{Rca + Rcp + Rp} \right) + Tp\left(\frac{Rcp + Rca}{Rca + Rcp + Rp} \right) + \frac{Tambiente\ Rp}{Rca + Rcp + Rp} \Rightarrow$$

$$TC' = \frac{1}{Rca + Rcp + Rp}\left(IA(Rca \cdot Rp) + (Rcp + Rca) \cdot Tp + Tambiente\ Rp \right)$$

$$Rth = \frac{L}{K \cdot S} \Rightarrow \frac{k}{L} = \frac{1}{Rth \cdot S}$$

$$Q'' = \frac{k}{L}(TC' - TP) \Rightarrow \quad Q'' = \frac{TC' - TP}{rthp \cdot A}$$

- Potencia superficial Q''.
- Temperatura del material conductor, pero procedente de la simulación TP.

$$Q'' = \frac{TC' - TP}{rthp \cdot A} \quad (5.5)$$

La potencia anterior Q'' es la que suministra el componente a la placa, y a cada iteración será diferente a la anterior debido a que TP variará.

En resumen se puede decir que cuanto mayor sea TP menor será la potencia que el componente suministrará a la placa, debido a que la placa no es capaz de radiarla al ambiente. Es más, si el valor de TP llega un valor muy elevado el componente no es capaz de liberar calor a través de la placa sino que absorberá calor a través de esta última. Esto está reflejado en el programa enviando una señal de aviso.

5.2.2 Potencia a través de los pines de los componentes

Suponiendo que tengamos el siguiente componente:

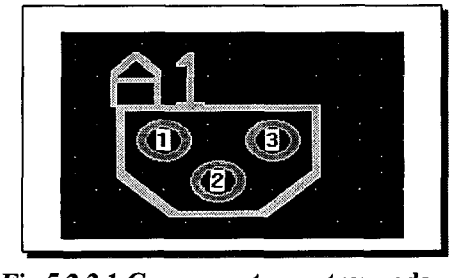


Fig 5.2.2.1 Componente con tres pads.

La interpretación térmica de este componente en la placa es:

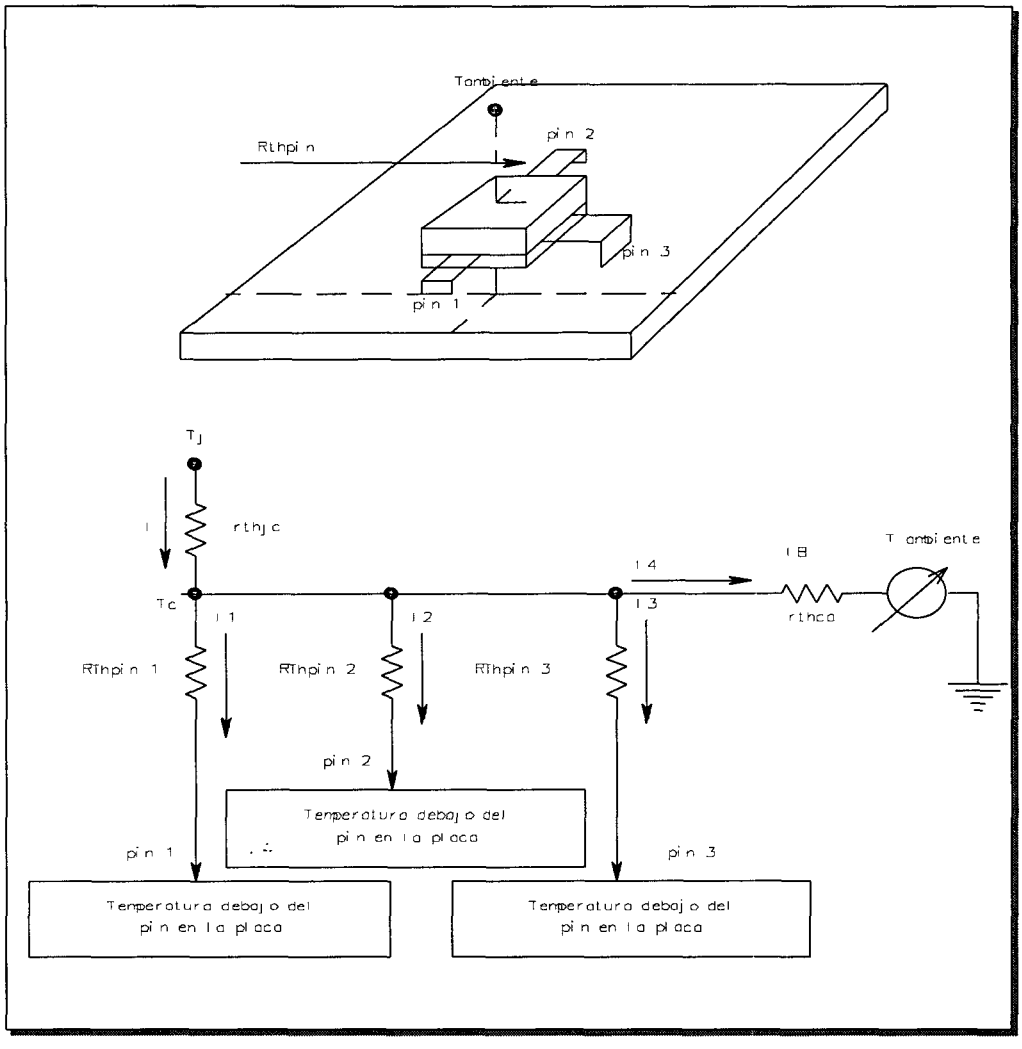


Fig. 5.2.2.2 Estudio de la potencia aplicada a la placa a través de los pines.

Planteando las ecuaciones de malla del circuito de la figura 5.2.2.2:

$$I4 = I - (I1 + I2 + I3)$$

$$\left.\begin{array}{l} Rthpin\ 1 = Rthpin\ 2 = Rthpin\ 3 = Rthpin \\ Rthpin \cdot I1 + Tpin\ 1 - Tpin\ 2 - Rthpin \cdot I2 = 0 \\ Tpin\ 2 - Tpin\ 3 + Rthpin \cdot I2 - Rthpin \cdot I3 = 0 \\ Tpin\ 3 - Tpin\ 4 + Rthpin \cdot I3 - Rthca \cdot (I - (I1 + I2 + I3)) = 0 \end{array}\right\} \Rightarrow$$

$$\begin{array}{ll} -Rthpin \cdot I1 + Rthpin \cdot I2 & = Tpin1 - Tpin2 \\ -Rthpin \cdot I2 + Rthpin \cdot I3 & = Tpin2 - Tpin3 \\ -Rthca \cdot I1 - Rthca \cdot I2 - (Rthca + Rthpin) \cdot I3 & = Tpin3 - Tpin4 - Rthca \cdot I \end{array}$$

Que de forma genérica, para poder resolverlo por un sistema de ecuaciones lineales como LU, queda:

$$\begin{array}{cccccccc} & 1 & & 2 & & 3 & \ldots & n \\ 1 & -Rthpin\cdot I1 + Rthpin\cdot I2 & & & & & & = Tpin1 - Tpin2 \\ 2 & & -Rthpin\cdot I2 + Rthpin\cdot I3 & & & & & = Tpin2 - Tpin3 \\ \vdots & & & & & & \ldots & \\ n & -Rthca\cdot I1 - Rthca\cdot I2 - Rtpin\cdot I3 - & & \ldots & & (Rthca + Rthpin)\cdot I_n & & = Tpin_n - Tpin_{n+1} - Rthca\cdot I \end{array}$$

El método empleado para resolver este sistema es la descomposición LU.

ALGUNAS UNIDADES USADAS EN LA MEMORIA

κ	$\frac{W}{m\ ^{o}C}$	➢ Conductividad
β	^{o}C	➢ Coeficiente volumétrico
g	m/s^{2}	➢ Gravedad
	$\frac{m^{2}}{s}$	➢ Viscosidad cinemática
c	$\frac{KJ}{Kg \cdot ^{\circ}C}$	➢ Calor específico
μ	$\frac{Kg}{m \cdot s}$	➢ Viscosidad dinámica
L	m	➢ Longitud de la placa
ρ	$\frac{Kg}{m^{3}}$	➢ Densidad Volumétrica
ΔT	^{o}C	➢ Diferencia de temperatura
Δx	m	➢ Incremento de espacio entre nodo y nodo
Δt	s	➢ Intervalo de tiempo entre dos iteraciones
ΔV_{m}	m^{2}	➢ Diferencial de volumen
$A = S$	m^{2}	➢ Área superficie
Rth	$\frac{W}{m\ ^{o}C}$	➢ Resistencia térmica
h	$\frac{W}{m^{2}\ ^{o}C}$	➢ Coeficiente de transferencia
Q''	W/ m	➢ Potencia superficial
Q'''	W/m^{3}	➢ Potencia Volumétrica (interior a la placa)
C_{m}	$\frac{KJ}{^{o}C}$	➢ Capacitancia

Tercera Parte

Trata de exponer las ideas del funcionamiento de ***Termín*** en su nivel más interno. Y las soluciones adoptadas para la realización del programa.

Termin 1.0
Simulador Térmico

Capítulo 6

Problemas y soluciones

6 Problemas y soluciones

6.1 LENGUAJE DE PROGRAMACIÓN

A la hora de optar por un lenguaje de programación surge la duda de cual usar. La gran variedad de lenguajes de programación existente en el mercado hace la elección ardua. Hay que establecer unos requisitos mínimos en función de lo que va hacer el programa, estos requisitos pueden ser:

- Gran potencia del cálculo.
- Gran potencia en gráficos.
- Legibilidad a la hora de leer el programa por un tercero.
- Fácil de dominar desde el principio.

Los requisitos anteriores no los cumple un lenguaje en concreto, el C++ se acerca mucho a lo pedido pero los dos últimos requisitos no los cumple. Por otro lado, sería útil que el lenguaje utilizado ya se conozca. Es aquí, donde surge la idea de usar *Delphi 2*, que utiliza el Pascal como herramienta de programación. En cuanto a los requisitos esgrimidos anteriormente se comporta satisfactoriamente bien.

6.2 EL PROGRAMA

En los siguientes apartados se explicará de forma resumida el comportamiento del programa, ampliando las partes más interesantes y confusas que de él puede encontrarse.

6.2.1 Flujo general del programa

El flujo de la figura 6.2.1.1 y 6.2.1.2 representa de una manera general el comportamiento del programa que se irá desmenuzando y comentando poco a poco.

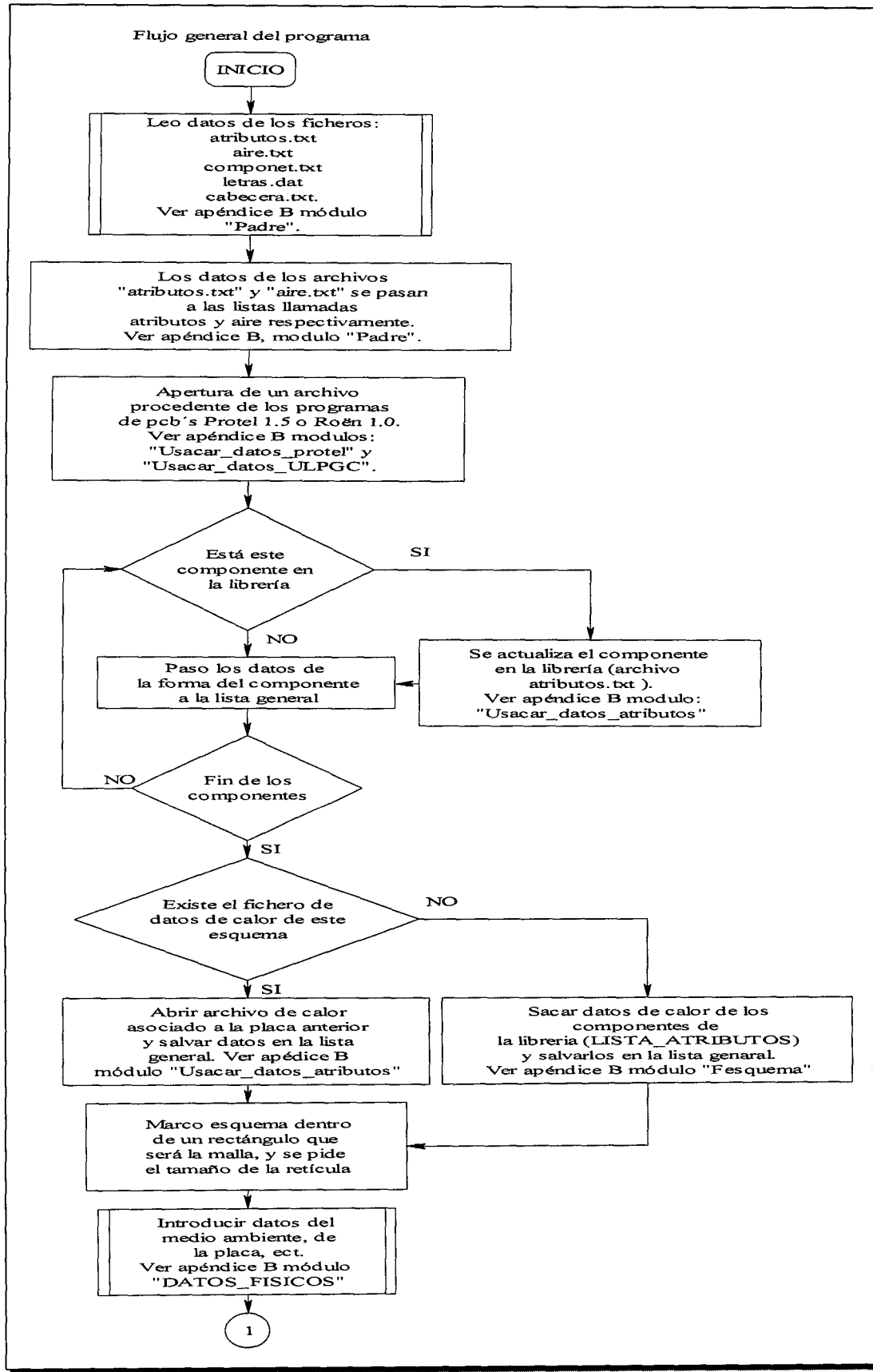


Fig. 6.2.1.1 Flujo general del programa.

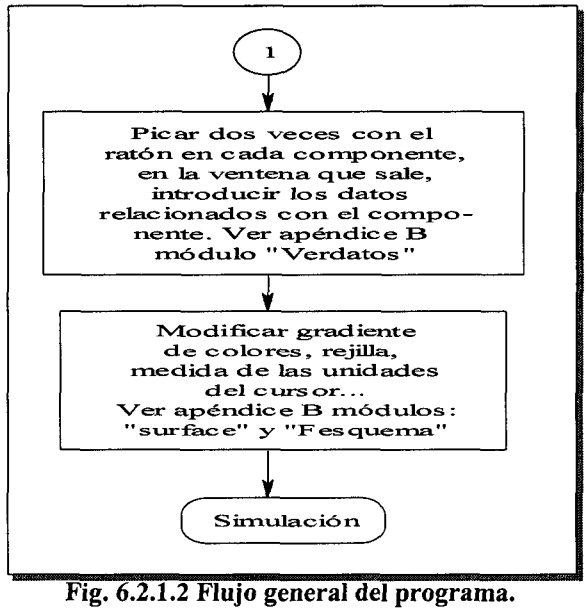


Fig. 6.2.1.2 Flujo general del programa.

6.2.2 Archivos utilizados

En figura 6.2.1.1 lo primero que se puede ver son los archivos utilizados de los cuales se explican su utilidad:

- **Atributos.txt:**

El archivo en cuestión es la librería del programa en cuanto a parámetros de calor se refiere, que se pueden ir ampliando cuando el esquema contenga un componente que no se encuentre en ella. A la hora de introducir los datos, es posible seleccionar un componente que actualmente esté en la librería y de ese modo pasarle los parámetros al nuevo componente, pudiendo modificar estos.

Parte del archivo *atributos.txt* se ve en la tabla 6.2.2.1.

Atributos.txt	
En este fichero están las características de calor	
De los componentes	
CONTENIDO:	
versión 1;	
INICIO_DATA_COMPONENTES	
_COMPONENTE	
DIP14	
_REFERENCIA	
XXXXXXXXX	
_POTENCIA (W)	*0,2500
_Tj (ºC)	*0,0000
_Tc (ºC)	*0,0000
_Tp (ºC)	*0,0000
_Q(W/m²)	*0,0000
_RTHJC (W/m)	*12,0000
_RTHCA (W/m)	*12,0000
_RTHCP (W/m)	*12,0000
_RTHP (W/m)	*0,0000
_LARGO (m)	*0,0100
_ANCHO (m)	*0,0050
_ESPESOR (m)	*0,0030
_RTHPIN (W/m)	*300,0000
_ESPESOR_PIN (m)	*0,001
_LARGO_PIN (m)	*0,0100
_ANCHO_PIN (m)	*0,0020
_CONDUCTIVIDAD_PIN (W/m ºC)	*14,0000

Tabla 6.2.2.1 Interior del archivo *atributos.txt.*

- **Componet.txt**

Este es el archivo donde se almacena los componentes actualmente en la librería. Este archivo se puede ampliar cada vez que se introduzca nuevos componentes. Parte del archivo componet.txt se ve en la tabla 6.2.2.2.

componet.txt
AXIAL0.5
DIP14
DIP24
MLL34
MLL41
PLCC100
SOT-89
TO-5
TO-66

Tabla 6.2.2.2 parte del archivo *component.txt.*

- **Aire.txt**

Los datos del archivo *aire.txt* se pasan a una lista simple, para que el cálculo se haga de una manera más rápida. Estos datos son los que se utilizan para calcular el coeficiente de transferencia de la placa del capitulo 3.

La tabla 6.2.2.3 es el contenido del archivo *aire.txt.*

PROPIEDADES DEL AIRE SECO A PRESION ATMOSFERICA (UNIDADES SI)					
Temperatura aire	Calor específico (c)	Densidad volumétrica (p)	Viscosidad dinámica(v)	Coriductividad (k)	nº de Prandtl (Pr)
0	1,0057	1,2923	0,00001331	0,02408	0,718
10	1,0058	1,2467	0,00001419	0,02487	0,716
20	1,0061	1,2042	0,00001509	0,02564	0,713
30	1,0064	1,1644	0,00001601	0,02601	0,712
40	1,0068	1,1273	0,00001696	0,0271	0,71
50	1,0074	1,0924	0,00001792	0,02771	0,709
60	1,008	1,0596	0,0000189	0,02852	0,708
70	1,0087	1,0287	0,0000199	0,02922	0,707
80	1,0095	0,9996	0,00002092	0,02991	0,706
90	1,013	0,9721	0,00002196	0,03059	0,705
100	1,0113	0,946	0,00002302	0,03127	0,704
110	1,0123	0,9213	0,0000241	0,03194	0,704
120	1,0134	0,8979	0,00002519	0,03261	0,703
130	1,0146	0,8756	0,00002631	0,03328	0,702
140	1,0159	0,8544	0,00002744	0,03394	0,702

Tabla 6.2.2.3 Contenido del archivo aire.txt.

- **Letras.dat**

Este archivo es de datos binarios y contiene las coordenadas de las rectas de las letras, que están almacenadas en un array fijo de 30. Esto es debido a que las letras de los componentes están construidas con rectas.

6.2.3 Archivos que son aceptados procedentes de otros programas

Los archivos aceptados son:

- Programa de PCB de *Protel 1.5*, en forma de texto.
- Programa de PCB *Roën 1.0.*
- Archivo de parámetros de calor de los componentes.

Este último archivo es casi de la misma forma que el de *atributos.txt*, en este caso los valores están particularizados para el esquema con el que este asociado, es decir, si el archivo es por ejemplo *fuente.pcb* de *Protel 1.5*, si este fue salvado con anterioridad dentro del programa *Termín 1.0*, tendrá asociado

el archivo *fuente.txt*, cuyo contenido son los datos de calor de los componentes del circuito, que podrá coincidir o no con los de la librería *atributos.txt*.

6.2.4 Forma de las matrices y vectores utilizados

Una de las ventajas de este programa es que las matrices y vectores son creadas en tiempo de ejecución, lo que el valor no está prefijado de antemano, es decir, que su tamaño puede ser escogido entre un rango amplio.

La forma de hacerlo es un poco complicada pero muy útil, a saber:

El primer paso es obtener el máximo valor que el compilador es capaz de manejar para una array, lo cual se consigue con una de las directivas que tiene *Delphi 2.0*, en concreto con la directiva *$f*, de la siguiente figura 2.6.4.1:

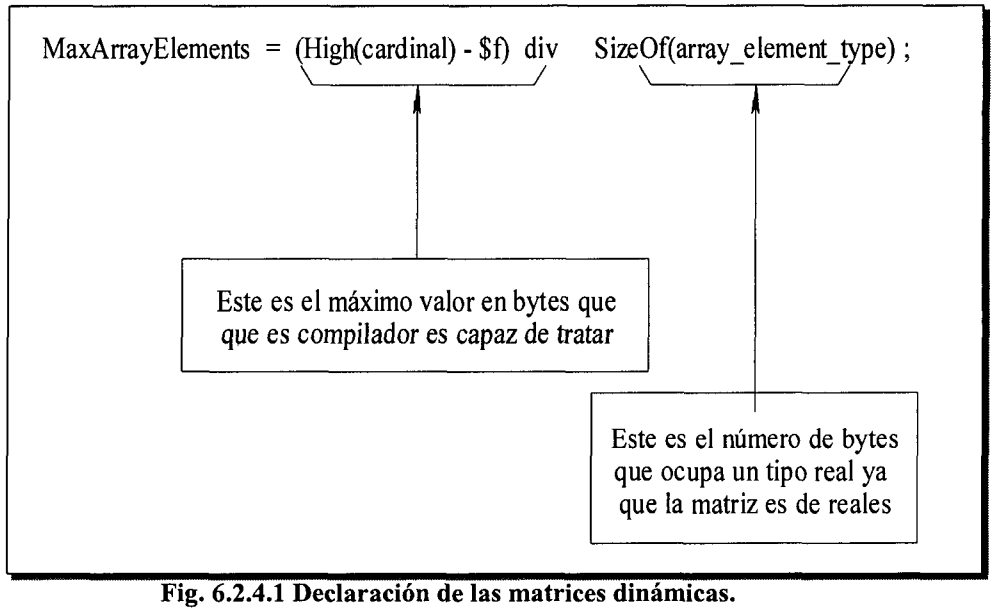


Fig. 6.2.4.1 Declaración de las matrices dinámicas.

El segundo paso es definir un tipo de *array*, y un puntero de ese tipo de *array*, es decir:

```
Type
TMyDynamicArray = array[1..MaxArrayElements] of array_element_type;
PMyDynamicArray = ^TMyDynamicArray
```

Hay que saber que ninguna memoria esta asignada cuando se define un tipo, así se puede definir *TmyDynamicArray* tan grande como el compilador y sistema operativo permita.

El tercer paso es declarar un puntero que apunte a un *array* de la forma anterior, por ejemplo:

```
Var
Vector : PMyDynamicArray;
```

Para que el puntero apunte a una cadena de elementos (*array_element_type*), hay que asignar memoria según el número de elementos que formará ese array, valor que se obtiene en tiempo de ejecución, por lo que una vez obtenido ese número, dentro del programa se reservará ese espacio de memoria de la forma:

```
getMem(vector, NumElementos * sizeof(array_element_type) )
```

La forma de acceder al vector anterior será de la manera siguiente:

```
for n := 0 to NumElements-1 do
vector^[n+1] := 2*n + 1;
```

La forma de obtener una matriz es muy similar, ya que las matrices en este programa están definida como un vector de vectores.

La desventaja de este método es que el compilador no es capaz de detectar cuando se intenta acceder a un elemento que no pertenece al *array* dinámico cuando se está compilando, la única forma de detectar este error es en tiempo de ejecución, debido a que sale un error típico de acceso fuera de memoria.

6.3 FORMA DE LA LISTA GENERAL DEL PROGRAMA

Cada nodo de la lista es un componente que contiene una estructura de la forma del componente, dicha forma es la obtenida del programa de PCB *Protel 1.5*, o *Roën 1.0* hablado anteriormente. Además en cada nodo tiene asociado una matriz y un vector cuya dimensión depende del número de pines que este componente tenga, la estructura de cada componente dentro del nodo tiene la forma de la figura 6.3.1.

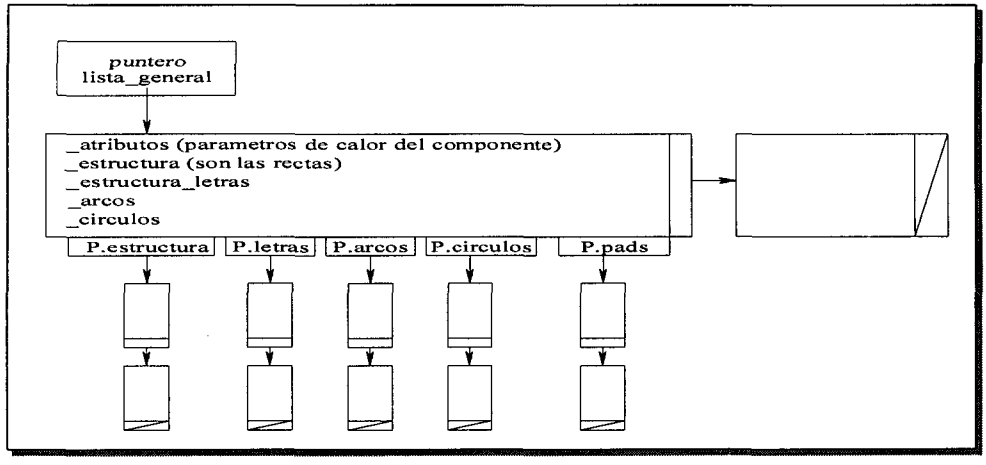


Fig. 6.3.1 forma de la lista general del programa.

6.4 FORMA DE LA ESTRUCTURA DE LOS COMPONENTES Y DE LOS ARCHIVOS ROËN 1.0 Y PROTEL 1.5

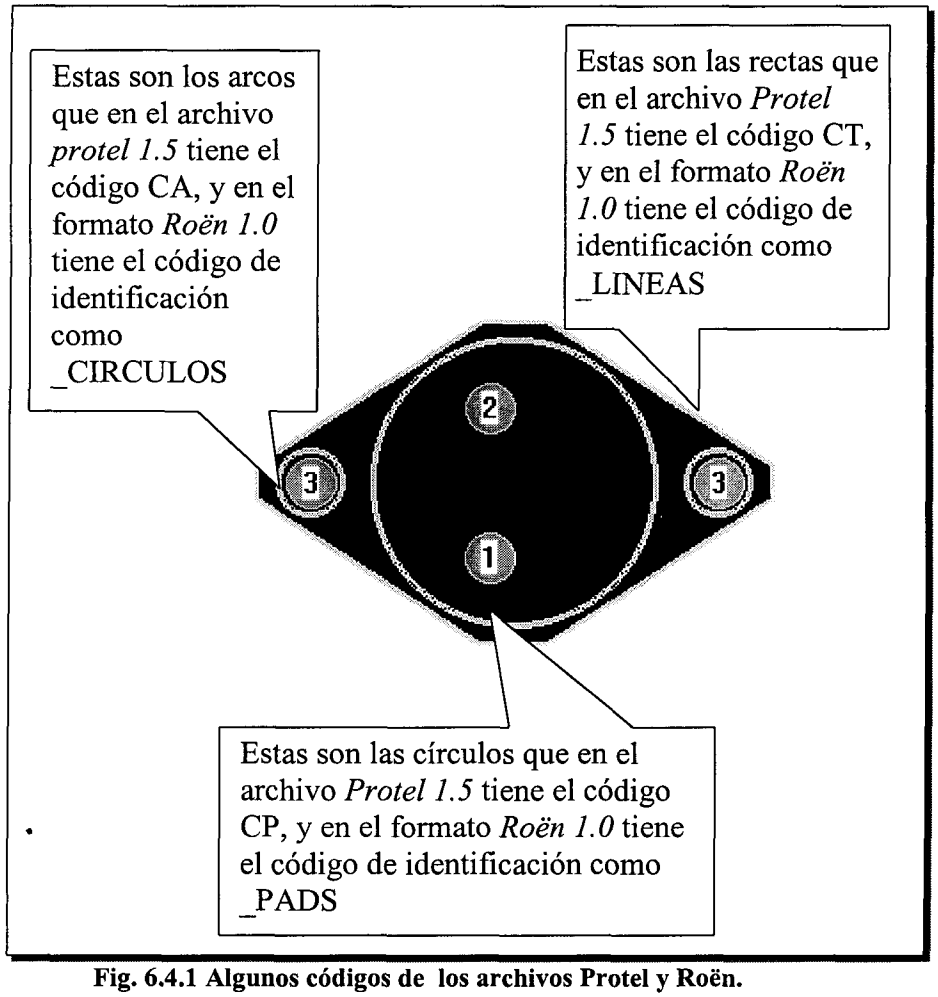


Fig. 6.4.1 Algunos códigos de los archivos Protel y Roën.

Las figuras 6.4.2 y 6.4.3 muestran algunas líneas del archivo *Protel 1.5* del componente de la figura 6.4.1.

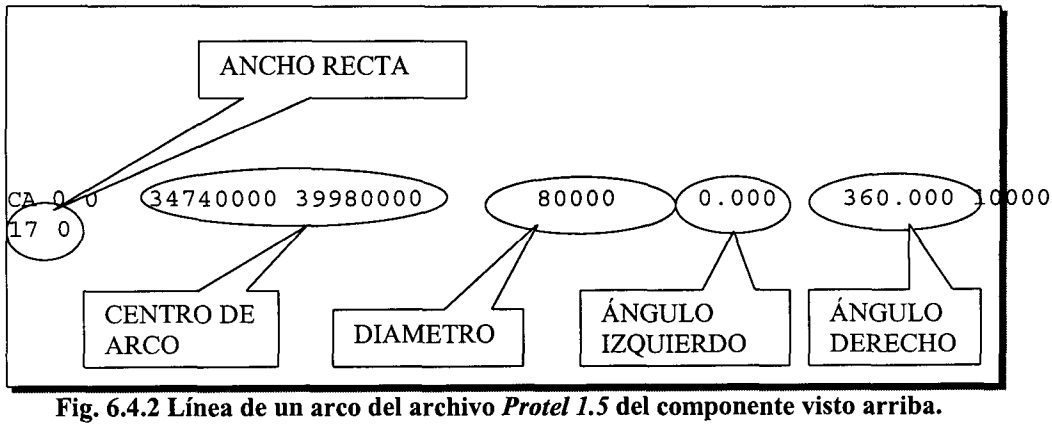


Fig. 6.4.2 Línea de un arco del archivo *Protel 1.5* del componente visto arriba.

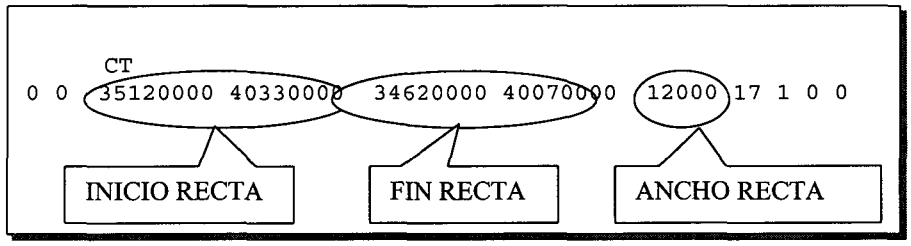


Fig. 6.4.3 Línea de una recta del archivo *Protel 1.5* del componente visto arriba.

El archivo Roën del componente de la figura 6.4.1 es el siguiente:

```
Última modificación el día1/01/96
Última resolución=0.2000000000,0,0
_COMPONENTE
TO-66 /1
TO-66
0.00000,1
Vacío
1150.00000,900.00000,100.00000,100.00000,28.00000,0
1150.00000,700.00000,100.00000,100.00000,28.00000,0
770.00000,800.00000,100.00000,100.00000,28.00000,0
1730.00000,800.00000,100.00000,100.00000,28.00000,0
_LINEAS
1150.00000,1150.00000,1350.00000,1150.00000
1150.00000,450.00000,1350.00000,450.00000
650.00000,690.00000,650.00000,890.00000
650.00000,690.00000,1150.00000,450.00000
1350.00000,450.00000,1850.00000,700.00000
1850.00000,700.00000,1850.00000,900.00000
1850.00000,900.00000,1350.00000,1150.00000
1150.00000,1150.00000,650.00000,890.00000
_ARCOS
1250.00000,800.00000,1500.00000,800.00000,1500.00000,800.
770.00000,800.00000,850.00000,800.00000,850.00000,800.00000
1730.00000,800.00000,1810.00000,800.00000,1810.00000,800.00
_CIRCULOS
_FIN_COMPONENTE
_PISTAS
_VIAS
_RELLENOS
_TEXTOS
_TALADROS_BORDE
_BORDE
_CONEXIONES
_FIN
```

Fig. 6.4.4 Estructura del archivo Roën 1.0.

6.5 FLUJO DE LA PARTE DE LA SIMULACIÓN

Los dibujos 6.5.1, 6.5.2 y 6.5.3 son los flujogramas de la parte del programa dedicado a la simulación. Se comentará partes puntuales de los mismos.

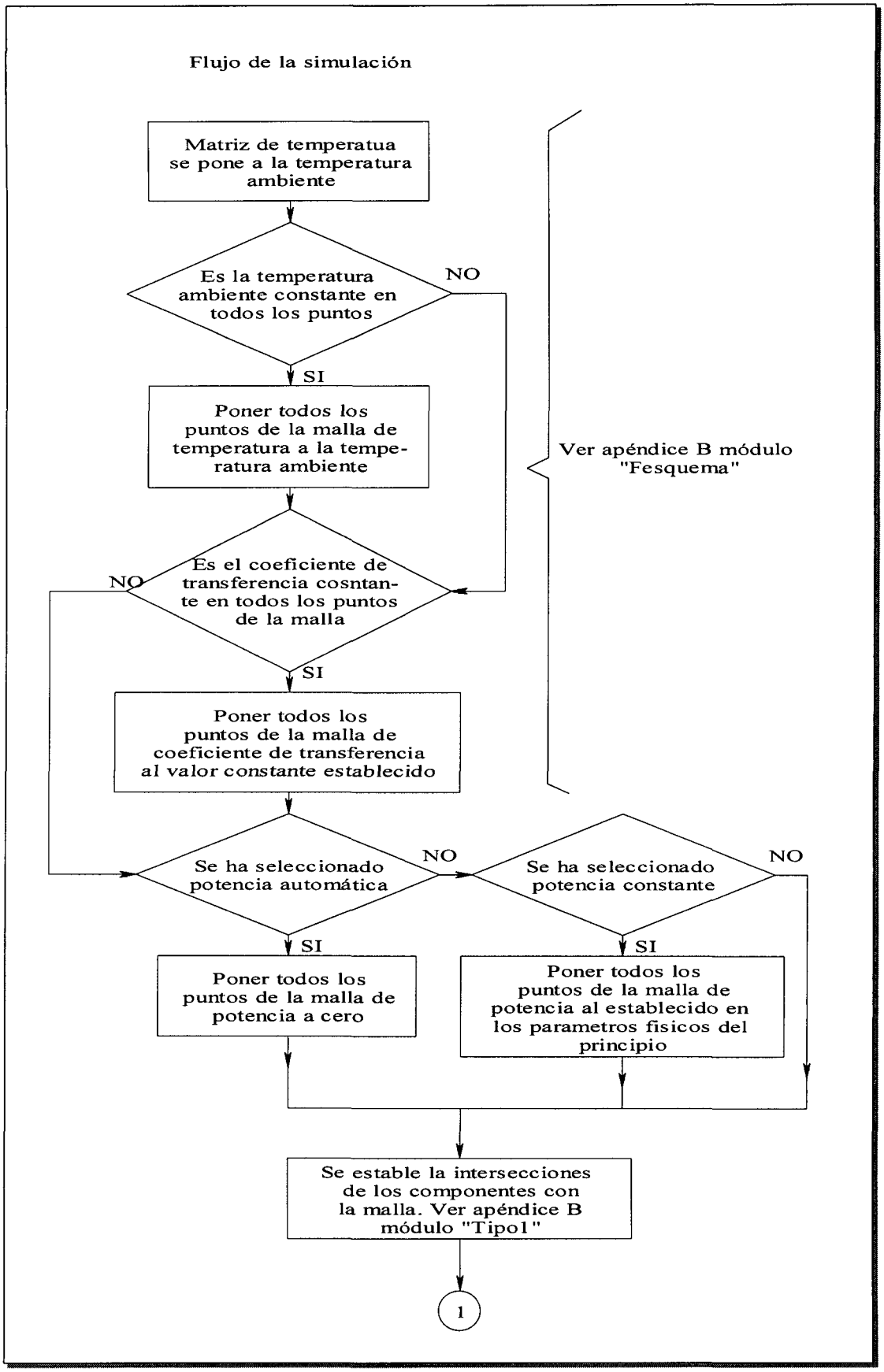


Fig. 6.5.1 Flujo de la simulación.

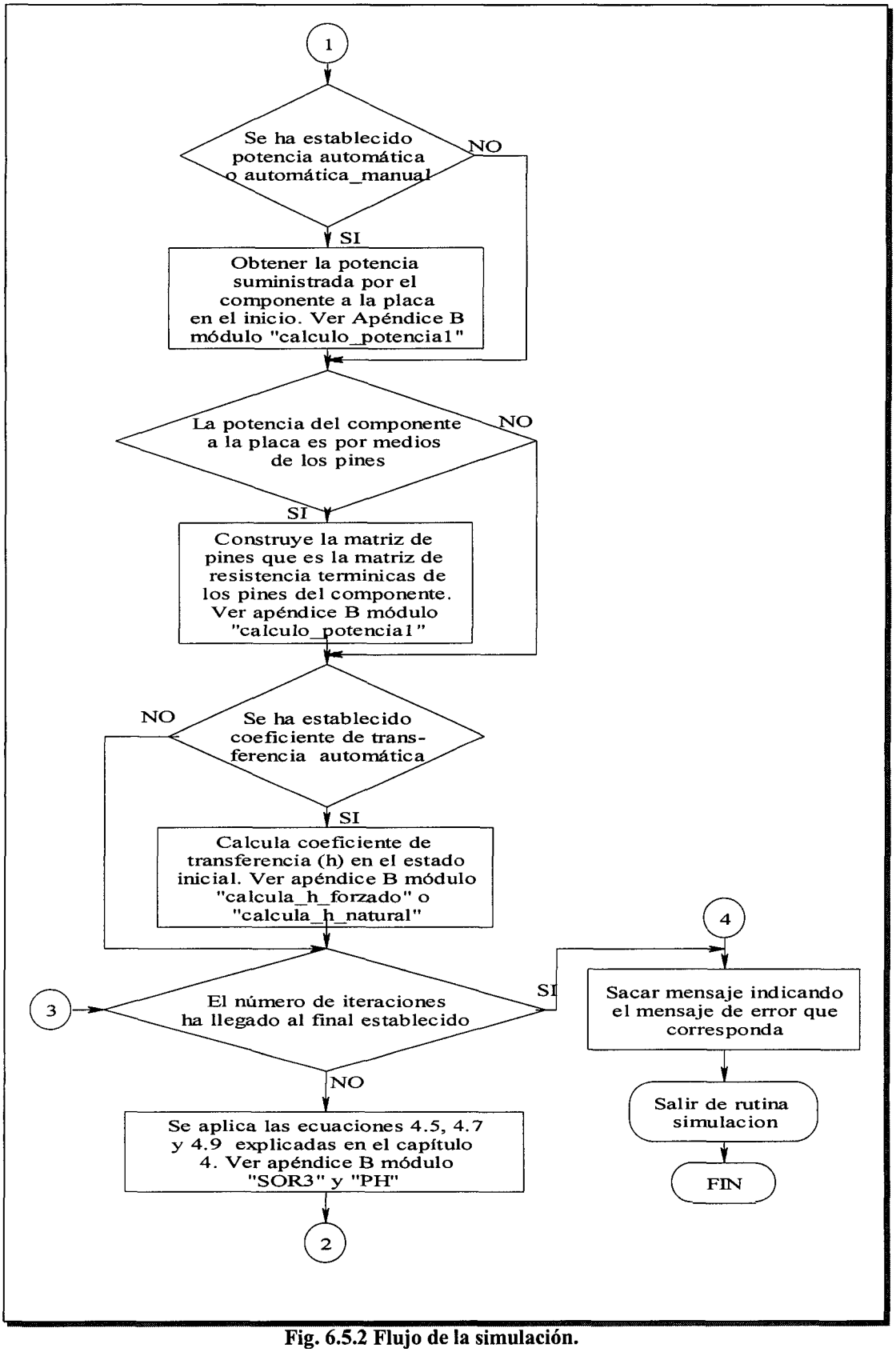


Fig. 6.5.2 Flujo de la simulación.

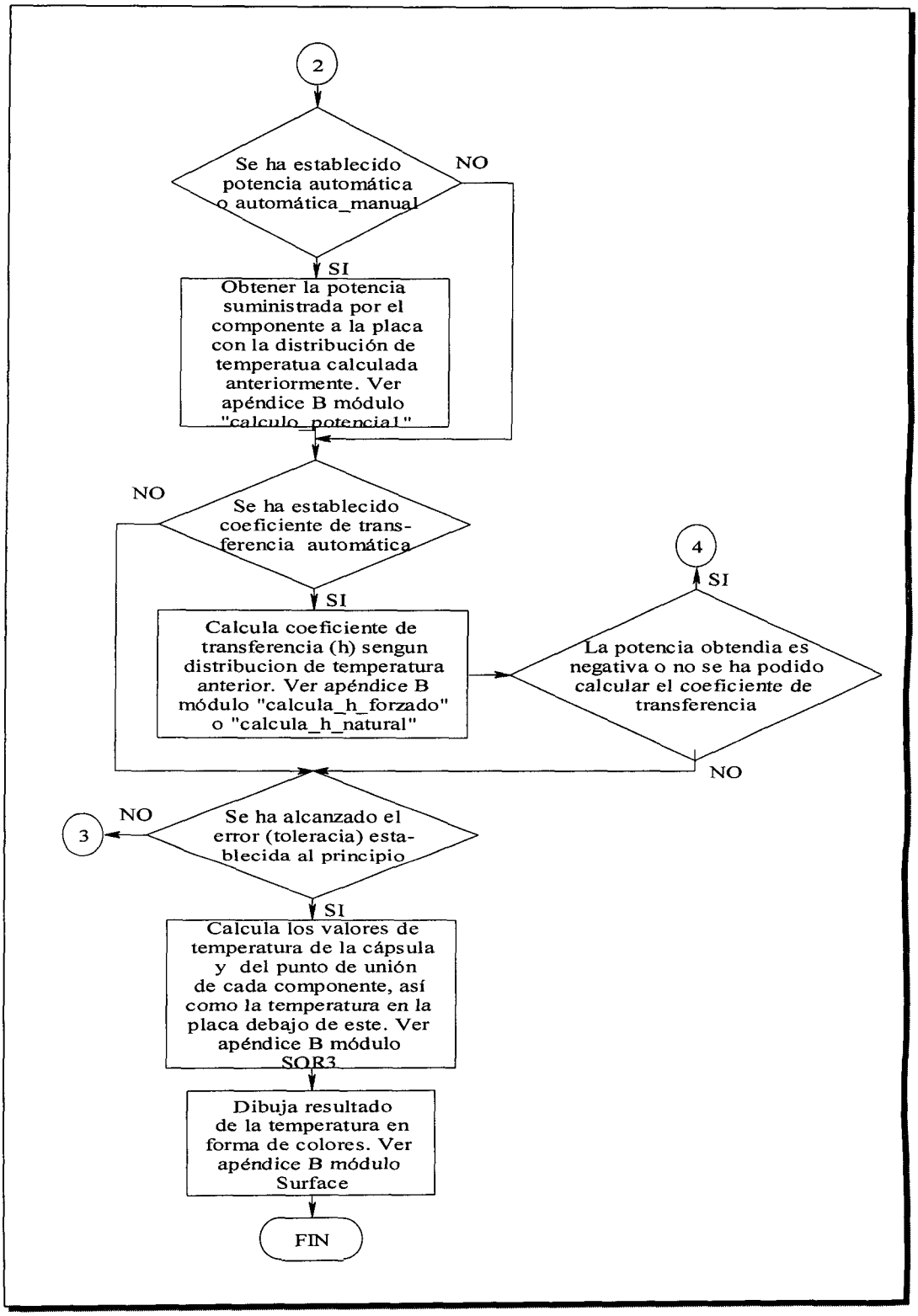


Fig. 6.5.3 Flujo de la simulación.

6.6 COMENTARIOS SOBRE LOS GRADIENTES DE COLOR

Las cuatro gradientes utilizados (cool, hot, gray y estándar) son los mismos que utiliza el programa de matemáticas *Matlab 5.0.*

El programa usa una paleta que puede ir de 9 a 255 colores/tonalidades para los cuatro gradientes excepto para estándar que va de 25 a 255 colores.

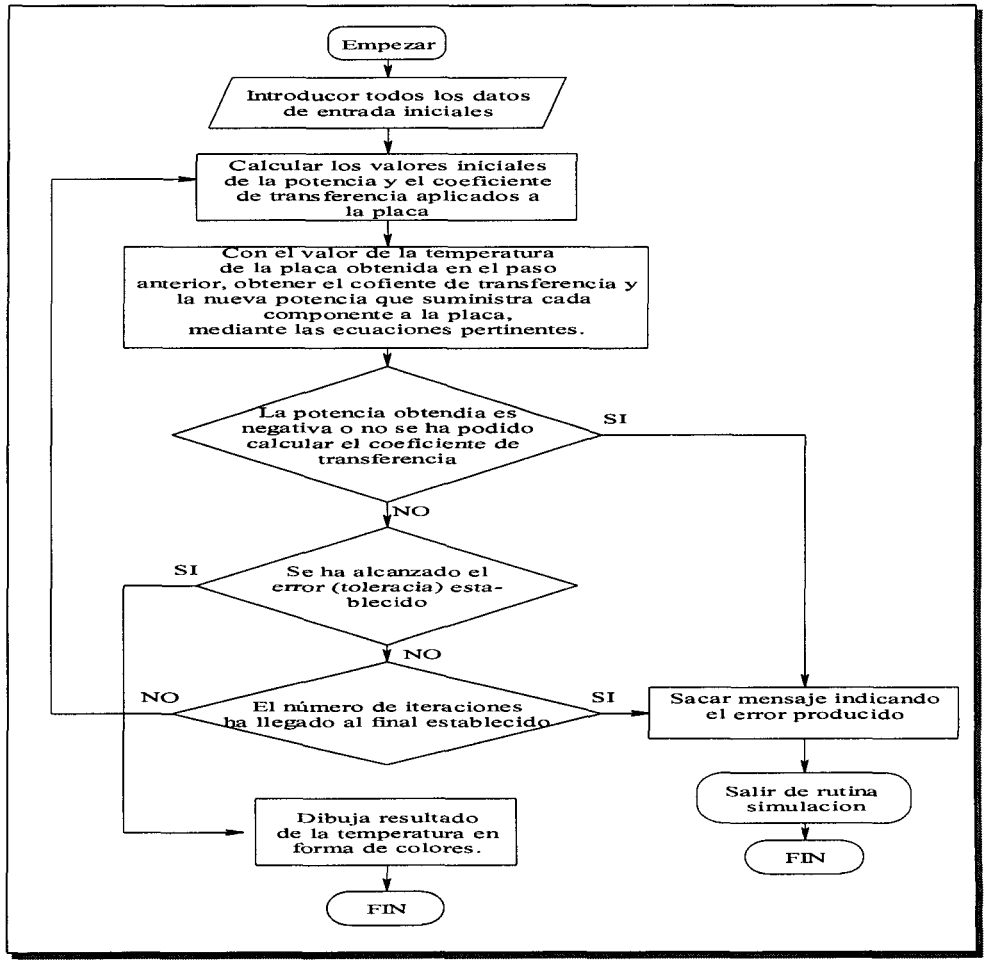
6.7 ALGORITMO PARA EL CÁLCULO DE LA TEMPERATURA

El método empleado para calcular las temperaturas de la placa es el capacitancia resistiva del capítulo 4, en concreto las ecuaciones 4.5, 4.7 y 4.9 en forma explícita, acelerando la convergencia mediante un método llamado SOR, el cual multiplica el interior que está entre paréntesis (tratado como residuo) de la ecuación explícita de la figura, por el factor ω ('Overrelaxation') cuyo valor optimo es la expresión de abajo, la cual sale de un estudio complejo de valores y autovalores.

$$T_{i,j}^{n+1} = T_{i,j}^{n} + \left(T_{i-1,j}^{n} - 4T_{i,j}^{n} + T_{i+1,j}^{n} + T_{i,j-1}^{n} + T_{i,j+1}^{n}\right)$$

$$T_{i,j}^{n+1} = T_{i,j}^{n} + \omega \cdot \left(T_{i-1,j}^{n} - 4T_{i,j}^{n} + T_{i+1,j}^{n} + T_{i,j-1}^{n} + T_{i,j+1}^{n}\right)$$

$$\omega = \frac{4}{2 + \sqrt{4 - \left(\cos\frac{\Pi}{fila} + \cos\frac{\Pi}{columna}\right)^2}}$$



Programa de simulación térmica termín 1.0

6.8 SOLUCIONES GRÁFICAS EN LA PROGRAMACIÓN

6.8.1 Técnicas de visualización de los componentes

Cada componente está definido como un registro, en el que se guardan los valores de rectas, curvas, círculos, etc. Los límites de cada componente están limitado por un rectángulo, de cual nos servimos para aplicar unos de los aceleradores gráficos que aquí se usan. Consiste en obtener la intersección de este rectángulo, con el recuadro que Windows utiliza para actualizar el dibujo en pantalla, a requerimiento de un movimiento de la barra de scroll de la ventana. Si existe intersección entre estos dos rectángulos se dibujará el componente, de lo contrario no se dibuja. De forma esquemática se verá mejor.

En el dibujo de la figura 6.8.1.1 representa una ventana típica de *Windows,* en la cual está distribuido los componentes (rectángulos 1 al 7), cuando se pulsa el botón 4 de la barra vertical, el recuadro 1 es el único que se actualiza, ya que el resto no se modifica para nada debido a que es el mismo. Luego cuando se hace la intersección de todos los componentes con el recuadro 1 se detecta que sólo hay que repintar el componente número 2, que es el único que tiene intersección con dicho recuadro. Para los restantes botones de las barras de movimiento se interpreta igual, pero ahora el recuadro que se compara en la intersección es otro, y el componente a dibujar probablemente será otro.

Esta es una técnica muy buena para acelerar la visualización de los componentes cuando se está moviendo la pantalla.

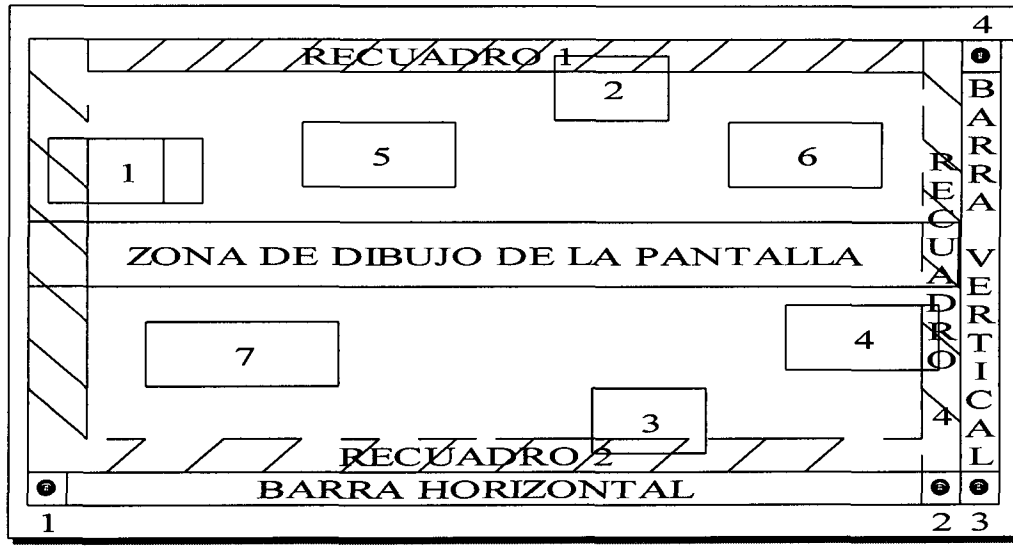


Fig. 6.8.1.1 Acelerar la visualización de los componentes.

6.8.2 Zoom de la pantalla

El zoom de la pantalla está muy logrado; se trata de un zoom que denomino aquí direccional, es decir, que es capaz de hacer un zoom sobre la parta de la pantalla en la que se encuentre situado el cursor.

Este Zoom funciona con las teclas "Re Pág" y "Av. Pág" y el cursor de la siguiente forma:

En la mayoría de los programas cuando se hace un zoom de la pantalla, las barras de desplazamiento no se modifican para nada, lo que el dibujo se esquina mucho o incluso desaparece de la pantalla. Para este zoom las barras se desplazan proporcional al nuevo zoom, y teniendo en cuenta la posición que tenía el curso antes de aplicarse el zoom, dando la sensación que el componente es el que se acerca a la pantalla y no se aleja de ella. Cuando se pulsa sobre la tecla "Re Pág" de producirá un aumento del zoom que será proporcional a potencias de dos (de 2 a 64). Si se pulsa la tecla "Av. Pág" se producirá una disminución de la pantalla proporcional a potencia de dos (de 1/64 a 64). Primero hay que obtener la posición que tendría la barra de movimiento de la pantalla a escala 1, a esta cantidad hay que restar o sumar (según sea barra vertical o horizontal) la posición del cursor. Luego se multiplica este último valor por el nuevo zoom y se le resta la posición del cursor de nuevo.

6.8.3 Trucos utilizados para el zoom y modos de pantallas utilizados

La utilización del zoom para dibujar las rectas, arcos, etc., fueron los siguientes:

Para dibujar las rectas escaladas sólo hay que pasar lo valores que están en pulgadas a pixeles (multiplican por el factor conversión PixelsPerInch), multiplicarlo por la escala y restarles la posición actual de la barra de scroll de la ventana. El modo de pantalla en este caso era *MM_ISOTROPIC,* es decir, pantalla en modo pixeles y punto de origen (0,0) en la esquina inferior izquierda. Para otros casos se usaba el modo de pantalla *MM_HIENGLISH*, es decir, pantalla en coordenadas de milésimas de pulgada y origen de ventana (0,0) en la esquina inferior izquierda. Las funciones anteriores son API´s de *Windows.*

```
X_entero:=round(recta.x1 * PixelsPerInch*escala-scroll_dibujo.x);
Y_entero:=round(recta.y1 * PixelsPerInch*escala-scroll_dibujo.y);
canvas.moveTo(x_entero,y_entero);
X_entero:=round(recta.x2 * PixelsPerInch*escala-scroll_dibujo.x);
Y_entero:=round(recta.y2 * PixelsPerInch*escala-scroll_dibujo.y);
canvas.lineto(x_entero,y_entero);
```

Fig. 6.8.3.1 Zoom con una recta.

8.3.4 Visualización de las matrices de colores

Para visualizar las matrices se usa un truco para actualizarla mucho más rápida en pantallla. Los cuadros de la malla se visualizan con la función de dibujar cuadrados porque para poderlos rellenar de colores es necesario saber la coordenada de la pantalla de cada uno de ellos. El truco es mejor verlo con un ejemplo:

En la figura 8.3.4.1 se ve la ventana del programa con una malla dibujada. Si queremos desplazar la pantalla hacia arriba pulsaremos el botón 4, al pulsar dicho botón, se hace la intersección entre el recuadro 1 y el rectángulo que limita la malla, del rectángulo intersección obtenido, se consigue la fila y columna del punto inicio y final de la parte de la malla correspondiente al rectángulo intersección, mediante un procedimiento sencillo de correspondencia entre coordenadas de pantalla y filas y columnas de la malla, de ese modo sólo se dibujará la parte de la malla limitada por la filas y columnas conseguidas anteriormente y no toda la malla. Este método acelera la visualización de la malla de una manera eficiente y rápida.

Fig 8.3.4.1 Método especial para visualizar las mallas del programa.

6.9 DISEÑO DE LA AYUDA

6.9.1 Introducción

Lo normal en un programa, es que disponga de una ayuda para obtener información sobre algo mientras se está utilizando dicho programa, sin tener que irse al manual de este, si se dispone de él, o buscar información en algún otro libro.

Pues *Termín 1.0*, no tenía porque ser menos y se le incorporó una ayuda. Se trata de una ayuda sensible al movimiento del ratón, en la cual da la impresión al usuario de que está trabajando directamente con el programa.

Para poder dotar de un sistema de ayuda como el que se implementó en *Termín 1.0*, se necesitó básicamente, cuatro elementos: un editor donde escribir el texto de ayuda, que debe estar en formato *RTF*; un compilador de ayudas,

Delphi 2 incorpora el *Help Compiler de Microsoft versión 4.10*, que se encuentra en el directorio *HELP\TOOLS* de *Delphi 2* y es una aplicación de Windows; el programa *WINDHELP*, que será el encargado de visualizar el contenido de la ayuda y el programa *Hotspot editor*, que no se distribuye con *Delphi 2*, pero lo podemos encontrar junto con *Borland C++*, Borland Pascal, etc. Con este programa podemos crear en la ayuda hiper-gráficos, nombre con el que se denomina a los gráficos que incluyen múltiples enlaces, situados en distintas posiciones de la imagen.

El editor más utilizado para crear los ficheros de texto de ayuda es Word, aunque puede usar cualquier otro editor que soporte el formato RTF, ya que el compilador de ayudas sólo lee dicho formato. Aunque Windows 95 o 98 existe un editor, llamado WordPad, que facilita la creación de archivos en formato RTF, este programa no le será útil para escribir ayudas, ya que no soporta todos los formatos necesarios para ello. Decir también que el editor utilizado es el Word de Office95 ya que el de Office97 daba problemas y no llegaba a compilar el fichero RTF.

Un archivo de texto en formato RTF no es directamente utilizable como ayuda de Windows, sino que es necesario compilarlo para generar un archivo de ayuda. De hecho, un archivo de ayuda puede incluir múltiples ficheros de texto en formato RTF, pero en nuestro caso solo incluye un solo fichero.

Para obtener una información detallada de cómo se crea y de las opciones disponible para el fichero RTF, consultar el libro *Programación avanzada con Delphi 2.0* de la página 493 a la 518 (capítulo 19).

6.9.2 Shed.exe

Este es el programa para crear los hiper-gráficos, es una aplicación que dispone Borland C++ 3.1. La interfaz de este programa es la mostrada en la figura 6.9.2.1

Fig.6.9.2.1 Programa para la creación de hiper-gráficos.

Antes de llamar a este programa, se debe tener el archivo con extensión *bmp* ya creado, o lo que es lo mismo disponer de la imagen a la cual se va a convertir en

hiper-gráfico. Se va al menú *File* y se elige el comando *Open*, con lo cual se carga la imagen que desea convertir en hiper-gráfico, que aparecerá en una ventana.

Cada enlace de un hiper-gráfico estará asociado a una determinada porción de la imagen, que tendremos que definir situando el cursor del ratón en una esquina, pulsando el botón izquierdo y a continuación desplazando el cursor hasta la esquina opuesta. Al hacerlo, verá cómo la zona queda delimitada por un recuadro. El siguiente paso será fijar los atributos del enlace, para lo que realizaremos una doble pulsación en cualquier punto de ese recuadro, provocando la aparición de la ventana que puede ver en la figura 6.9.2.2 En ella tendrá que facilitar el identificador de la página de ayuda a la que debe saltar este enlace, y que hemos definido al principio de cada página con una nota al pie y la marca personal #. En la lista desplegable *Type* se indicará si al pulsar sobre este enlace se debe saltar a esa página de ayuda o bien se debe visualizar una ventana emergente, *jump* o *pop-up* respectivamente. En el apartado *Attribute* se elegirá entre *Visible* o *Invisible*, según se desee que el recuadro que se ha trazado sea visible o no durante el uso del archivo de ayuda. Habitualmente los hiper-gráficos no son visibles, y su presencia es denotada por el cambio del icono del ratón al pasar sobre ellos.

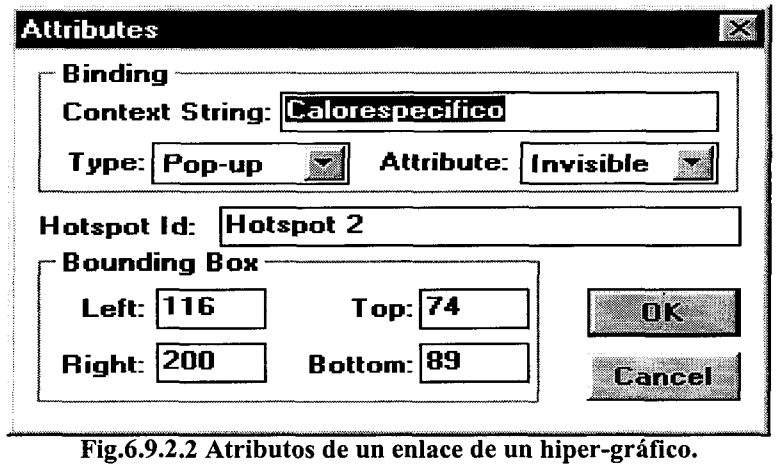


Fig.6.9.2.2 Atributos de un enlace de un hiper-gráfico.

Termin 4.0
Simulador Térmico

Cuarta Parte

Aspectos formales del proyecto

Termin 1.0
Simulador Térmico

Capítulo 7

Pliego de condiciones

7 Pliego de condiciones

7.1 INTRODUCCIÓN

El pliego de condiciones es, desde el punto de vista legal y contractual, el documento más importante del proyecto a la hora de su ejecución material.

El pliego de condiciones regula las relaciones entre el propietario, promotor del proyecto y los contratistas que lo van a ejecutar y deberá contener toda la información necesaria para que esas relaciones sean lo más fructíferas posible, máxime teniendo en cuenta la importancia de la componente económica en las mismas.

El pliego se desarrollará en tres partes:

a) Pliego de Condiciones Generales.

- Legales.
- Administrativas.

b) Pliego de Prescripciones Técnicas Particulares.

- Especificaciones de materiales y equipos.
- Especificaciones de ejecución.

c) Pliego de Cláusulas Administrativas Particulares.

- Condiciones económicas.

a) Pliego de Condiciones Generales.

Este apartado contendrá fundamentalmente una descripción general del contenido del proyecto, sus características principales, los aspectos legales y administrativos a tener en cuenta por los futuros.

b) ***Pliego de Prescripciones Técnicas Particulares.***

b.1 Especificaciones de materiales y equipos.

Aquí aparecerán perfectamente definidos todos los materiales, equipos, máquinas, instalaciones, etc. que constituyen el proyecto.

La definición se hará en función de códigos y reglamentos reconocidos como válidos para el proyecto, y en aquellos que no sean de aplicación se definirán expresamente todos los elementos que sean necesarios.

Las especificaciones harán referencia a Normas y Reglamentos oficiales y oficiosos españoles y extranjeras o internacionales.

Se hará especial referencia a la homologación de todos los materiales, equipos y maquinaria incluidos en el proyecto.

b.2 Especificaciones de ejecución.

La ejecución material del proyecto, su fabricación o construcción a partir de los materiales del proyecto especificados en el apartado anterior, se definirán exactamente en este apartado.

Si en el punto anterior se concreta lo ***QUE*** se va a utilizar en el proyecto, en éste hay que definir ***COMO*** se va a utilizar.

c) Pliego de Cláusulas Administrativas Particulares.

En este apartado se hará referencia directa a la forma de medir las obras ejecutadas, valorarlas y abonarlas.

7.2 CONDICIONES GENERALES

- Captura de los archivos de *Protel 1.5* y *Roën 1.0.*
- Simulación térmica de las placas de circuitos impresos.
- Posibilidad de dar el resultado de forma de mapas de colores o numéricamente.

7.3 PRESCRIPCIONES TÉCNICAS PARTICULARES

Los requisitos básicos del sistema que exige esta aplicación son los siguientes:

- *Procesador requerido:* Pentium/166Mhz o superior.
- *Sistema operativo:* Windows 95 o superior.
- *Memoria requerida:* 16 Mb o superior.
- *Tarjeta gráfica:* cualquiera que soporte 256 colores, recomendada 16 bits.
- *Resolución mínima de pantalla:* 640x480.
- *Fuente de letra óptima:* Standard de Windows.
- *Mínimo espacio de disco duro:* 2Mb.

Termin 1.0
Simulador Térmico

Capítulo 8

Instalación y configuración

8 Instalación y configuración

Para instalar *Termín 1.0* en un ordenador se necesita que disponga de la siguiente configuración como mínimo:

Procesador requerido: Pentium/166Mhz o superior.
Sistema operativo: Windows 95 o superior.
Memoria requerida: 16 Mb o superior.
Tarjeta gráfica: Mínimo 256 colores, recomendada 16 bits.
Resolución mínima de pantalla: 640×480.
Fuente de letra óptima: Standard de Windows.
Mínimo espacio de disco duro: 2Mb.

El simulador de Térmico *Termín 1.0* se suministra con un disco de 1,44MB con un programa de instalación. Para instalarlo sólo hay que seguir los pasos que nos va indicando el programa de instalación.

Una vez instalado, la estructura de directorio que crea en la unidad donde se ha instalado (por defecto es la unidad C:\), es la mostrada en la figura 1.

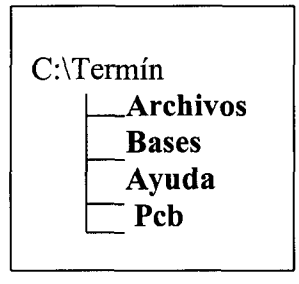


Fig. 1 Estructura de directorios

Por lo tanto, ya está disponible para su utilización con solo invocar al archivo *Termín.exe* o irse al grupo de programa "*Simulador Térmico*" y seleccionar *Termín*.

Por defecto el programa se instalará en la unidad *C* y en el directorio *Termín*, que se creará automáticamente en el proceso de instalación.

Termin 1.0
Simulador Térmico

Capítulo 9

Presupuesto

9 Presupuesto

9.1 INTRODUCCIÓN

Para el cálculo del presupuesto del proyecto, se ha seguido la Propuesta de barremos orientativos pera el cálculo de honorarios del *Colegio Oficial de Ingenieros Técnicos de Telecomunicación, Circular 011/04-11-96.*

Esta propuesta establece que para "Trabajos tarifados por tiempo empleados" se aplique la siguiente fórmula:

$$H=H_n*9700+He*10500$$

Siendo:

- H: honorarios
- H_n: horas en jornada normal
- H_e: horas fuera de la jornada normal

Los honorarios que se obtengan por la aplicación de la clave "H", se reducirán a medida que aumente el número de horas, a cuyo efecto serán multiplicados por los coeficientes reductores, con arreglo a la escala que muestra la Tabla 1:

HORAS	COEFICIENTE
Hasta 36 horas	C=1
Exceso de 36 horas hasta 72	C=0,9
Exceso de 72 horas hasta 108	C=0,8
Exceso de 108 horas hasta 144	C=0,7
Exceso de 144 horas hasta 180	C=0,65
Exceso de 180 horas hasta 360	C=0,6
Exceso de 360 horas hasta 540	C=0,55
Exceso de 540 horas hasta 720	C=0,5
Exceso de 720 horas hasta 1080	C=0,45
Exceso de 1080 horas	C=0,4

Tabla 1. Coeficientes de reducción para trabajos valorados por tiempo empleado.

En nuestro caso las horas dedicadas a realizar este proyecto, se han visto repartidas de la siguiente forma:

HORAS TOTALES	CANTIDAD
Dedicadas a Delphi 2.0	150h
Dedicadas a buscar información	250h
Dedicadas a otros programas	25h
Dedicadas a la realización del programa	1875h
Dedicadas a la memoria, manual y presentación	250h
Dedicadas a la realización de la ayuda	50h

Tabla 2 Horas dedicadas al proyecto.

Por lo que ha sido un total de 2600 horas, repartidas, según un pequeño sondeo de lo que ha sido este periodo de trabajo en:

- 2000 horas en jornada normal.
- 600 horas fuera de la jornada normal

$$H = (2000 * 9700) + (600 * 10500) = 25700000$$
$$H*C = 25700000 * 0,4 = 10280000$$

Total: 10.280.000 Ptas.

9.2 PRESUPUESTO DE EXPLOTACIÓN

En este apartado se intenta reflejar el coste de programa atendiendo al resultado de presupuesto de desarrollo, y al posible mercado que pudiera tener dicha aplicación.

En principio, al carecer de datos más específicos, tomaremos referencia de una cuota de mercado de 100 licencias (centros de enseñanza universitaria, profesionales, etc.). De esta manera se establece que el coste por cada uno de ellos será:

	OPERACIÓN	CANTIDAD(Ptas.)
PRECIO POR UNIDAD	10.280.000/100	102.800
GASTOS DIVERSOS	10% de 102.800	10.280
TOTAL POR LICENCIA	102.800+10.280	113.080

Tabla 3. Coste de programa por licencia

Por lo tanto el precio por licencia del programa será de:
CIENTO TRECEMIL OCHENTA (113.080 Ptas).
FIRMADO
Luis F. Guerra García.

Termin 1.0
Simulador Térmico

Capítulo 10

Protección legal

10 Protección legal

10.1 PROPIEDAD INTELECTUAL

La propiedad intelectual, que comprende las creaciones literarias, científicas, artísticas, programas de ordenador, etc., se diferencia de la industrial en que en aquella el derecho surge por la creación, sin ser necesario acudir a ningún registro para el nacimiento del derecho. A pesar de que no es preciso el registro para poseer el derecho, la Ley regula la existencia de un *Registro de la Propiedad Intelectual* en el que inscribir este tipo de creaciones.

La propiedad intelectual se regula en España por el *Real Decreto Legislativo 1/1996*, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizado, aclarando y armonizando las disposiciones legales vigentes sobre la materia. La Ley de Propiedad Intelectual del 11 de noviembre de 1987 (BOE de 7/11/1987), y en el artículo 10 recoge las materias de que es objeto:

1. Son objeto de propiedad intelectual todas las creaciones originales literarias, artísticas o científicas expresadas por cualquier medio o soporte, tangible o intangible, actualmente conocido o que se invente en el futuro, comprendiéndose entre ellas:

a) Los libros, folletos, impresos, epistolarios, escritos, discursos y alocuciones, conferencias, informes forenses, explicaciones de cátedra y cualesquiera otras obras de la misma naturaleza.

b) Las composiciones musicales, con o sin letra.

c) Las obras dramáticas y dramático-musicales, las coreografías, las pantomimas y, en general, las obras teatrales.

d) Las obras cinematográficas y cualesquiera otras obras audiovisuales.

e) Las esculturas y las obras de pintura, dibujo, grabado, litografía, y las historietas gráficas, tebeos o cómics, así como sus ensayos o bocetos y las demás obras plásticas, sean o no aplicadas.

f) Los proyectos, planos, maquetas y diseños de obras arquitectónicas y de ingeniería.

g) Los gráficos, mapas y diseño relativos a la topografía, la geografía y, en general, a la ciencia.

h) Las obras fotográficas y las expresadas por procedimiento análogo a la fotografía.

i) Los programas de ordenador.

2. El título de una obra, cuando sea original, quedará protegido como parte de ella.

10.2 LA PROTECCIÓN DE LOS PROGRAMAS DE ORDENADOR

Entre los diversos resultados de la investigación está el soporte lógico (*software*), un término más amplio que el utilizado normalmente de programa de ordenador, al comprender no sólo a éste, sino también manuales de utilización, los ordinogramas (representaciones esquemáticas de las secuencias de ejecución de un proceso), etc. Como dijimos anteriormente, los programas de ordenador no son patentables en sí mismo, ya que su protección como tales se encuentran en la propiedad intelectual, en los artículos 95 a 104 del Real Decreto Legislativo. Estos pueden ser el objeto de un proyecto de investigación.

Existen programas que, por su complejidad o precisión, necesitan el desarrollo de algoritmos complejos y gran cantidad de secuencias y pruebas. Además, pueden crearse programas como herramientas necesarias para llevar a efecto otras investigaciones o manejo de instrumental de precisión, no siendo el programa el objeto central del proyecto de investigación. Sin embargo, una vez desarrollado, un programa puede servir para que otros lo utilicen, con lo que tal programa adquiere entidad propia como producto para transferir.

De la misma forma que las anteriores materias, es necesario protegerlos para lograr una eficaz transferencia.

Como se dijo, los derechos de propiedad nacen por la simple creación, sin necesidad de hacer ningún registro. De todas formas, es conveniente comunicar a los terceros que están reservados los derechos de explotación del programa. Para ello basta con poner al principio de la obra (en el caso del programa, en la primera pantalla cuando arranque el programa, en los manuales en una de sus primeras páginas, y en todos los ordinogramas del mismo) el conocido símbolo © con precisión del lugar y año de divulgación. Sin embargo, es importante reforzar la protección de otras maneras. Se debe tener en cuenta que un programa de ordenador una vez creado y puesto en funcionamiento es muy fácil de copiar, por ello es conveniente reforzar la protección, para lo que se puede utilizar diversas vías:

a) Físicas y lógicas, consistentes en la utilización obligada de un mecanismo (disco, llave, etc.) al margen del programa, para poder ejecutarlo.

b) Regístrales, que consisten en preconstituir una prueba de autoría del programa, con objeto de poder utilizarla en caso de que alguien esté

a) Físicas y lógicas, consistentes en la utilización obligada de un mecanismo (disco, llave, etc.) al margen del programa, para poder ejecutarlo.

b) Registrales, que consisten en preconstituir una prueba de autoría del programa, con objeto de poder utilizarla en caso de que alguien esté explotando el programa sin consentimiento del legítimo titular de los derechos. Ello se puede hacer de diversas maneras, entre ellas:

Primera: Inscripción en el *Registro General de la Propiedad Intelectual.* Se crearía así una presunción de titularidad que los terceros tendrían que destruir.

Segunda: Registro en documento público ante notario, mediante el que se da fe de que en tal fecha se le presenta un programa, que el notario incorpora a su protocolo o cuyo contenido introduce en un sobre lacrado y sellado, para poder utilizarlo como prueba en caso de tener que acudir a los Tribunales.

