# Problema de Convección-Difusión

$$\frac{\partial u}{\partial t} + \vec{v} \cdot \vec{\nabla} u - \vec{\nabla} \cdot \left( k \vec{\nabla} u \right) = f$$

$$\frac{du}{dt} - \vec{\nabla} \cdot \left( k \vec{\nabla} u \right) = f$$

$$\frac{du}{dt} \approx \frac{u(\vec{x}, t_{n+1}) - u(\underline{\vec{x}}, t_n)}{\Delta t} = \frac{u^{n+1}(\vec{x}) - u^n(\underline{\vec{x}})}{\Delta t}$$

$$u^{n+1}(\vec{x}) - \Delta t \, \vec{\nabla} \cdot \left[ k(\vec{x}) \vec{\nabla} u^{n+1}(\vec{x}) \right] = \Delta t \, f^{n+1}(\vec{x}) + u^n(\underline{\vec{x}})$$

# Aproximación de la característica

$$\underline{x}_i = x_i(t_{n+1} - \Delta t) = x_i - v_i(\vec{x})\Delta t + \frac{\Delta t^2}{2}\vec{v}(\vec{x})\cdot\vec{\nabla}v_i(\vec{x}) + 0(\Delta t^3)$$

$$u^n(\underline{\vec{x}}) = u^n(\vec{x} - \Delta\vec{x}) = u^n(\vec{x}) - \sum_{i=1}^{2}\Delta x_i \frac{\partial u^n(\vec{x})}{\partial x_i} + \frac{1}{2}\left[2\Delta x_1\Delta x_2 \frac{\partial^2 u^n(\vec{x})}{\partial x_1 \partial x_2} + \sum_{i=1}^{2}\Delta x_i^2 \frac{\partial^2 u^n(\vec{x})}{\partial x_i^2}\right] + 0(\|\Delta\vec{x}\|^3)$$

$$u^n(\underline{\vec{x}}) = u^n(\vec{x}) - \Delta t\sum_{i=1}^{2} v_i(\vec{x})\frac{\partial u^n(\vec{x})}{\partial x_i} + \frac{\Delta t^2}{2}\sum_{i=1}^{2}\left[\vec{v}(\vec{x})\cdot\vec{\nabla}v_i(\vec{x})\right] + \frac{\Delta t^2}{2}\sum_{i=1}^{2}\sum_{j=1}^{2} v_i(\vec{x})v_j(\vec{x})\frac{\partial^2 u^n(\vec{x})}{\partial x_i \partial x_j} + 0(\Delta t^3)$$

$$u^{n+1} - \Delta t\,\vec{\nabla}\cdot\left[k\vec{\nabla}u^{n+1}\right] = \Delta t\, f^{n+1} + u^n - \Delta t\,\vec{v}\cdot\vec{\nabla}u^n + \frac{\Delta t^2}{2}\sum_i(\vec{v}\cdot\vec{\nabla}v_i)\frac{\partial u^n}{\partial x_i} + \frac{\Delta t^2}{2}\sum_{i,j} v_i v_j \frac{\partial^2 u^n}{\partial x_i \partial x_j}$$

## Estabilidad de la Formulación en 1-D

$$C \leq \left[ \frac{1}{P_e^{\,2}} + \frac{1}{3} \right]^{1/2} + \frac{1}{P_e}$$

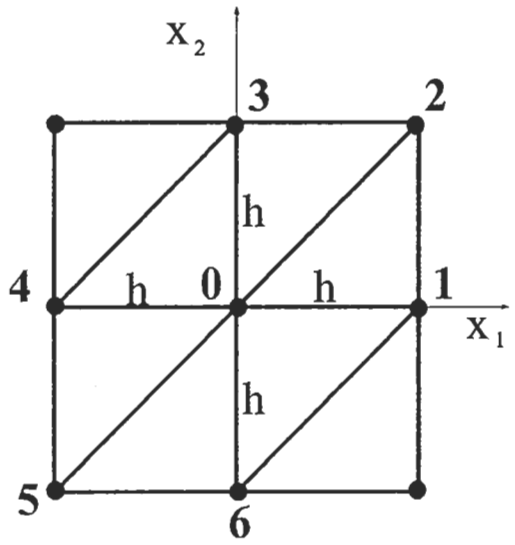
$$C = v\,\Delta t / h \qquad y \qquad P_e = v\,h / k$$

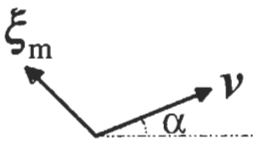
## Consistencia de la Formulación en 1-D

$$G(\eta) = G_e(\eta) + 0(\eta^3)$$

$$\eta = \xi\, h$$

# Estabilidad de la Formulación en 2-D



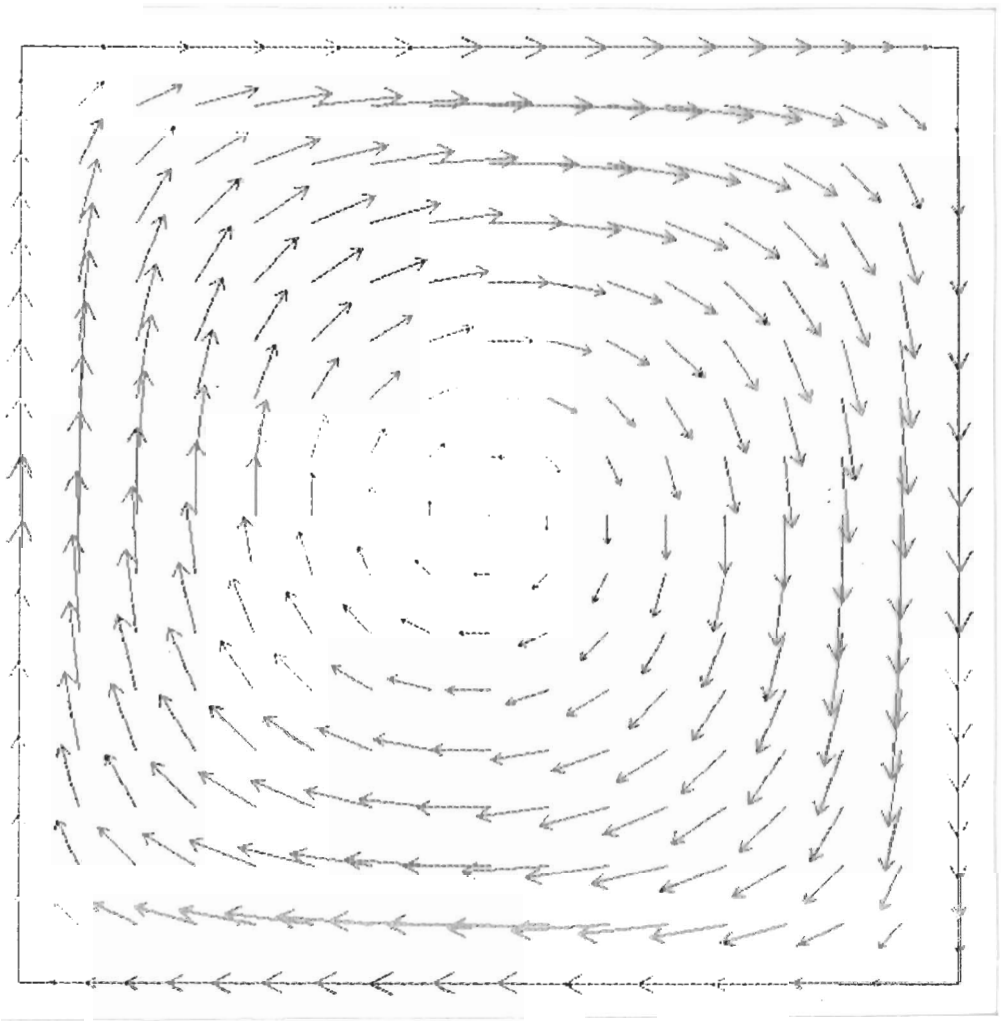




Dirección de propagación: eje $x_1$ ,

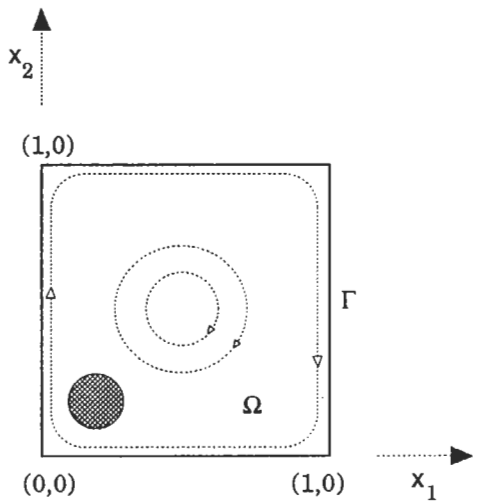
$$C \leq \frac{1}{\cos^2 \alpha}\left[\left(\frac{1}{P_e^2}+\frac{1}{3}\cos^2 \alpha\right)^{1/2}+\frac{1}{P_e}\right]$$

Dirección de propagación: $3\pi/4$,

$$C \leq \frac{1}{1-\operatorname{sen} 2\alpha}\left[\left(\frac{4}{P_e^2}+\frac{1}{3}(1-\operatorname{sen} 2\alpha)\right)^{1/2}+\frac{2}{P_e}\right]$$



# PROBLEMA DE CONVECCIÓN-DIFUSIÓN

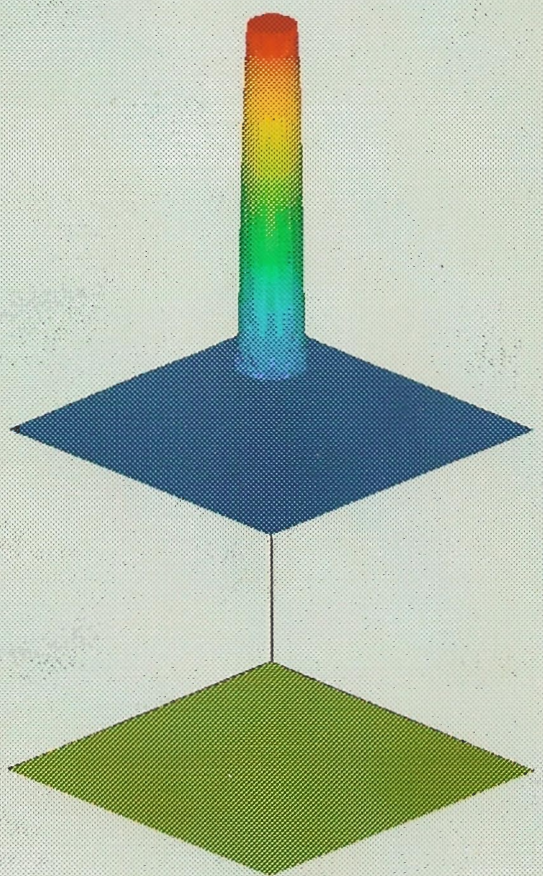



$$\left.\begin{aligned} \frac{\partial u}{\partial t} + \vec{v}\cdot\vec{\nabla}u - \vec{\nabla}\cdot(k\vec{\nabla}u) = 0 \quad &\text{en} \quad \Omega \\ k\frac{\partial u}{\partial n} = 0 \quad &\text{en} \quad \Gamma = \partial\Omega \end{aligned}\right\} \quad \text{donde} \quad \vec{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} \omega x_1 (1-x_1)(x_2 - 0.5) \\ \omega x_2 (0.5 - x_1)(1-x_2) \end{pmatrix}, \quad k = 0.1, \quad \omega = 2.5\cdot 10^4$$

$$u(\vec{x},0) = \begin{cases} 1 & \text{si} \quad d > 0.1 \\ 1 + e^{(0.1^2 - d^2)/(0.07^2 - d^2)} & \text{si} \quad 0.07 < d < 0.1 \\ 2 & \text{si} \quad d < 0.07 \end{cases}$$

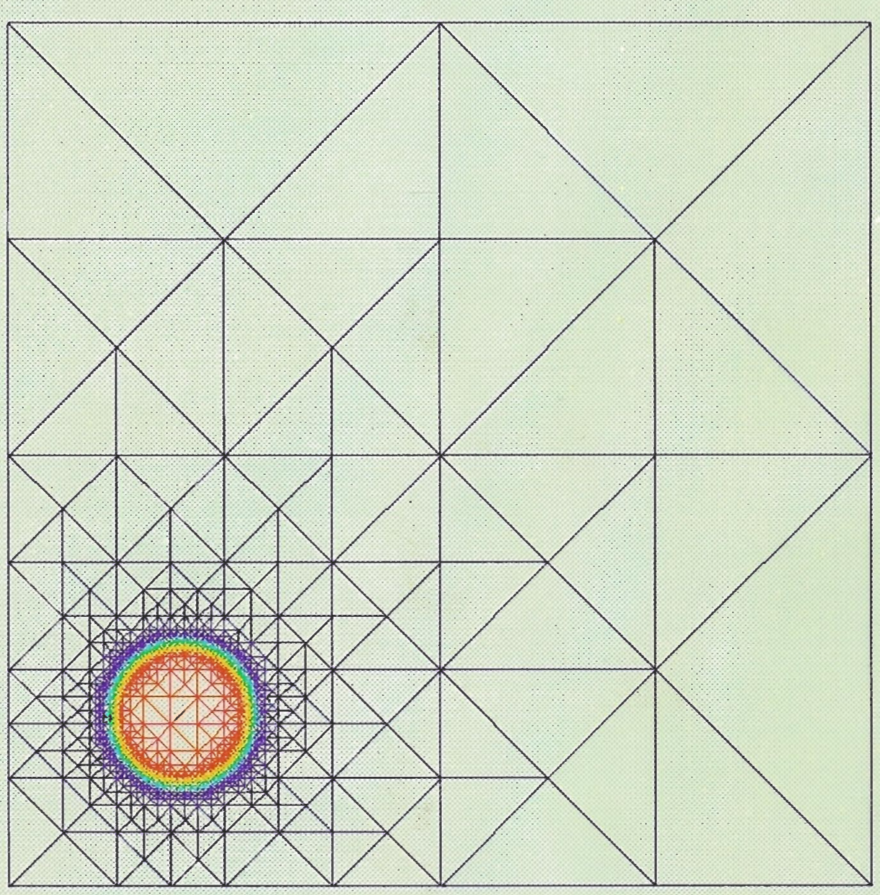
$$P_e = \frac{|\vec{v}|_{max} L}{k} = 3.125\cdot 10^4$$