➢ **Real Decreto Legislativo 1/1996**, por el que se aprueba el texto refundido de la ley de Propiedad Intelectual, BOE de 7 de noviembre de 1987.

Termin 1.0
Simulador Térmico

Capítulo 11

Aspectos a mejorar

11 Aspectos a mejorar

11.1 INTRODUCCIÓN

La integración de este capítulo en el proyecto, tiene como fin el poner de manifiesto las posibles mejoras en el futuro. La imposibilidad de realización de estas mejoras en esta versión de *Termiín 1.0*, es debido por una parte, a que al ser la primera versión de este programa de simulación, es casi imposible lograrlo todo en su nacimiento y por otra parte, el cálculo que entraña realizar el estudio de las ecuaciones de transferencia de calor en el componente y en el medio ambiente son de una complejidad muy alta.

11.2 ASPECTOS

Los distintos aspectos que se pueden mejorar son:

- Poder ampliar el estudio de las ecuaciones de malla de temperatura al componente y medio ambiente.
- Ampliar la librería de los datos físicos de los componentes y placa.
- Incorporar nuevas herramientas de cálculo como *Elementos Finitos*.
- Incorporación de nuevas herramientas gráficas para la visualización de las mallas de temperatura.
- Incorporación de un estudio estadístico para calcular la vida media del componente en función de la temperatura de trabajo.
- Incorporación de nuevos análisis, como respuesta impulsiva.
- Posibilidad de cargar en *Termín 1.0* circuitos creados por los distintos programas de diseño de PCBs como: Tango, Orcad, etc.
- Todos aquellas posibles mejoras que puedan aparecer tanto en el entorno, simulación, algoritmo de cálculo, etc. que puedan mejorar lo que ya está hecho.

Termin 1.0
Simulador Térmico

Apéndice A

Se trata del ***Manual de Usuario.*** En el se explica como manejarlo y las posibilidades de este programa.

INDICE DEL APÉNDICE A

A.0 Introducción

A.0.1 SIMULADOSR TÉRMICO DE PLACAS

El ***Simulador Térmico de placas*** *Termín 1.0* permite simular las placas de circuito impreso PCB procedentes de los programas *Protel 1.5* en forma de texto y *Roën 1.0* con una distribución de componentes dada. La simulación analiza la malla de nodos de la placa visualizando el resultado en forma de mapa de colores o numéricamente.

Esto permite tener un calculo aproximado del calor que disiparán cada componentes mediante la ayuda de un ordenador personal.

En los posteriores capítulos veremos información detallada sobre el manejo de *Termín 1.0*, tanto en la parte de preparación de los datos de los componentes, placa y medio ambiente, como en la parte de simulación.

A.0.2 INSTALACIÓN

Para instalar *Termín 1.0* en un ordenador se necesita que disponga de la siguiente configuración como mínimo:

- *Procesador requerido:* Pentium/166Mhz o superior.
- *Sistema operativo:* Windows 95 o superior.
- *Memoria requerida:* 16 Mb o superior.
- *Tarjeta gráfica:* Mínimo 256 colores, recomendada 16 bits.
- *Resolución mínima de pantalla:* 640×480.
- *Fuente de letra óptima:* Standard de Windows.
- *Mínimo espacio de disco duro:* 2Mb.

El simulador de Térmico *Termín 1.0* se suministra con un disco de 1,44MB con un programa de instalación. Para instalarlo sólo hay que seguir los pasos que nos va indicando el programa de instalación.

Una vez instalado, la estructura de directorio que crea en la unidad donde se ha instalado (por defecto es la unidad C:\), es la mostrada en la figura 1.

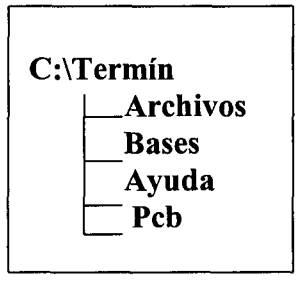


Fig. 1 Estructura de directorios

Por lo tanto, ya está disponible para su utilización con solo invocar al archivo *Termín.exe* o irse al grupo de programa "*Simulador Térmico*" y seleccionar *Termín*.

Por defecto el programa se instalará en la unidad C y en el directorio *Termín*, que se creará automáticamente en el proceso de instalación.

A.1 Pantalla general del programa

A.1 PANTALLA DE TERMÍN

La ventana principal de **Termín 1.0** esta dividida en varias áreas, la figura A.1.3.1 muestra como es la ventana principal y sus áreas:

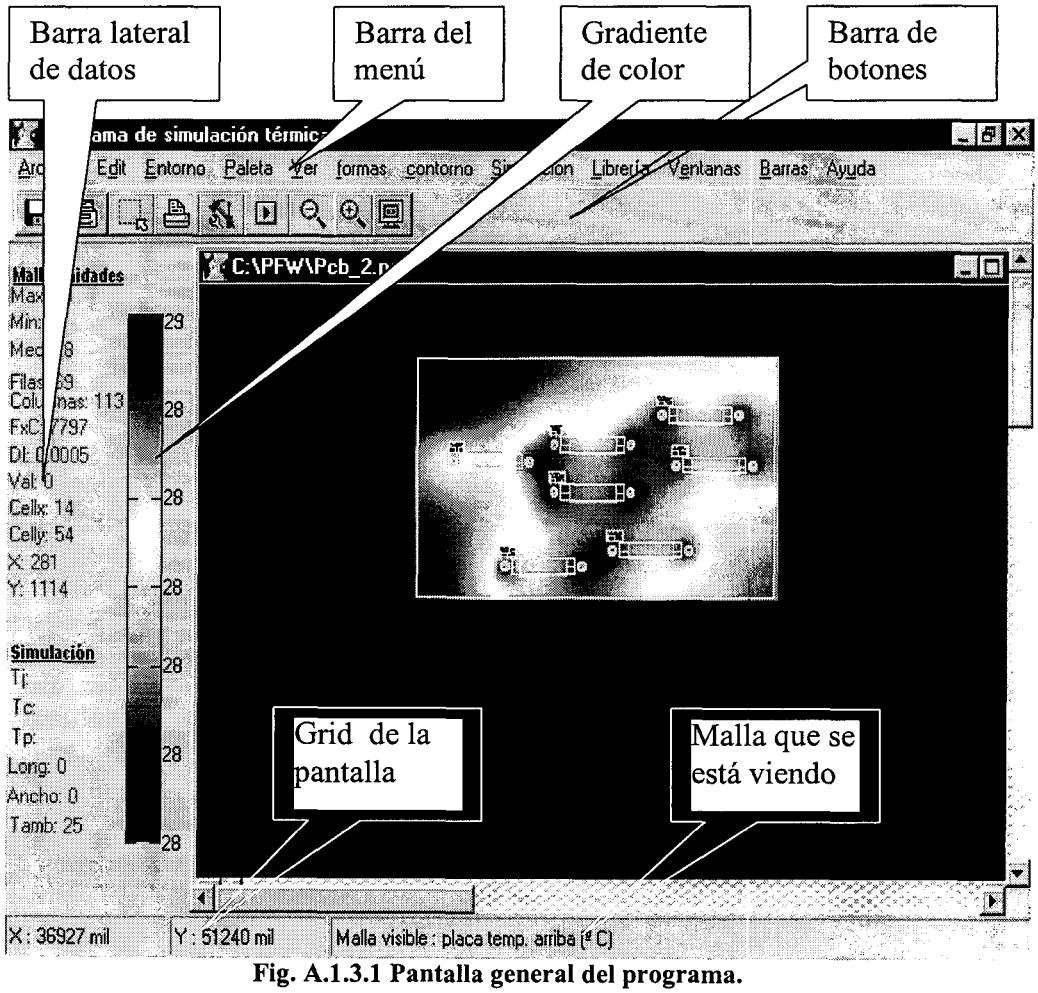


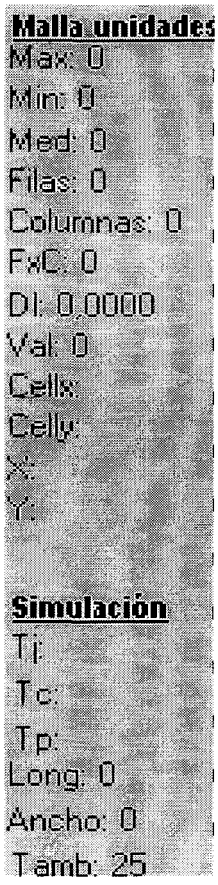
Fig. A.1.3.1 Pantalla general del programa.

A.2 Barras laterales

A.2 BARRAS LATERALES

A.2.1 Barra lateral de datos de la malla activa

Esta Barra ofrece de forma resumida los datos más importante de la malla activa (la que se está viendo en pantalla). Para cambiar la precisión de los datos hay que picar dos veces sobre la barra, los valores son los siguientes:



- **Max**: El máximo valor que tiene la malla.
- **Min:** El mínimo valor que tiene la malla.
- **Med:** Es la media de todos los valores distintos de cero.
- **Filas:** Número de filas de la malla.
- **Columna:** Número de columnas de la malla.
- **FxC:** Densidad de malla.
- **Dl:** Lado del cuadrado de la retícula de la malla, ver figura A.2.1 de abajo.
- **Val:** Es el valor número de cada celda al moverse con el ratón.
- **Cellx:** Valor de la celda x al moverse con el ratón.
- **Celly:** Valor de la celda y al moverse con el ratón.
- **X:** Coordenada x dentro del rectángulo de la malla.
- **Y:** Coordenada y dentro del rectángulo de la malla.
- **Tj:** Temperatura de la unión (hay que pulsar tecla F11).
- **Tc:** Temperatura de la cápsula (hay que pulsar tacla F11).
- **Tp:** Temperatura de la placa (hay que pulsar la tecla F11).
- **Long:** Longitud del rectángulo que forma la malla.
- **Ancho:** Ancho del rectángulo que forma la malla.
- **Tamb:** Temperatura ambiente.

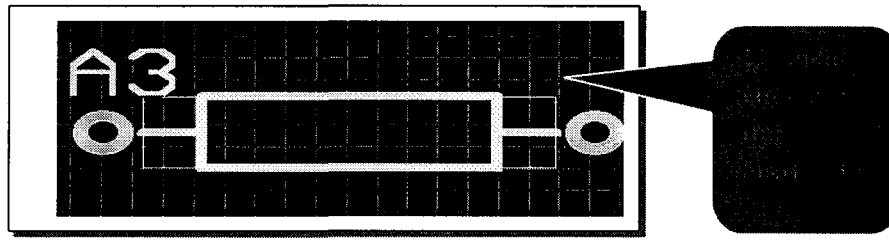
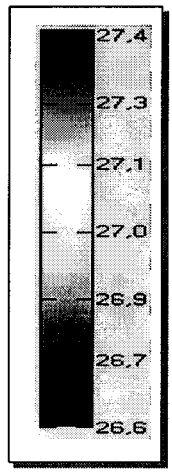


Fig. A.2.1 Imagen de parte de una malla encima de una resistencia.

A.2.2 Barra lateral de escala de colores

Escala.



La escala está dividida en el número de *tonalidades*. Para cambiar la precisión de escala hay que picar dos veces sobre la escala. La escala siempre hace referencia a la malla activa (visible en pantalla). Si se pica con el curso dentro del dibujo de colores saldrá un pequeño rectángulo en la parte inferior indicando el valor numérico del color que esta debajo del cursor.

A.3 Barra de menú

A.3 BARRA DE MENÚ

A.3.1 MENÚ ARCHIVO

A.3.1.1 Salvar 

Con esta opción estamos salvando las características térmicas de los componentes. Para ello el nombre del archivo debe ser mismo que el de la placa, pero con extensión txt, de ese modo al abrir la placa por segunda vez los datos térmicos que se cogen no son los de la librería, sino los específicos de esa placa. El botón de la barra de botones, desempeña la misma función.

Ejemplo de Parte de un archivo de datos térmicos:

Termín 1.0;
autor: Luis F. Guerra García

(Datos de la placa y medio ambiente):

INICIO_DATA_PLACA
V=Velocidad_viento (m/s):
0,0000
K=conductividad (W/m°C):
3,0000
p=desidad de temperatura (Kg/m³):
2,0000
c=Calor especifico (KJ/Kg °C):
1000,0000
HA=Coeficiente de transferencia de calor a lo alto de la placa (W/m² °C):
10,0000
HB=Coeficiente de transferencia de calor bajo la placa (W/m² °C):
10,0000
e=espesor (m):
0,0010
delta=cuadrícula de la malla (m):
0,0008
FIN_DATA_PLACA

INICIO_DATA_COMPONENTES

_COMPONENTE
DIP14
_REFERENCIA
A1
_POTENCIA (W) *0,3500
_Tj (°C) *29,2035

```
_Tc (ºC)            *25,0035
_Tp (ºC)            *0,0000
_Q(W/m²)            *60,4805
_K (W/mºC)          *0,0000
_RTHJC (W/m)        *12,0000
_RTHCA (W/m)        *12,0000
_RTHCP (W/m)        *12,0000
_RTHP (W/m)         *67,5775
_LARGO (W/m)        *0,0100
_ANCHO (W/m)        *0,0050
_ESPESOR (W/m)      *0,0030
_RTHPIN (W/m)       *2272,7273
_ESPESOR_PIN (m)    *0,0010
_LARGO_PIN (m)      *0,0100
_ANCHO_PIN (m)      *0,0020
_CONDUCTIVIDAD_PIN  *2,2000
FIN_COMPONENTE
```

A.3.1.2 Abrir 

Los archivos con los que se pueden realizar la simulación térmica son los procedentes de los programas de diseño de PCB´s llamado **Protel 1.5** en formato txt y **Roën 1.0**, programa este último, hecho por el alumno del mismo nombre, como proyecto fin de carrera, en la ULPGC. El botón de la barra de botones, desempeña la misma función.

A.3.2 MENÚ EDIT

A.3.2.1 Mover

Distintos tipos de movimientos, en esta versión sólo está implementado la de mover el componente. El movimiento puede ser con el ratón o con las teclas. El movimiento está ligado con la *rejilla cursor* ver apartado A.3.3.1

A.3.2.2 Poner datos manualmente

Permite introducir datos manualmente en la malla que en ese momento está activada (visible en pantalla), es decir, potencia, y coeficiente de transferencia. La forma de introducir estos datos es marcando con el botón izquierdo del ratón un rectángulo, y nuevamente el botón izquierdo cuando se tenga el rectángulo deseado. Tras lo cual saldrá una pantalla indicando las celdas que se han marcado, se introduce en el campo valor el nuevo valor para esas celdas; el nombre de la malla seleccionada se refleja en el título de la ventana abierta.

Tras introducir el dato se pulsan el botón *o.k.*, tras lo cual se puede seguir marcando rectángulos para introducir nuevos datos, o desactivar la opción con el botón derecho del ratón.

A.3.2.3 Rectificar rectángulo superficie

Permite modificar el rectángulo que tiene asociado cada componte, en relación con la superficie de contacto del componente con la placa. Valores estos

que se pueden ver pulsando dos veces en el componente y abriendo la carpeta de *dimensiones*. Estos valores estarán desactivados si la potencia suministrada a la placa está seleccionada a través de los pines.

La forma de rectificar este rectángulo es pulsando el botón izquierdo del ratón, marcar la nueva superficie y de nuevo el botón izquierdo. Para desactivar la opción, pulsar el botón derecho.

A.3.3 MENÚ ENTORNO

A.3.3.1 Modificar rejilla cursor

Permite modificar la rejilla actual del cursor. El cursor se puede mover con el ratón o con las teclas de flechas del teclado. Si el cursor se está moviendo mediante el ratón, las coordenadas del cursor X 2600 mil Y 2200 mil sólo se actualizarán cuando pasen por un nodo de la rejilla. Si son las teclas las que utilizamos para mover el cursor, este se moverán en saltos, cuyo tamaño corresponde con el dado a la rejilla.

A.3.3.2 Modificar rejilla visible

Permite modificar la rejilla visible de la pantalla. Tanto la rejilla visible (los puntos que se ven en pantalla) como la que limita el movimiento del cursor (apartado A.3.3.1), son individuales.

A.3.3.3 Otra rejilla cursor o visible

Permite modificar la rejilla tanto la del cursor como la visible, pudiendo introducir incluso decimales. Se puede escoger entre milésimas de pulgadas y milímetros.

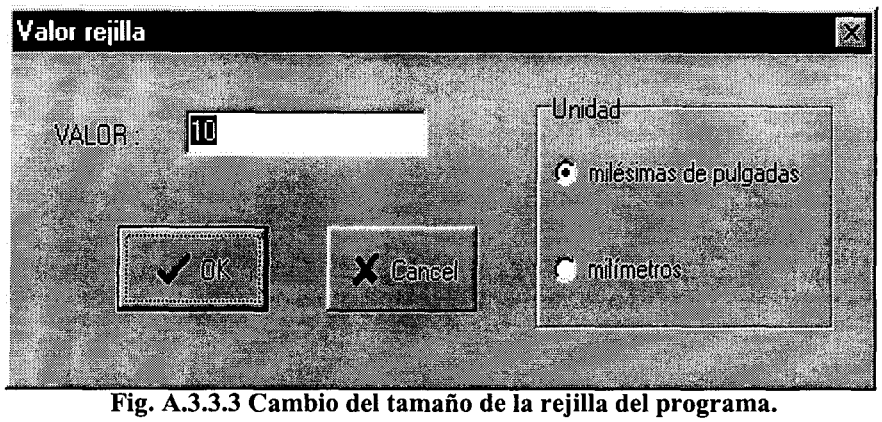


Fig. A.3.3.3 Cambio del tamaño de la rejilla del programa.

A.3.4 MENÚ PALETA

A.3.4.1 Gradiente

Las cuatro gradientes utilizados (cool, hot, gray y estándar) son los mismos que utiliza el programa **Matlab 5.0**.

El programa usa una paleta que puede ir de 9 a 255 colores/tonalidades para cada uno de los gradientes, excepto estándar que va de 25 a 255 colores.

A continuación se dan unas gráficas de cómo evolucionan los cuatro gradientes utilizados, teniendo en cuenta sus tres componentes de colores, la saturación máxima es 1. Este ejemplo está hecho para un total de 70 valores.

HOT

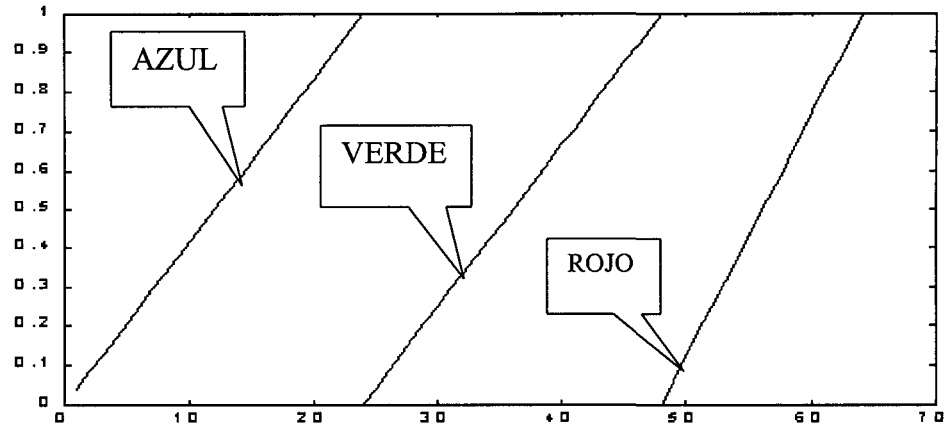


Fig. A.3.4.1 Gradiente de color llamado HOT

COOL

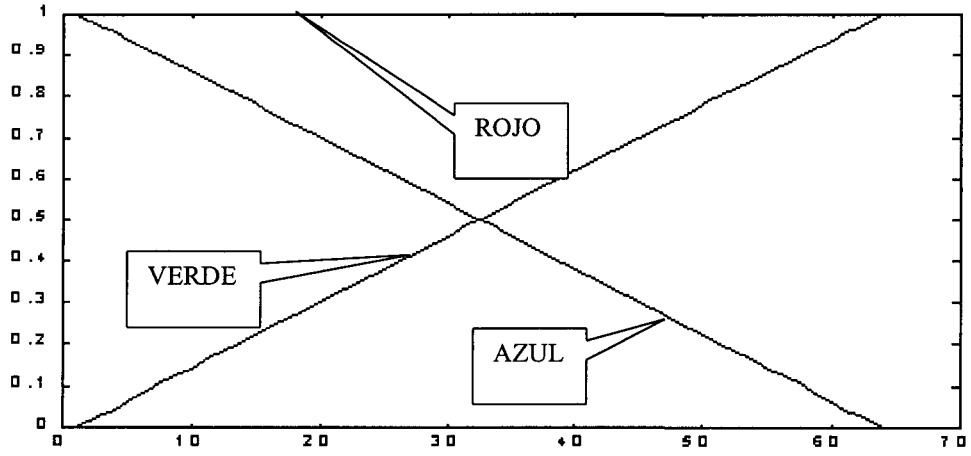


Fig. A.3.4.2 Gradiente de color llamado cool.

GRAY

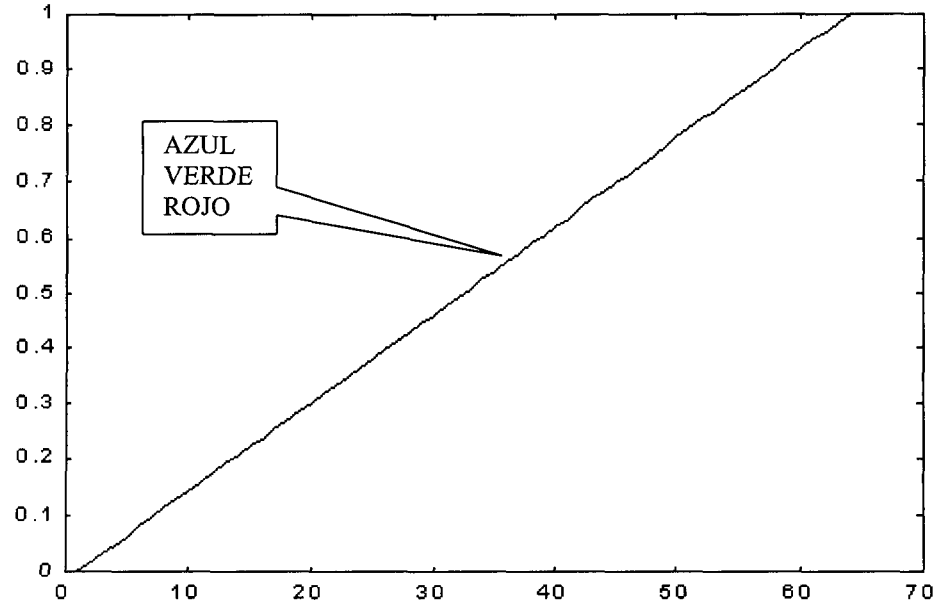


Fig. A.3.4.3 Gradiente de color de escala de grises

ESTANDAR

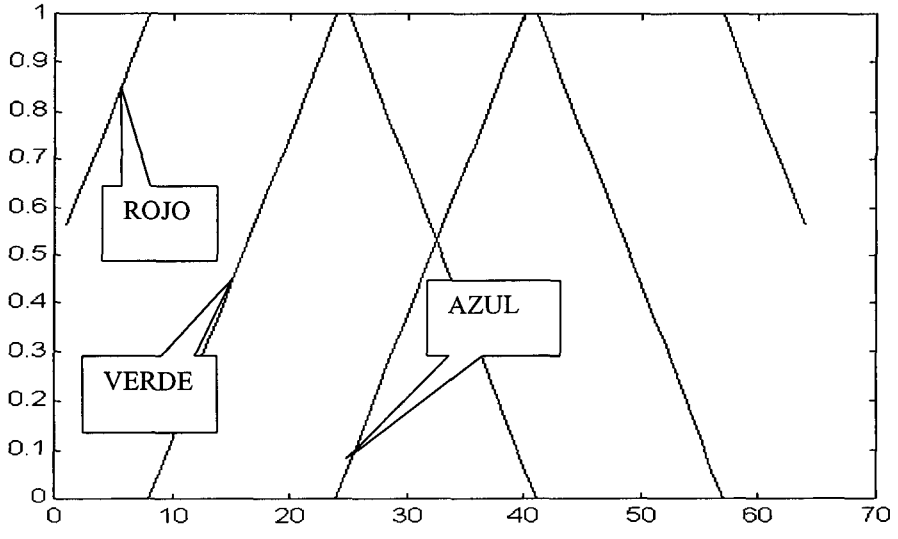


Fig. A.3.4.4 Gradiente de color de escala de estandar.

A.3.4.2 Tonalidades

Especifica el número de tonalidades del gradiente de color, que va de 6 colores o tonalidades a 255.

A.3.5 MENÚ VER

A.3.5.1 Partes del esquema

Permite activar o desactivar la visualización de diferentes partes del esquema de los componentes, rejilla, gradientes de color, cuadrados, etc.

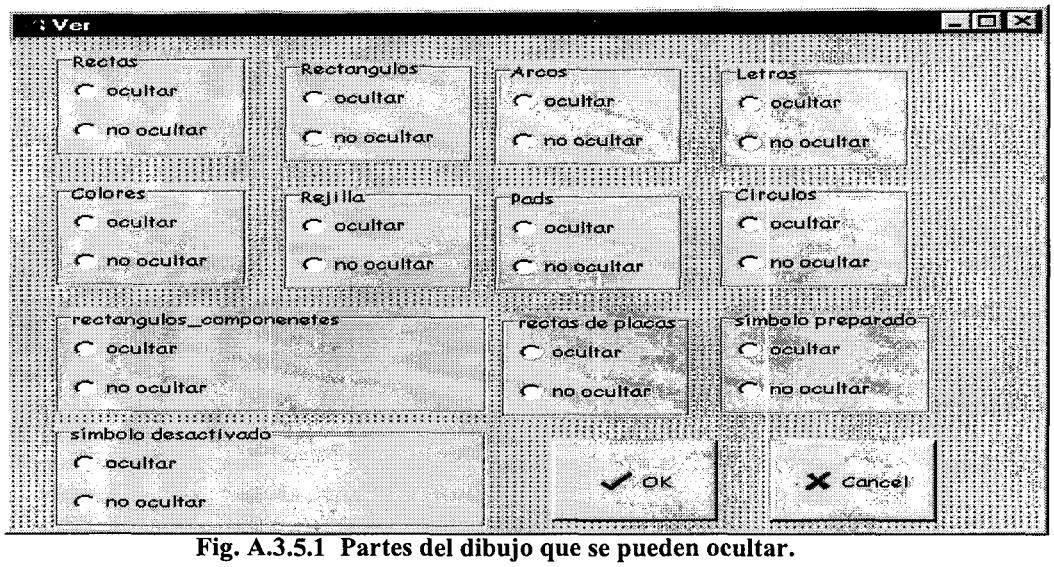


Fig. A.3.5.1 Partes del dibujo que se pueden ocultar.

A.3.5.2 Mallas

Permite poner como activa una de las siguientes mallas:

Tipos de malla.

La malla activa es aquella que esta visualizándose en la pantalla en ese momento. También está indicada en la barra inferior con el nombre de *Malla visible.*

Una malla activa significa además que el gradiente de color es referente a ella, que se puede Modificar el valor de sus celdas, mediante la opción *Poner datos manualmente* del menú *edit*, y que los valores de las barras laterales son valores de la malla activa.

A.3.5.3 Datos físicos 

Abre una ventana donde se puede configurar los datos físicos de la placa y el medio ambiente, ver apartado A.5.

A.3.6 FORMAS

A.3.6.1 Relleno

Activa el gradiente de color, dando a cada valor numérico de la celda, un color.

El color de las celdas dependerá del gradiente de color que se ha seleccionado (cool, hot, gray o estándar).

A.3.7 CONTORNO

A.3.7.1 Crear mallas 

Da la posibilidad de crear las mallas por primera vez (si las mallas no están creadas la mayoría de los menús y botones no funcionan) o rectificar las ya establecidas. Al hacer esto último se pierde toda la información que tenían las antiguas. El botón de la barra de botones, desempeña la misma función.

A.3.7.2 Imprimir 

Da la posibilidad de imprimir el contenido de la pantalla en la impresora, y la escala de colores. No es muy útil si no se dispone de una impresora de colores. El botón de la barra de botones, desempeña la misma función.

A.3.8 MENÚ SIMULACIÓN

A.3.8.1 Empezar simulación 

Inicia la simulación. Hay que tener en cuenta que los datos deben ser coherentes, para los componentes, placa y medio ambiente. Puede darse varias posibilidades en las que la simulación no termine con éxito:

- Que el componente no emita energía (calor) a la placa, sino que reciba, en cuyo caso sale un mensaje indicándolo.
- Que los datos que pide la simulación de las tablas de características (propiedades del aire seco a diferentes temperaturas) no estén, en cuyo caso, sale un mensaje indicándolo.
- Que los datos sean muy altos y se produzca un desbordamiento, en cuyo caso, sale un mensaje indicándolo.

El botón de la barra de botones, desempeña la misma función.

A.3.9 MENÚ LIBRERÍA

A.3.9.1 Componente

Permite modificar las características térmicas de este componente en la librería (archivo llamado *atributos.txt*). Debe tenerse en cuenta que se está modificando el archivo general de la librería y no el específico de la placa que está cargada en ese momento.

A.3.10 MENÚ BARRAS

A.3.10.1 Barras laterales colocación

Permite cambiar las barras laterales de lugar o ocultarlas.

A.4 Menú de datos de los componentes

A.4 ENTRADA AL MENÚ

A este menú se entra picando dos veces sobre el componente. Hay que aclarar que este menú se utiliza tanto para modificar los datos de los componentes del esquema actual, como para modificar los datos de los componentes de la librería. Para modificar los datos del este último, hay que elegir la opción del menú llamada *librería,* y dentro de aquí escoger el submenú *componente.*

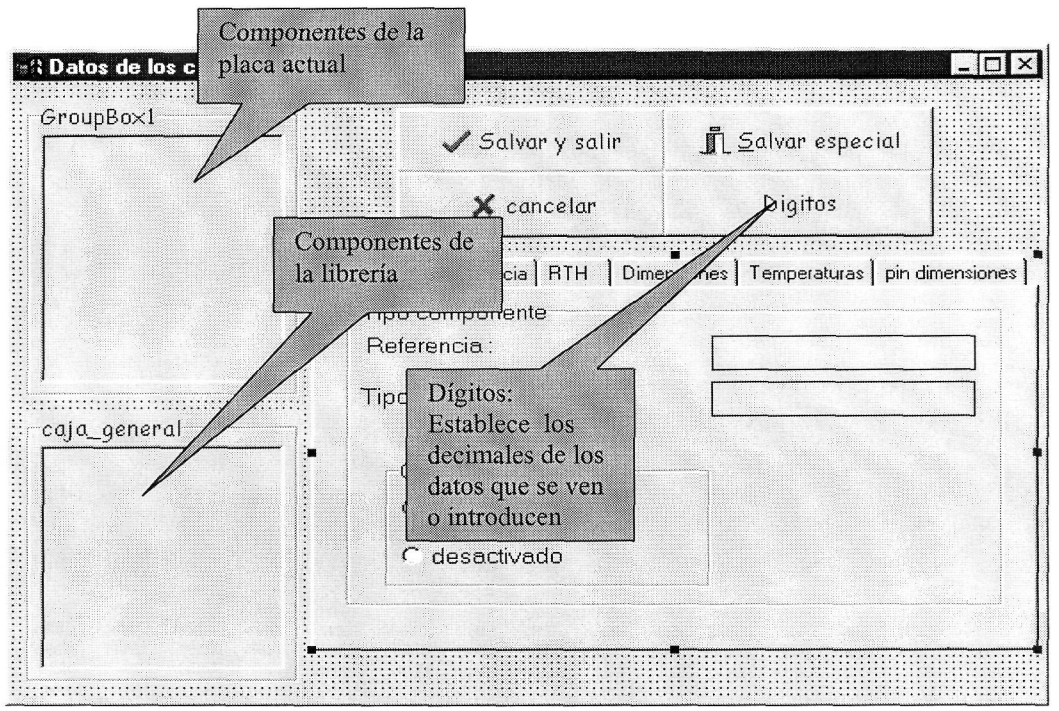


Fig. 4.1.1 Vista principal de la ventana de los componentes.

El botón de *salvar especial* si se pulsa se tendrá la opción de salvar los cambios de la siguiente manera:

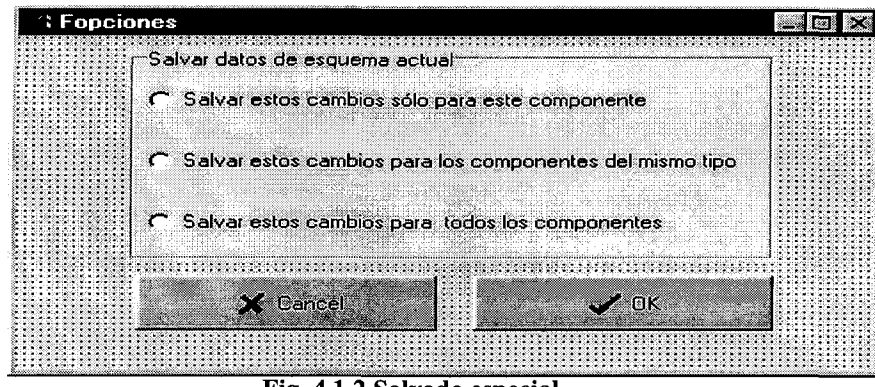


Fig. 4.1.2 Salvado especial

A.4.1 PARÁMETROS DE LOS PINES DE LOS COMPONENTES

La figura A.4.1.1 es el menú de los parámetros de los pines de cada componente. La conductividad del pin, no hay que confundirla con la conductividad de la superficie del componente con la placa, en el caso de que este toque a la placa con su cuerpo.

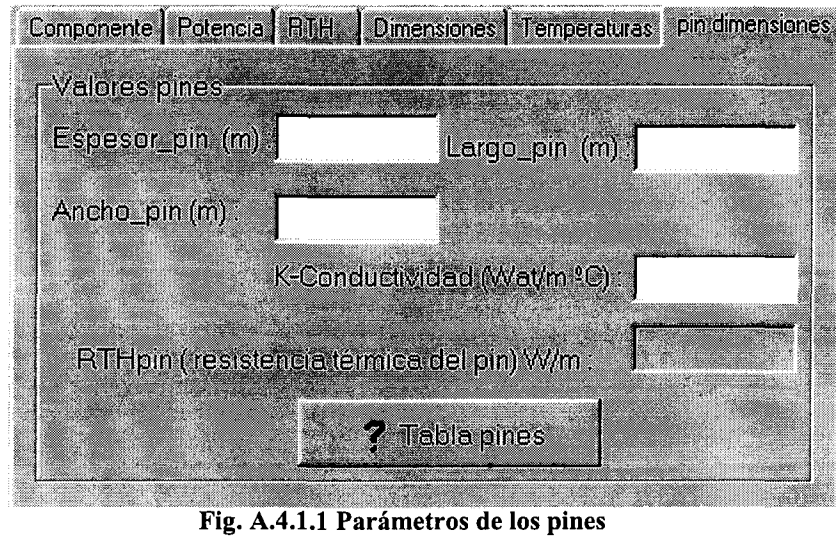


Fig. A.4.1.1 Parámetros de los pines

La tabla A.4.1.2 es una tabla de la conductividad de los pines para diferentes materiales, la cual esta sacada del articulo *Thermal Analysis and Design of Electronic.*

Alloy 42	14,67 W/m C°
Alloy 194	261 W/m C°
Kovar	16,74 W/m C°

Tabla A.4.1.2 Conductividad pines según material.

Para mantener una relación de la potencia calorífica suministrada por cada pin, se establece una matriz que será la que se utilice para obtener la

potencia emitida a la placa por cada pin. A la hora de introducir los datos de los componentes, se debe consultar los libros de características de los componentes, en los apartados referentes a las resistencias térmicas. Algunos de los datos serán muy difíciles de conseguir, pero se podrán sacar por estimación de los otros componentes que estén en librería

A.4.2 ZONA DE EMISIÓN DE CALOR SUPERFICIAL

La figura A.4.2.1 es el menú de los datos de la zona de emisión de calor a superficie. Por defecto las dimensiones de largo y ancho son las reales del componente, estas dimensiones forma un rectángulo el cual es el de emisión de calor a la placa, en modo superficie. Este rectángulo se puede modificar mediante el menú *edit* y dentro de aquí escoger la opción *rectifica_rectangulo_componente*. El espesor es el espesor del material entre el componente y la placa, este puede ser cualquier material que se conozca, la resistencia térmica de superficie igual, a la zona de emisión de calor a la superficie. El dibujo A.4.2.2 representa la dimensiones por defecto que tiene el rectángulo de emisión de calor a la placa que coincidirá al principio con las dimensiones del componente.

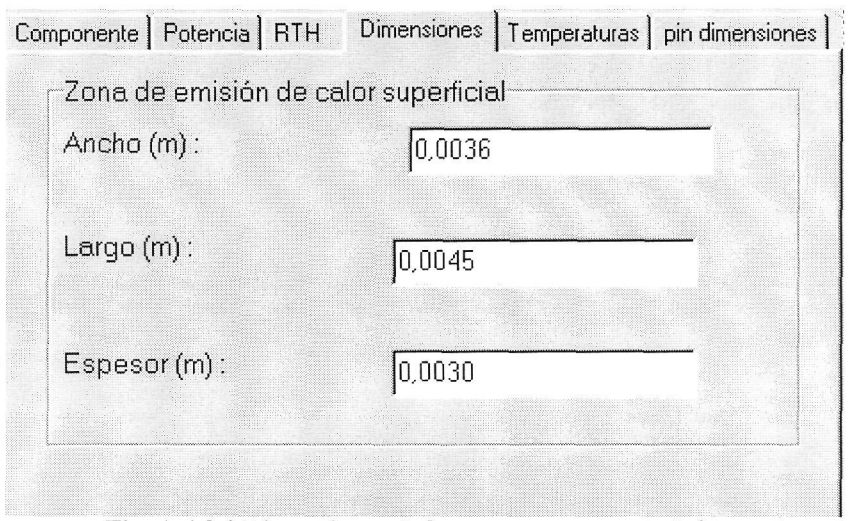


Fig. A.4.2.1 Dimensiones de los componentes por defecto

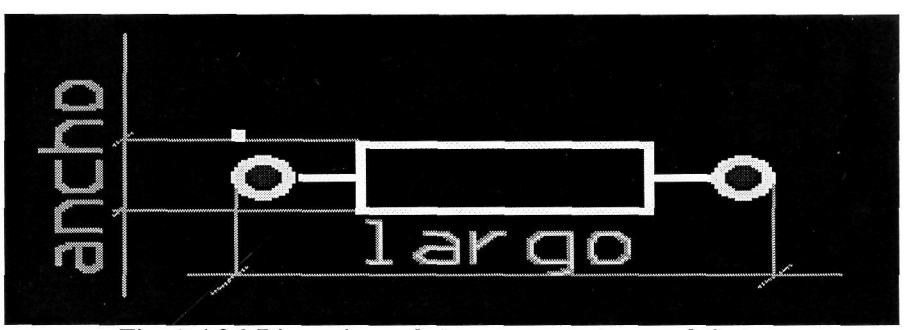


Fig. A.4.2.2 Dimensiones de los componentes por defecto.

A.4.3 TEMPERATURA EN LAS DIFERENTES PARTES DEL COMPONENTE Y PLACA

Se trata de una pantalla de datos donde se refleja e resultado de la simulación.

Las temperaturas que representa la figura A.4.3.1 son:

- Tj: Temperatura de la unión.
- Tc: Temperatura de cápsula.
- Tp: Temperatura de la placa (material conductor de la placa).

Estas temperaturas se obtienen después de la simulación.

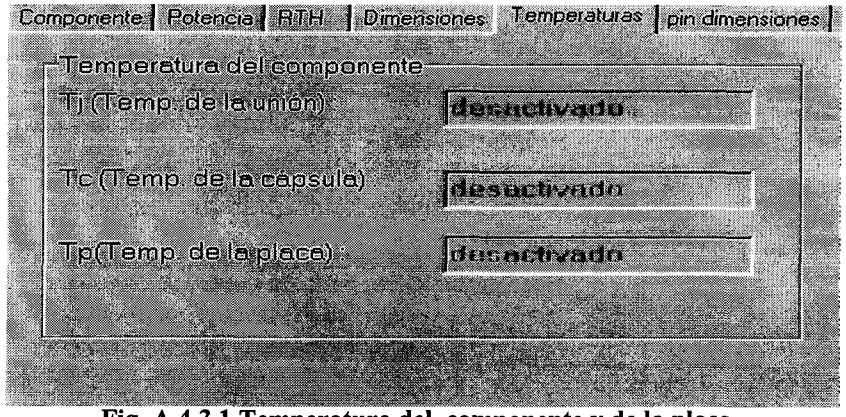


Fig. A.4.3.1 Temperatura del componente y de la placa.

A.4.4 POTENCIA DEL COMPONENTE Y TIPO DE POTENCIA

En la figura A.4.4.1 es el menú cuyos datos representa la potencia que desarrolla el componente, así como la potencia superficial que este suministrara a la placa, esta última es un dato obtenido al final de la simulación. Por otro lado se tiene la forma de emisión de potencia a la placa: si esta es a través de la superficie o por medio de los pines.

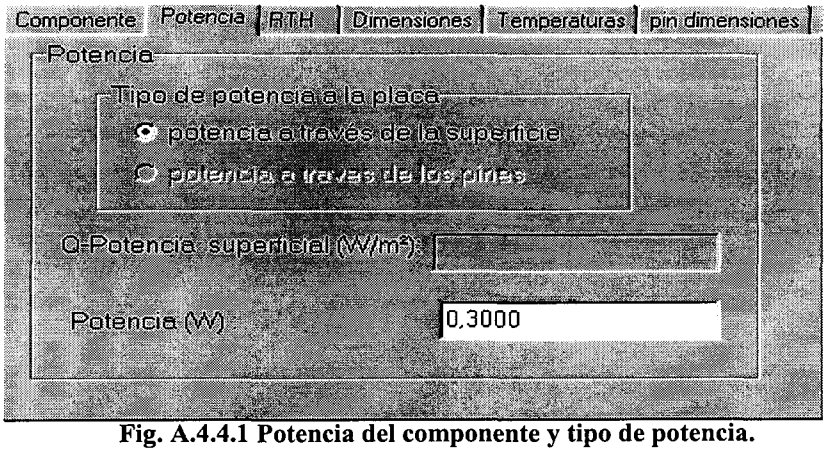


Fig. A.4.4.1 Potencia del componente y tipo de potencia.

A.4.5 DEFINICIÓN DEL COMPONENTE

La figura A.4.5.1 es un menú cuyos datos representa el tipo de componente y su referencia. El recuadro indicado como componente nos da la posibilidad de activar o desactivar el componente para la simulación.

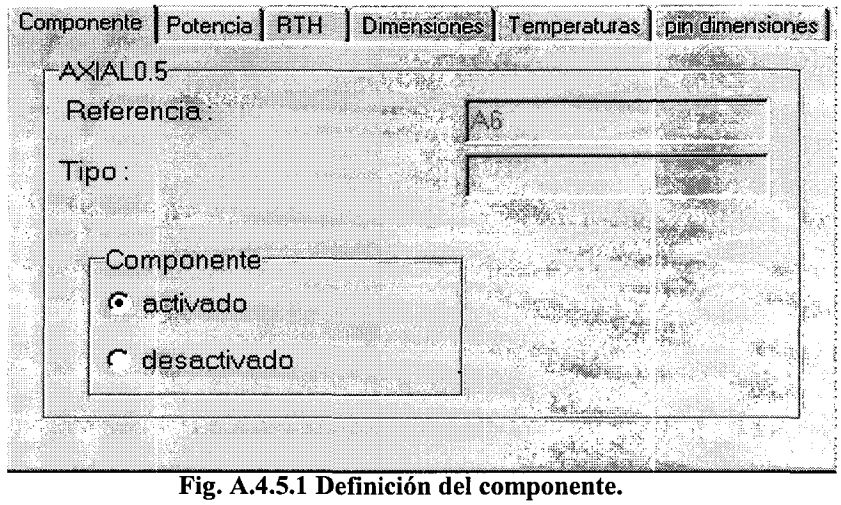


Fig. A.4.5.1 Definición del componente.

A.4.6 RESISTENCIAS TÉRMICAS DEL COMPONENTE

La figura A.4.6.1 es menú cuyos datos representa las diferentes resistencias térmicas del componente.

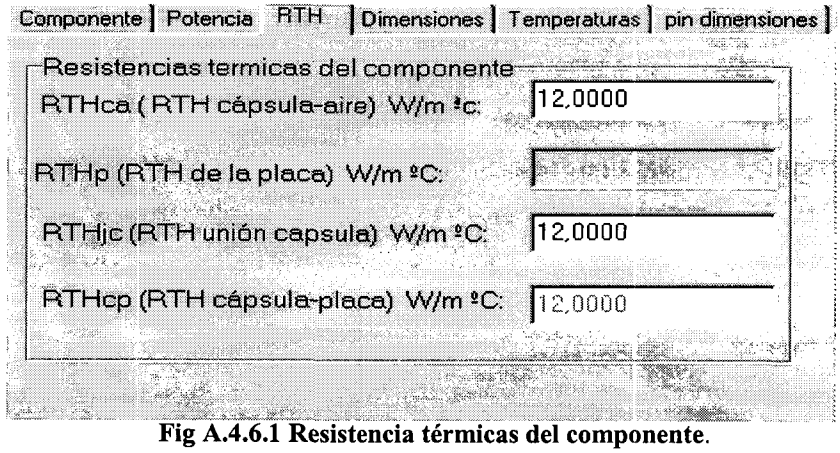
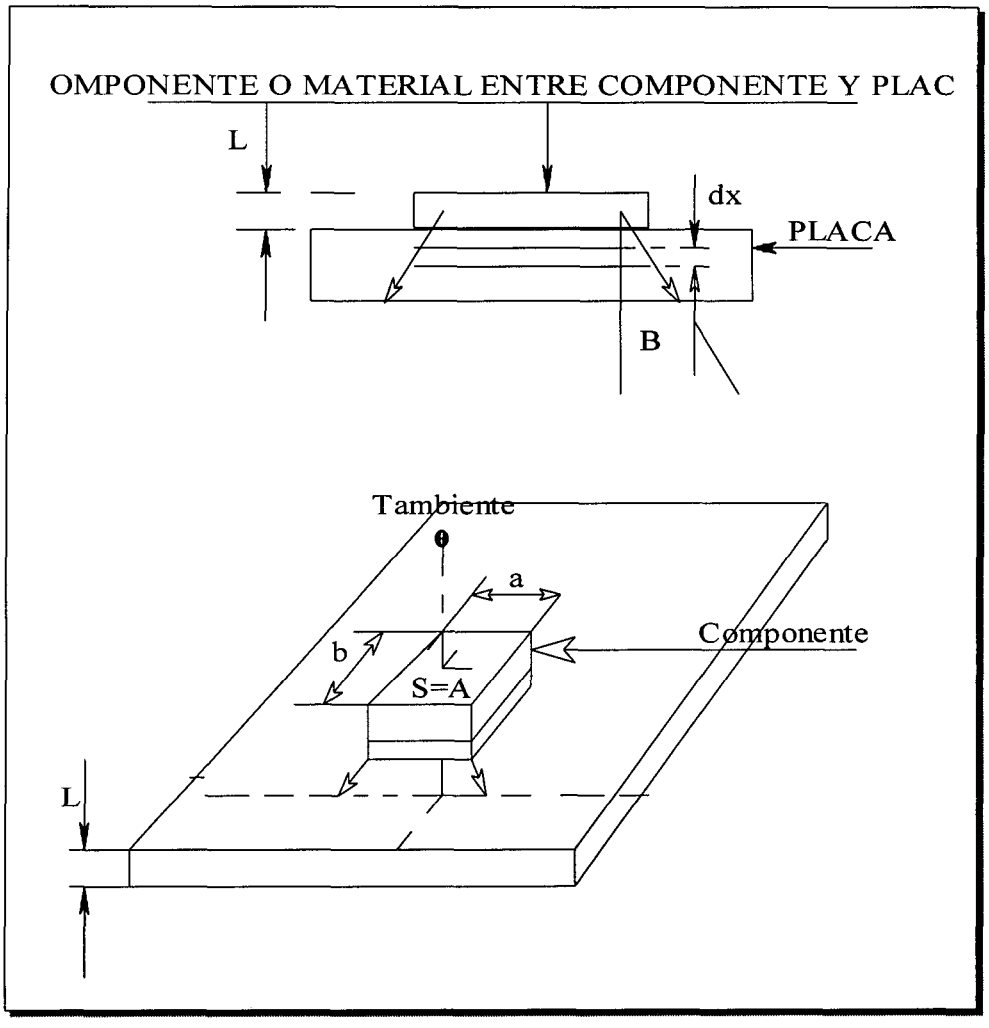


Fig A.4.6.1 Resistencia térmicas del componente.

La resistencia térmica de la placa RTHp (en este caso material aislante) debajo del componente se calcula de forma automática mediante el siguiente criterio:

El flujo de calor en las tres direcciones se divide en un flujo en la dirección del gradiente (x e y) de temperatura, y en otro lateral. Es debido a este último que la resistencia térmica disminuye, puesto que la superficie "encerrada" por el flujo de calor es mayor.



- B: ángulo de dispersión del flujo de calor.
- Dx: diferencial de longitud del camino seguido por e flujo de calor.
- S=a*b sección (superficial) del flujo de calor.

El ángulo de dispersión de B es el que forma el gradiente de temperatura en la dirección del flujo principal (que es el que une la parte más caliente, es decir el componente, con la más fría, es decir parte inferior del substrato que estará en contacto con el medio ambiente o en una "caja" de refrigeración) con el flujo lateral.

En la mayoría de los casos se toma este ángulo del valor de 45°.
Por tanto para calcular la resistencia térmica del substrato se considera que el flujo de calor está contenido en un cono de 45° de ángulo. El valor de esta resistencia se puede hallar por la integral:

$$Rth = \frac{1}{\kappa} \int_0^L \frac{dx}{S}$$

A continuación calculamos la resistencia térmica que presenta el substrato (placa) para distintas formas de los componentes.

➢ Componente (fuente calor) cuadrado:

En este casa a=b. Suponiendo que el material entre componente y placa es isotérmico (su conductividad y demás aspectos físicos se mantienen constante para todo la superficie), la resistencia térmica es:

$$RTh = \frac{1}{\kappa}\int_{o}^{l}\frac{dx}{S} = \frac{1}{\kappa}\int_{0}^{l}\frac{dx}{(a+2x)^2}$$

Realizando la integral:

$$Rth = \frac{1}{\kappa}\left|\frac{1}{2(a+2L)}\right|_{0}^{l}$$

$$Rth = \frac{L}{\kappa a(a+2L)}$$

➢ Componente (fuente de calor) rectangular:

$$Rth = \frac{1}{\kappa}\int_{0}^{L}\frac{dx}{S} = \frac{1}{\kappa}\int_{0}^{L}\frac{dX}{(b+2x)(a+2x)}$$

Integrado se tiene:

$$Rth = \frac{1}{2\kappa(b-a)}\ln\left(\frac{b}{a}\left(\frac{a+2L}{b+2L}\right)\right)$$

A.5 Ventana de parámetros físicos

A.5 VENTANA DE PARÁMETROS FÍSICOS

A esta tabla se entre una vez se halla creado la malla del esquema o mediante el botón:

Esta ventana es donde el usuario tiene que introducir los datos de la placa y del medio ambiente.

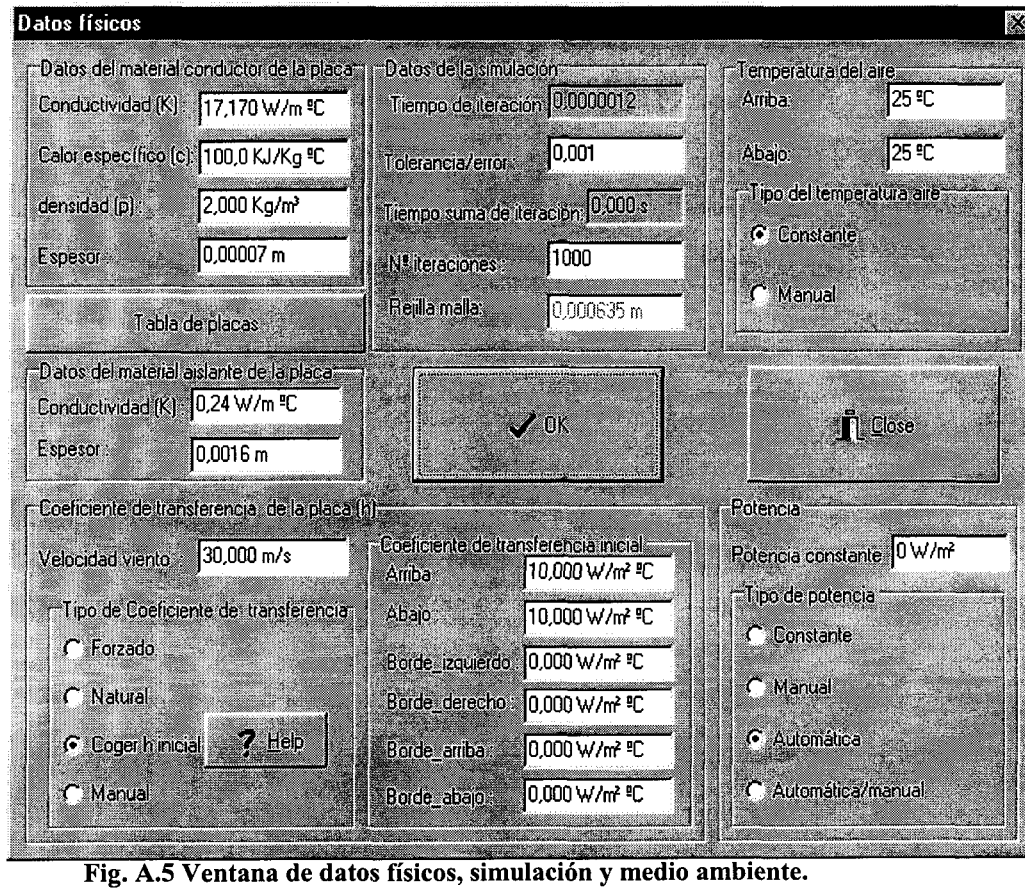


Fig. A.5 Ventana de datos físicos, simulación y medio ambiente.

A.5.1 DATOS DE LA PLACA

A.5.1.1 Datos del material conductor de la placa

A.5.1.1.1 Conductividad (k)

Es la conductividad del material conductor de la placa en W/m ºC, es un dato a suministra por el usuario, depende del grosor y del tipo de placa. El botón *Tabla de placas* abre una lista de algunas placas existentes en el mercado, con los diferentes datos físicos que se van necesitar.

A.5.1.1.2 Calor específico (c)

Es el calor específico del material conductor de la placa en KJ/Kg ºC, es un dato a suministra por el usuario, recordar que es el calor específico que va en función del grosor de las pistas y material conductor de las pistas. El botón *Tabla de placas* abre una lista de algunas placas existentes en el mercado, con los diferentes datos físicos que se van necesitar.

A.5.1.1.3 Densidad de placa (p)

Es la densidad del material conductor de la placa en Kg/m^3, es un dato a suministra por el usuario, dependerá del grosor de las pistas y material conductor de la placa. El botón *Tabla de placas* abre una lista de algunas placas existentes en el mercado, con los diferentes datos físicos que se van necesitar.

A.5.1.1.4 Espesor de la base conductora (delta)

Es el grosor del material conductor de la placa en metros. El botón *Tabla de placas* hay una lista con algunas placas del mercado con el correspondiente espesor del material conductor.

A.5.1.2 Datos de material aislante de la placa

A.5.1.2.1 Conductividad (K) de la base aislante

Es la conductividad del material aislante de la placa en W/m ºC, es un dato a suministra por el usuario, depende del tipo de placa. El botón *Tabla de placas* abre una lista de algunas placas existentes en el mercado, con los diferentes datos físicos que se van necesitar. Entre ellos la conductividad del material aislante.

A.5.1.2.2 Espesor de la base aislante

Es el espesor del material aislante de la placa en metros. Normalmente este espesor se cogerá 0,0016 m que es el estándar para las placas que tenemos en la *tabla de placas.*

A.5.2 DATOS DE LA SIMULACIÓN

A.5.2.1 Tiempo de iteración

El tiempo de iteración es el máximo tiempo permitido entre en paso *n* y el *n+1* del método explícito, que es el que se utiliza para el cálculo de los valores de temperatura de la placa. El programa lo da de forma automática mediante una ecuación, ya que es una restricción para el método explícito.

A.5.2.2 Error

Es el valor que el usuario debe establecer para determinar el error que se comete en los valores de la simulación. También determinara la velocidad de la simulación. Para valores muy chicos los tiempos de la simulación se pueden hacer muy largos. Todo lo contrario para valores grandes de error, pero en este último caso se conseguirán valores menos fiables. Se aconseja tomar como referencia para este campo "0,001".

A.5.2.3 Tiempo suma de iteraciones

Es simplemente la suma de los tiempos de iteraciones después de alcanzar la tolerancia o error.

NOTA: No corresponde con el tiempo total alcanzando, ya que el algoritmo posee un acelerador llamado SOR, que modifica este tiempo.

A.5.2.4 Número de iteraciones

Es valor máxima de iteraciones que el algoritmo puede llevar a cabo para obtener las temperaturas de la placa. Establece el número de iteraciones a partir del cual, sino se ha llegado a una solución final, emitirá un mensaje de aviso. En este caso podemos optar por incrementar el número de iteraciones o modificar los parámetros de la simulación.

A.5.2.5 Rejilla malla

Es el valor que el usuario introdujo a la hora de crear las mallas del sistema, lo que aquí el valor se expresa en metros, en lugar de milésimas de pulgadas. No es modificable por el usuario y la única forma de cambiar este valor es creando una nueva malla.

A.5.3 TEMPERATURA DEL AIRE

A.5.3.1 Tipo de temperatura del aire

La temperatura puede ser *manual* a *constante.*

Manual significa que el usuario puede establecer la temperatura del aire de la placa tanto arriba como abajo, en las zonas de la placa que quiera y el valor que quiera mediante la opción *Poner datos manualmente.*

Constante significa que la temperatura es establecida como una constante en toda la placa.

A.5.4 Coeficiente de transferencia

A.5.4.1 Velocidad viento

Es la velocidad del viento cuando se utiliza un ventilador que siempre estará situado en la parte izquierda de la placa. Este valor sólo hay que introducirlo si se elige en el cuadro de coeficiente de transferencia, el modo forzado.

A.5.4.2 Tipo de coeficiente de transferencia

El tipo de *coeficiente de transferencia* establece la perdida de calor de la superficie en este caso de la placa.

La relación de transferencia de calor por convección es generalmente una función compleja de la geometría de la superficie y temperatura, la temperatura y velocidad del fluido, y propiedades termofísicas del fluido.

El coeficiente de transferencia depende en primer lugar si el viento circundante que rodea el material es forzado o no, es decir, si es una convección forzada o natural.

Este flujo puede ser:

- Natural

El movimiento entre el fluido exterior y la placa tiene lugar debido a la gravedad. No hay ninguna otra fuerza exterior que provoque el movimiento térmico, entre la placa y el aire a una cierta distancia que se considera a temperatura constante.

- Forzado

El movimiento térmico entre el fluido exterior y la placa es debido a una fuerza exterior (ventilador, que en este proyecto estará situado siempre en la parte izquierda de la placa). Lo que el aire irá de izquierda aderecha.

Tanto el flujo natural como forzado pueden llegar a ser:

- laminar

Predomina en las bajas velocidades, para pequeños tamaños y para los fluidos más viscosos.

- Turbulento

La Transferencia de calor tiende a ser mucho más grande en flujos turbulentos que en flujos laminares, debido a la mezcla vigorosa del fluido.

A.5.4.3 Coger h inicial

Si se coge esta opción el valor predeterminado del coeficiente de transferencia es el que esta indicado en el recuadro llamado *coeficiente de transferencia inicial*, para diferentes partes de la placa.

A.5.4.4 Manual

Manual significa que el usuario puede establecer el coeficiente de transferencia de la placa tanto arriba como abajo mediante la opción *Poner datos manualmente*. Del menú del programa, ver apartado A.3.2.2.

A.5.5 COEFICIENTE DE TRANSFERENCIA INICIAL

Si en el menú *tipo de coeficiente de transferencia* se escogió *coger h inicial,* los valores indicados con: arriba, abajo, borde izquierdo, etc. Son los que representa el coeficiente de transferencia en las distintas partes de la placa.

A.5.6 POTENCIA

A.5.6.1 Tipo de potencia

La potencia puede ser:

- *Manua*l significa que el usuario puede establecer la potencia (es potencia superficial W /m^2) en las zonas de la placa que quiera y el valor que quiera mediante la opción *Poner datos manualmente*, del menú del programa.

- *Constante* significa que la Potencia es establecida como una constante en toda la placa.

- Automática significa que la potencia que el componente suministra a la placa es por medio de: las resistencias térmicas de los pines, la superficie del componente en contacto con la placa o por medio de un material entre el componente y la placa, explicado en el apartado A.4.4.

- Automática/manual que la potencia suministrada por el componente a la placa además de la automática simple, si puede añadir valores manualmente con la opción del menú *Poner datos manualmente.*

A.6 Funciones del teclado

A.6.1 TECLAS DE ZOOM DIRECCIONAL

La tecla para aumentar el tamaño del dibujo de la pantalla, es la tecla: "Re Pág", el aumento puede llegar hasta 64 veces del original y con saltos de potencia de dos (2,4,8 hasta 64).

La tecla para disminuir el tamaño de la pantalla es la tecla: "Av Pág", la disminución puede llegar hasta 1/64 veces del original y con saltos de potencia de dos (1/2,1/4 hasta 1/64).

A.6.2 TECLA DE VISUALIZACIÓN DE TEMPERATURA

Al pulsar la tecla F11 bajo un componente nos darán en la tabla de datos lateral el valor de las temperaturas de placa, componente y unión, de última simulación hecha.

Termín 1.0
Simulador Térmico

Apéndice B

En el se puede ver todo el código fuente de ***Termín 1.0***.

Función de módulos o fichas	Nombre del módulo
Calcula el coeficiente de transferencia forzado.	Calcula_h_forzado
Calcula el coeficiente de transferencia natural.	Calcula_h_natural
Obtener datos de la placa y medio ambiente.	Datos_físcos
Dibuja todos los componente y controla los menús de la ficha hija activa. Es la ficha más importante del programa.	Fesquema
Calcula la temperatura de los nodos laterales y esquina de la malla.	H2
Calcula la descomposición LU para calcular Sistema de ecuaciones de la potencia de los pines.	LU
Inserta datos en la malla de forma manual.	Manual
Oculta individualmente las partes del dibujo (rectas,arcos,colores...).	Ocultar
Opciones para guarda los datos de los componentes para el mismo tipo, individual o para todos los tipos.	Opciones
Es la ficha padre de todas las fichas hijas (Fesquema) controla los menús generales.	Padre
Calcula el valor de la temperatura de los nodos Centrales.	SOR3
Calcula y dibuja todo lo relacionado con los mapas de colores de la temperatura.	Surface
Activa los eventos de salir y entrar en los componentes Edit para mantener las unidades.	Tipo_eventos
Referencia de todos los tipos usados en el programa y gran variedades de procedimientos y funciones comunes a todos los módulos y fichas.	Tipo1
Cambia el valor de la unidad y el tipo.	Unidad
Saca o introduce los datos térmicos de los componentes del archivo asociado a la placa.	Usacar_datos_artributos
Saca los datos de los componentes del archivo de pcb procedente del programa Protel 1.5 que debe estar en forma de texto.	Usacar_datos_Protel
Saca los datos de los componentes del archivo de pcb procedente del programa Roën 1.0.	Usacar_datos_ULPGC
Con esta ficha se puede introducir los datos térmicos de los componentes y la librería asi como poder modificarlos.	Verdatos

unit calcula_h_forzado;

```
interface
uses
 dialogs,tipo1,windows,math;
const
zona_turbenta=5e5;
procedure calculo_h_forzado(var matrizt,matriztaire,matrizh,matriz_esp: tmatriz; var fisicos:
tparametros_fisicos;
var rectangulo_cuadricula: trect;  funcion_turbulento,
funcion_laminar: tfuncion; var lista_aire: tipoaire;var med_temp,cambio: double);
function nussent_forzado_lami1(rex,pr: real): real; far;
function nussent_forzado_turb1(rex,pr: real): real; far;

implementation

//Cálcula el coeficiente de transferecia en convección forzada//

procedure calculo_h_forzado(var matrizt,matriztaire,matrizh,matriz_esp: tmatriz; var fisicos:
tparametros_fisicos;
var rectangulo_cuadricula: trect;  funcion_turbulento,
funcion_laminar: tfuncion; var lista_aire: tipoaire;var med_temp,cambio: double);
var
c,f: integer;
puntero: tipoaire;
valor,alfa,aux_valor,tm,max_nul,rex,nux,posicionx,posiciony,h: real; {es la temperatura en zona pelicular
de la placa}
begin
aux_valor:=0;
for c:=1 to matrizt.max_columna do
   for f:=1 to matrizt.max_fila do begin
     {se calcula la media de temperatuea entre el aire y la placa}
     tm:=(matriztaire.matriz[f,c]+matrizt.matriz[f,c])/2;
     valor:=round(tm/10)*10;
     if valor<>aux_valor then begin
     aux_valor:=valor;
     puntero:=lista_aire;
     while (puntero<> nil) and not( puntero.aire.t=valor) do
     puntero:=puntero.prox;
     if puntero=nil then  begin
       showmessage('valor de temperatura fuera del margen de la tabla aire.txt');
       cambio:=0;
       exit;
     end;
     end;
     {calcula la posicion de la fila y columna y de milesima de pulgadas se pasa a metros}
     posicionx:=inmil_milime(rectangulo_cuadricula.left + (c-1) * fisicos.delta_in)/1000;
     posiciony:=inmil_milime(rectangulo_cuadricula.bottom + (f-1) * fisicos.delta_in)/1000;
     {se calcula el numero de reino local}
     rex:=fisicos.u * posicionx / puntero.aire.v;
     if rex>zona_turbenta then
     {si entre en este bucle es que es un efecto turbulento}
     {nux= es el valor de nussent}
     nux:=funcion_turbulento(rex,puntero.aire.pr)
     else
     {si entra en este bucle es que el efecto es laminar}
     nux:=funcion_laminar(rex,puntero.aire.pr);
     matrizh.matriz[f,c]:=nux * puntero.aire.k / posicionx;
```

```
if fisicos.h_minimo_arriba<matrizh.matriz[f,c] then fisicos.h_minimo_arriba:=matrizh.matriz[f,c];
with fisicos do begin
   alfa:=(p*c);
   dt:= alfa/(4*k/sqr(delta) +2* h_minimo_borde/delta+ h_minimo_arriba/espesor+
h_minimo_abajo/espesor);
   radio1:=k * dt /( p * c * sqr(delta));
   radio2:=   dt /( p * c * espesor);
   radio3:=   dt /( p * c * delta);
end;
end;
end;
```

//cálcula el coeficiente de transferencia de un flujo laminar//

```
function nussent_forzado_lami1(Rex,Pr: real): real; far;
begin
{VER LAS TABLAS PARA DETALLE DE ESTE MODELO}
result:=0.332 * SQRT(REx) * exp(1/3*ln(Pr));
end;
```