# PROBLEMA DE CONVECCION-DIFUSION



SOLUCION INICIAL

# PROBLEMA DE CONVECCION-DIFUSION



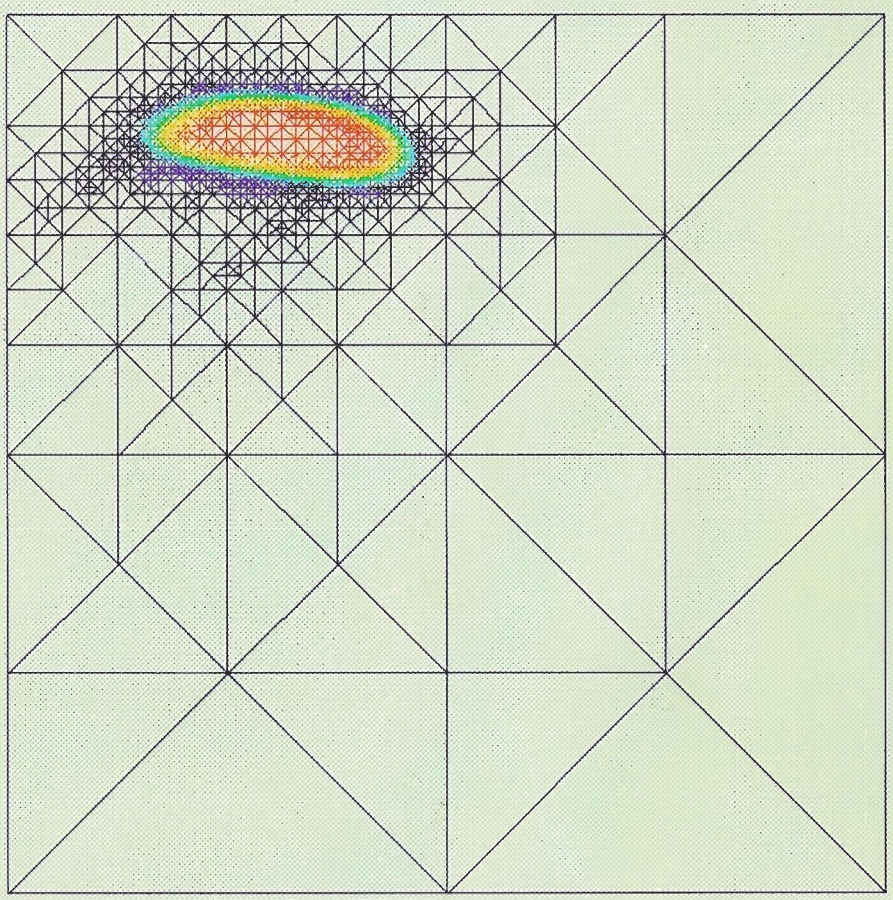
t=0.0; 1198 nodos

PROBLEMA DE CONVECCION-DIFUSION



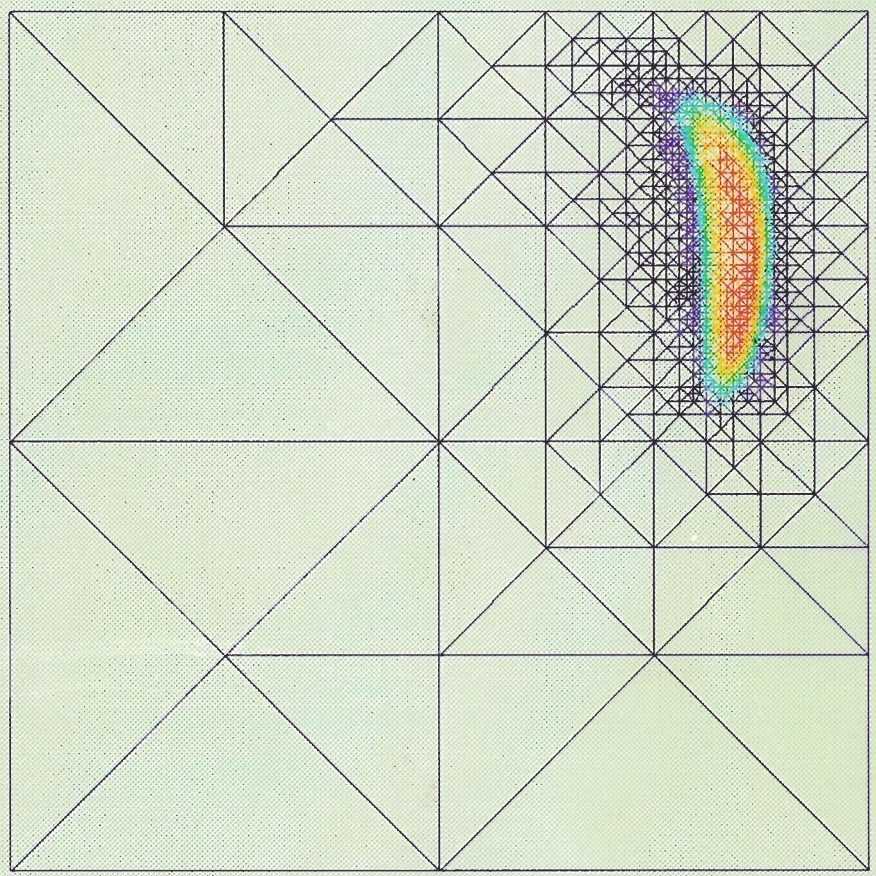
t=0.00014; 985 nodos

# PROBLEMA DE CONVECCION-DIFUSION



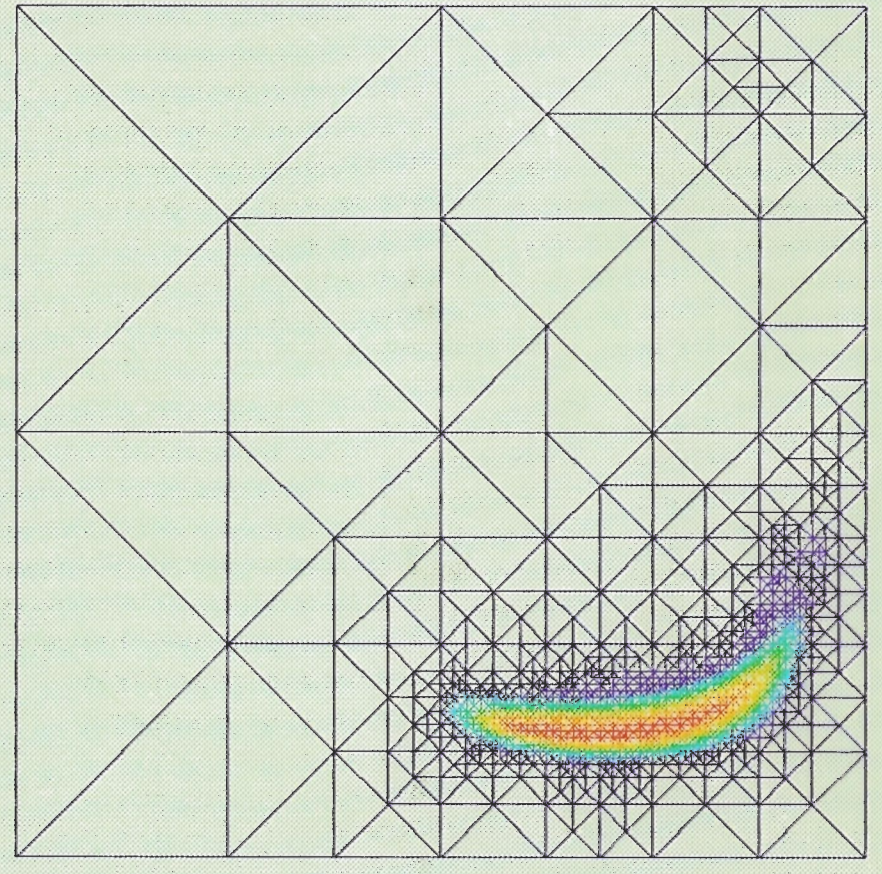
t=0.00039; 1183 nodos

# PROBLEMA DE CONVECCION-DIFUSION



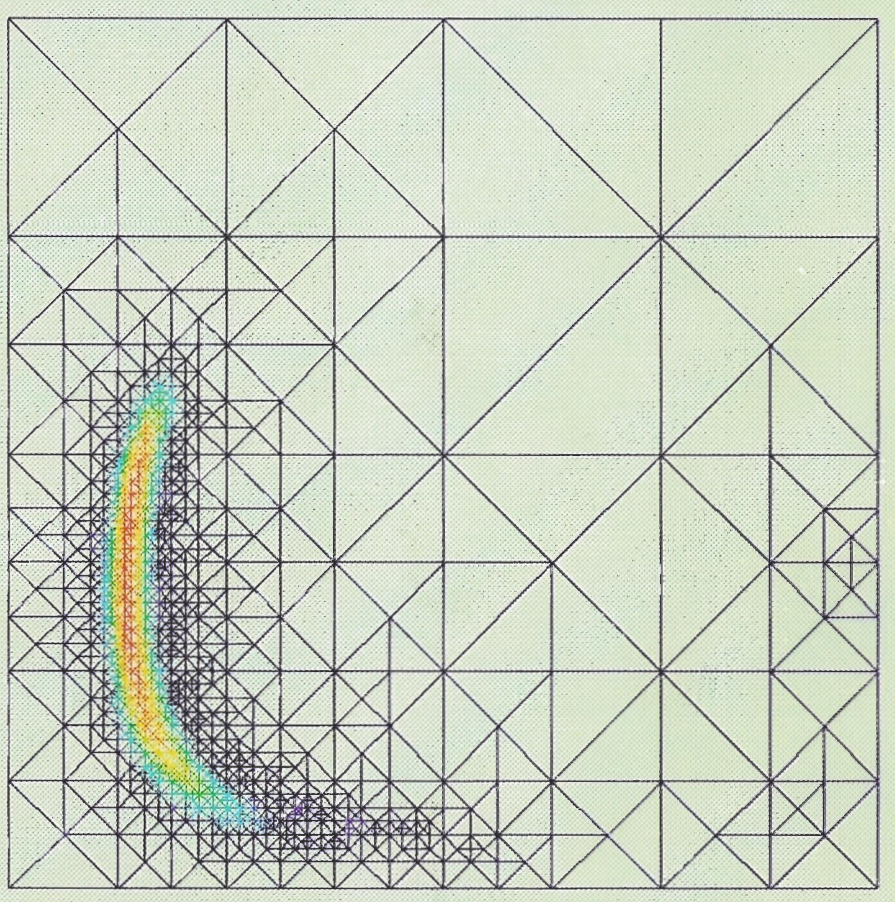
t=0.00067; 908 nodos

# PROBLEMA DE CONVECCION-DIFUSION

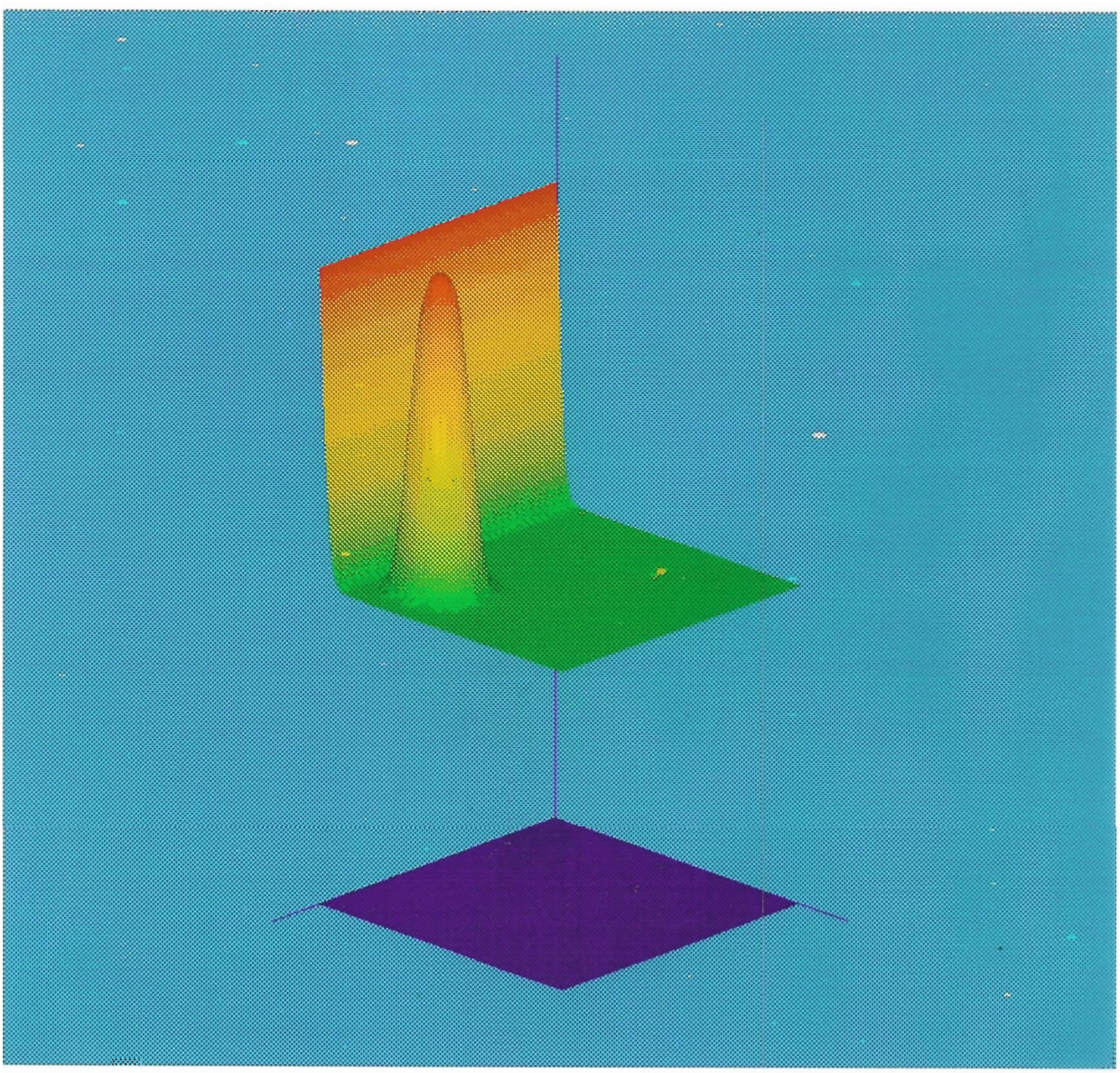


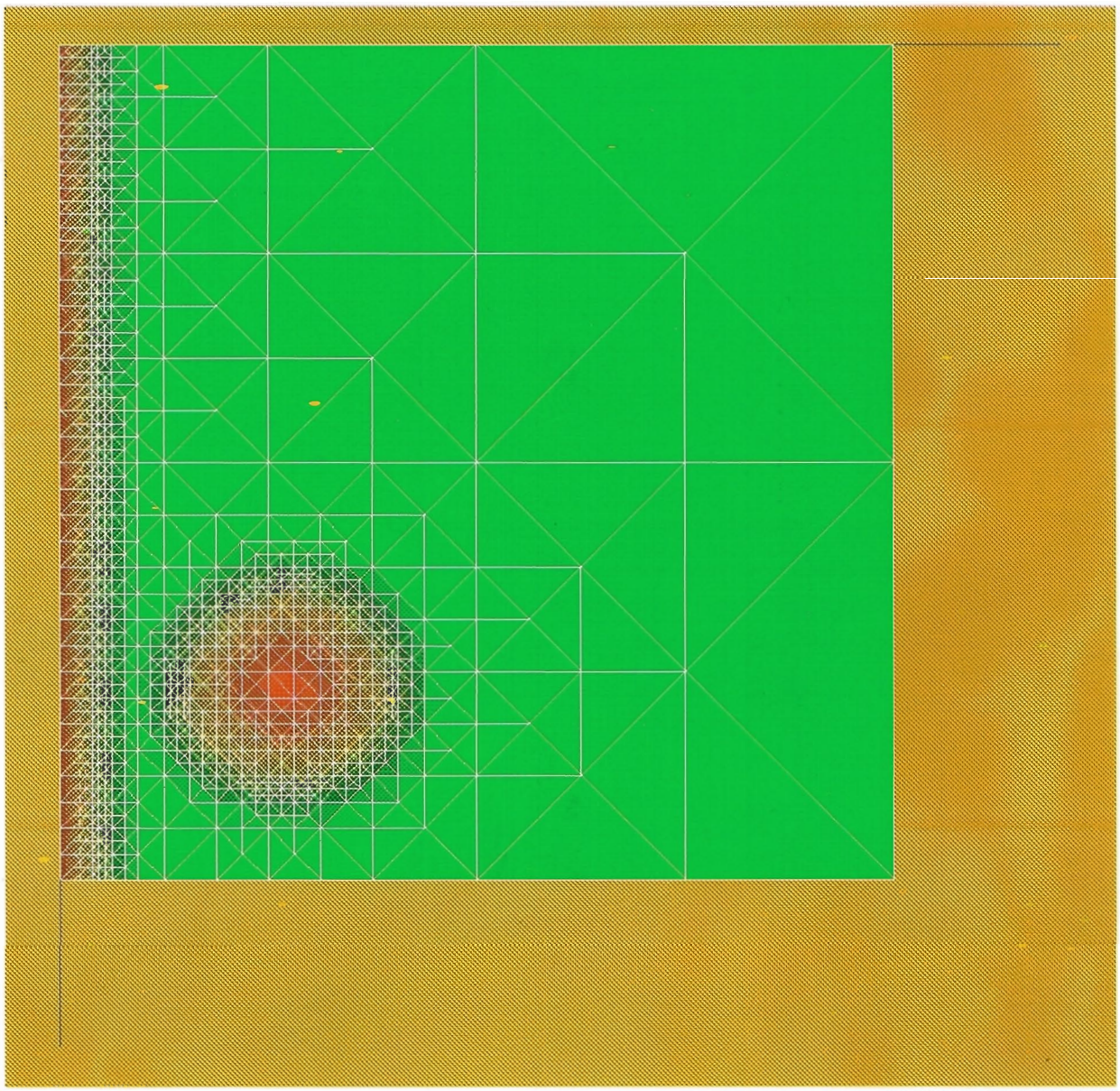
t=0.00098; 1105 nodos

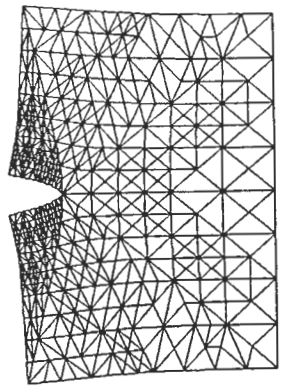
# PROBLEMA DE CONVECCION-DIFUSION



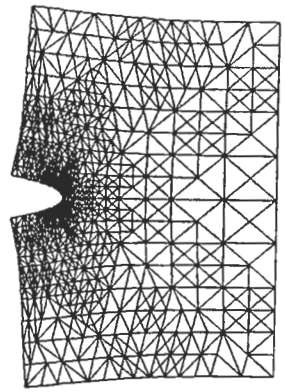
t=0.0013; 966 nodos



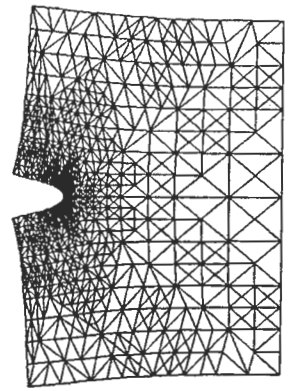




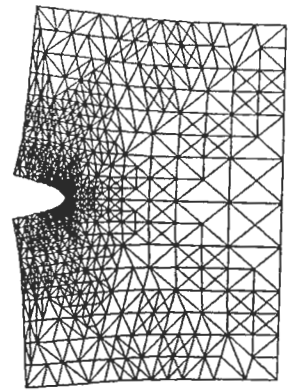
*Step 3, 421 nodes.*



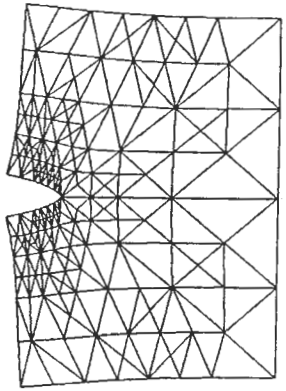
*Step 5, 797 nodes*



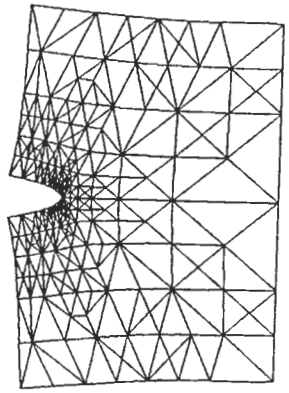
*Step 7, 1001 nodes*



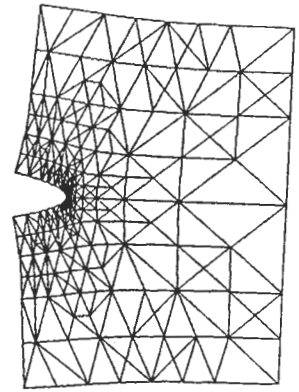
*Step 9, 1075 nodes*



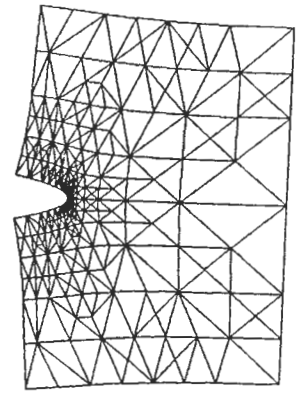
*Step 3, 175 nodes.*



*Step 5, 246 nodes*

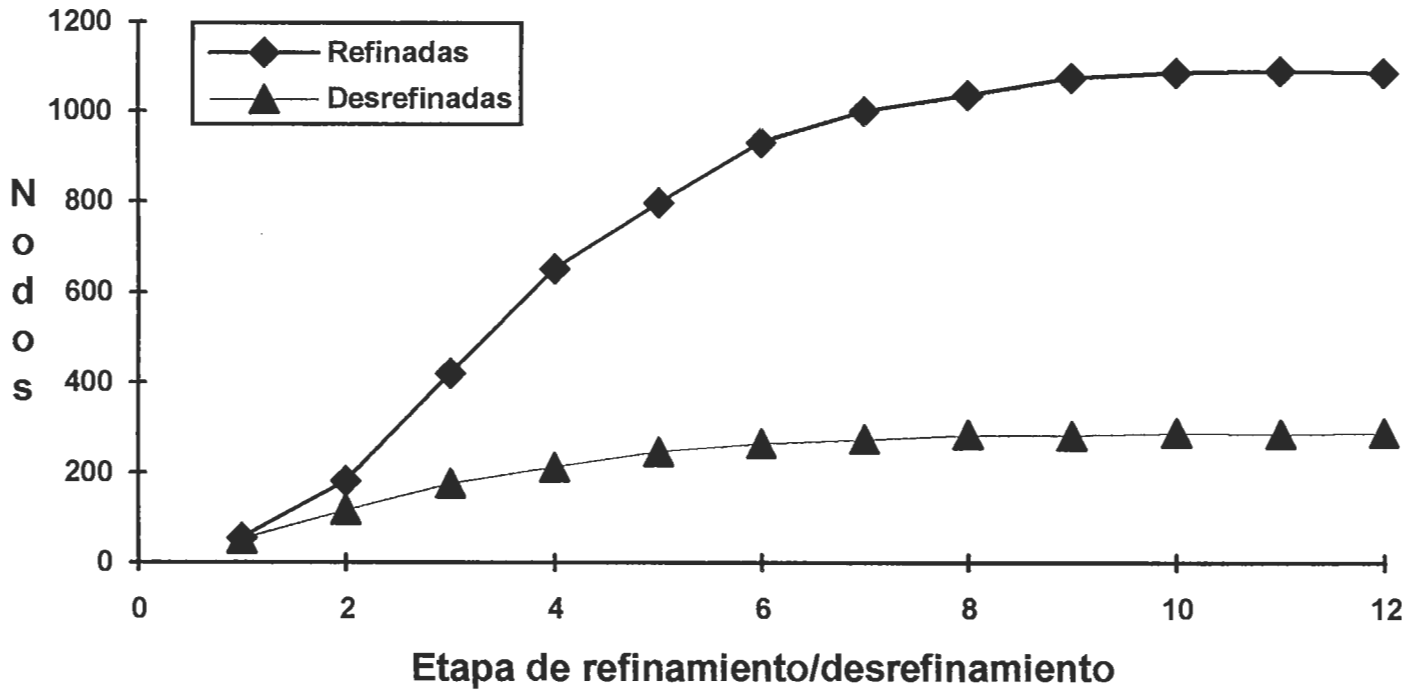


*Step 7, 273 nodes*



*Step 9, 282 nodes*

# Evolución del Número de Nodos



# CONCLUSIONS AND FUTURE RESEARCHES

**1.-** The refinement/derefinement combination of nested grids give:

- ❑ Easy applications of multigrid methods.
- ❑ Flexibility in meshes.
- ❑ Automatic control of the number of nodes.
- ❑ Bounded number of equations in the process.
- ❑ Very fast readaptation of the current mesh.
- ❑ A good method to approach any initial solution and geometry of the domain.
- ❑ A local refinement (derefining after a global refinement).

**2.-** Linear complexity of the improved derefinement algorithm.

**3.-** Generalization to three dimensional domains.

# GENERACIÓN DE MALLAS TRIDIMENSIONALES MEDIANTE LA TRIANGULACIÓN DE DELAUNAY

# CONTENIDOS

- **ENTRADA DE DATOS**

  **Definición de la geometría del objeto; puntos, líneas, superficies y regiones.**

  **Generación de puntos en el objeto.**

- **TRIANGULACIÓN DE DELAUNAY EN 3-D**

  **Algoritmo de Watson.**

  **Problemas asociados a errores de redondeo.**

  **Implementación del algoritmo.**

  **Control sobre los errores de redondeo para conseguir una triangulación aceptable.**

- **ELIMINACIÓN DE LA ENVUELTA CONVEXA DE TETRAEDROS**

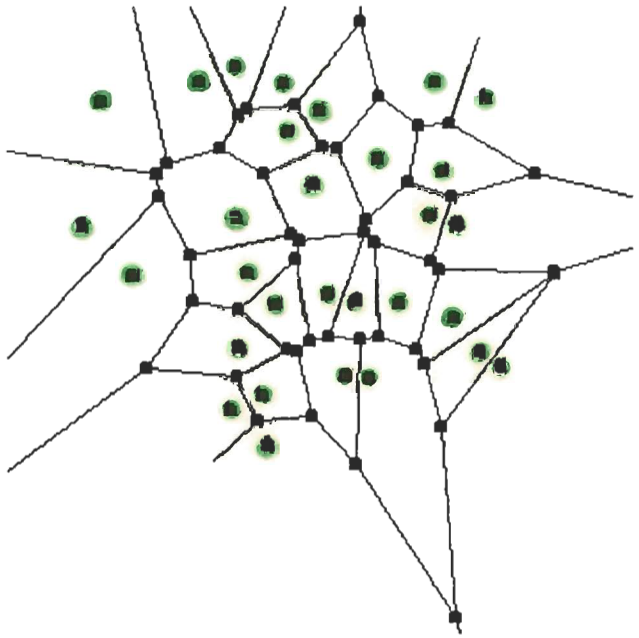
- **EJEMPLOS**

- **CONCLUSIONES Y LINEAS DE TRABAJO FUTURAS**

# Diagrama de Voronoi y Triangulación de Delaunay (I)

- Sea $X \subset E^3$ un conjunto de puntos diferentes no todos en el mismo plano.
- El poliedro de Voronoi, $V(x_i)$, asociado a $x_i \in X$, se define como:

$$V(x_i) = \{x \in E^3 \,/\, \mathrm{d}(x, x_i) \leq \mathrm{d}(x, x_j), \forall x_j \in X, i \neq j\}$$



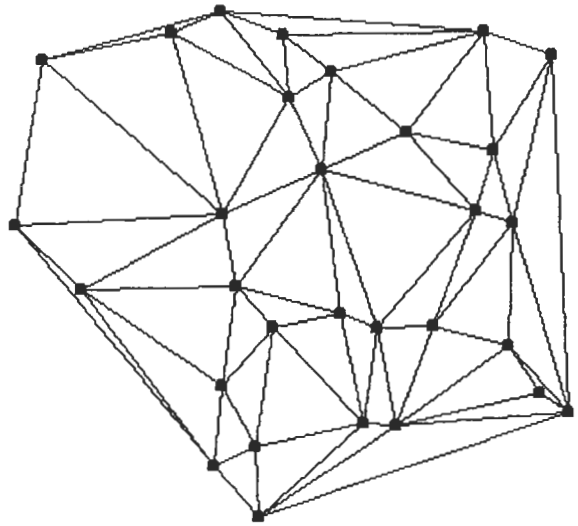
# Diagrama de Voronoi y Triangulación de Delaunay (II)

- Sea $R \subset X$ un subconjunto de cuatro o más puntos de $X$, no todos en el mismo plano, tal que existe un punto, $v_R \in E^3$, equidistante de todos los puntos de $R$ que satisface que

$$\mathrm{d}(v_R, x_k) < \mathrm{d}(v_R, x_i) \quad x_k \in R \text{ y } \forall x_i \in X - R.$$

- El politopo de Delaunay, $D(R)$, es la envolvente convexa de $R$.

- La triangulación de Delaunay, $\mathrm{D}(X)$, es el conjunto de todos los politopos de Delaunay.



# Algoritmo Incremental para la Construcción de la Triangulación de Delaunay

## 1. Triangulación inicial.

Se construye un prisma que contenga todos los puntos de $X$ y se descompone en cinco tetraedros de manera que formen una triangulación de Delaunay.

## 2. Adición del punto $x_{i+1}$ a la triangulación.

Hallamos el conjunto de tetraedros, $D_1^i$, cuyas esferas circunscritas contienen a $x_{i+1}$.

$$D_1^i = \{T_j \in D(X_i) / x_{i+1} \in B(T_j)\}$$

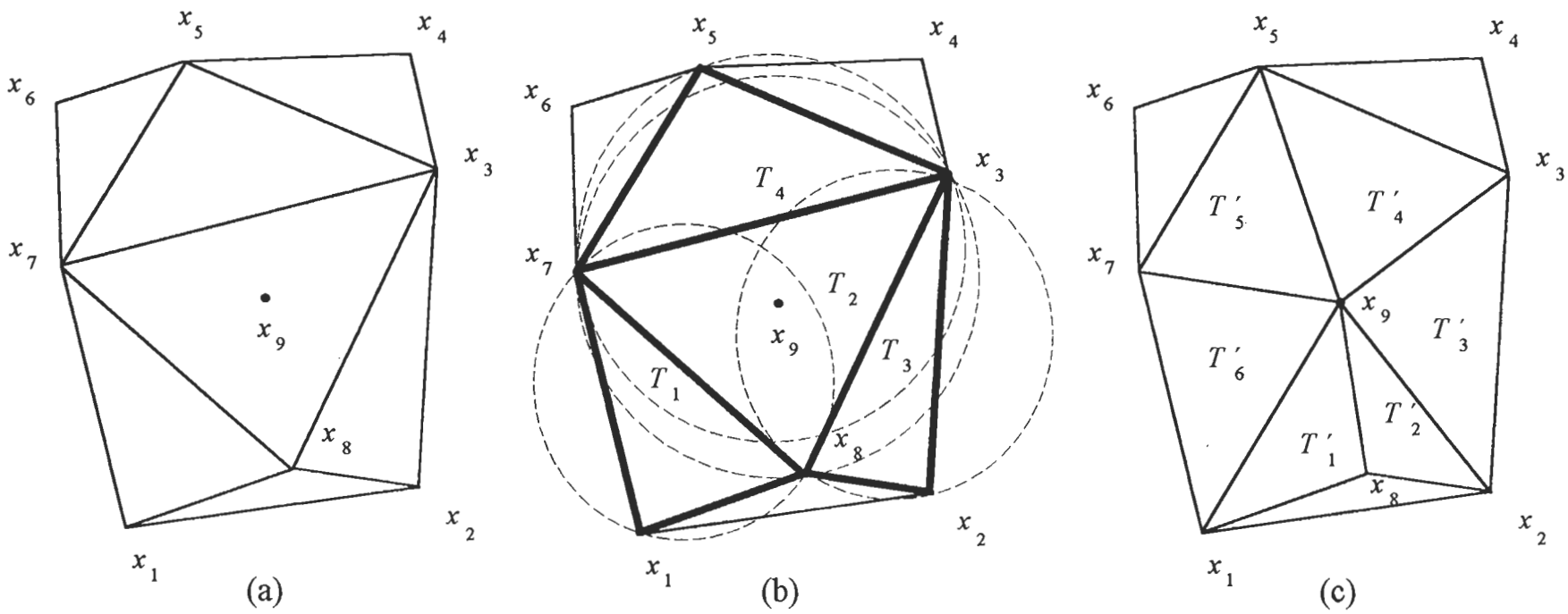
Sea $C_{f\,D_1}^i$ el conjunto de caras de la frontera de $D_1^i$, esto es, caras que sólo pertenecen a un tetraedro de $D_1^i$. Si $D(X_i)$ es la triangulación de los $i$ primeros puntos de $X$, la triangulación $D(X_{i+1})$ que incluye al punto $x_{i+1}$ se define como:

$$D(X_{i+1}) = (D(X_i) - D_1^i) \cup D_2^i$$

donde $D_2^i = \bigcup_j \{C_j, x_{i+1}\}$ es el conjunto de tetraedros formados con $x_{i+1}$ y las caras $C_j$ de $C_{f\,D_1}^i$.

Si las esferas circunscritas a los tetraedros de $D_1^i$ se calculan exactamente, el conjunto $D_1^i$ será en forma de estrella respecto a $x_{i+1}$ y no contendrá ningún otro punto de $X$ aparte de $x_{i+1}$; $D_2^i$ estará bien definido.

# Ejemplo de Adición de un Punto a la Triangulación

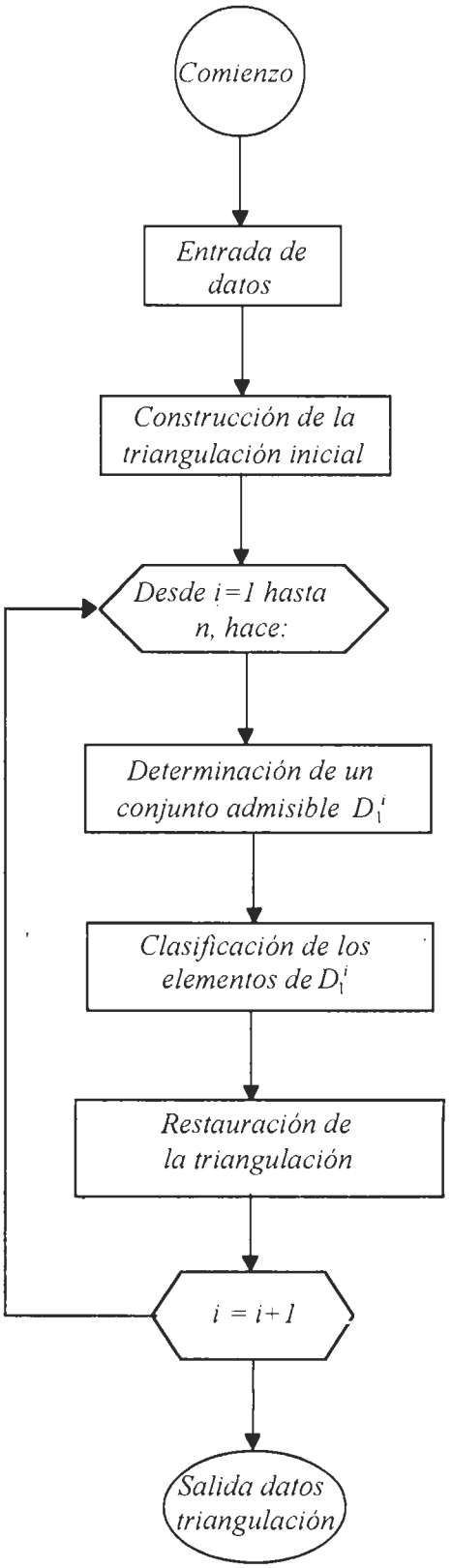



(a) Se añade el punto $x_9$ a la triangulación $\mathrm{D}(X_8)$ formada con los puntos $X_8 = \{x_1, \cdots, x_8\}$.

(b) Los tetraedros cuyas esferas circunscritas contienen a $x_9$ son $\mathrm{D}_1^8 = \{T_1, \ldots, T_4\}$.

(c) La triangulación que incluye a $x_9$ es $\mathrm{D}(X_9) = (\mathrm{D}(X_8) - \mathrm{D}_1^8) \bigcup \mathrm{D}_2^8$, donde $\mathrm{D}_2^8 = \{T_1', \ldots, T_6'\}$.

# Diagrama de flujo del programa

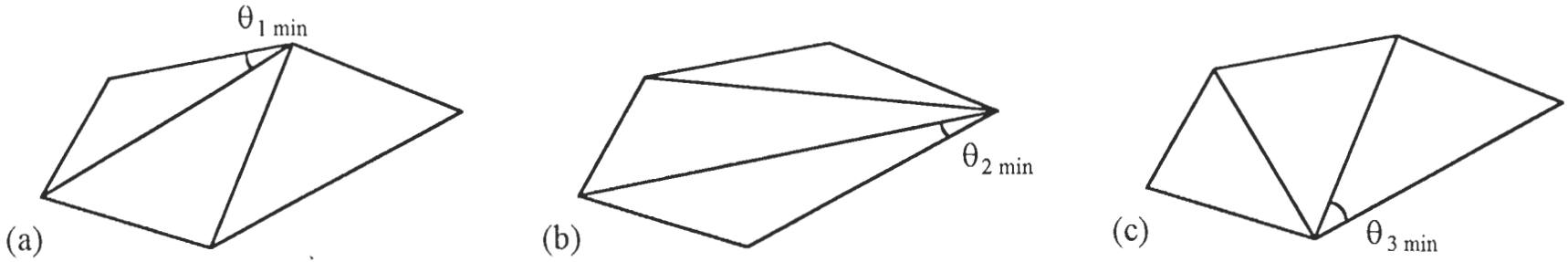


# Propiedades de la triangulación de Delaunay (III)

La triangulación de Delaunay presenta algunas propiedades que la hacen muy adecuada para la aplicación del M.E.F. En concreto, para un determinado conjunto de puntos del plano es la triangulación que produce los triángulos más regulares. Sin embargo, no existe ninguna generalización de este resultado en dimensión mayor que dos.

**Propiedad**

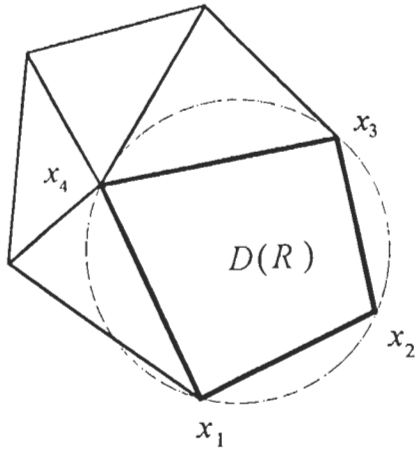
De todas las triangulaciones de un conjunto de puntos $X \subset E^2$ en posición general, la de Delaunay es aquella que hace máximo el ángulo mínimo de cualquier triángulo. □



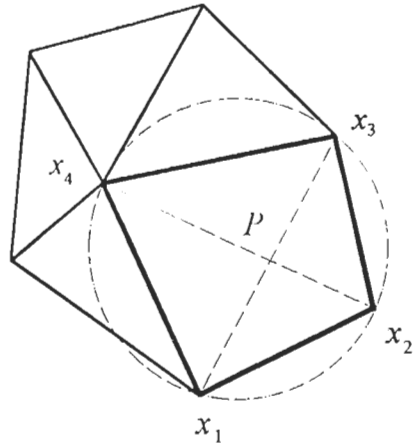

*La triangulación de Delaunay (c) es la que hace máximo el ángulo mínimo:* $\theta_{3min} > \theta_{2min} > \theta_{1min}$.

# Caso degenerado

Si $X$ no está en posición general hay celdas degeneradas (figura (a)). La descomposición en símplices de un politopo correspondiente a una celda degenerada (figura (b)) no es única Los algoritmos existentes para generar la triangulación de Delaunay tienen dificultades en esta situación. Las dificultades subsisten incluso cuando la disposición de puntos es tal que $X$ no está *claramente* en posición general, es decir, cuando hay más de $k+1$ puntos de $X$ *próximos* a la misma $k$-esfera.

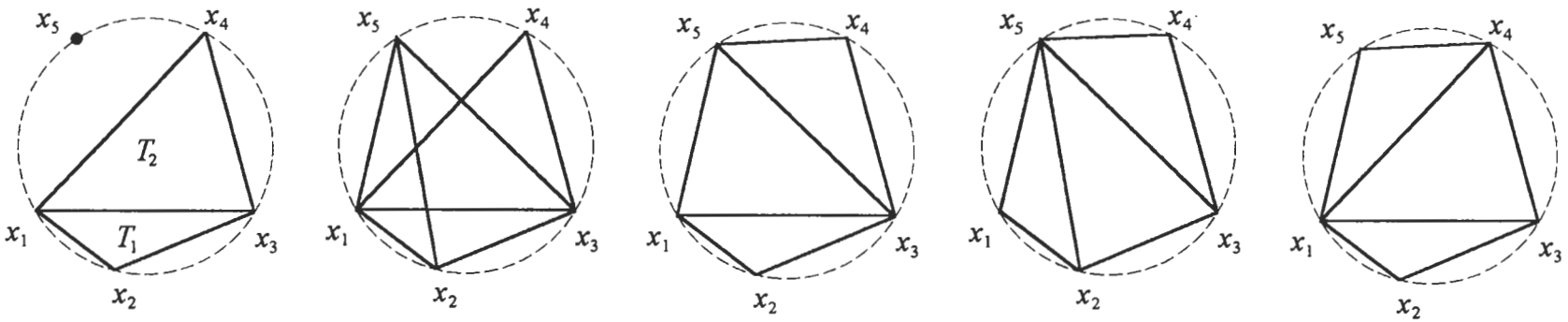



(a)




(b)

# Problemas Debidos a Errores de Redondeo (I)




(a) (b) (c) (d) (e)

- Adición de $x_5$ a la triangulación de $X_4 = \{x_1, \cdots, x_4\}$, figura (a).
- Las triangulaciones posibles, en función de la posición de $x_5$ calculada por el ordenador, son:

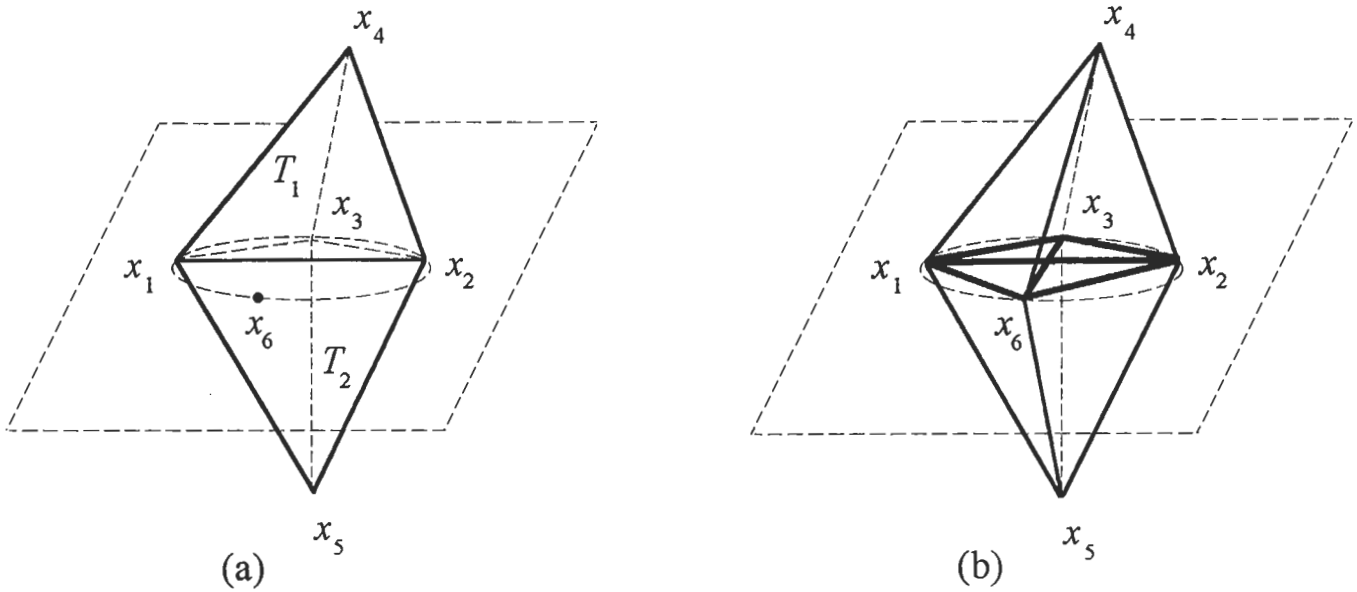
  (b) Si $x_5 \in B(T_1)$ y $x_5 \notin B(T_2)$, el algoritmo elimina $T_1$ y conserva $T_2$; triangulación no admisible.

  (c) Si $x_5 \notin B(T_1)$ y $x_5 \in B(T_2)$, el algoritmo conserva $T_1$ y elimina $T_2$; triangulación admisible.