//Cálcula el coeficiente de transferencia de un flujo turbulento//

```
function Nussent_forzado_turb1(Rex,Pr: real): real; far;
begin
{VER LAS TABLAS PARA DETALLE DE ESTE MODELO}
if (REx>ZONA_TURBENTA) and  (REx<1e7) then
result:=0.0296 * exp(0.8*ln(Rex)) * exp(1/3*ln(Pr))
else
result:=0.185 * Rex * exp(-2.584 * ln(Log10(Rex))) * exp(1/3*ln(Pr));
end;
end.
```

```
unit calcula_h_natural;
interface
uses
 Dialogs,tipo1,Windows;
CONST
ZONA_TURBENTA=5E5;
procedure calculo_h_natural(var matrizT,matrizTAIRE,MATRIZH,matriz_esp: Tmatriz; var fisicos:
Tparametros_fisicos;
var rectangulo_cuadricula: Trect;  funcion_turbulento,
funcion_laminar: Tfuncion; var lista_aire: tipoaire;var MED_TEMP,cambio: DOUBLE);
function Nussent_natural_lami1(RaL,Pr: real): real; far;
function Nussent_natural_turb1(RaL,Pr: real): real; far;
implementation
//Cálcula el coeficiente de transferencia para convección natural//

procedure calculo_h_natural(var matrizT,matrizTAIRE,MATRIZH,matriz_esp: Tmatriz; var fisicos:
Tparametros_fisicos;
var rectangulo_cuadricula: Trect;  funcion_turbulento,
funcion_laminar: Tfuncion; var lista_aire: tipoaire;var MED_TEMP,cambio: DOUBLE);
var
C,F: INTEGER;
puntero: tipoaire;
valor,temperatura_media,aux_valor,alfa,TM,max_nul,RaL,Lc,Gr,NUL,H: REAL; {ES LA TEMPERATURA
EN ZONA PELICULAR DE LA PLACA}
begin
aux_valor:=0;
temperatura_media:=(med_temp+fisicos.temp_media_aire_abajo)/2;
valor:=round(temperatura_media/10)*10;
      if valor<>aux_valor then begin
         puntero:=lista_aire;
      while (puntero<> nil) and not( puntero.aire.t=valor) do
      puntero:=puntero.prox;
      if puntero=nil then  begin
        showmessage('valor de temperatura fuera del margen de la tabla');
            exit;
      end;
      end;
with fisicos do
    Lc:=largo_malla * ancho_malla/(2*largo_malla+2*ancho_malla);
Gr:=exp(3*ln(Lc)) * 9.8 * (1/293.15) * temperatura_media/sqr(puntero.aire.V);
RaL:=100*Gr*puntero.aire.Pr;
IF (RaL<3E10) and (RaL>3E5) THEN begin
  NuL:=0.27 *  exp(1/4*ln(RaL))
  h:=NuL * fisicos.K/fisicos.largo_malla;}
  end
else
h:=exp(1/4*ln(temperatura_media/fisicos.largo_malla));
poner_matriz_valor(matrizh,h,1,matrizh.max_fila,1,matrizh.max_columna);
if FISICOS.h_MINIMO_ABAJO<h then
  FISICOS.h_MINIMO_ABAJO:=h;
WITH FISICOS DO BEGIN
ALFA:=(P*C);
DT:= ALFA/(4*K/SQR(delta) +2* h_MINIMO_BORDE/delta+ H_MINIMO_ARRIBA/ESPESOR+
h_MINIMO_ABAJO/ESPESOR);
RADIO1:=K * DT /( P * C * SQR(DELTA));
RADIO2:=   DT /( P * C * ESPESOR);
RADIO3:=   DT /( P * C * DELTA);
end;
end;
end.
```

unit DATOS_FISICOS;

```
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls,tipo1,tipo_eventos, Buttons, ExtCtrls,h_documento;

type
  TFfisicos = class(TForm)
    GroupBox3: TGroupBox;
    eDT: TEdit;
    Label10: TLabel;
    GroupBox1: TGroupBox;
    Label5: TLabel;
    Etol: TEdit;
    Label6: TLabel;
    ETMAX: TEdit;
    RGh: TRadioGroup;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    GroupBox2: TGroupBox;
    rgt: TRadioGroup;
    Etemp_ambiente_arriba: TEdit;
    Label7: TLabel;
    Label8: TLabel;
    Etemp_ambiente_abajo: TEdit;
    GroupBox4: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Ees: TLabel;
    Ek: TEdit;
    Ec: TEdit;
    Ep: TEdit;
    Eespesor: TEdit;
    GroupBox5: TGroupBox;
    EHabajo: TEdit;
    EHarriba: TEdit;
    EHborde_izquierdo: TEdit;
    Label11: TLabel;
    Label17: TLabel;
    EHborde_derecho: TEdit;
    EHborde_arriba: TEdit;
    Label16: TLabel;
    Label18: TLabel;
    EHborde_abajo: TEdit;
    GroupBox6: TGroupBox;
    Label9: TLabel;
    Eiteraciones: TEdit;
    Epotencia_dentro: TEdit;
    Label20: TLabel;
    RGTP: TRadioGroup;
    Label14: TLabel;
    Edelta: TEdit;
    Label12: TLabel;
    EUhorizontal: TEdit;
    GroupBox7: TGroupBox;
    Label4: TLabel;
```

```
    Ek_aislante: TEdit;
    Label13: TLabel;
    Eespesor_aislante: TEdit;
    Button1: TButton;
    procedure EVverticalKeyPress(Sender: TObject; var Key: Char);
    procedure EVverticalExit(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);

  private
  parametros_fisicos: Tpuntero_fisicos;
  digitos: integer;
  procedure sacar_datos;
  procedure poner_datos(parametros_fisicos: Tpuntero_fisicos);

    { Private declarations }
  public
  procedure entrada_datos(fisicos: Tpuntero_fisicos; digitos: integer);

    { Public declarations }
  end;
var
  Ffisicos: TFfisicos;

implementation

uses tabla1;

{$R *.DFM}

//Poner unidades en todos los datos//

procedure TFfisicos.entrada_datos(fisicos: Tpuntero_fisicos; digitos: integer);
  var
  i: integer;
  begin
    parametros_fisicos:=fisicos;
    poner_datos(parametros_fisicos);
    for i:=0 to Componentcount-1 do
    if components[i] is Tedit then
      (components[i] as Tedit).text:= poner_unidad(array_unidades[(components[i] as Tedit).tag],
      (components[i] as Tedit).text);
  end;

//extraer los datos físicos de los registro para que se visualicen//

procedure TFfisicos.poner_datos(parametros_fisicos: Tpuntero_fisicos);
begin
    with parametros_fisicos^ do begin
    Ek.TEXT:=floattostrF(k,ffFixed,8,digitos);
    Ec.TEXT:=floattostrF(c,ffFixed,8,digitos-2);
    Ep.TEXT:=floattostrF(p,ffFixed,8,digitos);
    EUhorizontal.TEXT:=floattostrF(U,ffFixed,8,digitos);
    EHarriba.TEXT:=floattostrF(H_arriba,ffFixed,8,digitos);
    EHabajo.TEXT:=floattostrF(H_abajo,ffFixed,8,digitos);
    EHborde_izquierdo.TEXT:=floattostrF(H_borde_izquierdo,ffFixed,8,digitos);
    EHborde_derecho.TEXT:=floattostrF(H_borde_derecho,ffFixed,8,digitos);
    EHborde_arriba.TEXT:=floattostrF(H_borde_arriba,ffFixed,8,digitos);
```

```
    EHborde_abajo.TEXT:=floattostrF(H_borde_abajo,ffFixed,8,digitos);
    Edelta.TEXT:=floattostrF(delta,ffFixed,8,digitos+3);
    Eespesor.text:=floattostrF(espesor,ffFixed,8,digitos+2);
    ETmax.text:=floattostrF(tmax,ffFixed,8,digitos);
    Eespesor_aislante.text:=floattostrF(espesor_aislante,ffFixed,8,digitos);
    EK_aislante.text:=floattostrF(K_aislante,ffFixed,8,digitos);
    end;
    Edelta.text:=poner_unidad(array_unidades[Edelta.tag],Edelta.text);
end;

//Sacar los nuevos datos para introducirlo en los registro correspondientes//

procedure TFfisicos.sacar_datos;
begin
  with parametros_fisicos^ do begin
    k:=strTOfloat(copy(Ek.text,0,pos(' ',Ek.text)-1));
    c:=strTOfloat(copy(Ec.text,0,pos(' ',Ec.text)-1));
    p:=strTofloat(copy(Ep.text,0,pos(' ',Ep.text)-1));
    Espesor_aislante:=strTofloat(copy(Eespesor_aislante.text,0,pos(' ',Eespesor_aislante.text)-1));
    K_aislante:=strTofloat(copy(EK_aislante.text,0,pos(' ',EK_aislante.text)-1));
    U:=strTOfloat(copy(EUhorizontal.text,0,pos(' ',EUhorizontal.text)-1));
    delta:=strTofloat(copy(Edelta.text,0,pos(' ',Edelta.text)-1));
    H_arriba:=strTOfloat(copy(EHarriba.text,0,pos(' ',EHarriba.text)-1));
    H_abajo:=strTOfloat(copy(EHabajo.text,0,pos(' ',EHabajo.text)-1));
    H_borde_izquierdo:=strTOfloat(copy(Ehborde_izquierdo.text,0,pos(' ',EHborde_izquierdo.text)-1));
    H_borde_derecho:=strTOfloat(copy(Ehborde_derecho.text,0,pos(' ',EHborde_derecho.text)-1));
    H_borde_arriba:=strTOfloat(copy(Ehborde_arriba.text,0,pos(' ',EHborde_arriba.text)-1));
    H_borde_abajo:=strTOfloat(copy(Ehborde_abajo.text,0,pos(' ',EHborde_abajo.text)-1));
    Q:=strTOfloat(copy(Epotencia_dentro.text,0,pos(' ',Epotencia_dentro.text)-1));
    espesor:=strTOfloat(copy(Eespesor.text,0,pos(' ',Eespesor.text)-1));
    TOL:=strTOfloat(copy(ETOL.text,0,pos(' ',ETOL.text)-1));
    TMAX:=strTOfloat(copy(ETMAX.text,0,pos(' ',ETMAX.text)-1));
    temp_ambiente_arriba:=strTOfloat(copy(etemp_ambiente_arriba.text,0,pos('
',etemp_ambiente_arriba.text)-1));
    temp_ambiente_abajo:=strTOfloat(copy(etemp_ambiente_abajo.text,0,pos('
',etemp_ambiente_abajo.text)-1));
    iteraciones:=strTOint(copy(eiteraciones.text,0,pos(' ',eiteraciones.text)-1));
  end;
end;


//Para poner  la unidades cada vez que se salga del componente edit//

procedure TFfisicos.EVverticalKeyPress(Sender: TObject; var Key: Char);
begin
mmKeyPress(Sender, Key);
end;

//Para poner  la unidades cada vez que se salga del componente edit//

procedure TFfisicos.EVverticalExit(Sender: TObject);
begin
( SENDER as Tedit ).TEXT:=QUITAR_ERRORES(( SENDER AS TEDIT).TEXT);
 ( SENDER as Tedit ).TEXT:= poner_unidad(array_unidades[( SENDER AS TEDIT).tag],
      ( SENDER AS TEDIT).text);
end;

//Actualiza la ficha por primera vez//

procedure TFfisicos.FormCreate(Sender: TObject);
```

```
begin
digitos:=3;
rgh.itemindex:=2;
rgtp.itemindex:=2;
rgt.itemindex:=0;
top:=0;
left:=10;
end;

procedure TFfisicos.BitBtn3Click(Sender: TObject);
begin
close
end;

//Poner los métodos de la potencia y coeficiente de transferencia//

procedure TFfisicos.BitBtn2Click(Sender: TObject);
begin
try
sacar_datos;
calcular_parametros_fisicos(parametros_fisicos);
with parametros_fisicos^ do begin
 EDT.TEXT:=floattostrF(DT,ffFixed,8,digitos+4);
 case  rgh.itemindex of
  0: estilo_h:=EHforzado;
  1: estilo_h:=EHnatural;
  2: estilo_h:=EHCONSTANTE;
  3: estilo_h:=EHMANUAL;
  ELSE
   estilo_h:=EHCONSTANTE;
end;
if rgt.items[rgT.itemindex]='Constante' then
   estilo_ambiente:=EAconstante;
if rgt.items[rgT.itemindex]='Manual' then
estilo_ambiente:=EAManual;
case  rgtp.itemindex of
 0: estilo_potencia:=EPconstante;
 1: estilo_potencia:=EPmanual;
 2: estilo_potencia:=EPautomatico;
 3: estilo_potencia:=EPautomatico_manual;
end;
end;
except
     showmessage('Hay valores que no pueden ser convertidos (ceros o letras');
end;
modalresult:=mrnone;
end;

//Visualiza la tabla de las placas//

procedure TFfisicos.Button1Click(Sender: TObject);
begin
Ftabla_placas.visible:=true;
end;
end.
```

```
unit Fesquema;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus,Usacar_datos_protel,tipo1, ComCtrls, frstatus,surface,sor3b,
  calculo_h_forzado,calcula_h_natural,Usacar_datos_atributos,
StdCtrls,manual,Usacar_datos_ULPGC,lu;
 const
 numero_de_matrices=9;

type
   Tesquema = class(TForm)
   MainMenu1: TMainMenu;
   edit: TMenuItem;
   mover: TMenuItem;
   Mcomponente: TMenuItem;
   Entorno1: TMenuItem;
   RejillaCursor: TMenuItem;
   unmil1: TMenuItem;
   dosmil: TMenuItem;
   diezmil: TMenuItem;
   veintemil: TMenuItem;
   cuarentamil: TMenuItem;
   cienmil: TMenuItem;
   quinientosmil: TMenuItem;
   rejillavisible: TMenuItem;
   Vdiezmil: TMenuItem;
   Vveintemil: TMenuItem;
   Vcuarentamil: TMenuItem;
   unmilimetro: TMenuItem;
   Paleta1: TMenuItem;
   gray1: TMenuItem;
   cool1: TMenuItem;
   hot1: TMenuItem;
   estandar1: TMenuItem;
   TONALIDADES1: TMenuItem;
   visibleinvisible1: TMenuItem;
   forma1: TMenuItem;
   relleno1: TMenuItem;
   cuadrado1: TMenuItem;
   ocultarver1: TMenuItem;
   Otro1: TMenuItem;
   OtarRejillacursor1: TMenuItem;
   contorno1: TMenuItem;
   Manual1: TMenuItem;
   Simulacion: TMenuItem;
   sor41: TMenuItem;
   Capas1: TMenuItem;
   TEMPPLACAARRIBA1: TMenuItem;
   TEMPPLACAABAJO1: TMenuItem;
   COEFTRANSARRIBAH1: TMenuItem;
   COEFTRANSABAJOH1: TMenuItem;
   Datosfisicos1: TMenuItem;
   POTENCIACOMPONENTES1: TMenuItem;
   TEMPAMBIENTEABAJO1: TMenuItem;
   TEMPAMBIENTEARRIBA1: TMenuItem;
   rellenocuadrado1: TMenuItem;
   Vunmilimetro: TMenuItem;
```

```
Vdiesmilimetro: TMenuItem;
N10mm1: TMenuItem;
Imprimir1: TMenuItem;
circulos1: TMenuItem;
Ponerrectangulosuperficie1: TMenuItem;
nada1: TMenuItem;
Cambiarunidad1: TMenuItem;
Ponerdatosmanualmente1: TMenuItem;
procedure dosmilClick(Sender: TObject);
procedure diezmilClick(Sender: TObject);
procedure veintemilClick(Sender: TObject);
procedure cuarentamilClick(Sender: TObject);
procedure cienmilClick(Sender: TObject);
procedure quinientosmilClick(Sender: TObject);
procedure VdiezmilClick(Sender: TObject);
procedure VveintemilClick(Sender: TObject);
procedure VcuarentamilClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure FormDestroy(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
procedure FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure FormPaint(Sender: TObject);
procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure McomponenteClick(Sender: TObject);
procedure FormDblClick(Sender: TObject);
procedure FormResize(Sender: TObject);
procedure unmilimetroClick(Sender: TObject);
procedure gray1Click(Sender: TObject);
procedure TONALIDADES1Click(Sender: TObject);
procedure cool1Click(Sender: TObject);
procedure hot1Click(Sender: TObject);
procedure estandar1Click(Sender: TObject);
procedure ver_datos_matrizClick(Sender: TObject);
procedure FormActivate(Sender: TObject);
procedure relleno1Click(Sender: TObject);
procedure cuadrado1Click(Sender: TObject);
procedure ocultarver1Click(Sender: TObject);
procedure OtarRejillacursor1Click(Sender: TObject);
procedure Otro1Click(Sender: TObject);
procedure Manual1Click(Sender: TObject);
procedure sor41Click(Sender: TObject);
procedure TEMPPLACAABAJO1Click(Sender: TObject);
procedure TEMPPLACAARRIBA1Click(Sender: TObject);
procedure COEFTRANSARRIBAH1Click(Sender: TObject);
procedure COEFTRANSABAJOH1Click(Sender: TObject);
procedure Datosfisicos1Click(Sender: TObject);
procedure POTENCIACOMPONENTES1Click(Sender: TObject);
procedure cuadrado2Click(Sender: TObject);
procedure datos_manualesClick(Sender: TObject);
procedure TEMPAMBIENTEABAJO1Click(Sender: TObject);
procedure TEMPAMBIENTEARRIBA1Click(Sender: TObject);
procedure rellenocuadrado1Click(Sender: TObject);
procedure N10mm1Click(Sender: TObject);
procedure VunmilimetroClick(Sender: TObject);
procedure VdiesmilimetroClick(Sender: TObject);
procedure unmil1Click(Sender: TObject);
```

```
    procedure Imprimir1Click(Sender: TObject);
    procedure circulos1Click(Sender: TObject);
    procedure Ponerrectangulosuperficie1Click(Sender: TObject);
    procedure nada1Click(Sender: TObject);
    procedure Cambiarunidad1Click(Sender: TObject);
    procedure Ponerdatosmanualmente1Click(Sender: TObject);
  private
   private
   escala_anterior: real;
   ant_x,ant_y: integer;
   existe_archivo_esquema_atributos: boolean;
   atributos: Tatributos;
   ahora: boolean;
   rectangulo_general,rect_pantalla: Trect;
   indice,valor: integer;
   y_min,x_max,y_max,x_min,x_menor,Y_mayor,x_mayor,y_menor: real;
   x_aux,y_aux,ventana_pulgada: integer;
   puntero_comp: Tipolista;
   ajuste_x,ajuste_Y: integer;
   escala: real;
   X_minimo,Y_minimo,y_maximo,x_maximo,a_x,a_Y: real;
   mio,arrastrar,activar_arrastrar,aceptar_dibujo,seguir_desplazando: boolean;
   pixel_pulgada,
   pixel_milimetro: real;
   origen,centro: Tpoint;
   x1_aux,y2_aux: real;
   scroll_dibujo,scroll_dibujo_anterior,posicion_aux,ant_posicion: Tpoint;
   pos_horz,bbb: integer;
   mueve_raton: boolean;
   tipo_primitivo_x,tipo_primitivo_Y: real;
   cursor_x,cursor_Y: integer;
   max_numero_elementos: longint;  //es el numero de filas o columna de la matriz//
   rejilla_cursor_escalada,rejilla_visible_escalada,rejilla_cursor,rejilla_visible: real;
   parar,parar1,salir: boolean;
   unidad: string;
   FUNCION_TURBULENTO,FUNCION_LAMINAR: Tfuncion;
   parametros_fisicos: Tparametros_fisicos;

matriz_esp,matrizK,matrizT,matrizQ,matrizTAIRE_ABAJO,matrizTAIRE_ARRIBA,matrizH_ARRIBA,
matrizH_BORDE,matrizH_ABAJO,matriz,UP: Tmatriz;
   temp_max,temp_min,posicion_cursorX,posicion_cursorY,posicion_cursor_antX,posicion_cursor_antY:
real;
   rectangulo_cuadricula_escalado: trect;
   brocha: Tbrush;
   ventana: Tcanvas;
   array_colores: colores;
   estilo_color: Testilo_colores;
   tonalidades,digitos: integer;
   conjunto_dibujo:  Tconjunto_dibujos;
   estilo_forma: Testilo_forma;
   conjunto_pulsacion,conjunto_movimiento: Testilo_pulsacion;
   estilo_unidades,estilo_unidades_visible: Testilo_unidades;
   lista_rectangulo,rectangulo_manual: tipolista_rectangulo;
   estilo_formato_archivo: Testilo_formato_archivo;
   temp_ambiente: real;
   delta: real;
   rectangulo_aux: Trect;
   lista_rectas: tiporecta;

   { Private declarations }
```

```
    procedure setcapa(valor: Testilo_capa);
    function pos_barra_vert :integer;
    procedure dibujar;
    procedure dibuja_rectas(var puntero: tipolista);
    procedure dibuja_rectangulo;
    procedure dibuja_rect(var rectangulo: Trect;const modo_dibujo: TPenMode;
    x,y: real);
    procedure dibuja_cruz(x,y: integer; const modo_dibujo: TPenMode);
    procedure dibuja_letras(var puntero: tipolista);
    procedure dibuja_arco(var puntero: tipolista);
    procedure dibuja_pads(var puntero: tipolista);
    procedure dibuja_circulos(var puntero: tipolista);
    procedure dibuja_rectas_placas(var lista_rectas: tiporecta);
    procedure dibuja_componente(var puntero: tipolista; const modo_dibujo: TPenMode; const color:
Tcolor);
    procedure dibuja_lista_rectangulo(var lista_rectangulo: tipolista_rectangulo);
    procedure dibuja_simbolo_preparado(var puntero: tipolista);
    procedure dibuja_simbolo_desactivado(var puntero: tipolista);
    procedure scroll_componente(const scrollx,scrolly: REAL;
     var puntero: tipolista);
    procedure modo_pantalla_MMisotropic; //pantalla en modo milécimas de pulgada//
    procedure modo_pantalla_MM_HIENGLISH;
    procedure dibujar_colores(rect_pantalla: trect;VAR MATRIZ: tMATRIZ; temp_max,temp_min: real);
    procedure dibuja_rejilla;
   procedure nueva_posicion(x,y: real);
   procedure quitar_componente_lista(var puntero: Tipolista);
   procedure poner_componente_lista(var puntero: Tipolista);
   procedure configurar_surface;
   procedure poner_letra(var puntero: tipolista);
   procedure incluir_letras;
   function desplazamiento(var scroll: Tpoint): Tpoint;
   function detectar_rect( aux_x,aux_y: real; var lista,puntero:
   tipolista): boolean;
   procedure construir_rect_comp;
   procedure sumar_rectangulo( var rectangulo_variable,rectangulo_fijo: trectangulo);
   procedure localizar_cotornos;
   procedure quitar_cuadrado_lista(var lista_cuadrado,nodo: tipolista_rectangulo);
   function poner_cuadrado_lista(var lista_rectangulo: tipolista_rectangulo; const rectangulo: Trect):
tipolista_rectangulo;
   procedure coreccion_rectangulo(var rectangulo: Trect);
   procedure eliminar_componente(var lista,nodo: Tipolista);
   procedure elimina_contenido_componente(var nodo: Tipolista);
   procedure sacar_componente;
   procedure poner_delta;
   procedure  posicionar_letras_ULPGC(var lista: tipolista);
   procedure poner_datos_manuales;
   procedure WMHSCROLL(var  message: TWMHSCROLL); message WM_HSCROLL;
   procedure WMVSCROLL(var  message: TWMVSCROLL); message WM_VSCROLL;
   property capa: Testilo_capa  write setcapa default ECTEMP_PLACA_ARRIBA;
   procedure dibuja_rectangulo_componte(var rectangulo_componente: Trectangulo);
   procedure zoom(key: word; x,y: integer);
   procedure cambiar_rectangulo_radiacion(var lista: tipolista; var rectangulo: Trect);
   procedure actualiza_rect_diagrama(var rectangulo_variable: Trectangulo;var lista: tipolista);
   procedure elimina_rectas_placa(var lista_rectas: tiporecta);

 public
   archivo_esquema: text;
   archivo_esquema_atributos:  Tstringlist;
   lista: tipolista;
```

```
    rectangulo_diagrama: Trectangulo;
  procedure  centra_punto_esquema;
  procedure poner_delta_rectangulo(delta_aux: real; var rectangulo: Trect);
  procedure construir_matrices( fila,columna: integer);
  function introducir_datos(var puntero: tipolista): boolean;

    { Public declarations }
   end;

var
  esquema: Tesquema;

implementation

uses PADRE,ocultar,Verdatos, DATOS_FISICOS, unidad;

{$R *.DFM}



{procedimiento que se utiliza para saber que parte de la pantalla se esta moviendo}

 procedure Tesquema.WMHSCROLL(var  message: TWMHSCROLL);
  begin
    inherited;
  case message.ScrollCode of

  SB_LINELEFT:
              if (parar and  parar1 ) then begin
              setrect(rect_pantalla,0,clientheight,horzScrollBar.increment,0);
              parar:=false;
              parar1:=false;
              end
              else
                 parar:=true;
  SB_LINERIGHT: if (parar and  parar1 ) then begin
             setrect(rect_pantalla,clientwidth-horzScrollBar.increment,clientheight,clientwidth,0);
              parar:=false;
              parar1:=false;
              end
              else
                 parar:=true;
  end;
  end;

{Procedimiento que se utiliza para saber que parte de la pantalla se esta moviendo}

  procedure Tesquema.WMVSCROLL(var  message: TWMVSCROLL);
  begin
    inherited;
  case message.ScrollCode of
  SB_LINEUP: if (parar and  parar1 ) then begin
              setrect(rect_pantalla,0,clientheight,clientwidth,clientheight-vertScrollBar.increment);
              parar:=false;
              parar1:=false;
              end
              else
                 parar:=true;
  SB_LINEDOWN:   if (parar and  parar1 ) then begin
             setrect(rect_pantalla,0,vertScrollBar.increment,clientwidth,0);
```

```
            parar:=false;
             parar1:=false;
             end
             else
                parar:=true;
  end;
  end;

 //Configuración de la superfice de colores ( cuadrícula de colores) //

 procedure Tesquema.configurar_surface;
 begin
 brocha:=canvas.brush;
 ventana:=canvas;
 temp_max:=65;
 temp_min:=0;
 tonalidades:=25;
 end;

 {Dectecta la nueva posición del cursor, debido a que se
 a pulsado algunas de las fechas del teclado}

 procedure Tesquema.nueva_posicion(x,y: real);
  begin
  rejilla_cursor_escalada:=rejilla_cursor * escala;
  x:=x + pixel_mil(horzScrollBar.position);
  y:=y + pixel_mil(pos_barra_vert);
  posicion_cursorX:=round( x / rejilla_cursor_escalada) * rejilla_cursor;
  posicion_cursorY:=round( y / rejilla_cursor_escalada) * rejilla_cursor;
  end;

 (*@**********************************************************************
  procedimientos para dibujar todos los detalles de los componentes
  más la rejilla, colores del gradiente
  **********************************************************************)

 procedure Tesquema.dibujar;
 var
 prueba: Tipolista;
 x_entero,y_entero,n,aux,f: integer;
 aux_rect,rectangulo_salida: Trect;
 scroll_mil: Tpoint;
 begin
 scroll_dibujo:=desplazamiento(scroll_dibujo);
 if parar then
    setrect(rect_pantalla,0,clientheight,clientwidth,0);

 //DIBUJA LOS COLORES//
 if CDcolores in conjunto_dibujo then
  dibujar_colores(rect_pantalla,matriz,matriz.max_valor,matriz.min_valor);

  dibuja_rectangulo_componte(rectangulo_diagrama);
 //DIBUJA LA REJILLA//
 if CDrejilla in conjunto_dibujo  then
  dibuja_rejilla;

 canvas.pen.color:=clyellow;
 scroll_dibujo:=desplazamiento(scroll_dibujo);
 //CAMBIO DE MODO DE PANTALLA//
 modo_pantalla_MMisotropic;
```

```
 //DIBUJA RECTAS DE LA PLACA//
if CDrectas_placa in conjunto_dibujo  then
dibuja_rectas_placas(lista_rectas);

//DIBUJA RECTANGULOS DE LAS LETRAS DE LOS COMPONENTES//
if CDrectangulo in conjunto_dibujo  then
dibuja_rectangulo;

CANVAS.PEN.MODE:=PMCOPY;
canvas.brush.style:=bsclear;
prueba:=lista;
while prueba <> nil do
begin
  aux_rect.left:=round(prueba.rectangulo.izquierda*PixelsPerInch*escala-scroll_dibujo.x);
  aux_rect.right:=round(prueba.rectangulo.derecha*PixelsPerInch*escala-scroll_dibujo.x);
  aux_rect.top:=round(prueba.rectangulo.arriba*PixelsPerInch*escala-scroll_dibujo.y);
  aux_rect.bottom:=round(prueba.rectangulo.abajo*PixelsPerInch*escala-scroll_dibujo.y);
 if  intersrect2(aux_rect,rect_pantalla) then
  begin

     //DIBUJA RECTAS//
     if CDrectas in conjunto_dibujo  then
      dibuja_rectas(prueba);

     //DIBAJA_ARCOS//
     if CDarcos in conjunto_dibujo  then
      dibuja_arco(prueba);

     //DIBUJA_CIRCULOS//
     if CDcirculos in conjunto_dibujo  then
     dibuja_circulos(prueba);

      //DIBUJA_PADS//
     if CDpads in conjunto_dibujo  then
     dibuja_pads(prueba);

     //DIBUJA_LETRAS//
     if CDletras in conjunto_dibujo  then
     dibuja_letras(prueba);

     //DIBUJO_RECTANGULO_COMPONENTE//
     if CDrectagulo_componente in conjunto_dibujo  then
     dibuja_rectangulo_componte(prueba.rectangulo_componente);

     //DIBUJO_SIMBOLO_PREPARADO//
     if CDSIMBOLO_PREPARADO in conjunto_dibujo  then
     dibuja_simbolo_preparado(prueba);

      //DIBUJA_SIMBOLO_DESACTIVADO//
     if CDSIMBOLO_DESACTIVADO in conjunto_dibujo then
     dibuja_simbolo_desactivado(prueba);

    end;
prueba:=prueba.prox;
end;

//DIBUJA RECTANGULOS de la lista rectangulos//
dibuja_lista_rectangulo(lista_rectangulo);
end;
```

```
//Dibuja rectas de la placa, no confundir con las del componente//

procedure Tesquema.dibuja_rectas_placas(var lista_rectas: tiporecta);
var
X_entero,Y_entero,i: integer;
grosor,colorr: integer;
puntero: tiporecta;
begin
grosor:=canvas.pen.width;
colorr:=canvas.pen.color;
canvas.pen.color:=clred;
puntero:=lista_rectas;
while puntero <> nil do begin
with  puntero^ do begin
canvas.pen.width:=round(PixelsPerInch*ancho*escala);
        X_entero:=round(x1 * PixelsPerInch*escala-scroll_dibujo.x);
        Y_entero:=round(y1 * PixelsPerInch*escala-scroll_dibujo.y);
        canvas.moveTo(x_entero,y_entero);
        X_entero:=round(x2 * PixelsPerInch*escala-scroll_dibujo.x);
        Y_entero:=round(y2 * PixelsPerInch*escala-scroll_dibujo.y);
        canvas.lineto(x_entero,y_entero);
        puntero:=puntero.prox;
        end;
end;
canvas.pen.color:=colorr;
canvas.pen.width:=grosor;
end;

//Dibuja rectas del componente//
procedure tesquema.dibuja_rectas(var puntero: tipolista);
var
X_entero,Y_entero,i: integer;
recta: tiporecta;
grosor,colorr: integer;
begin
grosor:=canvas.pen.width;
colorr:=canvas.pen.color;
canvas.pen.width:=round(PixelsPerInch*puntero.estructura.ancho*escala);
recta:=puntero.estructura.recta;
     while recta <> nil do
        begin
        X_entero:=round(recta.x1 * PixelsPerInch*escala-scroll_dibujo.x);
        Y_entero:=round(recta.y1 * PixelsPerInch*escala-scroll_dibujo.y);
        canvas.moveTo(x_entero,y_entero);
        X_entero:=round(recta.x2 * PixelsPerInch*escala-scroll_dibujo.x);
        Y_entero:=round(recta.y2 * PixelsPerInch*escala-scroll_dibujo.y);
        canvas.lineto(x_entero,y_entero);
        recta:=recta.prox;
        end;
canvas.pen.color:=colorr;
canvas.pen.width:=grosor;
end;

//Dibuja rectas del componente//

procedure Tesquema.dibuja_letras(var puntero: tipolista);
var
puntero_letra: tiporecorre_letras;
contador: integer;
```

```
x,y,x2,y2,ancho: real;
x3,y3,i: integer;
 grosor,colorr: integer;
begin
if escala>=1 then
begin
grosor:=canvas.pen.width;
colorr:=canvas.pen.color;
canvas.pen.width:=round(PixelsPerInch*0.006*escala);
  for i:=1 to 1  do
  begin
     ancho:=0;
     puntero_letra:=puntero.letras_referencia;
     while puntero_letra <> nil do
     begin
        contador:=4;
        with puntero_letra.posicion_letra.letras do
        begin
            while contador<=final_letra do
            begin
            x:=letra_array[contador].x/1000000+ancho;
            y:=letra_array[contador].y/1000000;
            x2:=x*cos(puntero.angulo_letra)-y*sin(puntero.angulo_letra)+puntero.inicio_letrax;
            y2:=y*cos(puntero.angulo_letra)+x*sin(puntero.angulo_letra)+puntero.inicio_letray;
             X3:=round(x2*PixelsPerInch*escala-scroll_dibujo.x);
             Y3:=round(y2*PixelsPerInch*escala-scroll_dibujo.y);
            canvas.moveTo(x3,y3);
            x:=letra_array[contador+1].x/1000000+ancho;
            y:=letra_array[contador+1].y/1000000;
            x2:=x*cos(puntero.angulo_letra)-y*sin(puntero.angulo_letra)+puntero.inicio_letrax;
            y2:=y*cos(puntero.angulo_letra)+x*sin(puntero.angulo_letra)+puntero.inicio_letray;
            X3:=round(x2*PixelsPerInch*escala-scroll_dibujo.x);
            Y3:=round(y2*PixelsPerInch*escala-scroll_dibujo.y);
             canvas.lineto(x3,y3);
            contador:=contador + 2;
           end;
       end;
     puntero_letra:=puntero_letra.prox;
     ancho:=ancho+ancho_letra;
      end;
  canvas.pen.width:=1;
  canvas.pen.color:=clblue;
  end;
  canvas.pen.color:=colorr;
  canvas.pen.width:=grosor;
 end;
end;

//Dibuja el rectangulo que abarcará el componente//

procedure Tesquema.dibuja_rectangulo;
var
prueba: tipolista;
izquierdaE,arribaE,derechaE,abajoE: integer;
 grosor,colorr: integer;
begin
canvas.brush.color:=clblack;
prueba:=lista;
CANVAS.PEN.MODE:=PMCOPY;
grosor:=canvas.pen.width;
```

```
colorr:=canvas.pen.color;
canvas.pen.width:=1;
if (escala<2) and (estilo_formato_archivo<>EFAULPGC) then
begin
    while prueba <> nil do
    begin
        with prueba.rectangulo_letra do
          begin
          izquierdaE:=round(izquierda*PixelsPerInch *escala-scroll_dibujo.x);
          derechaE:=round(derecha*PixelsPerInch*escala-scroll_dibujo.x);
          abajoE:=round(abajo*PixelsPerInch*escala-scroll_dibujo.y);
          arribaE:=round(arriba*PixelsPerInch*escala-scroll_dibujo.y);
          canvas.rectangle(izquierdaE,arribaE,derechaE,abajoE);
          end;
prueba:=prueba.prox;
    end;
end;
 canvas.pen.color:=colorr;
 canvas.pen.width:=grosor;
end;

//Dija arcos del componente//

procedure Tesquema.dibuja_arco(var puntero: tipolista);
var
grosor,colorr: integer;
puntero_aux: tipoarco;
begin
grosor:=canvas.pen.width;
colorr:=canvas.pen.color;
puntero_aux:=puntero.arcos;
while puntero_aux <> nil do
begin
    with puntero_aux.arco do
    begin
    canvas.pen.width:=round(PixelsPerInch*ancho*escala);
    canvas.arc(round(rectangulo.izquierda*PixelsPerInch *escala-scroll_dibujo.x),
          round(rectangulo.arriba*PixelsPerInch*escala-scroll_dibujo.y),
          round(rectangulo.derecha*PixelsPerInch*escala-scroll_dibujo.x),
          round(rectangulo.abajo*PixelsPerInch*escala-scroll_dibujo.y),
          round(inicio_arcox*PixelsPerInch*escala-scroll_dibujo.x),
          round(inicio_arcoy*PixelsPerInch*escala-scroll_dibujo.y),
          round(fin_arcox*PixelsPerInch*escala-scroll_dibujo.x),
          round(fin_arcoy*PixelsPerInch*escala-scroll_dibujo.y));
    end;
puntero_aux:=puntero_aux.prox;
end;
canvas.pen.color:=colorr;
canvas.pen.width:=grosor;
end;

//Dibuja circulos del componente//

procedure Tesquema.dibuja_circulos(var puntero: tipolista);
var
grosor,colorr: integer;
circulos: tipocirculo;
begin
grosor:=canvas.pen.width;
colorr:=canvas.pen.color;
```

```
circulos:=puntero.circulos;
canvas.pen.width:=1;
while circulos <> nil do  begin
    canvas.ellipse(round(circulos.rectangulo.izquierda*PixelsPerInch *escala-scroll_dibujo.x),
    round(circulos.rectangulo.arriba*PixelsPerInch*escala-scroll_dibujo.y),
    round(circulos.rectangulo.derecha*PixelsPerInch*escala-scroll_dibujo.x),
    round(circulos.rectangulo.abajo*PixelsPerInch*escala-scroll_dibujo.y));
circulos:=circulos.prox;
end;
canvas.pen.color:=colorr;
canvas.pen.width:=grosor;
end;

//Dibuja pads del componente//

procedure tesquema.dibuja_pads(var puntero: tipolista);
var
grosor,colorr: integer;
pad: tipopad;
begin
grosor:=canvas.pen.width;
colorr:=canvas.pen.color;
pad:=puntero.pad;
canvas.pen.width:=1;
while pad <> nil do  begin
    canvas.brush.color:=clsilver;
   Canvas.ellipse(round(pad.rectangulo_exterior.izquierda*PixelsPerInch *escala-scroll_dibujo.x),
        round(pad.rectangulo_exterior.arriba*PixelsPerInch*escala-scroll_dibujo.y),
        round(pad.rectangulo_exterior.derecha*PixelsPerInch*escala-scroll_dibujo.x),
        round(pad.rectangulo_exterior.abajo*PixelsPerInch*escala-scroll_dibujo.y));
   canvas.brush.color:=clblue;
   Canvas.ellipse(round(pad.rectangulo_interior.izquierda*PixelsPerInch *escala-scroll_dibujo.x),
        round(pad.rectangulo_interior.arriba*PixelsPerInch*escala-scroll_dibujo.y),
        round(pad.rectangulo_interior.derecha*PixelsPerInch*escala-scroll_dibujo.x),
        round(pad.rectangulo_interior.abajo*PixelsPerInch*escala-scroll_dibujo.y));
  pad:=pad.prox;
end;
canvas.pen.color:=colorr;
canvas.pen.width:=grosor;
end;

//Dibuja rejilla del esquema//

procedure Tesquema.dibuja_rejilla;
var
x,y: integer;
PUNTO: TPOINT;
ajusteX_barra,ajusteY_barra: real;
begin
rejilla_visible_escalada:=rejilla_visible*escala;
ajusteX_barra:=frac(pixel_mil(horzScrollBar.position) / rejilla_visible_escalada)*
rejilla_visible_escalada;
ajusteX_barra:=rejilla_visible_escalada - ajusteX_barra;
ajustey_barra:= frac(pixel_mil(pos_barra_vert) / rejilla_visible_escalada)* rejilla_visible_escalada;
ajusteY_barra:=rejilla_visible_escalada-ajustey_barra;
SetmapMode(canvas.handle,MM_HIENGLISH        );
SetViewportOrgEx(canvas.handle,0,clientheight,nil);
if rejilla_visible_escalada>=75  then
begin
punto.x:=pixel_mil_ajustado(clientwidth/rejilla_visible_escalada);
```

```
punto.y:=pixel_mil_ajustado(clientheight/rejilla_visible_escalada);
for y:=0 to punto.y do
for x:=0 to punto.x do
   canvas.pixels[ROUND(x*rejilla_visible_escalada+ajusteX_barra),
   round(y*rejilla_visible_escalada+ajusteY_barra)]:=CLWHITE;
end;
end;

//Dibuja colores del gradiente de temperatura//

procedure Tesquema.dibujar_colores(rect_pantalla: trect;VAR MATRIZ: tMATRIZ;
temp_max,temp_min: real);
begin
offsetrect(rect_pantalla, scroll_dibujo.x, scroll_dibujo.y);
setrect(rectangulo_cuadricula_escalado,round(rectangulo_aux.left*escala),
round(rectangulo_aux.top*escala),round(rectangulo_aux.right*escala),round(rectangulo_aux.bottom*esca
la));
SetmapMode(canvas.handle,MM_HIENGLISH        );
SetViewportOrgEx(canvas.handle,0,clientheight,nil);
try
Psurface(matriz,array_colores, rectangulo_cuadricula_escalado,rect_pantalla,
tonalidades,delta,temp_ambiente,matriz.max_valor,matriz.min_valor, escala,scroll_dibujo, ventana,
brocha, estilo_forma);
except begin
    showmessage('dato para visualizar incorrecto');
    estilo_forma:=cuadrado;
    end;
end;
end;

//Dibuja rectangulos auxiliares//

procedure Tesquema.dibuja_rect(var rectangulo: Trect;const modo_dibujo: TPenMode;
x,y: real);
var
izquierda,arriba,derecha,abajo: integer;
colorr: integer;
begin
canvas.pen.width:=1;
canvas.pen.color:=clwhite;
canvas.brush.style:=bsclear;
canvas.pen.mode:=modo_dibujo;
modo_pantalla_MMisotropic;  //se dibuja en pixeles con el punto 0,0 en el borde infereior izquierdo//
with rectangulo do begin
right:=round(x);
bottom:=round(y);
izquierda:=round(((left/1000)*PixelsPerInch *escala)-scroll_dibujo.x);
derecha:=round(((x/1000)*PixelsPerInch*escala)-scroll_dibujo.x);
abajo:=round(((y/1000)*PixelsPerInch*escala)-scroll_dibujo.y);
arriba:=round(((top/1000)*PixelsPerInch*escala)-scroll_dibujo.y);
end;
canvas.rectangle(izquierda,arriba,derecha,abajo);
end;

//Dibuja a v de componente preparado//

procedure Tesquema.dibuja_simbolo_preparado(var puntero: tipolista);
var
X_entero,Y_entero,i: integer;
x,y: real;
```

```
grosor,colorr: integer;
rectangulo: trectangulo;
begin
if  puntero.preparado and  puntero.atributos.activo then begin
grosor:=canvas.pen.width;
colorr:=canvas.pen.color;
canvas.pen.width:=round(PixelsPerInch*0.030*escala);
canvas.pen.color:=clgreen;
rectangulo:=puntero.rectangulo;
with rectangulo do begin
x:=(derecha-izquierda)/4;
y:=(arriba-abajo)/4;
izquierda:=izquierda+x;
derecha:=derecha-x;
arriba:=arriba-y;
abajo:=abajo+y;
x:=izquierda;
y:=(arriba-abajo)/2+abajo;
X_entero:=round(x * PixelsPerInch*escala-scroll_dibujo.x);
Y_entero:=round(y * PixelsPerInch*escala-scroll_dibujo.y);
canvas.moveTo(x_entero,y_entero);
x:=(derecha-izquierda)/2+izquierda;
y:=abajo;
X_entero:=round(x * PixelsPerInch*escala-scroll_dibujo.x);
Y_entero:=round(y * PixelsPerInch*escala-scroll_dibujo.y);
canvas.lineto(x_entero,y_entero);
x:=derecha;
y:=arriba;
X_entero:=round(x * PixelsPerInch*escala-scroll_dibujo.x);
Y_entero:=round(y * PixelsPerInch*escala-scroll_dibujo.y);
canvas.lineto(x_entero,y_entero);
end;
canvas.pen.color:=colorr;
canvas.pen.width:=grosor;
end;
end;

//dibuja la x de componente desactivado//

procedure Tesquema.dibuja_simbolo_desactivado(var puntero: tipolista);
var
X_entero,Y_entero,i: integer;
x,y: real;
grosor,colorr: integer;
rectangulo: trectangulo;
begin
if  not(puntero.atributos.activo) then begin
grosor:=canvas.pen.width;
colorr:=canvas.pen.color;
canvas.pen.width:=round(PixelsPerInch*0.030*escala);
canvas.pen.color:=clred;
rectangulo:=puntero.rectangulo;
with rectangulo do begin
x:=(derecha-izquierda)/4;
y:=(arriba-abajo)/4;
izquierda:=izquierda+x;
derecha:=derecha-x;
arriba:=arriba-y;
abajo:=abajo+y;
x:=derecha;
```

```
y:=abajo;
X_entero:=round(x * PixelsPerInch*escala-scroll_dibujo.x);
Y_entero:=round(y * PixelsPerInch*escala-scroll_dibujo.y);
canvas.moveTo(x_entero,y_entero);
x:=izquierda;
y:=arriba;
X_entero:=round(x * PixelsPerInch*escala-scroll_dibujo.x);
Y_entero:=round(y * PixelsPerInch*escala-scroll_dibujo.y);
canvas.lineto(x_entero,y_entero);
x:=izquierda;
y:=abajo;
X_entero:=round(x * PixelsPerInch*escala-scroll_dibujo.x);
Y_entero:=round(y * PixelsPerInch*escala-scroll_dibujo.y);
canvas.moveTo(x_entero,y_entero);
x:=derecha;
y:=arriba;
X_entero:=round(x * PixelsPerInch*escala-scroll_dibujo.x);
Y_entero:=round(y * PixelsPerInch*escala-scroll_dibujo.y);
canvas.lineto(x_entero,y_entero);
end;
canvas.pen.color:=colorr;
canvas.pen.width:=grosor;
end;
end;

//cambia presición rejilla a 2 mil//

procedure Tesquema.dosmilClick(Sender: TObject);
begin
rejilla_cursor:=2;
estilo_unidades:=EUpulgadas;
end;

//cambia presición rejilla a 10 mil//

procedure Tesquema.diezmilClick(Sender: TObject);
begin
rejilla_cursor:=10;
estilo_unidades:=EUpulgadas;
end;

//cambia presición rejilla a 20 mil//

procedure Tesquema.veintemilClick(Sender: TObject);
begin
rejilla_cursor:=20;
estilo_unidades:=EUpulgadas;
end;

//cambia presición rejilla a 40 mil//

procedure Tesquema.cuarentamilClick(Sender: TObject);
begin
rejilla_cursor:=40;
estilo_unidades:=EUpulgadas;
end;

//cambia presición rejilla a 100 mil//

procedure Tesquema.cienmilClick(Sender: TObject);
```

```
begin
rejilla_cursor:=100;
estilo_unidades:=EUpulgadas;
end;

//cambia presición rejilla a 500 mil//

procedure Tesquema.quinientosmilClick(Sender: TObject);
begin
rejilla_cursor:=500;
estilo_unidades:=EUpulgadas;
end;

//cambia presición rejilla visible a 10 mil//

procedure Tesquema.VdiezmilClick(Sender: TObject);
begin
rejilla_visible:=10;
estilo_unidades_visible:=EUpulgadas;
end;

//cambia presición rejilla visible a 20 mil//

procedure Tesquema.VveintemilClick(Sender: TObject);
begin
rejilla_visible:=20;
estilo_unidades_visible:=EUpulgadas;
end;

//cambia presición rejilla visible a 40 mil//

procedure Tesquema.VcuarentamilClick(Sender: TObject);
begin
rejilla_visible:=40;
estilo_unidades_visible:=EUpulgadas;
end;

{construye ventana esquema y inicializa todos sus datos
abre el archivo seleccionado, abre archivo de caracteristicas
térmicas asociado al esquema seleccionado}

procedure Tesquema.FormCreate(Sender: TObject);
var
valor: real;
columna,fila,i: integer;
apto: integer;
cadena: string;
begin
construir_matriz;
existe_archivo_esquema_atributos:=false;
archivo_esquema_atributos:=Tstringlist.create;
ahora:=false;
salir:=false;
tipo1.PixelsPerInch:=PixelsPerInch;
TRY
AssignFile(archivo_esquema, Fprincipal.ODabrir.filename);
Reset(archivo_esquema); { Intentar la apertura }
Except On Exception Do { si también falla }
  Begin
       ShowMessage('No es posible abrir el archivo'+extractfilename(Fprincipal.ODabrir.filename));
```

```
        close;
    end;
end;
cadena:=Fprincipal.ODabrir.filename;
delete(cadena,pos('.',cadena),length(cadena));
cadena:=cadena+'.txt';
if  fileexists(cadena) then begin
archivo_esquema_atributos.loadfromfile(cadena);
existe_archivo_esquema_atributos:=true;
end;
lista:=nil;
lista_rectas:=nil;
color:=clblack;
canvas.pen.color:=clyellow;
temp_ambiente:=25;
digitos:=0;
tipo1.PixelsPerInch:=PixelsPerInch;
vertScrollBar.range:=300000;
horzScrollBar.range:=300000;
mueve_raton:=true;
escala:=1;
tonalidades:=65;
estilo_color:=estandar;
paleta_estilo(array_colores,tonalidades,estilo_color);
Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.TMAX,@parametros_fisicos,matr
iz,matrizq,unidad );
vertScrollBar.position:=vertScrollBar.range;
sacar_componente;
if lista= nil then
exit;
configurar_surface;
conjunto_dibujo:=[CDrectas..CDsimbolo_desactivado];{-[CDcolores];}
estilo_forma:=cuadrado;
lista_rectangulo:=nil;
FUNCION_TURBULENTO:=Nussent_forzado_turb1;
FUNCION_LAMINAR:=Nussent_forzado_lami1;
delta:=60;
estilo_unidades:=EUpulgadas;
estilo_unidades_visible:=EUpulgadas;
rejilla_cursor :=1;
rejilla_visible:=10;
mueve_raton:=true;
lista_rectangulo:=nil;
estilo_forma:=circulo;
capa:=ECTEMP_PLACA_ARRIBA;
REPAINT;
centro_rect(lista,rectangulo_aux,delta,1);
arrastrar:=false;
Fprincipal.Actualiza_interfaz(EIficha_creada);
for I := 0 to visibleinvisible1.Count-1 do
visibleinvisible1.Items[I].Enabled := false;
for I := 0 to forma1.Count-1 do
forma1.Items[I].Enabled :=false;
for I := 0 to simulacion.Count-1 do
simulacion.Items[I].Enabled := false;
Ponerdatosmanualmente1.Enabled := false;
ocultarver1.Enabled := true;
Fprincipal.SPabrir.enabled:=true;
end;
```

```
//saca letras del archivo de letras y las asocia a los componentes//

procedure Tesquema.poner_letra(var puntero: tipolista);
var
tipo_letra: tiporecorre_letras;
puntero_entrada: tipoletra;
i: integer;
begin
for i:=length(puntero.atributos.referencia) downto 1 do
begin
    if existe_letra(Upcase(puntero.atributos.referencia[i]),lista_letras, puntero_entrada ) then
    begin
        new(tipo_letra);
        tipo_letra.posicion_letra:=puntero_entrada;
        tipo_letra.prox:=puntero.letras_referencia;
        puntero.letras_referencia:=tipo_letra;
    end;
end;
end;

{Se va pasando cada uno de los componnetes para sacar las
letras del archivo de letras}

procedure Tesquema.incluir_letras;
var
prueba: tipolista;
begin
prueba:=lista;
while prueba<> nil do
begin
poner_letra(prueba);
prueba:=prueba.prox;
end;
end;

{Se saca los componentes del archivo abierto, antes se averigua si es
procedente de protel o de Roën}

procedure Tesquema.sacar_componente;
var
i,x,y: integer;
posicion: Tpoint;
cadena: string[125];
prueba: TRECT;
puntero: tipolista;
begin
readln(archivo_esquema,cadena);
if pos(cabecera_protel,cadena)<>0 then begin
extraer_datos_protel(archivo_esquema,lista,lista_rectas);
estilo_formato_archivo:=EFAPROTEL;
end
else
if pos(cabecera_Roen,cadena)<>0 then begin
extraer_datos_ULPGC(archivo_esquema,lista);
estilo_formato_archivo:=EFAULPGC;
end
else begin
  ShowMessage('No es posible abrir el archivo '+extractfilename(Fprincipal.ODabrir.filename)
  +' Posiblemente no es del tipo adecuado o es otra versión');
```

```
  close;
  exit;
end;
if existe_archivo_esquema_atributos then
rellenar_lista_atributos(lista, archivo_esquema_atributos,@parametros_fisicos);
incluir_letras;
if lista=nil then
exit;
construir_rect_comp;
centra_punto_esquema;
puntero:=lista;
while puntero<> nil do begin
puntero.atributos.activo:=true;
puntero.preparado:=false;
if puntero.atributos.numero_pines>4 then
puntero.atributos.estilo_modo_potencia:=EMPcomponentes
else
puntero.atributos.estilo_modo_potencia:=EMPpines;
puntero:=puntero.prox;
end;
getcursorpos(posicion);
posicion:=ScreenTOclient(posicion);
punto_pixel_mil(posicion);
posicion_aux:=posicion;
rejilla_cursor_escalada:=rejilla_cursor*escala;
rejilla_visible_escalada:=rejilla_visible*escala;
end;

{Se obtiene el rectangulo que abarca el componente
incluido las dimensiones de las letras}

procedure Tesquema.construir_rect_comp;
var
prueba: Tipolista;
i: integer;
rectangulo_salida: trect;
pads: tipopad;
circulos: tipocirculo;
arcos: tipoarco;
begin
prueba:=lista;
if prueba.estructura.recta<> nil then begin
rectangulo_diagrama.izquierda:=prueba.estructura.recta.x1;
rectangulo_diagrama.derecha:=prueba.estructura.recta.x1;
rectangulo_diagrama.abajo:=prueba.estructura.recta.y1;
rectangulo_diagrama.arriba:=prueba.estructura.recta.y1;
end
else
if prueba.pad<> nil then
  rectangulo_diagrama:=prueba.pad.rectangulo_exterior
else
if prueba.circulos<> nil then
   rectangulo_diagrama:=prueba.circulos.rectangulo;
    while prueba <> nil do
    BEGIN
         obtener_rect(prueba);
        pads:=prueba.pad;
        while pads <> nil do
        begin
           sumar_rectangulo(prueba.rectangulo, pads.rectangulo_exterior);
```

```
            pads:=pads.prox;
          end;
          circulos:=prueba.circulos;
          while circulos <> nil do
          begin
            sumar_rectangulo(prueba.rectangulo, circulos.rectangulo);
            sumar_rectangulo(prueba.rectangulo_componente,circulos.rectangulo);
            circulos:=circulos.prox;
          end;
          arcos:=prueba.arcos;
          while arcos <> nil do
          begin
            sumar_rectangulo(prueba.rectangulo, arcos.arco.rectangulo);
            sumar_rectangulo(prueba.rectangulo_componente,arcos.arco.rectangulo);
            arcos:=arcos.prox;
          end;
          sumar_rectangulo(rectangulo_diagrama, prueba.rectangulo);
      prueba:=prueba.prox;
      end;
if estilo_formato_archivo=EFAULPGC then
  posicionar_letras_ULPGC(lista);
end;

{ se suma los distintos rectangulos de las partes del componente para
forma el rectangulo capsula (rectángulo que contine el componente) }

procedure Tesquema.sumar_rectangulo( var rectangulo_variable,rectangulo_fijo: trectangulo  );
 begin
      with rectangulo_fijo do
      begin
         if rectangulo_variable.izquierda>izquierda then
         rectangulo_variable.izquierda:=izquierda;
         if rectangulo_variable.derecha<derecha then
         rectangulo_variable.derecha:=derecha;
         if rectangulo_variable.arriba<arriba then
         rectangulo_variable.arriba:=arriba;
         if rectangulo_variable.abajo>abajo then
         rectangulo_variable.abajo:=abajo;
       end;
end;

//Destruye todas las lista creadas y las matrices //

procedure Tesquema.FormDestroy(Sender: TObject);
var
puntero: Tipolista;
begin
while lista<> nil do
begin
elimina_contenido_componente(lista);
elimina_rectas_placa(lista_rectas);
puntero:=lista;
lista:=lista^.prox;
dispose(puntero);
end;
destruir_dynamic_matriz(matrizT);
destruir_dynamic_matriz(matrizQ);
destruir_dynamic_matriz(matrizK);
destruir_dynamic_matriz(MATRIZTAIRE_ARRIBA);
destruir_dynamic_matriz(MATRIZTAIRE_ABAJO);
```

```
destruir_dynamic_matriz(MATRIZH_ARRIBA);
destruir_dynamic_matriz(MATRIZH_ABAJO);
destruir_dynamic_matriz(UP);
destruir_dynamic_matriz(matriz_esp);
end;

{Se averigua cuanto el desplazamiento que hay que realizar
según la posición de las barras de scroll}

function tesquema.desplazamiento(var scroll: Tpoint): Tpoint;
 begin
 scroll_dibujo_anterior:=scroll;
 result.x:=horzScrollBar.position;
 result.y:=vertScrollBar.range-clientheight-vertScrollBar.position;
 end;

 //Cierre de ficha//

procedure Tesquema.FormClose(Sender: TObject; var Action: TCloseAction);
begin
CloseFile(archivo_esquema);
archivo_esquema_atributos.free;
 if  extractfileext(caption) = '.pcb' then
   Fprincipal.MRUFileList.additem(caption);
   free;
   Fprincipal.Actualiza_interfaz(EIficha_destruida);
end;

{evemto de movimiento del ratón:
-       movimiento del cursor
-       movimiento de componetes
-        movimiento de rectángulos de diferentes tipo }

procedure Tesquema.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
var
punto,posicion: Tpoint;
r: integer;
begin
if mueve_raton then   {esto es para si nos estemos moviendonos bien con el raton o bien con el cursor}
  nueva_posicion(pixel_mil(x),pixel_mil(clientheight-y))
else
 mueve_raton:=true;
 if (conjunto_movimiento<>vacio) or (conjunto_pulsacion<>vacio) then begin
 dibuja_cruz(ant_x,clientheight-ant_y,pmxor);
 dibuja_cruz(x,clientheight-y,pmxor);
 end;
case conjunto_movimiento of
vacio: ;

mover_componente:
            if not(posicion_cursorX=posicion_cursor_antX) or
            not(posicion_cursorY=posicion_cursor_anty) and arrastrar  then begin
               dibuja_componente(puntero_comp,pmxor,clyellow);
               scroll_componente((posicion_cursorx-posicion_cursor_antX)/1000,(posicion_cursory-
posicion_cursor_anty)/1000,puntero_comp);
               dibuja_componente(puntero_comp,pmxor,clyellow);
               end;

mover_rect,datos_manuales,preparando_imprimir,
```

```
rectfica_rectangulo_superficie:
                begin
                dibuja_rect(rectangulo_manual.rectangulo,
pmxor,posicion_cursor_antx,posicion_cursor_anty);
                dibuja_rect(rectangulo_manual.rectangulo, pmxor,posicion_cursorx,posicion_cursory);
                end;

dimensionar_matriz: begin
                dibuja_rect(rectangulo_manual.rectangulo,
pmxor,posicion_cursor_antx,posicion_cursor_anty);
                dibuja_rect(rectangulo_manual.rectangulo, pmxor,posicion_cursorx,posicion_cursory);
               end;
end;
punto:=point(round(posicion_cursorx*escala),round(posicion_cursory*escala));
if matriz.activa and localizacion_matriz_punto(rectangulo_cuadricula_escalado,punto,delta,escala) then
if estilo_unidades=EUmilimetros then
FPRINCIPAL.modificar_valores(PUNTO.X,PUNTO.Y,matriz.matriz[punto.y,punto.x],
inmil_milime(posicion_cursorX-rectangulo_aux.left),
inmil_milime(posicion_cursorY-rectangulo_aux.bottom))
else
FPRINCIPAL.modificar_valores(PUNTO.X,PUNTO.Y,matriz.matriz[punto.y,punto.x],
posicion_cursorX-rectangulo_aux.left,posicion_cursorY-rectangulo_aux.bottom);
if estilo_unidades=EUmilimetros then
FPRINCIPAL.modificar_valores_posicion(inmil_milime(posicion_cursorX),
inmil_milime(posicion_cursorY),DIGITOS,' mm')
else
FPRINCIPAL.modificar_valores_posicion(posicion_cursorX,posicion_cursorY,DIGITOS,' mil');
posicion_cursor_antX:=posicion_cursorX;
posicion_cursor_antY:=posicion_cursorY;
ant_x:=x;
ant_y:=y;
end;

//desplaza todas las coordenadas del componente para que se puedan mover//

procedure Tesquema.scroll_componente(const scrollx,scrolly: REAL;
    var puntero: tipolista);
var
prueba: tipoarco;
puntero_pad: tipopad;
recta: tiporecta;
puntero_circulos: tipocirculo;
begin
recta:=puntero.estructura.recta;
     while recta <> nil do begin
       recta.x1:=recta.x1+scrollx;
       recta.y1:=recta.y1+scrolly;
       recta.x2:=recta.x2+scrollx;
       recta.y2:=recta.y2+scrolly;
       recta:=recta.prox;
       end;
offsetrectangulo(puntero.rectangulo,scrollx,scrolly);
offsetrectangulo(puntero.rectangulo_componente,scrollx,scrolly);
offsetrectangulo(puntero.rectangulo_letra,scrollx,scrolly);
puntero.inicio_letrax:=puntero.inicio_letrax+scrollx;
puntero.inicio_letray:=puntero.inicio_letray+scrolly;
prueba:=puntero.arcos;
while prueba<> nil do
   with prueba.arco do
   begin
```

```
    offsetrectangulo(rectangulo,scrollx,scrolly);
   fin_arcox:=fin_arcox+scrollx;
   fin_arcoy:=fin_arcoy+scrolly;
   inicio_arcox:=inicio_arcox+scrollx;
   inicio_arcoy:=inicio_arcoy+scrolly;
   prueba:=prueba.prox;
    end;
puntero_pad:=puntero.pad;
while puntero_pad<> nil do
    with puntero_pad^ do
    begin
    offsetrectangulo(rectangulo_interior,scrollx,scrolly);
    offsetrectangulo(rectangulo_exterior,scrollx,scrolly);
    puntero_pad:=puntero_pad.prox;
    end;
puntero_circulos:=puntero.circulos;
while puntero_circulos<> nil do
   begin
   offsetrectangulo(puntero_circulos.rectangulo,scrollx,scrolly);
   puntero_circulos:=puntero_circulos.prox;
   end;
end;

// dibuja el componente todo: arcos rectas .., utilizando la función xor entre todos los pixeles//

procedure Tesquema.dibuja_componente(var puntero: tipolista; const modo_dibujo: TPenMode; const
color: Tcolor);
begin
modo_pantalla_MMisotropic;
CANVAS.PEN.MODE:=modo_dibujo;
CANVAS.PEN.color:=color;
canvas.copymode:=cmpatcopy;
dibuja_rectas(puntero);
dibuja_arco(puntero);
dibuja_letras(puntero);
dibuja_pads(puntero);
dibuja_circulos(puntero);
end;

//Realiza el zoom//

procedure Tesquema.zoom(key: word; x,y: integer);
var
posicion: Tpoint;
tipo_x,tipo_Y: real;
 begin
 if not((x=ant_posicion.x) and  (y=ant_posicion.y)) then
 begin
 ant_posicion.x:=x;
 ant_posicion.y:=y;
 tipo_primitivo_x:=((x+horzScrollBar.position)/escala);
 tipo_primitivo_y:=((clientheight-y+pos_barra_vert)/escala);
  end;
 escala_anterior:=escala;
  if (key=VK_prior) then
  escala:=escala*2
  else
  escala:=escala/2;
  if escala>64 then
  escala:=escala_anterior
```

```
 else
 begin
 tipo_x:=((tipo_primitivo_x*escala));
 tipo_y:=((tipo_primitivo_Y*escala));
 horzScrollBar.position:=round(tipo_x-x);
 vertScrollBar.position:=round(vertScrollBar.range-clientheight-tipo_y+(clientheight-y));
 repaint;
  if ((conjunto_movimiento<>vacio) or (conjunto_pulsacion<>vacio)) {and
(scroll_dibujo_anterior<>scroll_dibujo)} then
 dibuja_cruz(ant_x,clientheight-ant_y,pmxor);
 end;
 if arrastrar then
 begin
 scroll_componente((posicion_cursorx-posicion_cursor_antX)/1000,(posicion_cursory-
posicion_cursor_anty)/1000,puntero_comp);
 dibuja_componente(puntero_comp,pmxor,clyellow);
 end;
end;

//Da la posición de la barra vertical de scroll de la ventana//

function tesquema.pos_barra_vert :integer;
begin
 result:=vertScrollBar.range-clientheight-vertScrollBar.position;
 end;

{controla las diferentes funciones de las funciones de las
teclas del teclado}

procedure Tesquema.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var
x,y: real;
posicion: TPOINT;
  begin
  getcursorpos(posicion);
  posicion:=screenTOclient(posicion);
  rejilla_cursor_escalada:=rejilla_cursor*escala;
case key of
VK_prior, VK_NEXT:
       zoom(key,posicion.x,posicion.y);
 VK_UP:   posicion_cursorY:=posicion_cursorY+rejilla_cursor;
 VK_DOWN:  posicion_cursorY:=posicion_cursorY-rejilla_cursor;
 VK_LEFT:  posicion_cursorX:=posicion_cursorX-rejilla_cursor;
 VK_RIGHT: posicion_cursorX:=posicion_cursorX+rejilla_cursor;
end;
case  key of
 VK_UP, VK_DOWN:
        begin
        Y:=posicion_cursorY * escala-pixel_mil(pos_barra_vert);
        posicion.y:=mil_pixel_ajustado(y);
        posicion.y:=clientheight-posicion.y;
        posicion:=clientTOscreen(posicion);
        mueve_raton:=false;
        Setcursorpos(posicion.x,posicion.y)
        end;
 VK_LEFT, VK_RIGHT:
       begin
        x:=(posicion_cursorX*escala)-pixel_mil(horzScrollBar.position);
        posicion.x:=mil_pixel_ajustado(x);
```

```
        posicion:=clientTOscreen(posicion);
        mueve_raton:=false;
        Setcursorpos(posicion.x,posicion.y);
        end;
end;
end;

//Evento de pintar la pantalla de la ventana//

procedure Tesquema.FormPaint(Sender: TObject);
begin
dibujar;
parar:=true;
parar1:=true;
end;



//Controla todos los eventos asociados a la pulsación de los botones de ratón//

procedure Tesquema.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
xx,yy: integer;
begin
xx:=round(posicion_cursorx);
yy:=round(posicion_cursory);
case button of
mbLeft:
     case conjunto_pulsacion of
     vacio:      exit;
     mover_componente:
                 if detectar_rect(posicion_cursorX/1000,posicion_cursory/1000,lista,puntero_comp) then
begin
                 arrastrar:=true;
                 scroll_componente((posicion_cursorx-posicion_cursor_antX)/1000,(posicion_cursory-
posicion_cursor_anty)/1000,puntero_comp);
                 dibuja_componente(puntero_comp,pmxor,clyellow);
                 conjunto_movimiento:=conjunto_pulsacion;
                 conjunto_pulsacion:=intermedio;
                 end;

     dimensionar_matriz,mover_rect,datos_manuales,
     rectfica_rectangulo_superficie:
                          begin
                          dibuja_rect(rectangulo_manual.rectangulo,
pmxor,rectangulo_manual.rectangulo.right,rectangulo_manual.rectangulo.bottom);
                         setrect(rectangulo_manual.rectangulo,xx,yy,xx,yy);
                         conjunto_movimiento:=conjunto_pulsacion;
                         conjunto_pulsacion:=intermedio;
                        end;

     intermedio:  BEGIN
               case  conjunto_movimiento of
                dimensionar_matriz: begin
                             localizar_cotornos;
                             end;

                mover_componente: begin
```

```
                    dibuja_componente(puntero_comp,pmcopy,clyellow);
                    actualiza_rect_diagrama(rectangulo_diagrama,lista);
                    arrastrar:=false;
                    repaint;
                    end;

            datos_manuales: begin
                        poner_datos_manuales;
                        repaint;
                        end;

            rectfica_rectangulo_superficie:
                        begin
                        if messagedlg ('Este sera el nuevo rectangulo de radiacion de',  MTwarning,
                        [MbYES , MbNO],0)=mryes then
                        cambiar_rectangulo_radiacion(lista,rectangulo_manual.rectangulo);
                        repaint;
                        end;
            end;
            conjunto_pulsacion:=conjunto_movimiento;
            conjunto_movimiento:=intermedio;
            dibuja_cruz(ant_x,clientheight-ant_y,pmxor);
            end;
    end;

mbRight:
      case conjunto_movimiento of
       vacio: exit;
       intermedio: begin
            case conjunto_pulsacion of
            dimensionar_matriz:
                        if messagedlg ('Esta sera la configuracion de las matrices si/no?',  MTwarning,
                        [MbYES , MbNO],0)=mryes then
                        begin
                            poner_delta;
                            dibuja_rect(rectangulo_manual.rectangulo,
pmxor,rectangulo_manual.rectangulo.right,rectangulo_manual.rectangulo.bottom);
                            quitar_cuadrado_lista(lista_rectangulo,rectangulo_manual);
                            Fprincipal.Actualiza_interfaz(EImatrices_creadas);
                            Datosfisicos1Click(self);
                        end;

             mover_rect,datos_manuales:
                        begin
                        dibuja_rect(rectangulo_manual.rectangulo,
pmxor,rectangulo_manual.rectangulo.right,rectangulo_manual.rectangulo.bottom);
                        quitar_cuadrado_lista(lista_rectangulo,rectangulo_manual);
                        end;

            imprimir,rectfica_rectangulo_superficie:
                    begin
                        dibuja_rect(rectangulo_manual.rectangulo,
pmxor,rectangulo_manual.rectangulo.right,rectangulo_manual.rectangulo.bottom);
                        quitar_cuadrado_lista(lista_rectangulo,rectangulo_manual);
                    end;
            end;
            conjunto_movimiento:=vacio;
            conjunto_pulsacion:=vacio;
            repaint;
            end;
```

```
      end;

mbMiddle: Fprincipal.valores_temperatura(lista,posicion_cursorx/1000,posicion_cursory/1000);
end;
end;


procedure Tesquema.McomponenteClick(Sender: TObject);
begin
conjunto_pulsacion:=mover_componente;
dibuja_cruz(ant_x,clientheight-ant_y,pmxor);
end;

{ esta función lo que hace es detectar el componente(rectangulo), cuya
coordenadas aux_Y y aux_x estan dentro de este rectangulo }

function Tesquema.detectar_rect( aux_x,aux_y: real; var lista,puntero:
 tipolista): boolean;
 var
prueba: Tipolista;
 begin
  result:=false;
  prueba:=lista;
  while prueba <> nil do
  BEGIN
          if puntorect_especial(prueba.rectangulo,aux_x,aux_y) then
          begin
          puntero:=prueba;
          messagebeep($fffff);
          result:=true;
          exit;
          end;
   prueba:=prueba.prox;
  end;
end;


//entra dentro de la configuración de los datos térmicos del componente//

procedure Tesquema.FormDblClick(Sender: TObject);
begin
if detectar_rect(posicion_cursorX/1000,posicion_cursory/1000,lista,puntero_comp) then begin
Fverdatos.modo_entrada
(ESmodificar_comoponente_esquema,puntero_comp,lista,@parametros_fisicos,matriz,matrizq);
Fverdatos.showModal;
end;
end;
procedure Tesquema.modo_pantalla_MMisotropic;
begin
SetmapMode(canvas.handle,MM_isotropic);
SetWindowExtEx(canvas.handle,clientwidth,clientheight,nil);
 SetViewportExtEx(canvas.handle,clientwidth,-clientheight,nil);
SetViewportOrgEx(canvas.handle,0,clientheight,nil);
end;

//cambia al modo de pantalla en milésimas de pulgada//

procedure Tesquema.modo_pantalla_MM_HIENGLISH;
begin
SetmapMode(canvas.handle,MM_HIENGLISH);
```

```
SetViewportOrgEx(canvas.handle,0,clientheight,nil);
end;

//quita componente de la lista general del programa//

procedure Tesquema.quitar_componente_lista(var puntero: Tipolista);
 var
 prueba: tipolista;
 begin
 if lista=puntero then
 lista:=puntero.prox
 else begin
 prueba:=lista;
 while prueba.prox <> puntero do
 prueba:=prueba.prox;
 prueba.prox:=puntero.prox;
 end;
 end;

//pone componente en la lista general del programa//

 procedure Tesquema.poner_componente_lista(var puntero: Tipolista);
 var
 prueba: tipolista;
 BEGIN
 puntero.prox:=lista;
 lista:=puntero;
 end;

//ventana se está ampliando//

procedure Tesquema.FormResize(Sender: TObject);
begin
setrect(rect_pantalla,0,clientheight,clientwidth,0);
end;

//cambia presicín de la rejilla a 1 mm//

procedure Tesquema.unmilimetroClick(Sender: TObject);
begin
rejilla_cursor:=milime_inmil(1);
estilo_unidades:=EUmilimetros;
end;

//dibuja gradiente de colores a grises//

procedure Tesquema.gray1Click(Sender: TObject);
begin
estilo_color:=gray;
paleta_estilo(array_colores, tonalidades,gray);
repaint;
Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.TMAX,@parametros_fisicos,matr
iz,matrizq,unidad );
end;

//cambia el número de tonalidades del gradiente de color//

procedure Tesquema.TONALIDADES1Click(Sender: TObject);
var
```

```
ClickedOK: Boolean;
NewString: string;
numero: integer;
begin
NewString:=INTtoStr(tonalidades+1);
 ClickedOK := InputQuery('Las tonalidades van de (6..255)', 'tonalidad :', NewString);
  if ClickedOK then
    try
    numero:=strTOint(NewString)-1;
    if (numero <255) and (numero >4)  then begin
    tonalidades:=numero;
    paleta_estilo(array_colores,tonalidades,estilo_color);
    repaint;
Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.tmax,@parametros_fisicos,matriz
,matrizq,unidad);
end
    else
        showmessage('El número de tonalidades es de: [6 a 255]');
    except
       showmessage('No es una cadena de números');
    end;
end;

//dibuja gradiente de colores a cool  (rosas)//

procedure Tesquema.cool1Click(Sender: TObject);
begin
estilo_color:=cool;
paleta_estilo(array_colores, tonalidades,cool);
repaint;
Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.tmax,@parametros_fisicos,matriz
,matrizq,unidad);
end;

//dibuja gradiente de colores a hot//

procedure Tesquema.hot1Click(Sender: TObject);
begin
estilo_color:=hot;
paleta_estilo(array_colores, tonalidades,hot);
repaint;
Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.tmax,@parametros_fisicos,matriz
,matrizq,unidad);
end;

//dibuja gradiente de colores a estader//

procedure Tesquema.estandar1Click(Sender: TObject);
begin
estilo_color:=estandar;
paleta_estilo(array_colores, tonalidades,estandar);
repaint;
Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.tmax,@parametros_fisicos,matriz
,matrizq,unidad);
end;
```

```
procedure Tesquema.ver_datos_matrizClick(Sender: TObject);
begin
conjunto_pulsacion:=mover_rect;
end;

//Evento que se activa cuando esta ventana es activada//

procedure Tesquema.FormActivate(Sender: TObject);
begin
Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.tmax,@parametros_fisicos,matriz
,matrizq,unidad);
end;

//cambia estilo de la malla a relleno (colores de gradiente)//

procedure Tesquema.relleno1Click(Sender: TObject);
begin
estilo_forma:=relleno;
repaint;
end;

//cambia estilo de la malla a la forma de cuadrados //

procedure Tesquema.cuadrado1Click(Sender: TObject);
begin
estilo_forma:=cuadrado;
repaint;
end;

//oculta la presencia de la malla//

procedure Tesquema.ocultarver1Click(Sender: TObject);
begin
    forma1.enabled:=not forma1.enabled;
   ver.poner_datos(conjunto_dibujo);
   ver.showmodal;
end;

//Cambia la presición del cursor a otra cantidad//

procedure Tesquema.OtarRejillacursor1Click(Sender: TObject);
var
NewString,cadena: string;
estilo_unidades_aux: testilo_unidades;
begin
estilo_unidades_aux:=estilo_unidades;
case  estilo_unidades of
EUpulgadas: NewString:=floattostrF(rejilla_cursor,ffFixed,8,digitos);
EUmilimetros: NewString:=floattostrF(inmil_milime(rejilla_cursor),ffFixed,8,digitos);
end;
   Funidad.entrada(NewString,estilo_unidades_aux);
 if Funidad.showmodal=mrok then
    try
    case estilo_unidades_aux of
    EUpulgadas: rejilla_cursor:=strTOfloat(NewString);
    EUmilimetros: rejilla_cursor:=milime_inmil(strTOfloat(NewString));
    end;
    estilo_unidades:=estilo_unidades_aux;
```

```
    if pos(',',NewString)<>0 then begin
    delete(NewString,1,pos(',',NewString));
    digitos:=length(NewString);
    end else digitos:=0;
    except;
    showmessage('No es una cadena de numeros');
    end;
end;

//cambia las unidades con las que se mueve el cursor milímetros a pulgadas//

procedure Tesquema.Otro1Click(Sender: TObject);
var
estilo_unidades_visible_aux: testilo_unidades;
NewString: string;
begin
estilo_unidades_visible_aux:=estilo_unidades_visible;
case  estilo_unidades_visible of
EUpulgadas: NewString:=floattostrF(rejilla_visible,ffFixed,8,digitos);
EUmilimetros: NewString:=floattostrF(inmil_milime(rejilla_visible),ffFixed,8,digitos);
end;
   Funidad.entrada(NewString,estilo_unidades_visible_aux);
 if Funidad.showmodal=mrok then
    try
    case estilo_unidades_visible_aux of
    EUpulgadas: rejilla_visible:=strTOfloat(NewString);
    EUmilimetros: rejilla_visible:=milime_inmil(strTOfloat(NewString));
    end;
    estilo_unidades_visible:=estilo_unidades_visible_aux;
    except;
    showmessage('No es una cadena de numeros');
    end;
end;

//Cambia el tamaño de la malla//

procedure Tesquema.Manual1Click(Sender: TObject);
begin
 if MessageDlg('Continuar con esto destruirá las matrices anteriores, continuar :  Sí/No?',
   mtWarning       , [mbYes, mbNo], 0) = mrYes then begin
   destruir_dynamic_matriz(matrizT);
   destruir_dynamic_matriz(matrizQ);
   destruir_dynamic_matriz(matrizk);
   destruir_dynamic_matriz(MATRIZTAIRE_ARRIBA);
   destruir_dynamic_matriz(MATRIZTAIRE_ABAJO);
   destruir_dynamic_matriz(MATRIZH_ARRIBA);
   destruir_dynamic_matriz(MATRIZH_ABAJO);
   destruir_dynamic_matriz(UP);
   destruir_dynamic_matriz(matriz_esp);
   matriz:=matrizT;
   Fprincipal.Actualiza_interfaz(EImatrices_destruidas);
   estilo_forma:=circulo;
   end
   else
   exit;
conjunto_pulsacion:=dimensionar_matriz;
rectangulo_manual:=poner_cuadrado_lista(lista_rectangulo,rect(0,0,0,0));
dibuja_cruz(ant_x,clientheight-ant_y,pmxor);
end;
```

```
//establece el tamaño de la retícula de la malla//

procedure Tesquema.localizar_cotornos;
  var
  rectangulo: Trect;
  x1,y1,x2,y2: real;
  ClickedOK: Boolean;
  NewString: string;
numero: integer;
  begin
  //hace que izquierda se siempre menor que derecha lo mismo para alto y bajo//
  coreccion_rectangulo( rectangulo_manual.rectangulo);
   ClickedOK := InputQuery('Los valores de la rejilla van de [20 a 100 mil]', 'rejilla:', NewString);
  if ClickedOK then
    try
    numero:=strTOint(NewString);
      if (numero >=20) and (numero <=100)  then
      poner_delta_rectangulo(numero,rectangulo_manual.rectangulo)
    else
        showmessage('La rejilla debe ir de 20 a 100 mil');
    except
       showmessage('No es una cadena de números');
    end;
    repaint;
  end;

 //quita rectangulo de la lista de rectangulos//

 procedure Tesquema.quitar_cuadrado_lista(var lista_cuadrado,nodo: tipolista_rectangulo);
 var
 prueba: tipolista_rectangulo;
 begin
 if lista_cuadrado=nodo then
 lista_cuadrado:=nodo.prox
 else begin
 prueba:=lista_cuadrado;
 while prueba.prox <> nodo do
 prueba:=prueba.prox;
 prueba.prox:=nodo.prox;
 end;
 dispose(nodo);
end;

//pone rectángulo en la lista de rectángulos//

 function Tesquema.poner_cuadrado_lista(var lista_rectangulo: tipolista_rectangulo; const rectangulo:
Trect): tipolista_rectangulo;
 var
 aux_lista: tipolista_rectangulo;
 begin
 new(aux_lista);
 aux_lista.rectangulo:=rectangulo;
 aux_lista.prox:=lista_rectangulo;
 lista_rectangulo:=aux_lista;
 result:=lista_rectangulo;
 end;

{corrige rectángulo por si izquierda es más grande que derecha o
abajo mayor que arriba}
```

```
 procedure Tesquema.coreccion_rectangulo(var rectangulo: Trect);
var
aux: integer;
 begin
 with rectangulo do
 begin
    if left> right then begin
      aux:=right;
      right:=left;
      left:=aux;
    end;
    if bottom>top then begin
      aux:=top;
      top:=bottom;
      bottom:=aux;
    end;
end;
end;

//dibuja recta recta rectángulo//

procedure Tesquema.dibuja_lista_rectangulo(var lista_rectangulo: tipolista_rectangulo);
var
prueba: tipolista_rectangulo;
begin
prueba:=lista_rectangulo;
while prueba <> nil do begin
     dibuja_rect(prueba.rectangulo, pmxor,prueba.rectangulo.right,prueba.rectangulo.bottom);
     prueba:=prueba.prox;
end;
end;

{se entra los datos térmicos del componente de la librería o
se entra  de una venta nueva para introducir manualmente los
datos si este componente no está en la librería}

function Tesquema.introducir_datos(var puntero: tipolista): boolean;
var
posicion_lista: integer;
posicion: longint;
apto,Clicked: word;
cadena: string;
puntero_aux: tipolista;
begin
puntero.atributos.activo:=true;
if not archivo_componentes.find(puntero.atributos.componente, posicion_lista) then
begin
   introducir_datos:=false;
   messagebeep($ffff);
   Clicked:=messagedlg ('El componente ( ' + puntero.atributos.COMPONENTE +
                  ' ) no está en la lista.'+ #10 +
                     'Si eliges NO el componente no entrará en la simulasión. ',  MTwarning,
                     [MbYES , MbNO,  MbAll          ],0);
   if  Clicked=mrYES then begin
      Fverdatos.modo_entrada
(ESentrar_componente_libreria,puntero,lista,@parametros_fisicos,matriz,matrizq);
      apto:=Fverdatos.showModal;
      if  apto<>mryes then
      eliminar_componente(lista,puntero);
   end
```

```
    else  begin
       eliminar_componente(lista,puntero);
       if  Clicked=mrall then
       if mryes= messagedlg ('Se ignorará todos los restantes componente, si?',  MTwarning,
                        [MbYES , MbNO],0) then
        introducir_datos:=true;
    end;
end
else
      if not existe_archivo_esquema_atributos then
      detecta_atributos_lista(puntero.atributos.componente,puntero.atributos);
end;

 //elimina un nodo de la lista de compoonentes//

 procedure Tesquema.eliminar_componente(var lista,nodo: Tipolista);
 var
 prueba: tipolista;
 begin
 //ENCONTRAR COMPONENTE DENTRO DE LA LISTA//
 if lista=nodo then
 lista:=nodo.prox
 else begin
 prueba:=lista;
 while prueba.prox <> nodo do
 prueba:=prueba.prox;
 prueba.prox:=nodo.prox
 end;
 elimina_contenido_componente(nodo);
 dispose(nodo);
end;

//elimina contenido componente antes de borrarlo de la lista de componentes//

procedure Tesquema.elimina_contenido_componente(var nodo: Tipolista);
 var
puntero: Tipolista;
puntero_letra2: tiporecorre_letras;
puntero_arco: tipoarco;
puntero_pad: tipopad;
puntero_letra: tipoletra;
puntero_recta: tiporecta;
puntero_circulo: tipocirculo;
begin
     while nodo.letras_referencia<> nil do
    begin
       puntero_letra2:=nodo.letras_referencia;
       nodo.letras_referencia:=nodo.letras_referencia.prox;
       dispose(puntero_letra2);
    end;
    while nodo.arcos<> nil do
    begin
       puntero_arco:=nodo.arcos;
       nodo.arcos:=nodo.arcos.prox;
       dispose(puntero_arco);
    end;
    while nodo.pad<> nil do
    begin
       puntero_pad:=nodo.pad;
       nodo.pad:=nodo.pad.prox;
```

```
        dispose(puntero_pad);
    end;
    while nodo.circulos<> nil do
    begin
        puntero_circulo:=nodo.circulos;
        nodo.circulos:=nodo.circulos.prox;
        dispose(puntero_circulo);
    end;
    while nodo.estructura.recta<> nil do
    begin
        puntero_recta:=nodo.estructura.recta;
        nodo.estructura.recta:=nodo.estructura.recta.prox;
        dispose(puntero_recta);
    end;
    destruir_dynamic_matriz(nodo.matriz_pines);
    destruir_dynamic_vector(nodo.vector_pines);
    destruir_dynamic_vector(nodo.Vector_pines_indx);
end;

//elimina las rectas de la placa tracks//

procedure Tesquema.elimina_rectas_placa(var lista_rectas: tiporecta);
 var
puntero_rectas: tiporecta;
begin
    while lista_rectas<> nil do
    begin
        puntero_rectas:=lista_rectas;
        lista_rectas:=lista_rectas.prox;
        dispose(puntero_rectas);
    end;
end;

{actualiza todos los datos de las matrices y ejecuta unidad de la
simulación}

procedure Tesquema.sor41Click(Sender: TObject);
var
puntero: tipolista;
begin
poner_matriz_valor(UP,25,1,matrizT.max_fila,1,matrizT.max_columna);
if parametros_fisicos.estilo_ambiente=EACONSTANTE THEN
with parametros_fisicos do begin
poner_matriz_valor(matriz_esp,temp_ambiente_arriba,1,matrizT.max_fila,1,matrizT.max_columna);
poner_matriz_valor(matrizTAIRE_ARRIBA,temp_ambiente_arriba,1,matrizT.max_fila,1,matrizT.max_c
olumna);
poner_matriz_valor(matrizTAIRE_ABAJO,temp_ambiente_abajo,1,matrizT.max_fila,1,matrizT.max_col
umna);
END;
if parametros_fisicos.estilo_H=EHCONSTANTE THEN
with parametros_fisicos do begin
poner_matriz_valor(MATRIZH_ARRIBA,h_arriba,1,MATRIZH_ARRIBA.max_fila,1,MATRIZH_ARR
IBA.max_columna);
poner_matriz_valor(MATRIZH_ABAJO,h_abajo,1,MATRIZH_ARRIBA.max_fila,1,MATRIZH_ARRI
BA.max_columna);
poner_matriz_valor(MATRIZH_borde,0,1,MATRIZH_ARRIBA.max_fila,1,MATRIZH_ARRIBA.max_
columna);
establecer_bordes(MATRIZH_borde,H_borde_derecho,H_borde_izquierdo,
h_borde_abajo,h_borde_arriba);
calcular_temp_maxmin(MATRIZH_borde,MATRIZH_borde.max_valor,MATRIZH_borde.min_valor);
```

```
calcular_temp_maxmin(MATRIZH_ABAJO,MATRIZH_ABAJO.max_valor,MATRIZH_ABAJO.min_v
alor);
calcular_temp_maxmin(MATRIZH_ARRIBA,MATRIZH_ARRIBA.max_valor,MATRIZH_ARRIBA.m
in_valor);
parametros_fisicos.h_minimo_abajo:=MATRIZH_ABAJO.min_valor;
parametros_fisicos.h_minimo_arriba:=MATRIZH_ARRIBA.min_valor;
parametros_fisicos.h_minimo_BORDE:=MATRIZH_BORDE.min_valor;
END;
if (parametros_fisicos.estilo_potencia=EPCONSTANTE) THEN
poner_matriz_valor(MATRIZQ,parametros_fisicos.Q,1,MATRIZQ.max_fila,1,MATRIZQ.max_columna
);
if parametros_fisicos.estilo_potencia=EPAUTOMATICO THEN
poner_matriz_valor(MATRIZQ,0,1,MATRIZQ.max_fila,1,MATRIZQ.max_columna);
calcular_temp_maxmin(matrizTAIRE_abajo,matrizTAIRE_ABAJO.max_valor,matrizTAIRE_ABAJO.m
in_valor);
calcular_temp_maxmin(matrizTAIRE_arriba,matrizTAIRE_arriba.max_valor,matrizTAIRE_arriba.min_
valor);
parametros_fisicos.temp_media_aire_arriba:=matrizTAIRE_arriba.valor_medio;
parametros_fisicos.temp_media_aire_abajo:=matrizTAIRE_abajo.valor_medio;
SOR3b.PSOR(matrizK,
MATRIZQ,matrizH_ARRIBA,matrizH_ABAJO,matrizH_BORDE,matrizTAIRE_ARRIBA,matrizTAIR
E_ABAJO, UP ,matriz_esp, parametros_fisicos,
rectangulo_aux, funcion_turbulento, funcion_laminar,lista_aire,lista);
calcular_temp_maxmin(UP,up.max_valor,up.min_valor);
calcular_temp_maxmin(MATRIZH_borde,MATRIZH_borde.max_valor,MATRIZH_borde.min_valor);
calcular_temp_maxmin(MATRIZH_ABAJO,MATRIZH_ABAJO.max_valor,MATRIZH_ABAJO.min_v
alor);
calcular_temp_maxmin(MATRIZH_ARRIBA,MATRIZH_ARRIBA.max_valor,MATRIZH_ARRIBA.m
in_valor);
calcular_temp_maxmin(MATRIZ_ESP,MATRIZ_ESP.max_valor,MATRIZ_ESP.min_valor);
calcular_temp_maxmin(MATRIZQ,MATRIZQ.max_valor,MATRIZQ.min_valor);
calcular_temp_maxmin(MATRIZK,MATRIZk.max_valor,MATRIZk.min_valor);
calcular_temp_maxmin(matrizTAIRE_ARRIBA,matrizTAIRE_ARRIBA.max_valor,matrizTAIRE_ARR
IBA.min_valor);
calcular_temp_maxmin(matrizTAIRE_ABAJO,matrizTAIRE_ABAJO.max_valor,matrizTAIRE_ABAJO
.min_valor);
matriz:=UP;
estilo_forma:=relleno;
capa:=ECTEMP_PLACA_ARRIBA;
repaint;
Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.tmax,@parametros_fisicos,matriz
,matrizq,unidad);
end;

//activa la matrices que se verá en pantalla//

procedure Tesquema.setcapa(valor: Testilo_capa);
begin
case valor of
ECTEMP_PLACA_ARRIBA: unidad:='placa temp. arriba (° C)';
ECPOTENCIA_COMPONENTE:  unidad:=' potencia componentes (W/ m^2)';
ECCOEF_TRNANS_ARRIBA:  unidad:= 'coef. transf. arriba (W/m² °C)';
ECCOEF_TRNANS_ABAJO:  unidad:= 'coef. transf. abajo (W/m² °C)' ;
ECCOEF_TRNANS_BORDE:  unidad:= 'coef. transf. borde (W/m² °C)' ;
ECTEMP_AMBIENTE_ABAJO:  unidad:= 'temp ambiente abajo (°C)' ;
ECTEMP_AMBIENTE_ARRIBA:  unidad:= 'temp ambiente arriba (°C)' ;
END;
try
```

```
Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.TMAX,@parametros_fisicos,matr
iz,matrizq,unidad );
EXCEPT
showmessage('Valores imposible de visualizar');
END;
end;

//cambia como activa la matriz de matrizh_borde//

procedure Tesquema.TEMPPLACAABAJO1Click(Sender: TObject);
begin
matriz:=matrizh_borde;
capa:=ECCOEF_TRNANS_BORDE;;
repaint;
end;

//cambia como activa la matriz de temperatura//

procedure Tesquema.TEMPPLACAARRIBA1Click(Sender: TObject);
begin
matriz:=UP;
capa:=ECTEMP_PLACA_ARRIBA;
repaint;
end;

//cambia como activa la matriz de MATRIZH_ARRIBA//

procedure Tesquema.COEFTRANSARRIBAH1Click(Sender: TObject);
begin
matriz:=MATRIZH_ARRIBA;
capa:=ECCOEF_TRNANS_ARRIBA;
repaint;
end;

//cambia como activa la matriz de MATRIZH_ABAJO//

procedure Tesquema.COEFTRANSABAJOH1Click(Sender: TObject);
begin
matriz:=MATRIZH_ABAJO;
capa:=ECCOEF_TRNANS_ARRIBA;
repaint;

end;

//abre la ventana de datos físicos//

procedure Tesquema.Datosfisicos1Click(Sender: TObject);
begin
Ffisicos.entrada_datos(@parametros_fisicos,digitos);
Ffisicos.showmodal;
case parametros_fisicos.estilo_h of
EHforzado: begin
        FUNCION_TURBULENTO:=Nussent_forzado_turb1;
        FUNCION_LAMINAR:=Nussent_forzado_lami1;
        end;
EHnatural: begin
        FUNCION_TURBULENTO:=Nussent_natural_turb1;
        FUNCION_LAMINAR:=Nussent_natural_lami1;
        end;
```

```
end;
end;

//cambia como activa la matriz de MATRIZQ//

procedure Tesquema.POTENCIACOMPONENTES1Click(Sender: TObject);
begin
matriz:=MATRIZQ;
capa:=ECPOTENCIA_COMPONENTE;
repaint;
end;

//activa recuadro para marcar algo//

procedure Tesquema.cuadrado2Click(Sender: TObject);
begin
conjunto_pulsacion:=mover_rect;
rectangulo_manual:=poner_cuadrado_lista(lista_rectangulo,rect(0,0,0,0));
dibuja_cruz(ant_x,clientheight-ant_y,pmxor);
end;

//construye las matrices//

procedure Tesquema.construir_matrices( fila,columna: integer);
begin
construir_dynamic_matriz(matrizT,fila,columna);
construir_dynamic_matriz(matrizTAIRE_ARRIBA,fila,columna);
construir_dynamic_matriz(matrizQ,fila,columna);
construir_dynamic_matriz(matrizk,fila,columna);
construir_dynamic_matriz(matrizTAIRE_ABAJO,fila,columna);
construir_dynamic_matriz(matrizH_ARRIBA,fila,columna);
construir_dynamic_matriz(matrizH_ABAJO,fila,columna);
construir_dynamic_matriz(up,fila,columna);
construir_dynamic_matriz(matriz_esp,fila,columna);
construir_dynamic_matriz(matrizh_borde,fila,columna);
poner_matriz_valor(matrizT,25,1,matrizT.max_fila,1,matrizT.max_columna);
poner_matriz_valor(matrizTAIRE_ARRIBA,25,1,matrizT.max_fila,1,matrizT.max_columna);
poner_matriz_valor(matrizTAIRE_ABAJO,25,1,matrizQ.max_fila,1,matrizQ.max_columna);
poner_matriz_valor(MATRIZH_ARRIBA,0,1,matrizQ.max_fila,1,matrizQ.max_columna);
poner_matriz_valor(MATRIZH_ABAJO,0,1,matrizQ.max_fila,1,matrizQ.max_columna);
poner_matriz_valor(MATRIZH_borde,0,1,matrizQ.max_fila,1,matrizQ.max_columna);
poner_matriz_valor(MATRIZk,0,1,matrizQ.max_fila,1,matrizQ.max_columna);
poner_matriz_valor(UP,25,1,matrizT.max_fila,1,matrizT.max_columna);
poner_matriz_valor(matrizQ,0,1,matrizQ.max_fila,1,matrizQ.max_columna);
calcular_temp_maxmin(MATRIZH_ARRIBA,MATRIZH_ARRIBA.max_valor,MATRIZH_ARRIBA.m
in_valor);
calcular_temp_maxmin(MATRIZH_ABAJO,MATRIZH_ABAJO.max_valor,MATRIZH_ABAJO.min_v
alor);
calcular_temp_maxmin(MATRIZQ,MATRIZQ.max_valor,MATRIZQ.min_valor);
calcular_temp_maxmin(up,up.max_valor,up.min_valor);
matriz:=matrizT;
Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.tmax,@parametros_fisicos,matriz
,matrizq,unidad);
estilo_forma:=cuadrado;
centro_rect(lista,rectangulo_aux,delta,1);
end;

//guarda el nuevo valor de reticula de malla tanto en metros como en pulgadas//
```

```
procedure Tesquema.poner_delta_rectangulo(delta_aux: real; var rectangulo: Trect);
begin
   rectangulo_aux:=rectangulo;
   delta:=delta_aux;
   parametros_fisicos.delta_in:=delta_aux;
   parametros_fisicos.delta:=inmil_milime(delta)/1000;
   parametros_fisicos.largo_malla:=inmil_milime(rectangulo.right-rectangulo.left)/1000;
   parametros_fisicos.ancho_malla:=inmil_milime(rectangulo.top-rectangulo.bottom)/1000;
end;

//construye matrices según el nuevo tamaño de la retícula de matrices//

procedure Tesquema.poner_delta;
var
columnas,filas: integer;
begin
   columnas:=round(abs(rectangulo_manual.rectangulo.right-
rectangulo_manual.rectangulo.left)/delta)+1;
   filas:=round(abs(rectangulo_manual.rectangulo.top-rectangulo_manual.rectangulo.bottom)/delta)+1;
   construir_matrices(filas,columnas);
end;


//activa la función de poner datos manualmente//

procedure Tesquema.datos_manualesClick(Sender: TObject);
begin
conjunto_pulsacion:=datos_manuales;
rectangulo_manual:=poner_cuadrado_lista(lista_rectangulo,rect(0,0,0,0));
dibuja_cruz(ant_x,clientheight-ant_y,pmxor);
end;

//pone datos manualmente en la malla activa//

procedure Tesquema.poner_datos_manuales;
var
 rectangulo: Trect;
 x1,y1,x2,y2: real;
 begin
 coreccion_rectangulo( rectangulo_manual.rectangulo);
 setrect(rectangulo,round(rectangulo_manual.rectangulo.left*escala),

round(rectangulo_manual.rectangulo.top*escala),round(rectangulo_manual.rectangulo.right*escala),roun
d(rectangulo_manual.rectangulo.bottom*escala));
 intersrect(rectangulo,rectangulo,rectangulo_cuadricula_escalado);
 localizacion_matriz_punto(rectangulo_cuadricula_escalado,rectangulo.topleft,delta,escala);
 localizacion_matriz_punto(rectangulo_cuadricula_escalado,rectangulo.bottomright,delta,escala);

punto_matriz_localizacion(rectangulo_cuadricula_escalado,rectangulo.left,rectangulo.top,x1,y1,delta,esc
ala);

punto_matriz_localizacion(rectangulo_cuadricula_escalado,rectangulo.right,rectangulo.bottom,x2,y2,delt
a,escala);

manual_datos.entrada_datos(rectangulo.topleft,rectangulo.bottomright,digitos,x1/escala,y1/escala,x2/esca
la,y2/escala,23,matriz,unidad);
 manual_datos.showmodal;
 calcular_temp_maxmin(MATRIZ,MATRIZ.max_valor,MATRIZ.min_valor);
```

```
 Fprincipal.modificar_gradiente(@array_colores, tonalidades,
matriz.max_valor,matriz.min_valor,temp_ambiente,parametros_fisicos.tmax,@parametros_fisicos,matriz
,matrizq,unidad);
 end;

//dibuja la cruz para cuando se está moviendo algo en la pantalla//

procedure Tesquema.dibuja_cruz(x,y: integer; const modo_dibujo: TPenMode);
begin
    modo_pantalla_MMisotropic;
    canvas.pen.mode:=modo_dibujo;
   canvas.pen.color:=clwhite;
   canvas.pen.width:=1;
   canvas.moveto(0,y);
   canvas.lineto(clientwidth,y);
   canvas.moveto(x,clientheight);
   canvas.lineto(x,0);
end;

//cambia como activa la malla matrizTAIRE_ABAJO//

procedure Tesquema.TEMPAMBIENTEABAJO1Click(Sender: TObject);
begin
matriz:=matrizTAIRE_ABAJO;
capa:=ECTEMP_AMBIENTE_ABAJO;
repaint;
end;

//cambia como activa la malla matrizTAIRE_ARRIBA//

procedure Tesquema.TEMPAMBIENTEARRIBA1Click(Sender: TObject);
begin
matriz:=matrizTAIRE_ARRIBA;
capa:=ECTEMP_AMBIENTE_ARRIBA;
repaint;
end;

//activa la forma de la malla a relleno (colores) y cuadrado a la vez//

procedure Tesquema.rellenocuadrado1Click(Sender: TObject);
begin
estilo_forma:=cuadrado_relleno;
repaint;
end;

//dibula el rectangulo del tamaño del componente//

procedure Tesquema.dibuja_rectangulo_componte(var rectangulo_componente: Trectangulo);
var
 grosor,colorr: integer;
rectangulo: trect;
begin
grosor:=canvas.pen.width;
colorr:=canvas.pen.color;
        with rectangulo_componente do

rectangulo:=rect(round(izquierda*1000),round(arriba*1000),round(derecha*1000),round(abajo*1000));
        dibuja_rect( rectangulo, pmcopy,rectangulo.right,rectangulo.bottom);
canvas.pen.color:=colorr;
canvas.pen.width:=grosor;
```

```
end;

//cambia presición de la rejilla de cursor a 10 mm//

procedure Tesquema.N10mm1Click(Sender: TObject);
begin
rejilla_cursor:=milime_inmil(10);
estilo_unidades:=EUmilimetros;
end;

//cambia presición de la rejilla visible a 1 mm//

procedure Tesquema.VunmilimetroClick(Sender: TObject);
begin
rejilla_visible:=milime_inmil(1);
estilo_unidades_visible:=EUmilimetros;
end;

//cambia presición de la rejilla visible a 10 mm//

procedure Tesquema.VdiesmilimetroClick(Sender: TObject);
begin
rejilla_visible:=milime_inmil(10);
estilo_unidades_visible:=EUmilimetros;
end;

//cambia presición de la rejilla del cursor a 1 mil//

procedure Tesquema.unmil1Click(Sender: TObject);
begin
rejilla_cursor:=1;
estilo_unidades:=EUpulgadas;
end;

//imprime lo que actualmente se ve en pantalla//

procedure Tesquema.Imprimir1Click(Sender: TObject);
begin
 if messagedlg ('Imprimir esta parte de la pantalla, Sí/No?',  MTwarning,
 [MbYES , MbNO],0)=mryes then begin
 repaint;
 print;
 color:=clblack;
 end;
 update;
end;

//centra el esquema en la pantalla//

procedure  Tesquema.centra_punto_esquema;
var
x,y: integer;
begin
SCROLL_DIBUJO:=desplazamiento(SCROLL_DIBUJO);
escala:=1;
x:=round((((rectangulo_diagrama.derecha-
rectangulo_diagrama.izquierda)/2+rectangulo_diagrama.izquierda)*PixelsPerInch*escala-
scroll_dibujo.x)-clientwidth/2);
y:=round(vertScrollBar.range-clientheight/2-(((rectangulo_diagrama.arriba-
rectangulo_diagrama.abajo)/2+rectangulo_diagrama.abajo)*PixelsPerInch*escala-scroll_dibujo.y));
```

```
horzScrollBar.position:=x;
vertScrollBar.position:=y;
end;

//cambia la forma de la malla a circulos//

procedure Tesquema.circulos1Click(Sender: TObject);
begin
estilo_forma:=circulo;
repaint;
end;

{posiciona las letras de los componentes procedentes del archivo de
pcb del programa Roën}

procedure   Tesquema.posicionar_letras_ULPGC(var lista: tipolista);
var
puntero: tipolista;
begin
puntero:=lista;
while puntero<> nil do begin
    with puntero^ do begin
    inicio_letrax:=rectangulo.izquierda;
    inicio_letray:=rectangulo.abajo-0.060;{altura_letra;}
    angulo_letra:=0;
    end;
puntero:=puntero.prox;
end;
end;

//cambia el tamaño del rectangulo de radiación del componente a la placa//

procedure Tesquema.cambiar_rectangulo_radiacion(var lista: tipolista; var rectangulo: Trect);
var
puntero: tipolista;
aux_rect: Trect;
x,y: real;
begin
puntero:=lista;
while puntero<>nil do begin
  aux_rect.left:=round(puntero.rectangulo_componente.izquierda*1000);
  aux_rect.right:=round(puntero.rectangulo_componente.derecha*1000);
  aux_rect.top:=round(puntero.rectangulo_componente.arriba*1000);
  aux_rect.bottom:=round(puntero.rectangulo_componente.abajo*1000);
 if  intersrect2(aux_rect,rectangulo) then  begin
    puntero.rectangulo_componente.izquierda:=rectangulo.left/1000;
    puntero.rectangulo_componente.derecha:=rectangulo.right/1000;
    puntero.rectangulo_componente.arriba:=rectangulo.top/1000;
    puntero.rectangulo_componente.abajo:=rectangulo.bottom/1000;
x:= inmil_milime(1000*(puntero.rectangulo_componente.derecha-
puntero.rectangulo_componente.izquierda))/1000;
y:=inmil_milime(1000*(puntero.rectangulo_componente.arriba-
puntero.rectangulo_componente.abajo))/1000;
if y>x then begin
  puntero.atributos.largo:=y;
  puntero.atributos.ancho:=x;
end
else begin
    puntero.atributos.largo:=x;
    puntero.atributos.ancho:=y;
```

```
    end;
end;
puntero:=puntero.prox;
end;
end;

//activa la función de cambiar rectángulo de radiacción//

procedure Tesquema.Ponerrectangulosuperficie1Click(Sender: TObject);
begin
conjunto_pulsacion:=rectfica_rectangulo_superficie;
rectangulo_manual:=poner_cuadrado_lista(lista_rectangulo,rect(0,0,0,0));
dibuja_cruz(ant_x,clientheight-ant_y,pmxor);
end;

//actualiza el tamaño del cuadrado que delimeta a todos los componentes//

procedure Tesquema.actualiza_rect_diagrama(var rectangulo_variable: Trectangulo;var lista: tipolista);
var
puntero: tipolista;
begin
puntero:=lista;
rectangulo_variable:=lista.rectangulo;
  while puntero<> nil do begin
  sumar_rectangulo(rectangulo_variable,puntero.rectangulo);
  puntero:=puntero.prox;
  end;
end;

//desctiva la visualización de la malla//

procedure Tesquema.nada1Click(Sender: TObject);
begin
estilo_forma:=nada;
repaint;
end;

//cambia unidades//

procedure Tesquema.Cambiarunidad1Click(Sender: TObject);
begin
if estilo_unidades=EUmilimetros then
estilo_unidades:=EUpulgadas
else
estilo_unidades:=EUmilimetros;
digitos:=3;
end;

//activa la función de cambiar datos manualmente de la malla//

procedure Tesquema.Ponerdatosmanualmente1Click(Sender: TObject);
begin
conjunto_pulsacion:=datos_manuales;
rectangulo_manual:=poner_cuadrado_lista(lista_rectangulo,rect(0,0,0,0));
dibuja_cruz(ant_x,clientheight-ant_y,pmxor);
end;

end.
```