  (d) Si $x_5 \in B(T_1)$ y $x_5 \in B(T_2)$, el algoritmo elimina $T_1$ y $T_2$; triangulación admisible.

  (e) Si $x_5 \notin B(T_1)$ y $x_5 \notin B(T_2)$, el algoritmo conserva $T_1$ y $T_2$; triangulación admisible.

# Problemas Debidos a Errores de Redondeo (II)




(a) Adición del punto $x_6$, coplanario con $x_1$ $x_2$ y $x_3$, a la triangulación formada por los tetraedros $T_1$ y $T_2$.

(b) Si la posición de $x_6$, calculada por el ordenador, es tal que $x_6 \in B(T_1)$ y $x_6 \notin B(T_2)$, se forma el *tetraedro* con volumen nulo, $\{x_1, x_2, x_3, x_6\}$. Esta situación puede ser peor aun si el punto $x_6$ estuviera por debajo del plano definido por $x_1$ $x_2$ y $x_3$ y el ordenador siguiera evaluando que $x_6 \in B(T_1)$ y $x_6 \notin B(T_2)$. En ese caso los puntos $\{x_1, x_2, x_3, x_6\}$ no formarían un tetraedro.

# Alternativas para solucionar las inconsistencias topológicas en la triangulación de Delaunay

- **Aritmética no exacta**

  Elaborar algún algoritmo capaz de crear una triangulación consistente, aunque ésta no sea exactamente de Delaunay.

  $\Rightarrow$ Problemas: Puede llegar a fallar cuando los puntos están muy proximos entre si.

- **Aritmética exacta**

1. Aplicar un factor de escala a los datos de la entrada de manera que los números en coma flotante se conviertan en enteros.
2. Efectuar los cálculos de los determinantes $\Delta(x_{i+1}, T_j)$ y $\det(T_j)$ (que establecen si el punto $x_{i+1}$ está o no dentro de la esfera circunscrita a $T_j$ y la orientación de este tetraedro, respectivamente) utilizando números enteros, exclusivamente.

   $\Rightarrow$ Problemas: Limitación en las cantidades que aparecen en los determinantes, y por tanto, en la longitud del dominio. Aumento en el tiempo de CPU en los cálculos. Persistencia de tetraedros *planos*.

# Construcción de $D(X_{i+1})$. Consideraciones Preliminares

- El conjunto $D_1^i$ debe satisfacer unas determinadas condiciones para que $D(X_{i+1})$ sea admisible.
- Definiciones:
  1. Diremos que la cara $C \in C^i_{f\,D_1}$ es $\varepsilon$-*visible* desde $x_{i+1}$ si y sólo si es visible desde ese punto, y además:

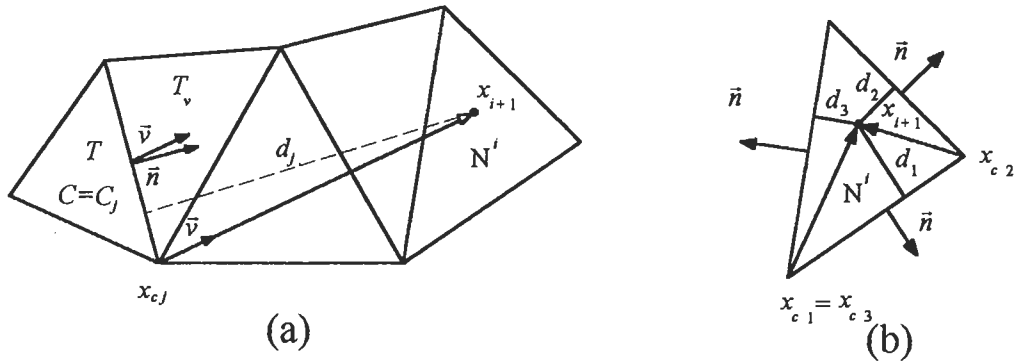
$$\min_{x \in C}\{\cos(\overline{x_{i+1}x}, \vec{n})\} > \varepsilon$$

     donde $\vec{n}$ es el vector unitario normal a $C$ y saliente de $D_1^i$ y $\varepsilon$ un determinado número positivo.

  2. Diremos que un politopo es $\varepsilon$-*en forma de estrella* desde $x_{i+1}$ si todas sus caras son $\varepsilon$-visibles desde ese punto.
- El conjunto $D_1^i$ debe satisfacer los siguientes requisitos:
  1. Un tetraedro $T$ pertenece a $D_1^i$ sólo si $x_{i+1} \in B(T)$.
  2. El conjunto $D_1^i$ debe ser $\varepsilon$-*en forma de estrella* desde $x_{i+1}$.
  3. El conjunto $D_1^i$ no debe contener ningún otro punto de $X$ aparte de $x_{i+1}$.

# Construcción de D($X_{i+1}$). Búsqueda del Núcleo (I)

- Definimos *núcleo*, $N^i$, como el tetraedro o conjunto de tetraedros que contienen a $x_{i+1}$.




- **Algoritmo para encontrar alguno de los tetraedros del núcleo.**

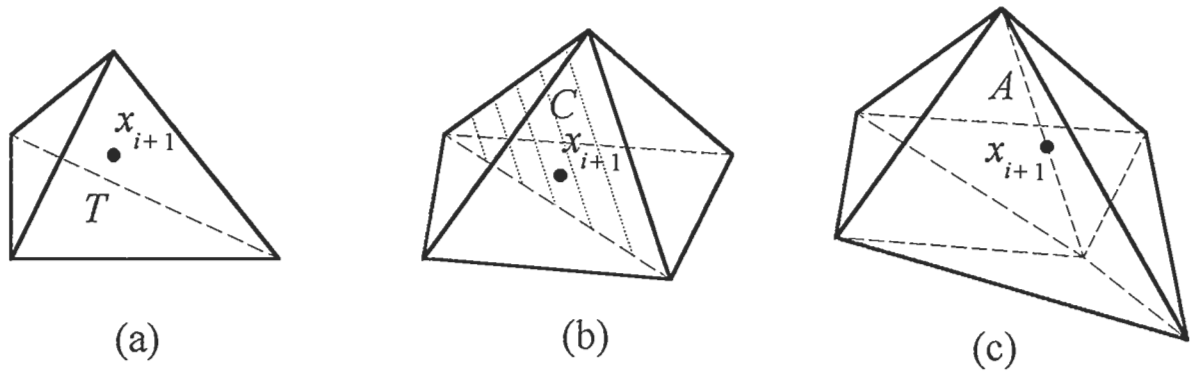
  Sea $T_u$ el último de los tetraedros construido en la triangulación.

  1. $T := T_u$

  2. Si ($\vec{n}\cdot\vec{v} \le 0 \quad \forall C \in T$) entonces $T \in N^i$; fin (figura (b)).
     en caso contrario, encontramos la cara $C$ de $T$ que maximiza $\vec{n}\cdot\vec{v}$ (figura (a)).

  3. Sea $T_v$ el tetraedro que comparte $C$ con $T$. Hacemos $T := T_v$ y retornamos al paso 2.

# Construcción de D($X_{i+1}$). Búsqueda del Núcleo (II)




(a) (b) (c)

- Determinación del núcleo:

  (a) Si $x_{i+1} \in \dot{T}$, el núcleo es $T$.

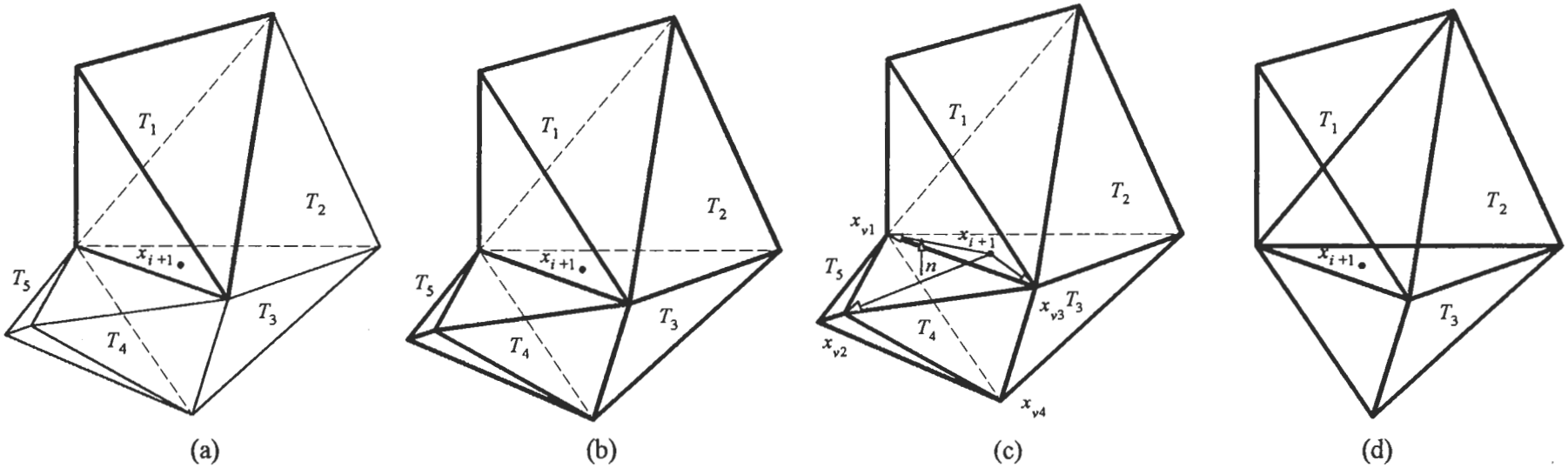
  (b) Si $x_{i+1} \in \dot{C}$, el núcleo es el par de tetraedros que comparten $C$.

  (c) Si $x_{i+1} \in \dot{A}$, el núcleo es el conjunto de tetraedros que comparten $A$.

- El núcleo es $\varepsilon$-en forma de estrella respecto del punto $x_{i+1}$.

- Encontramos el parámetro $\varepsilon_N$ para el que el núcleo es $\varepsilon$-en forma de estrella.

# Construcción de $\mathrm{D}(X_{i+1})$. Determinación de $\mathrm{D}_1^i$




(a) Núcleo, $\mathrm{N}^i = \{T_1\}$, y conjunto de tetraedros, $\{T_1, T_2, T_3, T_4, T_5\}$, cuyas esferas contienen a $x_{i+1}$.

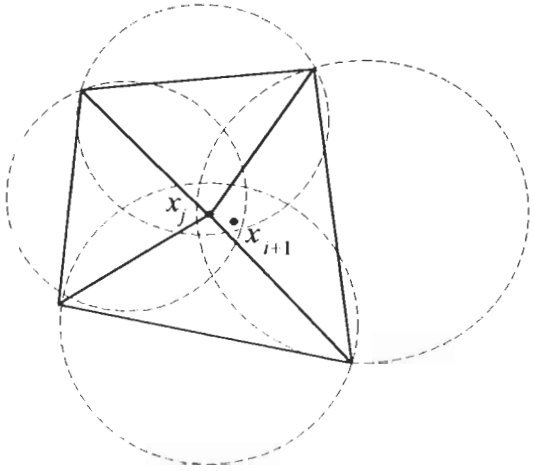
(b) Ampliación del núcleo hasta llegar a $\mathrm{D}_1^i = \{T_1, T_2, T_3, T_4, T_5\}$.

(c) El conjunto $\mathrm{D}_1^i$ no es $\varepsilon$-en forma de estrella desde $x_{i+1}$; por ejemplo, $\cos(\overline{x_{i+1}x_{v2}}, \vec{n}) < 0$.

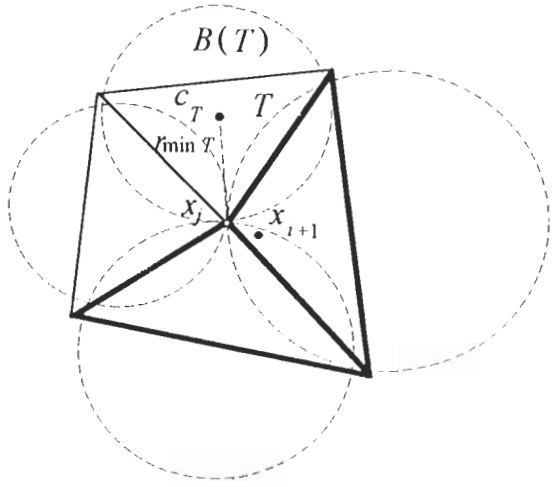
(d) El algoritmo comienza de nuevo la determinación de $\mathrm{D}_1^i$. Ahora, el tetraedro $T_4$ se considera como si su esfera circunscrita, $B(T_4)$, no contuviera a $x_{i+1}$, y por tanto, el algoritmo se detiene una vez se haya añadido $T_3$ al conjunto $\mathrm{D}_1^i$.

# Construcción de $D(X_{i+1})$. $D_1^i$ sólo debe contener a $x_{i+1}$

- El algoritmo debe evitar que ningún punto de $X$, aparte de $x_{i+1}$, esté contenido en $D_1^i$.




(a)




(b)

- Los errores de redondeo en el cálculo de las esferas circunscritas pueden hacer que éstas contengan algún punto de $X$ aparte de $x_{i+1}$ (a).

- Para cada tetraedro se calcula el *radio mínimo*, $r_{\min T} = \min\{d(c_T, V_T)\}$, de $B(T)$, donde $V_T$ es cualquiera de los vértices de $T$ y $c_T$ el centro de $B(T)$ (b).

- La determinación de $D_1^i$ se lleva a cabo con estos valores de los radios.

# Algoritmo para la determinación de $D_1^i$

***Se expande el núcleo con ciertos controles hasta que se obtiene*** $D_1^i$.

Consideremos $B^i = \{B(T) / T \in D(X_i)\}$ y $R_{B(T)}$ el radio de $B(T)$.

**procedimiento** CONJUNTO $D_1^i$

entrada: $D(X_i), B^i, N^i, x_{i+1}, \varepsilon, \varepsilon_N$; salida: $D_1^i$

$D_1^i := N^i$

hallamos $ND_1^i = \{T$ adyacentes a $N^i$ / $x_{i+1} \in B(T)\}$.

**mientras** $(ND_1^i \neq \varnothing)$ **hace**

$D_1^i := D_1^i \cup ND_1^i$ (* *se actualiza el conjunto* $D_1^i$ *)

hallamos $ND_1^i = \{T$ adyacentes a $D_1^i$ / $x_{i+1} \in B(T)\}$.

**fin mientras**

**si** ($D_1^i$ no es $\varepsilon$-en-forma-de-estrella respecto de $x_{i+1}$) **entonces**

hallamos $C_m = \{C \in$ frontera de $D_1^i$ / $C$ no es $\varepsilon$-visible$\}$.

hallamos $T_m = \{T \in D_1^i$ / alguna cara de $T \in C_m\}$.

**si** ($\exists\ T \in T_m \cap N^i$) **entonces** (* *el valor actual de* $\varepsilon$ *es demasiado restrictivo. Se substituye por* $\varepsilon_N$, *que es menor**)

$\varepsilon := \varepsilon_N$

CONJUNTO $D_1^i$ entrada: $D(X_i), B^i, N^i, x_{i+1}, \varepsilon, \varepsilon_N$; salida: $D_1^i$

(* *el algoritmo comienza de nuevo con un* $\varepsilon$ *menor**)

**si no**

$R_{B(T)} := 0, \forall\ T \in T_m$ (* *esto evita que el tetraedro* $T$ *sea incluido de nuevo en* $D_1^i$ *)

CONJUNTO $D_1^i$ entrada: $D(X_i), B^i, N^i, x_{i+1}, \varepsilon, \varepsilon_N$; salida: $D_1^i$

(* *el algoritmo comienza de nuevo considerando que las "esferas circunscritas" a* $T_m$ *tienen radios nulos* *)

**fin si**

**fin si**

se restauran los valores iniciales de aquellos radios que hayan sufrido variación durante el proceso.

**fin procedimiento.**

# Construcción de $D_2^i$

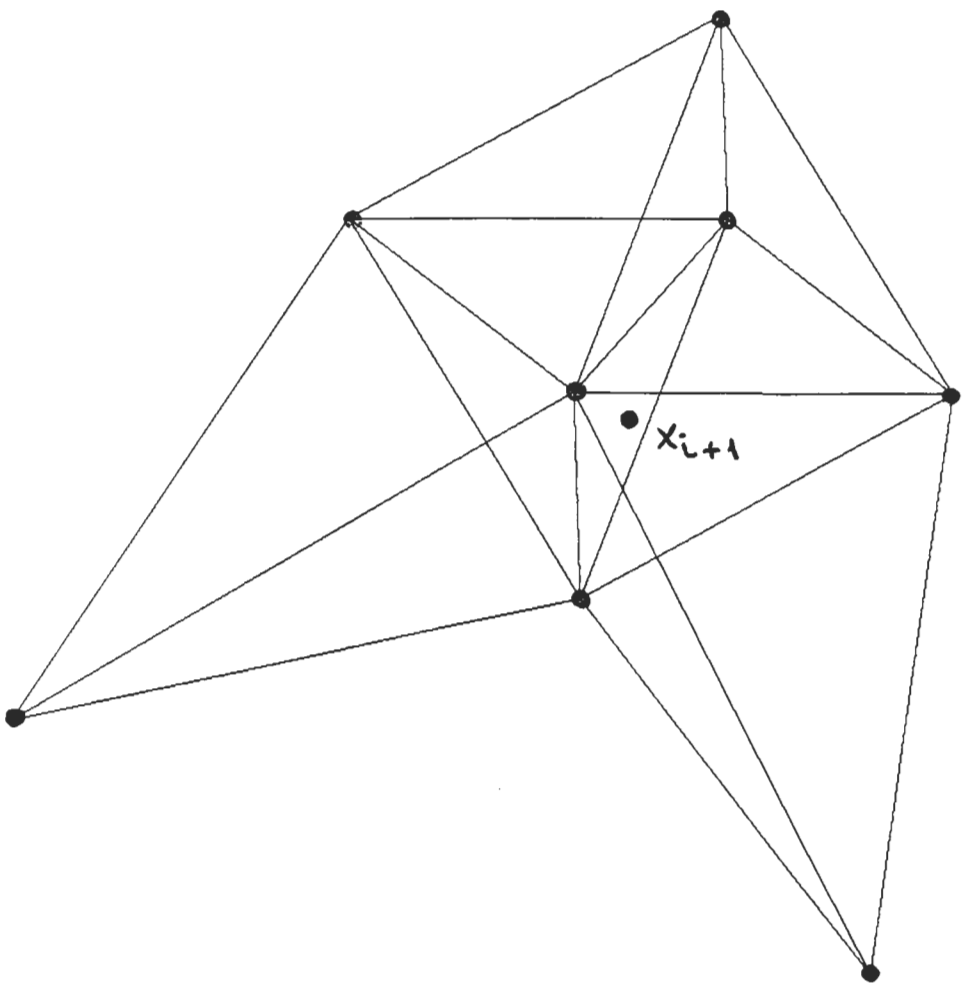
Una vez determinados:

- $D_1^i, C_1^i, A_1^i \in D(X_i)$: conjuntos de tetraedros, caras y aristas que desaparecen al añadir $x_{i+1}$.

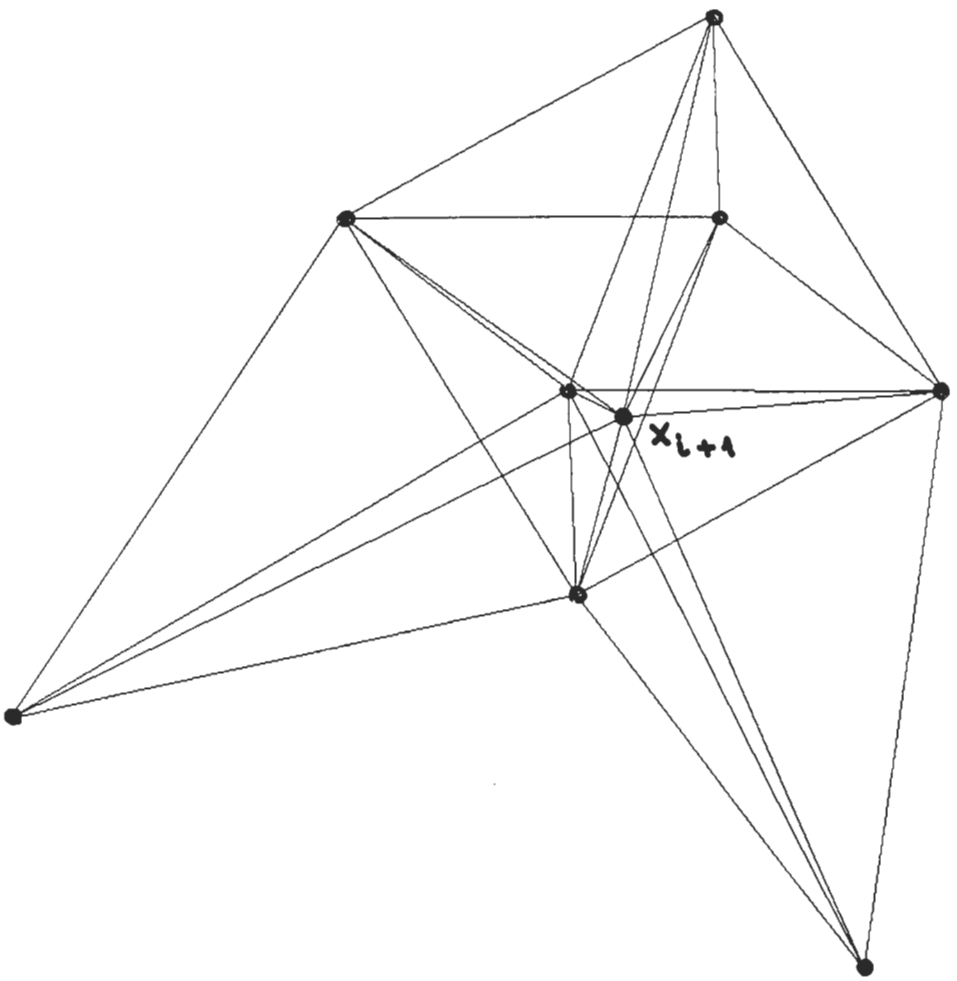
Construimos:

- $D_2^i, C_2^i, A_2^i \in D(X_{i+1})$: conjuntos de tetraedros, caras y aristas formadas a partir de $x_{i+1}$.
  - El conjunto $D_2^i$ se forma con el punto $x_{i+1}$ y las caras de la frontera de $D_1^i$.
  - El conjunto $C_2^i$ se forma con el punto $x_{i+1}$ y las aristas de la frontera de $D_1^i$.
  - El conjunto $A_2^i$ se forma con el punto $x_{i+1}$ y los vértices de la frontera de $D_1^i$.
- Se restauran las conexiones entre estos nuevos elementos.
- La triangulación de $D(X_{i+1})$ concluye.

**Conjunto de tetraedros T1i** $\equiv D_1^i$



**Conjunto de tetraedros T2i** $\equiv D_2^i$



# Estructura de datos del programa (I)

Cada punto, arista, cara y tetraedro están representados por un nodo, *nv*, *na*, *nc* y *nt*, respectivamente.

Coordenadas de los puntos de $X$: $x(nv), y(nv), z(nv)$.

Coordenadas del centro y radio de la esfera circunscrita a cada tetraedro: $x_e(nt), y_e(nt), z_e(nt), r_e(nt)$.

**Descripción de los elementos:**

1. Descripción del tetraedro por sus caras: $CT(i,nt) = nc \quad (1 \le i \le 4)$.
   El vector $CT$ proporciona las cuatro caras del tetraedro *nt*.
2. Descripción del tetraedro por sus vértices: $VT(i,nt) = nv \quad (1 \le i \le 4)$.
   El vector $VT$ proporciona los cuatro vértices del tetraedro *nt*.
3. Descripción de la cara por sus aristas: $AC(i,nc) = na \quad (1 \le i \le 3)$.
   El vector $AC$ proporciona las tres aristas de la cara *nc*.
4. Descripción de la cara por sus vértices: $VC(i,nc) = nv \quad (1 \le i \le 3)$.
   El vector $VC$ proporciona los tres vértices de la cara *nc*.
5. Descripción de la arista por sus vértices: $VA(i,na) = nv \quad (1 \le i \le 2)$.
   El vector $VA$ proporciona los dos vértices de la arista *nc*

$\Rightarrow$ Los vértices de estos elementos son los puntos de la triangulación, es decir, los puntos de $X$.

# Estructura de datos del programa (II)

**Conexiones entre elementos:**

1. Tetraedros que comparten una cara: $TCC(i,nc) = nt \quad (1 \le i \le 2)$.
   El vector *TCC* nos proporciona los tetraedros, *nt*, que comparten la cara *nc*. ($TCC(i,nc) = 0$, para $i$=1 ó 2, si la cara *nc* está sobre la envolvente exterior del dominio)

2. Caras que comparten una arista: $CCA(i,na) = nc \quad (1 \le i \le tcca(na))$.
   El vector *CCA* nos proporciona las caras, *nc*, que comparten la arista *na*. El vector *tcca* indica el total de caras adyacentes a la arista *na*.

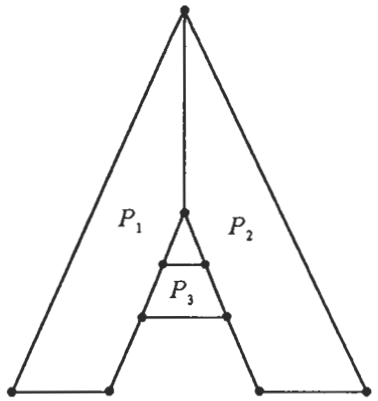
3. Aristas que comparten un vértice: $ACV(i,nv) = na \quad (1 \le i \le tacv(nv))$.
   El vector *ACV* nos proporciona las aristas, *na*, que comparten el vértice *nv*. El vector *tacv* indica el total de aristas adyacentes al vértice *nv*.

⇒ Ventajas: Es una estructura de datos es muy versátil. Por ejemplo, permite la aplicación del M.E.F con elementos basados en las aristas (edge elements), de gran utilidad en electromagnetismo. Facilita enormemente el ensamblaje de las matrices elementales en forma compacta.

⇒ Inconvenientes: Gran gasto de memoria, especialmente si el lenguaje de programación no permite el dimensionamiento dinámico.

# Definición del Dominio y Generación Automática de Puntos (I)

- La triangulación, T(Ω), del dominio queda determinada cuando se generan puntos sobre la superficie e interior de Ω con unas densidades apropiadas.
- La triangulación del dominio se obtiene sustrayendo de la triangulación total, D($X$), los tetraedros que no pertenecen a T(Ω).

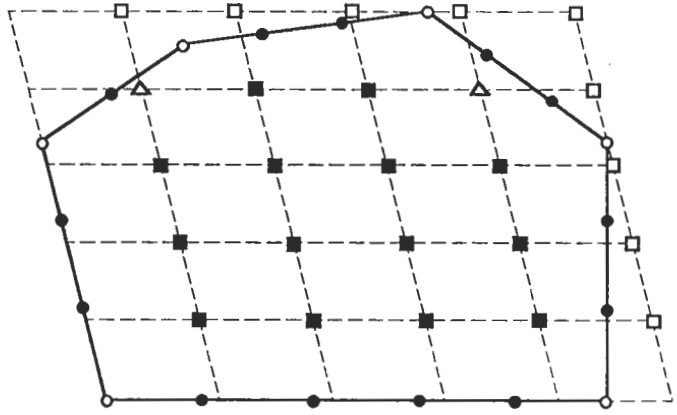



- Para simplificar la generación de puntos, el dominio Ω se descompone en poliedros convexos.
- Los poliedros convexos se definen por sus caras, las caras (convexas) por sus aristas y las aristas por sus vértices.
- El programa genera puntos automáticamente en el interior relativo de cada arista, cara y poliedro.
- La distancia entre puntos se especifica previamente en la entrada de datos.

# GENERACIÓN DE PUNTOS SOBRE LÍNEAS, SUPERFICIES Y REGIONES

- **LÍNEAS.**

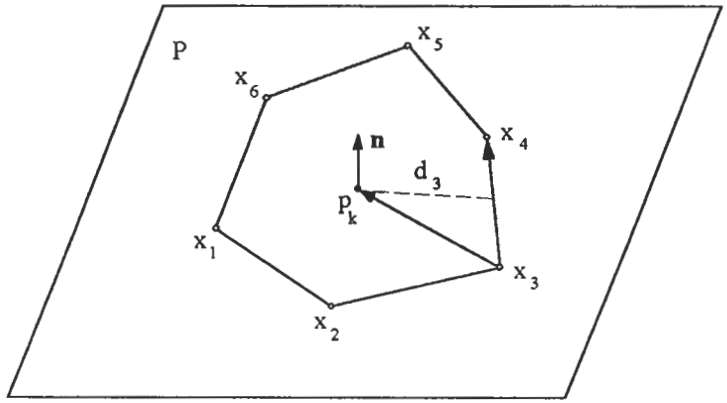
- **SUPERFICIES PLANAS POLIGONALES CONVEXAS.**



Origen del sistema
de referencia

- **REGIONES POLIÉDRICAS CONVEXAS:** De forma análoga a las superficies.

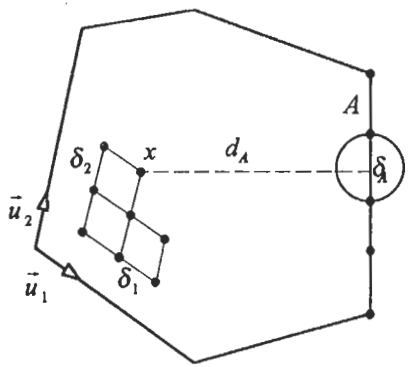
# PUNTOS INTERIORES A SUPERFICIES O REGIONES. CRITERIO

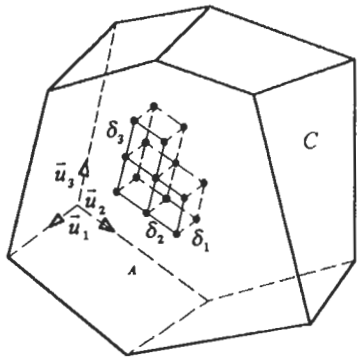



- Sea $\{x_1, x_2, \ldots, x_n\}$ el conjunto de vértices de una poligonal convexa, ordenados cíclicamente.
- Sea $\mathbf{n}$ el vector unitario normal a la superficie ($\mathbf{n} = \mathbf{x_1x_2} \times \mathbf{x_2x_3} \,/\, \| \mathbf{x_1x_2} \times \mathbf{x_2x_3} \|$).
- El punto $p_k$ está dentro de la poligonal $\Leftrightarrow \forall\ i=1,2,\ldots,n$ resulta $d_i = [(\mathbf{n} \times \mathbf{x_ix_{i+1}}) \cdot \mathbf{x_ip_k} \,/\, \| \mathbf{x_ix_{i+1}} \|] > 0$ considerando $x_{n+1} = x_1$; la cantidad $d_i$ representa la distancia a cada lado de la poligonal.

☞ Una idea similar se utiliza para generar puntos en el interior de las regiones poliédricas convexas en que se descompone el objeto.

# Definición del Dominio y Generación Automática de Puntos (II)




- **Generación de puntos en las aristas.**
  La distancia $\delta_A$ entre puntos interiores de una arista $A$ se especifica en la entrada de datos.

- **Generación de puntos en las caras.**
  Los puntos interiores se generan en las direcciones de $\vec{u}_1$ y $\vec{u}_2$ con unas determinadas distancias entre ellos, $\delta_1$ y $\delta_2$, respectivamente.
  Si $d_A > 1/2\,\delta_A$, la arista $A$ no es eliminada por efecto de los puntos interiores de la cara.

- **Generación de puntos interiores en los poliedros.**
  Los puntos interiores se generan en las direcciones de $\vec{u}_1$, $\vec{u}_2$ y $\vec{u}_3$ con unas determinadas distancias entre ellos. Si la distancia de los puntos interiores a la cara $C$ es mayor que $d_C = \frac{1}{2}\sqrt{\delta_1^2 + \delta_2^2} + \frac{1}{2}\max\{\delta_{A_i}\}$, donde $\delta_{A_i}$ es la distancia entre puntos en cada una de las aristas de $C$, entonces esta cara no es eliminada por efecto de los puntos interiores de poliedros adyacentes.

# CONSERVACIÓN DE LA GEOMETRÍA DEL OBJETO

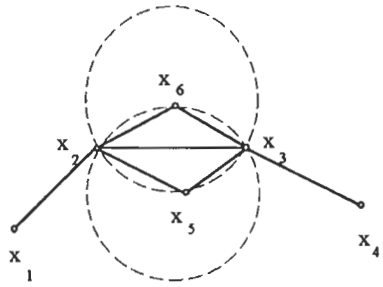


Fig.1

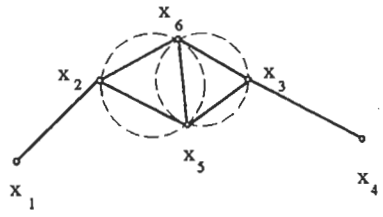


Fig.2

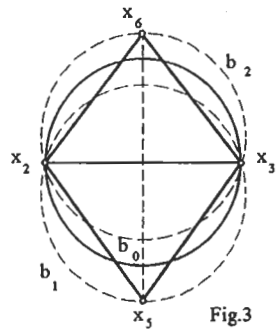


Fig.3

- La triangulación de la fig.1 no es una triangulación de Delaunay ya que no respeta la propiedad de las esferas vacías (según la cual, las esferas circunscritas a los tetraedros no contienen ningún punto de la triangulación). En cambio, la de la fig. 2 sí lo es.

- Supongamos que los segmentos $(x_1, x_2)$, $(x_2, x_3)$ y $(x_3, x_4)$ pertenecen a la geometría del objeto y que se deben insertar los puntos $x_5$ y $x_6$ a ambos lados del segmento $(x_2, x_3)$.

- Cuando se inserte el punto $x_6$ en la triangulación el segmento $(x_2, x_3)$ desaparecerá dando lugar al segmento $(x_5, x_6)$. No se ha respetado la geometría del objeto.

- Una condición suficiente para conservar el segmento $(x_2, x_3)$ es que las distancias de los puntos $x_5$ y $x_6$ a dicho segmento sea mayor que la mitad de la distancia entre $x_2$ y $x_3$.

- Si los puntos $x_5$ y $x_6$ **cumplen** la condición anterior se cumple:
  $x_5$ y $x_6 \not\subset b_0 \Rightarrow x_5 \not\subset b_2$ y $x_6 \not\subset b_1$ y por tanto la triangulación de Delaunay de los puntos $x_2$, $x_3$, $x_5$ y $x_6$ conserva el segmento $(x_2, x_3)$.

# Clasificación de los elementos de la triangulación (I)

OBJETIVO: Una vez realizada la triangulación $\mathrm{D}(X)$, a cada punto, arista, cara y tetraedro pertenecientes al dominio se le debe asignar un número de referencia que permita imponer las condiciones de contorno oportunas o asociarle un determinado tipo de material.

REALIZACIÓN: Supongamos que $V_i$, $C_i$, y $A_i$ son, respectivamente, los subvolúmenes convexos, superficies convexas y segmentos en los que se descompone el dominio. A cada uno de estos elementos se le ha asignado, en la entrada de datos, un número de referencia que posteriormente puede ser interpretado como una condición de contorno, un tipo de material, etc.

**Clasificación de los tetraedros de la triangulación:**

$$\forall t \in \mathrm{D}(X) \begin{cases} t \notin \text{ningún } V_i \Rightarrow t \notin \text{dominio} ; \; rnt(t) := 0 \\ t \in V_i ; \begin{bmatrix} \text{asignamos a } t \text{ el número de referencia de } V_i \\ \text{almacenamos los n}^{\text{o}} \text{ globales de estos tetraedros: } n := n+1;\ rnt(t) := n \end{bmatrix} \end{cases}$$

**Clasificación de las caras de la triangulación:**

$$\forall c \in D(X) \begin{cases} c \notin \textit{ningún } C_i \Rightarrow c \notin \textit{frontera del dominio ni a ninguna interfase entre materiales} ; \; rnc(c) := 0 \\ c \in C_i ; \begin{bmatrix} \textit{asignamos a } c \textit{ el número de referencia de } C_i \\ \textit{almacenamos el n}^{o} \textit{ global de estas caras: } n := n+1;\ rnc(c) := n \end{bmatrix} \end{cases}$$

# Clasificación de los elementos de la triangulación (II)

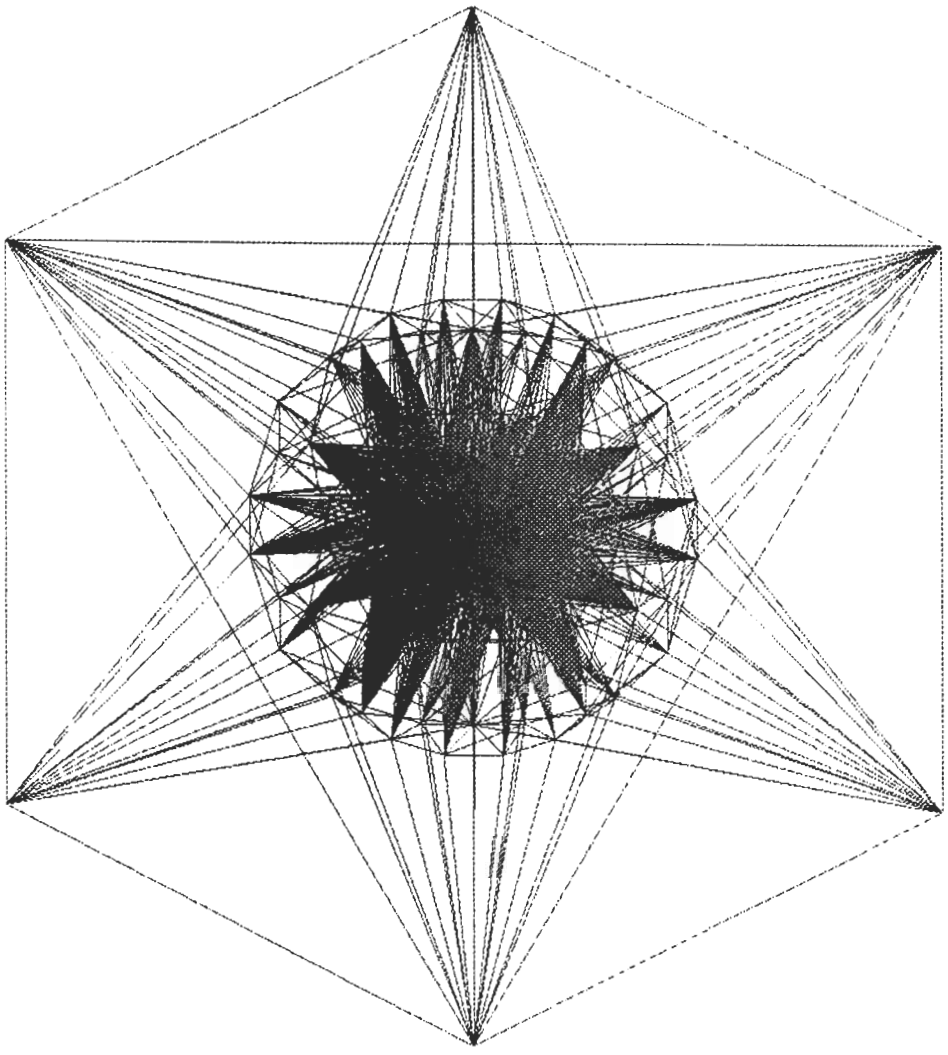
**Clasificación de las aristas de la triangulación:**

$$\forall a \in D(X) \begin{cases} a \notin \text{ningún } A_i,\ C_i,\ V_i \Rightarrow c \notin \text{al dominio} \\ a \in A_i; \begin{bmatrix} \text{asignamos } a \text{ a el número de referencia de } A_i \\ \text{almacenamos el } n^o \text{ global de estas aristas: } n := n+1;\ rna(a) := n \end{bmatrix} \\ a \in C_i; \begin{bmatrix} \text{asignamos } a \text{ a el número de referencia de } C_i \\ \text{almacenamos el } n^o \text{ global de estas aristas: } n := n+1;\ rna(a) := n \end{bmatrix} \\ a \in V_i; \begin{bmatrix} \text{asignamos } a \text{ a el número de referencia de } V_i \\ \text{almacenamos el } n^o \text{ global de estas aristas: } n := n+1;\ rna(a) := n \end{bmatrix} \end{cases}$$

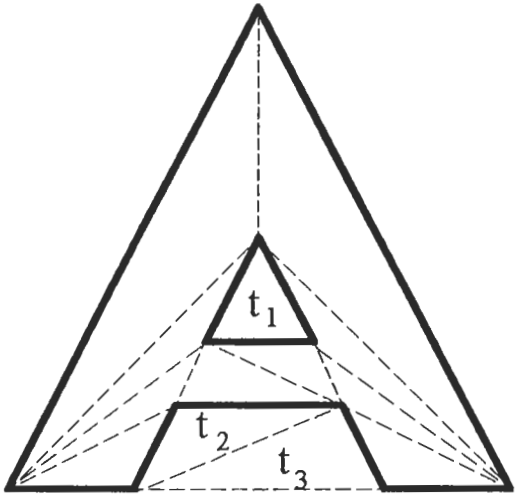
**Clasificación de los puntos de la triangulación:**

$$\forall v \in D(X) \begin{cases} v \in \text{ a alguno de los vértices de la envolvente del dominio} \Rightarrow \mathbf{v} \notin \text{dominio} \\ v \notin \text{ a ninguno de los vértices de la envolvente del dominio} \Rightarrow \mathbf{v} \in \text{dominio} \end{cases}$$

Nota: Para el cálculo de las eventuales integrales de línea que pudieran aparecer en el M.E.F., sólo se necesitaría conocer las aristas pertenecientes a alguno de los segmentos $A_i$. No obstante, cuando se utilizan elementos basados en las aristas (edge elements) éstas hacen el papel de nodos, y por tanto se hace necesaria su total clasificación.



# ELIMINACIÓN DE LOS TETRAEDROS NO PERTENECIENTES AL OBJETO

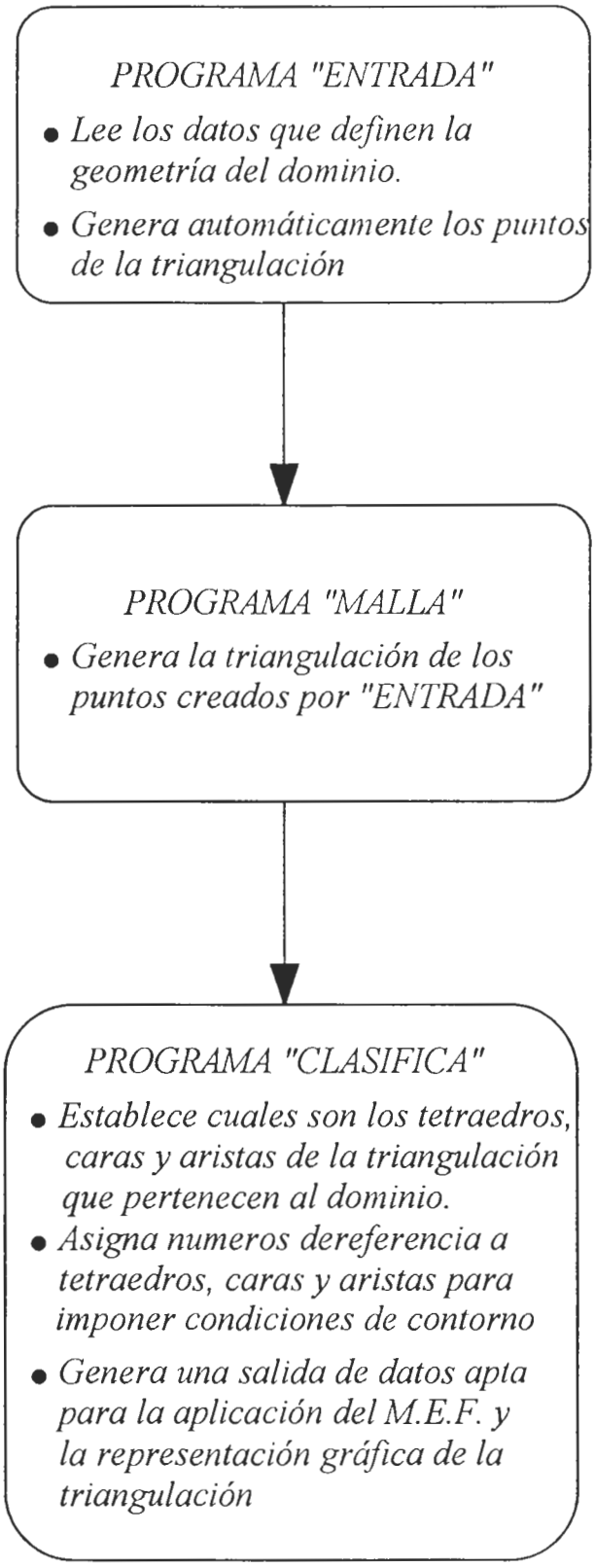


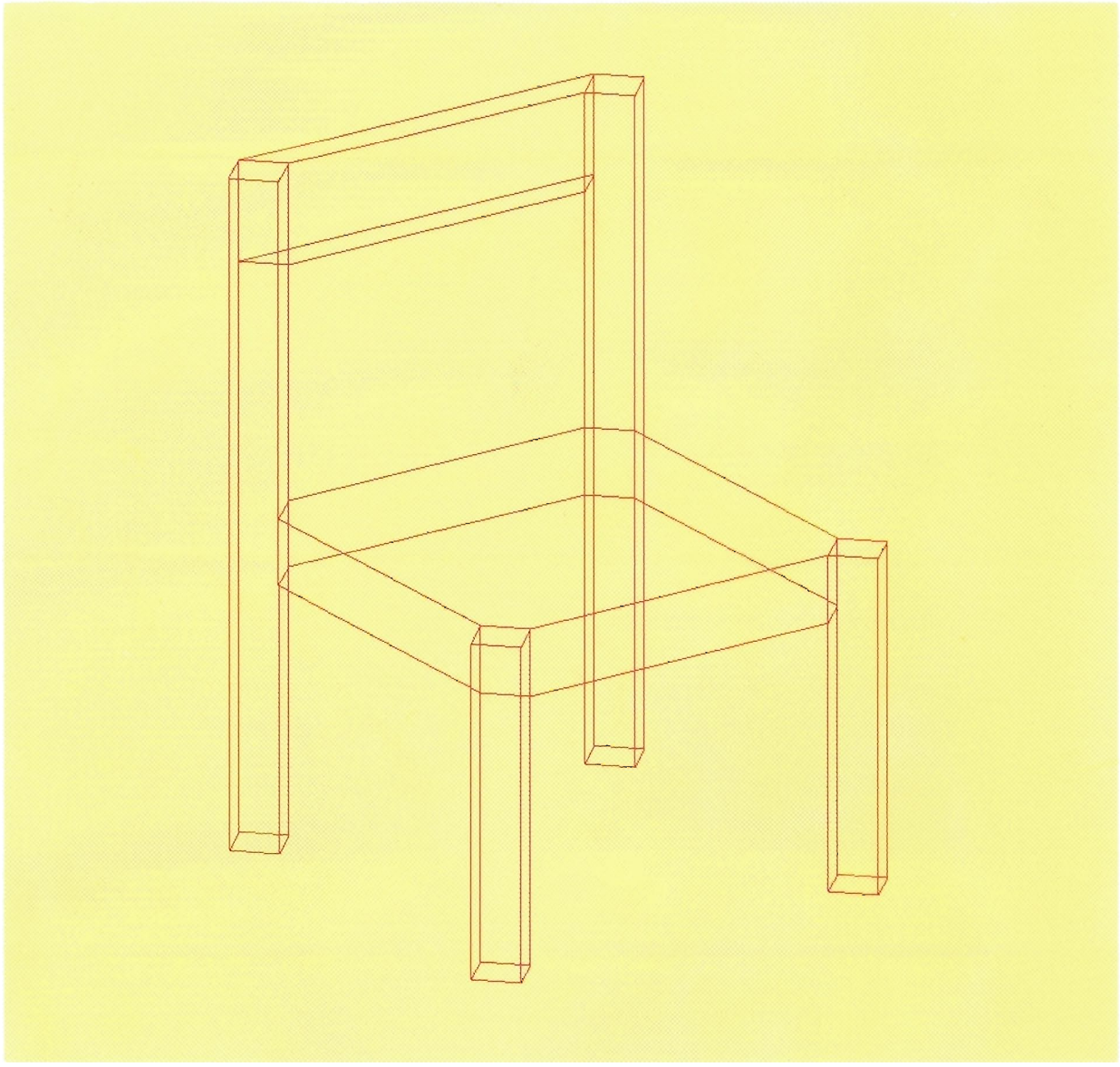

- El algoritmo de Watson crea una triangulación de la envuelta convexa de la nube de puntos. Es por tanto necesario eliminar los tetraedros exteriores al objeto.

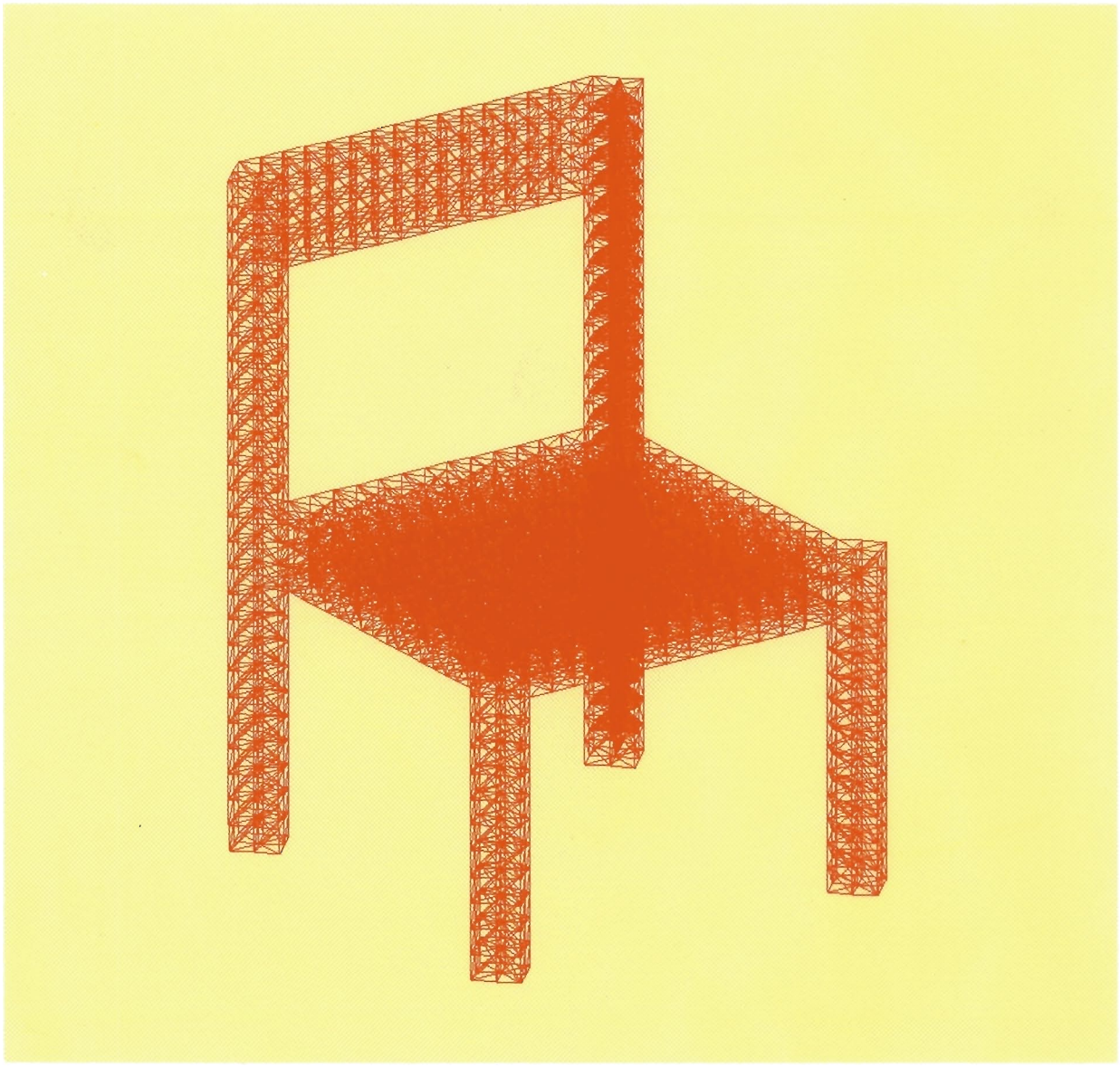
- Para suprimir estos tetraedros hallamos el centro geométrico de cada tetraedro y mediante un procedimiento similar al empleado para situar puntos dentro del objeto, localizamos aquellos que estén fuera del mismo.

- En la figura superior los triángulos $t_1$ , $t_2$ y $t_3$ tienen sus centros fuera del objeto, por tanto serán eliminados de la triangulación final.