unit H2;

```
interface
uses
 Dialogs,tipo1;

 //Cálculo los la temparatura de los nodos laterales y de las esquinas//

procedure PH(var MATRIZQ,matrizH_ARRIBA,matrizH_ABAJO,matrizH_BORDE,
matrizTAIRE_ARRIBA,matrizTAIRE_ABAJO,UP : TMATRIZ;var  fisicos: Tparametros_fisicos; VAR
w,CAMBIO,temp_med: DOUBLE);

implementation
procedure PH(var MATRIZQ,matrizH_ARRIBA,matrizH_ABAJO,matrizH_BORDE,
matrizTAIRE_ARRIBA,matrizTAIRE_ABAJO,UP : TMATRIZ;var  fisicos: Tparametros_fisicos; VAR
w,CAMBIO,temp_med: DOUBLE);

VAR
F,C: INTEGER;
RESID: REAL;
BEGIN
WITH up DO BEGIN
for F:=2 to max_fila-1 do  begin
```

```
                                  {
                                   _________
                                 4|         |
                                 3|         |
Se analiza este lado------------>|         |
                                 1|_________|
                                   1 2 3  4 5  }
```

```
RESID:=FISICOS.RADIO1 * (MATRIZ[F-1,1] + MATRIZ[F+1,1] +2*MATRIZ[F,2] -
4*MATRIZ[F,1])+
FISICOS.RADIO2 * (MATRIZQ.MATRIZ[F,1]-MATRIZ[F,1]*matrizH_abajo.matriz[F,1]-
MATRIZ[F,1]*matrizH_ARRIBA.MATRIZ[F,1]+
matrizH_arriba.MATRIZ[F,1] * matrizTAIRE_arriba.MATRIZ[F,1]+matrizH_abajo.MATRIZ[F,1] *
matrizTAIRE_abajo.MATRIZ[F,1])+
FISICOS.RADIO3 * (2*matrizH_borde.MATRIZ[F,1]*matrizTAIRE_arriba.MATRIZ[F,1]-
2*MATRIZ[F,1] * matrizH_borde.MATRIZ[F,1]);
IF  CAMBIO < ABS(RESID) THEN
CAMBIO:=ABS(RESID);
MATRIZ[F,1]:=MATRIZ[F,1] +W *  RESID;
    {
     _________
    4|         |
    3|         | <-----Se calcula en este lado
    2|         |
    1|_________|
      1 2 3 4 5  }
```

```
RESID:=FISICOS.RADIO1 * (MATRIZ[F-1,max_columna] + MATRIZ[F+1,max_columna] +
2*MATRIZ[F,max_columna-1] - 4*MATRIZ[F,max_columna])+
FISICOS.RADIO2 * (MATRIZQ.MATRIZ[F,max_columna]-
MATRIZ[F,max_columna]*matrizH_abajo.MATRIZ[F,max_columna]-
MATRIZ[F,max_columna]*matrizH_ARRIBA.MATRIZ[F,max_columna]+
 matrizH_arriba.MATRIZ[F,max_columna] * matrizTAIRE_arriba.MATRIZ[F,max_columna]+
matrizH_abajo.MATRIZ[F,max_columna] * matrizTAIRE_abajo.MATRIZ[F,max_columna])+
```

```
FISICOS.RADIO3 *
(2*matrizH_borde.MATRIZ[F,max_columna]*matrizTAIRE_arriba.MATRIZ[F,max_columna]-
2*MATRIZ[F,max_columna] * matrizH_borde.MATRIZ[F,max_columna]);
IF  CAMBIO < ABS(RESID) THEN
CAMBIO:=ABS(RESID);
MATRIZ[F,max_columna]:=MATRIZ[F,max_columna] +W *  RESID;
end;
for C:=2 to max_columna-1 do  begin
    {



     _________
   4|         |
   3|         |
   2|         |
   1|_________|
      ^
      |__se calcula en este lugar}

RESID:=FISICOS.RADIO1 * (MATRIZ[1,C-1] + MATRIZ[1,C+1] +
2*MATRIZ[2,C] - 4*MATRIZ[1,C])+
FISICOS.RADIO2 * (MATRIZQ.MATRIZ[1,C]-MATRIZ[1,C]*matrizH_abajo.MATRIZ[1,C]-
MATRIZ[1,C]*matrizH_ARRIBA.MATRIZ[1,C]+
 matrizH_arriba.MATRIZ[1,C] * matrizTAIRE_arriba.MATRIZ[1,C]+ matrizH_abajo.MATRIZ[1,C] *
matrizTAIRE_abajo.MATRIZ[1,C])+
FISICOS.RADIO3 * (2*matrizH_borde.MATRIZ[1,C]*matrizTAIRE_arriba.MATRIZ[1,C]-
2*MATRIZ[1,C]* matrizH_borde.MATRIZ[1,C]);
IF  CAMBIO < ABS(RESID) THEN
CAMBIO:=ABS(RESID);
MATRIZ[1,C]:=MATRIZ[1,C] +W *  RESID;

      { _ _se calcula en este lugar
    ___|___
   4|       |
   3|       |
   2|       |
   1|_______|
           }
RESID:=FISICOS.RADIO1 * (MATRIZ[max_fila,C+1] + MATRIZ[max_fila,C-1] +
2*MATRIZ[max_fila-1,C] - 4*MATRIZ[max_fila,C])+
FISICOS.RADIO2 * (MATRIZQ.MATRIZ[max_fila,C]-
MATRIZ[max_fila,C]*matrizH_abajo.MATRIZ[max_fila,C]-
MATRIZ[max_fila,C]*matrizH_ARRIBA.MATRIZ[max_fila,C]+
matrizH_arriba.MATRIZ[max_fila,C] * matrizTAIRE_arriba.MATRIZ[max_fila,C]+
matrizH_abajo.MATRIZ[max_fila,C] * matrizTAIRE_abajo.MATRIZ[max_fila,C])+
FISICOS.RADIO3 *
(2*matrizH_borde.MATRIZ[max_fila,C]*matrizTAIRE_arriba.MATRIZ[max_fila,C]-
2*MATRIZ[max_fila,C] * matrizH_borde.MATRIZ[max_fila,C]);
IF  CAMBIO < ABS(RESID) THEN
CAMBIO:=ABS(RESID);
MATRIZ[max_fila,C]:=MATRIZ[max_fila,C] +W *  RESID;
END;
                              {

                               _________
                              4|         |
                              3|         |
                              2|         |
    se calcula este punto---->1|_________|
                                1 2 3  4 5  }
```

```
RESID:=FISICOS.RADIO1 * (2*MATRIZ[2,1] + 2*MATRIZ[1,2]  - 4*MATRIZ[1,1])+
FISICOS.RADIO2 * (MATRIZQ.MATRIZ[1,1]-MATRIZ[1,1]*matrizH_abajo.matriz[1,1]-
MATRIZ[1,1]*matrizH_ARRIBA.MATRIZ[1,1]+
matrizH_arriba.MATRIZ[1,1] * matrizTAIRE_arriba.MATRIZ[1,1]+matrizH_abajo.MATRIZ[1,1] *
matrizTAIRE_abajo.MATRIZ[1,1])+
FISICOS.RADIO3 * (4*matrizH_borde.MATRIZ[1,1]*matrizTAIRE_arriba.MATRIZ[1,1]-
4*MATRIZ[1,1] * matrizH_borde.MATRIZ[1,1]);
IF  CAMBIO < ABS(RESID) THEN
CAMBIO:=ABS(RESID);
MATRIZ[1,1]:=MATRIZ[1,1] +W *  RESID;
                {
                 _________
                4|         |
                3|         |
                2|         |
                1|_________|<-----se calculo este
1 2 3  4 5  }

RESID:=FISICOS.RADIO1 * (2*MATRIZ[1,max_columna-1] + 2*MATRIZ[2,max_columna] +
 - 4*MATRIZ[1,max_columna])+
FISICOS.RADIO2 * (MATRIZQ.MATRIZ[1,max_columna]-
MATRIZ[1,max_columna]*matrizH_abajo.MATRIZ[1,max_columna]-
MATRIZ[1,max_columna]*matrizH_ARRIBA.MATRIZ[1,max_columna]+
 matrizH_arriba.MATRIZ[1,max_columna] * matrizTAIRE_arriba.MATRIZ[1,max_columna]+
matrizH_abajo.MATRIZ[1,max_columna] * matrizTAIRE_abajo.MATRIZ[1,max_columna])+
FISICOS.RADIO3 *
(4*matrizH_borde.MATRIZ[1,max_columna]*matrizTAIRE_arriba.MATRIZ[F,max_columna]-
4*MATRIZ[1,max_columna] * matrizH_borde.MATRIZ[1,max_columna]);
IF  CAMBIO < ABS(RESID) THEN
CAMBIO:=ABS(RESID);
MATRIZ[1,max_columna]:=MATRIZ[1,max_columna] +W *  RESID;
                                {
                                 _________
        se calculo este punto---->4|         |
                                3|         |
                                2|         |
                                1|_________|
                                  1 2 3  4 5  }

RESID:=FISICOS.RADIO1 * (2*MATRIZ[max_fila,2]+ 2*MATRIZ[max_fila-1,1] +
 - 4*MATRIZ[max_fila,1])+
FISICOS.RADIO2 * (MATRIZQ.MATRIZ[max_fila,1]-
MATRIZ[max_fila,1]*matrizH_abajo.MATRIZ[max_fila,1]-
MATRIZ[max_fila,1]*matrizH_ARRIBA.MATRIZ[max_fila,1]+
matrizH_arriba.MATRIZ[max_fila,1] * matrizTAIRE_arriba.MATRIZ[max_fila,1]+
matrizH_abajo.MATRIZ[max_fila,1] * matrizTAIRE_abajo.MATRIZ[max_fila,1])+
FISICOS.RADIO3 *
(4*matrizH_borde.MATRIZ[max_fila,1]*matrizTAIRE_arriba.MATRIZ[max_fila,1]-
4*MATRIZ[max_fila,1] * matrizH_borde.MATRIZ[max_fila,1]);
IF  CAMBIO < ABS(RESID) THEN
CAMBIO:=ABS(RESID);
MATRIZ[max_fila,1]:=MATRIZ[max_fila,1] +W *  RESID;
                {
                 _________
                4|         |<---se calculo este punto
                3|         |
                2|         |
                1|_________|
                  1 2 3  4 5  }
```

```
RESID:=FISICOS.RADIO1 * (2*MATRIZ[max_fila,max_columna-1] +2* MATRIZ[max_fila-
1,max_columna]+
- 4*MATRIZ[max_fila,max_columna])+
FISICOS.RADIO2 * (MATRIZQ.MATRIZ[max_fila,max_columna]-
MATRIZ[max_fila,max_columna]*matrizH_abajo.MATRIZ[max_fila,max_columna]-
MATRIZ[max_fila,max_columna]*matrizH_ARRIBA.MATRIZ[max_fila,max_columna]+
matrizH_arriba.MATRIZ[max_fila,max_columna] *
matrizTAIRE_arriba.MATRIZ[max_fila,max_columna]+
matrizH_abajo.MATRIZ[max_fila,max_columna] *
matrizTAIRE_abajo.MATRIZ[max_fila,max_columna])+
FISICOS.RADIO3 *
(4*matrizH_borde.MATRIZ[max_fila,max_columna]*matrizTAIRE_arriba.MATRIZ[max_fila,max_col
umna]- 4*MATRIZ[max_fila,max_columna] * matrizH_borde.MATRIZ[max_fila,max_columna]);
IF  CAMBIO < ABS(RESID) THEN
CAMBIO:=ABS(RESID);
MATRIZ[max_fila,max_columna]:=MATRIZ[max_fila,max_columna] +W *  RESID;
 END;
 END;
end.
```

```
unit LU;

interface
uses
Dialogs,tipo1;

//Obtine el resulsato del sistema de ecuaciones//

procedure ludcmp(var a: Tmatriz; var n: integer; var indx: Tvector;
var d: real);

//Construye las matricies L U //

procedure lubksb(var a: Tmatriz; var n: integer; var indx: Tvector;
var b: Tvector);


implementation

//Obtine el resulsato del sistema de ecuaciones//

procedure ludcmp(var a: Tmatriz;var  n: integer; var indx: Tvector;
var d: real);
const
tiny= 1.0e-20;
var
k,j,imax,i: integer;
sum,dum,big: real;
vv: Tvector;
begin
construir_dynamic_vector(vv,n);
d:=1.0;
for i:=1 to n do begin
   big:=0.0;
   for j:=1 to n do
      if abs(a.matriz[i,j])> big then big := abs(a.matriz[i,j]);
   if big =0.0 then begin
      showmessage('matriz singular');
      exit;
   end;
   vv.vector^[i] :=1.0/big;
end;
for j:= 1 to n do begin
   for i:= 1 to j-1 do begin
      sum:= a.matriz[i,j];
      for k:=1 to i-1 do
         sum:= sum-a.matriz[i,k]*a.matriz[k,j];
      a.matriz[i,j]:=sum;
   end;
   big:=0.0;
   for i:=j to n do begin
      sum:=a.matriz[i,j];
      for k:=1 to j-1 do
         sum:=sum-a.matriz[i,k]*a.matriz[k,j];
      a.matriz[i,j]:=sum;
      dum:=vv.vector^[i]*abs(sum);
      if dum >= big then begin
        big:=dum;
        imax:=i;
      end;
```

```
    end;
    if j<> imax then begin
      for k:= 1 to n do begin
        dum:=a.matriz[imax,k];
        a.matriz[imax,k]:=a.matriz[j,k];
        a.matriz[j,k]:=dum;
      end;
      d:=-d;
      vv.vector^[imax]:=vv.vector^[j];
    end;
    indx.vector[j]:=imax;
    if a.matriz[j,j]=0.0 then a.matriz[j,j]:=tiny;
    if j<> n then begin
      dum:=1.0/a.matriz[j,j];
      for i:=j+1 to n do
        a.matriz[i,j]:=a.matriz[i,j]*dum;
    end;
end;
    destruir_dynamic_vector(vv);
end;

procedure lubksb(var a: Tmatriz; var n: integer; var indx: Tvector;
var b: Tvector);
var
  j,ip,ii,i: integer;
  sum: real;
begin
    ii:=0;
    for i:=1 to n do begin
      ip:=round(indx.vector[i]);
      sum:=b.vector[ip];
      b.vector[ip]:=b.vector[i];
      if ii <> 0 then
        for j:=ii to i-1 do
          sum:=sum-a.matriz[i,j]*b.vector[j]
      else if sum <> 0.0 then
        ii:=i;
      b.vector[i]:=sum;
    end;
    for i:=n downto 1 do begin
      sum:=b.vector[i];
      for j:=i+1 to n do
        sum:=sum-a.matriz[i,j]*b.vector[j];
      b.vector[i]:=sum/a.matriz[i,i];
    end;
end;

end.
```

```
unit manual;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls,tipo1,tipo_eventos, Buttons;

type
  Tmanual_datos = class(TForm)
    GroupBox1: TGroupBox;
    Einicio_columna: TEdit;
    Efinal_columna: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    GroupBox2: TGroupBox;
    Evalor: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    kkkk: TLabel;
    Earriba: TEdit;
    Eizquierda: TEdit;
    BitBtn1: TBitBtn;
    Label6: TLabel;
    Einicio_fila: TEdit;
    Label7: TLabel;
    Efinal_fila: TEdit;
    Label8: TLabel;
    Eabajo: TEdit;
    Label9: TLabel;
    Ederecha: TEdit;
    procedure BitBtn1Click(Sender: TObject);
  private
  matriz: Tmatriz;
    { Private declarations }
  public
procedure entrada_datos(const punto1,punto2: tpoint; const digitos: integer; const x1,y1,x2,y2,potencia:
real;
 var aux_matriz: Tmatriz; var unidad: string);
    { Public declarations }
  end;

var
  manual_datos: Tmanual_datos;

implementation

{$R *.DFM}

//Nos da las coordenadas de la malla donde se pondrá el valor/es //

procedure Tmanual_datos.entrada_datos(const punto1,punto2: tpoint; const digitos: integer; const
x1,y1,x2,y2,potencia: real;
 var aux_matriz: Tmatriz; var unidad: string);
 var
i: integer;
begin
matriz:=aux_matriz;
Einicio_columna.text:=intTOstr(punto1.x);
Einicio_fila.text:=intTOstr(punto2.y);
```

```
Efinal_columna.text:=intTOstr(punto2.x);
Efinal_fila.text:=intTOstr(punto1.y);
Eizquierda.text:=floattostrF(x1,ffFixed,8,digitos);
Earriba.text:=floattostrF(y1,ffFixed,8,digitos);
Ederecha.text:=floattostrF(x2,ffFixed,8,digitos);
Eabajo.text:=floattostrF(y2,ffFixed,8,digitos);
caption:=unidad;
for i:=0 to Componentcount-1 do
begin
    if components[i] is Tedit then
    begin
     if not ((components[i] as Tedit).tag=0) then
    (components[i] as Tedit).text:= poner_unidad(array_unidades[(components[i] as Tedit).tag],
       (components[i] as Tedit).text);
    end;
end;
 end;

//Poner el valor para añadirlo a la malla//

procedure Tmanual_datos.BitBtn1Click(Sender: TObject);
begin
try
poner_matriz_valor(matriz,strtofloat(Evalor.text),strtoint(Einicio_fila.text),strTOint(Efinal_fila.text),strT
Oint(Einicio_columna.text),strTOint(Efinal_columna.text));
calcular_temp_maxmin(matriz,matriz.max_valor,matriz.min_valor);
close;
except
    showmessage('Hay algún dato mal escrito');
end;
end;
end.
```

unit ocultar;

```
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls,tipo1, Buttons;

type
  TVer = class(TForm)
    cuadro0: TRadioGroup;
    cuadro1: TRadioGroup;
    cuadro2: TRadioGroup;
    cuadro3: TRadioGroup;
    cuadro4: TRadioGroup;
    cuadro5: TRadioGroup;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    cuadro8: TRadioGroup;
    cuadro6: TRadioGroup;
    cuadro7: TRadioGroup;
    cuadro9: TRadioGroup;
    cuadro10: TRadioGroup;
    cuadro11: TRadioGroup;
    procedure FormActivate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
  private
  conjunto_dibujo: ^Tconjunto_dibujos;
    { Private declarations }
  public
procedure poner_datos(var aux_conjunto_dibujo: Tconjunto_dibujos);

    { Public declarations }
  end;

var
  Ver: TVer;

implementation
uses
padre;
{$R *.DFM}

procedure Tver.poner_datos(var aux_conjunto_dibujo: Tconjunto_dibujos);
begin
conjunto_dibujo:= @aux_conjunto_dibujo;
end;

{Se actualiza en los recuadro de visualización el estado de visuallización
de cada elemento del dibujo}

procedure TVer.FormActivate(Sender: TObject);
var
n: Testilo_dibujos;
begin
{(0) CDrectas, (1)CDrectangulo,CDarcos,CDcirculos,CDletras,CDcolores...);}
for n:=CDrectas to CDsimbolo_desactivado do
  with (findcomponent('cuadro'+intTostr(integer(n))) as Tradiogroup) do
  if n in conjunto_dibujo^ then
```

```
    itemindex:=Items.IndexOf('no ocultar')
  else
    itemindex:=Items.IndexOf('ocultar');
end;

//Se actualiza la nueva situación de los componentes de dibujo//

procedure TVer.BitBtn1Click(Sender: TObject);
var
n: Testilo_dibujos;
i:integer;
begin
for n:=CDrectas to CDsimbolo_desactivado do
  with (findcomponent('cuadro'+intTostr(integer(n))) as Tradiogroup) do
    if items[itemindex]='ocultar' then
    conjunto_dibujo^:=conjunto_dibujo^-[n]
  else
    conjunto_dibujo^:=conjunto_dibujo^+[n];
For i := 0 To Fprincipal.MDIChildCount-1 Do
   Fprincipal.MDIChildren[i].repaint;
close;
end;

procedure TVer.BitBtn2Click(Sender: TObject);
begin
close;
end;

end.
```

unit opciones;

```
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, Buttons,tipo1;
type
  TFopciones = class(TForm)
    salvar_tipo: TRadioGroup;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
  forma_copia:  Testilo_modificar_componente;
    { Private declarations }
  public
  procedure modo_forma_copia(estilo: Testilo_modificar_componente);
    { Public declarations }
  end;

var
  Fopciones: TFopciones;

implementation

{$R *.DFM}
{Se da a la opción si se quiere que se guarde los componentes
del mismo tipo a todos los componentes}

procedure TFopciones.BitBtn2Click(Sender: TObject);
begin
case forma_copia of
ESmodificar_comoponente_esquema:
 modalresult:= messagedlg (' Esto hara modificar los datos de los componentes' +
          #10+ '¿Realizar los cambios?' ,  MTwarning,
                    [MbYES , MbNO],0);
ESentrar_componente_libreria,ESmodificar_componente_libreria:
 modalresult:= messagedlg (' Esto hara modificar los datos de la libreria en atributos.txt' +
          #10+ '¿Realizar los cambios?' ,  MTwarning,
                    [MbYES , MbNO],0);
end;
end;

  procedure TFopciones.modo_forma_copia(estilo: Testilo_modificar_componente);
  begin
  forma_copia:=estilo;
  if forma_copia=ESmodificar_comoponente_esquema then
  salvar_tipo.visible:=true
  else
  salvar_tipo.visible:=false;
  end;

procedure TFopciones.BitBtn1Click(Sender: TObject);
begin
 modalresult:=mrcancel
end;
end.
```

```
unit PADRE;

interface

uses
  Windows, Messages, SysUtils, Classes,  Controls, Forms, Dialogs,
  Menus, ExtCtrls, StdCtrls, Buttons,tipo1,Fesquema, ComCtrls, frstatus,
  MRUList,Usacar_datos_atributos,Usacar_datos_protel,Usacar_datos_ULPGC,Verdatos;
const
ancho_gradiente=25;
type
  TFprincipal = class(TForm)
    Panel3: TPanel;
    MenuPrincipal: TMainMenu;
    Archivo1: TMenuItem;
    MDAbrir: TMenuItem;
    Nuevo1: TMenuItem;
    Salvar1: TMenuItem;
    N1: TMenuItem;
    Salir1: TMenuItem;
    Libreria: TMenuItem;
    Componente: TMenuItem;
    Ventana1: TMenuItem;
    Cascada1: TMenuItem;
    vertical: TMenuItem;
    horizontal: TMenuItem;
    N2: TMenuItem;
    Minimizartodas1: TMenuItem;
    Abrirtodas1: TMenuItem;
    Cerrartodas1: TMenuItem;
    ODabrir: TOpenDialog;
    Ver2: TMenuItem;
    izquierda1: TMenuItem;
    izquierda2: TMenuItem;
    derecha1: TMenuItem;
    ocultar1: TMenuItem;
    Panel1: TPanel;
    gradiante: TPaintBox;
    MRUFileList: TMRUFileList;
    lmax: TLabel;
    lmin: TLabel;
    lmed: TLabel;
    lcellx: TLabel;
    lcelly: TLabel;
    LVAL: TLabel;
    lLargo: TLabel;
    lANCHO: TLabel;
    LTAMB: TLabel;
    panel2: TPanel;
    Lxposicion: TLabel;
    lyposicion: TLabel;
    lunidad: TLabel;
    Label2: TLabel;
    lTj: TLabel;
    lTc: TLabel;
    lTp: TLabel;
    SDsalvar: TSaveDialog;
    datos1: TMenuItem;
    Izquierda4: TMenuItem;
    derecha3: TMenuItem;
```

```
  ocultarver1: TMenuItem;
  SpeedButton4: TSpeedButton;
  SpeedButton1: TSpeedButton;
  SPimprimir: TSpeedButton;
  SpeedButton6: TSpeedButton;
  SpeedButton7: TSpeedButton;
  StatusBar1: TStatusBar;
  zoom: TSpeedButton;
  Lfilas: TLabel;
  Lcolumnas: TLabel;
  LFxC: TLabel;
  Lreticula: TLabel;
  SpeedButton9: TSpeedButton;
  SPabrir: TSpeedButton;
  SpeedButton2: TSpeedButton;
  procedure FormCreate(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure Cascada1Click(Sender: TObject);
  procedure verticalClick(Sender: TObject);
  procedure horizontalClick(Sender: TObject);
  procedure Cerrartodas1Click(Sender: TObject);
  procedure Salir1Click(Sender: TObject);
  procedure MDAbrirClick(Sender: TObject);
  procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  procedure MRUFileListMRUItemClick(Sender: TObject; AFilename: string);
  procedure gradientePaint(Sender: TObject);
  procedure Panel1Resize(Sender: TObject);
  procedure gradianteMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
  procedure izquierda2Click(Sender: TObject);
  procedure derecha1Click(Sender: TObject);
  procedure ocultar1Click(Sender: TObject);
  procedure cargar_lista_aire;
  procedure FormDestroy(Sender: TObject);
   procedure cambio_decimales;
  procedure gradianteDblClick(Sender: TObject);
  procedure Salvar1Click(Sender: TObject);
  procedure Izquierda3Click(Sender: TObject);
  procedure derecha2Click(Sender: TObject);
  procedure ocultar2Click(Sender: TObject);
  procedure Izquierda4Click(Sender: TObject);
  procedure derecha3Click(Sender: TObject);
  procedure ocultarver1Click(Sender: TObject);
  procedure ComponenteClick(Sender: TObject);
  procedure SpeedButton7Click(Sender: TObject);
  procedure SpeedButton4Click(Sender: TObject);
  procedure SpeedButton3Click(Sender: TObject);
  procedure SpeedButton1Click(Sender: TObject);
  procedure zoomClick(Sender: TObject);
  procedure SpeedButton6Click(Sender: TObject);
  procedure SPimprimirClick(Sender: TObject);
  procedure SpeedButton9Click(Sender: TObject);
  procedure SpeedButton2Click(Sender: TObject);
  procedure SPabrirClick(Sender: TObject);

 private
 valor: boolean;
 matriz,matrizQ: Tmatriz;
Archivo_aire: text;
 DECIMAL1,
 DECIMAL2:  integer;
```

```
  fisicos: Tpuntero_fisicos;
  dibujo_rect: Trect;
  estilo_color: Testilo_colores;
  array_colores: Tpuntero_colores;
  tonalidades,r,ancho_rect: integer;
  temp_max,temp_min,alto_rect,temp_ambiente,rango_temp,valor_dato,TIEMPO: real;
   procedure dibujar_letras;
   procedure WMNCMOUSEMOVE(var message: TWMNCMOUSEMOVE); message
  WM_NCMOUSEMOVE;
   procedure WMMOUSEMOVE(var message: tWMMOUSEMOVE); message WM_MOUSEMOVE;
   procedure WMNCHitTest(var message: TWMNcHitTest); message WM_NCHitTest;
   { Private declarations }
  public
  procedure valores_temperatura(var lista: tipolista; x,y: real);
  procedure  modificar_gradiente(colores: tpuntero_colores; tonalidades_aux: integer;
max,min,temp_ambiente_aux,TIEMPO: real;
   fisicos_aux:  Tpuntero_fisicos; var matriz_aux,matriz_aux_q: Tmatriz; var unidad: string);

procedure modificar_valores (cellx,celly: integer;  valor_dato,X,Y: real);
procedure modificar_valores_posicion(x,y: real; digitos: integer;const  posicion_unidad: openstring);
 procedure Actualiza_interfaz(const interfaz: Testilo_interfaz);
 procedure desacativa_comun_menu(valor: boolean);

{ Public declarations }
 end;
var
 Fprincipal: TFprincipal;
 archivo_cabecera:  Tstringlist;
 archivo_atributos:  Tstringlist; //es la librería de componentes//
 lista_atributos: tipoatributos;
 archivo_componentes: Tstringlist;
 archivo_letras: file_letras;
 lista_letras: tipoletra;
 lista_aire: tipoaire;

implementation

uses  DATOS_FISICOS, portada;


{$R *.DFM}

//cierra tosdas las ventanas que están abiertas//

procedure TFprincipal.Cerrartodas1Click(Sender: TObject);
Var
  N: Integer;
begin
    { Cerrar todas las ventanas }
   For N := 0 To MDIChildCount-1 Do
      MDIChildren[N].Close;
end;

procedure TFprincipal.Salir1Click(Sender: TObject);
begin
   { Antes de salir cerrar las ventanas }
   Cerrartodas1Click(Self);
   close;
end;
```

```
//Crear una ficha con una configuración nueva de componentes//

procedure TFprincipal.MDAbrirClick(Sender: TObject);
begin
if Fprincipal.ODabrir.execute then
      if  extractfileext(ODabrir.filename) = '.pcb' then
      begin
        Fprincipal.ODabrir.historylist.add(ODabrir.filename);
        Fprincipal.SDsalvar.historylist.add(ODabrir.filename);
         TRY
        application.createform(Tesquema,esquema);
       esquema.caption:=ODabrir.filename;
        esquema.show;
        tile;
end;


procedure TFprincipal.verticalClick(Sender: TObject);
begin
tilemode:=Tbvertical;
tile;
end;

procedure TFprincipal.Cascada1Click(Sender: TObject);
begin
cascade;
end;

//Abre los distintos archivos utilizados librería la cual la pasa
a una lista. Otros archivos que se abrean són letras, componentes,
datos del aire//

procedure TFprincipal.FormCreate(Sender: TObject);
begin
lista_aire:=nil;
cargar_lista_aire;
DECIMAL1:=1;
panel1.visible:=false;
archivo_cabecera:=Tstringlist.create;
archivo_componentes:=Tstringlist.create;
archivo_atributos:=Tstringlist.create;
    Try
      archivo_atributos.loadfromfile(path_atributos);
     hacer_lista_atributos(lista_atributos,archivo_atributos);
    Except On Exception Do { si también falla }
     Begin
        ShowMessage('No es posible abrir el archivo atributos.txt');
        halt;
   end;
  END;
 TRY
   archivo_cabecera.loadfromfile(path_cabecera);
      Except On Exception Do { si falla }
     Begin
         ShowMessage('No es posible abrir el archivo cabecera.txt');
         halt;
     end;
  END;
  TRY
   archivo_componentes.loadfromfile(path_componente);
```

```
      Except On Exception Do { Si falla }
     Try
       Rewrite(archivo_letras); { Intentar la creación }
       archivo_componentes.sort;
   Except on Exception do
   begin
   Showmessage('No es posible abrir el archivo componet.txt');
   halt;
   end;
   end;
   END;
   AssignFile(archivo_letras, path_letras);
    Try
       Reset(archivo_letras); { Intentar la apertura }
       lista_letras:=hacer_lista_letras;
       closefile(archivo_letras);
     Except On Exception Do { si también falla }
      Begin
         ShowMessage('No es posible abrir el archivo letras.dat');
    end;
  END;
 Actualiza_interfaz(EIinicio_entrada);
 END;

//Cirra todas las fichas y borra las listas creadas, porque se va
a cerrar el programa//

procedure TFprincipal.FormClose(Sender: TObject; var Action: TCloseAction);
var
puntero_letra: tipoletra;
begin
while lista_letras <> nil do
begin
    puntero_letra:=lista_letras;
    lista_letras:=lista_letras.prox;
    dispose(puntero_letra);
end;
ARCHIVO_COMPONENTES.FREE;
ARCHIVO_cabecera.FREE;
archivo_atributos.free;
end;

procedure TFprincipal.horizontalClick(Sender: TObject);
begin
  Tilemode:= TBhorizontal;
  tile;
end;

procedure TFprincipal.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
begin
    archivo_componentes.sort;
    archivo_componentes.saveTofile(path_componente);
    archivo_atributos.saveTofile(path_atributos);
end;

//Hace la lista aire del archivo aire//

procedure TFprincipal.cargar_lista_aire;
```

```
var
cadena: string[125];
nodo: tipoaire;
begin
    Try
       AssignFile(Archivo_aire,path_aire);
        Reset(archivo_aire); { Intentar la apertura }
    readln(archivo_aire,cadena);
    while not eof(archivo_aire) do
    begin
     new(nodo);
     nodo.prox:=lista_aire;
     lista_aire:=nodo;
    cadena:=copy(cadena, pos(#9, cadena)+1, length(cadena));
    lista_aire.aire.t:=strTofloat(copy(cadena, 0, pos(#9, cadena)-1));
    cadena:=copy(cadena, pos(#9, cadena)+1, length(cadena));
    lista_aire.aire.c:=strTofloat(copy(cadena, 0, pos(#9, cadena)-1));
    cadena:=copy(cadena, pos(#9, cadena)+1, length(cadena));
    lista_aire.aire.p:=strTofloat(copy(cadena, 0, pos(#9, cadena)-1));
    cadena:=copy(cadena, pos(#9,cadena)+1, length(cadena));
    lista_aire.aire.v:=strTofloat(copy(cadena, 0, pos(#9, cadena)-1));
    cadena:=copy(cadena, pos(#9,cadena)+1, length(cadena));
    lista_aire.aire.k:=strTofloat(copy(cadena, 0, pos(#9, cadena)-1));
    cadena:=copy(cadena, pos(#9,cadena)+1, length(cadena));
    lista_aire.aire.pr:=strTofloat(cadena);
    readln(archivo_aire,cadena);
    end;
  Except begin{ Si falla }
        ShowMessage('No es posible abrir el archivo aire.dat');
   end;
end;
end;

//Actualiza interfaz  de botones y menú, en función de que se cree una
nueva ficha, se destruya las mallas o se creen las mallas//

 procedure TFprincipal.Actualiza_interfaz(const interfaz: Testilo_interfaz);
var
 estado: boolean;
I,N: INTEGER;
 begin
 case interfaz of
 EIficha_destruida,EIinicio_entrada:
            begin
            estado:= MDIchildcount <> 0;
            panel1.visible:=estado;
            panel2.visible:=estado;
            salvar1.enabled:=estado;
            Libreria.enabled:=estado;
            for i:=0 to Componentcount-1 do
            if components[i] is TSpeedButton then
            (components[i] as TSpeedButton).enabled:=estado;
            SpeedButton1.enabled:=estado;
            zoom.enabled:=estado;
            SPimprimir.enabled:=estado;
            SPabrir.enabled:=true;
            end;
EIficha_creada:
          begin
            panel3.enabled:=true;
```

```
            panel1.visible:=true;
            panel2.visible:=true;
            salvar1.enabled:=true;
            Libreria.enabled:=true;
            for i:=0 to Componentcount-1 do
            if components[i] is TSpeedButton then
            (components[i] as TSpeedButton).enabled:=false;
            SpeedButton1.enabled:=true;
            zoom.enabled:=true;
            SpeedButton1.enabled:=true;
             SPimprimir.enabled:=true;
          end;
 EImatrices_creadas:
            begin
              desacativa_comun_menu(true);
               for i:=0 to Componentcount-1 do
               if components[i] is TSpeedButton then
               (components[i] as TSpeedButton).enabled:=true;
            end;
 EImatrices_destruidas:
            begin
              desacativa_comun_menu(false);
              SpeedButton1.enabled:=true;
               for i:=0 to Componentcount-1 do
               if components[i] is TSpeedButton then
               (components[i] as TSpeedButton).enabled:=false;
               SPabrir.enabled:=true;
            end;
end;
 end;

//Actualiza interfaz  de botones y menú, en función de que se cree una
nueva ficha, se destruya las mallas o se creen las mallas//

 procedure TFprincipal.desacativa_comun_menu(valor: boolean);
 var
 I: integer;
 begin
     for I := 0 to (activeMDIchild as Tesquema).visibleinvisible1.Count-1 do
     (activeMDIchild as Tesquema).visibleinvisible1.Items[I].Enabled := valor;
       for I := 0 to (activeMDIchild as Tesquema).forma1.Count-1 do
     (activeMDIchild as Tesquema).forma1.Items[I].Enabled := valor;
        for I := 0 to (activeMDIchild as Tesquema).forma1.Count-1 do
     (activeMDIchild as Tesquema).forma1.Items[I].Enabled := valor;
        for I := 0 to (activeMDIchild as Tesquema).simulacion.Count-1 do
     (activeMDIchild as Tesquema).simulacion.Items[I].Enabled := valor;
(activeMDIchild as Tesquema).Ponerdatosmanualmente1.Enabled := valor;
 end;

procedure TFprincipal.MRUFileListMRUItemClick(Sender: TObject;
  AFilename: string);
var
 ficha: Tesquema;
begin
   ODabrir.filename:=afilename;
    application.createform(Tesquema,esquema);
   esquema.caption:=afilename;
   esquema.show;
   tile;
end;
```

```
//pinta el gradiente de color (barra de colores con número) del programa//


procedure TFprincipal.gradiantePaint(Sender: TObject);

var

n,arriba,izquierda: integer;

begin

if not (MDIchildcount=0) then
with gradiante do begin
SetmapMode(canvas.handle,MM_HIENGLISH);
SetViewportOrgEx(canvas.handle,0,clientheight,nil);
    izquierda:=3;
    arriba:=pixel_mil_ajustado(clientheight-40);
    alto_rect:=pixel_mil((clientheight-80)/(tonalidades+1));
    dibujo_rect.left:=pixel_mil_ajustado(izquierda);
    Font.Height:=5;
    dibujo_rect.right:=pixel_mil_ajustado(clientwidth)-
(canvas.textwidth(intTostr(Trunc(temp_max)))+canvas.textwidth('0')* DECIMAL1);
    ancho_rect:= dibujo_rect.right-dibujo_rect.left;
    for n:=tonalidades downto 0 do
    begin
        dibujo_rect.top:=arriba-round((tonalidades-n)*alto_rect);
        dibujo_rect.bottom:=arriba-round((tonalidades+1-n)*alto_rect);

       canvas.brush.color:=rgb(array_colores[n].pered,array_colores[n].pegreen,array_colores[n].peblue);
        FillRect(canvas.handle,dibujo_rect,canvas.brush.handle);
    end;
    canvas.pen.color:=clblack;
    canvas.brush.style:=bsclear;
    dibujo_rect.top:=arriba;
    canvas.Rectangle(dibujo_rect.left,dibujo_rect.top,
    dibujo_rect.right,dibujo_rect.bottom);
    dibujar_letras;
 end;
end;

// Actualiza  barra de datos del programa porque se va a cambiar de malla//

 procedure  TFprincipal.modificar_gradiente(colores: tpuntero_colores; tonalidades_aux: integer;
max,min,temp_ambiente_aux,TIEMPO: real;
  fisicos_aux:  Tpuntero_fisicos; var matriz_aux,matriz_aux_q: Tmatriz; var unidad: string);
begin
    matriz:= matriz_aux;
    StatusBar1.panels.Items[2].Text :='unidad : '+unidad;
   array_colores:=colores;
   tonalidades:=tonalidades_aux;
   temp_max:=max;
   temp_min:=min;
   fisicos:=fisicos_aux;
   cambio_decimales;
   temp_ambiente:=temp_ambiente_aux;
   rango_temp:=(temp_max-temp_min)/tonalidades;
   panel1.width:=(canvas.textwidth(intTostr(Trunc(temp_max)))+canvas.textwidth('0')*
DECIMAL1)+ancho_gradiente;
```

```
    gradiante.repaint;
    Lfilas.caption:='Filas: '+intTOstr(matriz_aux.max_fila);
    Lcolumnas.caption:='Columnas: '+intTOstr(matriz_aux.max_columna);
    LFxC.caption:='FxC: '+intTOstr(matriz_aux.max_columna*matriz_aux.max_fila);
    Lreticula.caption:='Dl: '+floattostrF(fisicos.delta,ffFixed,8,4);
end;

procedure TFprincipal.Panel1Resize(Sender: TObject);
begin
gradiante.width:=panel1.width;
gradiante.height:=panel1.height;
end;

{Si se pulsa detro dentro de la barra de colores del gradiente de color
sale un rectángulo con la temperatura en ese punto de la barra }

procedure TFprincipal.gradianteMouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
var
valor: real;
n,xx,yy: integer;
  begin
xx:=pixel_mil_ajustado(x);
yy:=pixel_mil_ajustado(gradiante.clientheight-y);
if puntorect(dibujo_rect,point(xx,yy)) then begin
    n:=trunc((yy-dibujo_rect.bottom)/alto_rect);
    valor:=rango_temp*n+temp_min;
   gradiante.hint:=floattostrF(valor,ffFixed,8,2)+'/'+floattostrF(valor+rango_temp,ffFixed,8,2);
end;
end;

//Dibuja la escala de números de la barra del gradiente de color//

procedure TFprincipal.dibujar_letras;
var
n,valores: integer;
cadena: string[10];
begin
if tonalidades<24 then
valores:=tonalidades
else
valores:=6;
with gradiante do begin
canvas.brush.style:=bsclear;
canvas.pen.color:=clblack;
    for  n:=0 to valores+1 do begin
        cadena:=floattostrF(tonalidades/valores*n*rango_temp+temp_min,ffFixed,8,DECIMAL1-1);

canvas.textout(dibujo_rect.right,round(round(tonalidades/valores*n)*alto_rect+canvas.textheight('0')/2+d
ibujo_rect.bottom),cadena);

canvas.moveTo(dibujo_rect.left,round(round(tonalidades/valores*n)*alto_rect+dibujo_rect.bottom));

canvas.lineto(round(dibujo_rect.left+ancho_rect/3),round(round(tonalidades/valores*n)*alto_rect+dibujo
_rect.bottom));

canvas.moveTo(round(dibujo_rect.left+2*ancho_rect/3),round(round(tonalidades/valores*n)*alto_rect+di
bujo_rect.bottom));

canvas.lineto(dibujo_rect.right,round(round(tonalidades/valores*n)*alto_rect+dibujo_rect.bottom));
```

```
    end;

end;
end;

procedure TFprincipal.izquierda2Click(Sender: TObject);
begin
panel1.align:=alleft;
end;

procedure TFprincipal.derecha1Click(Sender: TObject);
begin
panel1.align:=alright;
end;

procedure TFprincipal.ocultar1Click(Sender: TObject);
begin
panel1.visible:=not panel1.visible;
end;

//Se detruye todas las lista creadas//

procedure TFprincipal.FormDestroy(Sender: TObject);
var
puntero: tipoaire;
puntero_atributos: tipoatributos;
begin
    while lista_aire<> nil do begin
       puntero:=lista_aire;
       lista_aire:=lista_aire.prox;
        dispose(puntero);
    end;
    while lista_atributos<> nil do begin
       puntero_atributos:=lista_atributos;
       lista_atributos:=lista_atributos.prox;
        dispose(puntero_atributos);
    end;
end;

// Modifica valores de la barra de datos//

procedure TFprincipal.modificar_valores (cellx,celly: integer;  valor_dato,X,Y: real);
begin
lcelly.caption:='Celly: '+INTTOSTR(CELLY);
lcellX.caption:='Cellx: '+INTTOSTR(CELLX);
lval.caption:='Val: '+floattostrF(valor_dato,ffFixed,8,decimal2);
lXPOSICION.caption:='X: '+floattostrF(X,ffFixed,8,decimal2);
lYPOSICION.caption:='Y: '+floattostrF(Y,ffFixed,8,decimal2);
END;

// Modifica valores de la barra de datos//

procedure TFprincipal.cambio_decimales;
BEGIN
lval.caption:='Val: '+floattostrF(valor_dato,ffFixed,8,decimal2);
LMAX.CAPTION:='Max: '+floattostrF(TEMP_MAX,ffFixed,8,DECIMAL2);
LMIN.CAPTION:='Min: '+floattostrF(TEMP_MIN,ffFixed,8,DECIMAL2);
ltAMB.caption:='Tamb: '+floattostrF(TEMP_AMBIENTE,ffFixed,8,DECIMAL2);
llargo.caption:='Long: '+floattostrF(fisicos.largo_malla,ffFixed,8,DECIMAL2);
lancho.caption:='Ancho: '+floattostrF(fisicos.ancho_malla,ffFixed,8,DECIMAL2);
```

```
lMed.caption:='Med: '+floattostrF(matriz.valor_medio,ffFixed,8,DECIMAL2);
end;

// Modifica valores de la barra de datos//

procedure TFprincipal.valores_temperatura(var lista: tipolista; x,y: real);
 var
puntero: Tipolista;
 begin
  puntero:=lista;
  while puntero <> nil do
  BEGIN
          if puntorect_especial(puntero.rectangulo,x,y) then
          begin
          ltj.caption:='Tj: '+floattostrF(puntero.atributos.tj,ffFixed,8,DECIMAL2);
          ltc.caption:='Tc: '+floattostrF(puntero.atributos.tc,ffFixed,8,DECIMAL2);
          ltp.caption:='Tp: '+floattostrF(puntero.atributos.tp,ffFixed,8,DECIMAL2);
          exit;
          end;
   puntero:=puntero.prox;
  end;
end;

//Cambia la presición de la escala de temperatura del gradiente de color//
procedure TFprincipal.gradianteDblClick(Sender: TObject);
var
ClickedOK: Boolean;
NewString: string;
numero: integer;
begin
 ClickedOK := InputQuery('Los valores van de [0..3]', 'el valor es:', NewString);
  if ClickedOK then
    try
    numero:=strTOint(NewString);
      if (numero <4) and (numero >=0)  then
         if (sender as Tcomponent).tag=1 then begin
         DECIMAL1:=numero+1;
         panel1.width:=(canvas.textwidth(intTostr(Trunc(temp_max)))+canvas.textwidth('0')*
DECIMAL1)+ancho_gradiente;
         gradiante.repaint;
         end
         else begin
         decimal2:=numero;
         cambio_decimales;
         end
    else
         showmessage('El número de decimales  es de: [0 a 3]');
    except
       showmessage('No es una cadena de numeros');
    end;
    end;

{Salva los datos térmicos de la placa, en un archivo con el mismo
nombre que el programa, pero con exteción txt}

procedure TFprincipal.Salvar1Click(Sender: TObject);
var
cadena: string;
ruta: TFileName;
begin
```

```
with (activeMDIchild as Tesquema) do begin
ruta:=caption;
delete(ruta,pos('.',ruta),length(ruta));
ruta:=ruta+'.txt';
SDsalvar.filename:=ruta;
if SDsalvar.execute then
salvar_archivo_atributos_esquema(archivo_esquema_atributos,archivo_cabecera,lista,fisicos);
archivo_esquema_atributos.saveTofile(SDsalvar.filename);
reset(archivo_esquema);
readln(archivo_esquema,cadena);
if pos(cadena,cabecera_protel)=0 then
poner_datos_protel(archivo_esquema,lista)
else
poner_datos_ULPGC(archivo_esquema,lista);
end;
end;

procedure TFprincipal.Izquierda3Click(Sender: TObject);
begin
panel2.align:=alleft;
end;

procedure TFprincipal.derecha2Click(Sender: TObject);
begin
panel2.align:=alright;
end;

procedure TFprincipal.ocultar2Click(Sender: TObject);
begin
panel2.visible:=not panel2.visible;
end;

procedure TFprincipal.Izquierda4Click(Sender: TObject);
begin
panel2.align:=alleft;
end;

procedure TFprincipal.derecha3Click(Sender: TObject);
begin
panel2.align:=alright;
end;

procedure TFprincipal.ocultarver1Click(Sender: TObject);
begin
panel2.visible:=not panel2.visible;
end;

//Abre la librería del programa//

procedure TFprincipal.ComponenteClick(Sender: TObject);
var
puntero: tipolista;
begin
new(puntero);
puntero.atributos:=lista_atributos.atributos;
Fverdatos.modo_entrada (ESmodificar_componente_libreria,puntero,(activeMDIchild as
Tesquema).lista,@fisicos,matriz,matrizQ);
 Fverdatos.showmodal;
dispose(puntero);
end;
```

```
procedure TFprincipal.SpeedButton7Click(Sender: TObject);
begin
(activeMDIchild as Tesquema).sor41Click(self);
end;

procedure TFprincipal.SpeedButton4Click(Sender: TObject);
begin
Salvar1Click(self);
end;

// Modifica valores de la barra de datos//

procedure TFprincipal.modificar_valores_posicion(x,y: real; digitos: integer;const  posicion_unidad:
openstring);
begin
StatusBar1.panels.Items[0].Text :='X : '+floattostrF(x,ffFixed,8,digitos)+posicion_unidad;
StatusBar1.panels.Items[1].Text :='Y : '+floattostrF(Y,ffFixed,8,digitos)+posicion_unidad;
END;

//Cambia los datos de la malla activa manualmente//

procedure TFprincipal.SpeedButton1Click(Sender: TObject);
begin
(activeMDIchild as Tesquema).Manual1Click(self);
end;

//zomm para ampliar//

procedure TFprincipal.zoomClick(Sender: TObject);
var
key: word;
Shift: TShiftState;
begin
key:=VK_next;
(activeMDIchild as Tesquema).FormKeyDown(self, Key,
  Shift);
end;

//Abre ventana de datos físicos//

procedure TFprincipal.SpeedButton6Click(Sender: TObject);
begin
(activeMDIchild as Tesquema).Datosfisicos1Click(self);
end;

//Imprime el esquema//

procedure TFprincipal.SPimprimirClick(Sender: TObject);
begin
(activeMDIchild as Tesquema).Imprimir1Click(self);
end;

// zoom para disminuir//

procedure TFprincipal.SpeedButton9Click(Sender: TObject);
var
key: word;
Shift: TShiftState;
begin
```

```
key:=VK_prior;
(activeMDIchild as Tesquema).FormKeyDown(self, Key,
  Shift);
end;

//Centra esquema//

procedure TFprincipal.SpeedButton2Click(Sender: TObject);
begin
(activeMDIchild as Tesquema).centra_punto_esquema;
 (activeMDIchild as Tesquema).Refresh;
end;

//Abre archivo de placa//

procedure TFprincipal.SPabrirClick(Sender: TObject);
begin
if Fprincipal.ODabrir.execute then
      if  extractfileext(ODabrir.filename) = '.pcb' then
      begin
        Fprincipal.ODabrir.historylist.add(ODabrir.filename);
        Fprincipal.SDsalvar.historylist.add(ODabrir.filename);
         TRY
        application.createform(Tesquema,esquema);
       esquema.caption:=ODabrir.filename;
        esquema.show;
        tile;
        EXCEPT
        END;
       end;
end;
end.
```