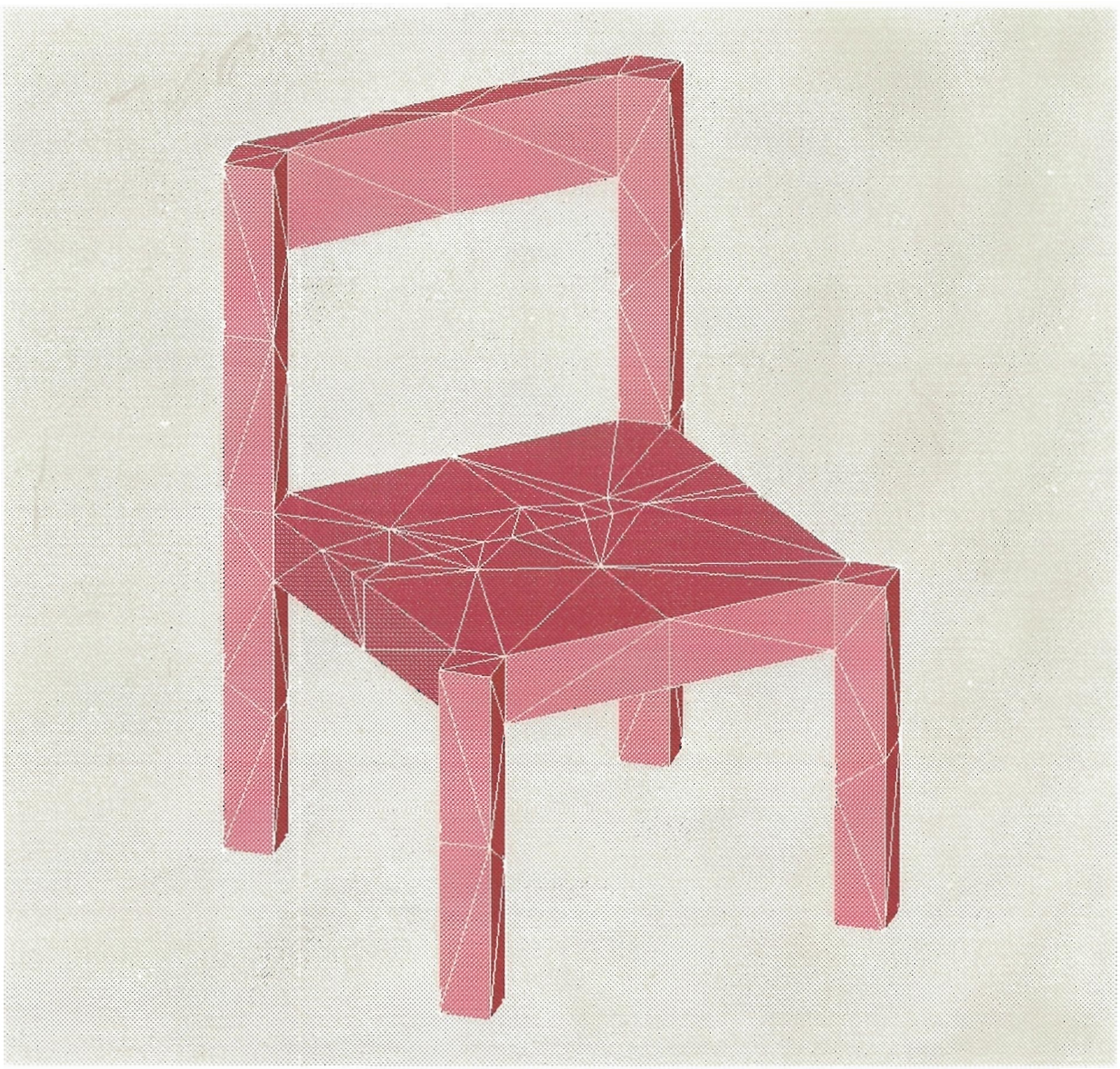
# Programas pricipales

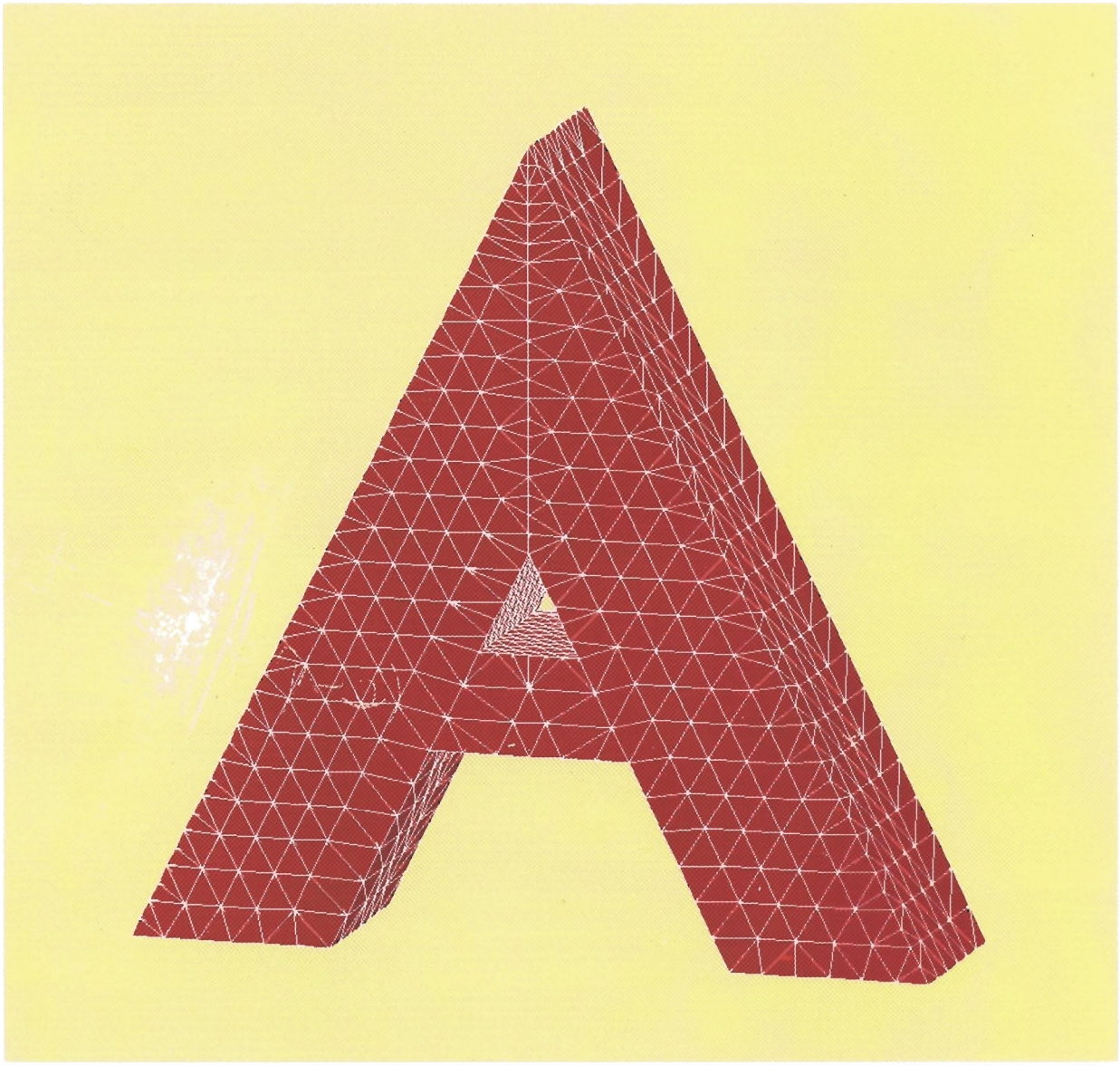


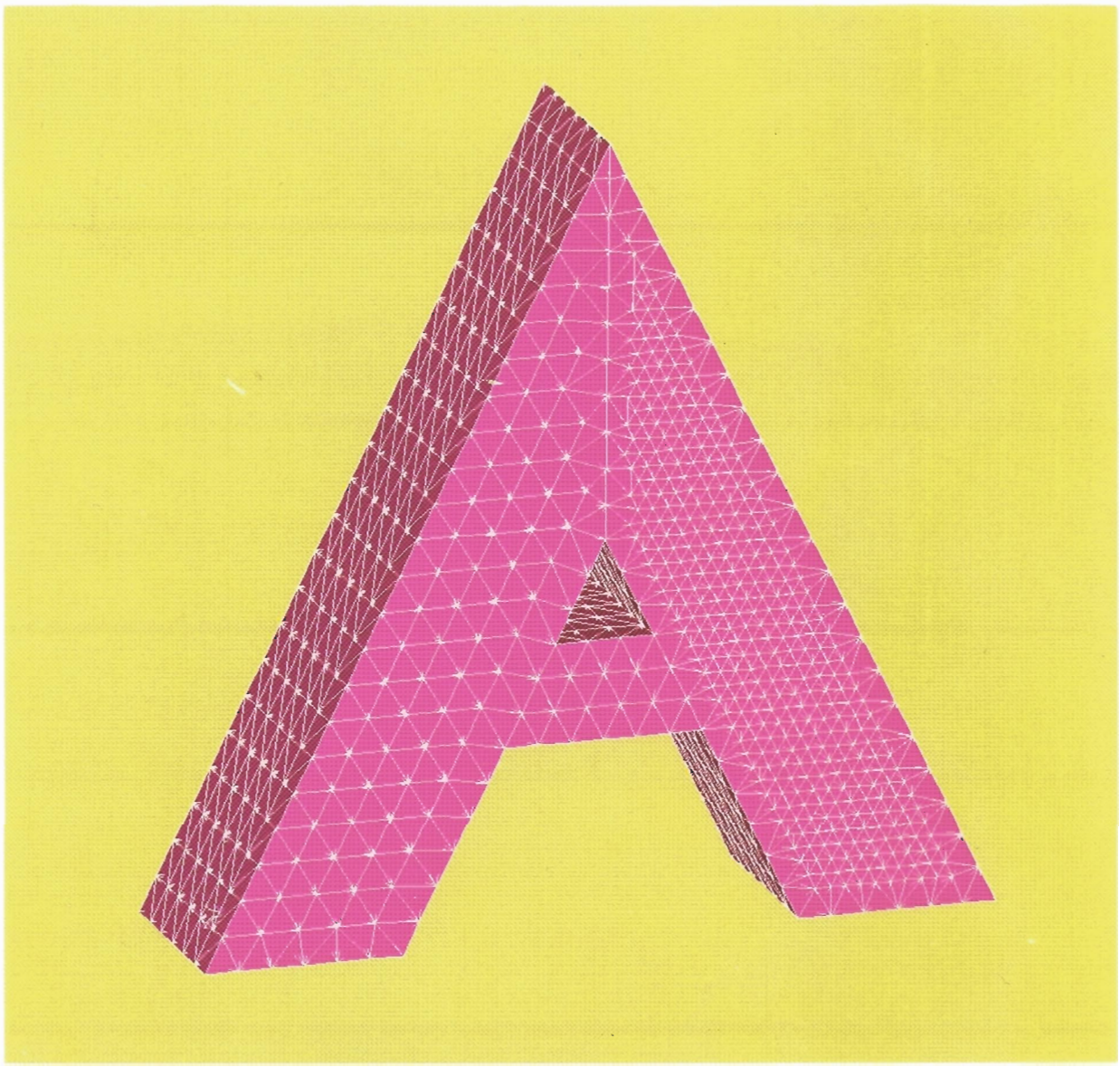




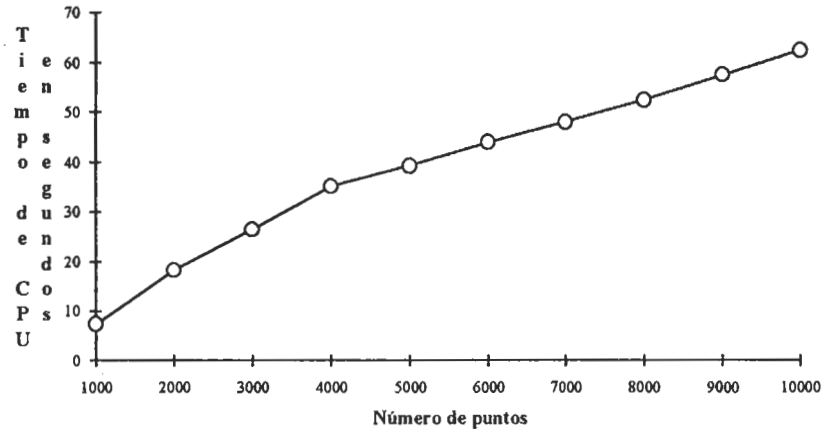
Problema de Triangulación No conforme con la frontera del dominio



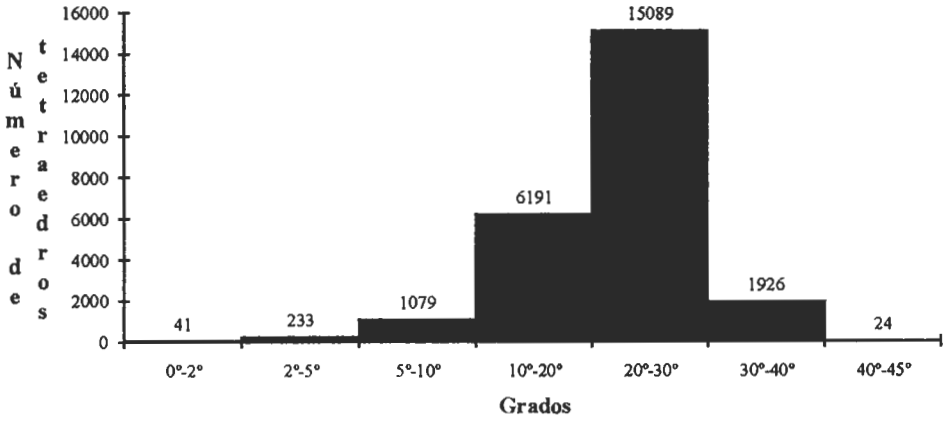




# Tiempo de CPU y Medida de Calidad de la Triangulación




(a)




(b)

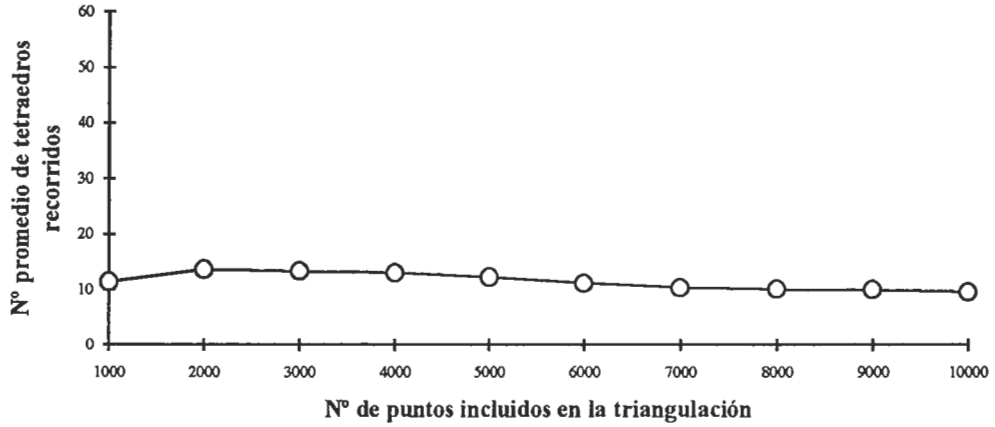
- Figura (a): Tiempo de CPU en segundos en función del número de puntos en una estación de trabajo HP-730. Aproximadamente se obtiene una complejidad lineal en función del número de puntos.
- Figura (b): Número de tetraedros frente al ángulo $\phi_p$ en grados. Se puede dar una medida de la calidad de un tetraedro $T$ mediante la expresión $\min_{1\le p\le 4}\{\phi_p\}$, donde:

$$\phi_p = \text{sen}^{-1}(1-\cos^2\alpha_p-\cos^2\beta_p-\cos^2\gamma_p+2\cos\alpha_p\cos\beta_p\cos\gamma_p)^{1/2}$$

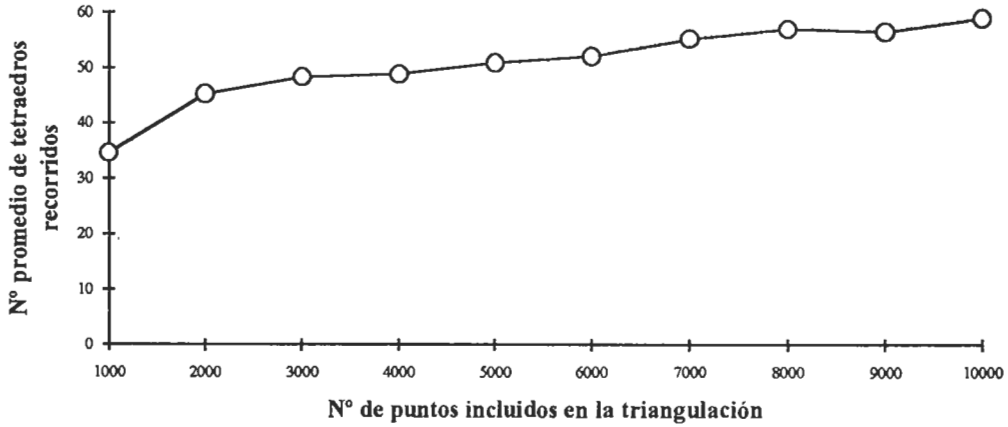
y $\alpha_p$, $\beta_p$, y $\gamma_p$ son los ángulos planos correspondientes al vértice $p$. El valor de $\phi_p$ es 45° para un tetraedro equilátero y 0° para uno plano.

# Características del Algoritmo de Búsqueda del Núcleo

(a)

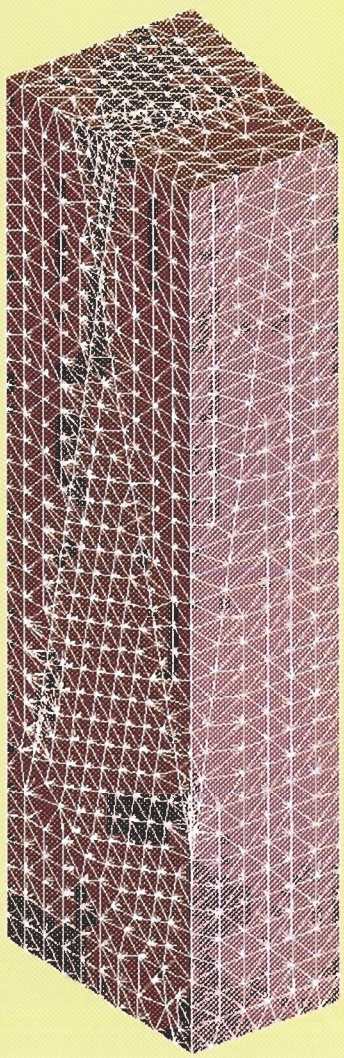


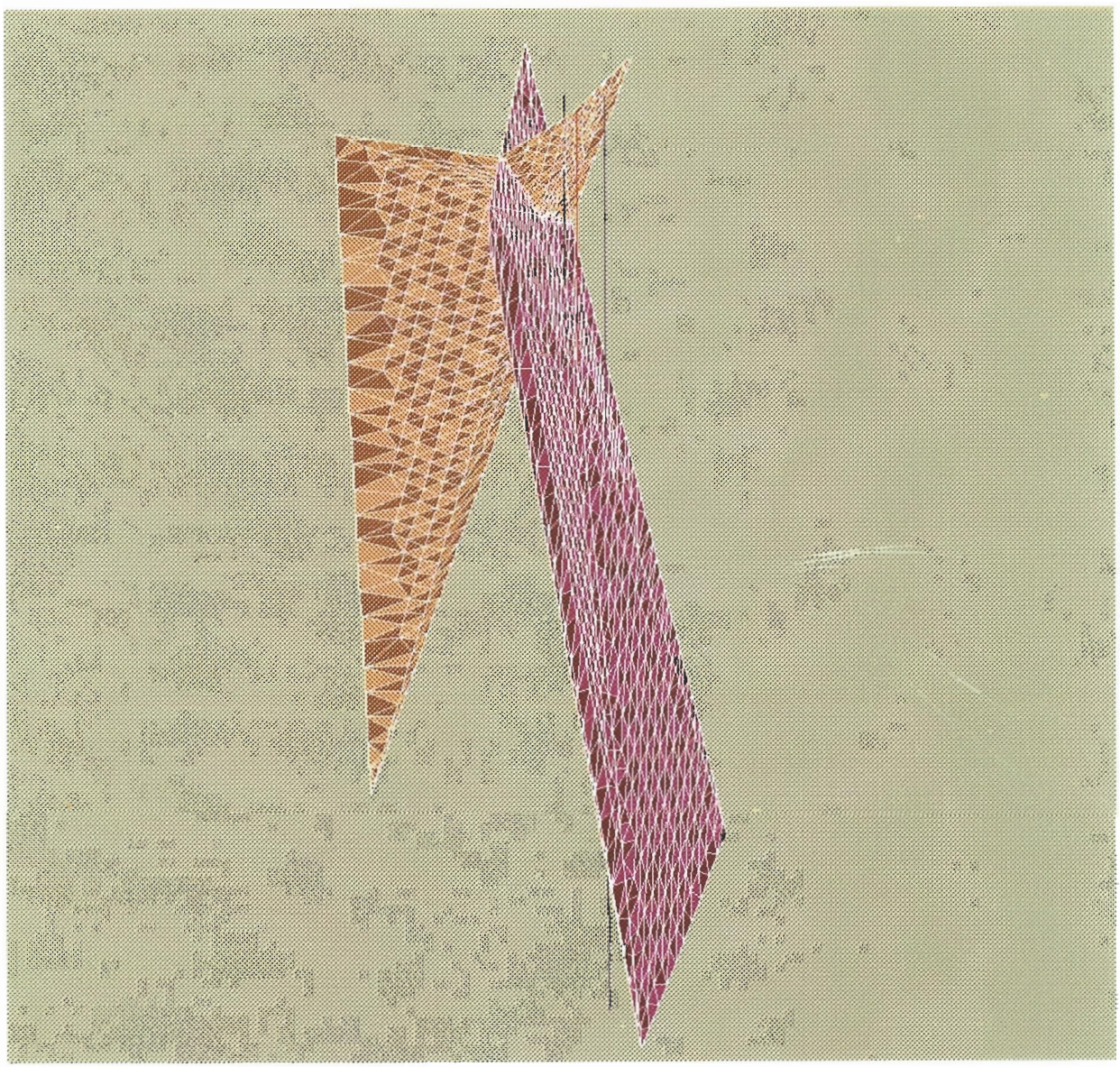

(b)

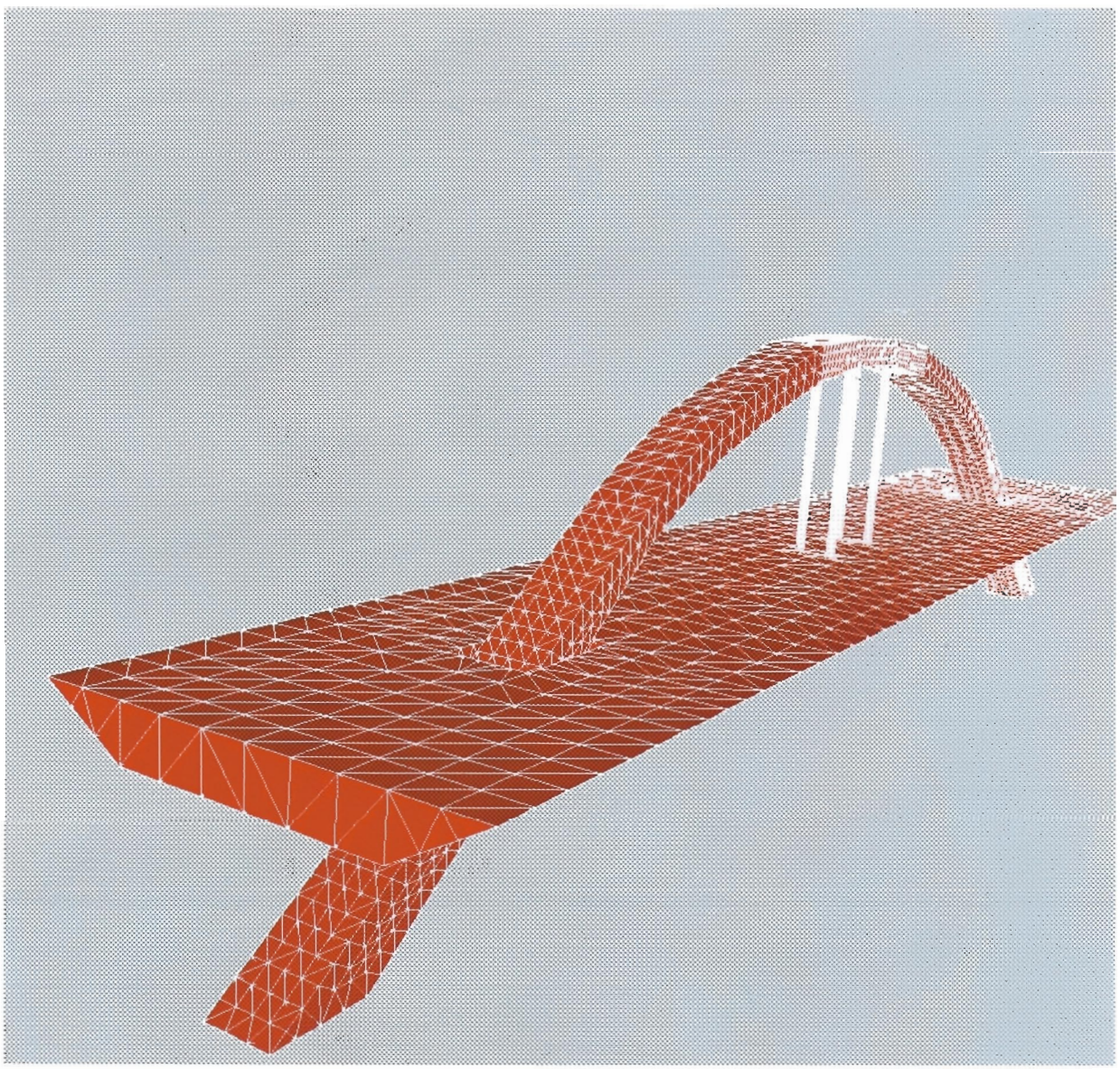


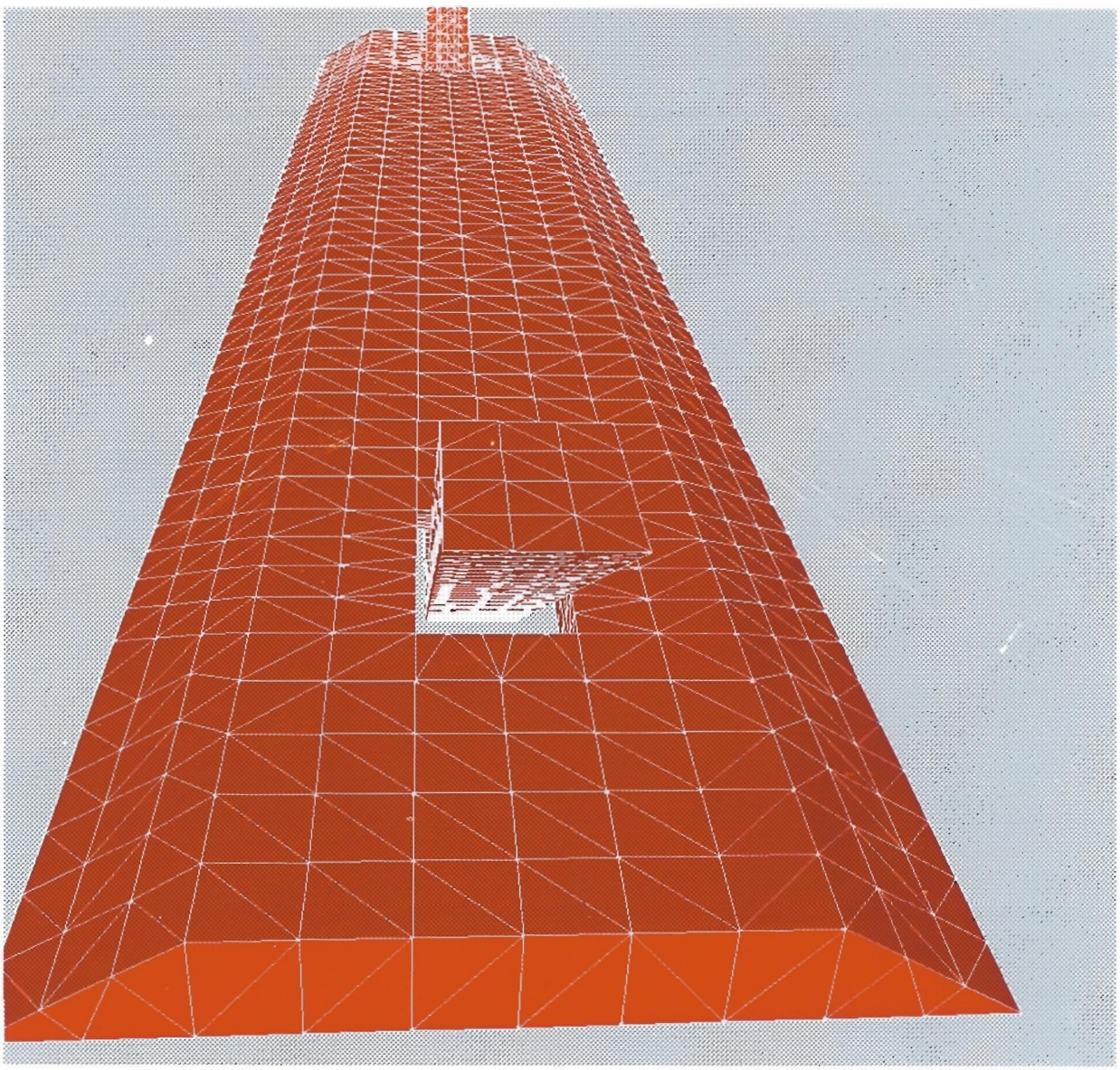

- Nº promedio de tetraedros recorridos para encontrar el núcleo, en función del nº de puntos. Empezando por el último tetraedro incluido en la triangulación (a) y empezando por el nº 1 (b).

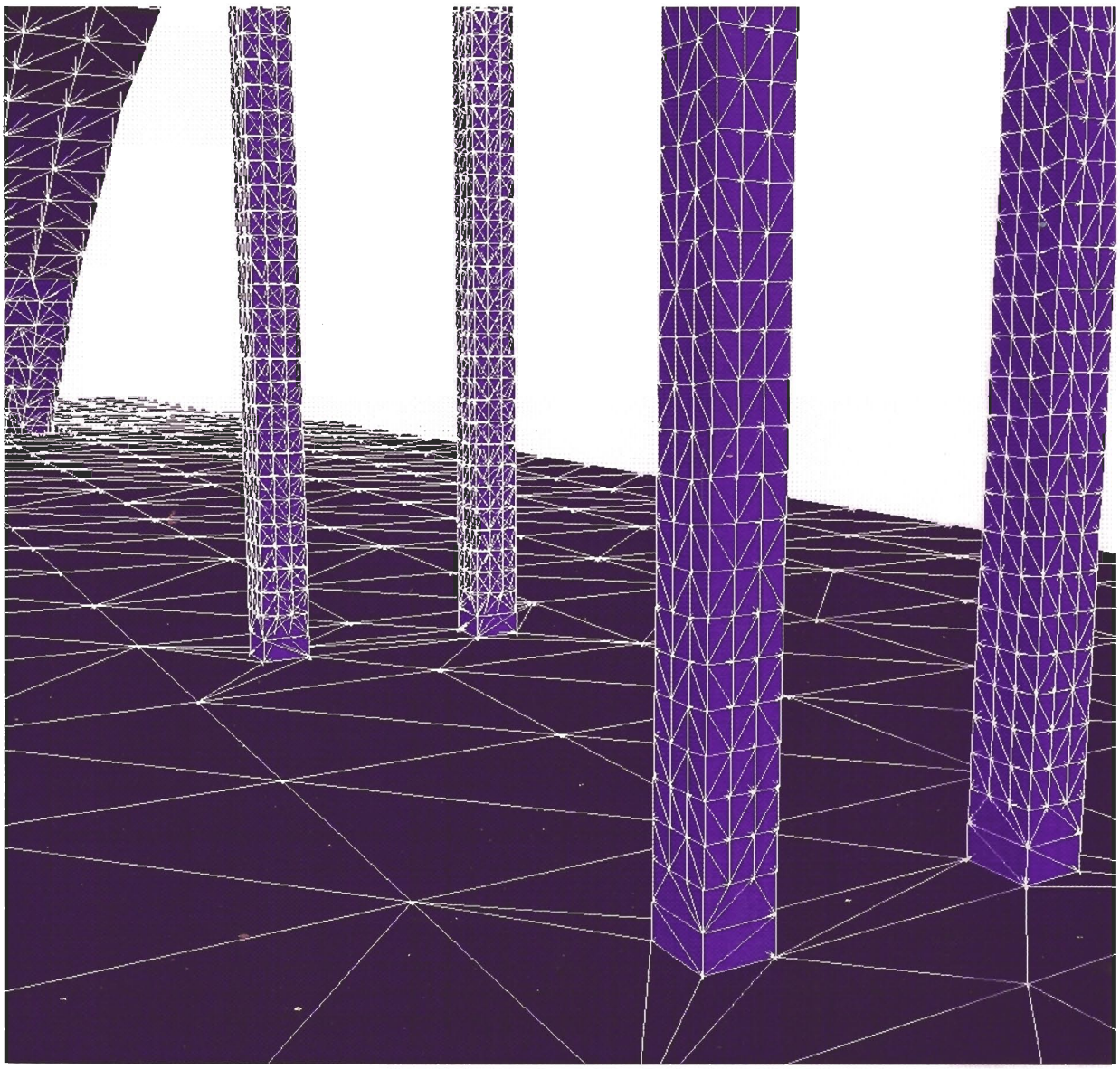
# DOMINIO CON FRACTURAS Y SONDEOS











# Almacenamiento compacto de la matriz

## Ejemplo de almacenamiento compacto de la matriz del sistema

Matriz del sistema

$$\begin{pmatrix} k_{11} & k_{12} & 0 & k_{14} & 0 & 0 \\ k_{21} & k_{22} & 0 & 0 & 0 & 0 \\ 0 & k_{32} & k_{33} & k_{34} & 0 & 0 \\ k_{41} & 0 & k_{43} & k_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & k_{55} & k_{56} \\ 0 & 0 & 0 & 0 & k_{65} & k_{66} \end{pmatrix}$$

$\longleftrightarrow$

Matriz compacta

$$\begin{pmatrix} k_{11} & k_{12} & k_{14} \\ k_{22} & k_{21} & \\ k_{33} & k_{32} & k_{34} \\ k_{44} & k_{41} & k_{43} \\ k_{55} & k_{56} & \\ k_{66} & k_{65} & \end{pmatrix}$$

$\cup$

Matriz de posiciones

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

⇨ El generador de malla suministra la información necesaria para construir la matriz de posiciones **sin hacer búsquedas en toda la malla**.