unit SOR3b;

```
interface
uses
Dialogs,tipo1,H2,calculo_potencia1,Windows,calcula_h_forzado,calcula_h_natural,lu;
function matriz_punto(const rectangulo_cuadricula_escalado: Trect;
var punto: tpoint; delta: real; escala: real): boolean;
procedure poner_temperatura_lista(var lista: tipolista; var
matrizT,matrizQ,matrizTAIRE_abajo,matrizTaire_arriba: Tmatriz;
var fisicos: Tparametros_fisicos);
procedure PSOR(var
MATRIZk,MATRIZQ,matrizH_ARRIBA,matrizH_ABAJO,matrizH_BORDE,matrizTAIRE_ARRIBA,
matrizTAIRE_ABAJO, UP ,MATRIZ_esp: Tmatriz; var fisicos: Tparametros_fisicos ;
rectangulo_cuadricula: Trect; funcion_turbulento,
funcion_laminar: Tfuncion; var lista_aire: tipoaire; var lista: tipolista );

implementation

{Es el procedimiento principal de la simulación, en él se calcula los
valores de los nodos centrales de las llamadas y las llamadas a los procedimientos
para calcular los nodos laterales, esquina, potencia en cada iteración y
calculo del coeficiente de transferencia de cada nodo de la malla}

procedure PSOR(var
MATRIZk,MATRIZQ,matrizH_ARRIBA,matrizH_ABAJO,matrizH_BORDE,matrizTAIRE_ARRIBA,
matrizTAIRE_ABAJO, UP ,MATRIZ_esp: Tmatriz; var fisicos: Tparametros_fisicos ;
rectangulo_cuadricula: Trect; funcion_turbulento,
funcion_laminar: Tfuncion; var lista_aire: tipoaire; var lista: tipolista );
const
maxits=1000;
EPS=0.001;
var
error_de_colocacion: boolean;
N,A,C,F: integer;
TIME,RJAC,W,CAMBIO,RESID,TEMP,MED_TEMP,MED_TEMP_ARRIBA,MED_TEMP_ABAJO:
double;
begin
RJAC:=sqr(COS(PI/MATRIZQ.MAX_COLUMNA) + COS(PI/MATRIZQ.MAX_FILA));
W:=4/(2 + SQRT(4 - RJAC));  //ES LA VARIABLE DE ACELERACION DEL ALGORITMO SOR//
TIME:=0;
   { _____
     | dentro    |
     |  es        |
     | el analisis |
     |_____ _ | }

MED_TEMP_ARRIBA:=25;
fisicos.tmax:=0;
centro_rect(lista,rectangulo_cuadricula,fisicos.delta_in,1);
cambio:=1;
if (fisicos.estilo_potencia=EPautomatico) or (fisicos.estilo_potencia=EPautomatico_manual) then
potencia_primaria(lista,up,matrizQ,matrizTAIRE_ARRIBA,matrizTAIRE_abajo,fisicos,rectangulo_cuad
ricul      a,fisicos.delta_in,cambio);
  MED_TEMP_ARRIBA:=fisicos.temp_media_aire_arriba;
if (fisicos.estilo_h=EHFORZADO) THEN
  calculo_h_forzado(up,matrizTAIRE_ABAJO,MATRIZH_ABAJO,MATRIZ_ESP, fisicos,
rectangulo_cuadricula, funcion_turbulento,
```

```
  funcion_laminar,lista_aire,MED_TEMP_ARRIBA,cambio);
IF (fisicos.estilo_h=EHNATURAL) THEN
  calculo_h_NATURAL(up,matrizTAIRE_ABAJO,MATRIZH_ABAJO,MATRIZ_ESP, fisicos,
rectangulo_cuadricula, funcion_turbulento,
  funcion_laminar,lista_aire,MED_TEMP_ARRIBA,cambio);
MED_TEMP_ARRIBA:=0;
WITH UP DO BEGIN
FOR N:=1 to fisicos.iteraciones DO BEGIN
    fisicos.TMAX:=fisicos.dt+fisicos.Tmax;
    CAMBIO:=0.0;
    MED_TEMP_ARRIBA:=0;
   FOR F:=2 to MAX_FILA - 1 DO
     FOR C:=2 TO MAX_COLUMNA - 1 DO  BEGIN
        RESID:=FISICOS.RADIO1 * (MATRIZ[F+1,C] + MATRIZ[F-1,C] + MATRIZ[F,C+1] +
        MATRIZ[F,C-1] - 4*MATRIZ[F,C])+
        FISICOS.RADIO2 *
(MATRIZQ.MATRIZ[F,C]+matrizTAIRE_ARRIBA.MATRIZ[F,C]*matrizH_ARRIBA.MATRIZ[F,C]+
matrizTAIRE_ABAJO.MATRIZ[F,C]*matrizH_ABAJO.MATRIZ[F,C]-
        MATRIZ[F,C] * matrizH_ARRIBA.MATRIZ[F,C]- MATRIZ[F,C] *
matrizH_ABAJO.MATRIZ[F,C]);
        IF  CAMBIO < ABS(RESID) THEN
        CAMBIO:=ABS(RESID);
        MATRIZ[F,C]:=MATRIZ[F,C] +W *  RESID;
        MED_TEMP_ARRIBA:=MED_TEMP_ARRIBA+MATRIZ[F,C];
   END;
   MED_TEMP_ARRIBA:=MED_TEMP_ARRIBA/(MAX_COLUMNA * MAX_FILA);
PH(MATRIZQ,matrizH_ARRIBA,matrizH_ABAJO,matrizH_BORDE,
matrizTAIRE_ARRIBA,matrizTAIRE_ABAJO,UP,FISICOS,W,CAMBIO,MED_TEMP_ARRIBA);
if (fisicos.estilo_potencia=EPautomatico) or (fisicos.estilo_potencia=EPautomatico_manual) then
  error_de_colocacion:=nueva_potencia(lista,
up,matrizQ,matrizTAIRE_ARRIBA,matrizTAIRE_abajo,cambio,rectangulo_cuadricula,fisicos,fisicos.del
ta_in);
if (fisicos.estilo_h=EHFORZADO) THEN
  calculo_h_forzado(up,matrizTAIRE_ABAJO,MATRIZH_ABAJO,MATRIZ_ESP, fisicos,
rectangulo_cuadricula, funcion_turbulento,
  funcion_laminar,lista_aire,MED_TEMP_ARRIBA,cambio);
IF (fisicos.estilo_h=EHNATURAL) THEN
  calculo_h_NATURAL(up,matrizTAIRE_ABAJO,MATRIZH_ABAJO,MATRIZ_ESP, fisicos,
rectangulo_cuadricula, funcion_turbulento,
  funcion_laminar,lista_aire,MED_TEMP_ARRIBA,cambio);
      if  (CAMBIO < fisicos.TOL) then  begin
      if error_de_colocacion then
     poner_temperatura_lista(lista,up,matrizQ,matrizTAIRE_ABAJO,matrizTAIRE_arriba,fisicos);
     exit;
     END;
end;

END;
 Showmessage('Las interaciones son insuficientes para alcanzar el error');
 poner_temperatura_lista(lista,up,matrizQ,matrizTAIRE_ARRIBA,matrizTAIRE_arriba,fisicos);
end;

{Se en carga de calcular los valores de temperatura de la unión,cápsula y placa
de cada componente}

procedure poner_temperatura_lista(var lista: tipolista; var
matrizT,matrizQ,matrizTAIRE_abajo,matrizTaire_arriba: Tmatriz;
var fisicos: Tparametros_fisicos);
var
suma_potencia_pines: real;
```

```
 n: integer;
 puntero: Tipolista;
 tBUP,tcc,vcc,media_valor_calor,media_valor_temperatura,media_vcc,valor: real;
 pads: tipopad;
 begin
   puntero:=lista;
   while puntero <> nil do begin
    if puntero.atributos.cell.x<>0 then  begin
     with puntero.atributos do BEGIN
     case puntero.atributos.estilo_modo_potencia of
     EMPpines:
           begin
           pads:=puntero.pad;
           n:=1;
           media_valor_calor:=0;
           media_valor_temperatura:=0;
           suma_potencia_pines:=0;
             while pads<> nil do begin
                suma_potencia_pines:=suma_potencia_pines+puntero.vector_pines.vector[n];
                valor:=puntero.vector_pines.vector[n]/(espesorpin*anchopin);
               media_valor_calor:=media_valor_calor+valor;
          media_valor_temperatura:=media_valor_temperatura+matrizT.matriz[pads.cell.y,pads.cell.x];
               pads:=pads.prox;
               INC(n);
             end;
           calor:=media_valor_calor/puntero.atributos.numero_pines;
           Tc:=rthca*(potencia-suma_potencia_pines)+matrizTAIRE_arriba.matriz[cell.y,cell.x];
           Tj:=potencia*rthjc+Tc;
           tp:=media_valor_temperatura/puntero.atributos.numero_pines;
           end;
   EMPcomponentes:
   begin
   tc:=1/(Rthca+Rthcp+rTHP) * (potencia*(Rthca*(Rthcp+rthp))+
   matrizT.matriz[cell.y,cell.x]*RThca+
   matrizTAIRE_ABAJO.matriz[cell.y,cell.x]*(Rthcp+rthp));
   Tp:=matrizT.matriz[cell.y,cell.x];
   Tj:=potencia*rthjc+TC;
   end;
      end;
      end;
end;
  puntero:=puntero.prox;
end;
end;

//Consigue en nodo centrado  debajo de cada componente //

function matriz_punto(const rectangulo_cuadricula_escalado: Trect;
var punto: tpoint; delta: real; escala: real): boolean;
var
delta_escalada: real;
begin
if puntorect(rectangulo_cuadricula_escalado,punto) then begin
    delta_escalada:=delta*escala;
    punto.x:=round((punto.x-rectangulo_cuadricula_escalado.left)/delta_escalada)+1;
    punto.y:=round((punto.y-rectangulo_cuadricula_escalado.bottom)/delta_escalada)+1;
end;
end;

end.
```

unit surface;

```
interface
uses
Dialogs,Windows,Graphics,Forms,Classes,tipo1,sor,SysUtils;

procedure Psurface(var matriz: Tmatriz;var array_colores: colores;
  rectangulo_cuadricula_escalado,rect_pantalla: Trect; tonalidades: integer;
delta,temp_ambiente: real; temp_max,temp_min, escala: real;  scroll_dibujo: Tpoint; var ventana:
Tcanvas; var  brocha: Tbrush; const estilo_forma: Testilo_forma);
function cuadricula_Adibujar(var matriz: Tmatriz;
var rectangulo_cuadricula_escalado,rect_pantalla: Trect;
delta: real; escala: real): Trect;
function punto_matriz_localizacion(const rectangulo_cuadricula_escalado: Trect;
 var fila,columna: integer; var salidax,saliday: real; delta: real; escala: real): boolean;
procedure paleta_estilo(var array_colores: colores;var  tonalidades: integer;
estilo: Testilo_colores);
procedure paleta_hot(var array_colores: colores; num_colores: integer);
procedure paleta_cool(var array_colores: colores; num_colores: integer);
procedure paleta_gray(var array_colores: colores; num_colores: integer);
procedure paleta_estandar(var array_colores: colores; num_colores: integer);
procedure establecer_contornos(var  matriz: Tmatriz);
procedure establecer_bordes(var  matriz: Tmatriz; derecha,izquierda,abajo,arriba: real);

implementation

// Dibuja la malla de colores de la temperatura//

procedure Psurface(var matriz: Tmatriz;var array_colores: colores;
  rectangulo_cuadricula_escalado,rect_pantalla: Trect; tonalidades: integer;
delta,temp_ambiente: real; temp_max,temp_min, escala: real;  scroll_dibujo: Tpoint; var ventana:
Tcanvas; var  brocha: Tbrush; const estilo_forma: Testilo_forma);
var
columna,fila,ajusteX_barra,ajusteY_barra,n: integer;
rango_temp,delta_escalada,izquierda,derecha,abajo,arriba: real;
cuadricula_visible,dibujo_rect,rectangulo_aux,rrr: Trect;
puntox,puntoy: real;
begin
setrect(rect_pantalla, pixel_mil_ajustado(rect_pantalla.left),pixel_mil_ajustado(rect_pantalla.top),
pixel_mil_ajustado(rect_pantalla.right),pixel_mil_ajustado(rect_pantalla.bottom));
 (*se coprueba que el rectangulo a dibujar esta dentro de la pantalla
si no esta se sale porque no se ha  dibujarse nada*)
cuadricula_visible:=cuadricula_Adibujar(matriz, rectangulo_cuadricula_escalado,
rect_pantalla,delta,escala);
if isrectvacio(cuadricula_visible) then
  exit
else begin
    delta_escalada:=delta * escala;
    rango_temp:=(temp_max-temp_min)/(tonalidades);
    if rango_temp=0 then rango_temp:=1;
    punto_matriz_localizacion( rectangulo_cuadricula_escalado,
    cuadricula_visible.left,cuadricula_visible.bottom,puntox,puntoy,delta,escala);
    scroll_dibujo:=punto_pixel_mil(scroll_dibujo);
    izquierda:=(puntox-delta_escalada/2-scroll_dibujo.x);
    derecha:=(puntox+delta_escalada/2-scroll_dibujo.x);
    abajo:=(puntoy-delta_escalada/2-scroll_dibujo.y);
    arriba:=(puntoy+delta_escalada/2-scroll_dibujo.y);
    rectangulo_aux:=dibujo_rect;
     (*lo que se hace con esto es que empiecen en cero, para poderlo utilizar
     para construir el rectangulo que se quiere dibujar*)
```

```
  cuadricula_visible.right:=cuadricula_visible.right-cuadricula_visible.left;
    cuadricula_visible.top:=cuadricula_visible.top-cuadricula_visible.bottom;
     ventana.pen.style:=pssolid;
     ventana.pen.width:=2;
     ventana.pen.color:=rgb($00,$7f,$00);
     ventana.brush.color:=rgb($00,$7f,$00);
     ventana.brush.style:=bsclear;
 for columna:=0 to cuadricula_visible.right do  begin
  dibujo_rect.left:=round((izquierda+columna*delta_escalada));
  dibujo_rect.right:=round((derecha+columna*delta_escalada));
     for fila:=0 to cuadricula_visible.top do begin
         dibujo_rect.bottom:=round(abajo+fila*delta_escalada);
         dibujo_rect.top:=round(arriba+fila*delta_escalada);
         case estilo_forma of
         relleno: begin
   n:=round((matriz.matriz^[cuadricula_visible.bottom+fila,cuadricula_visible.left+columna]-
  temp_min)/rango_temp);
 ventana.brush.color:=rgb(array_colores[n].pered,array_colores[n].pegreen,array_colores[n].peblue);
                ventana.FillRect(dibujo_rect);
               end;
         cuadrado_relleno: begin
n:=round((matriz.matriz^[cuadricula_visible.bottom+fila,cuadricula_visible.left+columna]-
temp_min)/rango_temp);
ventana.brush.color:=rgb(array_colores[n].pered,array_colores[n].pegreen,array_colores[n].peblue);
                ventana.Rectangle(dibujo_rect.left,dibujo_rect.top,dibujo_rect.right,dibujo_rect.bottom);
                    end;
         cuadrado:
ventana.Rectangle(dibujo_rect.left,dibujo_rect.top,dibujo_rect.right,dibujo_rect.bottom);
circulo:  ventana.ellipse(dibujo_rect.left,dibujo_rect.top,dibujo_rect.right,dibujo_rect.bottom);
         valores:  begin
                ventana.pen.color:=clYellow;
                ventana.TextRect(dibujo_rect,dibujo_rect.left,
dibujo_rect.top,floattostrF(matriz.matriz^[cuadricula_visible.bottom+fila,cuadricula_visible.left+columna
],ffFixed,8,1));
                end;
         end;
         end;
end;
 ventana.brush.style:=bsclear;
 ventana.pen.style:=pssolid;
 ventana.pen.width:=2;
 dibujo_rect:=rectangulo_cuadricula_escalado;
 offsetrect(dibujo_rect,-scroll_dibujo.x,-scroll_dibujo.y);
 ventana.pen.color:=clwhite;
 ventana.rectangle(dibujo_rect.left,dibujo_rect.top,dibujo_rect.right,dibujo_rect.bottom);
end;
end;

(*con esto se comprueva que exista intersección entre el rectangulo pantalla
y el rectágulo de la malla,sí es asi entoces se obtiene los puntos
de rectangulo de la reticula que hay que dibujar, sino se da el rectangulo
como vacio*)

function cuadricula_Adibujar(var matriz: Tmatriz;
var rectangulo_cuadricula_escalado,rect_pantalla: Trect;
delta: real; escala: real): Trect;
var
aux: integer;
begin
 if intersrect(result,rectangulo_cuadricula_escalado,rect_pantalla) then
```

```
   begin
   result.top:=result.top+round(delta*escala);
   result.bottom:=result.bottom-round(delta*escala);
   result.left:=result.left-round(delta*escala);
   result.right:=result.right+round(delta*escala)
   end;
   intersrect(result,result,rectangulo_cuadricula_escalado);
   localizacion_matriz_punto(rectangulo_cuadricula_escalado,result.topleft,delta,
   escala);
   localizacion_matriz_punto(rectangulo_cuadricula_escalado,result.BottomRight,delta,
   escala);
end;

//Se Obtine la posición de pantalla de una fila y columna //

function punto_matriz_localizacion(const rectangulo_cuadricula_escalado: Trect;
 var fila,columna: integer; var salidax,saliday: real; delta: real; escala: real): boolean;
var
delta_escalada: real;
 begin
 delta_escalada:=delta*escala;
salidax:=rectangulo_cuadricula_escalado.left + (fila-1) * delta_escalada;
saliday:=rectangulo_cuadricula_escalado.bottom + (columna-1) * delta_escalada;
end;

procedure paleta_estilo(var array_colores: colores;var  tonalidades: integer;
estilo: Testilo_colores);
begin
 case estilo of
 gray: paleta_gray(array_colores,tonalidades);
 cool: paleta_cool(array_colores,tonalidades);
 hot:  paleta_hot(array_colores,tonalidades);
 estandar: paleta_estandar(array_colores,tonalidades);
 end;
 end;

//calcula tonalidades del estilo gray//

procedure paleta_gray(var array_colores: colores; num_colores: integer);
var
m,n: integer;
begin
    m:=num_colores;
    for n:=0 to m do begin
       array_colores[n].pered:=round($ff* n/m);
       array_colores[n].pegreen:= array_colores[n].pered;
       array_colores[n].peblue:= array_colores[n].pered;
    end;
end;

//obtiene tonalidades del estilo cool//

procedure paleta_cool(var array_colores: colores; num_colores: integer);
var
n,m: integer;
begin
    m:=num_colores;
    for n:=0 to m do  begin
       array_colores[n].pered:=round($ff * n/m);
       array_colores[n].pegreen:=round($ff*(1- n/m));
```

```
      array_colores[n].peblue:=$ff;
    end;
end;

//obtiene tonalidades del estilo standar//

procedure paleta_estandar(var array_colores: colores; num_colores: integer);
var
p,n,t: integer;
begin
    p:=num_colores div 4;
    t:=p div 2;
    for n:=0 to t-1 DO BEGIN
      array_colores[n].pered:=$00;
      array_colores[n].pegreen:=$00;
      array_colores[n].peblue:=round($ff*(t+n+1)/p);
    end;
    for n:=0 to p DO BEGIN
      array_colores[t+n].pered:=$00;
      array_colores[t+n].pegreen:=round($ff * n/p);
      array_colores[t+n].peblue:=$ff;
    end;
    for n:=0 to p DO BEGIN
      array_colores[p+t+n].pered:=round($FF* n/p);
      array_colores[p+t+n].pegreen:=$ff;
      array_colores[p+t+n].peblue:=round($ff* (p-n)/p);
    end;
    for n:=0 to p DO BEGIN
      array_colores[2*p+t+n].pered:=$ff;
      array_colores[2*p+t+n].pegreen:=round($ff* (p-n)/p);
      array_colores[2*p+t+n].peblue:=$00;
    end;
    for n:=0 to  (num_colores-(3*p+t))  DO BEGIN
      array_colores[3*p+t+n].pered:=round($ff* (p-n)/p);
      array_colores[3*p+t+n].pegreen:=$00;
      array_colores[3*p+t+n].peblue:=$00;
    end;
end;

//obtiene tonalidades del estilo hot//

procedure paleta_hot(var array_colores: colores; num_colores: integer);
var
m,n,t: integer;
BEGIN
          m:=num_colores;
          n:=trunc(3/8*m);
          {se saca las tonalidades del color rojo}
          for t:=0 to n-1 do
             array_colores[t].pered:=round($ff* (t+1)/n);
          for t:=n to m do
             array_colores[t].pered:=$ff;
          {se saca las tonalidades del color verde}
          for t:=0 to n-1 do
             array_colores[t].pegreen:=$00;
          for t:=n to 2*n do
             array_colores[t].pegreen:=round($ff* (t-n+1)/n);
          for t:=2*n to m do
              array_colores[t].pegreen:=$ff;
           {se saca las tonalidades del color azul}
```

```
        for t:=0 to (2*n)-1 do
          array_colores[t].peblue:=$00;
        for t:=2*n to m do
          array_colores[t].peblue:=round($ff *(t-(2*n))/(m-2*n));
end;
end.
```

unit tipo_eventos;

```
interface

const
 array_unidades: array[0..14] of string[10] =
            (' ',
            ' ºC',' RTHJ', ' W/m ºC',' calor',' mm',' mil', ' W', ' m/s', ' s', ' W/m² ºC', ' m', ' Kg/m³', ' KJ/Kg
ºC', ' W/m²' );

procedure mmKeyPress(Sender: TObject; var Key: Char) far;
FUNCTION QUITAR_ERRORES( CADENA: OPENsTRING): STRING far ;
procedure FF(Sender: TObject);
function poner_unidad(unidad: openstring; const cadena: openstring): string far;

implementation

procedure ff(Sender: TObject);
begin
end;
procedure mmKeyPress(Sender: TObject; var Key: Char);
begin
      { Sólo permitir la entrada de caracteres='0'..'9',',','e',#8,'-','+','E' }
    case key of
     '0'..'9',',','e',#8,'-','+','E': key:=key;
      else
      Key := #0;
    end;
end;

{@**************************************************************************
Lo único que hace esta función es teniendo una cadena (que es la cantidad
en cifras, reprensentando una magnitud ) y la unidad de esa magnitud, todo
lo que no sea numeros 'e', 'E', '+', '-' y ',' se elimina y luego al final se pone la
unidad.
**************************************************************************}

 function poner_unidad(unidad: openstring; const cadena: openstring): string;
var
contador: integer;
begin
result:=cadena;
contador:=length(result);
contador:=1;
while contador<= length(result) do
begin
  case result[contador] of
   '0'..'9','+','-','e','E',',': contador:=contador+1;
  else
  begin
      delete(result,contador,1);
      if not contador=1 then
       contador:=contador-1
  end;
 end;
end;
result:= result + unidad;

end;
end.
```

unit tipo1;

```
interface
uses
 Controls,classes,extctrls,graphics,Windows,Dialogs;
const
ancho_letra=0.058987;
altura_letra=0.060;
cabecera_protel2='¡PCB FILE';
cabecera_protel= 'VERSION 2.00';
cabecera_roen= 'Última modificación';
path_atributos='\archivos\atributos.txt';
path_aire='\archivos\aire.txt';
path_cabecera='\archivos\cabecera.txt';
path_componente='\archivos\componet.txt';
path_letras='\archivos\letras.DAT';
path_ayuda='\ayuda\termín.hlp';

MaxArrayElements = (High(cardinal) - $f) div    SizeOf(real) ;

type

Testilo_formato_archivo=(EFAPROTEL,EFAULPGC);
Testilo_unidades=(EUmilimetros,EUpulgadas);
Testilo_modo_potencia=(EMPpines,EMPpuntos,EMPcomponentes);
Testilo_potencia=(EPconstante,EPmanual,EPautomatico,EPautomatico_manual);
Testilo_h=(EHtipo0,EHforzado,EHnatural,EHtipo3,EHCONSTANTE,EHMANUAL);
Testilo_ambiente=(EAMANUAL,EACONSTANTE);
Testilo_modificar_componente=(ESmodificar_comoponente_esquema,ESentrar_componente_libreria,ES
modificar_componente_libreria);
Testilo_capa=(ECTEMP_PLACA_ARRIBA,ECPOTENCIA_COMPONENTE,ECCOEF_TRNANS_AR
RIBA,ECCOEF_TRNANS_ABAJO,ECCOEF_TRNANS_BORDE
,ECTEMP_AMBIENTE_ARRIBA,ECTEMP_AMBIENTE_ABAJO,ECTEMP_AMBIENTE_BORDE,E
CONDUCTIVIDAD_PLACA);
Testilo_colores=(cool,gray,hot,estandar);
Testilo_forma=(relleno,circulo,cuadrado,cuadrado_relleno,valores,nada);
Testilo_interfaz=(EIficha_destruida,EIficha_creada,EImatrices_creadas,EImatrices_destruidas,
EIinicio_entrada);
Testilo_dibujos=(CDrectas,CDrectangulo,CDarcos,CDletras,CDcolores,CDrejilla,CDpads,CDcirculos,C
Drectagulo_componente,CDrectas_placa,
CDSIMBOLO_PREPARADO,CDSIMBOLO_DESACTIVADO);
Testilo_pulsacion=(vacio,intermedio,mover_rect,mover_componente,datos_manuales,dimensionar_matri
z,
preparando_imprimir,imprimir,rectfica_rectangulo_superficie);
Tposicion_atributos= array [1..20] of integer;
Tconjunto_dibujos= set of Testilo_dibujos;
TMyDynamicvector =  array [1..MaxArrayElements] of real;
PMyDynamicvector = ^TMyDynamicvector;
TMyDynamicmatriz = array[1..MaxArrayElements] of PMyDynamicvector;
PMyDynamicmatriz =  ^TMyDynamicmatriz;
TMyDynamicmatriz3D = array[1..MaxArrayElements] of PMyDynamicmatriz;
PMyDynamicmatriz3D =  ^TMyDynamicmatriz3D;

colores= array [0..255] of TPALETTEENTRY;
Tpuntero_colores= ^colores;

Tvector= record
          vector: PMyDynamicvector;
          NumElementos: integer;
      end;
```

```
Tmatriz= record
          matriz: PMyDynamicmatriz;
          max_fila,
          max_columna: integer;
          max_valor,min_valor: real;
          valor_medio: real;
          activa: boolean;
       end;

Tmatriz3D= record
          matriz: PMyDynamicmatriz3D;
          max_fila,
          max_columna,max_ancho: integer;
       end;
Tlista_aire= record
         t,p,c,v,k,pr: real;
         end;

tipoAIRE=^AIRE_nodo;
   AIRE_nodo= record
          AIRE: Tlista_aire;
          prox: tipoAIRE;
          end;

Tparametros_fisicos=record
                Q,U,C,P,K,DELTA,delta_in,R1,RADIO1,RADIO2,RADIO3,RADIO4,DT,ESPESOR: real;
                K_aislante,espesor_aislante: real;
                H_MINIMO_ARRIBA,h_MINIMO_ABAJO,h_MINIMO_BORDE: REAL;
                estilo_h: Testilo_h;
                h_arriba,h_abajo,h_borde_izquierdo,h_borde_derecho,
                h_borde_arriba,h_borde_abajo: real;
                TMAX,TOL: real;
                iteraciones: integer;
                estilo_ambiente: Testilo_ambiente;
                temp_ambiente_arriba,temp_ambiente_abajo: real;
                estilo_potencia: Testilo_potencia;
                estilo_modo_potencia: Testilo_modo_potencia;
                potencia_dentro,potencia_izquierda,potencia_derecha,pontecia_arriba,potencia_abajo: real;
                ancho_malla,largo_malla: real;
                temp_media_aire_arriba: real;
                temp_media_aire_abajo: real;
                end;

Tpuntero_fisicos= ^Tparametros_fisicos;
CADENA30=  string[30];
general_array= array[1..38] of tpoint;
 Tatributos=record
          Referencia,componente,tipo,material: cadena30;
          Ancho,Espesor,largo,largopin,anchopin,espesorpin: real;
          conductividad,conductividadpin,potencia,calor: real;
          Xposicion, Yposicion: integer;
          cell: Tpoint;
          Tj,Tc,Tp: real;
          RTHp,RTHca,RTHjc,RTHcp,rthpin: real;
          estilo_modo_potencia:  Testilo_modo_potencia;
          activo: boolean;
          numero_pines: integer;
        end;
 tipoatributos=^atributos_nodo;
```

```
atributos_nodo= record
        atributos: Tatributos;
        prox: tipoatributos;
        end;

Tpuntero_atributos=^Tatributos;
Trectangulo=  record
          izquierda,derecha,arriba,abajo: real;
          end;

Tletra=   record
            letra_char: char;
            letra_array  :general_array;
            final_letra: integer;
        end;

tipoletra=^letra_nodo;
letra_nodo= record
        letras: Tletra;
        prox: tipoletra;
        end;

 tiporecorre_letras=^recorre_letras;
 recorre_letras=record
            posicion_letra: tipoletra;
            prox: tiporecorre_letras;
            end;

 tiporecta=^recta_nodo;
 recta_nodo= record
          X1,Y1,X2,Y2: real ;
          ancho: real;
        prox: tiporecta;
        end;

Testructura=record
             recta: tiporecta;
             ancho: real;
            rectangulo: Trect;
         end;

Tarco =    record
            angulo_1,angulo_2: real;
            centro_ejex: real;
            centro_ejey: real;
            radio: real;
            rectangulo: Trectangulo;
            inicio_arcox: real;
            inicio_arcoy: real;
            fin_arcox: real;
            fin_arcoy: real;
            ancho: real;
        end;

tipoarco=^arco_nodo;
arco_nodo= record
        arco: Tarco;
        prox: tipoarco;
        end;
```

```
  tipopad=^pad_nodo;
  pad_nodo= record
          rectangulo_exterior,rectangulo_interior: trectangulo;
          rect_localiza_pin_matriz: Trect;
          cell: tpoint;
          prox: tipopad;
          end;
  tipocirculo=^circulo_nodo;
  circulo_nodo= record
          rectangulo: trectangulo;
          ancho: real;
          prox: tipocirculo;
          end;

tipolista=^lista_nodo;
lista_nodo=record
          enabled: boolean;
          preparado: boolean;
          atributos: Tatributos;
          estructura: Testructura;
          x,y: integer;
          prox: tipolista;
          letras_referencia: tiporecorre_letras;
          arcos: tipoarco;
          pad: tipopad;
          circulos: tipocirculo;
          rectangulo: Trectangulo;
          rectangulo_letra: Trectangulo;
          rectangulo_componente: Trectangulo;
          inicio_letrax: real;
          inicio_letray: real;
          angulo_letra: real;
          rect_localiza_matriz: Trect;
          matriz_pines: Tmatriz;
          vector_pines: tvector;
          Vector_pines_indx: Tvector;
          end;

 tipolista_rectangulo=^rectangulo_nodo;
 rectangulo_nodo=record
            rectangulo: Trect;
            prox: tipolista_rectangulo;
            end;
 Tfuncion= function (REx,Pr: real): real;
 file_atributos= file of Tatributos;
 file_letras= file of Tletra;

 function existe_letra(letra: char; puntero_letras: tipoletra;var puntero_salida: tipoletra ): boolean;
function hacer_lista_letras: tipoletra;
procedure construir_dynamic_vector(var valor: Tvector; elementos: integer);
procedure destruir_dynamic_vector(var valor: Tvector);
procedure construir_dynamic_matriz(var valor: Tmatriz; max_f,max_c: integer);
procedure destruir_dynamic_matriz(var valor: Tmatriz);
function in_pixel(const valor: real): integer;
function pixel_mil(const valor: real): real;
function pixel_mil_ajustado(const valor: real): integer;
function mil_pixel_ajustado(const valor: real): integer;
function punto_pixel_mil(const punto: Tpoint): Tpoint;
function mil_pixel(const valor: real): real;
function inmil_milime_ajustado(const valor: real): integer;
```

```
function inmil_milime(const valor: real): real;
function milime_inmil(const valor: real): real;
procedure calcular_temp_maxmin(var matriz: Tmatriz;var  temp_max, temp_min: real);
procedure poner_matriz_valor(var matriz: Tmatriz; valor: real;
inicio_fila,final_fila,inicio_columna,final_columna: integer);
procedure poner_vector_valor(var vector: Tvector; valor: real);
function isrectvacio(var rect_salida: Trect): boolean;
function  intersrect(var rect_salida: Trect;  rect1,rect2: trect): boolean;
function  intersrect2(var rect1,rect2: trect): boolean;
function puntorect(const rect: Trect; const punto: Tpoint): boolean;
function puntorect_especial(const rect: Trectangulo; const x,y: real): boolean;
function localizacion_punto(const rectangulo: Trectangulo;
const x,y, delta, escala: real; var punto: Tpoint): boolean;
function localizacion_matriz_punto(const rectangulo_cuadricula_escalado: Trect;
var punto: tpoint; delta: real; escala: real): boolean;
procedure centro_rect(var lista: tipolista;
var rectangulo_cuadricula: Trect; delta,escala: real);
procedure offsetrectangulo(var rect: Trectangulo;const x,y: real);
function detecta_atributos_lista(const componente: openstring; var atributos: Tatributos): boolean;
procedure calcular_parametros_fisicos( parametros_fisicos: Tpuntero_fisicos);
procedure calculo_posicion_potencia_malla(var puntero: tipolista; var matriz: Tmatriz;var
rect_localiza_matriz,rectangulo_cuadricula: Trect; delta,valor: real);
function obtener_rect(var puntero: tipolista): boolean;
function  intersrect2_especial(var rect1,rect2: Trectangulo): boolean;
function IntPow (Base: real;  NonNegInt: integer): real;

var
PixelsPerInch: integer;


implementation
uses padre;

procedure construir_dynamic_vector(var valor: Tvector; elementos: integer);
begin
with valor do  begin
NumElementos:=elementos;
getMem(vector, NumElementos * sizeof(real) );
end;
end;

procedure destruir_dynamic_vector(var valor: Tvector);
begin
with valor do
FreeMem(vector, NumElementos * sizeof(real) );
end;

procedure construir_dynamic_matriz(var valor: Tmatriz; max_f,max_c: integer);
var
n: integer;
begin
with valor do begin
max_fila:=max_f;
max_columna:=max_c;
getMem(matriz, max_fila * sizeof(PMyDynamicvector) );
for n:=1 to max_fila do
getMem(matriz^[n], max_columna * sizeof(real) );
activa:=true;
end;
end;
```

```
procedure construir_dynamic_matriz3D(var valor: Tmatriz3D; max_f,max_c,max_a: integer);
var
x,y: integer;
begin
with valor do begin
max_fila:=max_f;
max_columna:=max_c;
max_ancho:=max_a;
getMem(matriz, max_ancho * sizeof(PMyDynamicmatriz) );
for y:=1 to max_ancho do begin
getMem(matriz[y], max_fila * sizeof(PMyDynamicvector) );
      for x:=1 to max_fila do
      getMem(matriz^[y,x], max_columna * sizeof(real) );
end;
end;
end;

procedure destruir_dynamic_matriz(var valor: Tmatriz);
var
n: integer;
begin
with valor do
begin
for n:=1 to max_fila do
FreeMem(matriz^[n],  max_fila * sizeof(real) );
freeMem(matriz, max_columna * sizeof(PMyDynamicvector) );
activa:=false;
end;
end;

procedure destruir_dynamic_matriz3D(var valor: Tmatriz3D);
var
n,y: integer;
begin
with valor do
begin
for n:=1 to max_ancho do
 begin
   for y:=1 to max_fila do
      FreeMem(matriz[n,y],  max_fila * sizeof(real) );
   freeMem(matriz[n], max_columna * sizeof(PMyDynamicvector) );
 end;
   freeMem(matriz, max_ancho * sizeof(PMyDynamicmatriz) );
end;
end;

//comprueba que exista la letra en la librería de letras//

function existe_letra(letra: char; puntero_letras: tipoletra; var puntero_salida: tipoletra  ): boolean;
var
prueba: tipoletra;
begin
result:=false;
prueba:= puntero_letras;
while prueba <> nil do
begin
   if  prueba.letras.letra_char=letra then
   begin
      puntero_salida:=prueba;
```

```
        result:=true;
        exit;
    end;
prueba:=prueba.prox;
end;
end;

//con el archivo letra se hace una lista//

function hacer_lista_letras: tipoletra;
var
simbolo: Tletra;
tipo_letras: tipoletra;
begin
seek(archivo_letras, 0);
result:=nil;
    while not eof(archivo_letras) do
    begin
        read(archivo_letras,simbolo);
        new(tipo_letras);
        tipo_letras.letras:=simbolo;
        tipo_letras.prox:=result;
        result:=tipo_letras;
    end;
end;

//convierte pulgadas en pixeles//

function in_pixel(const valor: real): integer;
begin
result:=round(valor *  PixelsPerInch);
end;

//convierte pixeles en milésimas de pulgada//

function pixel_mil(const valor: real): real;
begin
result:= valor / PixelsPerInch * 1000;
end;

//convierte milésimas de pulgada apixeles//

function mil_pixel(const valor: real): real;
 begin
result:=valor / 1000 * PixelsPerInch;
end;

//convierte pixeles en milésimas de pulgada quitando decimales de la conversión//

function pixel_mil_ajustado(const valor: real): integer;
begin
result:=round(valor / PixelsPerInch * 1000);
end;

//convierte milésimas de pulgada en pixeles quitando decimales de la conversión//

function mil_pixel_ajustado(const valor: real): integer;
begin
result:=round(valor / 1000 * PixelsPerInch);
end;
```

```
function punto_pixel_mil(const punto: Tpoint): Tpoint;
begin
result.x:=round((punto.x/PixelsPerInch)*1000);
result.y:=round((punto.y/PixelsPerInch)*1000);
end;

//convierte milimetros a milésimas de pulgada  quitando decimales de la conversión//

function inmil_milime_ajustado(const valor: real): integer;
begin
result:=round((valor/1000)*25.4);
end;

//Convierte milimetros a milésimas de pulgada//

function milime_inmil(const valor: real): real;
begin
result:=(valor/25.4)*1000;
end;

//Convierte  milésimas de pulgada a milimetros//

function inmil_milime(const valor: real): real;
begin
result:=valor/1000 * 25.4;
end;

{Returns Base raised to the nonnegative integer power NonNegInt.}

function IntPow (Base: real;  NonNegInt: integer): real;
begin
  if NonNegInt = 0 then IntPow := 1
  else if odd (NonNegInt) then IntPow := Base * IntPow (Base, NonNegInt - 1)
  else IntPow := sqr (IntPow (Base, NonNegInt div 2));
end;

//Calcula la temperatura máxima y mínima de una malla//

procedure calcular_temp_maxmin(var matriz: Tmatriz;var  temp_max, temp_min: real);
var
fila,columna: integer;
valor: real;
x:integer;
begin
with matriz do
begin
    temp_max:=matriz[1,1];
    temp_min:=matriz[1,1];
     valor:=0;
     x:=0;
    for fila:=1 to max_fila do
       for columna:=1 to max_columna do
          begin
          if matriz[fila,columna]<>0 then  begin
              x:=x+1;
          valor:=valor+matriz[fila,columna];
          end;
          if temp_max < matriz^[fila,columna] then temp_max:=matriz^[fila,columna];
          if temp_min > matriz^[fila,columna] then temp_min:=matriz^[fila,columna];
```

```
          end;
end;
if x>0 then
   matriz.valor_medio:=valor/x;
end;

//Se pone parte de la malla a un cierto valor//

procedure poner_matriz_valor(var matriz: Tmatriz; valor: real;
inicio_fila,final_fila,inicio_columna,final_columna: integer);
var
fila,columna: integer;
begin
    for fila:=inicio_fila to final_fila do
       for columna:=inicio_columna to final_columna do
          matriz.matriz^[fila,columna]:=valor;
end;

//Poner un vector a un cierto valor//

procedure poner_vector_valor(var vector: Tvector; valor: real);
var
n: integer;
begin
    for n:=1 to vector.NumElementos do
       vector.vector^[n]:=valor;
 end;

 //Decteta si un rectángulo está vacio//

 function isrectvacio(var rect_salida: Trect): boolean;
var
aux: integer;
begin
    aux:=rect_salida.top;
    rect_salida.top:=rect_salida.bottom;
    rect_salida.bottom:=aux;
    if not isrectempty(rect_salida) then
      begin
          aux:=rect_salida.top;
          rect_salida.top:=rect_salida.bottom;
          rect_salida.bottom:=aux;
          result:=false;
       end
    else   result:=true;
end;

//Comprobar si exite intersección entre los rectángulos rect1 y rect2//

function  intersrect2(var rect1,rect2: trect): boolean;
begin
    result:=false;
    if  (rect1.top<rect2.bottom) or  (rect1.bottom>rect2.top) or
       (rect1.right<rect2.left) or  (rect1.left>rect2.right) then
       { exit}
     else
        result:=true;
end;

//Comprobar si exite intersección entre los rectángulos rect1 y rect2//
```

```
function  intersrect2_especial(var rect1,rect2: Trectangulo): boolean;
begin
    result:=false;
    if  (rect1.arriba<rect2.abajo) or  (rect1.abajo>rect2.arriba) or
       (rect1.derecha<rect2.izquierda) or  (rect1.izquierda>rect2.derecha) then

     else
        result:=true;
end;

 //Calcula el rectángulo  intersección de rec1 y rect2//

function  intersrect(var rect_salida: Trect;  rect1,rect2: trect): boolean;
var
aux: integer;
begin
    aux:=rect1.top;
    rect1.top:=rect1.bottom;
    rect1.bottom:=aux;
    aux:=rect2.top;
    rect2.top:=rect2.bottom;
    rect2.bottom:=aux;
    if intersectrect(rect_salida,rect1,rect2) then
      begin
          aux:=rect_salida.top;
          rect_salida.top:=rect_salida.bottom;
          rect_salida.bottom:=aux;
          result:=true;
       end
   else   result:=false;
end;

//Detecta si un punto está dentro de un rectángulo//

function puntorect(const rect: Trect; const punto: Tpoint): boolean;
begin
if (punto.y >= rect.bottom) and (punto.y <= rect.top) and (punto.x>=rect.left)
and (punto.x<= rect.right) then
   result:=true
else result:=false;
end;

//Detecta si un punto está dentro de un rectángulo//

function puntorect_especial(const rect: Trectangulo; const x,y: real): boolean;
begin
if (y >= rect.abajo) and (y <= rect.arriba) and (x>=rect.izquierda)
and (x<= rect.derecha) then
   result:=true
else result:=false;
end;

//Desplaza el rectángulo rect una cierta cantidad//

procedure offsetrectangulo(var rect: Trectangulo;const x,y: real);
begin
with rect do begin
    izquierda:=izquierda+x;
    derecha:=derecha+x;
```

```
    abajo:=abajo+y;
    arriba:=arriba+y;
end;
end;

//Obtenr la celda x,y de la malla que esta debajo de cada componente//

procedure centro_rect(var lista: tipolista;
var rectangulo_cuadricula: Trect; delta,escala: real);
var
puntero: tipolista;
rect_intersec,punto_extremo_componente: Trect;
x,y: real;
punto: tpoint;
rectangulo_esquema: trectangulo;
circulos: tipopad;
arriba,abajo,izquierda,derecha: real;
begin
puntero:=lista;
rectangulo_esquema.izquierda:=rectangulo_cuadricula.left;
rectangulo_esquema.derecha:=rectangulo_cuadricula.right;
rectangulo_esquema.abajo:=rectangulo_cuadricula.bottom;
rectangulo_esquema.arriba:=rectangulo_cuadricula.top;
while puntero <> nil do
 begin
     with  puntero.rectangulo_componente do
     if not (localizacion_punto(rectangulo_esquema,
     izquierda*1000,arriba*1000,delta, escala,puntero.atributos.cell) and
     localizacion_punto(rectangulo_esquema,
     derecha*1000,abajo*1000,delta, escala,puntero.atributos.cell)) or
     (puntero.atributos.activo=false) then begin
     puntero.atributos.cell.x:=0;
     puntero.atributos.cell.y:=0;
     end
     else begin
     x:=(puntero.rectangulo_componente.izquierda+(puntero.rectangulo_componente.derecha-
puntero.rectangulo_componente.izquierda)/2)*1000;
     y:=(puntero.rectangulo_componente.abajo+(puntero.rectangulo_componente.arriba-
puntero.rectangulo_componente.abajo)/2)*1000;
     localizacion_punto(rectangulo_esquema,x,y,delta, escala,puntero.atributos.cell);
localizacion_punto(rectangulo_esquema,puntero.rectangulo_componente.izquierda*1000,puntero.rectang
ulo_componente.arriba*1000,delta,escala ,puntero.rect_localiza_matriz.topleft);
localizacion_punto(rectangulo_esquema,puntero.rectangulo_componente.derecha*1000,puntero.rectangul
o_componente.abajo*1000,delta, escala,puntero.rect_localiza_matriz.bottomright);
     circulos:=puntero.pad;
     while circulos<> nil do begin
localizacion_punto(rectangulo_esquema,circulos.rectangulo_exterior.izquierda*1000,circulos.rectangulo_
exterior.arriba*1000,delta,escala ,circulos.rect_localiza_pin_matriz.topleft);

localizacion_punto(rectangulo_esquema,circulos.rectangulo_exterior.derecha*1000,circulos.rectangulo_e
xterior.abajo*1000,delta, escala,circulos.rect_localiza_pin_matriz.bottomright);
         x:=(circulos.rectangulo_exterior.izquierda+(circulos.rectangulo_exterior.derecha-
circulos.rectangulo_exterior.izquierda)/2)*1000;
         y:=(circulos.rectangulo_exterior.abajo+(circulos.rectangulo_exterior.arriba-
circulos.rectangulo_exterior.abajo)/2)*1000;
         localizacion_punto(rectangulo_esquema,x,y,delta, escala,circulos.cell);
         circulos:=circulos.prox;
     end;
     end;
puntero:=puntero.prox;
```

```
  end;
end;


{ Dado la coordenada x,y de la pantalla obtener celda x,y de la malla
que alverga esta coordenada }

function localizacion_punto(const rectangulo: Trectangulo;
const x,y, delta, escala: real; var punto: Tpoint): boolean;
var
delta_escalada: real;
begin
result:=false;
if puntorect_especial(rectangulo,x,y) then begin
    delta_escalada:=delta*escala;
    punto.x:=round((x-rectangulo.izquierda)/delta_escalada)+1;
    punto.y:=round((y-rectangulo.abajo)/delta_escalada)+1;
    result:=true;
end;
end;

{ Dado la celda x,y de la malla obtener posición x,y de la pantalla
que esta dentro de esta celda }

function localizacion_matriz_punto(const rectangulo_cuadricula_escalado: Trect;
var punto: tpoint; delta: real; escala: real): boolean;
var
delta_escalada: real;
begin
result:=false;
if puntorect(rectangulo_cuadricula_escalado,punto) then begin
    delta_escalada:=delta*escala;
    punto.x:=round((punto.x-rectangulo_cuadricula_escalado.left)/delta_escalada)+1;
    punto.y:=round((punto.y-rectangulo_cuadricula_escalado.bottom)/delta_escalada)+1;
    result:=true;
end
end;

//Dectecta si el componente está dentro de la lista atributos//

function detecta_atributos_lista(const componente: openstring; var atributos: Tatributos): boolean;
 var
prueba: Tipoatributos;
aux_atributos: tatributos;
 begin
 result:=false;
 aux_atributos:=atributos;
 prueba:=lista_atributos;
 while (prueba <> nil) do begin
 if prueba.atributos.componente=componente then begin
 atributos:=prueba.atributos;
 result:=true;
      with atributos do
      begin
      referencia:=aux_atributos.referencia;
      tipo:=aux_atributos.tipo;
      numero_pines:=aux_atributos.numero_pines;
      end;
 exit;
 end;
```

```
 prueba:=prueba.prox;
 end;
 end;

//Calcular parametros necesesarios para la simulación//

procedure calcular_parametros_fisicos(parametros_fisicos: Tpuntero_fisicos);
VAR

ALFA: REAL;
begin
with parametros_fisicos^ do begin
     // PARAMETROS DE LOS COMPONENTES //
     ALFA:=(P*C);
     DT:= ALFA/(4*K/SQR(DELTA)+4*
h_minimo_borde/delta+h_minimo_arriba/ESPESOR+h_minimo_abajo/ESPESOR);
 RADIO1:=K * DT /( P * C * SQR(DELTA));
 RADIO2:=   DT /( P * C * ESPESOR);
 RADIO3:=   DT /( P * C * DELTA);
END;
end;

//Poner  potencia en la malla limita por el rectangulo que forma el pad con la malla//

procedure calculo_posicion_potencia_malla(var puntero: tipolista; var matriz: Tmatriz;var
rect_localiza_matriz,rectangulo_cuadricula: Trect; delta,valor: real);
var
columna,fila: integer;
valor_escala: real;
begin
    if puntero.atributos.cell.x<>0 then begin
       for  columna:=rect_localiza_matriz.left to rect_localiza_matriz.right do
         for fila:=rect_localiza_matriz.bottom to rect_localiza_matriz.top do
          matriz.matriz[fila,columna]:=valor;
    end;
end;

// Obtener el rectanguo que limita la estructura del componente//

function obtener_rect(var puntero: tipolista): boolean;
var
i,n: integer;
recta: tiporecta;
x,y: real;
begin
obtener_rect:=true;
recta:=puntero.estructura.recta;
if puntero.estructura.recta<> nil then begin
puntero.rectangulo_componente.izquierda:=PUNTERO.estructura.recta.x1;
puntero.rectangulo_componente.derecha:=puntero.estructura.recta.x1;
puntero.rectangulo_componente.arriba:=puntero.estructura.recta.Y1;
puntero.rectangulo_componente.abajo:=puntero.estructura.recta.Y1;
end
else begin
obtener_rect:=false;
exit;
end;
with puntero.rectangulo_letra do
   if  (izquierda=0) and (derecha=0) and (arriba=0) and (abajo=0) then
      puntero.rectangulo:=puntero.rectangulo_componente
```

```
    else
       puntero.rectangulo:=puntero.rectangulo_letra;
        while recta <> nil do begin
              for n:=1 to 2 do begin
                if n=1 then begin
                x:=recta.x1;
                y:=recta.y1;
                end
                else begin
                x:=recta.x2;
                y:=recta.y2;
                end;
               if puntero.rectangulo_componente.izquierda>x then
                 puntero.rectangulo_componente.izquierda:=x
               else
               if puntero.rectangulo_componente.derecha<x then
                 puntero.rectangulo_componente.derecha:=x;
               if puntero.rectangulo_componente.arriba<y then
                puntero.rectangulo_componente.arriba:=y
               else
               if  puntero.rectangulo_componente.abajo>y then
                    puntero.rectangulo_componente.abajo:=y;
               if puntero.rectangulo.izquierda>x then
                  puntero.rectangulo.izquierda:=x
               else
               if puntero.rectangulo.derecha<x then
                 puntero.rectangulo.derecha:=x;
               if puntero.rectangulo.arriba<y then
               puntero.rectangulo.arriba:=y
               else
               if  puntero.rectangulo.abajo>y then
                puntero.rectangulo.abajo:=y;
           end;
        recta:=recta.prox;
end;      end;

end.
```

```
unit unidad;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, Buttons, tipo1;

type
  TFunidad = class(TForm)
    Evalor: TEdit;
    Label1: TLabel;
    BitBtn1: TBitBtn;
    BitBtn3: TBitBtn;
    RGunidad: TRadioGroup;
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    NewString_puntero: ^string;
    estilo_unidades_puntero: ^Testilo_unidades;
    { Private declarations }
  public
procedure  entrada(var NewString: string; var estilo_unidades: Testilo_unidades);
    { Public declarations }
  end;

var
  Funidad: TFunidad;

implementation

{$R *.DFM}

//introduce unidad y tipo unidad (milésima de pulagada o miliemtros//

procedure  TFunidad.entrada(var NewString: string; var  estilo_unidades: Testilo_unidades);
begin
    NewString_puntero:=@NewString;
    estilo_unidades_puntero:=@estilo_unidades;
    case estilo_unidades of
    EUpulgadas: RGunidad.itemindex:=0;
    EUmilimetros: begin
                RGunidad.itemindex:=1;
                end;
     end;
     Evalor.text:=NewString;
end;

//Detecta tipo de unidad//

procedure TFunidad.BitBtn3Click(Sender: TObject);
begin
NewString_puntero^:=Evalor.text;
case rgunidad.itemindex of
0: estilo_unidades_puntero^:=EUpulgadas;
1:  estilo_unidades_puntero^:=EUmilimetros;
end;
modalresult:=mrok;
end;
```

```
procedure TFunidad.BitBtn1Click(Sender: TObject);
begin
modalresult:=mrcancel
end;
end.
```

unit Usacar_datos_atributos;

```
interface
uses
tipo1,Classes,SysUtils,Dialogs;

type
Tcadena_128=string[128];

procedure poner_archivo_atributos(var atributos: Tatributos; var archivo: Tstringlist;
digitos: integer; forma_copia: Testilo_modificar_componente);
procedure sacar_cadena_atributos(var valor: real; const cadena: Tcadena_128);
procedure  rellenar_lista_atributos(var lista: tipolista; var archivo: Tstringlist;
const fisicos: Tpuntero_fisicos);
procedure hacer_lista_atributos(var lista: Tipoatributos; var archivo: Tstringlist);
procedure salvar_archivo_atributos_esquema(var archivo,archivo_cabecera: Tstringlist; lista: tipolista;
var fisicos:  Tpuntero_fisicos );
procedure poner_datos_fisicos(var archivo,archivo_cabecera: Tstringlist; var fisicos: Tpuntero_fisicos;
DIGITOS: INTEGER);
procedure sacar_datos_fisicos(var archivo: Tstringlist;const fisicos: Tpuntero_fisicos);
procedure  compente_fuera(var atributos: Tatributos; var archivo: Tstringlist; base: integer);
procedure sacar_atributos_lista(var atributos: Tatributos; var archivo: Tstringlist;
digitos: integer);

implementation

procedure sacar_cadena_atributos(var valor: real; const cadena: Tcadena_128);
var
cadena_aux: string[125];
begin
cadena_aux:=copy(cadena, pos('*', cadena)+1, length(cadena));
valor:=strTOfloat(cadena_aux);
end;

{Hace la lista atributos la cual contendra todos los datos de los componentes
procedente del archivo asociado a la placa}

procedure hacer_lista_atributos(var lista: Tipoatributos; var archivo: Tstringlist);
var
n,base: integer;
aux_lista: tipoatributos;
cadena: string[125];
begin
base:=archivo.IndexOf('INICIO_DATA_COMPONENTES')+1;
while base<=archivo.count-1 do
if  archivo.Strings[base]='_COMPONENTE' THEN
BEGIN
        try
        inc(base);
        new(aux_lista);
        aux_lista.prox:=lista;
        lista:=aux_lista;
        compente_fuera(lista.atributos,archivo,base);
        base:=base+20;
        lista.atributos.activo:=true;
        Except On Exception Do { si  falla }
         ShowMessage('archivo esta dañado o falta de memoria');
        end;
END
ELSE inc(base);
```

```
 end;

 {Guardo los datos de los componentes del archivo asociado a la placa
 archivo que ya existe es decir, que se creo con anterioridad}

 procedure poner_archivo_atributos(var atributos: Tatributos; var archivo: Tstringlist;
 digitos: integer; forma_copia: Testilo_modificar_componente);
 var base,n: integer;
 begin
  base:=archivo.IndexOf(atributos.componente);
  if base<>-1  then
  base:=base-1
  else begin
  archivo.delete(archivo.count-1);
  base:=archivo.count;
  for n:=1 to 24 do
  archivo.add('');
  archivo.Strings[ARCHIVO.COUNT-1]:=('FIN_DATA_COMPONENTES');
 end;
    with atributos do begin
     //se construye el formato que tendrán el archivo que esta asociado a la placa//
     archivo.Strings[base]:=('_COMPONENTE');
     archivo.Strings[base+1]:=componente;
     archivo.Strings[base+2]:=('_REFERENCIA');
     archivo.Strings[base+3]:=referencia;
     archivo.Strings[base+4]:=('_POTENCIA (W)
*'+floattostrF(atributos.potencia,ffFixed,8,digitos));
     archivo.Strings[base+5]:=('_Tj (ºC)          *'+floattostrF(atributos.tj,ffFixed,8,digitos));
     archivo.Strings[base+6]:=('_Tc (ºC)          *'+floattostrF(atributos.tc,ffFixed,8,digitos));
     archivo.Strings[base+7]:=('_Tp (ºC)          *'+floattostrF(atributos.tp,ffFixed,8,digitos));
     archivo.Strings[base+8]:=('_Q(W/m²)          *'+floattostrF(atributos.calor,ffFixed,8,digitos));
     archivo.Strings[base+9]:=('_K (W/mºC)
*'+floattostrF(atributos.conductividad,ffFixed,8,digitos));
     archivo.Strings[base+10]:=('_RTHJC (W/m)     *'+floattostrF(atributos.rthjc,ffFixed,8,digitos));
     archivo.Strings[base+11]:=('_RTHCA (W/m)     *'+floattostrF(atributos.rthCA,ffFixed,8,digitos));
     archivo.Strings[base+12]:=('_RTHCP (W/m)     *'+floattostrF(atributos.rthcp,ffFixed,8,digitos));
     archivo.Strings[base+13]:=('_RTHP (W/m)      *'+floattostrF(atributos.rthp,ffFixed,8,digitos));
     archivo.Strings[base+14]:=('_LARGO (m)       *'+floattostrF(atributos.LARGO,ffFixed,8,digitos));
     archivo.Strings[base+15]:=('_ANCHO (m)       *'+floattostrF(atributos.ancho,ffFixed,8,digitos));
     archivo.Strings[base+16]:=('_ESPESOR (m)     *'+floattostrF(atributos.espesor,ffFixed,8,digitos));
     archivo.Strings[base+17]:=('_RTHPIN (W/m)    *'+floattostrF(atributos.rthpin,ffFixed,8,digitos));
     archivo.Strings[base+18]:=('_ESPESOR_PIN (m)
*'+floattostrF(atributos.espesorpin,ffFixed,8,digitos));
     archivo.Strings[base+19]:=('_LARGO_PIN (m)
*'+floattostrF(atributos.LARGOpin,ffFixed,8,digitos));
     archivo.Strings[base+20]:=('_ANCHO_PIN (m)
*'+floattostrF(atributos.anchopin,ffFixed,8,digitos));
     archivo.Strings[base+21]:=('_CONDUCTIVIDAD_PIN
*'+floattostrF(atributos.conductividadpin,ffFixed,8,digitos));
     archivo.Strings[base+22]:=('FIN_COMPONENTE');
     END;
 archivo.saveTOfile(path_atributos);
 end;

 {Guarda los datos de los componentes del archivo asociado a la placa
 por primeravez}

 procedure sacar_atributos_lista(var atributos: Tatributos; var archivo: Tstringlist;
 digitos: integer);
 begin
```

```
   with atributos do begin
    //se construye el formato que tendrán  archivo //
    archivo.add('_COMPONENTE');
    archivo.add(componente);
    archivo.add('_REFERENCIA');
    archivo.add(referencia);
    archivo.add('_POTENCIA (W)        *'+floattostrF(atributos.potencia,ffFixed,8,digitos));
    archivo.add('_Tj (ºC)            *'+floattostrF(atributos.tj,ffFixed,8,digitos));
    archivo.add('_Tc (ºC)             *'+floattostrF(atributos.tc,ffFixed,8,digitos));
    archivo.add('_Tp (ºC)             *'+floattostrF(atributos.tp,ffFixed,8,digitos));
    archivo.add('_Q(W/m²)              *'+floattostrF(atributos.calor,ffFixed,8,digitos));
    archivo.add('_K (W/mºC)            *'+floattostrF(atributos.conductividad,ffFixed,8,digitos));
    archivo.add('_RTHJC (W/m)           *'+floattostrF(atributos.rthjc,ffFixed,8,digitos));
    archivo.add('_RTHCA (W/m)           *'+floattostrF(atributos.rthCA,ffFixed,8,digitos));
    archivo.add('_RTHCP (W/m)           *'+floattostrF(atributos.rthcp,ffFixed,8,digitos));
    archivo.add('_RTHP (W/m)           *'+floattostrF(atributos.rthp,ffFixed,8,digitos));
    archivo.add('_LARGO (m)            *'+floattostrF(atributos.LARGO,ffFixed,8,digitos));
    archivo.add('_ANCHO (m)            *'+floattostrF(atributos.ancho,ffFixed,8,digitos));
    archivo.add('_ESPESOR (m)           *'+floattostrF(atributos.espesor,ffFixed,8,digitos));
    archivo.add('_RTHPIN (W/m)         *'+floattostrF(atributos.rthpin,ffFixed,8,digitos));
    archivo.add('_ESPESOR_PIN (m)       *'+floattostrF(atributos.espesorpin,ffFixed,8,digitos));
    archivo.add('_LARGO_PIN (m)        *'+floattostrF(atributos.LARGOpin,ffFixed,8,digitos));
    archivo.add('_ANCHO_PIN (m)        *'+floattostrF(atributos.anchopin,ffFixed,8,digitos));
    archivo.add('_CONDUCTIVIDAD_PIN (W/m
ºC)*'+floattostrF(atributos.conductividadpin,ffFixed,8,digitos));
    archivo.add('FIN_COMPONENTE');
    END;
end;


//Añade al archivo los datos físicos de la placa y medio ambiente}

procedure poner_datos_fisicos(var archivo,archivo_cabecera: Tstringlist; var fisicos: Tpuntero_fisicos;
DIGITOS: INTEGER);
var
base,n: integer;
begin
    archivo.clear;
    for n:=0 to archivo_cabecera.count-1 do
    archivo.add(archivo_cabecera.Strings[n]);
    base:=archivo.IndexOf('INICIO_DATA_PLACA');
    archivo.Strings[base+2]:=floattostrF(fisicos.u,ffFixed,8,digitos);
    archivo.Strings[base+4]:=floattostrF(fisicos.k,ffFixed,8,digitos);
    archivo.Strings[base+6]:=floattostrF(fisicos.p,ffFixed,8,digitos);
    archivo.Strings[base+8]:=floattostrF(fisicos.c,ffFixed,8,digitos);
    archivo.Strings[base+10]:=floattostrF(fisicos.H_arriba,ffFixed,8,digitos);
    archivo.Strings[base+12]:=floattostrF(fisicos.h_abajo,ffFixed,8,digitos);
    archivo.Strings[base+14]:=floattostrF(fisicos.espesor,ffFixed,8,digitos);
    archivo.Strings[base+16]:=floattostrF(fisicos.delta,ffFixed,8,digitos);
end;

//Sacar los datos físicos del archivo asociado a  la placa//

procedure sacar_datos_fisicos(var archivo: Tstringlist;const fisicos: Tpuntero_fisicos);
var
base: integer;
begin
    base:=archivo.IndexOf('INICIO_DATA_PLACA');
    fisicos.u:=strtofloat( archivo.Strings[base+2]);
    fisicos.k:=strtofloat( archivo.Strings[base+4]);
```

```
    fisicos.p:=strtofloat( archivo.Strings[base+6]);
    fisicos.c:=strtofloat( archivo.Strings[base+8]);
    fisicos.h_arriba:=strtofloat( archivo.Strings[base+10]);
    fisicos.h_abajo:=strtofloat( archivo.Strings[base+12]);
    fisicos.espesor:=strtofloat( archivo.Strings[base+14]);
    fisicos.delta:=strtofloat( archivo.Strings[base+16]);
end;

{rellena la lista atributos (lista de los parametros de los componente)
bien del archivo asociado a la placa si existe, de lo contraria de la librería
(archivo llamado atributos.txt}

procedure  rellenar_lista_atributos(var lista: tipolista; var archivo: Tstringlist;
const fisicos: Tpuntero_fisicos);
var
n,base: integer;
cadena: string;
puntero: tipolista;
begin
sacar_datos_fisicos(archivo,fisicos);
puntero:=lista;
while puntero<>nil do begin
    base:=archivo.IndexOf(puntero.atributos.referencia);
    if base<>-1 then begin
    try
    compente_fuera(puntero.atributos,archivo,base-2);
    base:=base+20;
     Except On Exception Do { si  falla }
        ShowMessage('archivo está dañado');
    end;
    end
    else  //si no esta en el archivo buscalo en la lista//
    detecta_atributos_lista(puntero.atributos.componente,puntero.atributos);
puntero:=puntero.prox;
end;
END;

//lee la lista de los componentes para salvarlos//

procedure salvar_archivo_atributos_esquema(var archivo,archivo_cabecera: Tstringlist; lista: tipolista;
var fisicos:  Tpuntero_fisicos );
var
prueba: tipolista;
begin
    poner_datos_fisicos(archivo,archivo_cabecera, fisicos,4);
    prueba:=lista;
  while prueba<>nil do begin
       sacar_atributos_lista(prueba.atributos,archivo,4);
       prueba:=prueba.prox;
   end;
archivo.add('FIN_DATA_COMPONENTES');
end;

//Salva los datos del los componentes al archivo asociado con la placa//

procedure  compente_fuera(var atributos: Tatributos; var archivo: Tstringlist; base: integer);
begin
       with atributos do begin
       componente:=archivo.Strings[base];
       referencia:=archivo.Strings[base+2];
```

```
        sacar_cadena_atributos(potencia, archivo.Strings[base+3]);
        sacar_cadena_atributos(Tj, archivo.Strings[base+4]);
        sacar_cadena_atributos(Tc, archivo.Strings[base+5]);
        sacar_cadena_atributos(Tp, archivo.Strings[base+6]);
        sacar_cadena_atributos(calor, archivo.Strings[base+7]);
        sacar_cadena_atributos(conductividad,archivo.Strings[base+8]);
        sacar_cadena_atributos(rthjc,archivo.Strings[base+9]);
        sacar_cadena_atributos(rthca, archivo.Strings[base+10]);
        sacar_cadena_atributos(rthcp, archivo.Strings[base+11]);
        sacar_cadena_atributos(rthp, archivo.Strings[base+12]);
        sacar_cadena_atributos(largo, archivo.Strings[base+13]);
        sacar_cadena_atributos(ancho, archivo.Strings[base+14]);
        sacar_cadena_atributos(espesor, archivo.Strings[base+15]);
        sacar_cadena_atributos(rthpin, archivo.Strings[base+16]);
        sacar_cadena_atributos(espesorpin, archivo.Strings[base+17]);
        sacar_cadena_atributos(largopin, archivo.Strings[base+18]);
        sacar_cadena_atributos(anchopin, archivo.Strings[base+19]);
        sacar_cadena_atributos(conductividadpin, archivo.Strings[base+20]);
        end;
END;

end.
```

```
unit Usacar_datos_protel;

interface
uses
   tipo1,SysUtils;
    procedure sacar_letra(var coordenadas: openstring; var puntero: tipolista);
    procedure sacar_rectas_componente(var coordenadas: openstring;
        var puntero: tipolista);
    procedure sacar_arco(var coordenadas: openstring; var puntero: tipolista);
    procedure poner_Arco(var puntero: tipoarco);
    procedure extraer_datos_protel(var archivo_protel: text; var lista: tipolista;
    var lista_rectas: tiporecta);
    procedure poner_datos_protel(var archivo_protel: text; var lista: tipolista);
    procedure sacar_pads(var coordenadas: openstring; var puntero: tipolista);

implementation
uses
  Fesquema;

{ Saca datos de las rectas de la placa a la lista general del programa}

procedure sacar_rectas_placa(var coordenadas: openstring;
var puntero: tiporecta);
var
contador: integer;
begin
coordenadas:=copy(coordenadas, pos('0 0', coordenadas)+3, length(coordenadas));
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.x1:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.y1:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.x2:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.y2:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.ancho:=strToint(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
end;

{ Saca datos de las rectas del componente a la lista general del programa
asociado al componente actual del archivo}

procedure sacar_rectas_componente(var coordenadas: openstring;
        var puntero: tipolista);
var
contador: integer;
begin
coordenadas:=copy(coordenadas, pos('0 0', coordenadas)+3, length(coordenadas));
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.estructura.recta.x1:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.estructura.recta.y1:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.estructura.recta.x2:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.estructura.recta.y2:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.estructura.ancho:=strToint(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
end;
```

```
{ Saca datos de las letras del componente a la lista general del programa
asociado al componente actual del archivo}

procedure sacar_letra(var coordenadas: openstring; var puntero: tipolista);
var
contador: integer;
cadena: cadena30;
begin
coordenadas:=copy(coordenadas, pos('0 0 ', coordenadas)+4, length(coordenadas));
puntero.inicio_letrax:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.inicio_letray:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
coordenadas[pos('.',coordenadas)]:=',';
puntero.angulo_letra:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1));
puntero.angulo_letra:=puntero.angulo_letra*pi/180;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
delete(coordenadas,1, 6 );
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.rectangulo_letra.izquierda:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.rectangulo_letra.abajo:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.rectangulo_letra.derecha:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
puntero.rectangulo_letra.arriba:=strTofloat(coordenadas)/1000000;
end;

{ Saca datos de los pads del componente a la lista general del programa
asociado al componente actual del archivo}

procedure sacar_pads(var coordenadas: openstring; var puntero: tipolista);
var
tipo_circulo: tipopad;
centro_ejex,centro_ejey,diametro_ancho,diametro_alto,diametro_interior: real;
begin
new(tipo_circulo);
tipo_circulo.prox:=puntero.pad;
puntero.pad:=tipo_circulo;
inc(puntero.atributos.numero_pines);
with puntero.pad^ do
begin
coordenadas:=copy(coordenadas, pos('0 0 ', coordenadas)+4, length(coordenadas));
centro_ejex:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
centro_ejey:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
diametro_ancho:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
diametro_alto:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
diametro_interior:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
 rectangulo_exterior.izquierda:=centro_ejex-diametro_ancho/2;
 rectangulo_exterior.derecha:=centro_ejex+diametro_ancho/2;
 rectangulo_exterior.abajo:=centro_ejey-diametro_alto/2;
 rectangulo_exterior.arriba:=centro_ejey+diametro_alto/2;
 rectangulo_interior.izquierda:=centro_ejex-diametro_interior/2;
 rectangulo_interior.derecha:=centro_ejex+diametro_interior/2;
```

```
 rectangulo_interior.abajo:=centro_ejey-diametro_interior/2;
 rectangulo_interior.arriba:=centro_ejey+diametro_interior/2;
end;
puntero.atributos.anchopin:=inmil_milime(diametro_ancho*1000/2)*1000;
puntero.atributos.espesorpin:=inmil_milime(diametro_alto*1000/2)*1000;
end;

{ Saca datos de los pads del componente a la lista general del programa,
asociado al componente actual del archivo}

procedure sacar_arco(var coordenadas: openstring; var puntero: tipolista);
var
tipo_arco: tipoarco;
begin
new(tipo_arco);
tipo_arco.prox:=puntero.arcos;
puntero.arcos:=tipo_arco;
with tipo_arco.arco do
begin
coordenadas:=copy(coordenadas, pos('0 0 ', coordenadas)+4, length(coordenadas));
centro_ejex:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
centro_ejey:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
radio:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
coordenadas[pos('.',coordenadas)]:=',';
angulo_1:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1));
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
coordenadas[pos('.',coordenadas)]:=',';
angulo_2:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1));
coordenadas:=copy(coordenadas, pos(' ', coordenadas)+1, length(coordenadas));
ancho:=strTofloat(copy(coordenadas, 0, pos(' ', coordenadas)-1))/1000000;
end;
poner_arco(tipo_arco);
end;

{ Saca datos de los arcos del componente a la lista general del programa,
asociado al componente actual del archivo}

procedure poner_Arco(var puntero: tipoarco);
var
angulo_radianes: real;
y: integer;
begin
with puntero.arco do
begin
   rectangulo.izquierda:=centro_ejex-radio;
   rectangulo.derecha:=centro_ejex+radio;
   rectangulo.arriba:=centro_ejey+radio;
   rectangulo.abajo:=centro_ejey-radio;

       angulo_radianes:=angulo_1*pi/180;
       inicio_arcox:=cos(angulo_radianes)* radio+centro_ejex;
       inicio_arcoy:=sin(angulo_radianes)* radio+centro_ejey;

       angulo_radianes:=angulo_2*pi/180;
       fin_arcox:=cos(angulo_radianes)* radio+centro_ejex;
       fin_arcoy:=sin(angulo_radianes)* radio+centro_ejey;
end;
```

```
end;


// Detecta los codigos del archivo protel//

procedure extraer_datos_protel(var archivo_protel: text; var lista: tipolista;
var lista_rectas: tiporecta);
var
cadena,coordenadas: string[125];
aux_lista: Tipolista;
aux_recta: tiporecta;
salir:  boolean;
aux_lista_rectas: tiporecta;
begin
readln(archivo_protel,cadena);
salir:=false;
while (cadena<>'DEFAULTS')  and (not salir)  do
 begin
 if cadena='FT' then begin
 new(aux_lista_rectas);
 aux_lista_rectas.prox:=lista_rectas;
 lista_rectas:=aux_lista_rectas;
 readln(archivo_protel,cadena);
 sacar_rectas_placa(cadena,lista_rectas);
 end
 else
    if cadena='COMP' then
    begin
        new(aux_lista);
        aux_lista.atributos.numero_pines:=0;
        aux_lista.letras_referencia:=nil;
        aux_lista.arcos:=nil;
        aux_lista.circulos:=nil;
        aux_lista.estructura.recta:=nil;
        aux_lista.pad:=nil;
        aux_lista.prox:=lista;
        lista:=aux_lista;
        lista.enabled:=true;
        lista.atributos.activo:=true;
        readln(archivo_protel,lista.atributos.componente);
        readln(archivo_protel,cadena);
        readln(archivo_protel,cadena);
        readln(archivo_protel,cadena);
        sacar_letra(cadena, lista);
        readln(archivo_protel,cadena);
        lista.atributos.Referencia:=cadena;
        readln(archivo_protel,cadena);
        readln(archivo_protel,cadena);
        readln(archivo_protel,cadena);
        lista.atributos.tipo:=cadena;
         while cadena<> 'ENDCOMP' do
         begin
            if cadena='CT' then
            begin
                readln(archivo_protel,cadena);
                new(aux_recta);
                aux_recta.prox:=lista.estructura.recta;
                lista.estructura.recta:=aux_recta;
                sacar_rectas_componente(cadena,lista);
                readln(archivo_protel,cadena);
```

```
            end
            else
            begin
               if cadena='CA' then
            begin
               readln(archivo_protel,cadena);
               sacar_arco(cadena,lista);
               readln(archivo_protel,cadena);
            end
            else
                  if cadena='CP' then
            begin
               readln(archivo_protel,cadena);
               sacar_pads(cadena,lista);
               readln(archivo_protel,cadena);
            end
            else
               readln(archivo_protel,cadena);
            end;
         end;

construir_dynamic_matriz(aux_lista.matriz_pines,aux_lista.atributos.numero_pines,aux_lista.atributos.nu
mero_pines);
         construir_dynamic_vector(aux_lista.vector_pines_indx,aux_lista.atributos.numero_pines);
         construir_dynamic_vector(aux_lista.vector_pines,aux_lista.atributos.numero_pines);
         salir:=esquema.introducir_datos(aux_lista);
      end
      else
         readln(archivo_protel,cadena);
end;
end;

end.
```