# CONCLUSIONES Y PERSPECTIVAS FUTURAS

- **CONCLUSIONES.**

**1. El algoritmo de refinamiento/desrefinamiento de mallas encajadas proporciona:**

- **Facilidad para la aplicación del método multimalla.**
- **Mallas flexibles con control automático del número de nodos (incógnitas).**
- **Adaptación muy rápida de la malla (complejidad lineal).**
- **Un buen método para aproximar cualquier solución inicial o geometría.**
- **Un refinamiento local (desrefinando después de un refinamiento global).**

**2. El algoritmo de triangulación tridimensional proporciona:**

- **Solución de los problemas debidos a errores de redondeo.**
- **Posibilidad de fijar diferentes densidades de puntos en aristas, caras y poliedros.**
- **Generación de mallas de buena calidad.**
- **Pruebas experimentales con una complejidad casi lineal.**

- **PERSPECTIVAS FUTURAS.**

- **Desarrollo del algoritmo de refinamiento/desrefinamiento en 3-D.**
- **Mejora de la entrada de datos y conformidad con la frontera en 3-D.**
- **Aplicación de las técnicas de mallado en problemas de Ingeniería en 3-D.**

# HISTORIA

## Problema de elasticidad con sólo refinamiento

## Clasical formulation:

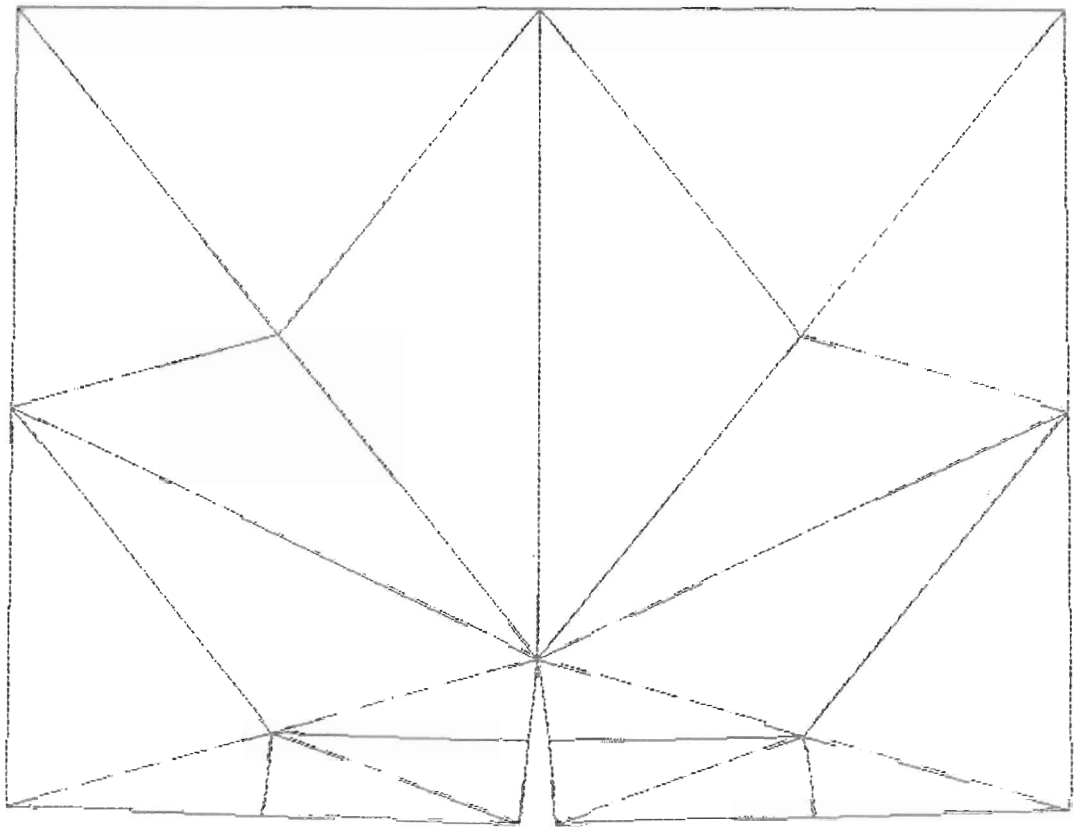
$$\sum_{j=1}^{2} \frac{\partial}{\partial x_j} \sigma_{ij}(\vec{u}) = 0 \quad \text{in} \quad \Omega, \qquad \text{where i = 1, 2}$$

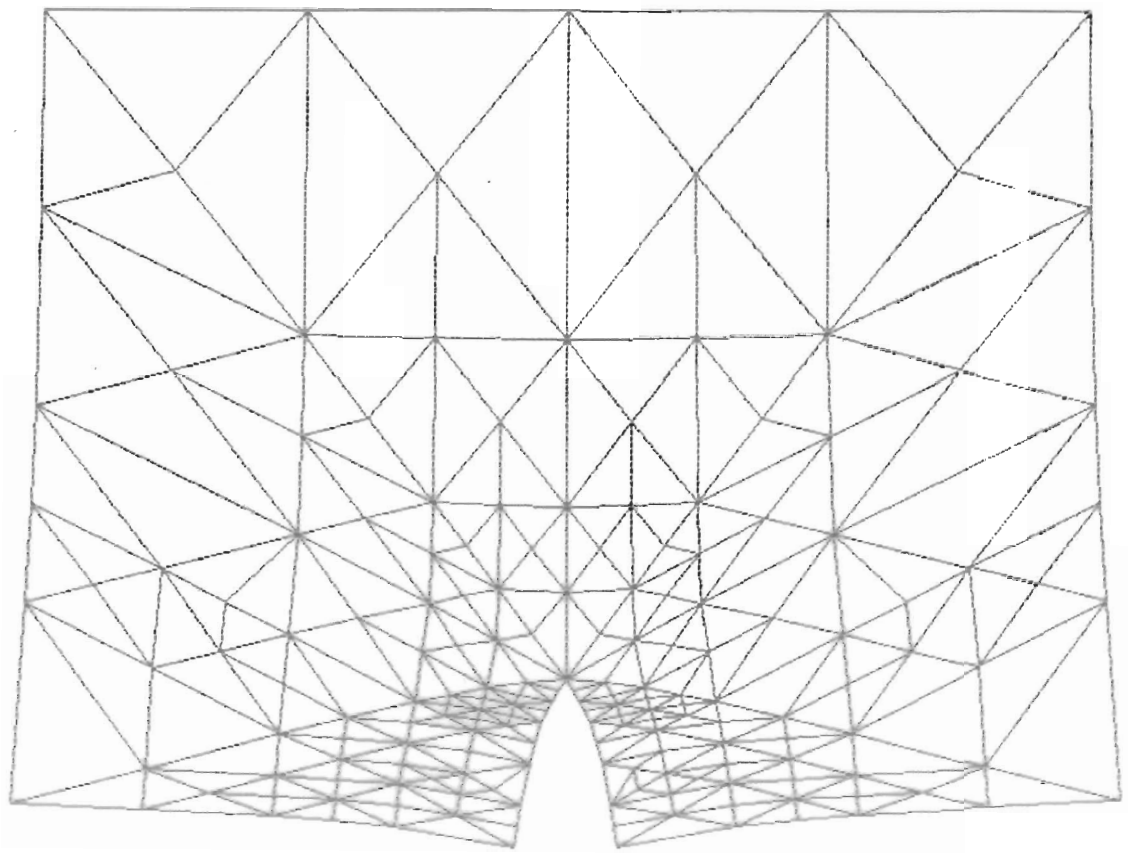
$$\vec{u}_i = \vec{0} \quad \text{on} \quad \Gamma_0, \qquad u_2(A) = 0$$

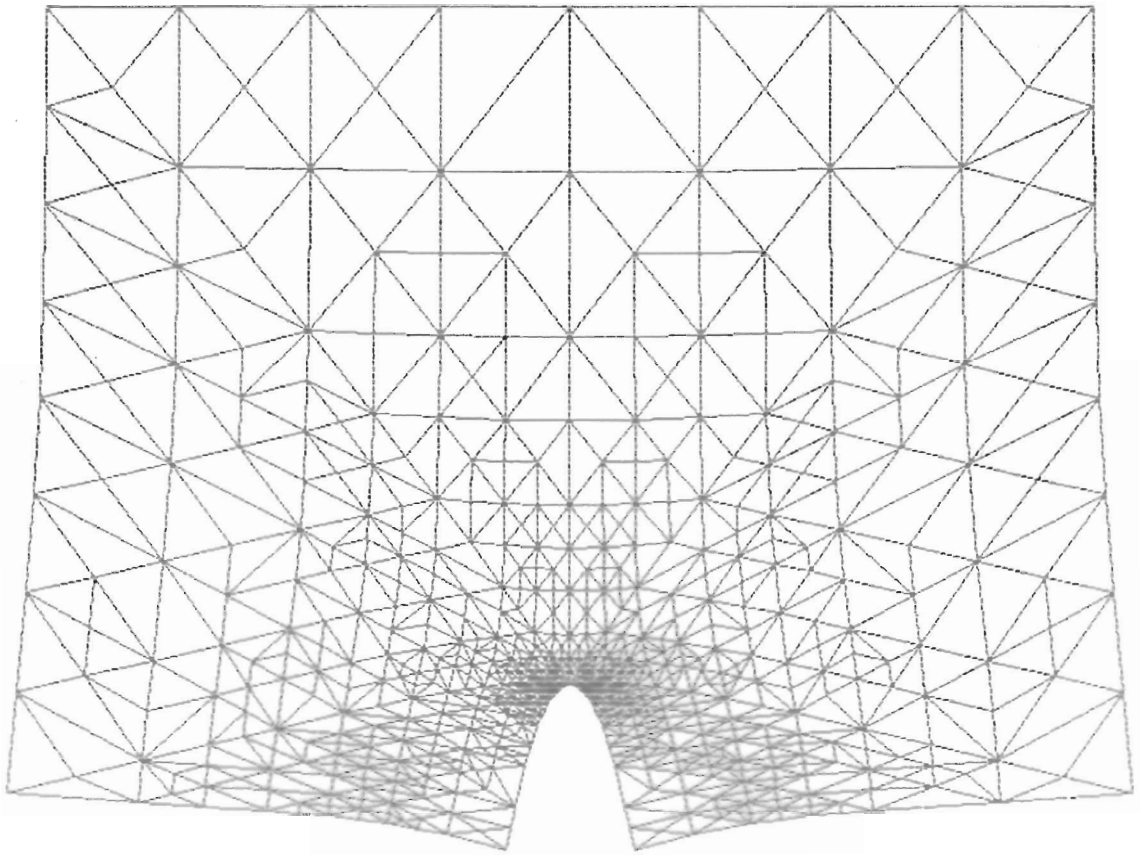
$$\sum_{j=1}^{2} \sigma_{1j}(\vec{u}) n_j = 0 \quad \text{on} \quad \Gamma_{1,1}, \qquad \sum_{j=1}^{2} \sigma_{2j}(\vec{u}) n_j = 1 \quad \text{on} \quad \Gamma_{1,1}$$

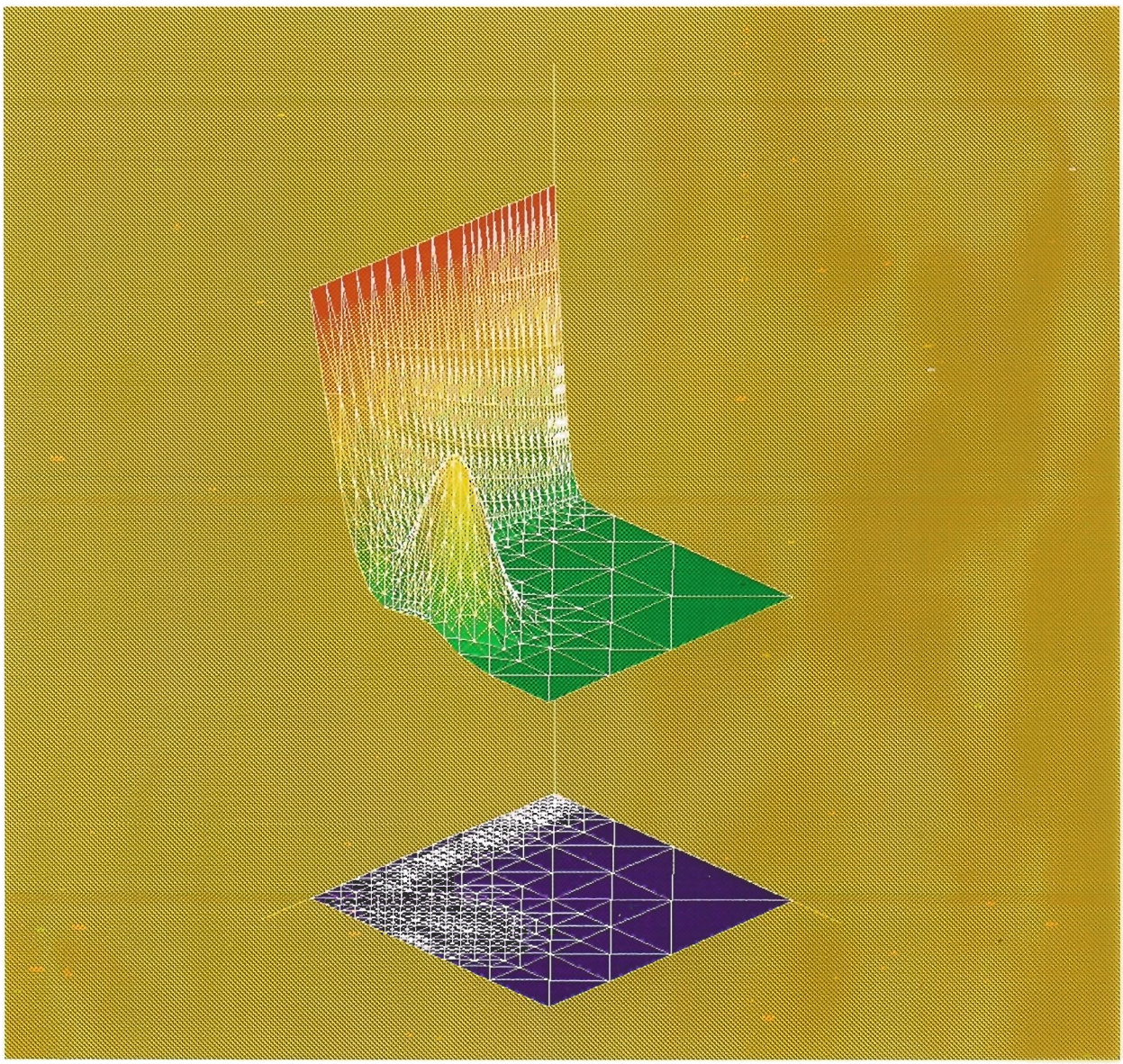
$$\sum_{j=1}^{2} \sigma_{1j}(\vec{u}) n_j = 0 \quad \text{on} \quad \Gamma_{1,2}, \qquad \sum_{j=1}^{2} \sigma_{2j}(\vec{u}) n_j = -1 \quad \text{on} \quad \Gamma_{1,2}$$

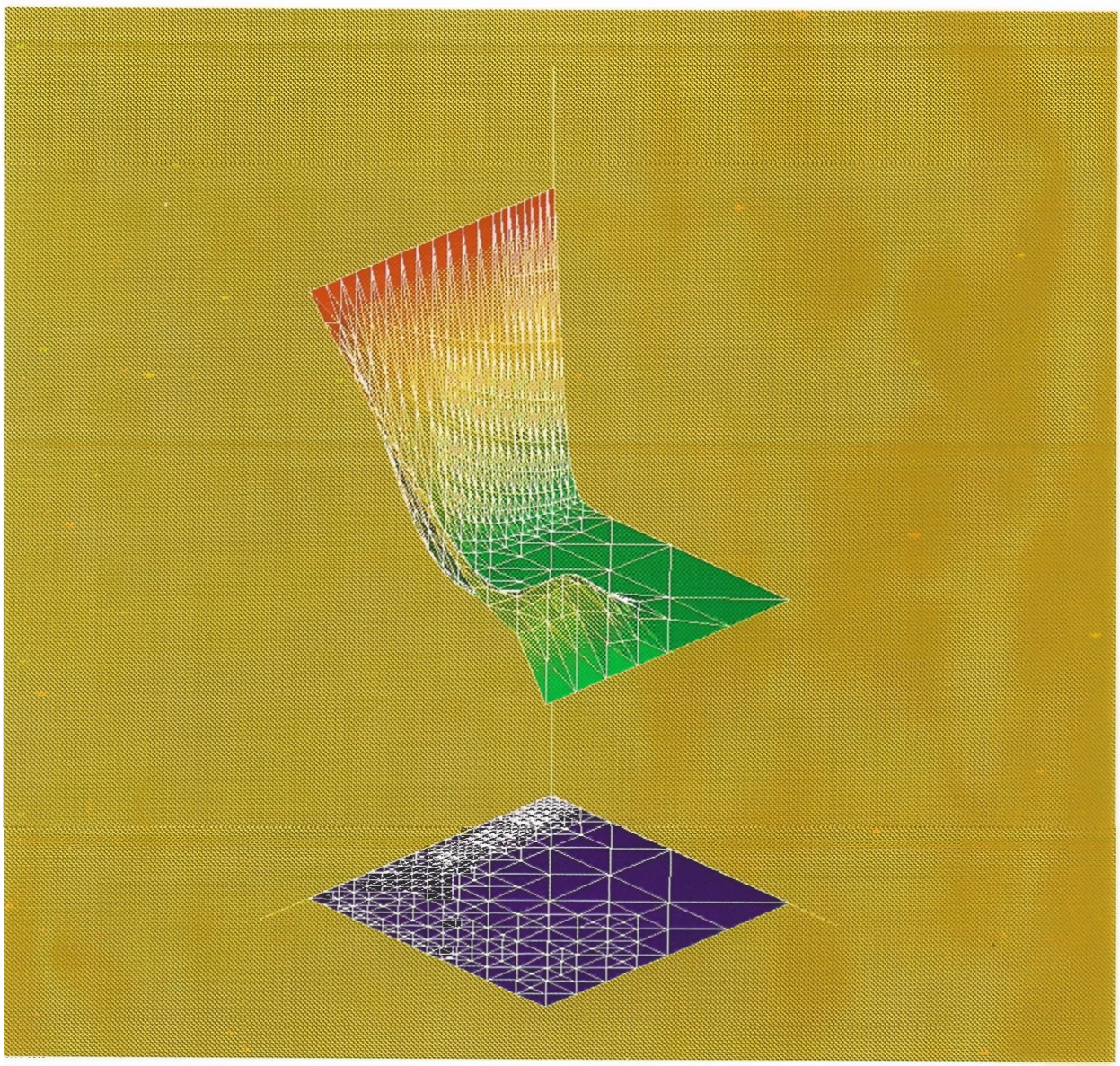
$$\sum_{j=1}^{2} \sigma_{ij}(\vec{u}) n_j = 0 \quad \text{where} \quad i = 1,2 \quad \text{on} \quad \Gamma_{1,3}$$