unit Usacar_datos_ULPGC;

```
interface
uses
   tipo1,SysUtils;
const

    FACTOR_DIVISION=1000;

  procedure sacar_pads(var coordenadas: openstring; var puntero: tipolista);
  procedure sacar_rectas(var coordenadas: openstring;var puntero: tipolista);
  procedure sacar_arco(var coordenadas: openstring; var puntero: tipolista);
  procedure extraer_datos_ULPGC(var archivo_ULPGC: text; var lista: tipolista);
  procedure poner_datos_ULPGC(var archivo_ULPGC: text; var lista: tipolista);

implementation
uses Fesquema;

{ Saca los datos de las rectas del componente a la lista general del programa,
asociado al componente actual del archivo}

procedure sacar_rectas(var coordenadas: openstring;var puntero: tipolista);
VAR
aux_recta: tiporecta;
begin
new(aux_recta);
aux_recta.prox:=puntero.estructura.recta;
puntero.estructura.recta:=aux_recta;
puntero.estructura.recta.x1:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-
1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
puntero.estructura.recta.y1:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-
1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
puntero.estructura.recta.x2:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-
1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
puntero.estructura.recta.y2:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-
1))/FACTOR_DIVISION;
puntero.estructura.ancho:=0.006;
end;

{ Saca los datos de los arcos del componente a la lista general del programa,
asociado al componente actual del archivo}

procedure sacar_arco(var coordenadas: openstring; var puntero: tipolista);
var
tipo_arco: tipoarco;
begin
new(tipo_arco);
tipo_arco.prox:=puntero.arcos;
puntero.arcos:=tipo_arco;
with tipo_arco.arco do begin
centro_ejex:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
centro_ejey:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
inicio_arcox:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
inicio_arcoy:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
```

```
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
fin_arcox:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
fin_arcoy:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
ancho:=0.006;
radio:=abs((fin_arcoy-inicio_arcoy)/2);
rectangulo.izquierda:=centro_ejex-radio;
rectangulo.derecha:=centro_ejex+radio;
rectangulo.arriba:=centro_ejey+radio;
rectangulo.abajo:=centro_ejey-radio;
end;
end;

{ Saca los datos de los pads del componente a la lista general del programa,
asociado al componente actual del archivo}

procedure sacar_pads(var coordenadas: openstring; var puntero: tipolista);
var
tipo_pad: tipopad;
centro_ejex,centro_ejey,diametro_ancho,diametro_alto,diametro_interior: real;
begin
new(tipo_pad);
tipo_pad.prox:=puntero.pad;
puntero.pad:=tipo_pad;
inc(puntero.atributos.numero_pines);
with puntero.pad^ do
begin
centro_ejex:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
centro_ejey:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
diametro_ancho:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
diametro_alto:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
diametro_interior:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
 rectangulo_exterior.izquierda:=centro_ejex-diametro_ancho/2;
 rectangulo_exterior.derecha:=centro_ejex+diametro_ancho/2;
 rectangulo_exterior.abajo:=centro_ejey-diametro_alto/2;
 rectangulo_exterior.arriba:=centro_ejey+diametro_alto/2;
 rectangulo_interior.izquierda:=centro_ejex-diametro_interior/2;
 rectangulo_interior.derecha:=centro_ejex+diametro_interior/2;
 rectangulo_interior.abajo:=centro_ejey-diametro_interior/2;
 rectangulo_interior.arriba:=centro_ejey+diametro_interior/2;
end;
end;

{ Saca los datos de los circulos del componente a la lista general del programa,
asociado al componente actual del archivo}

procedure sacar_circulos(var coordenadas: openstring; var puntero: tipolista);
var
tipo_circulo: tipocirculo;
centro_ejex,centro_ejey,radio: real;
begin
new(tipo_circulo);
tipo_circulo.prox:=puntero.circulos;
puntero.circulos:=tipo_circulo;
centro_ejex:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
```

```
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
centro_ejey:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
coordenadas:=copy(coordenadas, pos(',', coordenadas)+1, length(coordenadas));
radio:=strTofloat(copy(coordenadas, 0, pos('.', coordenadas)-1))/FACTOR_DIVISION;
with puntero.circulos^ do
begin
rectangulo.izquierda:=centro_ejex-radio;
rectangulo.derecha:=centro_ejex+radio;
rectangulo.abajo:=centro_ejey-radio;
rectangulo.arriba:=centro_ejey+radio;
end;
end;

{ Saca los datos de los rectágulos de las letras del componente a la lista general del programa,
asociado al componente actual del archivo}

procedure sacar_rectangulo_letra(var puntero: tipolista);
begin
if puntero.circulos<>nil then
puntero.rectangulo_letra:=puntero.circulos.rectangulo
else
if puntero.pad<>nil then
puntero.rectangulo_letra:=puntero.pad.rectangulo_exterior;
end;

// Detecta los codigos del archivo Roën 1.0//

procedure extraer_datos_ULPGC(var archivo_ULPGC: text; var lista: tipolista);
var
cadena,coordenadas: string[125];
aux_lista: Tipolista;
salir:  boolean;
begin
readln(archivo_ULPGC,cadena);
salir:=false;
while (cadena<>'_FIN') and (not salir) do
 begin
    if cadena='_COMPONENTE' then
    begin
       new(aux_lista);
       aux_lista.letras_referencia:=nil;
       aux_lista.arcos:=nil;
       aux_lista.circulos:=nil;
       aux_lista.estructura.recta:=nil;
       aux_lista.pad:=nil;
       aux_lista.atributos.numero_pines:=0;
       aux_lista.prox:=lista;
       lista:=aux_lista;
       lista.enabled:=true;
       lista.atributos.activo:=true;
       readln(archivo_ULPGC,lista.atributos.referencia);
       readln(archivo_ULPGC,lista.atributos.componente);
       readln(archivo_ULPGC,cadena);
       readln(archivo_ULPGC,cadena);
        while cadena<> '_FIN_COMPONENTE' do
        begin
            readln(archivo_ULPGC,cadena);
             while  cadena<>'_LINEAS' do begin
             sacar_padS(cadena,lista);
             readln(archivo_ULPGC,cadena);
```

```
            end;
          if cadena='_LINEAS' then begin
            readln(archivo_ULPGC,cadena);
            while  cadena<>'_ARCOS' do begin
            sacar_rectas(cadena,lista);
            readln(archivo_ULPGC,cadena);
            end;
          end;
          if cadena='_ARCOS' then begin
          readln(archivo_ULPGC,cadena);
             while  cadena<>'_CIRCULOS' do begin
             sacar_ARCO(cadena,lista);
             readln(archivo_ULPGC,cadena);
             end
          end;
          if cadena='_CIRCULOS' then begin
          readln(archivo_ULPGC,cadena);
             while  cadena<>'_FIN_COMPONENTE' do begin
             sacar_circulos(cadena,lista);
             readln(archivo_ULPGC,cadena);
             end
          end;
        end;
        sacar_rectangulo_letra(lista);
construir_dynamic_matriz(aux_lista.matriz_pines,aux_lista.atributos.numero_pines,aux_lista.atributos.nu
mero_pines);
        construir_dynamic_vector(aux_lista.vector_pines_indx,aux_lista.atributos.numero_pines);
        construir_dynamic_vector(aux_lista.vector_pines,aux_lista.atributos.numero_pines);
       salir:=esquema.introducir_datos(aux_lista)
    end
    else
readln(archivo_ULPGC,cadena)
end;
end;

end.
```

```
unit Verdatos;

interface


uses
  SysUtils,WinTypes, {WinProcs, Messages} Classes,{ Graphics,} Controls,
  Forms, Dialogs,Mask, Buttons, TabNotBk,
  ComCtrls, StdCtrls, ExtCtrls,tipo1,tipo_eventos,opciones;

type

  TFverdatos = class(TForm)
    caja_general: TGroupBox;
    editor_general: TListBox;
    caja_lista: TGroupBox;
    editor_lista: TListBox;
    BitBtn1: TBitBtn;
    BOK: TBitBtn;
    Bcancelar: TBitBtn;
    BSalirYsalvar: TBitBtn;
    pestanas: TTabbedNotebook;
    caja2: TGroupBox;
    Label1: TLabel;
    Label14: TLabel;
    EReferencia: TEdit;
    Etipo: TEdit;
    caja3: TGroupBox;
    Label10: TLabel;
    Ecalor: TEdit;
    caja5: TGroupBox;
    Label3: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    EAncho: TEdit;
    ELargo: TEdit;
    EEspesor: TEdit;
    caja6: TGroupBox;
    Label17: TLabel;
    Label18: TLabel;
    Label28: TLabel;
    Etj: TEdit;
    Etp: TEdit;
    Etc: TEdit;
    caja4: TGroupBox;
    Label6: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    Erthp: TEdit;
    Label5: TLabel;
    Erthcp: TEdit;
    ERTHca: TEdit;
    ERthjc: TEdit;
    Rvalor_pines: TGroupBox;
    Label7: TLabel;
    Eespesorpin: TEdit;
    Label12: TLabel;
    Elargopin: TEdit;
    Label13: TLabel;
    Eanchopin: TEdit;
```

```
    Label21: TLabel;
    Econductividadpin: TEdit;
    Label22: TLabel;
    Erthpin: TEdit;
    Label15: TLabel;
    EPotencia: TEdit;
    RGcomponente: TRadioGroup;
    RGPotencia: TGroupBox;
    Rpotencia_superficie: TRadioButton;
    Rpotencia_pines: TRadioButton;
    Button1: TButton;
    procedure BcancelarClick(Sender: TObject);
    procedure ERTHJKeyPress(Sender: TObject; var Key: Char);
    procedure ERTHJExit(Sender: TObject);
    procedure BOKClick(Sender: TObject);
    procedure BSalirYsalvarClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure EX_posicionKeyPress(Sender: TObject; var Key: Char);
    procedure editor_generalKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure editor_generalMouseDown(Sender: TObject;
      Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure Rpotencia_superficieClick(Sender: TObject);
    procedure RGcomponenteClick(Sender: TObject);
    procedure Rpotencia_pinesClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
      atributos_para_general: Tatributos;
      puntero_aux,puntero_de_entrada: tipolista;
      matriz,matrizQ: Tmatriz;
      prueba: tipolista;
      ok,salvarYsalir,cancelar: boolean;
      forma_copia,forma_copia_interior: Testilo_modificar_componente;

      digitos: integer;
      componet: cadena30;

      lista: tipolista;
      fisicos: Tpuntero_fisicos;

      procedure poner_atributos(var atributos: tatributos ; estilo: Testilo_modificar_componente);
      procedure poner_1vez_atributos(var puntero: tipolista);
      procedure  cambios_datos(valor: integer; var sender :TObject; clase : integer);
      procedure  poner_datos_lista;
      PROCEDURE actualiza_datos(var atributos: Tatributos);
       procedure salvar_mismo_tipo_componente(var lista: tipolista; componente: cadena30);
      procedure salvar_todo_componente(var lista: tipolista);
      procedure extraer_atributos(var atributos: Tatributos);
      function modificar_componente(var puntero: tipolista ): boolean;
      procedure rectificar_datos_en_libreria(var puntero_de_entrada: tipolista; atributos_para_general:
Tatributos;
      forma_copia: Testilo_modificar_componente);
      procedure rectificar_lista_atributos(var puntero_de_entrada: tipolista;
      lista_atributos: tipoatributos);
      procedure desactiva_edit_superficie;
      procedure desactivo_activo_edit;
```

```
   { Private declarations }
 public
procedure modo_entrada (forma:Testilo_modificar_componente;var puntero_entrada,lista_aux: tipolista;
fisicos_aux: Tpuntero_fisicos; var matriz_aux_t,matriz_aux_q: tmatriz );

   { Public declarations }
 end;

var
 Fverdatos: TFverdatos;
implementation
uses
   padre,Usacar_datos_atributos, tabla2;

{$R *.DFM}

//Ventana de mensage de guardar datos o no//

procedure TFverdatos.BcancelarClick(Sender: TObject);
begin
if forma_copia=ESentrar_componente_libreria then
 if messagedlg ( 'Si eliges NO, el componente ('+ puntero_aux.atributos.COMPONENTE +') no entrará en
la simulación. ',  MTwarning,
                    [Mbcancel , MbNO],0)=MRno THEN
modalresult:=mrno
else
modalresult:=mrnone
else
modalresult:=mrno;
end;


procedure TFverdatos.ERTHJExit(Sender: TObject);
begin
 {@*********************************************************************
 Lo que hace las dos lineas de abajo es poner las unidades al final de
 cada companente EDIT, basandonos en la propiede TAG que tiene todos los
 componentes, así, sabemos que indice ocupa en la matriz donden están las
 cadenas unidades llamado array_unidades, TAG tiene un indice que le di
 en tiempo de construcion que le he dado yo en relacion con la colocación
 de las unidades en la matriz
 *********************************************************************}

{( SENDER as Tedit ).TEXT:=QUITAR_ERRORES(( SENDER AS TEDIT).TEXT);
 ( SENDER as Tedit ).TEXT:= poner_unidad(array_unidades[( SENDER AS TEDIT).tag],
       ( SENDER AS TEDIT).text);}
end;

procedure TFverdatos.BOKClick(Sender: TObject);
begin
Fopciones.salvar_tipo.itemindex:=0;
puntero_aux.preparado:=true;
modificar_componente(puntero_aux);
modalresult:=mrok;
end;

procedure TFverdatos.ERTHJKeyPress(Sender: TObject; var Key: Char);
begin
     { Sólo permitir la entrada de caracteres='0'..'9',',','e',#8,'-','+','E' }
mmKeyPress(Sender,Key);
```

```
end;

{@**********************************************************************
El modo de entrada se refiere a como se entra en está ficha. Para
poder desabilitar los botones pertinentes
**********************************************************************}

procedure TFverdatos.modo_entrada (forma:Testilo_modificar_componente;var
puntero_entrada,lista_aux: tipolista;
fisicos_aux: Tpuntero_fisicos; var matriz_aux_t,matriz_aux_q: tmatriz );
begin
forma_copia:=forma;
componet:=puntero_entrada.atributos.componente;
puntero_aux:=puntero_entrada;
atributos_para_general:=puntero_entrada.atributos;
puntero_de_entrada:=puntero_aux;
fisicos:=fisicos_aux;
matriz:=matriz_aux_t;
matrizQ:=matriz_aux_q;
forma_copia_interior:=ESmodificar_comoponente_esquema;
top:=25;
left:=5;
lista:=lista_aux;
//Se pone todos los componentes de la libreria componentes en combo-box general//
editor_general.items.clear;
editor_general.items.AddStrings(archivo_componentes);
//Se ponen todos los componentes que se tienen en ese momento en la lista en el combo_box
componentes//
poner_datos_lista;
caja2.caption:=puntero_AUX.atributos.componente;
RGcomponente.enabled:=true;
RGPotencia.enabled:=true;
if puntero_aux.atributos.estilo_modo_potencia=EMPpines then
Rpotencia_pines.checked:=true
else
Rpotencia_superficie.Checked := True;
if puntero_aux.atributos.activo then
RGcomponente.itemindex:=0
else
RGcomponente.itemindex:=1;
caja_general.caption:='Librería componentes';
caja_lista.caption:='lista de componentes';
case forma of
ESmodificar_comoponente_esquema:
 begin
 desactivo_activo_edit;
 bok.enabled:=true;
 actualiza_datos(puntero_aux.atributos);
 poner_atributos(puntero_aux.atributos,ESmodificar_comoponente_esquema);
 end;
ESentrar_componente_libreria,ESmodificar_componente_libreria:
 begin
   bok.enabled:=false;
   RGcomponente.enabled:=false;
   RGPotencia.enabled:=false;
    EReferencia.text:=puntero_de_entrada.atributos.referencia;
   Etipo.text:=puntero_de_entrada.atributos.tipo;
   actualiza_datos(puntero_aux.atributos);
  poner_atributos(puntero_aux.atributos,ESentrar_componente_libreria);
end;
```

```
end;
end;

{@*****************************************************************
Pone los datos de los atributos de los componente para que se vean
*****************************************************************}

procedure Tfverdatos.poner_atributos(var atributos: tatributos ; estilo: Testilo_modificar_componente);
begin
if not atributos.activo then
RGcomponente.itemindex:=1
else  begin
RGcomponente.itemindex:=0;
with atributos do  begin
if   forma_copia_interior=ESmodificar_comoponente_esquema then begin
  if estilo=ESmodificar_comoponente_esquema then
    begin
      EReferencia.text:=referencia;
      Etipo.text:=tipo;
      end;
if RGPotencia.enabled then
 case estilo_modo_potencia of
 EMPcomponentes:  Rpotencia_superficie.Checked := True;
 EMPpines: Rvalor_pines.enabled:=true;
end;
end;
ElargoPin.text:=floattostrF(largopin,ffFixed,8,digitos);
EEspesorpin.text:=floattostrF(ESPESORpin,ffFixed,8,digitos);
EAnchopin.text:=floattostrF(anchopin,ffFixed,8,digitos);
EConductividadpin.text:=floattostrF(conductividadpin,ffFixed,8,digitos);
ERthpin.TEXT:=floattostrF(Rthpin,ffFixed,8,digitos);
caja2.caption:=componente;
EAncho.text:=floattostrF(ancho,ffFixed,8,digitos);
Elargo.text:=floattostrF(largo,ffFixed,8,digitos);
EEspesor.text:=floattostrF(espesor,ffFixed,8,digitos);
Epotencia.text:=floattostrF(potencia,ffFixed,8,digitos);
ERthjc.TEXT:=floattostrF(Rthjc,ffFixed,8,digitos);
ERthca.TEXT:=floattostrF(rthca,ffFixed,8,digitos);
ERthcp.TEXT:=floattostrF(Rthcp,ffFixed,8,digitos);
 if numero_pines<=4 then  begin
 Rpotencia_superficie.enabled:=true;
 Rpotencia_pines.enabled:=true
 end
else begin
 Rpotencia_superficie.enabled:=true;
 Rpotencia_pines.enabled:=false;
 end;
end;
end;
end;

{@*****************************************************************
Esta función hace lo contrario a la anterior , saca los datos de los
componentes EDIT y los introduce el registro atributos, ante se comvierte
la cadena a coma flotante, en caso de que hubiera algún componente  EDIT
sin ningún dato la función devolvera la variable de salida como true
*****************************************************************}

function Tfverdatos.modificar_componente(var puntero: tipolista ): boolean;
begin
```

```
result:=false;
try
case Fopciones.salvar_tipo.itemindex  of
0 :       extraer_atributos(puntero.atributos);
1 :       salvar_mismo_tipo_componente(lista,puntero.atributos.componente);
2 :       salvar_todo_componente(lista);
end;
Except
begin
   result:=true;
   messagebeep($ffff);
   Showmessage (' Hay datos decimales mal escritos o sin ningún dato'); { Si falla }
end;
end;
end;

//Salva los datos para todos los componentes del mismo tipo//

procedure Tfverdatos.salvar_mismo_tipo_componente(var lista: tipolista; componente: cadena30);
 var
 aux: tipolista;
 begin
 aux:=lista;
 while aux<>nil do
 begin
     if aux.atributos.componente=componente then begin
     aux.preparado:=true;
     extraer_atributos(aux.atributos);
     end;
     aux:=aux.prox;
 end;
 end;

//Salva los datos para todos los componentes//

 procedure Tfverdatos.salvar_todo_componente(var lista: tipolista);
 var
 aux: tipolista;
 begin
 aux:=lista;
 while aux<>nil do
 begin
     aux.preparado:=true;
     extraer_atributos(aux.atributos);
     aux:=aux.prox;
 end;
 end;

//Extraer los datos dependiendo de la manera que se entro//

procedure Tfverdatos.extraer_atributos(var atributos: Tatributos);
begin
case Rgcomponente.itemindex of
1:
atributos.activo:=false;
0: begin
atributos.activo:=true;
 with atributos do begin
    if (forma_copia=ESentrar_componente_libreria) OR
(forma_copia=ESmodificar_componente_libreria) THEN begin
```

```
        tj:=0;
        tp:=0;
        tc:=0;
        calor:=0;
        referencia:='XXXXXXXXX';
      end;
 try
   IF Fopciones.salvar_tipo.itemindex=0 THEN BEGIN
     Ancho:=strTOfloat(Eancho.text);
     largo:=strTOfloat(Elargo.text);
     Espesor:=strTOfloat(EEspesor.text);
   END;
if numero_pines>4 then
estilo_modo_potencia:=EMPcomponentes
else begin
    if Rpotencia_superficie.checked then
      estilo_modo_potencia:=EMPcomponentes
    else begin
        estilo_modo_potencia:=EMPpines;
        Conductividadpin:=strTOfloat(Econductividadpin.texT);
        largopin:=strTOfloat(Elargopin.text);
        Espesorpin:=strTOfloat(EEspesorpin.text);
        anchopin:=strTOfloat(Eanchopin.text);
        rthpin:=largopin/(conductividadpin*anchopin*espesorpin);
        end;
    end;
Rthca:=strTofloat(ERthca.text);
rthjc:=strTofloat(ERthjc.text);
Rthcp:=strTofloat(ERthcp.text);
potencia:=strTofloat(Epotencia.text);
potencia:=strTOfloat(Epotencia.text);
except
on Ematherror do begin
   messagebeep(0);
   showmessage('hay datos que no pueden ser convertidos');
   end;
end;
end;
end;
end;
end;

//Poner datos por 1° vez el componente no está en librería//

procedure Tfverdatos.poner_1vez_atributos(var puntero: tipolista);
var
vv: integer;
begin
with puntero.atributos do
begin
EReferencia.text:=referencia;
Etipo.text:=tipo;
end;
end;

procedure TFverdatos.FormCreate(Sender: TObject);
begin
digitos:=4;
top:=0;
left:=0;
```

```
end;

//Sólo admite este tipo de datos, a la hora de introducirlos//

procedure TFverdatos.EX_posicionKeyPress(Sender: TObject; var Key: Char);
begin
      { Sólo permitir la entrada de caracteres='0'..'9',#8,'-','+'}
    case key of
     '0'..'9',#8,'-','+': key:=key;
      else
      Key := #0;
    end;
end;

//Se mueve por la lista de los elementos de la librería//

procedure TFverdatos.editor_generalKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var
valor: integer;
  begin
  with (sender as Tlistbox) do
  begin
  if itemindex<>-1  then
  case key of
  VK_UP: begin
       if itemindex>0   then
         valor:=itemindex-1
       else
         valor:=0;
       cambios_datos(valor, sender, tag);
       end;
VK_DOWN: begin
       if itemindex<items.count -1   then
         valor:=itemindex+1
       else
         valor:=items.count-1 ;
       cambios_datos(valor,sender,tag );
       end;

  end;
  end;

end;

{Se busca los datos en la lista de la librería o la lista de componente
actual de la placa}

procedure  TFverdatos.cambios_datos(valor: integer; var sender :TObject;
 clase: integer);
 var
 posicion_lista: longint;
prueba: tipolista;
  begin
case clase of
//esta forma es la lista de los componentes generales//
1:     begin
      detecta_atributos_lista((sender as Tlistbox).items[valor],atributos_para_general);
      desactivo_activo_edit;
      actualiza_datos(atributos_para_general);
```

```
      forma_copia_interior:=ESmodificar_componente_libreria;
     desactivo_activo_edit;
      end;

//Esta forma es  componentes del esquema actual//
2: begin
      puntero_aux:=lista;
      while  (sender as Tlistbox).items[valor]<> puntero_aux.atributos.referencia do
      puntero_aux:=puntero_aux.prox;
      atributos_para_general:=puntero_aux.atributos;
      actualiza_datos(atributos_para_general);
      forma_copia_interior:=ESmodificar_comoponente_esquema;
      desactivo_activo_edit;
  end;
end;
end;


procedure TFverdatos.editor_generalMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
cambios_datos((sender as Tlistbox).itemindex,sender,(sender as Tlistbox).tag);
end;
 procedure  TFverdatos.poner_datos_lista;
 var
 prueba: tipolista;
 contador: integer;
 begin
 editor_lista.items.clear;
 prueba:=lista;
 while  prueba<> nil do
 begin
     editor_lista.items.add(prueba.atributos.referencia);
     prueba:=prueba.prox;
 end;
 end;

//Cambia la presión de los datos//

procedure TFverdatos.BitBtn1Click(Sender: TObject);
var
valor: integer;
ClickedOK: Boolean;
NewString,cadena: string;
begin
 ClickedOK := InputQuery('introsducir nuevo valor de digitos', 'el valor nuevo es:', NewString);
  if ClickedOK then
    try
    valor:=strTOint(NewString);
    digitos:= valor;
    except
    end;
actualiza_datos(atributos_para_general);
poner_atributos( atributos_para_general, forma_copia);
end;

//Visualiza los datos de la temperatura de la simulación//

PROCEDURE TFverdatos.actualiza_datos(var atributos: Tatributos);
begin
```

```
WITH atributos do
if (matriz.activa) and (cell.x<>0) and (activo) and
(forma_copia<>ESmodificar_componente_libreria) then begin
  ERthp.TEXT:=floattostrF(rthp,ffFixed,8,digitos);
  Etp.TEXT:=floattostrF(tp,ffFixed,8,digitos);
  Ecalor.TEXT:=floattostrF(matrizQ.matriz[cell.y,cell.x],ffFixed,8,digitos);
  Etc.TEXT:=floattostrF(tc,ffFixed,8,digitos);
  Etj.TEXT:=floattostrF(tj,ffFixed,8,digitos);
  end
else begin
  Etp.TEXT:='desactivado';
  Ecalor.TEXT:='desactivado';
  Etc.TEXT:='desactivado';
  Etj.TEXT:='desactivado';
  Erthp.TEXT:='desactivado';
   end;
end;

{Averiguar la forma de entrada (librería, componente nuevo o
modificar componente de esquema y salva los cambios}

procedure TFverdatos.BSalirYsalvarClick(Sender: TObject);
var
posicion_lista: integer;
lista_atributos_aux: tipoatributos;
posicion: Tposicion_atributos;
begin
     Fopciones.modo_forma_copia(forma_copia);
     if Fopciones.showModal=mryes then
     begin
       case  forma_copia of
       ESmodificar_comoponente_esquema:
       modificar_componente(puntero_aux);
       ESentrar_componente_libreria:
       begin
          rectificar_datos_en_libreria(puntero_de_entrada, atributos_para_general,forma_copia);
          if not archivo_componentes.find(puntero_de_entrada.atributos.componente, posicion_lista) then
           begin
              archivo_componentes.add(puntero_de_entrada.atributos.componente);
              archivo_componentes.sort;   //se ordena la  lista//
              new(lista_atributos_aux);
              lista_atributos_aux.atributos:=puntero_de_entrada.atributos;
              lista_atributos_aux.prox:=lista_atributos;
              lista_atributos:=lista_atributos_aux;
           end;
         end;
       ESmodificar_componente_libreria:
       rectificar_datos_en_libreria(puntero_de_entrada, atributos_para_general,forma_copia);
       end;
 modalresult:=mryes;
 end
end;

//Rectifica los datos de la librería//

procedure TFverdatos.rectificar_datos_en_libreria(var puntero_de_entrada: tipolista;
atributos_para_general: Tatributos;
forma_copia: Testilo_modificar_componente);
var
aux_atributos: Tatributos;
```

```
begin
try
   aux_atributos:=puntero_de_entrada.atributos;
   extraer_atributos(atributos_para_general);
   puntero_de_entrada.atributos:=atributos_para_general;
   with puntero_de_entrada.atributos do
   begin
      tipo:=aux_atributos.tipo;
      numero_pines:=aux_atributos.numero_pines;
      if forma_copia=ESentrar_componente_libreria then begin
      componente:=aux_atributos.componente;
      referencia:=aux_atributos.referencia;
      end;
   end;
   poner_archivo_atributOs(puntero_de_entrada.atributos, archivo_atributos,4,forma_copia);
   if forma_copia=ESmodificar_componente_libreria then
   rectificar_lista_atributos(puntero_de_entrada,lista_atributos);
Except
begin
messagebeep($ffff);
Showmessage (' Hay datos decimales mal escritos o sin ningún dato'); { Si falla }
end;
end;
end;

//Modifica la lista general del programa de los componentes//

procedure TFverdatos.rectificar_lista_atributos(var puntero_de_entrada: tipolista;
lista_atributos: tipoatributos);
var
puntero: tipoatributos;
begin
puntero:=lista_atributos;
   while puntero<> nil do begin
       if puntero.atributos.componente=puntero_de_entrada.atributos.componente then
         puntero.atributos:=puntero_de_entrada.atributos;
   puntero:=puntero.prox;
   end;
end;


procedure TFverdatos.FormActivate(Sender: TObject);
begin
top:=40;
left:=35;
end;

procedure TFverdatos.Rpotencia_superficieClick(Sender: TObject);
begin
puntero_aux.atributos.estilo_modo_potencia:=EMPcomponentes;
desactivo_activo_edit;
end;


procedure TFverdatos.desactiva_edit_superficie;
begin
end;

{Activa o desactiva los edir correspondiente según sea
potencia por los pines o atraves de la superficie}
```

```
procedure TFverdatos.desactivo_activo_edit;
var
i: integer;
begin
case forma_copia of
ESmodificar_comoponente_esquema:  begin
     if Rgcomponente.itemindex=1 then begin
       puntero_aux.atributos.activo:=false;
       for i:=0 to Componentcount-1 do
          if components[i] is Tedit then
          with  (components[i] as Tedit) do begin
          enabled:=false;
          text:='desactivado';
          end;
     end
     else begin
        puntero_aux.atributos.activo:=true;
        for i:=0 to Componentcount-1 do
        if  components[i] is Tedit then
        with  (components[i] as Tedit) do
        if tag<>0 then begin
          if puntero_aux.atributos.estilo_modo_potencia=EMPpines then begin
             if tag=4 then begin
             enabled:=true;
             text:='';
             end
             else begin
             enabled:=false;
             text:='';
             end
          end
          else begin
              if tag=3 then begin
              enabled:=true;
              text:='';
              end
              else begin
             enabled:=false;
             text:='';
             end;
          END;
         end
          else
             enabled:=true;
  end;
end;
ESmodificar_componente_libreria,ESentrar_componente_libreria:
begin
for i:=0 to Componentcount-1 do
  if components[i] is Tedit then
  with  (components[i] as Tedit) do begin
  if tag<>0 then
  enabled:=true
  else
  text:='desactivado';
  end;
end;
end;
atributos_para_general.activo:=puntero_aux.atributos.activo;
```

```
poner_atributos(atributos_para_general, forma_copia);
end;



procedure TFverdatos.RGcomponenteClick(Sender: TObject);
begin
desactivo_activo_edit;
end;


procedure TFverdatos.Rpotencia_pinesClick(Sender: TObject);
begin
puntero_aux.atributos.estilo_modo_potencia:=EMPpines;
desactivo_activo_edit;
end;

// Visualiza tabla de pines//

procedure TFverdatos.Button1Click(Sender: TObject);
begin
Ftabla_pines.showmodal;
end;

end.
```

```
unit calculo_potencia1;

interface
uses
Dialogs,tipo1,windows,lu;
procedure potencia_primaria(var lista: tipolista; var matrizT,matrizQ,matrizTAIRE_ARRIBA,
matrizTAIRE_ABAJO: Tmatriz;var fisicos: Tparametros_fisicos; var rectangulo_cuadricula: Trect; const
delta: real;var cambio: double);
function nueva_potencia(var lista: tipolista; var matrizT,matrizQ,
matrizTAIRE_ARRIBA,matrizTAIRE_ABAJO: Tmatriz;
var cambio: DOUBLE; var rectangulo_cuadricula: Trect; var fisicos: Tparametros_fisicos; var delta:
real):boolean;
procedure localizar_rect_matriz(var lista: Tipolista; var matrizQ: tmatriz;
rectangulo_cuadricula: Trect; const  delta,escala: real);
procedure volcado_matriz_chica_grande(var lista: Tipolista; var matrizT,matriz_esp: tmatriz;
rectangulo_cuadricula: Trect; const  delta: real);
procedure construye_matriz_pines(var matriz: Tmatriz; var puntero: tipolista;
suma_extra: real);
function calcula_potencia_pin_nueva(var puntero: Tipolista; var
matrizT,matrizQ,matrizTAIRE_ARRIBA,
matrizTAIRE_ABAJO: Tmatriz;var rectangulo_cuadricula: Trect;  delta: real; var cambio: double):
boolean;
procedure construye_vector_pines(var matrizT,matrizTAIRE_ARRIBA: tmatriz; var b: Tvector; var
puntero: tipolista);
function calcula_potencia_pin_primaria(VAR puntero: Tipolista; var matrizT,matrizQ,
matrizTAIRE_ABAJO,matrizTAIRE_ARRIBA: Tmatriz;
var rectangulo_cuadricula: Trect; var fisicos: Tparametros_fisicos; delta: real): boolean;


implementation

//calcula la potencia en el primer instante//

procedure potencia_primaria(var lista: tipolista; var matrizT,matrizQ,matrizTAIRE_ARRIBA,
matrizTAIRE_ABAJO: Tmatriz;
var fisicos: Tparametros_fisicos; var rectangulo_cuadricula: Trect; const delta: real;var cambio: double);
var
prueba: tipolista;
Vc,VBUP,Q,valor,d: real;
begin
TRY
prueba:=lista;
while prueba <> nil do begin
if (prueba.atributos.cell.x<>0) and (prueba.atributos.activo) then begin
with prueba.atributos do begin
case estilo_modo_potencia of
EMPpines:  begin
      calcula_potencia_pin_primaria(prueba,matrizT,matrizQ
,matrizTAIRE_ARRIBA,matrizTAIRE_ABAJO,rectangulo_cuadricula,fisicos, delta);
       //prepara las matrices pines //
       construye_matriz_pines(prueba.matriz_pines,prueba,0);
       ludcmp(prueba.matriz_pines, prueba.matriz_pines.max_fila, prueba.vector_pines_Indx, d);
      end;
EMPcomponentes:
    begin
    if largo=ancho then
    Rthp:=fisicos.espesor_aislante/(fisicos.k_aislante*ancho*(ancho+2*fisicos.espesor_aislante))
    else
    if largo>ancho then
```

```
    RTHp:=(1/(2*fisicos.k_aislante*(largo-ancho)) * abs(ln(largo/ancho*
(2*fisicos.espesor+ancho)/(2*fisicos.espesor_aislante+largo) )))
    else
    RTHp:=(1/(2*fisicos.k_aislante*(ancho-largo)) * abs(ln(ancho/largo*
(2*fisicos.espesor+largo)/(2*fisicos.espesor_aislante+ancho) )));
    VBUP:=potencia*(1 -
(RTHP+Rthcp)/(Rthca+RTHP+Rthcp))*RTHP+matrizTAIRE_ABAJO.MATRIZ[cell.y,cell.x];
    valor:=(VBUP-matrizTAIRE_ABAJO.MATRIZ[cell.y,cell.x])/(Rthp*largo*ancho);
    calculo_posicion_potencia_malla( prueba,
matrizQ,prueba.rect_localiza_matriz,rectangulo_cuadricula, delta,valor);
    end;
end;
end;
end;
prueba:=prueba.prox;
end;
except
    showmessage('Datos de algun componente a cero, o no aceptables');
   end;
end;

//calcula potencia distinta de la inicial//

function nueva_potencia(var lista: tipolista; var matrizT,matrizQ,
matrizTAIRE_ARRIBA,matrizTAIRE_ABAJO: Tmatriz;
var cambio: DOUBLE; var rectangulo_cuadricula: Trect; var fisicos: Tparametros_fisicos; var delta:
real): boolean;
var
prueba: tipolista;
vcc,Q,valor: real;
begin
nueva_potencia:=true;
prueba:=lista;
while prueba <> nil do begin
if (prueba.atributos.cell.x<>0) and (prueba.atributos.activo) then begin
    with prueba.atributos do
    {si entra en este if es que la potencia no sale si no entra no que
    hay que cantar error }
    case estilo_modo_potencia of
    EMPpines:
     if calcula_potencia_pin_nueva(prueba,
matrizT,matrizQ,matrizTAIRE_ARRIBA,matrizTAIRE_ABAJO,rectangulo_cuadricula,delta,cambio)
then begin
        messagedlg('! EL COMPONENTE ( '+ PRUEBA.ATRIBUTOS.REFERENCIA+' ) NO ESTA
EMITIENDO '+ #10 +
         'ENERGIA, SINO RECIBIENDO ', MTwarning,[],0);
          nueva_potencia:=false;
         cambio:=0;
         EXIT;
         end;
   EMPcomponentes:
     begin
        VCC:=1/(Rthca+Rthcp+rTHP) * (potencia*(Rthca*rthp)+ (RThca+Rthcp)*
        matrizT.matriz[cell.y,cell.x] +
        matrizTAIRE_ABAJO.matriz[cell.y,cell.x]*Rthp);
        valor:=(Vcc-matrizT.matriz[cell.y,cell.x])/(Rthp*largo*ancho);
        if valor<0 then begin
         messagedlg('! EL COMPONENTE ( '+ PRUEBA.ATRIBUTOS.REFERENCIA+' ) NO ESTA
EMITIENDO '+ #10 +
        'ENERGIA, SINO RECIBIENDO', MTwarning,[],0);
```

```
          nueva_potencia:=false;
          cambio:=0;
          EXIT;
          end else
         calculo_posicion_potencia_malla( prueba, matrizQ,
prueba.rect_localiza_matriz,rectangulo_cuadricula, delta,valor);
       end;
     end;
end;
prueba:=prueba.prox;
end;
end;

//calcula potencia inical de los pines//

function calcula_potencia_pin_primaria(VAR puntero: Tipolista; var matrizT,matrizQ,
matrizTAIRE_ABAJO,matrizTAIRE_ARRIBA: Tmatriz;
var rectangulo_cuadricula: Trect; var fisicos: Tparametros_fisicos; delta: real): boolean;
var
d,vcc,valor,rthp,vbup,RTH: real;
i,n: integer;
pads: tipopad;
begin
with puntero.atributos do
begin
d:=0;
pads:=puntero.pad;
n:=1;
calcula_potencia_pin_primaria:=false;
RTH:=rthpin/numero_pines;
    while  pads<> nil do begin
    VBUP:=+
    matrizTAIRE_ABAJO.MATRIZ[pads.cell.y,pads.cell.x];
     Vcc:=potencia*((Rth)-
(sqr(Rth)/(Rthca+Rth)))+matrizTAIRE_ABAJO.MATRIZ[pads.cell.y,pads.cell.x];
     valor:=(Vcc-vbup)/(Rthpin*espesorpin*anchopin);
         if valor<0 then
          messagedlg('! EL COMPONENTE ( '+  puntero.ATRIBUTOS.REFERENCIA+' )  NO ESTA
EMITIENDO '+ #10 +
         'Error grave de datos', MTwarning,[],0);
         calculo_posicion_potencia_malla( puntero, matrizQ,
pads.rect_localiza_pin_matriz,rectangulo_cuadricula, delta,valor);
        pads:=pads.prox;
        inc(n);
    end;
end;
end;

//calcula la potencia , ya no es la inicial//

function calcula_potencia_pin_nueva(var puntero: Tipolista; var
matrizT,matrizQ,matrizTAIRE_ARRIBA,
matrizTAIRE_ABAJO: Tmatriz;var rectangulo_cuadricula: Trect;  delta: real; var cambio: double):
boolean;
var
d,vcc,valor: real;
i,n: integer;
pads: tipopad;
begin
construye_vector_pines(matrizT,matrizTAIRE_ARRIBA,puntero.vector_pines,puntero);
```

```
lubksb(puntero.matriz_pines,  puntero.matriz_pines.max_fila, puntero.vector_pines_indx,
puntero.vector_pines);
calcula_potencia_pin_nueva:=false;
with puntero.atributos do
begin
   pads:=puntero.pad;
    n:=1;
   while pads<> nil do begin
   vcc:=rthpin*puntero.vector_pines.vector[n]+matrizT.matriz[pads.cell.y,pads.cell.x];
    valor:=puntero.vector_pines.vector[n]/(espesorpin*anchopin);
    if valor<0 then begin
    calcula_potencia_pin_nueva:=true;
    end;
    calculo_posicion_potencia_malla( puntero, matrizQ,
pads.rect_localiza_pin_matriz,rectangulo_cuadricula, delta,valor);
    inc(n);
    pads:=pads.prox;
 end;
end;

end;


//construye la matriz del sistema de pines//

procedure construye_matriz_pines(var matriz: Tmatriz; var puntero: tipolista;
suma_extra: real);
var
columna,fila: integer;

begin
for fila:=1 to puntero.atributos.numero_pines-1 do  begin
matriz.matriz[fila,fila]:=-(puntero.atributos.rthpin+suma_extra);
matriz.matriz[fila,fila+1]:=puntero.atributos.rthpin+suma_extra;
end;
for columna:=1 to  puntero.atributos.numero_pines-1 do
matriz.matriz[puntero.atributos.numero_pines,columna]:=-(puntero.atributos.rthca);
 matriz.matriz[puntero.atributos.numero_pines,puntero.atributos.numero_pines]:=-
(puntero.atributos.rthca+puntero.atributos.rthpin+suma_extra);
end;

//construye los vectores de los pines//

procedure construye_vector_pines(var matrizT,matrizTAIRE_ARRIBA: tmatriz; var b: Tvector; var
puntero: tipolista);
var
n: integer;
pads: tipopad;
begin
pads:=puntero.pad;
for n:=1 to puntero.atributos.numero_pines-1 do begin
b.vector[n]:=matrizT.matriz[pads.cell.y,pads.cell.x]-matrizT.matriz[pads.prox.cell.y,pads.prox.cell.x];
pads:=pads.prox;
end;
b.vector[puntero.atributos.numero_pines]:=matrizT.matriz[pads.cell.y,pads.cell.x]-
matrizTAIRE_ARRIBA.MATRIZ[puntero.atributos.cell.y,puntero.atributos.cell.x]-
puntero.atributos.potencia*  puntero.atributos.rthca;
end;

//localiza rectangulo de la matriz//
```

```
procedure localizar_rect_matriz(var lista: Tipolista; var matrizQ: tmatriz;
rectangulo_cuadricula: Trect; const  delta,escala: real);
var
rect_intersec: trect;
columna,fila: integer;
puntero: Tipolista;
divisor: real;
begin
puntero:=lista;
while puntero <> nil do
if puntero.atributos.cell.x<>0 then begin
     rect_intersec.left:=round(puntero.rectangulo.izquierda*1000);
     rect_intersec.top:=round(puntero.rectangulo.arriba*1000);
     rect_intersec.right:=round(puntero.rectangulo.derecha*1000);
     rect_intersec.bottom:=round(puntero.rectangulo.abajo*1000);
     if intersrect( rect_intersec ,rectangulo_cuadricula,rect_intersec) then
          begin
            localizacion_matriz_punto(rectangulo_cuadricula,
            rect_intersec.TopLeft,delta, escala);
            localizacion_matriz_punto(rectangulo_cuadricula,
            rect_intersec.BottomRight,delta, escala);
          puntero.rect_localiza_matriz:=rect_intersec;
              divisor:=(rect_intersec.top-rect_intersec.bottom)*(rect_intersec.right-rect_intersec.left);
            for fila:=rect_intersec.bottom to rect_intersec.top do
              for columna:=rect_intersec.left to rect_intersec.right do

                  matrizQ.matriz^[fila,columna]:=300;
          end;
     puntero:=puntero.prox;
     end;

end;

end.
```

Termin 1.0
Simulador Térmico

Apéndice C

Bibliografía y direcciones de Internet de donde se ha sacado información para la realización de este proyecto.

Referencias Bibliográficas

[1] Francisco Charte
"Programación con Delphi", Ed. Anaya multimedia, S.A. 1996

[2] Charls Petzold
"Programación en Windows", Ed. Anaya multimedia, S.A. 1992

[3] Francisco Charte
"Programación avanzada con Delphi 2.0", Ed. Anaya multimedia, S.A. 1997

[4] William H. Press Brian P. Flannery, Sawl A. Tenkolsky, Williaw T. Vetterling.
"Numerical Recipes in Pascal", The Art of Scientific computing

[5] Alan J. Chapman
"Transmisión del calor", Ed. Bellisco, librería Editorial

[6] W. M. Rohsenow, J. P. Hartnett, E. N. Ganic
"Handbook od Het transfer fundamentals", Ed. MAC Graw Hill

[7] Hou-Cheng Huang, Asif S. Usmani
"Finite Element Analysis for Heat Transfer", Ed. Spring_Verlag

[8] "MATLAB User´s Guide",Ed. The Math Works Inc 1992

[9] D. J. Dean
"Thermal Design Aspects", Ed.

[10] Manuel Melis Maynar, Enrique de Miguel Anasagasti
"Cálculo Numérico (métodos amplicaciones)", Ed. Rueda

[11] M J Fagan
"Finite element Analysis", Ed. Longman

[12] Curtis F. Gerald, Patrick O. Wheatley
Applied Numerical Analysis", Ed. Addison-Wesley

Direcciones de Internet

[1] http://www.borland.com

[2] http://www.matlab.com

[3] http://www.delphi.com

termin 1.0
Simulador Térmico

ANEXO

Thermal Analysis and Design of Electronic Equipment

S. RAJARAM
Bell Laboratories, Whippany, New Jersey 07981, USA

ABSTRACT

Reliability of electronic equipment and devices is strongly dependent on temperature. Consequently, thermal design is an important part of the system design process. This paper is an overview of the entire thermal design process as applied to an air cooled system. Systematic analytical and experimental techniques are outlined to ascertain cooling requirements for electronic equipment to ensure reliable operation. With the trend towards ever increasing component and power densities, such a systematic thermal design approach is essential at an early stage in the equipment design.

INTRODUCTION

Thermal design and evaluation of the thermal performance of electronic systems is a major part of the system design process because of reliability considerations. Reliability depends upon the ability to control and maintain component and board temperatures at a relatively low value. It has been widely known for some time that reliability of electronic devices and equipment is primarily dependent on temperature. Operating devices at elevated temperatures inevitably leads to failure due to chemical reactions and creep in interconnections, and, diffusion in solid state devices to name a few disastrous effects. The rate of failure from such causes may be related directly to temperature in an Arrenhius type of mathematical expression. Thus, the failure rate shows an exponential dependence on component temperature.

Since temperature is recognized as the most severe and important environmental stress condition for electronics, thermal design has been and continues to be an active branch of Heat Transfer studies requiring applications of advanced cooling techniques. The trend towards increasingly higher performance and speed is driving power densities ever higher by dense packaging of VLSI circuits to reduce signal propagation delays. To take advantage of VLSI however, greatly enhanced heat removal techniques are necessary. High speed digital circuits using sub - micron technology have been fabricated dissipating up to 1 mw per gate. A VLSI containing 10^5 such gates dissipates 100w. An array of such high heat dissipating packages spaced less than half an inch apart to minimize signal propagation delays poses an extremely challenging cooling problem. Trilogy's experimental and pioneering Wafer Scale Integration (WSI) concept consisted of a 3 inch diameter wafer dissipating 1200w. The heat density for WSI is about 26 w/cm^2. We can get an appreciation for the severity of thermal problems associated with VLSI electronics today by referring to Fig. 1 which shows thermal loading and operating temperatures for various physical phenomena [1,2].

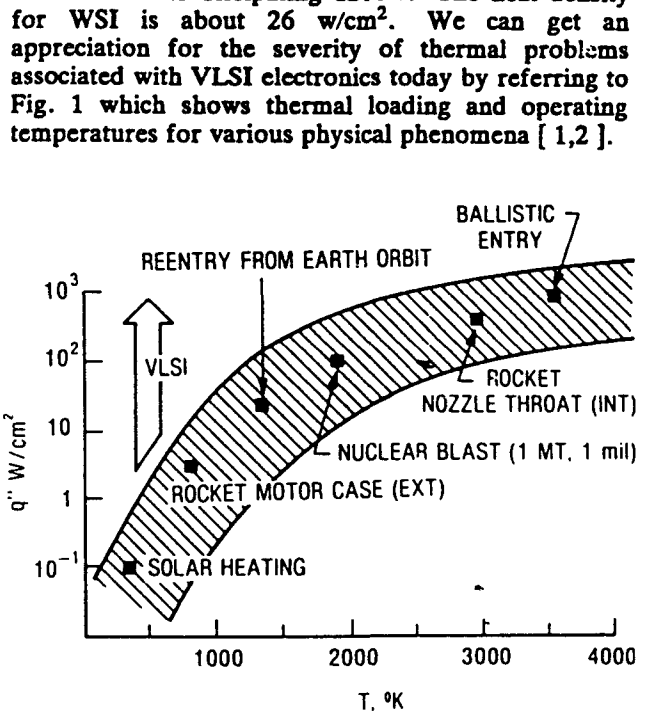


FIGURE 1. MICROELECTRONICS HEAT FLUXES. [1]

We observe from this figure that the thermal loading in VLSI chips is equivalent to that of vehicle reentry from earth orbit. From a device reliability point of view, it is necessary to maintain VLSI junction temperatures between 80 and 125 ° C. Therefore, while temperatures in the reentry process are allowed

to reach 1500 ° K, VLSI electronic circuits are constrained to function reliably at temperatures below 500 ° K. This stringent requirement has resulted in ever increasing sophisticated cooling and thermal design techniques.

Most electronic systems make use of all basic modes of heat transfer to some extent, although one mode may dominate the design. Air cooling is perhaps the most common cooling technique for telecommunication and computer equipment. For small heat loads, the air cooling may be by natural convection, while for large heat loads forced convection air cooling with the help of fans is used. The largest amount of heat is transferred by convection as air flows over the components and boards. However, some of the heat is conducted from the component to the boards and from the boards to the frame. In addition, hot components lose heat by radiation to the surrounding cooler components, boards and the chassis walls. Individual components dissipating large amounts of heat are usually fitted with fins which then provide extended surfaces from which heat is removed by conduction and air convection over these surfaces. For aircraft, space and missile applications where weight and volume are severe limitations, cold plate technology with compact heat exchangers are employed. The recent technique of multi - chip array packages used in computers has led to sophisticated air and water cooled heat exchanger designs. Among these, the IBM 4381 module employs air impingement cooling, while the Hitachi SiC RAM module and the Mitsubishi High Thermal Conduction Module employ finned heat sinks which dissipate the heat to air flowing over these fins. On the other hand the IBM 3081 Thermal Conduction Module (TCM), the Honeywell Silent Liquid Integral Cooler (SLIC) Module and the NEC SX Liquid Cooling Module employ water cooled cold plate designs to cool the multi chip array.

With the trend towards dense packaging of devices and the consequent increase in thermal density, other modes of heat transfer such as immersion cooling in dielectric flourocarbons with and without pool boiling, and heat pipes are being pursued for future applications. These trends in electronic equipment and packaging have also created an awareness of the need for the proper thermal design and analysis at an early stage of equipment design. Thermal effects which may have once been considered to be of secondary importance, may prove to be of serious consequence today. For example, an uncertainty of 10 ° C/W for a device dissipating 0.2W translates into an error of only 2 ° C. However, the same uncertainty for a device dissipating 2W, which is common today, translates into an error of 20 ° C, which could be disastrous. There is an effort today to perform systematic thermal studies of various cooling techniques and applications, in order to provide thermal design guidelines for electronic equipment and packages. The importance of electronic cooling may be judged by the appearance of three texts related to thermal techniques used in electronic equipment design [3,4,5,34].

Although cooling techniques in electronic equipment use many branches of Heat Transfer, the choice of a particular method depends upon the type of thermal loading condition and the intended application and environment. Most commercial equipment used in an office environment such as telephone equipment and computers still predominantly use air cooling techniques for thermal control. The environmental conditions are generally benign for such applications. Air cooling is also the simplest and cheapest technique for these applications. For applications in airplanes, satellites, spacecraft and missiles, the requirements vary tremendously because of the varying environmental conditions. However, all of these applications require that the cooling technique used, minimize weight and volume, and this requires efficient means of extracting heat. Airplane electronics are usually cooled by forced convection bleed air from the engine compressor. The air is cooled by throttling and dried before being used. Since humidity is known to cause failure of electronics, cooling air does not come in direct contact with the devices. The equipment is therefore cooled by air cooled heat exchangers called cold plates. Cooling of missile electronics depends upon the two conditions that are encountered, captive or free flight. If the flight duration is relatively small, then the cooling provided during the captive phase may be sufficient. However, if the flight duration is very long, then a separate cooling system must be provided for thermal control of the electronics for several hours. Military applications of electronic cooling employ both air cooled and liquid cooled cold plate heat exchangers. Spacecraft and satellite electronic cooling systems make use of radiation heat transfer to deep space where the temperature is 0 ° K (-273 ° C). However, special treatment of spacecraft surfaces is required to prevent absorption of large amounts of solar radiation. Liquid cooled cold plates are also used for space applications where the fluid is circulated through space radiators which lose the heat to deep space. Conduction heat transfer may also be used to get the heat to the surface. It is important to maintain smooth surfaces and high contact pressures to minimize thermal resistances across interfaces. Sealed closed loop air cooling systems may also be used, but the heat from the air has to be removed by a heat exchanger. Also, the sealed box must be structurally strong to sustain the pressure difference that exists in the vacuum of outer space. For ship and submarine applications, the electronics are usually packaged in large rugged cabinets and enclosures. Liquid cooled cold plate heat exchangers as well as open and closed loop air cooling techniques

are used. The electronics are sometimes mounted in individual drawers.

It is therefore easy to surmise that the cooling technique employed depends upon the application as well as the environmental conditions under which the electronics have to operate reliably. An accurate assessment of the expected thermal environment is necessary for defining the appropriate cooling requirement and subsequent design and analysis. With this in mind, the emphasis on cooling specification is based on the intended use of the equipment such as in offices, aircrafts, missiles, spacecraft, ships, etc. This avoids the difficulty of defining blanket cooling requirements to envelop all possible applications. Typical mission requirements for electronics aboard a combat jet aircraft are shown in Fig. 2 [6].

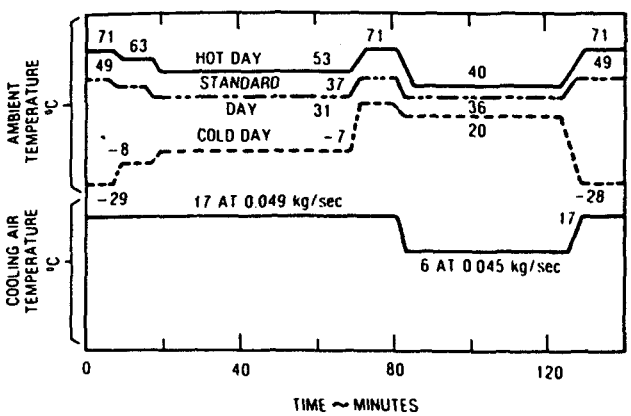


FIGURE 2. THERMAL ENVIRONMENT FOR TYPICAL COMBAT JET MISSION. [6]

The intention here is to give the thermal designer a feel for the relative magnitudes of temperatures and rates of change of temperature for this application.

RELIABILITY

As was stated earlier, temperature plays an important role in the failure of electronics and equipment. Some of the common failure mechanisms that are directly controlled or accelerated by temperature are chemical reactions, species diffusion, material decomposition, ion and surface charge movement, dielectric breakdown and electromigration [7]. The reaction rate for failure mechanisms due to physical or chemical processes that are dependent on temperature, are governed by an Arrhenius equation

$$R_r = R_{r0} \exp\left(\frac{-E}{kT}\right) \qquad (1)$$

where E is the activation energy in electron volt (ev), k the Boltzmann constant and T the absolute temperature.. Also, the time to failure is related to temperature by the equation [8]

$$\tau_f = \tau_0 \exp\left(\frac{E}{kT}\right) \qquad (2)$$

where R_{r0} and τ_0 are constants. From equations (1) and (2) we observe that the product of the reaction rate R and the time to failure τ_f is a constant. If the failure causing mechanism is accelerated according to equation (1), then the acceleration factor due to increased temperature from T_1 to T_2 is given by the ratio

$$\frac{\tau_{f1}}{\tau_{f2}} = \frac{R_{r2}}{R_{r1}} = \exp\left[\frac{E}{k}\left(\frac{1}{T_1} - \frac{1}{T_2}\right)\right] \qquad (3)$$

The activation energy E is a measure of the effect of temperature on the physical or chemical process causing failure. Hence, failure data may be characterized in terms of the activation energy E. Shown in Tables 1 and 2 are the acceleration factors and the time equivalent of 40 years at 60 ° C for failure mechanisms characterized by activation energies of 0.5 ev and 1.0 ev [8].

Table 1. Acceleration Factors As A Function Of T And E [8].

Acceleration Factor		
T (° C)	E = 1.0eV	E = 0.5 eV
300	2.2×10^6	1500
250	3.2×10^5	570
200	3.1×10^4	176
150	1700	41
125	300	17
85	11.5	3.4

Table 2. Time Equivalent To 40 Years As A Function Of T and E In Hours [8].

Time In Hours		
T (°C)	E = 1.0 eV	E = 0.5 eV
300	0.2	233
250	1.1	616
200	11	2000
150	200	8526
125	1200	20200
85	30000	103000

We observe that the acceleration factor at 300 ° C for E = 1.0 ev is four orders of magnitude higher than at E = 0.5 ev. Also, at 200 ° C, 11 hours of operation for E = 1.0 ev is equivalent to 40 years of operation at 60 ° C. The table very clearly demonstrates the drastic effects of temperature on device failure rate.

The failure rate $\lambda(\tau)$ of an integrated circuit (IC) generally varies with time as shown in Fig. 3 and is known as the bath tub curve.

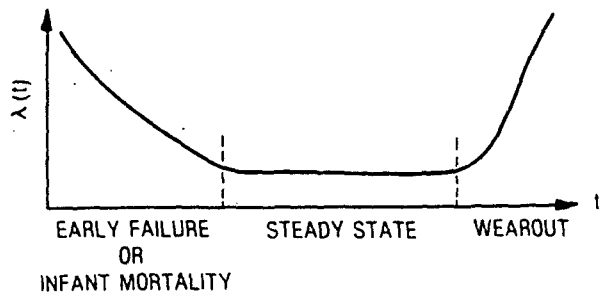


FIGURE 3. FAILURE RATES FOR TYPICAL ICs WITH TIME. [7,8]

This curve has three distinct characteristics. They are, infant mortality, steady state and wearout. The failure rates during infant mortality are high, but decreases with time. These infant mortality failures are generally due to manufacturing defects. Beyond this region, failures occur at a much lower rate and remain at a fairly constant level. This region is normally associated with normal operation of equipment and is of primary interest for reliability purposes. Finally, after a long duration of operation, the failure rate increases once again with time and this is the wearout region. This concept arises from physical wear which eventually leads to failure. The probability $F(\tau)$ that the device will fail in time τ may be related to $\lambda(\tau)$ by the equation

$$F(\tau) = 1 - \exp\left[\int_0^{\tau} \lambda(x)dx\right] = 1 - \exp(-\lambda\tau) \quad (4)$$

since $\lambda(\tau)$ is a constant for electronic devices during the useful life period. Reliability is generally defined as the probability that an item will perform a required function under certain conditions and for a certain period of time. Mathematically, we define a reliability function $R_f(\tau)$, which is the probability that the device will function to time τ without failure.

$$R_f(\tau) = 1 - F(\tau) = \exp(-\lambda\tau) \quad (5)$$

Also, the Mean Time Between Failure (MTBF) which is the total test time divided by the number of failures can be shown to be $= \frac{1}{\lambda}$.

THERMAL DESIGN

In this context, thermal design may be defined as the application of heat transfer studies to the design of electronic equipment. The objective of thermal design (or control) is to ensure that the temperatures of individual components are maintained within the specified limits for reliability of the equipment. The evidence of the importance of this subject can be seen from the numerous articles being published. Despite the number of articles and wide use of thermal design in electronics, generalized heat transfer data is still not available. As an example, let us consider forced air cooling of equipment which is still the most popular for its good performance, low cost and ease of implementation. The increasing demands on air cooling due to higher power densities have brought the "science" of air cooling to its practical limit. This has made the accurate prediction of thermal performance imperative without much recourse to rely on a "blow it and see approach". Most of the analytical techniques to date have invoked classical heat transfer correlations intended for circular tubes and flat plates. Convective heat transfer in electronic equipment is an empirical science requiring the ability to translate real - life designs to idealized configurations. This ability is gained by experience since there are considerable differences between the condition treated in text books and the actual thermal and geometrical conditions in electronic equipment. Thus, analysis and predictions of cooling of electronic equipment has been an art as much as a science.

It is noteworthy however, that the lack of generalized analytical data is an indication of the effectiveness, flexibility and developmental potential of past equipment designs. It is also important to realize however, that, with the present trends in packaging and higher thermal density, the designer may not be able to afford the luxury of such flexibility. Therefore, the suitability of a cooling technique must be assessed at an early stage of the design itself, so that alternative cooling schemes may be considered, if necessary. In addition to higher heat loads and system complexities, the need for faster design cycles has emphasized the need for better prediction techniques. Because of the lack of generalized thermal design data, most heat transfer work related to electronic equipment cooling has been traditionally done on a system by system basis. Although experimental work related to electronic equipment is an absolute necessity, the almost unlimited number of variables involved makes a full experimental study extremely time consuming, and often untractable. There is therefore a conscious effort to improve the ability to predict thermal performance for a wide variety of conditions.

In the following sections, we will review the thermal design process of an electronic system that is air cooled. The thermal design process will begin at the overall system (or equipment) level and systematically proceed to lower levels leading to individual devices or packages. Although the

example pertains to an air cooled system, the procedure and philosophy of such a systematic study may be used for systems that use other cooling techniques as well. The basic principle of such a systematic thermal design is to always keep the cooling problem at hand in a simple and tractable form. This is accomplished by breaking up the problem into convenient sub - sections, and then using the results from one sub - section as boundary conditions or input to the next one. In this way, no attempt is made to solve the problem in its entire glorious (and very complicated) detail, but in a few simpler steps that are easier to deal with. This step wise procedure is followed until we can predict the junction temperatures of individual devices. This is the information that the reliability engineer is ultimately interested in for reliability predictions. Such an approach is a very convenient tool for evaluation of thermal performance of new equipment at an early stage in design.

Fig. 4 shows the hierarchy of thermal analysis for electronic equipment.

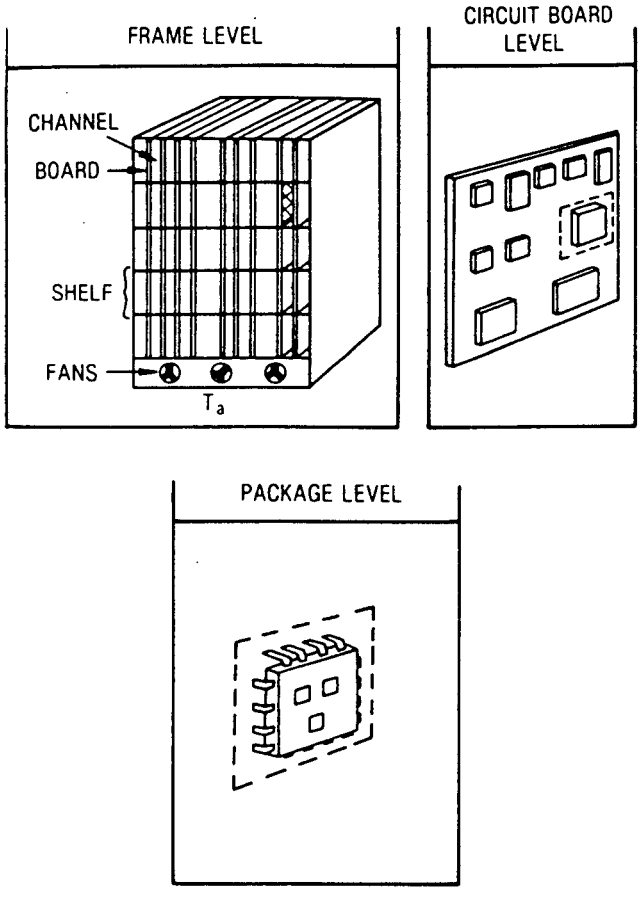


FIGURE 4. HIERARCHY OF ELECTRONIC EQUIPMENT THERMAL ANALYSIS.

The system shown in the figure consists of several shelves (say from 2 to 6) that are stacked on top of each other. Each shelf contains circuit boards that are plugged into card guides, and the boards may be spaced between 1/2 inch to 1 inch apart. In addition, several circuit board slots may be vacant and the boards at different levels may be staggered. On each board are mounted components (devices) that have different geometric features and power dissipations. The components densities may be different in each board. For structural rigidity, the circuit boards are usually supported by two or three cross support plates that run the entire width of the equipment. This system is forced air cooled by a set of fans as shown, or they could be cooled by air using natural convection. This is the typical geometrical layout for telecommunication equipment in which the circuit boards may be 4 to 10 inches high. For forced convection cooling, the front of the equipment is covered, while for natural convection cooling, the front may be open or closed. For natural convection cooled systems, the boards may be 4 to 5 inches high.

In any electronic equipment, conduction, convection and radiation modes of heat transfer are present. Heat that is generated by the chip is dissipated by conduction to the case of the component, and from the case to the circuit board through the pins. The heat is then conducted through the circuit board. The heat conducted to the component case and through the circuit boards is carried away by the air flowing over the boards and components. Finally, radiation heat transfer takes place between a hot component to the adjacent board. In natural convection cooling, radiation often acts as nearly an equal partner to dissipate heat.

Since a single comprehensive thermal analysis of an entire equipment is impossible for most systems of practical interest, the thermal design process is divided into three steps as outlined below:

(1) The first step is called the equipment or frame level, and is an analysis at the global level. At this level, the designer has information pertaining to the entire system, such as cabinet dimensions, circuit board height, width and spacing, power dissipation and the design constraints, such as maximum allowable board and case temperatures. This level of analysis enables the designer to ascertain the system cooling requirements such as air flow rates and velocities between boards, and the mixed mean air temperatures. At this point, if forced air cooling is necessary, an appropriate fan must be chosen. The combination of the fan performance data with the system characteristics will provide an estimate of the air flow rates, velocities and temperatures within the equipment. The system level therefore provides a global estimate of cooling requirements and conditions.

(2) The information obtained from the system level

analysis is now used for the next step which is called the circuit board level. This level of analysis enables the designer to obtain detailed information regarding one or more circuit boards. The information required by the designer for analysis at this level are the circuit board spacing and thermal properties, component layout on the board, component geometry and power dissipation. The results obtained are the component case temperatures, board temperature profile and air temperature variation within the channel formed between circuit boards. This level of analysis may be done individually or for several circuit boards. The usefulness of this level of analysis is to warn the designer of potential hot spots on the circuit boards and suggest alternate ways of laying out devices. Also, as will be seen later, the board temperature beneath a device is an important parameter in determining the junction temperature of devices.

(3) The final level of thermal design is called the component level in which an isolated component is studied to determine the chip junction temperature. This level of study may usually be done for selected high heat dissipating components to determine if the junction temperatures are within the maximum allowable limits. If not, more air flow using larger fans and/or heat sinks on components are necessary to reduce chip junction temperatures. It is important to note that any system changes requires that the step by step thermal design process be evaluated again, since the system flow characteristics are altered. We will now discuss the 3 thermal design steps in detail.

SYSTEM LEVEL THERMAL DESIGN

We will now go through the thermal design process for the type of equipment outlined earlier in Fig. 4. The system is to be cooled by air and the designer is required to work under certain constraints, such as junction temperature $T_j < 85\,^\circ C$,maximum circuit board temperature $T_b < 70\,^\circ C$, and the maximum ambient air temperature $T_a = 50\,^\circ C$. Also in forced convection using fans, the noise level should be less than 70 db at a distance of 3 feet from the fans. As a starting point, Fig. 5 is a useful tool in ascertaining the limits of several modes of cooling that may be used for electronics [5].
This figure shows the temperature difference between the heat dissipating surface and the ambient cooling medium for a range of heat fluxes. The limits of different cooling techniques may be determined from this.

Natural Convection (Air)

Many electronic equipment where heat dissipations are low, still use natural convection air cooling. It has the advantage of low cost and negligible maintenance. Its disadvantage is the low

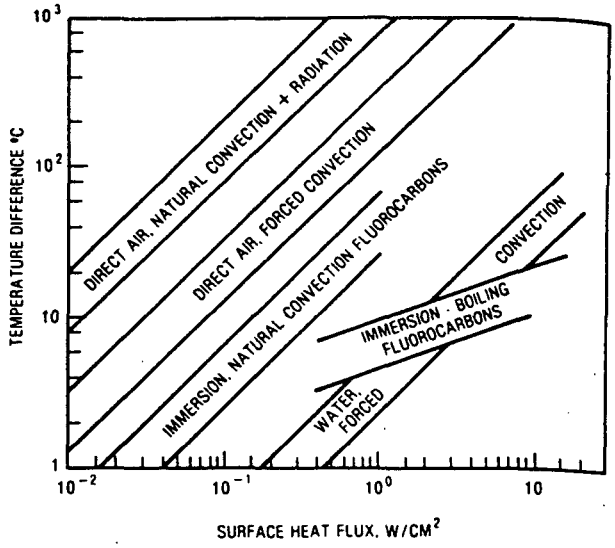


FIGURE 5. TEMPERATURE DIFFERENCE VERSUS HEAT FLUX FOR VARIOUS MODES OF COOLING.

heat load that can be effectively cooled. Since natural convection air cooling has been used extensively, there are several articles on the subject [9 - 17]. Most of the studies on natural convection cooling for electronic equipment are based on uniform heat dissipating channels formed by circuit boards. This is a valid assumption because the copper tracks in the board assist in spreading the heat from various components uniformly resulting in a constant heat flux surface. Heat is dissipated almost equally from both sides. The variables in the problem are the board spacings, component heights, channel height and heat dissipation rates. For this application, the principle non - dimensional parameters are the modified Rayleigh number Ra'' and the Nusselt number Nu which are derived in [5] and are defined below:

$$Ra'' = \frac{q'' \beta g s_e^5}{L \nu \alpha \kappa} \qquad (6)$$

$$Nu = \frac{q'' s_e}{\kappa (T_b - T_{a0})} \qquad (7)$$

where s_e represents the effective flow width between channels and is obtained by subtracting the circuit board width and the average height of components from the center line distance between adjacent boards, L is the entire length of the free convection channel and interruptions by circuit board shelves are

ignored, and T_{a0} is the ambient air temperature entering at the bottom of the channel. q'' is the heat flux from the board. Natural convection correlations for electronic equipment are reviewed by Johnson [17] in detail, and the appropriate design recommendations are also provided. The correlations are summarized below:

Aung et al [9,10]

For narrow or tall channels where the flow is fully developed

$$Nu = 0.144 (Ra'')^{0.50} \quad \text{for } 0 < Ra'' < 50 \qquad (8)$$

For wide or short channels (approximating a single plate)

$$Nu = 0.524 (Ra'')^{0.20} \quad \text{for } Ra'' > 700 \qquad (9)$$

Wirtz and Stutzman [12] for $3 < Ra'' < 10^6$

$$Nu = \frac{0.144 (Ra'')^{0.50}}{[1 + 0.0156 (Ra'')^{0.9}]^{0.33}} \qquad (10)$$

Bar-Cohen and Rohsenow [14] for $1 < Ra'' < 10^6$

$$Nu = [\frac{48}{Ra''} + \frac{2.5}{(Ra'')^{0.40}}]^{-0.50} \qquad (11)$$

Birnbrier [15]

$$Nu = 0.20 (Ra'')^{0.31} \quad \text{for } 300 < Ra'' < 10^6 \qquad (12)$$

Johnson [17] recommends the universal correlations of Wirtz and Stutzman [12] and Bar-Cohen Rohsenow [14] based on comparison with experimental data collected at AT&T Bell Laboratories. In the $10 < Ra'' < 10^4$ range, which is the region of practical interest, these two correlations are equally accurate. In the range of $Ra'' < 10$ and $Ra'' > 10^4$, equations (8) and (9) respectively of Aung [9,10] are recommended. Given a power dissipation, circuit board spacing, component height and channel heights, these correlations enable the designer to determine the maximum board temperature.