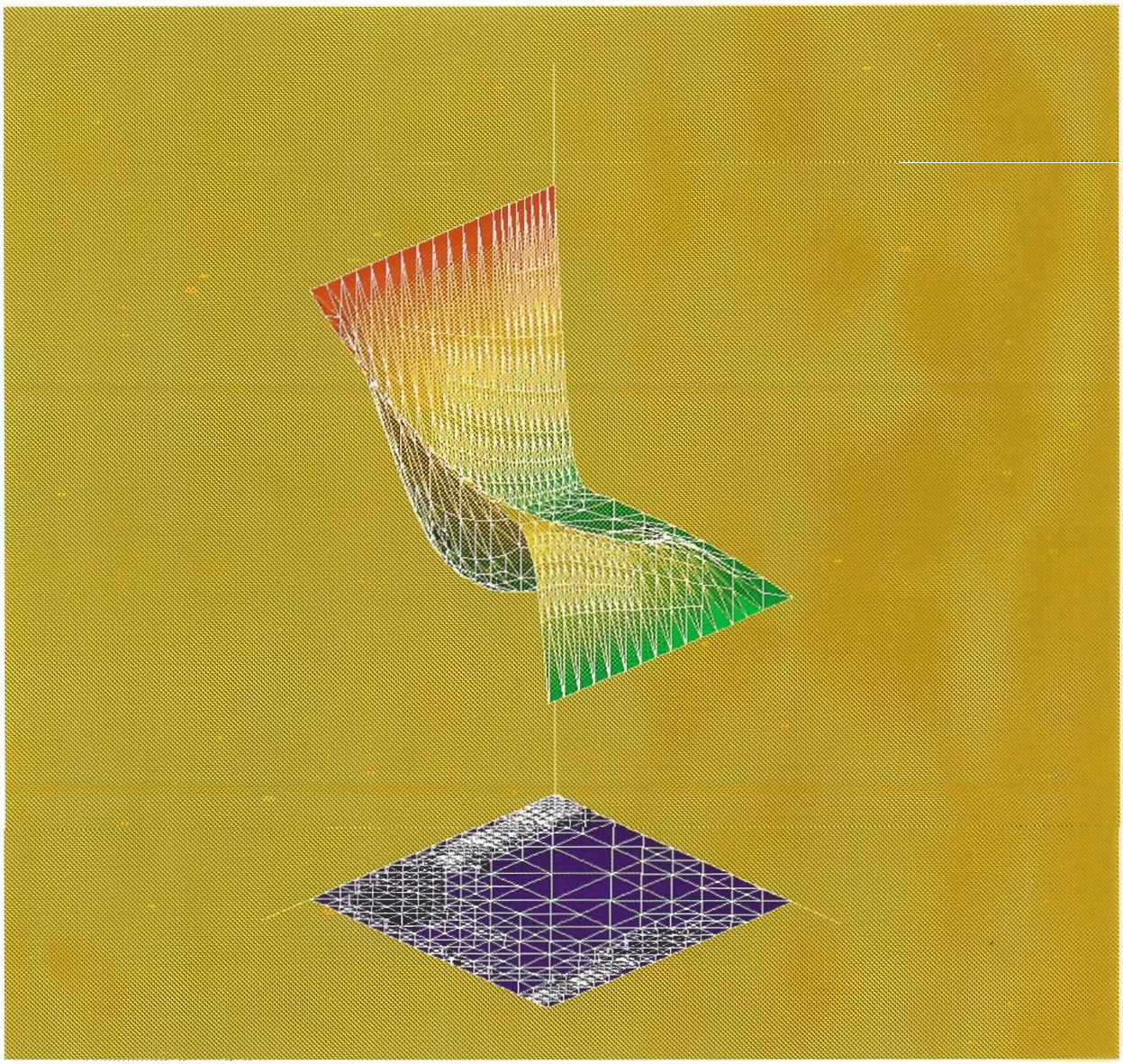


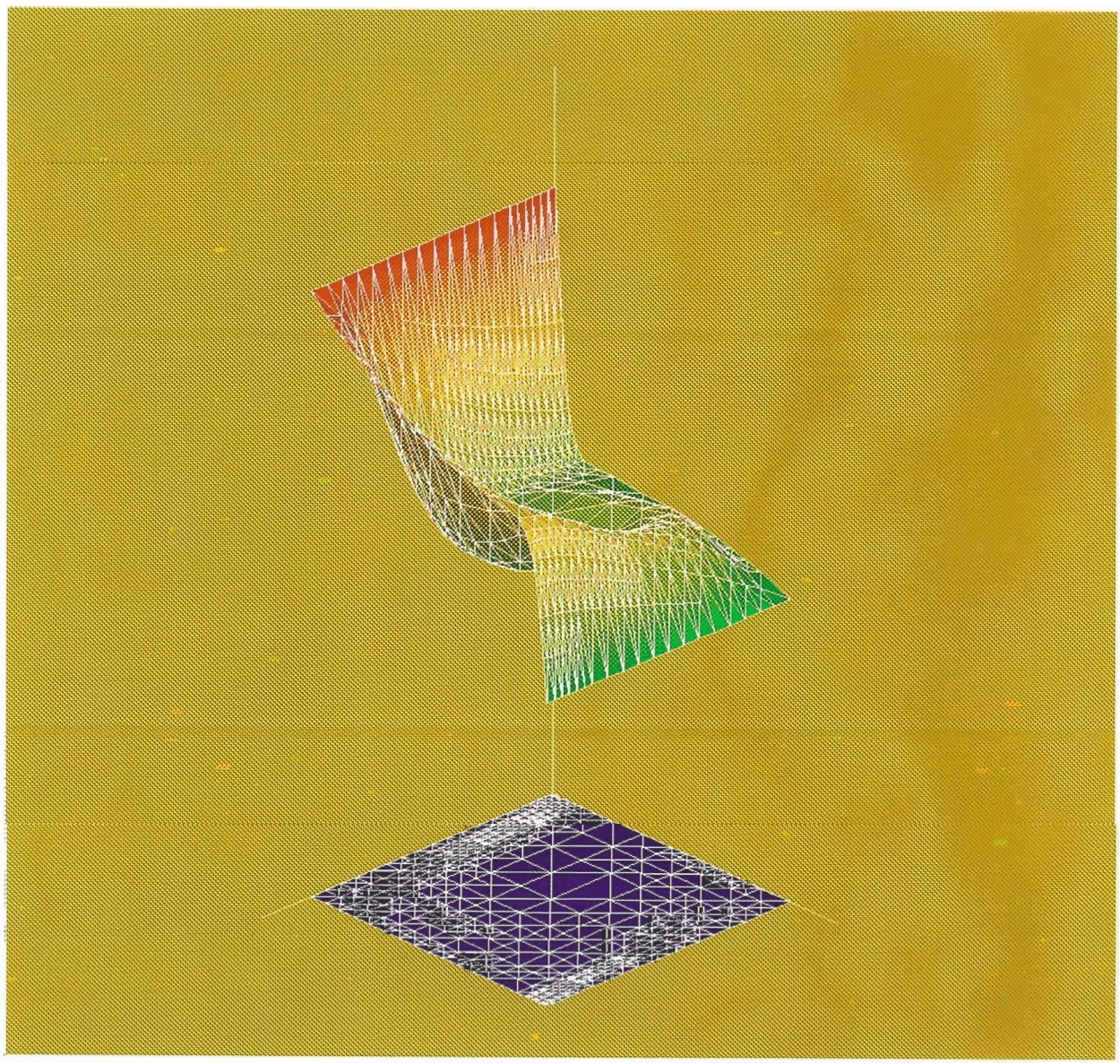


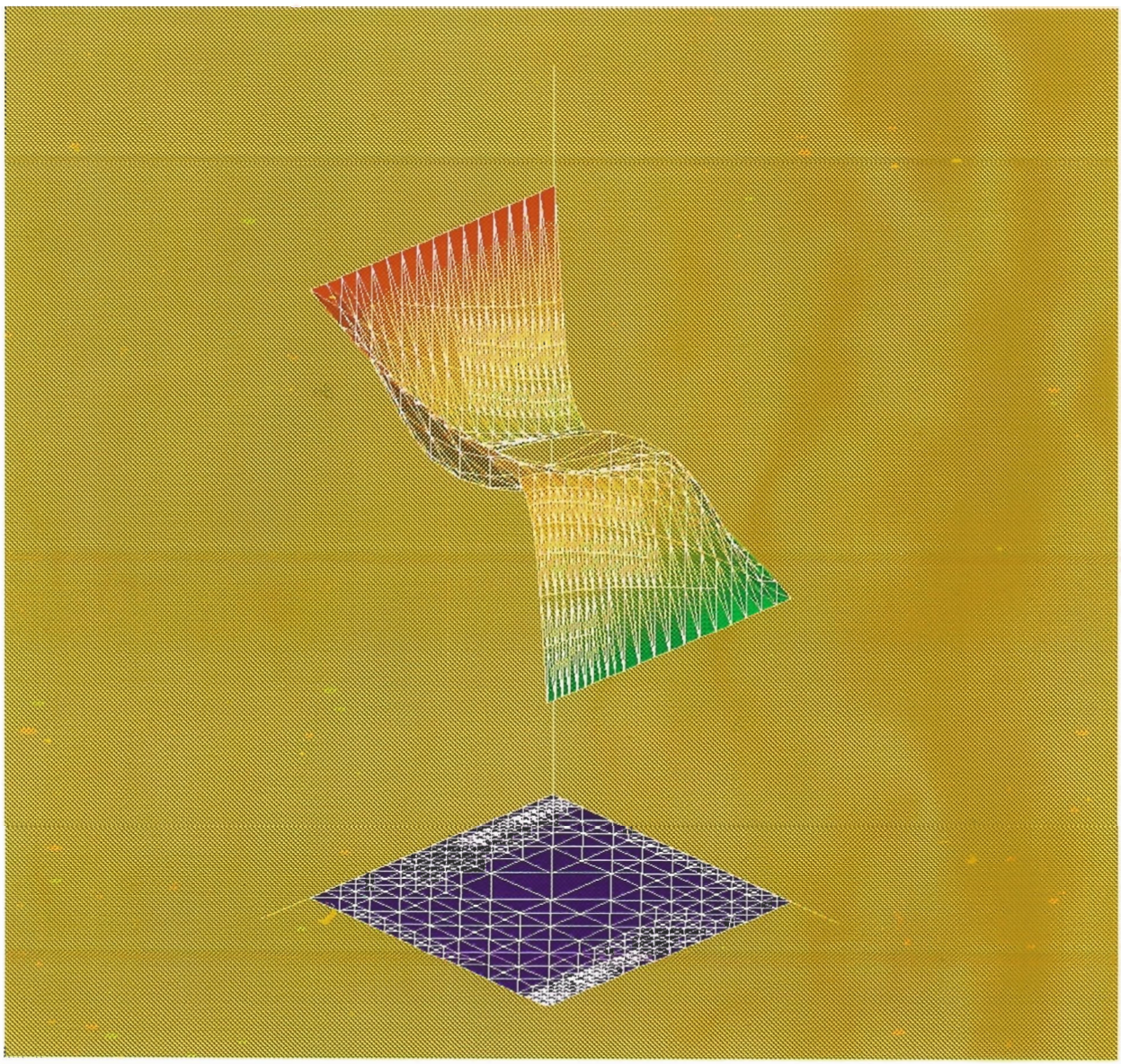


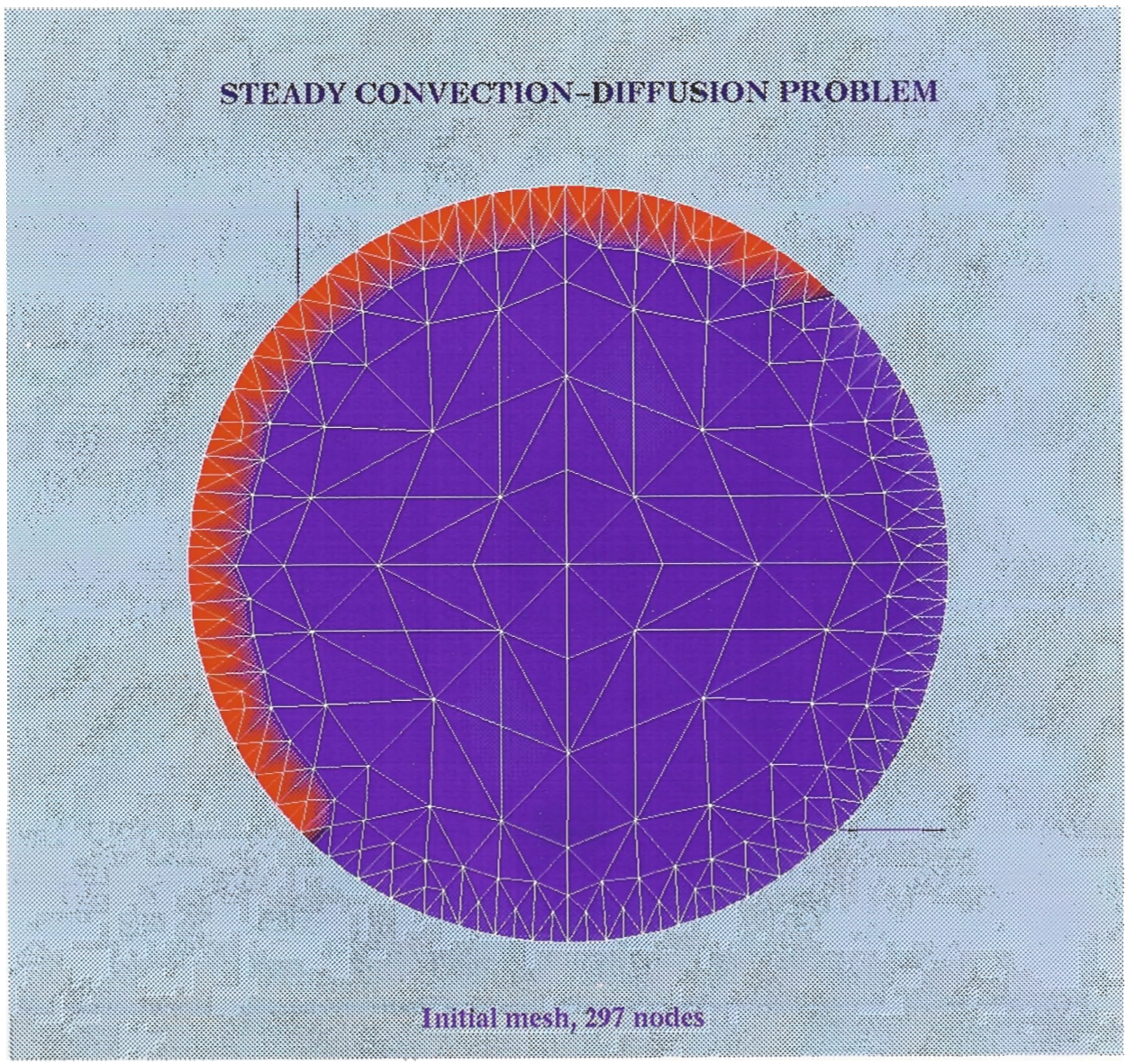


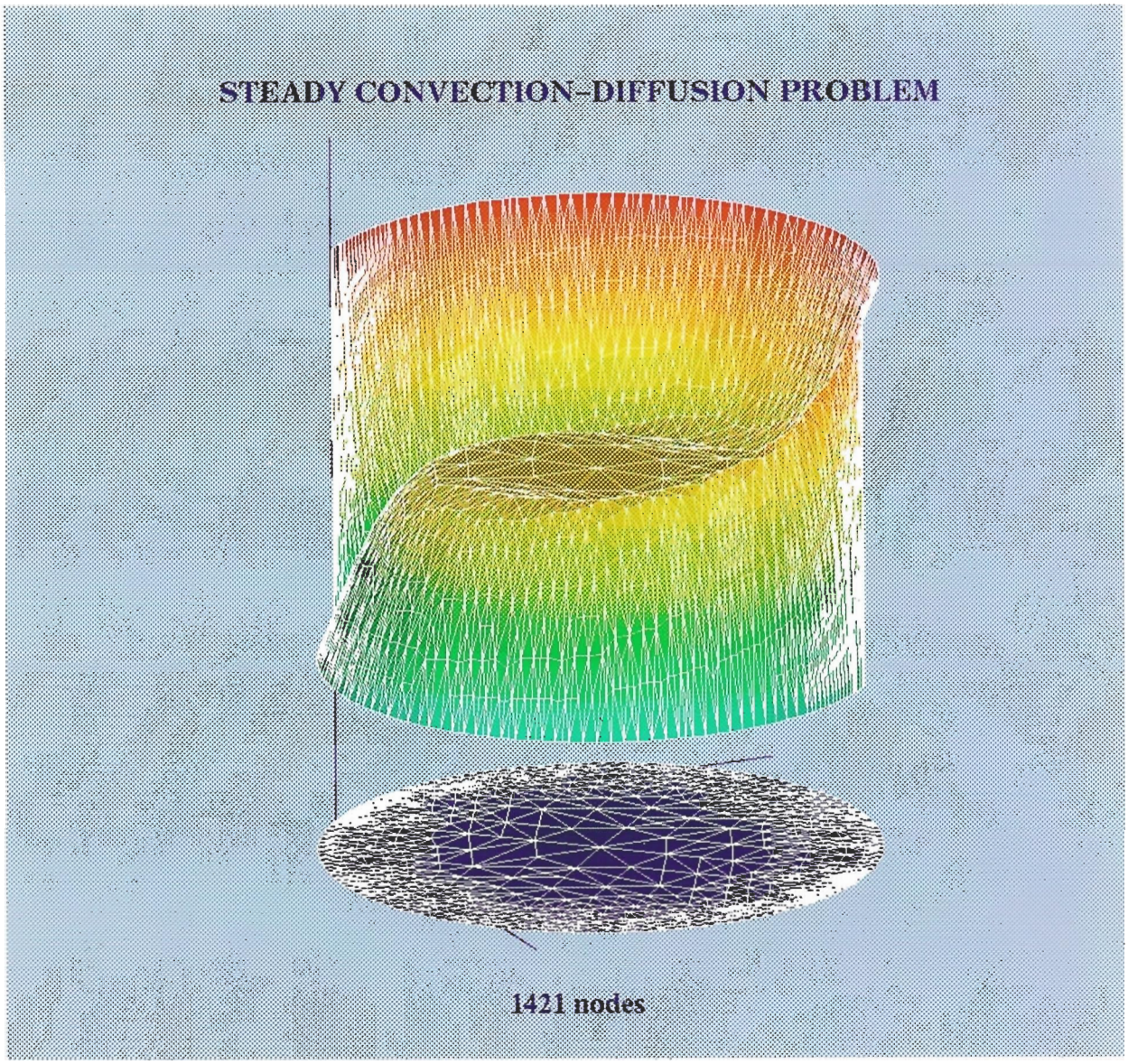


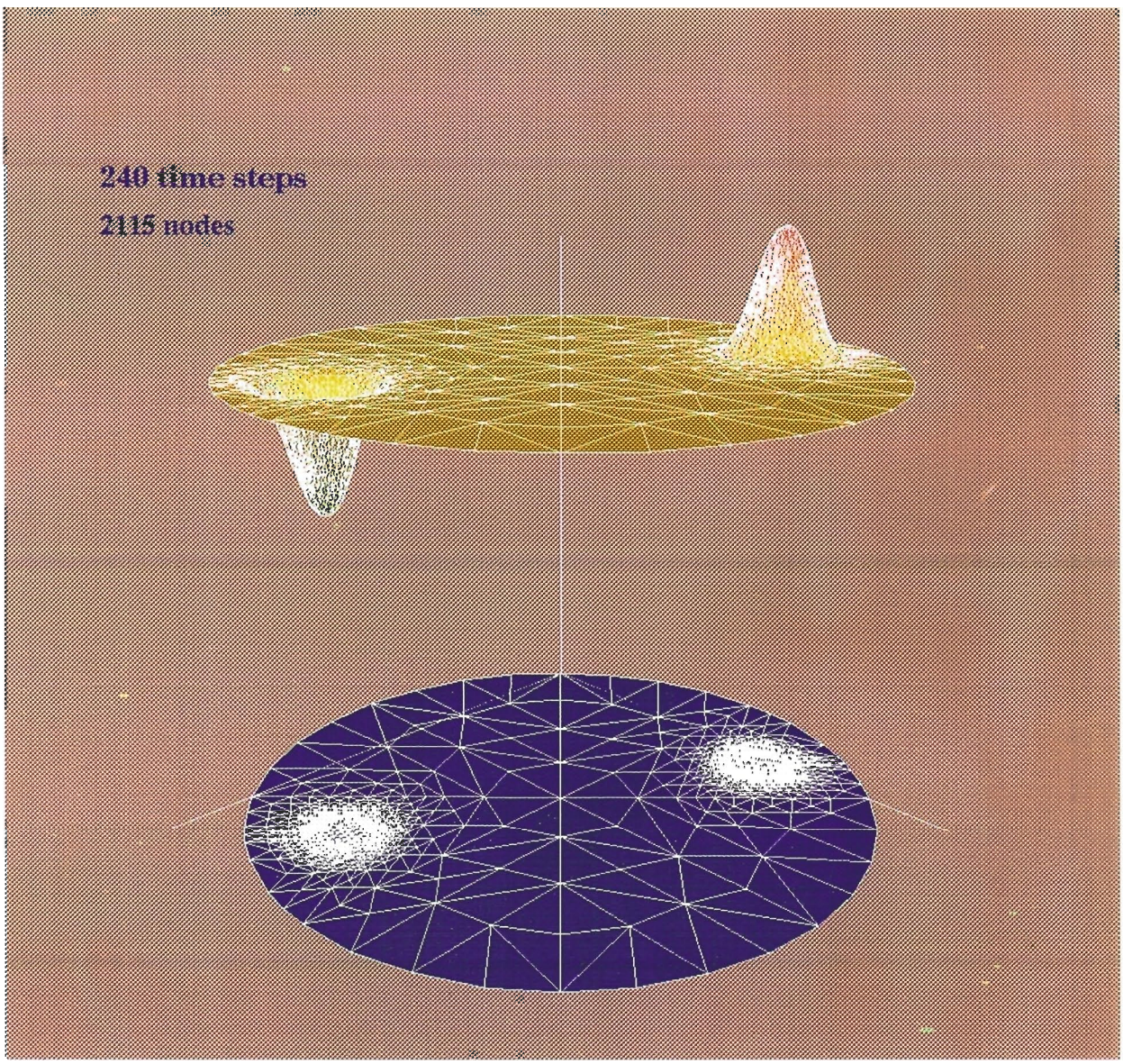


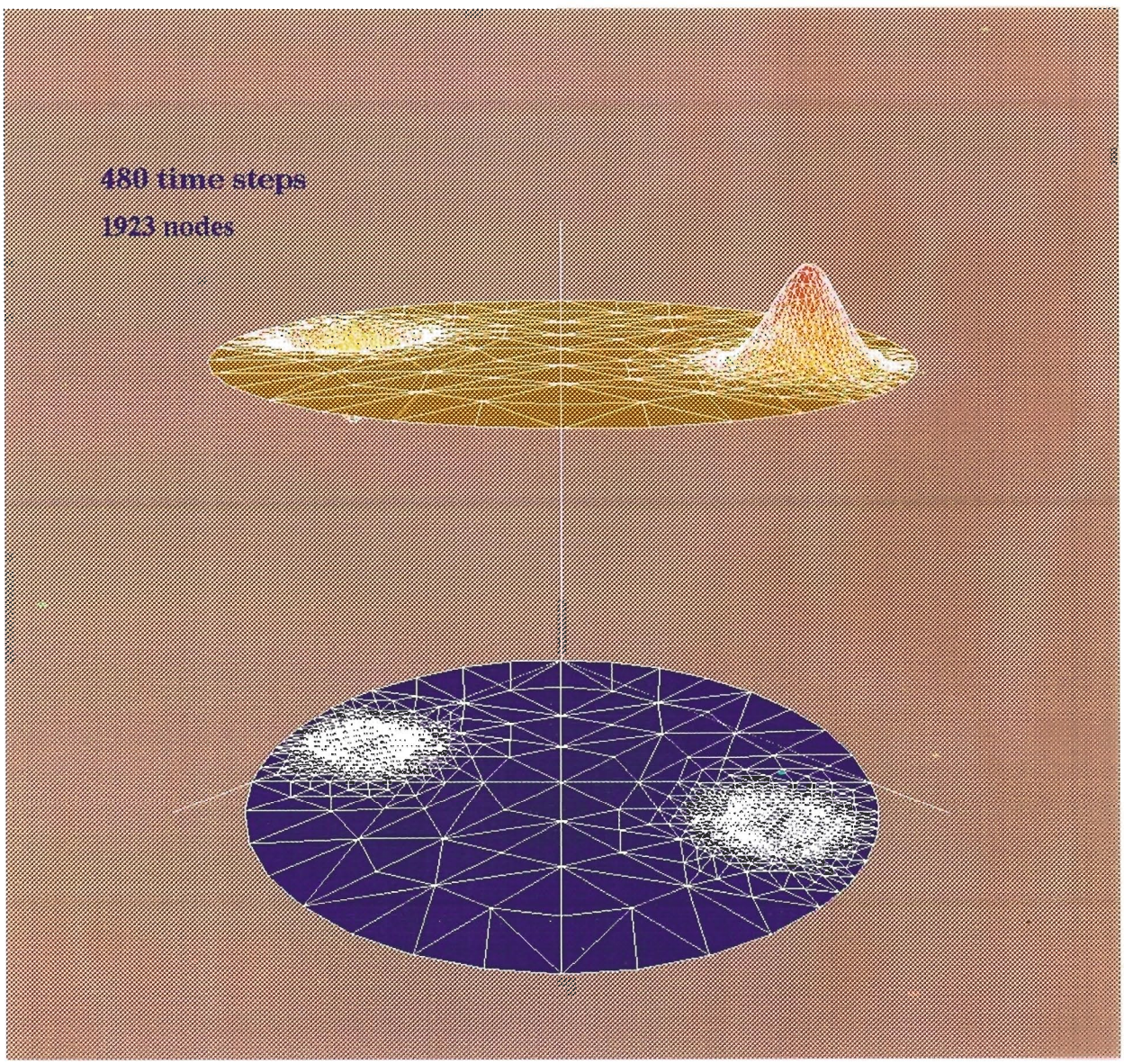


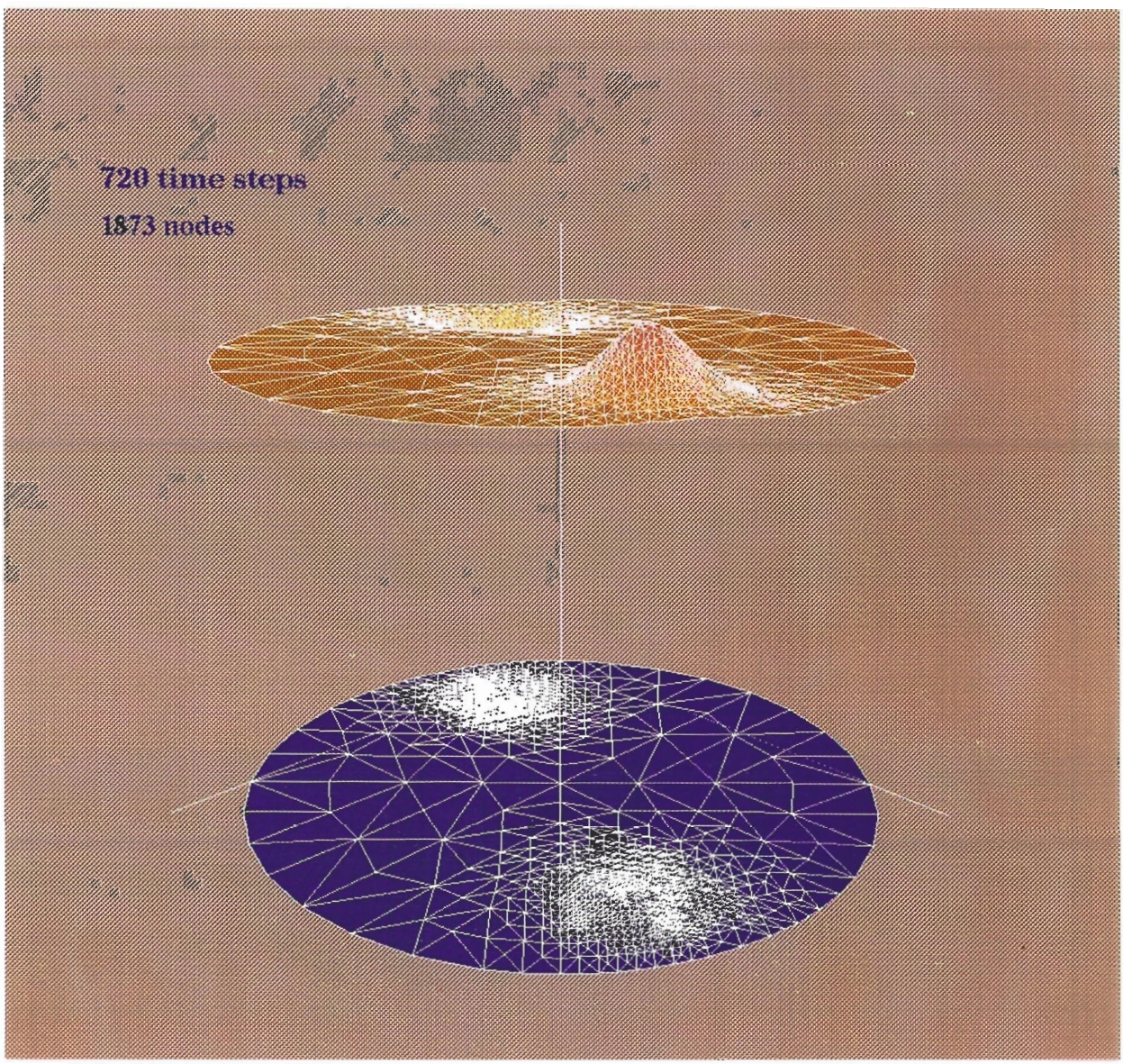


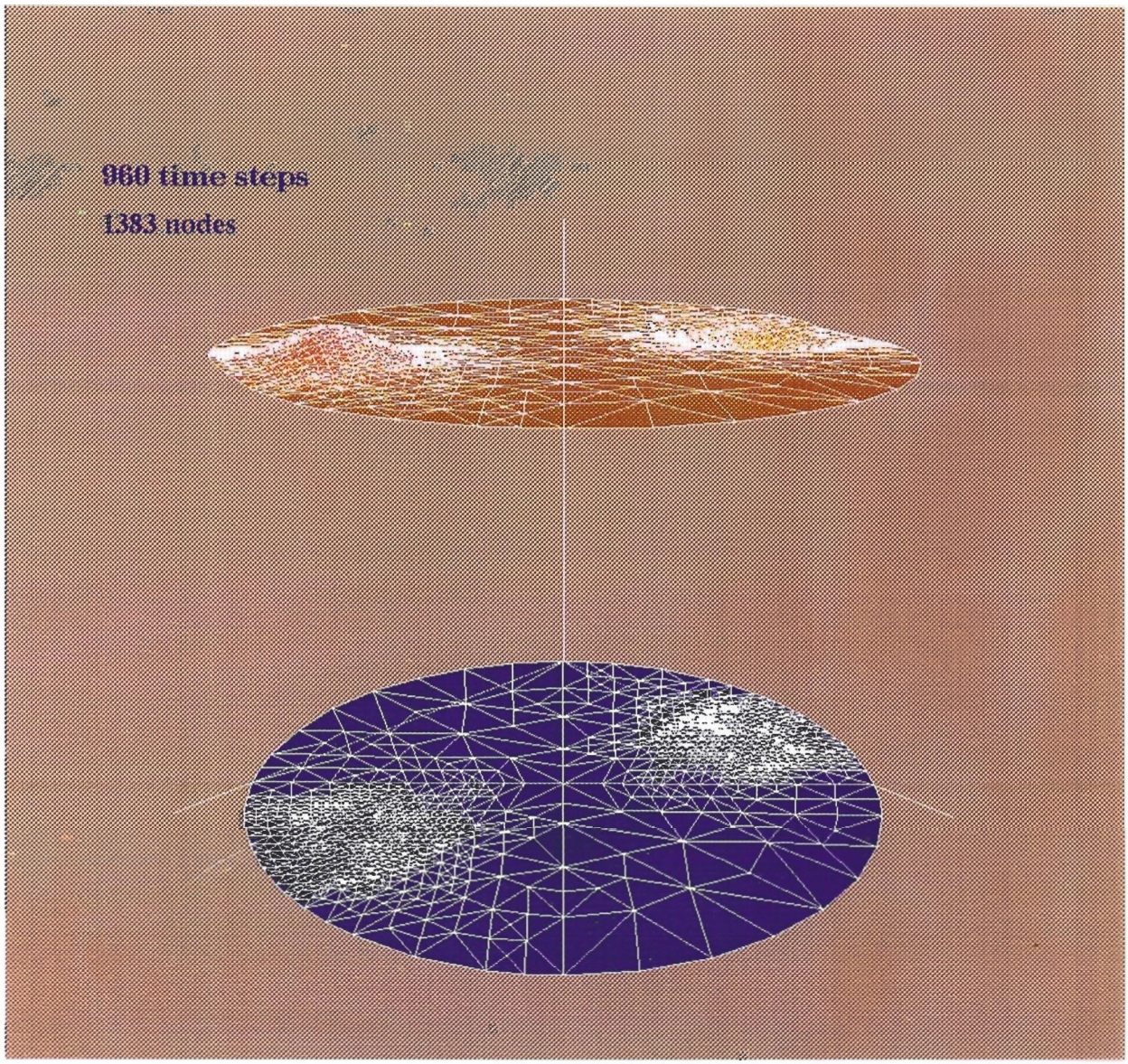

STEADY CONVECTION–DIFFUSION PROBLEM
Initial mesh, 297 nodes

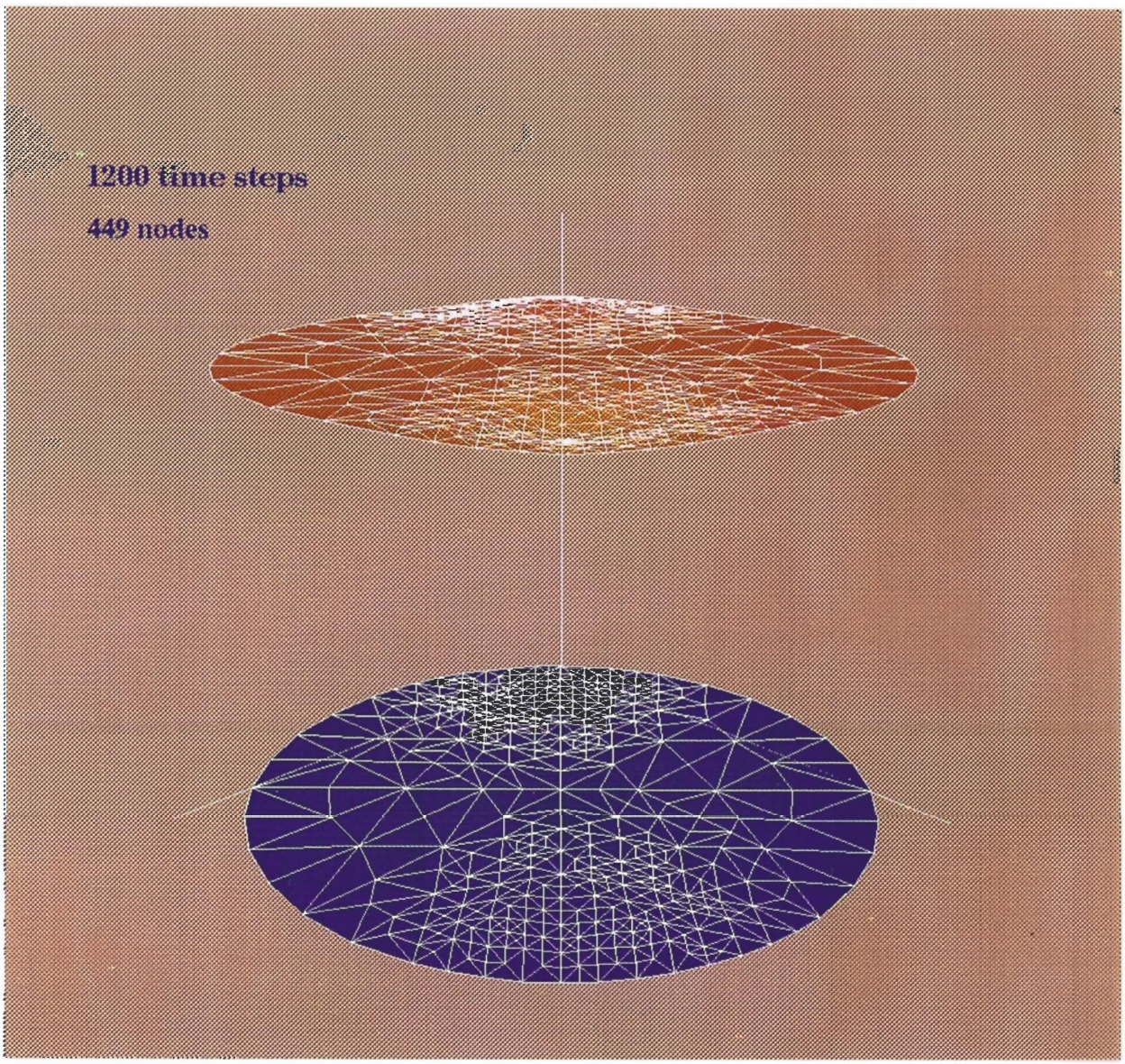

STEADY CONVECTION-DIFFUSION PROBLEM
1421 nodes


240 time steps
2115 nodes


480 time steps
1923 nodes


720 time steps
1873 nodes


960 time steps
1383 nodes

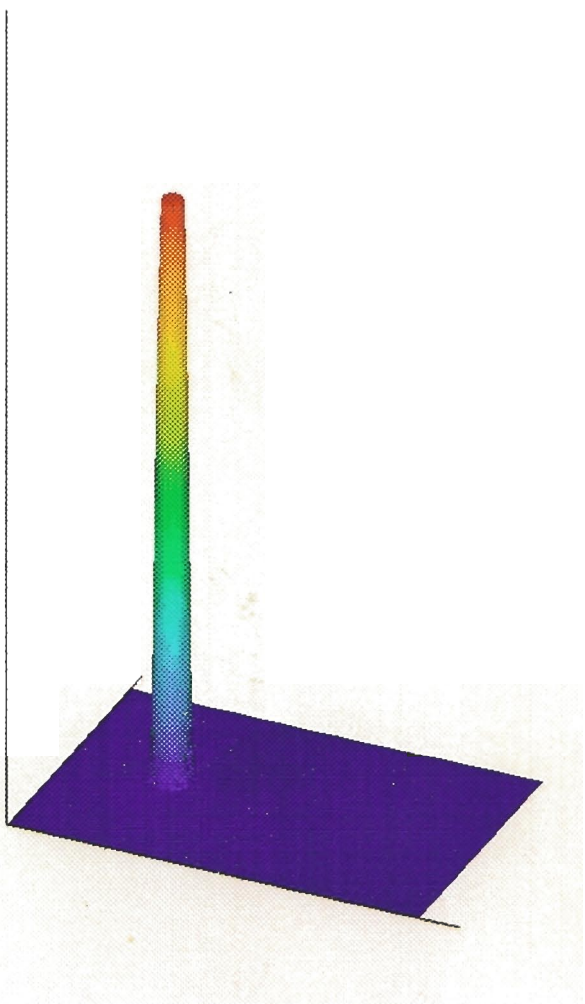

1200 time steps
449 nodes

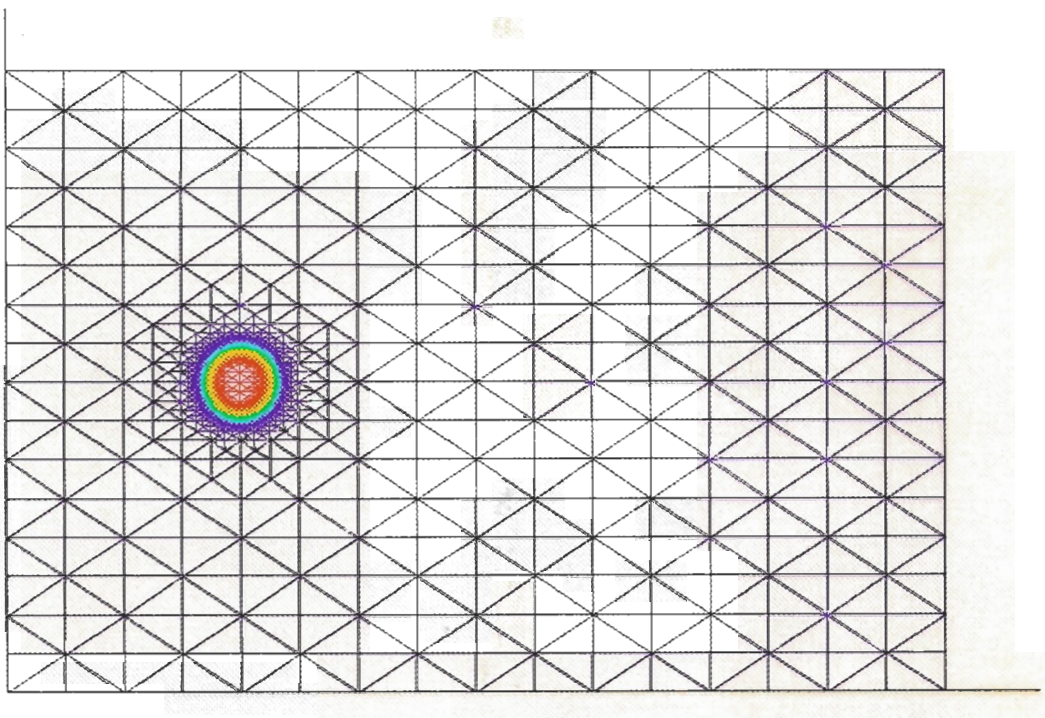
# SIMULACIÓN DE PROPAGACIÓN DE FUEGO

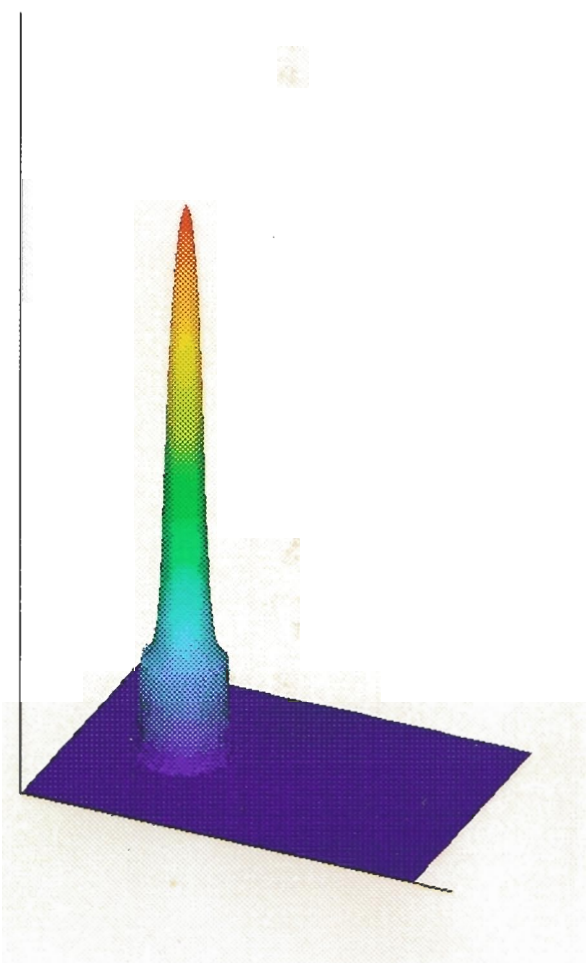
$$\frac{\partial T}{\partial t} - \mathrm{div}(K\vec{\nabla}T) + \vec{v}.\vec{\nabla}T + h(T)(T - T_\infty) = f(t,T)$$