Forced Convection (Air)

Since natural convection air cooling is limited to cooling low thermal loads, many of higher performance systems which have more dense packaging of devices, rely on forced air cooling to provide proper thermal control. As we can see from Fig. 5, forced convection cooling can remove 5 to 10 times the amount of heat than natural convection. The use of forced convection cooling depends on the choice of an appropriate fan for the system that will provide the required flow rate. Selection of a fan depends upon the pressure and flow characteristics, size and weight, acoustic noise, life and cost [34]. Fans are generally of two types, axial flow and centrifugal flow fans. Axial flow fans are generally used for electronic cooling applications because of their relatively small sizes and weight compared to centrifugal fans. The two types of axial flow fans commonly used for electronic cooling applications are the tubeaxial and the vaneaxial fans. Tubeaxial fans are used extensively in commercial applications and are light, have a very long life, make less noise, and are low in cost. Such fans used for commercial applications typically provide less than 200 cfm at a pressure head of less than 0.5 inch of water. Vaneaxial fans are capable of providing flows at much higher pressures, say up to 5 inches of water for the same flow rates. This is accomplished by operating the vaneaxial fans at very high speeds. Consequently, the high performance of vaneaxial fans is obtained at the expense of very high acoustic noise, higher power, and very short life. Also, operation of the vaneaxial fan requires 400 cycle power. A very interesting comparison between a tubeaxial and a vaneaxial fan is given in [34], where both fans deliver 160 cfm at zero back pressure. The tubeaxial fan has a pressure capability of $\approx$ 0.3 inch of water, weighs 1 pound, uses 14 watts of power, and generates an acoustic noise of 45 dB. The vaneaxial fan has a pressure capability of 4 inches of water, weighs 1 pound, uses 130 watts of power, and generates an acoustic noise of 68 dB. The expected life of the tubeaxial fan is 10 years, while that of the 400 cycle vaneaxial fan is 5000 hours (less than seven months). The often stated complaint of fans not being reliable, perhaps has its origin in the choice of a vane axial fan for cooling electronics. There are a wide variety of fans available, ranging from small flow rate axial fans to centrifugal blowers of very large capacity. The biggest limitation on fan selection is perhaps noise, since many electronic equipment such as computers and telecommunication equipment are used in an office environment. Fan power may also be another limitation in the selection.

Choice of an appropriate fan requires the knowledge of the fan performance characteristics as well as the system pressure drop characteristics. The fan performance characteristics are provided by the manufacturer. Fig. 6 shows a typical fan performance characteristic where the fan static pressure Δp (in inch of water) is plotted against the volumetric flow rate Q (in cubic feet per minute, cfm).

Super - imposed upon this fan curve are three different curves labelled 1,2 and 3 called system curves. The system curve is a measure of the system impedance to air flow and is a unique characteristic of the system. Electronic equipment usually have complex geometries consisting of contractions, expansions, bends, obstructions, plenums, orifices and filters. There is pressure loss as air goes through

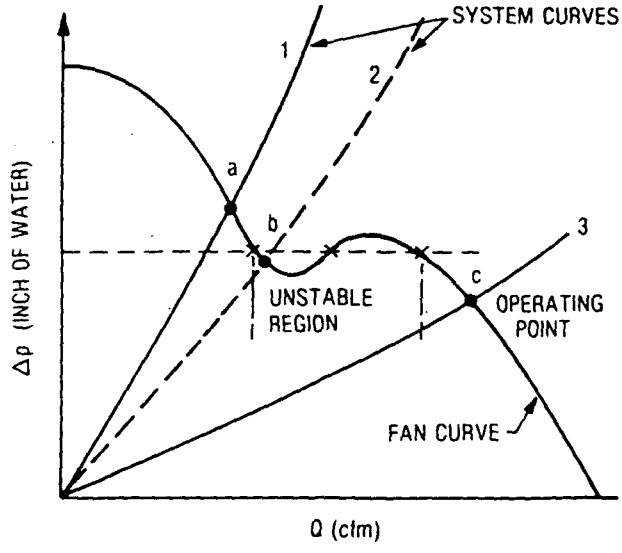


FIGURE 6. TYPICAL FAN PERFORMANCE CURVE.

these geometrical variations and over the boards and components. This pressure loss is dependent upon the velocity (and hence flow rate) and is represented by the system curve. Any geometrical changes, such as board spacing or component height will alter the system curve. The intersection of the fan and system curves gives the operating point of the particular fan - system combination. From this operating point, the designer can determine the system pressure drop and the air flow rate from the fan.

Most large capacity axial flow fans show a characteristic hump in the performance curve as shown in Fig. 6. The best point of operation for the fan is the intersection with system 3 at point c, beyond the hump of the fan curve. There are two reasons for operating the fan in this region. Firstly, the flow rate is large, and secondly, this is also the region of maximum fan efficiency or least pumping power. Fan pumping power is given by Q. Δp. At the point a, the flow rate delivered by the fan is small and is therefore not a good choice for cooling system 1. The most important aspect of fan selection is to avoid a point of intersection such as b, in the hump region of the fan curve. The hump region of the fan curve represents an unstable portion of fan performance. For a single static pressure corresponding to the intersection point b, there are 3 possible flow rates, given by the intersection of this constant pressure line with the fan curve. The fan flow rate will therefore oscillate between these 3 values and is an unstable operation. This instability leads to increased fan noise and to rapid fan failure. Such an operation can be detected by increased fan noise. Obviously, for reliable fan operation and forced convection cooling, an operating point such as c is desirable.

Many electronic equipment using forced convection cooling require more than one fan to provide the flow. These fans may be connected in series or in parallel. From the flow characteristics of a single fan, it is easy to construct the performance characteristic for several fans in series or in parallel. For two or more fans connected in series, the total volume flow rate remains the same, while the overall total pressure will be the sum of the individual fan total pressures. Thus, fans in series are useful where the the system curve has a large pressure drop for small flow rates (high system resistance). Only a series arrangement can give an increase in flow while overcoming the higher pressure loss. However, for two or more fans connected in parallel, the total pressure difference across each fan will be the same, but the total volume flow rate will be the sum of the individual volume flow rates for each fan. Thus, fans in parallel operation are useful where the system pressure drop is low, and a substantial flow increase can be achieved. The characteristics for two fans in series and in parallel are shown in Figs. 7 and 8 respectively [4].

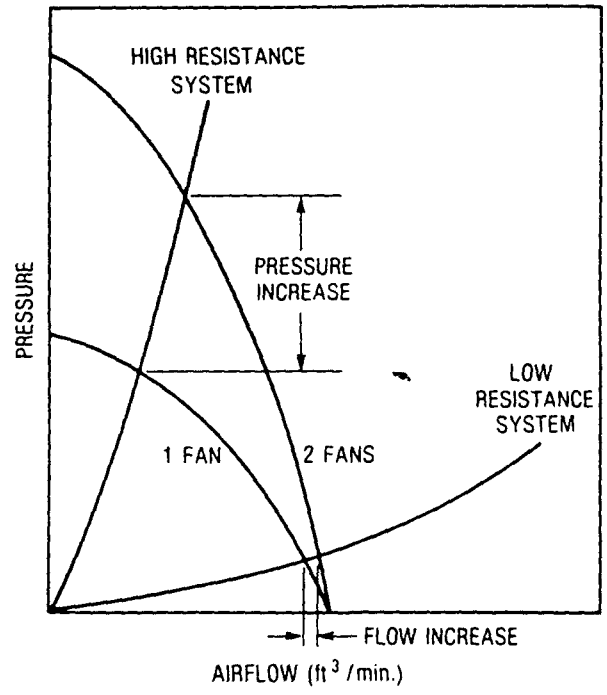


FIGURE 7. PERFORMANCE CURVES FOR 2 FANS IN SERIES. [4] Reprinted by permission. Copyright © 1984 by Van Nostrand Reinhold Company Inc.

It is important to remember that if there is an unstable region in the fan performance curve, connecting fans in parallel may spread this region over a wider flow rate. If the system curve passes through the unstable region of a parallel fan system, different fans may operate simultaneously at different

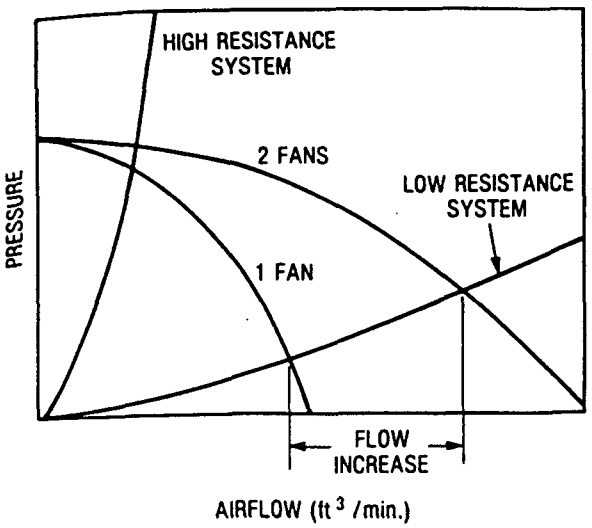


FIGURE 8. PERFORMANCE CURVES FOR 2 FANS IN PARALLEL. [4] Reprinted by permission.

operating points, once again leading to fan noise and rapid fan failure. The physical principles, design and applications of fans are covered in detail by Osborne [18].

To chose a fan for cooling a particular equipment, the first step in the process is to determine the system characteristic curve. The best way to determine this is by measurement. However, pressure loss predictions are often required at the design stage itself. Simple analytical and numerical formulations based on fundamental fluid dynamics principles are extremely useful and fast in estimating system curves. Pressure losses associated with flow through any system are of two distinct types. These are frictional losses and dynamic losses. Frictional losses occur due to flow over surfaces such as plates and tubes, and the pressure drop associated with this is given by

$$\Delta p = \frac{4 \rho f L V^2}{2 g_c D_h} \qquad (13)$$

The friction factor f is given in terms of the Reynolds number $Re = \frac{V D_h}{\nu}$ and flow surface conditions, and may be obtained from any text on fluid dynamics. D_h is the hydraulic diameter.

Dynamic losses on the other hand result from changes in velocity in the system. Such velocity changes can occur due to bends, orifices, flow area changes such as contractions and expansions, filters and obstructions. In fluid mechanics literature, these are sometimes referred to as "minor" losses. However, the geometrical variations and complications mentioned above that cause dynamic

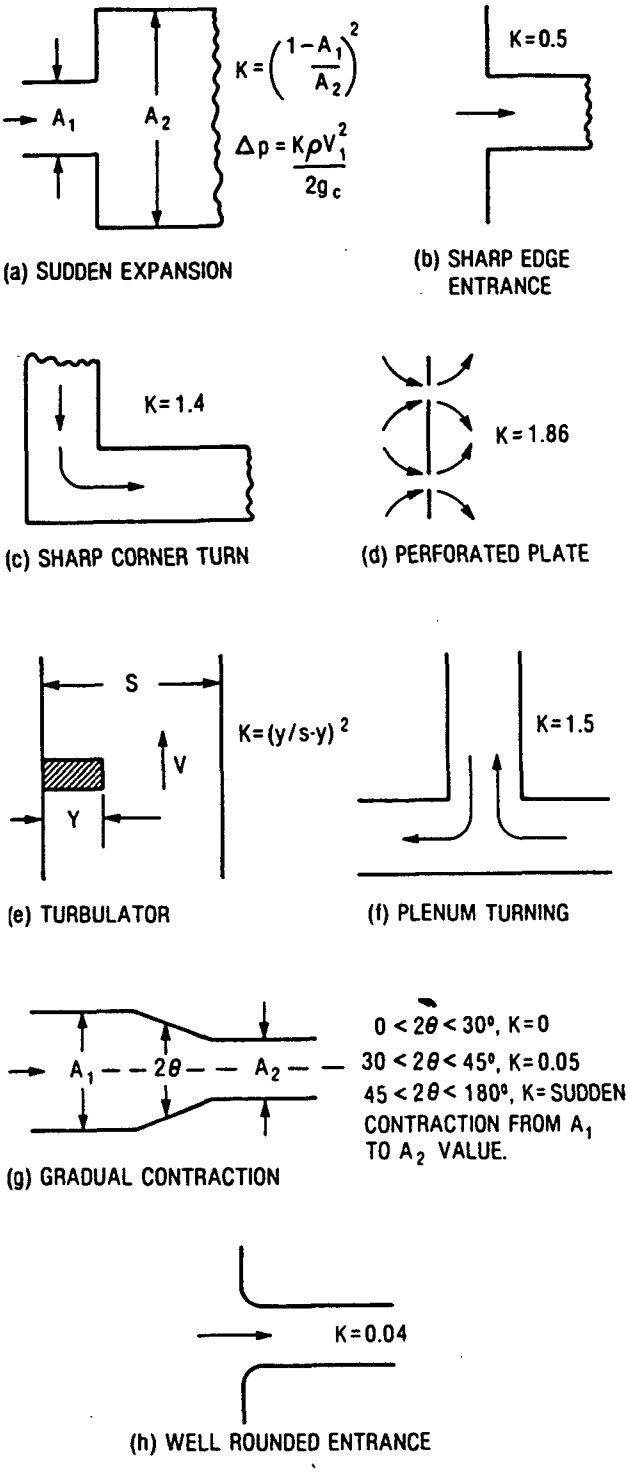


FIGURE 9. LOSS COEFFICIENT K FOR VARIOUS GEOMETRIES.

losses in a flow system, are common features in most electronic equipment designs. They are in fact the major factor in pressure losses through equipment, and frictional effects are secondary. Dynamic losses are expressed as

$$\Delta p = K \frac{\rho V^2}{2 g_c} \qquad (14)$$

where K is called the loss coefficient which is determined experimentally for most cases. Values for some cases are shown in Fig. 9.
Expansion and turn losses are always high, and so are flow through perforated or slotted plates. A smooth entrance has a much smaller pressure loss than a sharp edged entrance. Also, a gradual contraction is preferable to a sudden contraction. The total pressure drop is the sum of the frictional and dynamic effects. The pressure loss for flow through circuit boards, with turbulators and card guides is given by Wills [19] as

$$\Delta p = \rho V^2 \left[\frac{0.033 L}{s^{1.3} V^{0.30}} + 1.9 \sum K \right] \qquad (15)$$

This is an experimental correlation in the air velocity range of 0.20 to 8 m/sec (40 to 1575 ft/min). In equation (15), ρ is the density (Kg/ m^3), V is the average velocity based on the minimum flow cross - sectional area (m/sec), L is the streamwise circuit board length (m), s is the circuit board pitch (m), and Δp the pressure in N/m^2 (1 N/m^2 $\approx$ 0.004 inch of water). If card guides or turbulators are not used, then $\sum K = 0$. This experimental correlation was obtained for circuit board pitch in the range of 10 to 20 mm. Note that the component heights are implicitly considered in the evaluation of V.

In order to construct a model of the equipment to determine system pressure losses, it is necessary to identify the different components of the losses, such as bends, contractions, expansions, obstructions, etc. Once this is accomplished, a flow resistance network for the system may be formed analogous to the electrical resistance network. This network model may be solved numerically using standard iterative techniques. Because of the tremendous geometrical complexities inherent in electronic equipment, it is almost impossible to solve the Navier - Stokes equations to obtain pressure loss and flow information. The flow resistance network (lumped parameter approach) model is by far the simplest and quickest technique to obtain useful information regarding system pressure loss and average mass flow rates and velocities. In this lumped parameter flow resistance model, the flow region is divided into a set of discrete nodes, and the nodes are connected together by appropriate flow resistances. The resistance network model for a test cabinet is shown in Fig. 10 [20].

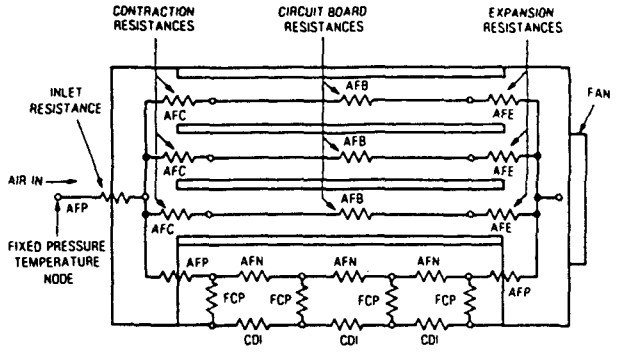


FIGURE 10. RESISTOR NETWORK FLOW MODEL. [20]

This fan cooled cabinet consists of three circuit board channels and a shielded power supply unit along one side. The fan is used to pull air through the system. The air enters the front of the equipment through a perforated inlet and branches into four parallel paths. Three of these air flow paths are between the circuit cards and the fourth is through the power supply chamber. The air from these paths then collects at the fan inlet and is drawn out of the equipment. The major flow resistances to air flow are the following:

Inlet resistances through the perforated front panel are shown as AFP. The contraction resistances from the front plenum to the circuit card channels are represented by AFC. Resistances due to flow between circuit boards are AFB. Expansion resistances from the circuit boards to the rear plenum are AFE. Perforated inlet and exit resistances to and from the power supply chamber are also shown as AFP. AFN are resistances for flow over power supply unit. Also shown in this figure are some thermal resistances. Resistances CDI are thermal conduction resistances in the side panel of the equipment due to the power transistors mounted on it. Resistances FCP are forced convection thermal resistances for convection from the side panel to the power supply air stream. This simple model shows how both flow and thermal effects may be represented by a single model which can simultaneously solve for both pressure (air velocity) and temperature. The equation in matrix form to determine the pressure p is given by

$$p_i \sum_{j=1}^{n} \frac{1}{R_{ij}} - \sum_{j=1}^{n} \frac{p_j}{R_{ij}} = S_i \qquad (16)$$

where R_{ij} is the flow resistance and S_i is the mass source term which is present for fan nodes only. Solution of equation (16) provides the pressure loss for various system flow rates which may be used to determine the system curve. Equation (16) is a mathematical representation of mass balance at a node i connected to nodes 1,2,3,.... n as shown in Fig. 11.

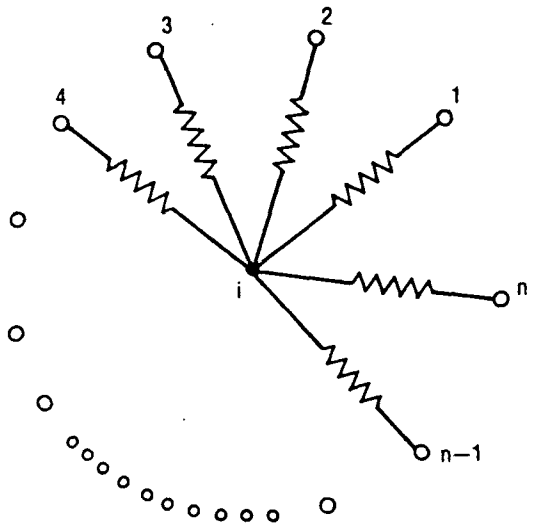


FIGURE 11. RESISTANCES CONNECTING NODES i TO n.

As in any resistor network, the mass flow rate $\dot{m}_{ij}$ in a resistance connecting nodes i and j as shown in Fig. 12, with flow from i to j may be expressed as

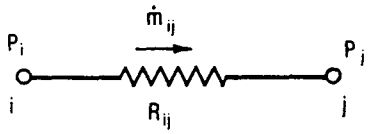


FIGURE 12. FLOW RESISTANCE CONNECTING NODES i TO j.

$$\dot{m}_{ij} = \frac{p_i - p_j}{R_{ij}} = \frac{\Delta p_{ij}}{R_{ij}} \qquad (17)$$

Making use of the equation for mass flow $\dot{m} = \rho A_c V$, and the expressions for friction and dynamic pressure losses from equations (13) and (14), we may express the flow resistance as

$$R_{ij} = \left[\frac{1}{2 g_c} \left(K + \frac{4 f L}{D_h} \right) \frac{\Delta p_{ij}}{\rho A_c^2} \right]^{1/2} \qquad (18)$$

Note that the flow resistance is a non - linear function of the pressure drop Δp_{ij} itself, and an iterative procedure is required to solve equation (16). Because of this non-linearity, the pressure drop solution requires more iterations to converge than a comparable thermal network problem. Also note that in Fig. 12, flow is from i to j and $p_i > p_j$. This is true for any flow resistor except the fan. If the resistor shown in Fig. 12 represents a fan, then for flow from i to j, $p_j > p_i$, and the effective fan flow resistance must be evaluated from the performance curve.

Electronic equipment are often required to function reliably at different altitudes. To maintain the same temperature levels within the equipment at higher altitudes, we need to maintain the same mass flow rates. Since density of air decreases with altitude, the same mass flow rates can only be maintained with higher flow velocities. However, the pressure losses increase with increasing velocities. Therefore, proper analysis is required to determine the effects of altitude variations (and density) on fan and system characteristics. For this purpose, a density ratio $\sigma = \frac{\rho}{\rho_0}$ is used, where the subscript 0 refers to standard conditions. The system characteristic may be expressed as

$$\sigma \Delta p = C \dot{m}^n \qquad (19)$$

where C is a constant. The value of n varies from 1 for laminar flows to 2 for fully turbulent flow. For electronic equipment where dynamic losses are predominant, n is closer to 2, and may be 1.8 to 1.9. When losses are predominantly due to frictional effects, such as flow in compact core heat exchangers, n is closer to 1. For the case where n=2, a same fan - system combination will provide the same volume flow rate (cfm) at all altitudes as shown in Fig. 13 [21].

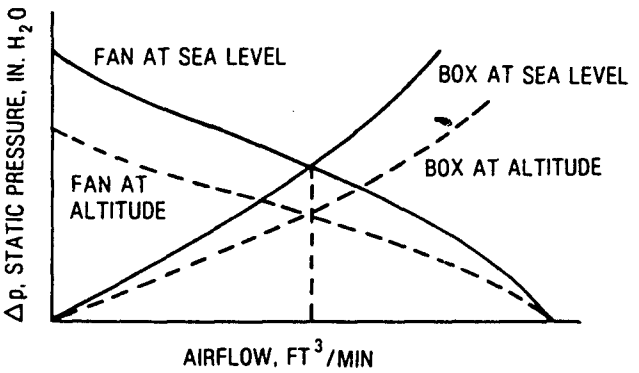


FIGURE 13. FAN AND SYSTEM CURVE OPERATING POINT CONSTANT FOR n = 2. [21]

The mass flow rate will of course be reduced by the factor σ. However, for $n \neq 2$, the volume flow rate (cfm) will not be the same at all altitudes. Since Δp varies inversely with density, a plot of σΔp versus $\dot{m}$ provides a single system curve for all altitudes. This single curve may then be effectively used with different fan curves to determine the effects of altitudes and obtain operating points as shown in Fig. 14 for a specific system [21].

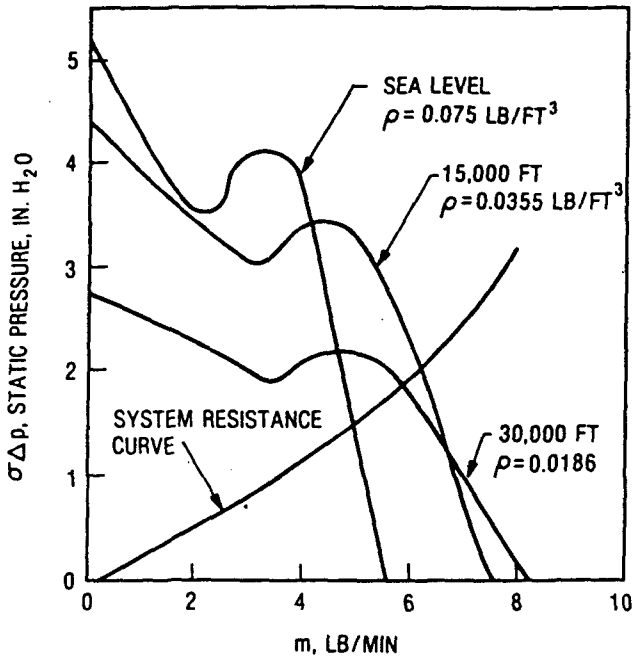


FIGURE 14. SINGLE SYSTEM CURVE MATCH POINT FOR DIFFERENT DENSITIES. [21]

In a similar way, we may also derive a fan characteristic that may be represented by a single curve for all altitudes for constant fan speed. It is obtained by plotting $\frac{\Delta p}{\sigma}$ versus volume flow rate Q for the fan. This is obtained directly from the fan laws, since Δp is proportional to ρ for a fan with fixed speed and diameter. This single curve is useful for matching a specific fan curve with different system curves, which change with altitude as shown in Fig. 15.

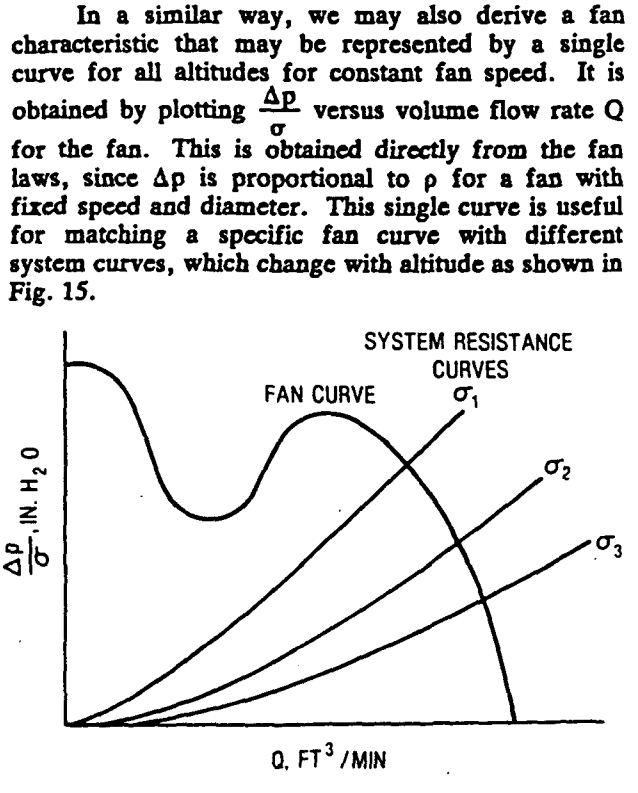


FIGURE 15. SINGLE FAN CURVE MATCHED WITH CHANGING SYSTEM CURVES. [21]

It is important to determine the system curve accurately since this has a major effect on the air mass flow rates at higher altitudes. Fig. 16 shows two system curves with n = 1.5 and n = 2 matched against a fan at different altitudes [21].

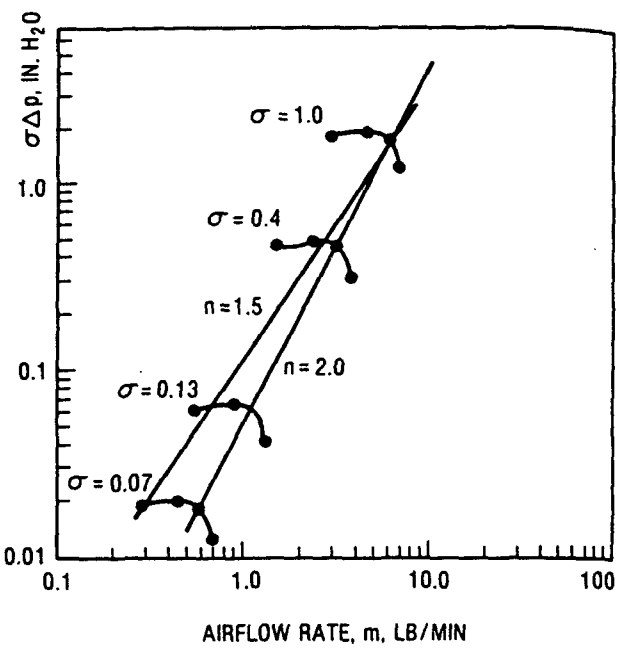


FIGURE 16. OPERATING POINTS FOR DIFFERENT DENSITIES FOR n = 1.5 AND n = 2. [21]

For σ = 0.13 (altitude of 55000 feet), the air flow rate for n = 2 is 1.1 lbm/min and that for n = 1.5 is 0.70 lbm/min. Also, at higher altitudes, the difference in mass flow rates between n = 2 and n < 2 becomes larger which underscores the importance of the system curve. Some fans designed for airborne applications have variable speed motors that increase the fan speed as the density decreases, thereby partly compensating for the reduced mass flow at higher altitudes.

Most electronic equipment require a filter to keep out dust, especially when ambient air is used for cooling. Pressure drop characteristics of the filter should be known, and are usually provided by the manufacturer. The filter is usually a major source of pressure loss in the system and should not be neglected. Choice of proper fan position is also of importance for some systems. If the fan is made to pull air through the system, then the power dissipated by the fan is absorbed by the air only after cooling the system. Also, the air flow distribution is usually more uniform when air is drawn through. However, in a system in which air is pulled through, the equipment will be at a negative pressure inside. If

there are any cracks or openings in the equipment frame, there is the risk of dust leaking into the equipment. This may be a serious concern for high voltage equipment where dust can cause sparking inside.

BOARD LEVEL THERMAL DESIGN

Intersection of the fan and system curves gives the operating point for the system. From this we obtain the air flow rates and average velocities through various sections of the equipment. As a first step, these air flow rates may be used to obtain air temperatures at different portions of the equipment as it picks up heat from the devices. Thus, the ambient temperatures around devices are defined. In order to do board level thermal analysis, knowledge of the convective heat transfer coefficient h is required. The average flow velocities calculated from the pressure drop analysis is used for this purpose. Accurate prediction of heat transfer coefficients for electronic cooling applications is a very difficult task. The reasons for this are the complex geometries, obstructions, bends, plenums etc. that are commonly encountered. The approach for analysis has been to use heat transfer correlations in literature for flat plates and tubes and then use "engineering judgement" for the equipment under consideration. Therefore thermal studies were generally done experimentally on a system by system basis. However, more general heat transfer data for forced convection air cooled circuit boards are becoming available because of its importance. Wills [19] presented useful convective heat transfer correlations which included some of the factors such as board spacing, air velocity, component dimensions, packing density and position relative to the air stream flow direction, and card guides. The correlation is based on air flow over an array of uniformly powered ceramic DIPs soldered on to circuit boards whose pitch varied from 10 to 20 mm. Experiments were performed for velocities from 0 to 8 m/sec (0 to 1575 ft/min), and heat transfer correlations were calculated from DIP case to ambient thermal resistance measurements.

For circuit boards fitted into card guides

$$h = 5.3\,\frac{b}{d} + \frac{6.2\,(\rho V)^{0.80}}{(N d)^{0.36}}\left(\frac{d}{b} - 1\right)^{0.13} \qquad (20)$$

In equation (20), b and d are in m, ρ is in kg/m^3, V in m/sec, and h is in w/m^2 °C. N is the component number in the flow direction, and is an indication of the boundary layer development from air entrance to the boards. Thus, h for the first component (N=1) is larger than components downstream (N=2,3,4,etc.) as the boundary layer develops. These variables are defined in Fig. 17.

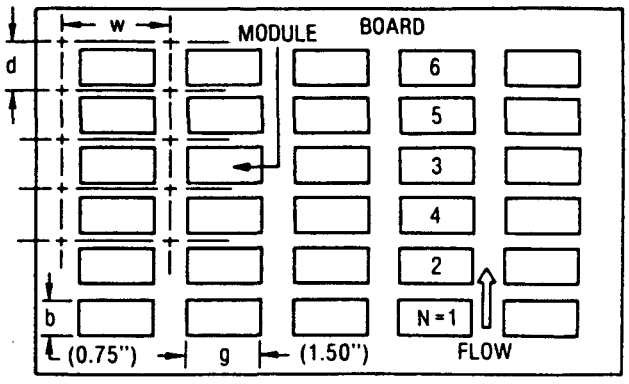


FIGURE 17. DEFINITION OF BOARD AND COMPONENT VARIABLES.

Equation (20) is independent of board spacing s and one equation is provided without any transition from laminar to turbulent flow. The experimental data agreed with the correlation to within 5%. For high velocity range, the heat transfer correlation is given in a more familiar form as

$$Nu = 0.19\,Re^{0.7}\left(\frac{d}{b} - 1\right)^{0.13} Pr^{1/3} \qquad (21)$$

where Nu and Re are based on the distance from the circuit board leading edge. Equations (21) yields values of h that are typically twice those obtained from the flat plate correlation

$$Nu = 0.0288\,Re^{0.8}\,Pr^{1/3} \qquad (22)$$

Without card guides at the leading edge, the heat transfer correlation is given by

$$h = 5.3\,\frac{b}{d} + \frac{7.6\,(\rho V)^{0.80}}{(N d)^{0.2}}\left(\frac{d}{b} - 1\right)^{0.13} \qquad (23)$$

Equations (20), (21) and (23) given above relate to the first level of circuit boards. If the electronic system consists of several levels of circuit boards as shown in Fig. 4, then the flow development breaks up as air enters each new level. The thermal boundary layers begin to develop again causing the heat transfer coefficient to increase as air enters the new level. Therefore, the component and board temperatures at the bottom of a new level may be expected to be lower than those at the top of the previous level, even though the air temperature has increased. The study also found that the heat transfer coefficient was the same from the circuit board and the components for the same streamwise location. In the above correlations, V is the average flow velocity based on the free flow area between circuit boards. Thus, the component height.t is implicitly built into the correlation.

Heat flows from the component case to the board through the pins, and the boards effectively act as fins to dissipate the heat which is eventually carried away by the air flowing over it. Heat dissipation into the boards is an important factor because of the presence of circuit paths and/or ground and power planes. The presence of copper increases the effective thermal conductivity of the board. Experience shows that the heat dissipated by the devices is nearly divided equally between convection from the component cases and convection from the boards themselves. This important factor of heat dissipation into and through the boards is taken into account in the correlations given by Wills [19] above. The equivalent thermal conductivity of epoxy boards depends upon the amount of copper present in it as shown in Table 3 [22].

Table 3. Equivalent Thermal Conductivities Of Epoxy Boards [22]. Courtesy of International Society Of Hybrid Microelectronics (ISHM).

Material	Thermal Conductivity (W/m °C)
Epoxy Glass (No Copper)	0.294
Epoxy Glass (1 oz. Copper)	9.11
Epoxy Glass (2 oz. Copper)	17.71
Epoxy Glass (4 oz. Copper)	35.13

Table 4 [22] shows the thermal conductivities of other materials used in circuit boards and substrates.

Table 4. Thermal Conductivities Of Board Materials [22]. Courtesy of ISHM.

Material	Thermal Conductivity (W/m °C)
Alumina (90%)	16.7
Beryllia (99.5)	168
Copper	386
Epoxy Glass	0.294
Gold	314
Kovar	16.7
Silicon	125
Solder (60/40)	49.3

Forced convection heat transfer from arrays of flat blocks and ribbed surfaces have also been studied by Sparrow et al [23], Wirtz and Dykshoorn [24] and Lehmann and Wirtz [25]. [23] is an experimental study of heat transfer and pressure drop for forced convection air flow over arrays of heat generating rectangular modules mounted on one wall of a channel to resemble components mounted on a board. Results were obtained for fully populated arrays, selectively missing modules, and enhancements due to barriers (turbulators). Results were obtained for t/b = 0.375, (d-b)/b = 0.25 and S/b=1. The ratio of turbulator height to board spacing was 0.50 and 0.625. Heat transfer results were deduced from its analogy to mass transfer using naphthalene sublimation techniques. They found that the heat transfer coefficient remained fairly constant after five rows of modules, indicating a fully developed region. In this region the heat transfer correlation obtained for Re = 2000, 3700 and 7000 is

$$\frac{h\,b}{\kappa} = Nu = 0.0935\,Re^{0.72} \qquad (24)$$

where $Re = \frac{V(s-t)}{\nu}$, and V the average velocity based on the minimum free flow area between boards. The effect of a missing module is to enhance the heat transfer coefficient from the neighboring modules. The largest enhancement occurs on modules where there is a missing module just upstream of it, and this may be as large as 40%. Modules that lie just upstream of the missing module may only get an enhancement of between 8 and 20%. The enhancement is larger at lower Re because of flow recirculation and penetration into the cavity, while at higher Re, the flow skims the surface. The presence of turbulators also enhances the heat transfer coefficients on modules that are two rows downstream substantially, from 39 to 100%. This is essentially due to flow separation and reattachment. The correlation given by equation (24) and the monotonic pressure drop results obtained show no distinct transition from laminar to turbulent flow, as was also observed by Wills [19].

Wirtz and Dykshoorn [24] report similar results for forced convection air velocity over uniformly heated array of square modules, 5 columns wide and 8 rows deep in the flow direction. For these studies, t/b=0.25, (d-b)/b=1.0, and S/t varied from 1.25 to 4.62. They found that except for the first row of modules, where the heat transfer coefficient was 15 to 20% higher,modules in the remaining rows had a constant value. The channel wall spacing varied from 9.5 mm (0.375 inch) to 29.2 mm (1.15 inch) and the air flow velocity ranged from 1 to 10 m/sec (197 to 1970 ft/min). Their heat transfer results may be correlated by the equation

$$\frac{h\,b}{\kappa} = Nu = 0.348\,Re^{0.60} \qquad (25)$$

where $Re = \frac{Vb}{\nu}$ is based on the free stream approach velocity based on the area between the boards before the modules. The heat transfer coefficients given by equation (25) are slightly higher than that obtained from equation (24). This is because the modules were more densely packed in [23]. Larger spacings between components allows for

more penetration and recirculation in the cavity, which translates into higher heat transfer coefficients. This study also found that the effect of wall to wall distance on module heat transfer coefficient was negligible. This was also observed by Wills [19] in his studies with DIPs. They also state that the heat transfer correlation of Arvizu and Moffatt [27] in terms of an array velocity is not the best approach, since this shows a board spacing dependence, which is actually negligible.

In [25], Lehmann and Wirtz studied forced convection from a two dimensional channel in which one surface was fitted with repeating ribs to represent component projections from a board. Both flow visualization in the channel and heat flux measurements from a heated rib using a Mach - Zender interferometer were used. The channel had 12 equally spaced ribs, and only the eleventh rib in the flow direction was heated. The region containing the last 3 ribs was the test section, which was in the fully developed region. The rib geometry was fixed with b = 50 mm, t = d-b = 12.5 mm and the channel wall spacing s varied from 25 to 75 mm. Experiments were conducted for the range of $667 < Re < 3000$ and $0.5 < s/b < 1.5$. Here, Re is defined in terms of the minimum free flow height and average velocity as $Re = V(s-t)/\nu$. In addition, a Reynolds number based on rib (component) length b defined as $Re_b = \frac{V b}{\nu}$ was used. The experimental range for this is $1330 < Re_b < 12000$. The overall component heat transfer coefficient correlated with the channel Re is

$$\frac{h b}{\kappa} = Nu = \frac{1}{c_1(s)} Re^{0.6} \qquad (26)$$

where $c_1(s)$ was a constant dependent upon the channel spacing and varied from 1.41 for s/b = 0.50 to 4.06 for s/b = 1.5. However, flow visualization studies using smoke wire showed that variations in channel spacing had a relatively small effect on the flow pattern for a fixed flow rate. This indicated that the heat transfer correlation should be independent of channel spacing as well. This was found to be true if the correlation was based on the component Reynolds number Re_b as

$$\frac{h b}{\kappa} = Nu = \frac{1}{c_2(s)} Re_b^{0.6} \qquad (27)$$

For $c_2(s) = 3.64$, this correlation fit the data to 8.6%, and for all practical purposes, from the various studies so far, the heat transfer correlations may be considered to be independent of board spacing.

In any system, components on a board have different heights which affect the flow and heat transfer from them. Effects of component height variations were studied by Sparrow et al [26]. This is an extension of the work reported in [23] where all the components were of uniform height such that t/s = 3/8. In [26], an occasional module at certain chosen positions were missing, or the height changed such that t/s = 0, 1/4, 3/8 or 5/8. It was observed that the heat transfer enhancement at the t/s = 5/8 module was uniformly about 80% over the t/s = 3/8 baseline case for the entire range of Re from 2000 to 7000. However, for the t/s = 1/4 module, the enhancement ranged from 6% at Re = 2000 to 26% for Re = 7000. In general, the results showed that any alteration to the basic array of uniform modules such as a missing module, a module of increased or decreased height enhanced heat transfer. This is advantageous in a practical sense since such geometrical variations are common in circuit boards. These variations cause hydrodynamic disturbances that promote mixing and recirculation, which enhance heat transfer. However, it should always be borne in mind that such enhancements are always at the expense of a larger pressure loss. Thus, for a given fan, the flow rate will drop, which will effectively reduce the heat transfer coefficient. Ultimately therefore, the designer is forced to make a reasonable compromise between heat transfer enhancement and pressure drop.

The correlations given in this section are useful to the designer to perform board level analysis with all the components distributed on it. They enable one to determine the component case temperatures and board temperature distribution. A simple model for one row of devices on a board is shown in Fig. 18.

The model shown here is once again the lumped parameter resistance network thermal model which is chosen here for its flexibility and ease. The solution is obtained by using equation (16), with temperature replacing pressure and defining the appropriate thermal resistances and power dissipations. All the major contributions to thermal dissipation from the device to the ambient are included. Resistances 1 to 5 represent conduction from the DIP case to the board through the pins. Resistances 6 to 9 represent conduction through the circuit board. Resistances 10 to 14 represent convection from the DIP case to the coolant air, and 15 to 19 are convection from the back of the circuit board to the air. 20 to 24 are radiation resistances between the DIP case to the adjacent board. Finally, 25 to 29 represent air temperature rise as it picks up heat from the devices and boards. Also, the air velocities and mass flow rates are obtained from the previous system level analysis. A single node represents a component case temperature and is considered an average value. P in Fig. 18 is the power dissipated by the active device (junction) which is conducted to the case. From the case, part of the heat is dissipated to the ambient air by convection and the remaining is conducted through

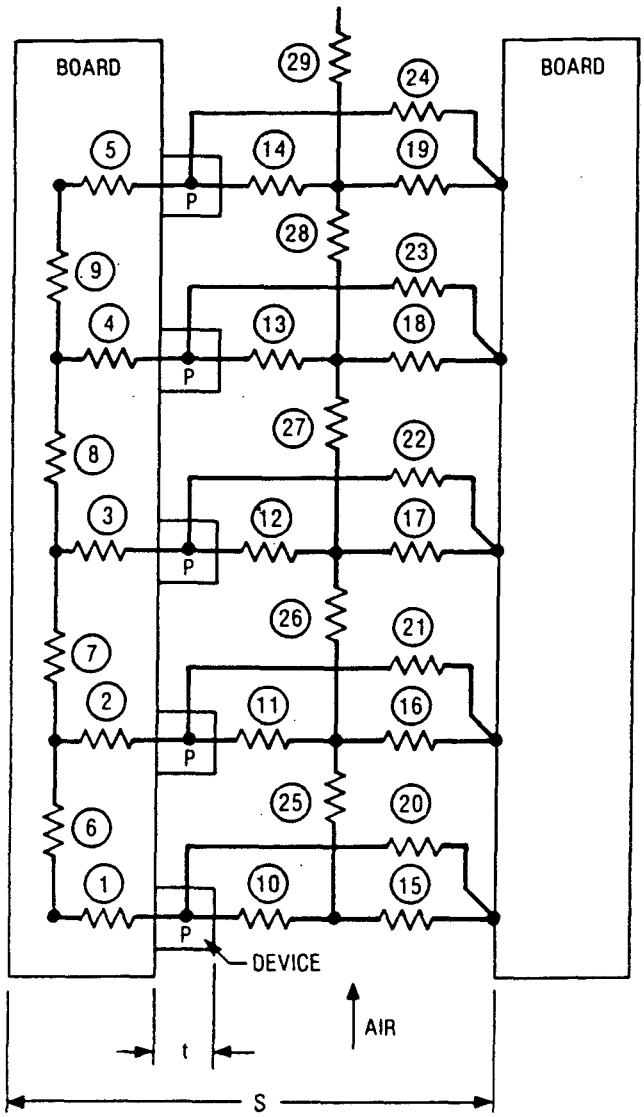


FIGURE 18. SIMPLE THERMAL NETWORK MODEL FOR BOARD ANALYSIS.

the pins to the board. From the boards, heat is dissipated by convection of air flowing over it. In addition, heat is conducted through the board itself. This model may be extended to an entire board without difficulty. Only the number of nodes and resistances will increase. For simplicity, the board temperature below the component is also represented by a single node. This is usually sufficient, since the temperature gradients through the board thickness are small. Radiation from components to boards is also included. The complete board analysis may also be performed using finite differences or finite element modelling. Such a model enables the thermal designer to quickly determine hot spots on boards and identify critical components. It is possible to quickly perform a parametric analysis to determine effects of such variables as flow velocities, board spacings, component heights and power dissipations. In addition, component layout variations may also be studied. Component layout is important because we also want to minimize board temperature gradients as well. Large temperature gradients in boards can cause severe thermal stresses which lead to board warpage. This may cause components to be pulled off of boards.

This completes the system and board level thermal analysis procedures. Although modelling is an invaluable tool to quickly evaluate thermal performance of an equipment design, direct experimental measurements and/or simulation is absolutely essential. As stated earlier, thermal design and analysis in electronic equipments is an art as much as it is a science, essentially because of its inherent complexities. Inevitably, assumptions have to be made for any thermal analysis. Experimental verification is necessary to confirm that the assumptions are valid and physically reasonable. Simulation of a board may be accomplished in two ways. The first method is to populate the board with resistor DIPs that may powered to give the required thermal dissipation. These DIPs are available in a wide range of resistor values. This is a good method because it simulates actual devices with pins soldered to the board. However, making the board is complicated because the necessary tracks for power and ground connections have to be made. Thus photolithographic and etching processes are involved. Board design and manufacturing may become expensive and time consuming. A simpler technique is to attach thin patch heaters on to boards. These patch heaters are available in a wide range of sizes and power densities. The components on the boards may be simulated by aluminum blocks which provide the necessary flow obstructions. This arrangement also provides equal heat dissipation through the Al modules and through the back of the boards. Such an experimental simulation is also valuable in ascertaining the accuracy and validity of several heat transfer correlations discussed earlier, as applied to the particular equipment. Favourable comparisons between experimental simulation and analysis provides confidence to extend the analysis to a wide range of operating parameters.

PACKAGE LEVEL THERMAL DESIGN

At this stage of the thermal design process, we have information about the board temperature distribution, component case temperatures, average air velocities and temperatures around components. This sets the stage for the final step in the thermal design procedure which is the package (component, device) level. The objective of this step is to obtain active device junction temperatures. This is the

ultimate goal of any thermal design of electronic equipment as it is required for reliability predictions. The maximum allowable operating junction temperature is normally restricted to be below 85 ° C (or even 125 ° C for some applications). However, it is desirable to keep it much below this value, since device life is approximately reduced in half for every 10 ° C rise in junction temperature. Package level thermal characterization determines whether the operating limits are attainable, and if not, forces the designer to consider alternate cooling techniques and/or enhancements such as attaching heat sinks.

There are many types of Integrated Circuit (IC) packages such as 16, 24 or 40 pin DIPs, Chip Carriers (CC) with 68 or 100 input/output paths (I/O) and Pin Grid Arrays (PGA) with up to 149 I/O. The packages themselves may be made of plastic or ceramic. From a thermal standpoint, ceramic packages are better because they spread the heat more uniformly, due to higher thermal conductivity. However, plastic packages are economical and are still the most widely used for many commercial applications. Plastic packages however pose thermal problems due to the lower thermal conductivity of the molding compounds. Careful thermal design is necessary for proper heat dissipation.

A 16 pin DIP is shown schematically in Fig. 19 [29].

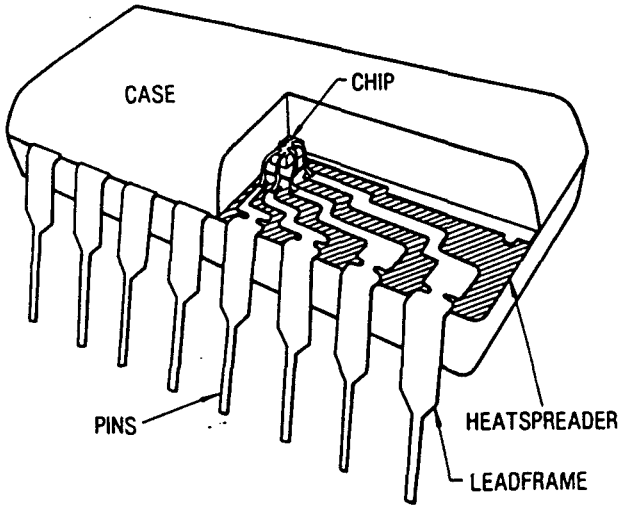


FIGURE 19. SCHEMATIC OF 16 PIN DIP.

It consists of a chip, chip attach medium, metal lead frame, wire bond from the chip to the metal lead frame and the overmolded plastic. Also shown is the possibility of including a heatspreader internal to the molded package. Heat that is generated by the chip is carried away by the plastic, the tie bar and the wire bonds to the lead frame. Part of the heat is carried away by convection and radiation from the DIP surface and the remaining is conducted into the boards through the pins. Heat dissipation from the chip to the outside may be divided into four primary heat paths in plastic DIP which are discussed in [28]. The thermal conductivities of the components in a plastic DIP are given in Table 5 [32].

Table 5. Thermal Conductivities Of Dip Materials [32]. Courtesy of ISHM.

Material	Thermal Conductivity (W/m °C)
Molding Compounds	
A	0.63
B	0.92
C	1.47
D	2.09
LeadFrame	
Alloy42	14.67
Cu, Olin 194	259.0
Heatspreader	
Al	238.4
Cu, Olin 194	259.0
Al_2O_3 (96%)	25.9
Chip	
Silicon	157.0

From this table, we observe that the leadframe has a thermal conductivity that is one to two orders of magnitude higher than that of the molding compound. The silicon chip itself has a high thermal conductivity. It is easy to see that thermal characterization of IC packages are very dependent on the geometry and material properties of its various components. From a practical thermal design consideration, we require a simple conceptual model of all IC packages that is useful for quick evaluation of junction temperatures. Based on this consideration, a simple model of heat flow from the junction to the ambient is shown in Fig. 20.

It shows that there are effectively two parallel heat flow paths; (1) from the junction to the case and to the ambient and (2) from the junction to case and then through the pins (leads) to the circuit board and to ambient. This conceptual model also shows that part of the thermal resistance is due to variables such as material properties and geometry, and may be considered as internal resistance. The other part of the resistance is due to system variables such as cooling techniques and ambient conditions, and may be considered an external resistance. Since heat flows from the junction to case to ambient, the heat balance

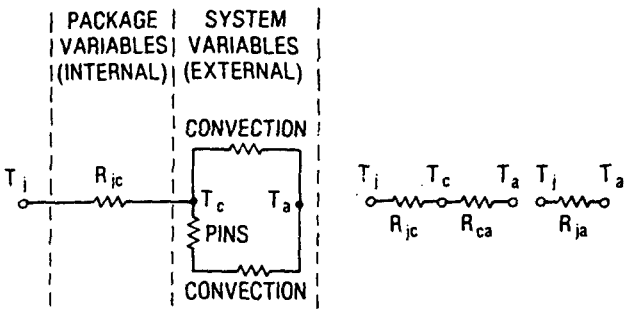


FIGURE 20. OVERALL MODEL FOR HEAT FLOW FROM JUNCTION TO AMBIENT.

equation is

$$P = \frac{(T_j - T_a)}{(R_{jc} + R_{ca})} = \frac{(T_j - T_a)}{R_{ja}} \qquad (28)$$

where R_{jc}, R_{ca} and R_{ja} are thermal resistances from junction to case, case to ambient, and junction to ambient respectively such that $R_{ja} = R_{jc} + R_{ca}$. Therefore, R_{ja} combines the effects of package dependent variables (internal) and the system dependent variables (external) into a single term that is of great practical value.

R_{ja} may be determined by numerical modelling of packages using finite differences or finite element techniques, and also by experimental methods. If the geometry and material properties are known, numerical modelling permits a parametric study to evaluate R_{ja} as a function of heat transfer coefficient, ambient and circuit board temperatures. Numerical modelling of IC packages have to take into consideration the wide range of thermal conductivities as shown in Tables 3,4 and 5, as well as the range of dimensions which varies from a few mils (1 mil = 0.001 inch) for the die thickness to the order of inches for package overall dimensions. Also, since the heat flow is essentially 3 dimensional, the model can become complex, large and time consuming. The more common approach is to determine R_{ja} values experimentally by monitoring the device junction temperature T_j and knowing T_a and P. This method will be discussed in detail later.

IC manufacturers usually provide values of R_{jc} and/or R_{ja} for their packages. These are normally obtained from test data by immersing the package in a constant temperature fluid bath and assuming that the case is at the same bath temperature. If the thermal designer can determine the external thermal resistance R_{ca}, then R_{ja} and T_j may be determined. However, this is seldom accurate, and an in - house experimental test program is necessary. R_{jc} is not a constant and usually depends upon R_{ja}. Also, the temperature of the circuit board into which the package is mounted and the power level also affect R_{ja}. Unless the manufacturer specifies the exact conditions under which measurements were made, R_{jc} or R_{ja} values provided are of little value. They may be used only as a guide. The R_{ja} values supplied by the manufacturer are usually worst case values and the calculated junction temperatures from these are conservative. It is very important for the thermal designer to be aware of the package environment and make appropriate corrections to judiciously avoid under or over design.

Experimental Package Characterization

The objective of experimental characterization is to provide an accurate value of R_{ja}. The experimental procedure consists of bonding a test chip into the package. The package is mounted on the circuit board and cooled with a known air flow velocity over it. A typical test chip may be as shown in Fig. 21.

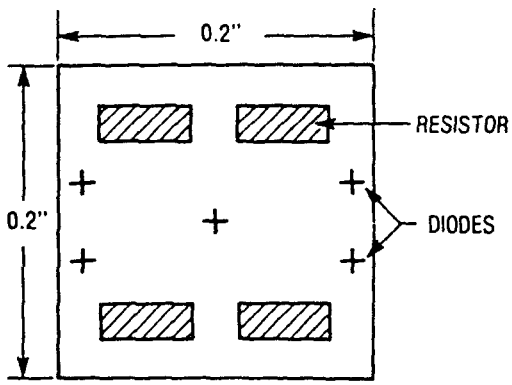


FIGURE 21. SCHEMATIC OF TYPICAL TEST CHIP.

This test chip consists of 4 symmetrically placed resistors and an array of diodes that are used for temperature measurement. The resistors are placed symmetrically so as to obtain a fairly uniform temperature distribution across the chip. Since silicon has a high thermal conductivity (nearly 1/3 that of copper), a symmetric resistor distribution minimizes temperature gradients across the chip. There is no standard test chip and each company has its own version. This makes it difficult at times to make meaningful comparisons between thermal resistance data for an identical package from different sources because of test chip and measurement procedure variations. There is therefore a strong trend in the industry to define a standard test chip and procedure for measuring package thermal characteristics which should eliminate this difficulty. The main purpose for using a test chip is to have exact control over chip power dissipation and temperature monitoring. This

will not be possible for an arbitrary VLSI chip in a package. Monitoring of the diodes provides a convenient and direct method for temperature measurement. The diode is a P-N junction and conducts electricity mainly in one direction. The characteristics of an ideal forward biased diode are given by

$$I = I_i \left[\exp \left(\frac{e V_d}{k T} \right) - 1 \right] \qquad (29)$$

where, I is the forward biased diode current, V_d the voltage, k Boltzmann constant, e the charge of electron, T the absolute temperature and I_i the reverse bias saturation current. Equation (29) may be rearranged as

$$T = \frac{e V_d}{k \ln \left(\frac{I}{I_i} + 1 \right)} \qquad (30)$$

Therefore, for constant current I, equation (30) states that the temperature is linearly proportional to voltage, since all of the other quantities are constant. The diode is calibrated by measuring the forward biased voltage V_d for a small value of constant current I = 0.100 mA at different temperatures. Even a 2 point calibration is usually adequate because of the excellent linearity in accordance to equation (30). The temperatures at which the diodes are calibrated should include the range of temperatures expected. A typical calibration curve is shown in Fig. 22.
Once the diode calibration is available, the packages are powered and cooled with a known air flow velocity over the boards. The voltage across the diode is then measured for a constant I = 0.100 ma. The diode T versus V_d calibration then gives the temperature. The average of temperatures obtained from several diodes is then used as the junction temperature T_j to evaluate R_{ja}. For different packages, R_{ja} values are plotted versus air flow velocities [28] as shown in Figs. 23 and 24 which are invaluable to thermal designers in calculating junction temperatures.

The velocities are always average values through the test section between boards. Observe that R_{ja} is reduced considerably by blowing air at 2.5 and 5.1 (500 and 1000 ft/min) as compared by natural convection cooling. Also, the thermal resistance depends upon the molding compounds A,B,C and D. The user should always remember that such R_{ja} values are only applicable for environments similar to test conditions.

In the experimental evaluation mentioned above, no mention was made of R_{jc}, the junction to case thermal resistance. There is a very fundamental problem with measuring and defining this quantity.

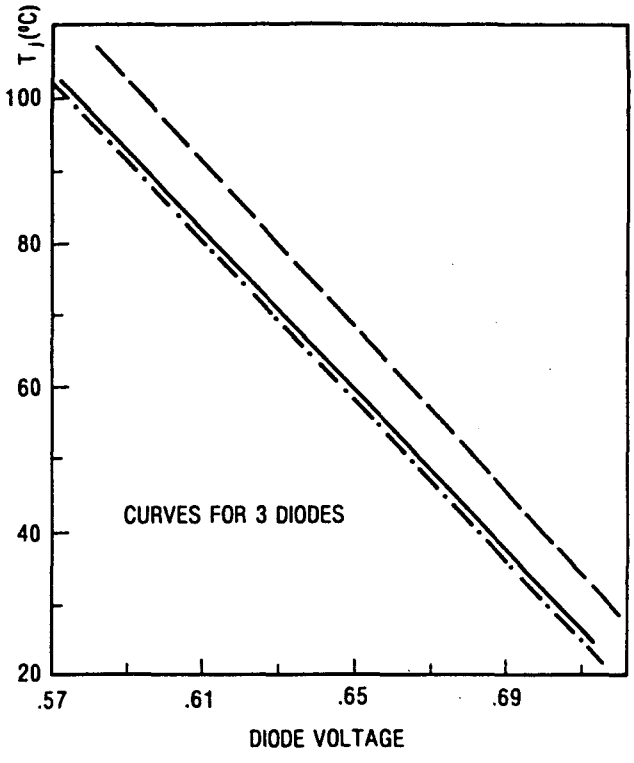


FIGURE 22. TYPICAL DIODE CALIBRATION CURVE.

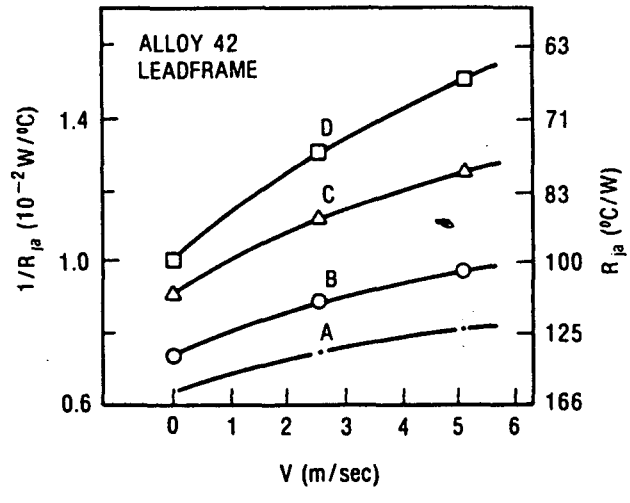


FIGURE 23. THERMAL RESISTANCE OF 16 PIN PLASTIC DIP WITH ALLOY 42 LEADFRAME. [28]

This has to do with the package material itself. For a ceramic package, the thermal conductivity is relatively high, and the temperature gradient on the package case is small. A thermocouple measurement on the center of the case may be realistically considered as the average case temperature. However, if we take a plastic DIP, the low thermal conductivity will cause

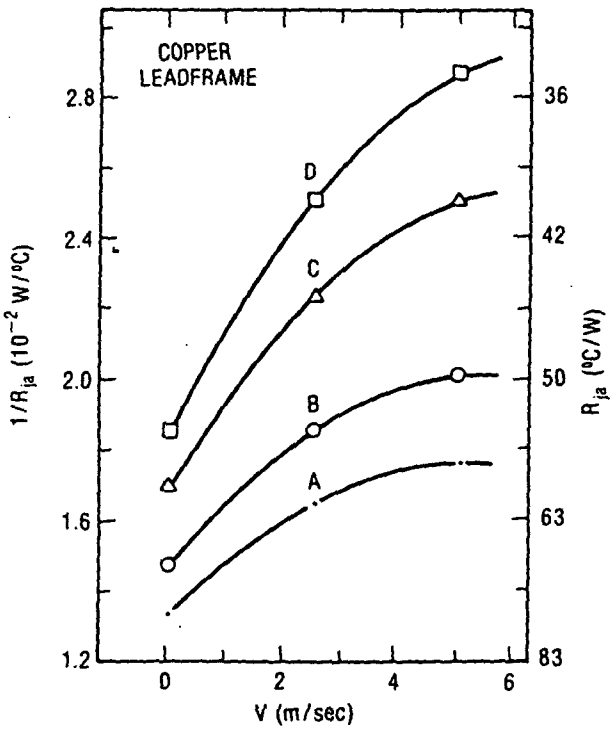


FIGURE 24. THERMAL RESISTANCE OF 16 PIN PLASTIC DIP WITH COPPER LEADFRAME. [28]

large temperature gradients between the center to the edge of the case. This will get worse as the DIP size increases (say, from 16 to 40 pin DIP). In this case an average temperature and an R_{jc} based on this is not meaningful. The value of R_{jc} for a DIP based on a case temperature measured directly behind the chip (which appears to be recommended practice) will be significantly lower than the value based on a case temperature measured on the end of the package. This discrepancy obviously gets worse with larger packages. The answer to this dilemma is to lump the various effects into a single quantity R_{ja} which is of practical value.

Package variables such as size and material properties, as well as environmental variables such as air flow velocity and circuit board temperature affect R_{ja}. These are discussed in detail for 16 and 40 pin plastic DIPs in [28] and [29], and for pin grid arrays in [30]. The effects of different molding compounds and lead frames on R_{ja} for 16 and 40 pin Plastic DIPs is shown in Figs. 25 and 26.

There is a significant reduction in thermal resistance R_{ja} with the copper leadframe over the alloy 42. For the 16 pin DIP in Fig. 25, depending on the molding compound, for natural convection cooling, the copper leadframe can dissipate nearly 2.6 times more heat than the alloy 42 package. Molding compound D allows a 75% increase in heat dissipation over

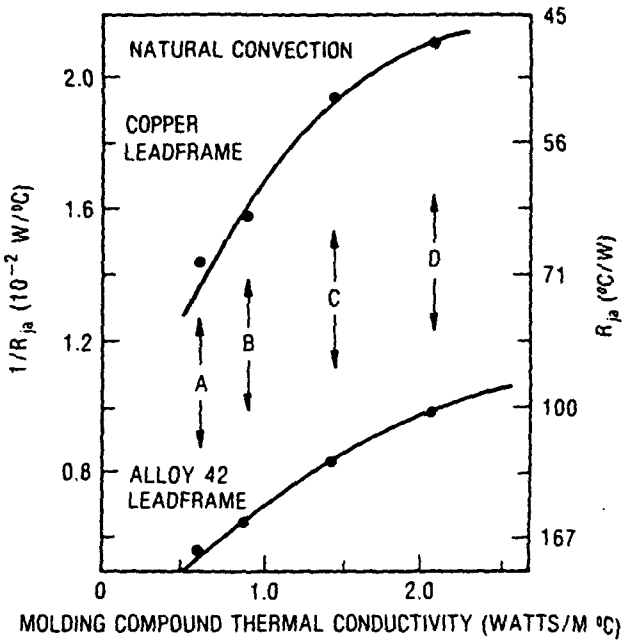


FIGURE 25. THERMAL RESISTANCE OF 16 PIN PLASTIC DIP AS A FUNCTION OF MOLDING COMPOUND. [28,29,32] Courtesy of ISHM.

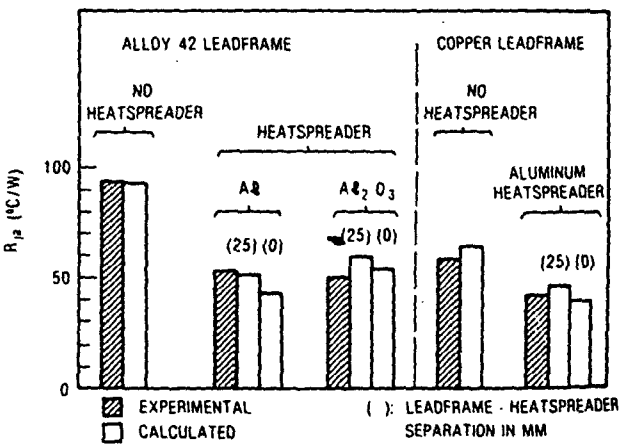


FIGURE 26. THERMAL RESISTANCE OF 40 PIN PLASTIC DIP IN NATURAL CONVECTION. [28,29]

compound A for alloy 42, and a 45% increase for the copper leadframe. Similar results are obtained for 40 pin DIPs as well. The 40 pin plastic DIP has nearly five times more package surface area than the 16 pin DIP and this effectively reduces the R_{ja} value. Referring to Fig. 26, using mold compound A for a 40 pin DIP, the copper leadframe has a R_{ja} value of 58 ° C/W while the alloy 42 has a value of 94 ° C/W.

Also shown are the reduction in R_{ja} for both the Alloy 42 and the copper leadframes, with the addition of an aluminum or alumina heatspreader internal to the mold package. Observe that with an Al heatspreader, the thermal resistance for the alloy 42 leadframe is reduced substantially, while the improvement is only small for the copper leadframe. The alumina heatspreader also does a good job of reducing R_{ja} for the alloy 42, even though its thermal conductivity is one tenth that of aluminum, because its thermal conductivity is still 40 times that of the molding compound. The physical separation between the leadframe and the heatspreader also affects the thermal resistance as seen in Fig. 26. The variation of R_{ja} with number of leads (pins) is shown in Fig. 27 for air velocity of 2.54 m/sec [31].

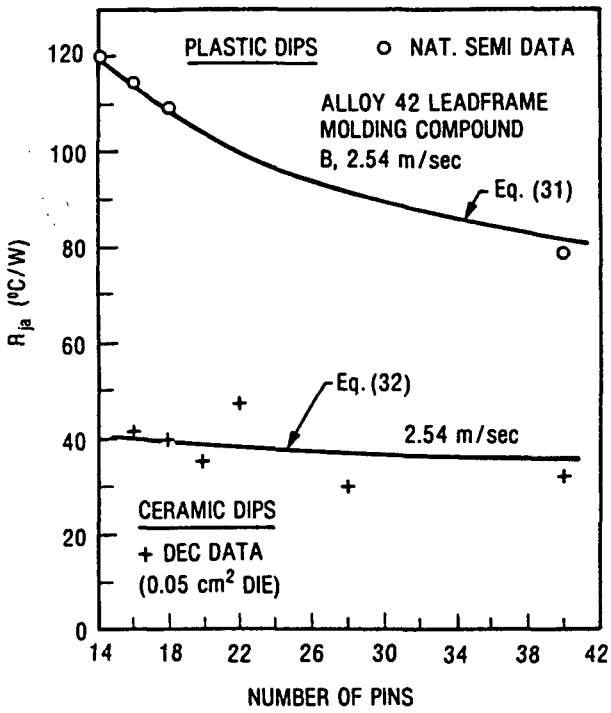


FIGURE 27. THERMAL RESISTANCE OF DIPs AS A FUNCTION OF NUMBER OF PINS.
[31] Courtesy of ISHM

We observe that package size (hence number of pins) is important for plastic DIPs because conduction resistance internal to the package forms a major part of total resistance. For ceramic DIPs which have higher thermal conductivity and hence lower package conduction resistance, R_{ja} only depends weakly on package size (or number of pins). Equations have been developed by Digital Equipment Corporation and are given by Hannemann [31]. For plastic DIPs with molding compound B and alloy 42

$$R_{ja} = 372\, n_l^{-0.37} \exp\left(-0.083\, V^{0.73} \right) \qquad (31)$$

and for ceramic DIPs

$$R_{ja} = 395\, n_l^{-0.57} \exp\left(-4.47\, V^{0.54}\, n^{-0.94} \right) \qquad (32)$$

where n_l is the number of leads, V is the air velocity in m/sec and R_{ja} is in ° C/W.

Equations (31) and (32) for R_{ja} versus air velocity are plotted in Fig. 28 for 16 pin plastic and ceramic DIPs along with the other data [31].

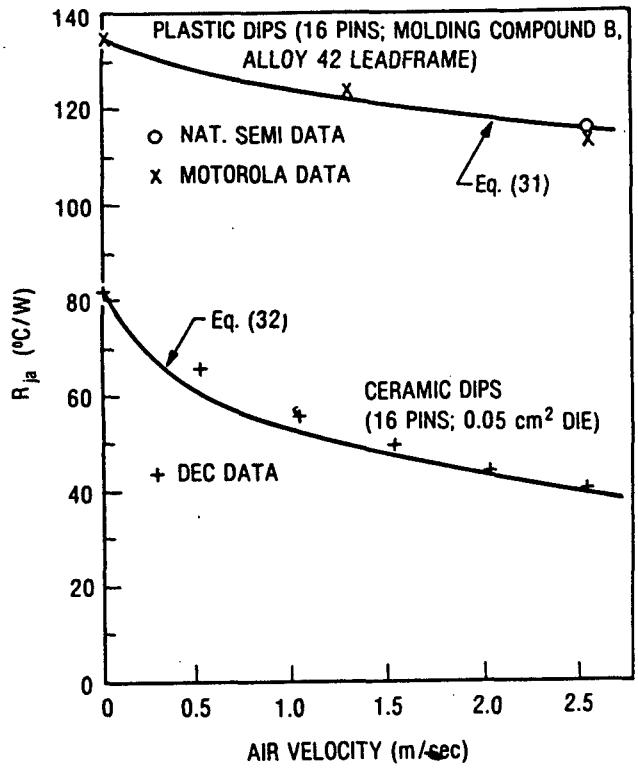


FIGURE 28. THERMAL RESISTANCE OF 16 PIN PLASTIC AND CERAMIC DIPs.
[31] Courtesy of ISHM

As would be expected, the ceramic DIP resistance is more dependent on air velocity since the convective resistance (external) is the dominant factor. The dependence on air velocity is weaker for plastic DIPs since the internal conduction resistance is dominant.

R_{ja} is also a function of chip power level, particularly in natural convection [29]. This dependence for a 40 pin plastic DIP is shown in Fig. 29.

We observe that R_{ja} decreases with chip power level and the trend is more prominent for packages with a high R_{ja}. The decrease in R_{ja} for higher power levels is due to the reduction in the case to ambient thermal resistance. The case temperatures are higher at higher power levels leading to enhanced convection

91 - 6603500

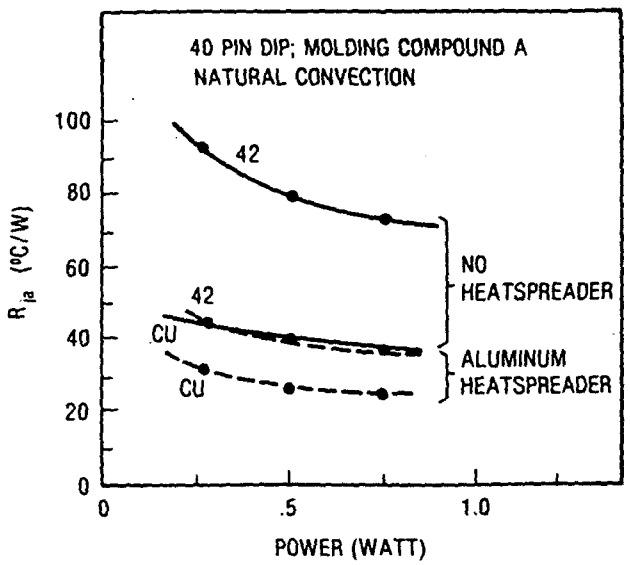


FIGURE 29. THERMAL RESISTANCE OF 40 PIN PLASTIC DIP VERSUS POWER DISSIPATION. [29]

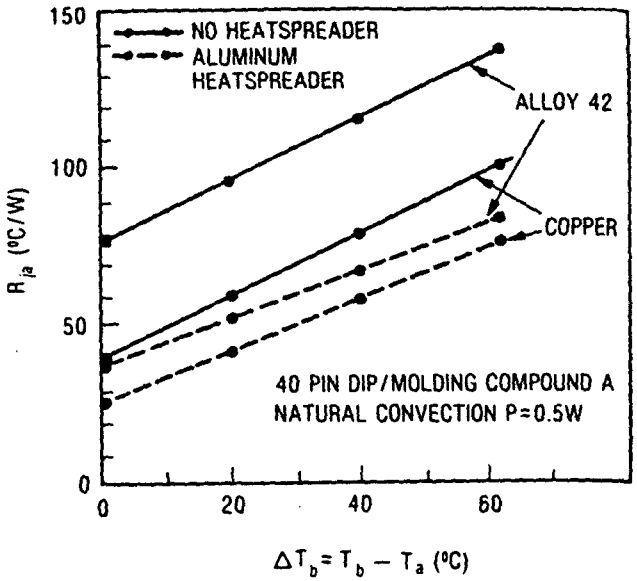


FIGURE 30. THERMAL RESISTANCE OF 40 PIN PLASTIC DIP VERSUS BOARD TEMPERATURE. [29]

and radiation. Thus, we may expect a weak dependence of R_{ja} with chip power for forced convection cooling where heat transfer coefficients are higher.

A very important parameter that affects R_{ja} is the board temperature T_b since heat flows from the case to the board through the pins. Implicitly, this implies the board material as well, since the board thermal conductivity determines the board temperature beneath the device. This may be determined from the board level analysis discussed in previous sections. Unless convection coefficients are extremely high, the board temperature is always above ambient. Usually, there is heat conducted into the board from other devices as well. The change of R_{ja} with board temperature rise above ambient, $\Delta T_b = T_b - T_a$ for 40 and 16 pin plastic DIPs in natural convection were measured experimentally [29] and are shown in Fig. 30 and 31.

For all the cases studied, the variation of R_{ja} (and hence T_j) with ΔT_b is linear. The internal resistance of R_{ja} for any value of ΔT_b may be written as

$$R_{ja} = \frac{R_{ja1}(\Delta T_b - \Delta T_{b2}) + R_{ja2}(\Delta T_{b1} - \Delta T_b)}{(\Delta T_{b1} - \Delta T_{b2})} \quad (33)$$

1 and 2 refer to two board temperature conditions at which R_{ja} values are calculated. The slopes of the curves $\frac{\partial T_j}{\partial(\Delta T_b)}$ are shown in the figures. All of the curves appear to have a constant slope. For the 40

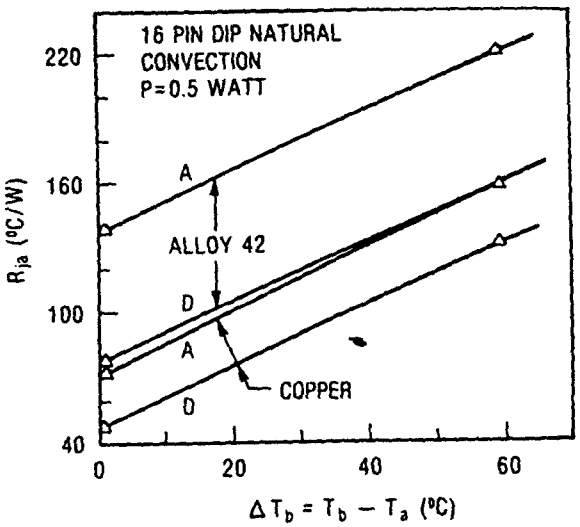


FIGURE 31. THERMAL RESISTANCE OF 16 PIN PLASTIC DIP VERSUS BOARD TEMPERATURE. [29]

pin DIP the increase in R_{ja} is $\approx$ 0.9 ° C/W for each degree increase in ΔT_b and for the 16 pin DIP it is $\approx$ 1.3 ° C/W for every degree increase in ΔT_b. Once again, the board temperature is more important for the 16 pin DIPs than for the 40 pin DIPs. Also, the effect of board temperature on R_{ja} will be less for forced convection than for natural convection. In any case, assuming that the board temperature is equal to the ambient will underestimate the junction temperature T_j and may be in serious error.

Although DIPs have been in use for the majority of applications, their future use may be limited because of the need for high I/O devices. There is a general trend towards increased board density to improve performance and reduce cost. These requirements are being met with the use of PGA and CC, both of which come in the ceramic and plastic packages. Thermal studies on PGA are given in [30] and CC and PGA are given in [32]. Thermal performance is given for a 68 terminal leadless ceramic CC dissipating 4 W and for 149 pin PGA package dissipating 12 W [32], both cooled by forced convection air. Both use bipolar chips which were bonded to the ceramic substrate inside an open cavity.

Chip carriers are usually classified into the cavity up or the cavity down configurations and these are shown in Fig. 32 [32].

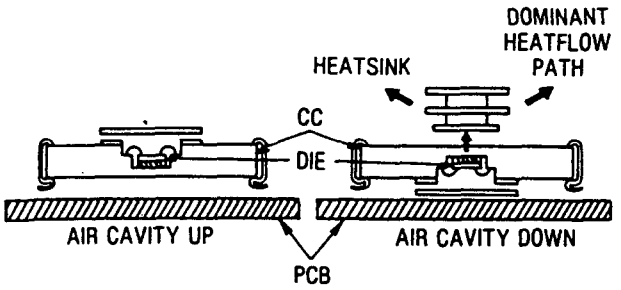


FIGURE 32. SCHEMATIC OF 68 TERMINAL CHIP CARRIER (CC).
[32] Courtesy of ISHM

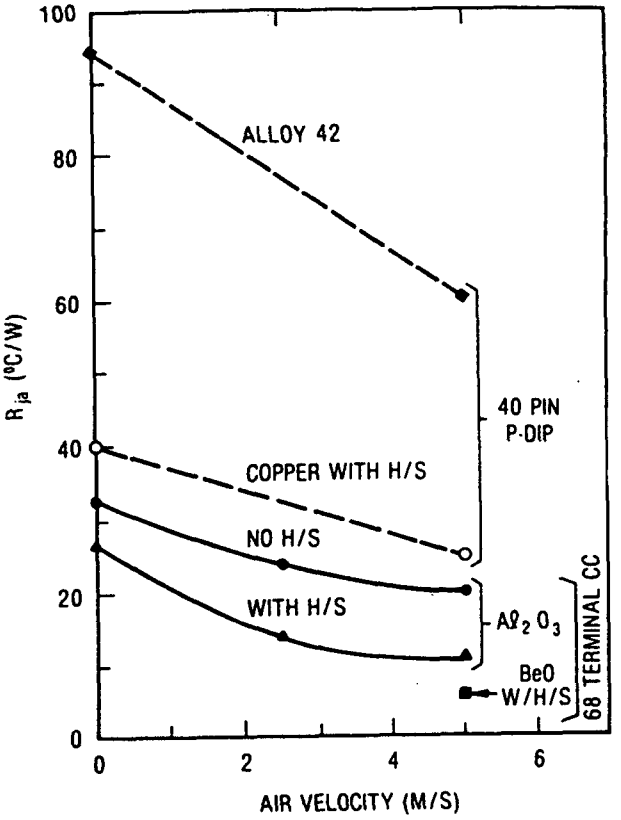


FIGURE 33. THERMAL RESISTANCE OF CHIP CARRIERS VERSUS AIR VELOCITY.
[32] Courtesy of ISHM

In either case, heat flows from the chip to the substrate and then to the package case to be removed. Obviously, from a cooling point of view, the cavity down configuration is better, because the heat has a direct path to the substrate above it where it may be removed by convection with or without a heat sink. In the cavity up configuration, the heat cannot be removed effectively from the substrate near the circuit board. Heat can be removed from the package case at the top, but the heat path is longer. R_{ja} for a 68 terminal CC is shown in Fig. 33 as a function of air flow velocity, with and without heat sinks [32]. In these ceramic packages, the major resistance is external (case to ambient), and good cooling techniques can effectively reduce the R_{ja}, as for example with a heat sink. Also, BeO substrate has a lower thermal resistance than Al_2O_3.

A 149 PGA package is shown in Fig. 34 with a heat sink [32]. The package once again may be a cavity up or cavity down configuration to house the chip. The ceramic substrate could be alumina, beryllia or a combination of both. The primary heat path is from the chip to the substrate and to the case. R_{ja} and R_{jc} for a cavity down PGA are shown in Fig. 35 for a Al_2O_3 substrate and Al_2O_3 - BeO substrate.

In forced air cooling, it is clear that the Al_2O_3-BeO substrate has a lower R_{ja} than the Al_2O_3. Also, the difference between the two becomes greater at higher velocities , emphasizing the importance of external cooling, since this is the more dominant thermal resistance for these packages. Also, observe that R_{jc} for both the substrates are nearly constant over the entire air velocity range, with the Al_2O_3 package having a thermal resistance that is nearly 3 times as large as the Al_2O_3 - BeO package.

A variable that is also of potential importance in the evaluation of R_{ja} is the size of the chip itself. This will be important for cases where the die size is very small, the power level is very high or the package thermal resistance very low. These conditions do not exist for DIPs. The variation of R_{ja} with chip size is shown in Fig. 36 for 16 pin plastic and ceramic DIPs [31].

The effect of chip size on R_{ja} is small for chip size >

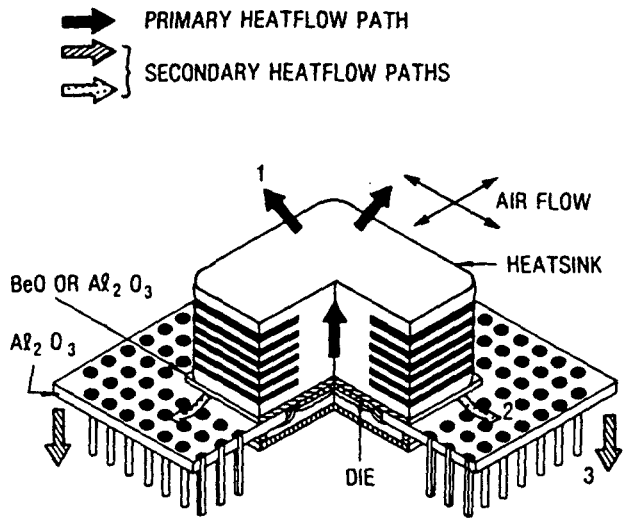


FIGURE 34. SCHEMATIC OF 149 PIN PIN GRID ARRAY (PGA).
[32] Courtesy of ISHM

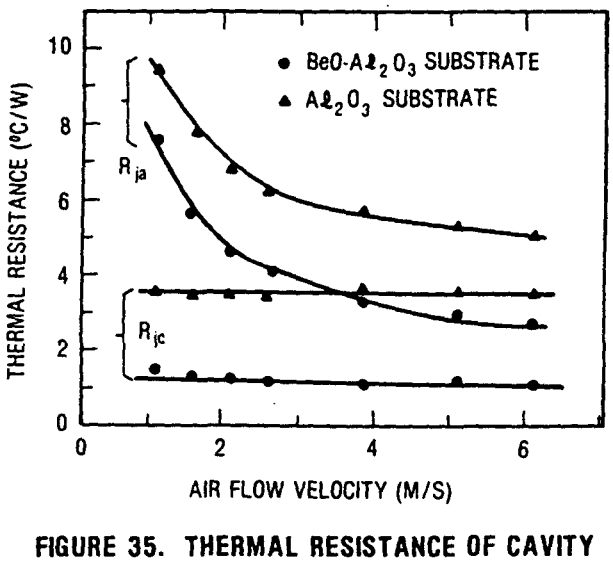


FIGURE 35. THERMAL RESISTANCE OF CAVITY DOWN PGA VERSUS AIR VELOCITY.
[32] Courtesy of ISHM

0.025 cm^2. Since most devices of practical interest are larger than this, chip size is not of serious concern for DIPs. Also, note that a change of 2 to 3 ° C/w for a DIP whose R_{ja} is about 110 ° C/w is only an error of 2 to 3%.

A knowledge of R_{ja} for any of these packages such as DIPs, PGA, of CC enables the designer to determine the chip junction temperature. From the earlier system and board level analysis, we have determined the average air velocities and the board temperature profiles. Knowing the chip power dissipation, the junction temperature may be determined from

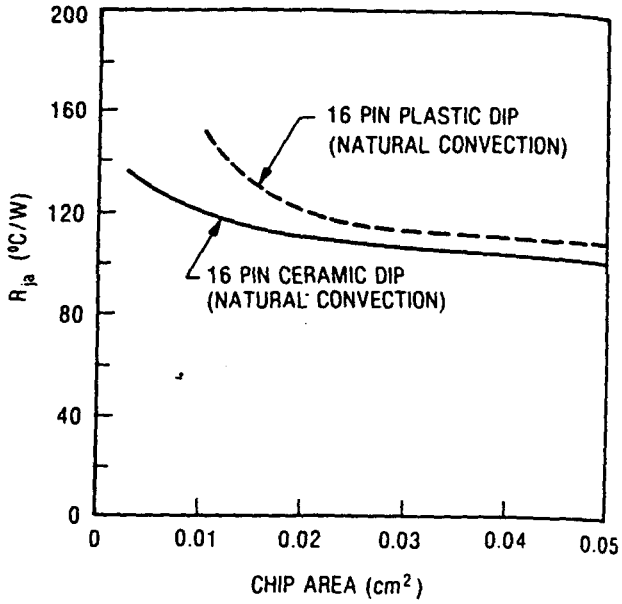


FIGURE 36. THERMAL RESISTANCE OF 16 PIN DIPs VERSUS CHIP SIZE.
[31] Courtesy of ISHM

equation (28). Determination of the chip temperature completes the first cycle of the thermal design process. If the chip temperature exceeds the maximum allowable value, the designer has to evaluate alternatives such as different cooling techniques, bigger fans to provide larger mass flow rate and higher velocities and/or heat sinks on critical components in order to reduce junction temperature. The three step simplified thermal design procedure must be followed until all the thermal constraints at the system, circuit board and device (chip) level are satisfied.

OTHER COOLING TECHNIQUES

The thermal design procedure outlined in the earlier sections dealt with forced air convection cooling, with air flowing directly over the components and boards. This direct air flow over the components is sometimes called impingement cooling. A potential problem with this direct air flow is contamination. Air drawn from the external source (ambient) may contain water vapor and/or other contaminants which are detrimental to electronic equipment. Therefore, military specifications prohibit the use of impingement cooling with air drawn from the atmosphere unless the water vapor and contaminants are removed. Impingement cooling is allowed for closed loop systems. Also, the cooling capacity of direct air cooling is limited to about 0.5 w/ in^2. This low heat removal capacity is critical in designing equipment for aerospace applications where weight and volume have to be minimized. For these

applications, the higher heat removal capacity is achieved with the use of compact heat exchangers.

The first of these is called the cold plate or cold wall cooling. In this method, the walls of the equipment serve as heat exchangers. Heat from the components mounted on the card is conducted to the edge of the card. The PCBs are laminated with aluminum or copper to increase the overall thermal conductivity, since the heat dissipation depends entirely on conduction through the board. From the edge of the board, heat is conducted across the contact interface to the heat exchanger. The cold plate heat exchanger is shown schematically in Fig. 37 [3].

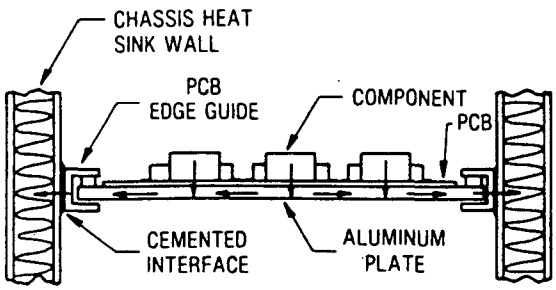


FIGURE 37. COLD PLATE PCB COOLING WITH HEAT EXCHANGER.

Obviously, the effectiveness of this cooling method depends critically upon the board conduction resistance and the interface contact resistance between the board and the heat exchanger. The boards are plugged into guides that are fastened to the heat exchanger. In order to minimize the interface thermal resistance, high pressures on smooth and dry surfaces with large contact area are required. Different types of board edge guides are shown in Fig. 38 [3].

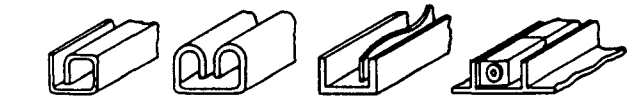


FIGURE 38. CIRCUIT BOARD EDGE GUIDES.

The thermal resistances of these edge guides are expressed in °C inch/W. These guides are satisfactory for applications at sea level and medium altitudes up to 50000 feet. At altitudes of 100000 feet, test data show that the resistance increases by about 30% for guides (a) to (c), and about 5% for guide (d) [3]. Wedge guides (d) are the best for outer space applications since they provide high interface pressures. Bolted interfaces are also used extensively for these cases, since they provide high interface pressures. However, bolted interfaces sacrifice the board plug - in capability which is very desirable for repair. Cold plate cooling with air heat exchangers have a cooling capacity of 1 w/ in^2.

In cold plate cooling, the conduction heat path through the board is long. This may cause large temperature gradients in the board. An improvement to the cold plate cooling is the flow through module or the hollow core PCB. In this method, the heat exchanger is made an integral part of two circuit boards in the hollow core between them, or the heat exchanger could be made as an integral part of a single circuit board as shown in Fig. 39 [5]. This cooling technique is suitable for cooling Very High Speed Integrated Circuits (VHSIC).

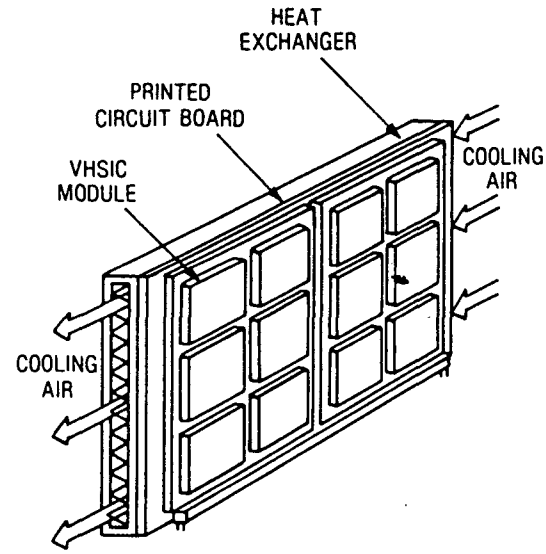


FIGURE 39. HOLLOW CORE PCB COOLING WITH HEAT EXCHANGER.

The heat path from the components to the heat exchanger is directly through the thickness of the board. Therefore, this is the most effective cooling technique and the cooling capacity with air is about 2 w/in^2. The most critical problem with the hollow core

PCB cooling, is providing a proper air seal that must be capable of being engaged and disengaged many times without air leakage. Vibration may also be a problem with hollow core heat exchangers, since the circuit boards have a low resonant frequency. This may lead to rapid fatigue failures. Stiffening ribs may have to be provided in the hollow core between boards, or even external to the boards if more stiffening is required.

Both the cold plate and flow through methods depend upon compact heat exchangers for heat removal. The design of these compact heat exchangers are done using the techniques outlined by Kays and London [33]. These require the calculation of the Colburn factor, and the friction factor f, from which we can calculate the heat transfer coefficient h and the pressure drop Δp for the heat exchanger. Results for many compact heat exchanger surfaces are provided in [33]. These require the knowledge of air flow rates which may be evaluated by the techniques outlined in earlier sections. The system pressure losses including plenums, orifices, bends and heat exchanger must be matched to the fan performance curve. From these, the variation of board temperature for a given heat dissipation may be calculated. Obviously, the heat carrying capacity can be increased by nearly an order of magnitude by using liquid cooling with water, at a higher cost and greater system complexity. Liquid cooling of microwave equipment and travelling wave tubes, thermoelectric cooling for infrared detectors, and the theory of immersion cooling and heat pipes are discussed in [5]. Thermal management of the recent high performance multi - chip modules such as the IBM Thermal Conduction Module (TCM) using water cooled cold plate, and the Mitsubishi High Thermal Conduction Module (HTCM) using air cooled heat sinks are discussed in detail in [1].

CONCLUSIONS

This paper is an overview of the thermal design process applied to electronic equipment to ensure that the devices operate at temperatures below the allowable limits for reliability purposes. The overall thermal design process is broken down into three levels and these are called the system level, board level and the package (device, component) level. The principle of a such a systematic procedure is to keep the problem in a simple and tractable form. The results obtained from one level are used as input to the next level. This step wise procedure is followed until we arrive at the chip junction temperature. This procedure is a convenient tool for equipment designers at an early stage to ascertain cooling techniques. The hierarchy of thermal design is explained by using a telecommunication equipment as an example. This equipment consists of several shelves of circuit cards that are stacked on top of each other. The devices mounted on the cards are cooled by air using natural or forced convection. Although the example pertains to an air cooled system, the basic philosophy of thermal design outlined is the same for other cooling techniques as well. Air cooling is still the most popular method for its low cost and ease of implementation.

The first step called the system level is an analysis at the global level which enables the designer to obtain a global estimate of cooling requirements. The experimental and modelling methods for obtaining system pressure drop and flow rates are outlined. The air velocities are then used for the next step, which is the board level analysis. From this level, the designer can obtain the board temperature profile and identify hot spots on the board, and also obtain component case temperatures. This level may be done for an individual board or a number of boards. The final level is the package (device) level from which the junction temperature is obtained. The method of determining the junction to ambient thermal resistance experimentally using a test chip is outlined. The thermal resistance depends upon the package materials and the external conditions such as board temperatures and cooling air velocity. Results for DIPs, CC and PGA are given. Determination of the device junction is the ultimate goal of the thermal design process. If the junction temperatures are above the allowable limits, then changes such as adding heat sinks and/or larger fans must be considered, and the thermal design cycle must be repeated. Simple models based on resistor network flow and thermal models are an extremely practical and useful tool for thermal analysis. Experimental measurements are an absolute necessity in the thermal design procedure to confirm the validity of several assumptions made for the analysis. Due to the increasing component and power densities of equipment today, thermal design and control has become an important and necessary step for reliable operation.

ACKNOWLEDGEMENTS

The author would like to acknowledge the assistance of Andrew C. Petropoulos of AT&T Bell Laboratories, Whippany in preparing this paper.

NOMENCLATURE

A	area, m^2
b	component depth, m
c	constant
d	component width, m
D_h	hydraulic diameter, m
e	electron charge, C
E	activation energy, J
f	friction factor, dimensionless
F	probability of failure

g	acceleration due to gravity, m/s^2
h	heat transfer coefficient, W/m^2K
I	current, A
k	Boltzmann constant
K	loss coefficient
L	flow length, m
$\dot{m}$	flow rate, kg/s
n	constant
n_l	number of leads
N	component number
Nu	Nusselt number, dimensionless
p	pressure, Pa
$Pr = \nu/\alpha$	Prandtl number, dimensionless
q''	heat flux,W/ m^2
Q	volume flow rate, m^3/s
$Ra'' = q'' \beta g s_e^5 / L\nu\alpha\kappa$	Rayleigh number,dimensionless
$Re = VD_h/\nu$	Reynolds number,dimensionless
R_r	reaction parameter/s
R_f	reliability function
R_{ij}	resistance, degree K/W, Pa.s/Kg
s	circuit board pitch, m
s_e	effective flow width, m
S	source term, W, Kg/s
t	component height, m
T	temperature, K
V	velocity, m/s
V_d	diode voltage, V
α	thermal diffusivity, m^2/s
β	volume expansion coefficient,1/K
κ	thermal conductivity, W/mK
λ	failure rate, 1/s
ν	kinematic viscosity, m^2/s
ρ	density, Kg/m^3
$\sigma = \rho/\rho_0$	density ratio
τ	time, s
Subscripts	
a	ambient
b	board
ca	case to ambient
i,j	nodes
j	junction
ja	junction to ambient
jc	junction to case
0	reference value

REFERENCES

1. Bar-Cohen, A., Thermal Management Of Air And Liquid-Cooled Multi-Chip Modules, 23rd Natl. Heat Transfer Conf., Denver, 1985.
2. Oktay, S., Hannemann, R., and Bar-Cohen, A., High Heat from A Small Package, Mechanical Engineering, pp. 36-42, March 1986.
3. Steinberg, D.S., Cooling Techniques For Electronic Equipment, John Wiley & Sons, 1980.
4. Ellison, G.N., Thermal Computations For Electronic Equipment, Van Nostrand Reinhold Company, 1984.
5. Kraus, A.D., and Bar-Cohen, A., Thermal Analysis and Control of Electronic Equipment, Hemisphere Publishing Corporation, 1983.
6. Morrison, G. N., Kallis, J.M., Strattan, L.A., Jones, I.R., and Lena, A.L., RADC Thermal Guide For Reliability Engineers, RADC-TR-82-172, June 1972.
7. Reliability Information Notebook, AT&T Bell Laboratories, Fifth Edition, 1985.
8. Bertram, W. J., "Yield And Reliability", in VLSI Technology, ed. S. M. Sze, McGraw-Hill, 1983.
9. Aung, W., Kessler, T.J., and Beitin, K.I., Free Convection Cooling Of Electronic Systems, IEEE Trans., Parts, Hybrids and Packaging, vol. PHP-9, no. 2, pp. 75-86, 1973.
10. Aung, W., Fletcher, L.S., and Sernas V., Developing Laminar Free Convection Between Vertical Plates with Asymmetric Heating, Int. J. Heat Mass Transfer, vol. 15, pp. 2293-2308, 1972.
11. Aung, W., Fully Developed Laminar Free Convection Between Vertical Plates Heated Asymmetrically, Int. J. Heat Mass Transfer, vol. 15, pp. 1577-1580, 1972.
12. Wirtz, R.A., and Stutzman, R.J., Experiments on Free Convection Between Vertical Plates With Symmetric Heating, J. Heat Transfer, vol. 104, pp. 501-507, 1982.
13. Churchill, S.W., and Usagi, R., A General Expression for the Correlation of Rates of Heat Transfer and Other Phenomena, AICHE Journal, vol. 18, no. 6, pp. 1121-1128.
14. Bar-Cohen, A., and Rohsenow, W.M., Thermally Optimum Spacing of Vertical Natural Convection Cooled Parallel Plates, J. Heat Transfer, vol. 106, pp. 116-123, 1984.
15. Birnbrier, H., Experimental Investigations on the Temperature Rise of Printed Circuit Boards in Open Cabinets with Natural Ventilation, in Heat Transfer In Electronic Equipment, HTD vol. 20, ASME, 1980.
16. Coyne, J.C., An Analysis of Circuit Board Temperatures in Electronic Equipment Frames Cooled by Natural Convection, in Fundamentals of Natural Convection/Electronic Equipment Cooling, HTD vol. 32, ASME, 1984
17. Johnson, C.E., Evaluation Of Correlations for Natural Convection Cooling of Electronic Equipment, Heat Transfer Engineering, vol. 7, no. 1, pp. 19-28, 1986.
18. Osborne, W.C., Fans, Pergamon Press, 1982.

INTERACTIVE THERMAL MODELING OF ELECTRONIC CIRCUIT BOARDS

William M. Godfrey, Kaveh Taghavi, Clifford J. Cremers, and M. Pinar Mengüc
Department of Mechanical Engineering
University of Kentucky
Lexington, Kentucky

Abstract— The numerical modeling of Printed Circuit Board (PCB) configurations is becoming more prevelant as denser electronic packages are manufactured. Determining an optimal PCB design typically requires modeling and testing several different chip configurations. At present, much of the time and cost spent modeling these various configurations is repetitive in nature, with the same chip or board being regenerated several times before obtaining the most efficient design. By using a decoupled, iterative, numerical technique, a more systematic and efficient method for determining PCB temperature distributions has been developed and incorporated into a controlling algorithm which utilizes any standard finite element code. Steady state temperature distributions are obtained for each decoupled chip/board component by iterating upon decoupled interface boundary conditions until convergence is obtained. The main advantages of this decoupling technique are the elimination of repetitive numerical-model generation and the high level of interactive design provided.

NOMENCLATURE

c_p	specific heat, $kJ/(kgK)$
h	heat-transfer coefficient, $kW/(m^2K)$
k	thermal conductivity, $kW/(mK)$
q	heat flux, kW/m^2
$\dot{q}$	heat generation, kW/m^3
r	position vector, $r(x,y,z)$
t	time, s
T	temperature, K
x,y,z	Cartesian coordinates

Greek Symbols

λ	relaxation coefficient
ρ	density, kg/m^3

Subscripts and Superscripts

b	board
c	chip
i	interface
j	current iteration number
ni	non-interface
s	surface
ss	steady state
∞	surrounding

1 INTRODUCTION

The thermal design of printed circuit boards (PCBs) has become a critical process in the early development stages of electronic packages. This increasingly important design consideration has been brought about by rapid advancements in chip technology and packaging techniques. Chip packages are expected to provide the computer industry with power densities in excess of 100 W/cm^2 by the mid-1990's (Bar-Cohen et al., 1986). These developments have prompted thermal engineers to put a strong emphasis on optimal chip location early in the design process. In determining the optimal chip configuration, it is usually necessary to consider several chip/board models. The techniques used to generate and solve these models range from analytical and computer aided numerical models to experimental procedures.

Experimental testing is the most accurate method of obtaining correlations and temperature distributions for any particular PCB. For this reason, experiments are usually performed on fabricated models of proposed PCBs before mass producing them. However, the advantage of obtaining data from experimentation must be weighed against the large amount of time and expense required to operate such experiments. If the initial experimental results suggest a relocation of chips, the PCB model must be refabricated and the experimental procedure begun anew. As chip packages are made denser, the number of such design alterations will continue to increase as thermal engineers strive to produce optimal chip/board configurations. This reality limits the cost effectiveness of experimentation as a practical means of determining initial temperature distributions across PCB surfaces. Hence, if a numerical or analytical solution exists for a given PCB model, experimental testing is usually postponed until the final design stage, leaving the initial design stages to more cost effective techniques.

Analytical models like those developed by Pinto et al. (1986), Haji-Sheikh et al. (1988), and Funk et al. (1990) are frequently used to provide PCB temperature distributions. The accuracy of these temperature distributions is dependent upon the accuracy of the parameters used in the model. Furthermore, the applicability of these models is often limited to simple geometries and boundary conditions. Even with these limitations, analytical modeling offers many advantages over experimental modeling. For instance, relocat-

ing chips in order to obtain the temperature distribution in different designs is more efficiently done using analytical rather than experimental techniques. With the help of computers, analytical solutions can be obtained much quicker than experimental results. While these features make analytical models more cost effective than experimentation, it must be noted that many real-life problems involving complex geometries do not have readily available analytical solutions.

Fueled by the accessibility of computers, numerical solution techniques have become the most frequently used tool for determining temperature distributions in both individual components (Rodamaker et al., 1987) and complex PCB configurations. Of particular interest in this study are the powerful finite element computer codes that permit even nontechnical users to generate and solve complex PCB models. Several interactive finite element modeling procedures exist that produce thermal solutions for PCBs, including those developed by Nalbandian (1987) and Shiang et al. (1987).

As with analytical solutions, the accuracy of numerical solutions depends upon the accuracy of the heat transfer coefficients, thermophysical properties, and other parameters input by the user. Given this accuracy, the benefits of fast computer solution times and extensive modeling capabilities make the use of finite element codes more cost effective than analytical or experimental techniques. The advantages associated with numerical finite element design procedures are weakened somewhat if changes in chip locations are desired after an initial PCB configuration has been modeled. When chips are relocated, a new model must be constructed using the same chips and board, resulting in repetitive model generation.

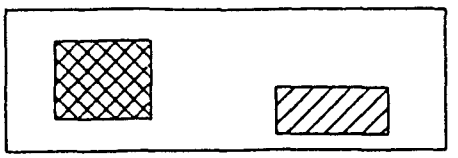
Because most finite element programs offer diverse applications, their specific use for PCB design often lacks the systematic approach sought by thermal design groups. This paper describes the development of an efficient procedure for determination of PCB temperature distributions using any readily available finite element program. The incorporation of this procedure into an algorithm requires finite element solutions to be utilized within an iterative procedure. This algorithm refines the conventional finite element design process by eliminating the repetitious generation of PCB models typically required to test different chip configurations.

To illustrate the conventional design procedure, consider the chip/board model shown in Figure 1a. If this configuration's temperature distribution did not meet design requirements, the configuration shown in Figure 1b might be generated and tested. If this second configuration also failed design requirements, it might be necessary to test the chips on a larger board, shown in Figure 1c, before obtaining the desired temperature distribution. In this example, three different models were generated. If, however, a more complex PCB configuration was being designed, a larger number of models would probably need to be generated before determining the optimal configuration. In the current algorithm, a decoupling procedure is utilized to eliminate this repetitive numerical-model generation.

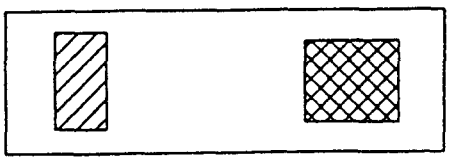
The objectives of this paper are to describe the methodology behind this newly developed procedure and to present how PCB temperature distributions are obtained by iterating upon finite element solutions within a controlling algorithm.

2 GOVERNING EQUATIONS

The configuration shown in Figure 2 consists of a single chip attached to a board using direct contact surface mount technology (SMT). The primary emphasis of this model will be placed upon the chip/board *interface boundary conditions*. Hence, the surface boundary conditions, the chip heat generation, and all component properties will be assumed known and treated as input parameters. Heat conduction within this configuration is governed by the partial differential equation



(a) Initial configuration tested



(b) Chip locations changed from (a)

(c) Same chips tested on larger board

Figure 1: Three models tested for optimal configuration.

$$\nabla \cdot [k(r,T)\nabla T] + \dot{q}(r,T) = \rho(r,T)c_p(r,T)\left(\frac{\partial T}{\partial t}\right) \qquad (1)$$

where r is any point on the chip/board configuration. Further references to k, $\dot{q}$, ρ, and c_p should be implicitly taken as functions of both r and T.

Considering only the chip and its local coordinate system, Eq. (1) can be written as

$$\frac{\partial}{\partial x_c}\left(k_c\frac{\partial T_c}{\partial x_c}\right) + \frac{\partial}{\partial y_c}\left(k_c\frac{\partial T_c}{\partial y_c}\right) + \frac{\partial}{\partial z_c}\left(k_c\frac{\partial T_c}{\partial z_c}\right) + \dot{q}_c = \rho_c c_{p,c}\left(\frac{\partial T_c}{\partial t}\right) \qquad (2)$$

Possible boundary conditions on the exposed chip surfaces, where $r_c = r_{c,s}$, are prescribed heat flux, prescribed temperature, and radiation or convection heat transfer. Applying the most practical and common boundary condition of convection heat transfer on all five exposed surfaces simplifies the boundary conditions to the single equation:

$$h(r_{c,s}, T_{c,s}, T_\infty)(T_{c,s} - T_\infty) = -k_{c,s}\left(\frac{\partial T_{c,s}}{\partial r_{c,s}}\right) \qquad (3)$$

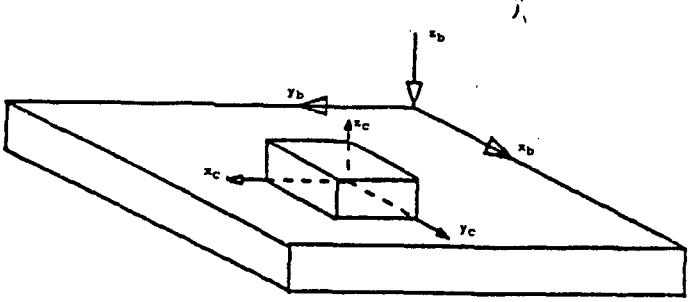


Figure 2: Single chip on board configuration (SMT).

In similar fashion, the formulation of Eq. (1) for the board and its local coordinate system can be expressed as

$$\frac{\partial}{\partial x_b}(k_b\frac{\partial T_b}{\partial x_b}) + \frac{\partial}{\partial y_b}(k_b\frac{\partial T_b}{\partial y_b}) + \frac{\partial}{\partial z_b}(k_b\frac{\partial T_b}{\partial z_b}) = \rho_b c_{p,b}(\frac{\partial T_b}{\partial t}) \quad (4)$$

The imposition of convection boundary conditions on the exposed surfaces of the board, where $r_b = r_{b,s}$, is described by

$$h(r_{b,s}, T_{b,s}, T_\infty)(T_{b,s} - T_\infty) = -k_{b,s}(\frac{\partial T_{b,s}}{\partial r_{b,s}}) \quad (5)$$

The conditions existing at the chip/board interface are equal heat flux and temperature, shown respectively as

$$-k_{c,i}(\frac{\partial T_{c,i}}{\partial r_{c,i}}) = -k_{b,i}(\frac{\partial T_{b,i}}{\partial r_{b,i}}) \quad (6)$$

$$T_{c,i} = T_{b,i} \quad (7)$$

These governing equations have been used as a basis in formulating the methodology of the newly developed PCB design procedure.

3 METHODOLOGY

Each of the following subsections pertain to specific processes that have been incorporated into the current design procedure.

3.1 DECOUPLING

At present, chip/board finite element models are generated using coupled chip and board components as shown in Figures 1a–c. While this approach seems logical, PCB designers will attest that this generation method becomes very repetitive when several different configurations of the same components are to be modeled. If a decoupled approach is used instead, this type of repetitive work can be eliminated.

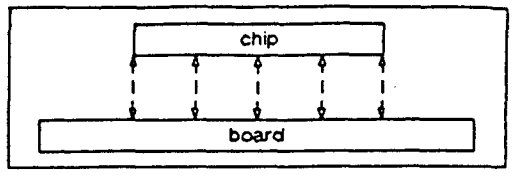
As illustrated in Figure 3a, decoupling a chip/board at its interface involves physically detaching the chip from the board. It must be emphasized that while decoupling gives each component its own identity in appearance, a mathematical dependency still remains between each separated chip and board interface. This dependency will be explained by decoupling the previously defined governing equations into two separate formulations, one for the chip and another for the board.

Equation (2) governs the heat conduction in the decoupled chip shown in Figure 3b. Convective boundary conditions placed on the top and four side surfaces of the chip can be described using Eq.(3). The bottom interface condition is what differentiates the decoupled analytical model from the coupled analytical model. The coupled model requires both conditions in Eq.(6) and Eq.(7) to exist at the interface boundary, whereas only one of these conditions can be prescribed at the the decoupled chip interface. Hence, the chip interface boundary condition could be either

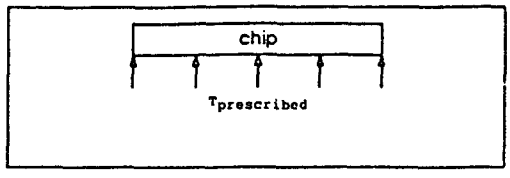
$$T_{c,i} = T_{input} \quad or \quad q_{c,i} = q_{input}. \quad (8)$$

The prescribed temperature boundary condition, as indicated in Figure 3b, has been selected. However, one should realize that the decoupling method will work with either prescribed condition.

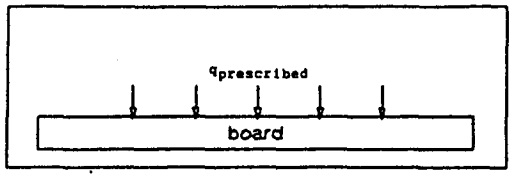
Heat conduction in the decoupled board model, shown in Figure 3c, is described by Eq.(4). The application of convective boundary conditions on the bottom and four side surfaces of the board is expressed using Eq.(5). Upon decoupling the PCB, the top surface of the decoupled board appears to be completely exposed, hence, the need arises to more narrowly define $r_{b,s}$. Choosing $r_{b,s_{ni}}$ to represent the non-interface (ni) area on the top surface of the decoupled



(a) Decoupling procedure



(b) Decoupled chip with prescribed temp. b.c.



(c) Decoupled board with prescribed heat flux

Figure 3: Decoupled components and their prescribed boundary conditions.

board model, the convection boundary condition of Eq.(5) can more precisely be expressed for the top surface of the board as

$$h(T_{b,s_{ni}}, T_\infty)(T_{b,s_{ni}} - T_\infty) = -k(T_{b,s_{ni}})(\frac{\partial T_{b,s_{ni}}}{\partial r_{b,s_{ni}}}) \quad (9)$$

As previously mentioned, only one of the two interface boundary conditions can be applied to these decoupled models. Since the temperature boundary condition has already been applied to the decoupled chip, the heat flux boundary condition, shown as

$$q_{b,i} = (q_{c,i})_{output} \quad (10)$$

must be prescribed on the board interface area in order to maintain equivalence between the coupled and decoupled analytical models.

The mathematical dependency between these two decoupled models results from the fact that while each model contains only one prescribed interface boundary condition, both conditions must exist on both models to obtain the steady state solution of the original coupled model. It would take unprecedented expertise to arbitrarily prescribe chip interface temperatures and board interface heat fluxes such that the steady state decoupled analytical solutions provided chip interface heat fluxes and board interface temperatures equivalent to those input as boundary conditions. To make this equivalence attainable, the following iterative procedure has been developed.

3.2 ITERATIVE PROCEDURE

In order to obtain a steady state solution identical to that provided by the coupled approach, an iterative procedure must be implemented to obtain convergence of boundary conditions between every decoupled interface. Rather than arbitrarily prescribing interface boundary conditions for both the decoupled chip and board models, only one condition must be prescribed using this iterative approach. To make initial guessing easier, the temperature boundary condition will

be prescribed as an initial input to the iterative process; however, this approach will work regardless of which boundary condition is initially input.

Once prescribing the initial chip-interface temperature, $(T_{c,i})^0_{prescribed}$, the iterative process begins and follows the procedure outlined in Figure 4. The chip model is solved first, after which the steady state chip-interface heat-flows, $(q_{c,i})^0_{ss}$, are obtained and applied to the corresponding board-interface area as prescribed heat-flow boundary-conditions, $(q_{b,i})^0_{prescribed}$. This form of substitutive iteration frequently utilizes a relaxation coefficient, λ, to obtain quicker convergence. The following general equation provides a means of calculating the relaxed value of $(q_{b,i})^j_{prescribed}$ for iterations after the first.

$$(q_{b,i})^j_{prescribed} = (1-\lambda)(q_{b,i})^{j-1}_{prescribed} + \lambda(q_{c,i})^j_{ss} \qquad (11)$$

While this iteration method has been selected, others such as the Newton Method or the Secant Method could have been used (Hornbeck, 1975).

Following the flow chart in Figure 4, the board model is then solved, providing the steady state board interface temperatures, $(T_{b,i})^0_{ss}$. For iterations after the first, these temperatures are checked for convergence with the previously prescribed chip interface temperatures, $(T_{c,i})^j_{prescribed}$. If convergence is not obtained, the $(T_{b,i})^j_{ss}$ are applied as the next iteration's chip interface boundary conditions, $(T_{c,i})^{j+1}_{prescribed}$. The general equation for calculating the relaxed value for $(T_{c,i})^{j+1}_{prescribed}$ is:

$$(T_{c,i})^{j+1}_{prescribed} = (1-\lambda)(T_{c,i})^j_{prescribed} + \lambda(T_{b,i})^j_{ss} \qquad (12)$$

This iterative process can be extended to multiple chip configurations in a similar manner, provided all the chips are solved prior to solving the board model. Iterations cease when convergence of interface temperatures is obtained.

4 IMPLEMENTATION

The decoupling and iterative methods described above form the core around which the current algorithm has been developed. These methods utilize numerical solutions which, in this case, are provided by the ANSYS Finite Element Program. For reference purposes, this Decoupled, Iterative, Numerical-TEChnique algorithm is named DINTEC. DINTEC controls the decoupling and iterative processes while utilizing the numerical solutions from any standard finite element code, such as ANSYS.

DINTEC's development and computer operation utilizes Fortran77, ANSYS, and the Restructured Extended Executor language (REXX). The REXX batch environment controls the intermittent execution of Fortran77 programs and ANSYS analyses. DINTEC is currently operating under the CMS environment on an IBM 3090 supercomputer, and the algorithm is capable of operating in both scalar and vector mode.

It should be mentioned that with only minor changes, DINTEC could be programmed to operate in a different batch environment or utilizing a different finite element code. The finite element code used will determine both the construction and composition of the decoupled models used by DINTEC.

5 MODELING OF DECOUPLED COMPONENTS

The finite element modeling procedure and the necessary input statements will be presented for the decoupled chip and board models.

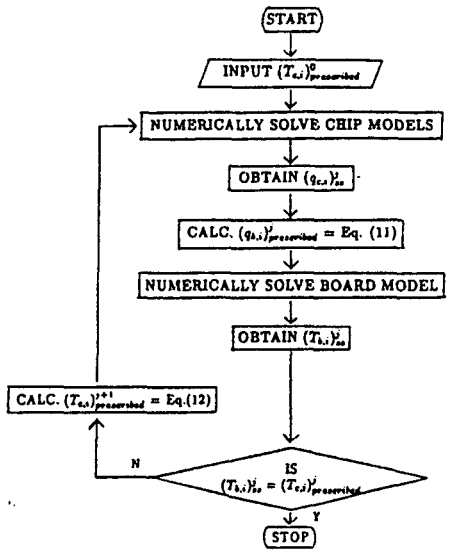


Figure 4: Flow chart of iterative procedure.

5.1 CHIP MODEL

Modeling a decoupled chip is very similar to modeling a coupled PCB in that both approaches have the same sequential file contents of: analysis definition, property definition, mesh generation, and boundary condition specification. The principal difference between the two approaches is that the decoupled chip model must contain an input statement allowing interface temperature boundary conditions to be read from an external file, whereas in the coupled approach, no such interface boundary conditions exist. These boundary conditions are programmed as variable input due to the iterative nature of the solution technique. Decoupled chip models must also contain post-processing statements that write steady state interface heat flows to an external file, which, after being manipulated, can be utilized as input to the decoupled board model.

5.2 BOARD MODEL

Decoupled board models have the same sequential file format that coupled PCBs have except that decoupled board models must contain preprocessing statements that read interface heat-flow boundary-conditions from an external file. Also unique to these decoupled board models are postprocessing statements that write steady-state interface-temperatures to an external file, which, after being manipulated, can be used as input to a decoupled chip in the next interation.

The finite element mesh of a decoupled board is dependent on the specified locations of decoupled chips, as shown in Figure 5. The decoupling procedure has the inherent restriction of a 1:1 node correspondence between chip/board interface areas. This restriction creates a fine board mesh at chip-interface locations. The element size within this board interface area is bounded by the corresponding chip size and furthermore by the mesh size within the chip model. Working with this limitation, the only way to decrease the total number of board elements is to gradually increase the mesh size with distance from the interface area, as shown in Figure 5.

Taking all of this into consideration requires that the mesh-generation section of the board model file be very flexible. Possible ways of creating an operable board mesh include using a sophisticated optimization routine or allowing the user to interactively

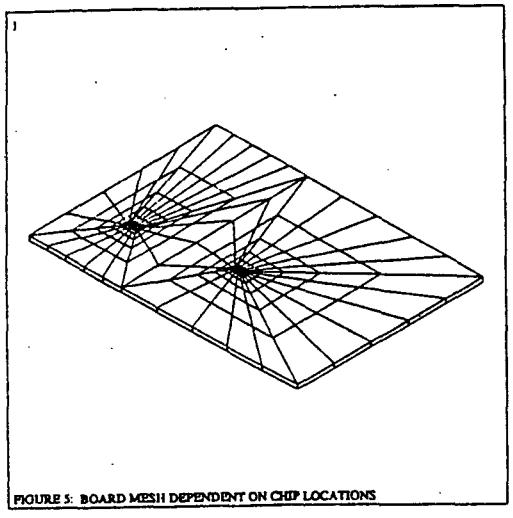


Figure 5: Board mesh dependent on chip locations.

divide the board into sections. This latter board meshing technique requires some geometrical input from the user. Depending on where the chips are positioned, the user must interactively divide the board into rectangular regions, each containing a single chip. These divisions are shown by dashed lines in Figure 6.

This procedure for creating a mesh for use with finite elements requires that the division information input by the user be written to the board-model file. , Hence, all decoupled board models must contain a command to read in the division information entered by the user. Such variable input is programmed into the statements of the finite element models of each component.

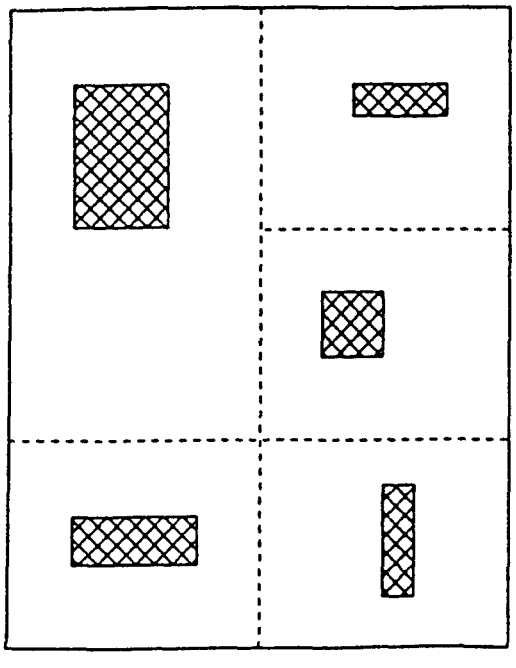


Figure 6: Typical board divisions for decoupling approach.

5.3 COMPONENT LIBRARY

The advantage of decoupling a PCB and creating separate component models is that the same chip or board will never have to be remodeled, provided the original models are stored in a component library after their initial construction. This library will be utilized to store decoupled chip and board models for use in redesigns of the same PCB or even for use in future designs. As the number of models in the library accumulates, chances are many newly proposed PCB configurations will contain previously modeled components. In fact, some newly proposed PCBs could even be modeled without generating a single component model.

6 ALGORITHM STRUCTURE

DINTEC's structure is comprised of an initial preparatory segment followed by the iterative solution segment. Both segments, as described below, are controlled by a batch program written in REXX.

6.1 PREPARATORY SEGMENT

This segment of the algorithm contains REXX statements which execute a sequence of Fortran77 programs. The first Fortran77 program provides the user the opportunity to generate new finite element models. If this is desired, the user is prompted for input necessary to construct a chip or board model. The remaining Fortran77 programs create and initialize necessary files and prompt the user for relevant input. Table 1 has been constructed to give an overview of the more important user input.

Table 1: USER INPUT DURING PREPARATORY SEGMENT

Program Prompt	User Input	Comments
Generate new models?	Yes or No	If yes, model interactively.
Use prefabricated model?	Yes or No	If yes, select from library.
Enter chip locations.	*x,y chip centroid*	*Repeated for each chip.*
Enter board sections.	x,y nodal coordinates	Several inputs required.
Enter chip mesh size.	Elem. div. in x,y dir.	$Elem_x = Elem_y$ not req'd.
Enter $(T_{c,i})_{prescribed}$	$(T_{c,i})_{prescribed}$	*Uniform for every chip.*
Enter convergence criteria.	Temp. *value*	$0.0 < value \leq 1.0$

6.2 SOLUTION SEGMENT

This segment does not require any user input and is primarily composed of the iterative loop shown in Figure 4. Controlled by REXX statements, this loop intermittently executes Fortran77 programs and ANSYS analyses until convergence of interface temperatures has been obtained. Once obtained, general iterative statistics are provided as well as an ANSYS output FILE12 for each component in the configuration. Temperature distributions as well as other steady state information are readily obtainable from this output.

7 SAMPLE PROBLEM

Consider the example PCB configuration with two chips shown in Figure 7. It is desired to obtain the temperature distribution across this PCB by utilizing decoupled ANSYS solutions within DINTEC. Assuming that decoupled models for these two chips and board exist in the component library, the preparatory segment will skip over the model generation and prompt the user for necessary inputs, of which the most important are shown in Table 2.

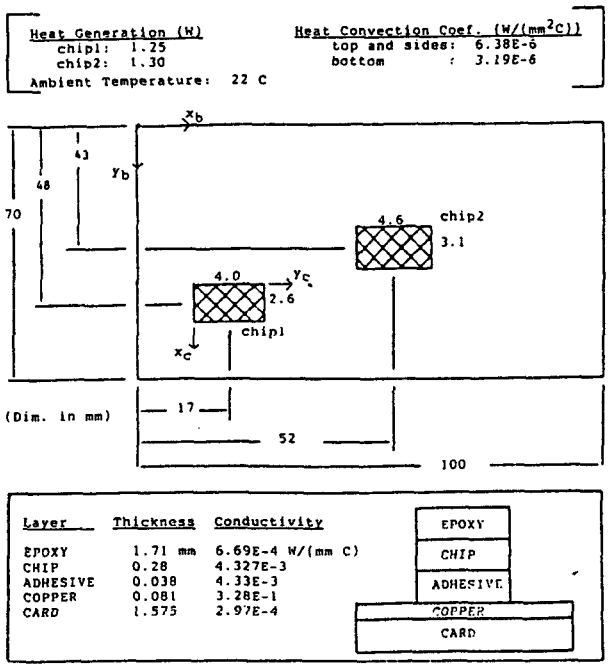


Figure 7: Input board for sample problem.

Table 2: USER INPUT FOR SAMPLE PROBLEM		
Program Prompt	User Input	Comments
Generate new models?	No	Models already exist.
Use prefabricated model?	Yes	Select 2 chips and board.
Enter location of chip1 in mm.	17,48	Centroid of chip1.
Enter location of chip2 in mm.	52,43	Centroid of chip2.
Enter board sections.	x,y nodal coordinates	See Figure 5.
Enter chip mesh size.	$Elem_x = 5, Elem_y = 5$	36 nodes at chip interface.
Enter $(T_{c,i})_{prescribed}$, in K.	400	Uniform for every chip.
Enter convergence criteria, K.	0.1	$0.0K < value \leq 1.0K$

After receiving this input, DINTEC proceeds to the noninteractive solution segment where the iterative process, as described in Figure 4, is performed until convergence is obtained. As shown in Figure 8, only three iterations are required to obtain convergence from an initially prescribed temperature of 400K. The interface temperature contours of the decoupled chip1 (Figure 9a) and of the decoupled board (Figure 9b) are provided to show how similar the temperature distributions are after only the third iteration. The equivalence of decoupled-interface temperatures is also evident from Figures 10a and 10b, which present cross-sectional temperature contours for chip1 and for that section of the board directly beneath chip1.

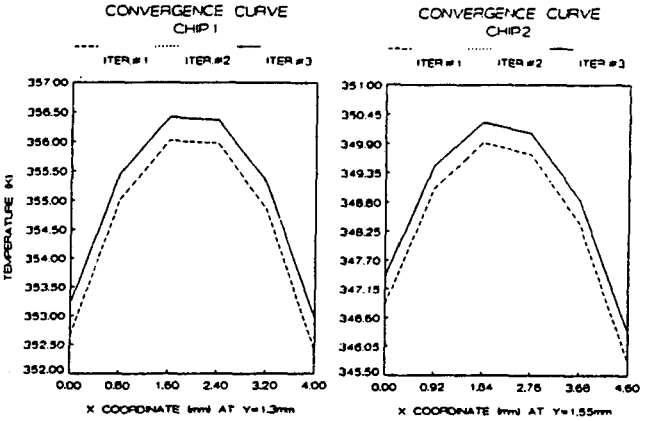


Figure 8: Convergence of interface temperatures. (Initial prescribed-interface temp.=400K.)

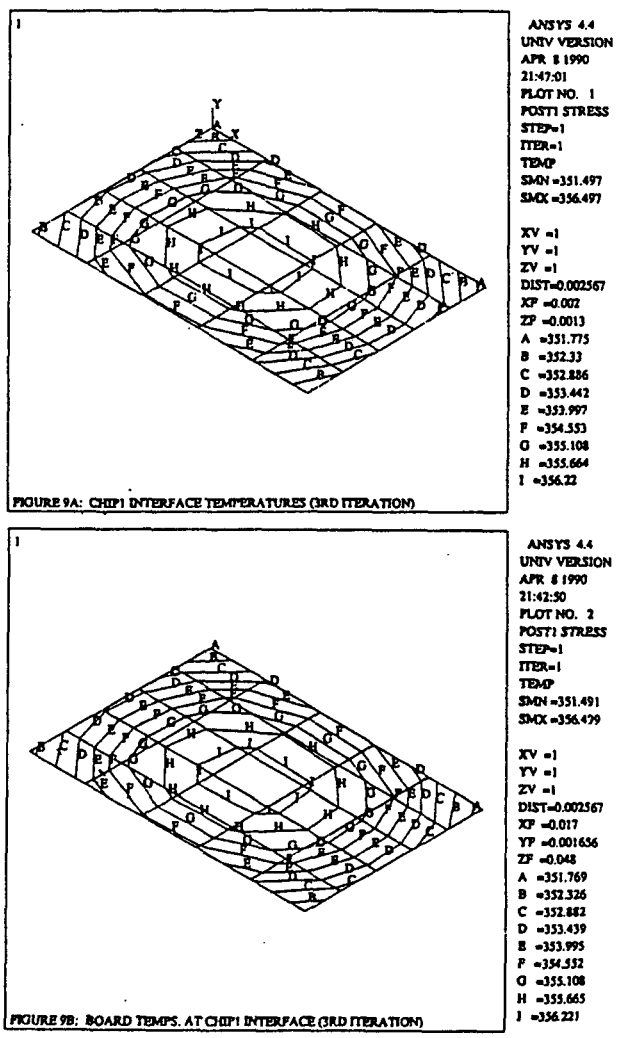


Figure 9: Interface temperature contours.
(a) Chip 1 interface
(b) Board interface

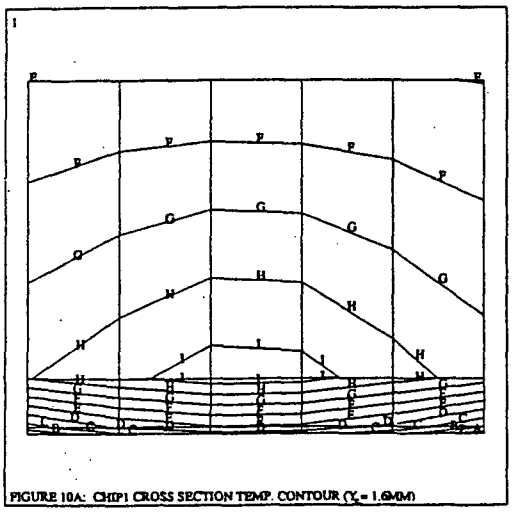


FIGURE 10A: CHIP1 CROSS SECTION TEMP. CONTOUR (Y= 1.6MM)

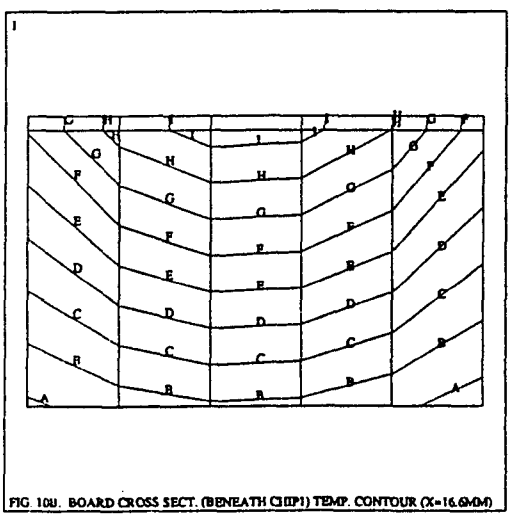


FIG 10B. BOARD CROSS SECT. (BENEATH CHIP1) TEMP. CONTOUR (X=16.6MM)

Figure 10: Cross section temperature contours.
(a) Chip 1 (Y=1.6mm)
(b) Board (X=16.6mm)

8 CONCLUDING REMARKS

In the configurations tested, DINTEC has provided steady state solutions that have converged to within 0.3 percent of the original coupled model in as few as three iterations. This accuracy should not come as a suprise since the decoupled models are equivalent to the coupled model. While this paper presented the decoupling of chips on only one side of a board, it should be realized that this procedure will also work for configurations with chips on both sides of a board. DINTEC's rapid convergence combined with the elimination of repetitive numerical-model generation makes this decoupling design-procedure attractive.

ACKNOWLEDGEMENTS

This work was supported by a grant provided by IBM. The authors would also like to acknowledge the University of Kentucky Computing Center for their assistance and support.

REFERENCES

Bar-Cohen, A., Mudawar, I. and Whalen, B., 1986, "Future Challenges for Electronic Cooling", *Research Needs in Electronic Cooling*, F.P. Incropera, (ed.) publ. National Science Foundation and Purdue University, pp. 70–77.

Funk, J.N., Mengüç, M.P., Taghavi, K., and Cremers, C.J., 1990, "An Analytical Model for Finding the Temperature Distribution on Electronic Circuit Boards", in preparation.

Haji-Sheikh, A. and Lakshminarayanan, R., 1988, "A Multimethod Study of Thermal Conduction in Bodies in Contact", *Cooling Technology for Electronic Equipment*, W. Aung, (ed.) Hemisphere Publishing Corp. pp. 641-654.

Hornbeck, R.W., *Numerical Methods*, Quantum Publishers, New York, 1975.

Pinto, E.J., and Mikic, B.B., 1986, "Temperature Prediction on Substrates and Integrated Circuit Chips", *Heat Transfer in Electronic Equipment*, A. Bar-Cohen, (ed.) ASME Publ. HTD-Vol. 57.

Nalbandian, R., 1987, "Automatic Thermal and Dynamic Analysis of Electronic Printed Circuit Boards By ANSYS Finite Element Program", *Proc. Conf. on Integrating Design and Analysis*, Newport Beach, CA, March 31–April 3, 1987, pp. 11.16–11.54.

Rodamaker, M.C. and Patton, D.D., 1987, "ANSYS Applications in Electronics", *Proc. Conf. on Int. Des. and Anal.*, Newport Beach, CA, March 31–April 3, 1987, pp. 11.1–11.15.

Shiang, J-J., Staszak, Z.J., and Prince, J.L., 1987, "APTMC: An Interface Program for Use with ANSYS for Thermal and Thermally Induced Stress Modeling/Simulation of VLSI Packaging", *Proc. Conf. on Int. Des. and Anal.*, Newport Beach, CA, March 31–April 3, 1987, pp. 11.55-11.69.

Swanson Analysis Systems Inc., *ANSYS Engineering Analysis System*, Houston, Pennsylvania, Vers. 4.3, 1987.

with v a constant. As it happens, we already know analytically that the general solution of this equation is a wave propagating in the positive x-direction,

$$u = f(x - vt) \tag{17.1.7}$$

where f is an arbitrary function. However, the numerical strategies that we develop will be equally applicable to the more general equations represented by (17.1.1). In some contexts, equation (17.1.6) is called an *advective* equation, because the quantity u is transported by a "fluid flow" with a velocity v.

How do we go about finite differencing equation (17.1.6) (or, analogously, 17.1.1)? The straightforward approach is to choose equally spaced points along both the t- and x-axes. Thus denote

$$\begin{aligned} x_j &= x_0 + j\Delta x, \qquad j = 0, 1, \ldots, J, \\ t_n &= t_0 + n\Delta t, \qquad n = 0, 1, \ldots, N \end{aligned} \tag{17.1.8}$$

Let u_j^n denote $u(t_n, x_j)$. We have several choices for representing the time derivative term. The obvious way is to set

$$\left.\frac{\partial u}{\partial t}\right|_{j,n} = \frac{u_j^{n+1} - u_j^n}{\Delta t} + O(\Delta t) \tag{17.1.9}$$

This is called *forward Euler* differencing (cf. equation 15.1.1). While forward Euler is only first-order accurate in Δt, it has the advantage that one is able to calculate quantities at timestep $n+1$ in terms of only quantities known at timestep n. For the space derivative, we can use a second-order representation still using only quantities known at timestep n:

$$\left.\frac{\partial u}{\partial x}\right|_{j,n} = \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + O(\Delta x^2) \tag{17.1.10}$$

The resulting finite-difference approximation to equation (17.1.6) is called the FTCS representation (Forward Time Centered Space),

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -v\left(\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right) \tag{17.1.11}$$

which can easily be rearranged to be a formula for u_j^{n+1} in terms of the other quantities. The FTCS scheme is illustrated in Figure 17.1.1. It's a fine example of an algorithm that is easy to derive, takes little storage, and executes quickly. Too bad it doesn't work! (See below.)

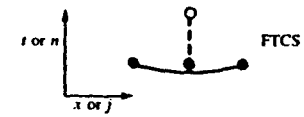


Figure 17.1.1. Representation of the Forward Time Centered Space (FTCS) differencing scheme. In this and subsequent figures, the open circle is the new point at which the solution is desired; filled circles are known points whose function values are used in calculating the new point; the solid lines connect points that are used to calculate spatial derivatives; the dashed lines connect points that are used to calculate time derivatives. The FTCS scheme is generally unstable for hyperbolic problems and cannot usually be used.

The FTCS representation is an *explicit* scheme. This means that u_j^{n+1} for each j can be calculated explicitly from the quantities that are already known. Later we shall meet *implicit* schemes, which require us to solve implicit equations coupling the u_j^{n+1} for various j. (Explicit and implicit methods for ordinary differential equations were discussed in §15.6.) The FTCS algorithm is also an example of a *single-level* scheme, since only values at time level n have to be stored to find values at time level $n+1$.

von Neumann Stability Analysis

Unfortunately, equation (17.1.11) is of very limited usefulness. It is an *unstable* method, which can only be used (if at all) to study waves for a short fraction of one oscillation period. To find alternative methods with more general applicability, we must introduce the *von Neumann stability analysis*.

The von Neumann analysis is local: we imagine that the coefficients of the difference equations are so slowly varying as to be considered constant in space and time. In that case, the independent solutions, or *eigenmodes*, of the difference equations are all of the form

$$u_j^n = \xi^n e^{ikj\Delta x} \tag{17.1.12}$$

where k is a real spatial wave number (which can have any value) and $\xi = \xi(k)$ is a complex number that depends on k. The key fact is that the time dependence of a single eigenmode is nothing more than successive integer powers of the complex number ξ. Therefore, the difference equations are unstable (have exponentially growing modes) if $|\xi(k)| > 1$ for *some* k. The number ξ is called the *amplification factor* at a given wave number k.

To find $\xi(k)$, we simply substitute (17.1.12) back into (17.1.11). Dividing by ξ^n, we get

$$\xi(k) = 1 - i\frac{v\Delta t}{\Delta x}\sin k\Delta x \tag{17.1.13}$$

whose modulus is > 1 for *all* k; so the FTCS scheme is unconditionally unstable.

If the velocity v were a function of t and x, then we would write v_j^n in equation (17.1.11). In the von Neumann stability analysis we would still treat v as a constant, the idea being that for v slowly varying the analysis is local. In fact, even in the case of strictly constant v, the von Neumann analysis does not rigorously treat the end effects at $j = 0$ and $j = N$.

More generally, if the equation's right-hand side were nonlinear in u, then a von Neumann analysis would linearize by writing $u = u_0 + \delta u$, expanding to linear order in δu. Assuming that the u_0 quantities already satisfy the difference equation exactly, the analysis would look for an unstable eigenmode of δu.

Despite its lack of rigor, the von Neumann method generally gives valid answers and is much easier to apply than more careful methods. We accordingly adopt it exclusively. (See, for example, Richtmyer and Morton for a discussion of other methods of stability analysis.)

Lax Method

The instability in the FTCS method can be cured by a simple change due to Lax. One replaces the term u_j^n in the time derivative term by its average (Figure 17.1.2):

$$u_j^n \to \frac{1}{2}\left(u_{j+1}^n + u_{j-1}^n\right) \tag{17.1.14}$$

This turns (17.1.11) into

$$u_j^{n+1} = \frac{1}{2}\left(u_{j+1}^n + u_{j-1}^n\right) - \frac{v\Delta t}{2\Delta x}\left(u_{j+1}^n - u_{j-1}^n\right) \tag{17.1.15}$$

Substituting equation (17.1.12), we find for the amplification factor

$$\xi = \cos k\Delta x - i\frac{v\Delta t}{\Delta x}\sin k\Delta x \tag{17.1.16}$$

The stability condition $|\xi|^2 \le 1$ leads to the requirement

$$\frac{|v|\Delta t}{\Delta x} \le 1 \tag{17.1.17}$$

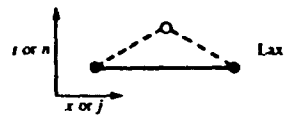


Figure 17.1.2. Representation of the Lax differencing scheme, as in the previous figure. The stability criterion for this scheme is the Courant condition.

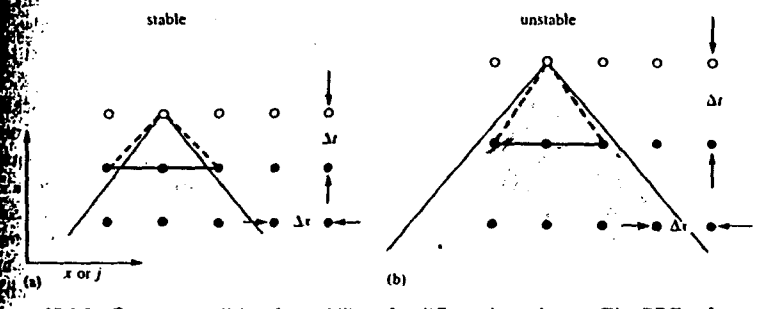


Figure 17.1.3. Courant condition for stability of a differencing scheme. The PDEs of an initial value problem imply that the value at a point depends on information within some domain of dependency to the past, shown here shaded. A differencing scheme has its own domain of dependency determined by the choice of points on one time slice (shown as connected solid dots) whose values are used in determining a new point (shown connected by dashed lines). A differencing scheme is Courant stable if the differencing domain of dependency is larger than that of the PDE's, as in (a), and unstable if the relationship is the reverse, as in (b).

This is the famous Courant-Friedrichs-Lewy stability criterion, often called simply the *Courant condition*. Intuitively, the stability condition can be understood as follows (Figure 17.1.3): The quantity u_j^{n+1} in equation (17.1.15) is computed from information at points $j-1$ and $j+1$ at time n. In other words, x_{j-1} and x_{j+1} are the boundaries of the spatial region that is allowed to communicate information to u_j^{n+1}. Now recall that in the continuum wave equation, information actually propagates with a maximum velocity v. If the point u_j^{n+1} is outside of the shaded region in Figure 17.1.3, then it requires information from points more distant than the differencing scheme allows. Lack of that information gives rise to an instability. Therefore, Δt cannot be made too large.

The surprising result, that the simple replacement (17.1.14) stabilizes the FTCS scheme, is our first encounter with the fact that differencing PDEs is an art as much as a science. To see if we can demystify the art somewhat, let us compare the FTCS and Lax schemes by rewriting equation (17.1.15) so that it is in the form of equation (17.1.11) with a remainder term:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = v\left(\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right) + \frac{1}{2}\left(\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta t}\right) \tag{17.1.18}$$

But this is exactly the FTCS representation of the equation

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2\Delta t}\nabla^2 u \tag{17.1.19}$$

where $\nabla^2 = \partial^2/\partial x^2$ in one dimension. We have, in effect, added a diffusion term to the equation, or, if you recall the form of the Navier-Stokes equations for viscous fluid flow, a dissipative term. The Lax scheme is thus said to have *numerical dissipation*, or *numerical viscosity*. We can see this also in the amplification factor. Unless $|v|\Delta t$ is exactly equal to Δx, $|\xi| < 1$ and the amplitude of the wave decreases spuriously.

Isn't a spurious decrease as bad as a spurious increase? No. The scales that we hope to study accurately are those that encompass many gridpoints so that they have $k\Delta x \ll 1$. (The spatial wave number k is defined by equation 17.1.12.) For these scales, the amplification factor can be seen to be very close to one, in both the stable and unstable scheme. The stable and unstable schemes are therefore about equally accurate. For the unstable scheme, however, short scales with $k\Delta x \sim 1$, *which we are not interested in*, will blow up and swamp the interesting part of the solution. Much better to have a stable scheme in which these short wavelengths die away innocuously. Both the stable and the unstable schemes are *inaccurate* for these short wavelengths, but the inaccuracy is of a tolerable character when the scheme is stable.

When the independent variable $\mathbf{u}$ is a vector, then the von Neumann analysis is slightly more complicated. For example, we can consider equations (17.1.3), rewritten as

$$\frac{\partial}{\partial t}\begin{bmatrix} r \\ s \end{bmatrix} = -\frac{\partial}{\partial x}\begin{bmatrix} vs \\ vr \end{bmatrix} \tag{17.1.20}$$

The Lax method for this equation is

$$\begin{aligned} r_j^{n+1} &= \frac{1}{2}(r_{j+1}^n + r_{j-1}^n) + \frac{v\Delta t}{2\Delta x}(s_{j+1}^n - s_{j-1}^n) \\ s_j^{n+1} &= \frac{1}{2}(s_{j+1}^n + s_{j-1}^n) + \frac{v\Delta t}{2\Delta x}(r_{j+1}^n - r_{j-1}^n) \end{aligned} \tag{17.1.21}$$

The von Neumann stability analysis now proceeds by assuming that the exact mode is of the following (vector) form,

$$\begin{bmatrix} r_j^n \\ s_j^n \end{bmatrix} = \xi^n e^{ikj\Delta x}\begin{bmatrix} r^0 \\ s^0 \end{bmatrix} \tag{17.1.22}$$

Here the vector on the right-hand side is a constant (both in space and in time), eigenvector, and ξ is a complex number, as before. Substituting (17.1.22)

(17.1.21), and dividing by the power ξ^n, gives the homogeneous vector tion

$$\begin{bmatrix} (\cos k\Delta x) - \xi & i\dfrac{v\Delta t}{\Delta x}\sin k\Delta x \\ i\dfrac{v\Delta t}{\Delta x}\sin k\Delta x & (\cos k\Delta x) - \xi \end{bmatrix} \cdot \begin{bmatrix} r^0 \\ s^0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{17.1.23}$$

admits a solution only if the determinant of the matrix on the left vanes, a condition easily shown to yield the two roots ξ

$$\xi = \cos k\Delta x \pm i\frac{v\Delta t}{\Delta x}\sin k\Delta x \tag{17.1.24}$$

e stability condition is that both roots satisfy $|\xi| \le 1$. This again turns t to be simply the Courant condition (17.1.17).

Other Varieties of Error

Thus far we have been concerned with *amplitude error*, because of its timate connection with the stability or instability of a differencing scheme. Other varieties of error are relevant when we shift our concern to accuracy, ather than stability.

Finite-difference schemes for hyperbolic equations can exhibit dispersion, or *phase errors*. For example, even if we set $v\Delta t/\Delta x = 1$ in equation (17.1.16), we find

$$\xi = e^{-ik\Delta x} \tag{17.1.25}$$

An arbitrary initial wave packet is a superposition of modes with different k's. At each timestep the modes get multiplied by different phase factors (17.1.25), depending on their value of k. After a while, the phase relations of the modes can become hopelessly garbled and the wave packet disperses. Note from (17.1.25) that the dispersion becomes large as soon as the wavelength becomes comparable to the grid spacing Δx.

A third type of error is one associated with nonlinear hyperbolic equations and is therefore sometimes called *nonlinear instability*. For example, a piece of the Euler or Navier-Stokes equations for fluid flow looks like

$$\frac{\partial v}{\partial t} = -v\frac{\partial v}{\partial x} + \dots \tag{17.1.26}$$

The nonlinear term in v can cause a transfer of energy in Fourier space om long wavelengths to short wavelengths. This results in a wave profile epening until a vertical profile or "shock" develops. Since the von Neumann alysis suggests that the stability can depend on $k\Delta x$, a scheme that was