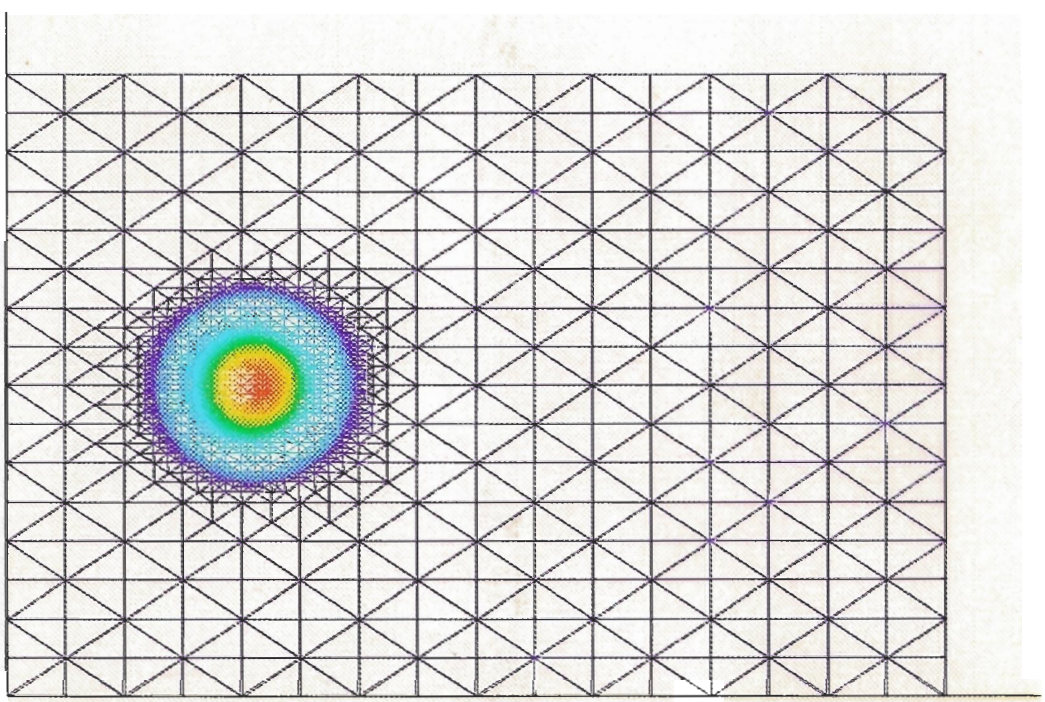
$$\frac{dC_A}{dt} = -\lambda_A C_A \; ; \; \lambda_A = Ae^{-\frac{E_A}{RT}}$$

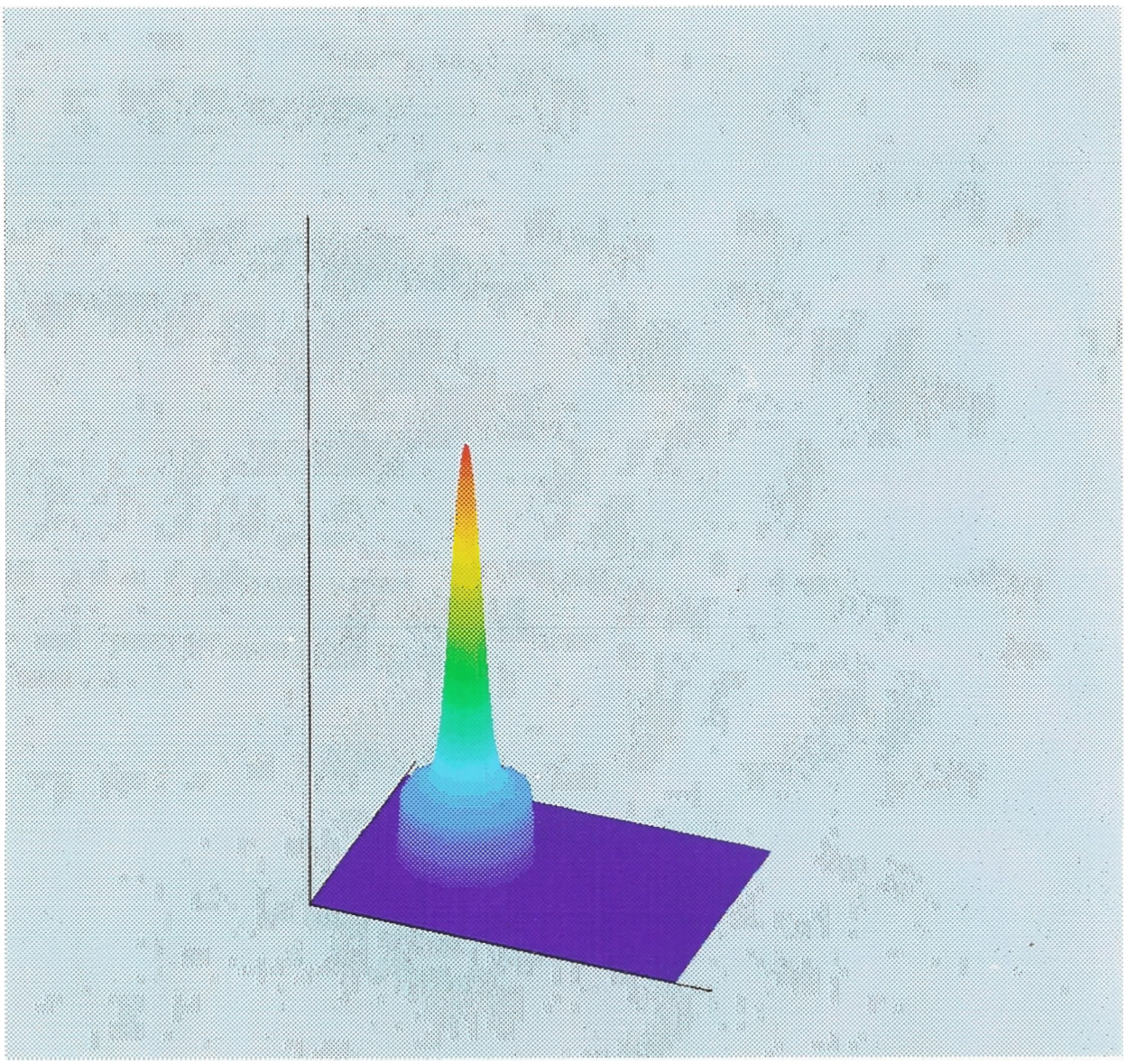
- Análisis mediante diferencias finitas y elementos finitos.
- Estudio de estabilidad del esquema en diferencias finitas (paso de tiempo).











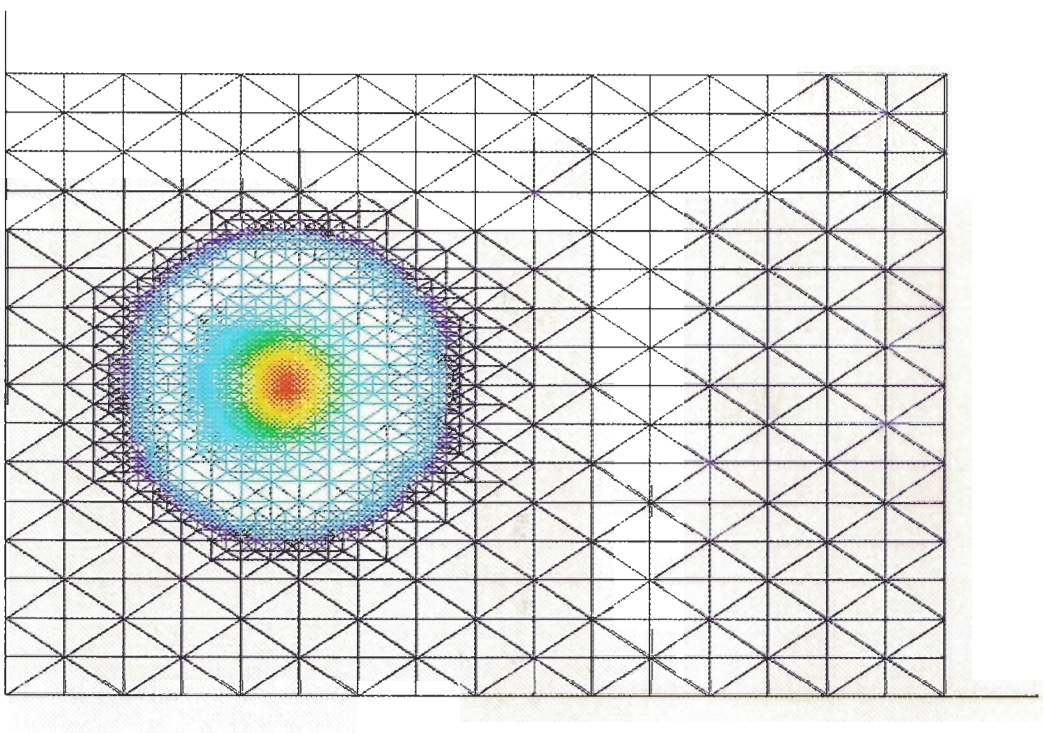
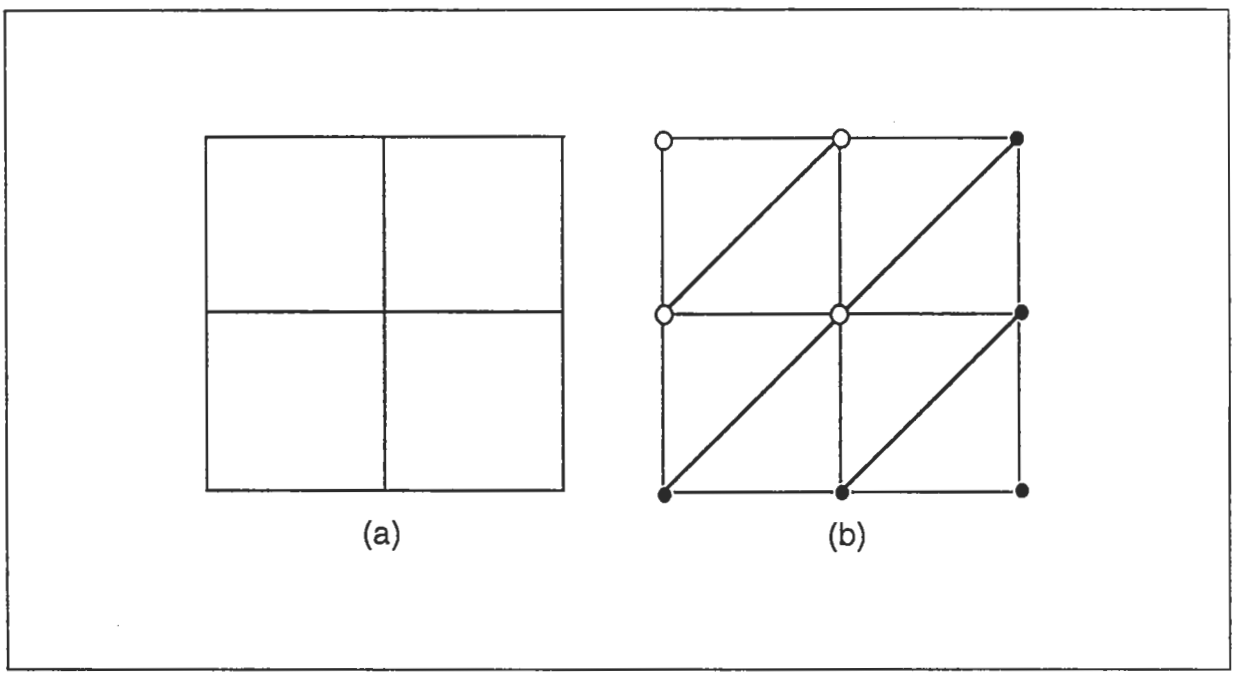
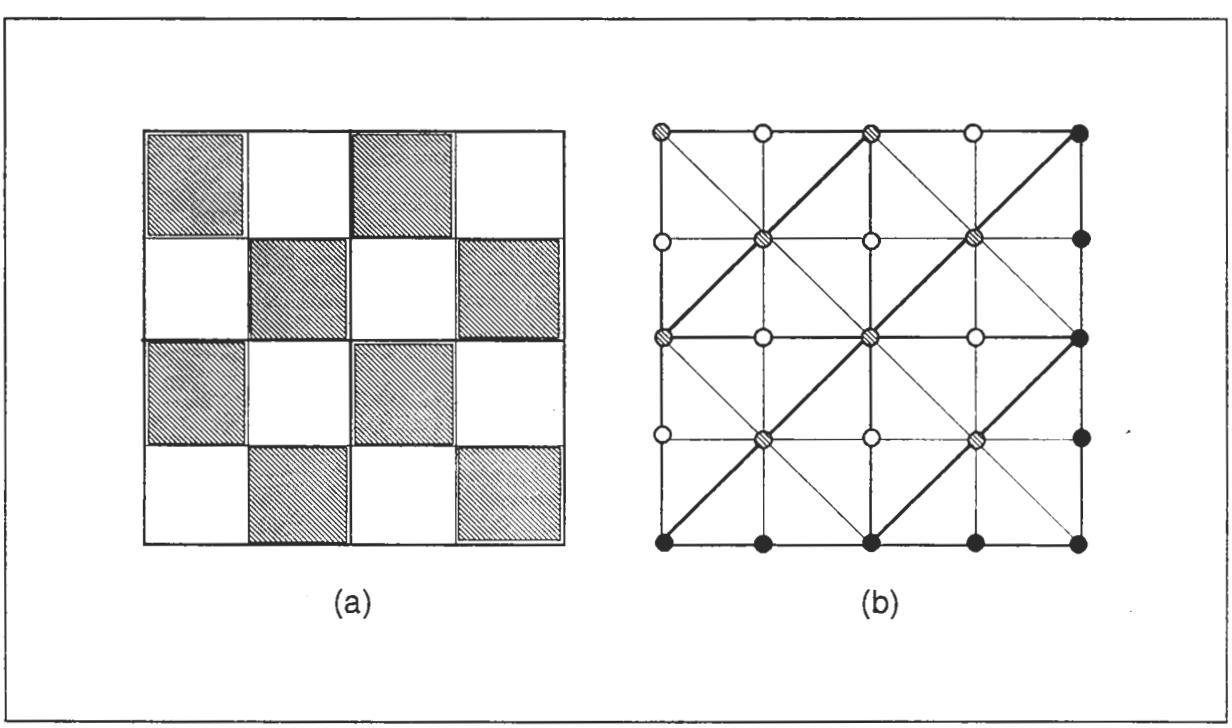


*Imagen clásica 256x256.*




*(a) Imagen 2x2 pixels, (b) malla triangular asociada*




*(a) Imagen 4x4 pixels, (b) malla triangular*

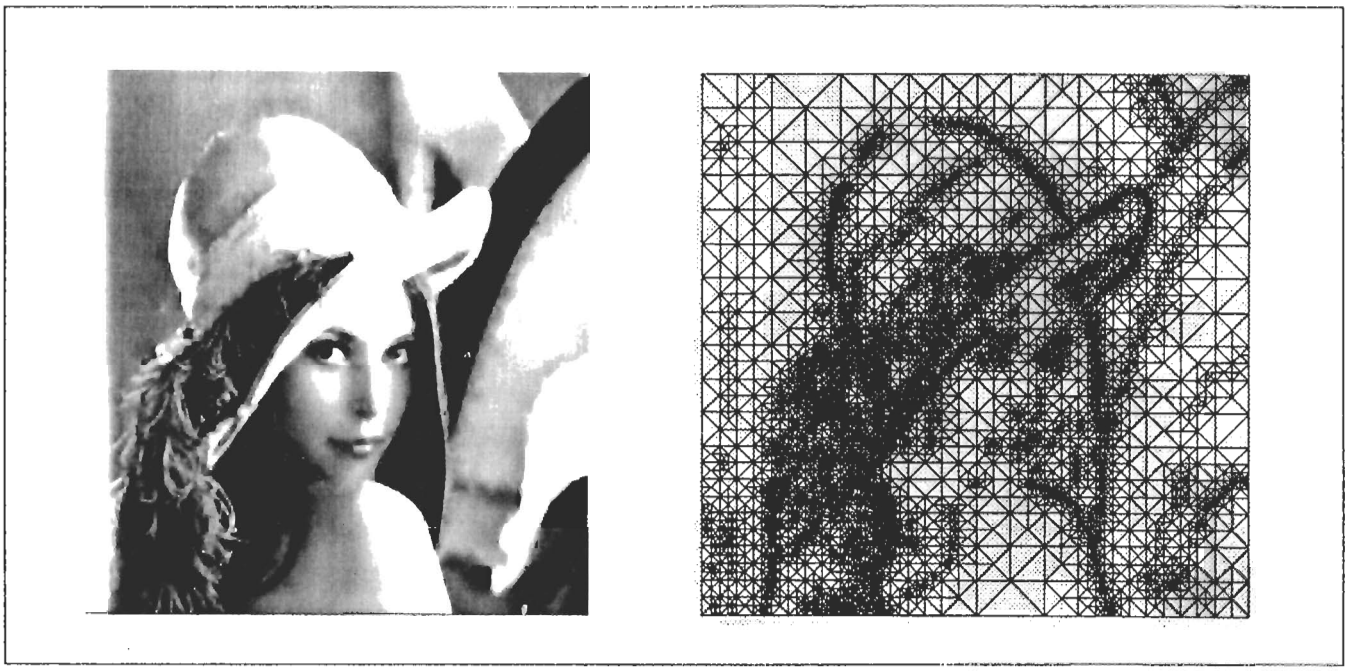


*Imagen y malla para $\varepsilon = 30$: 7.736 nodos, SNR(dB) = 22.58.*

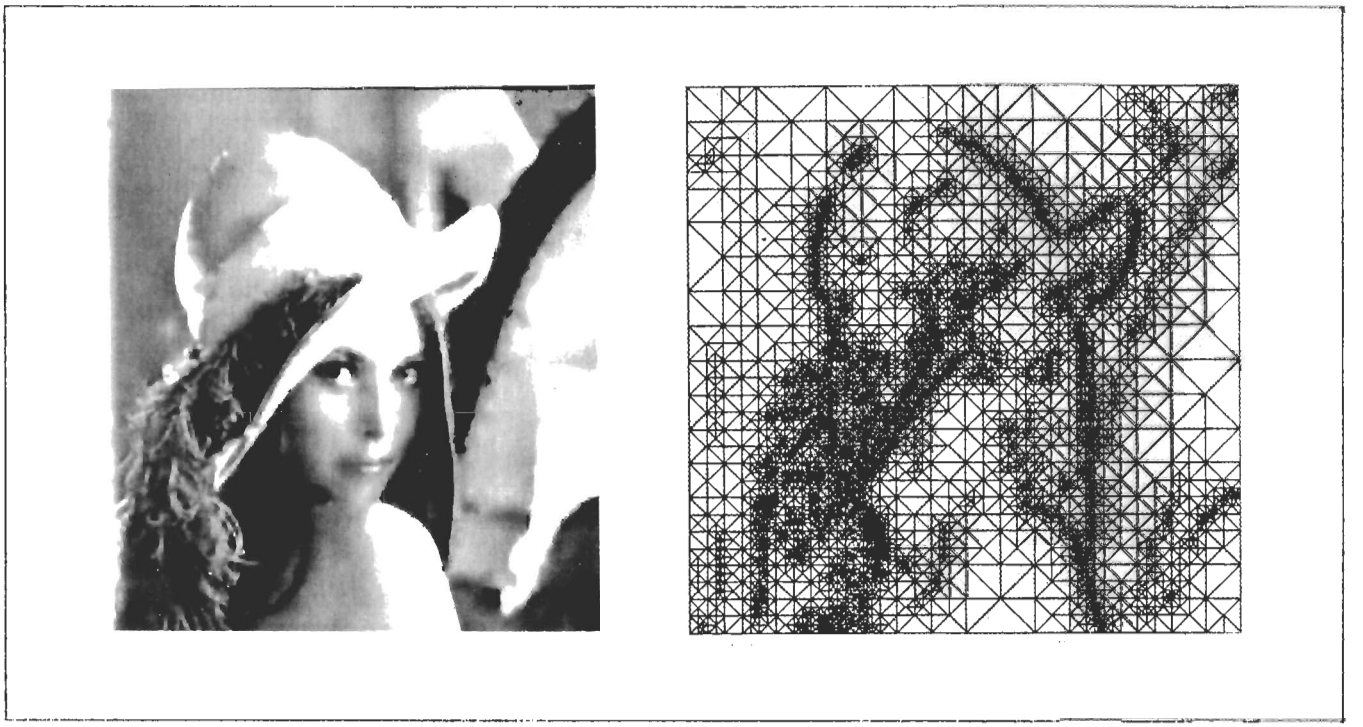
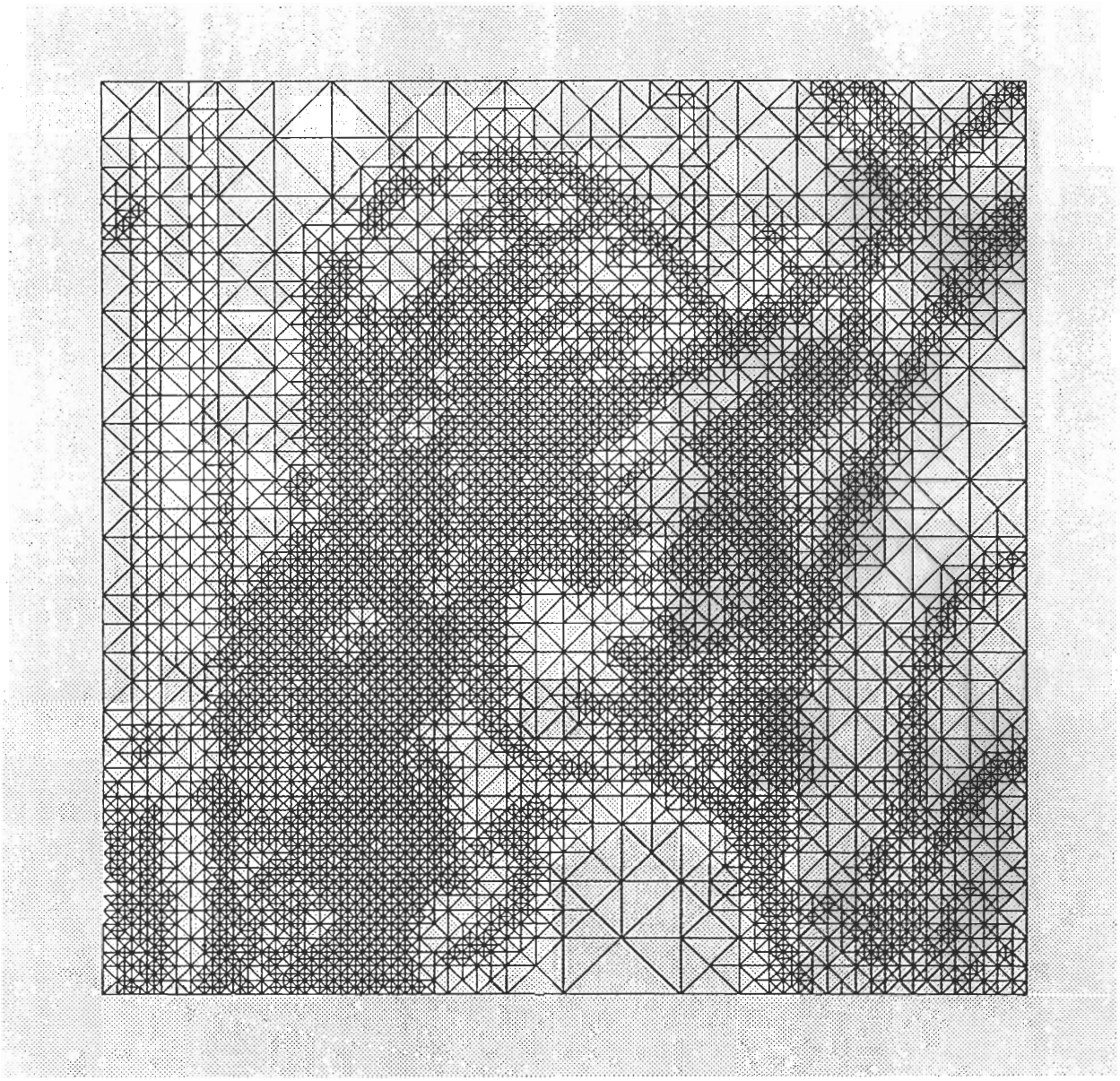
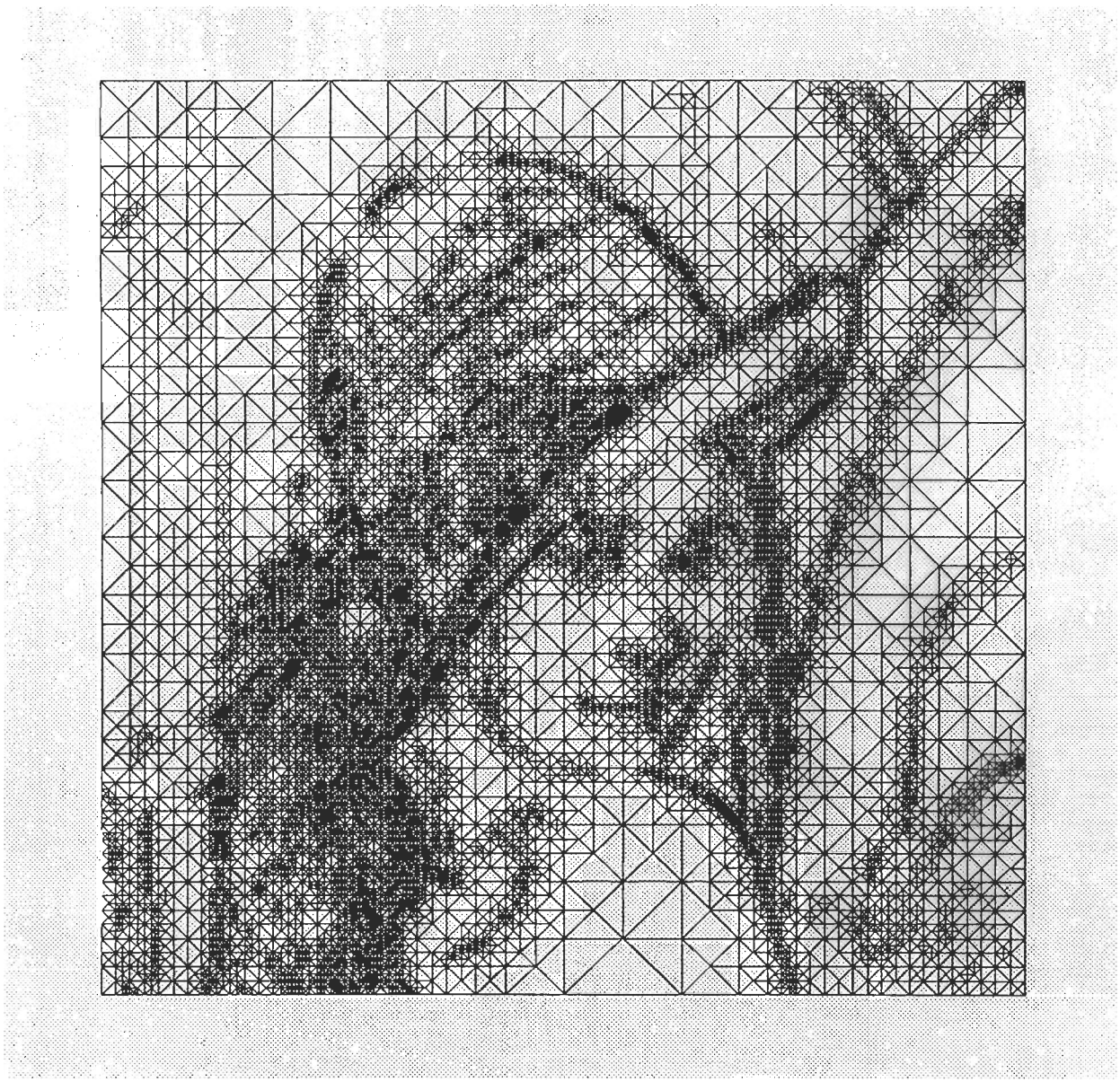
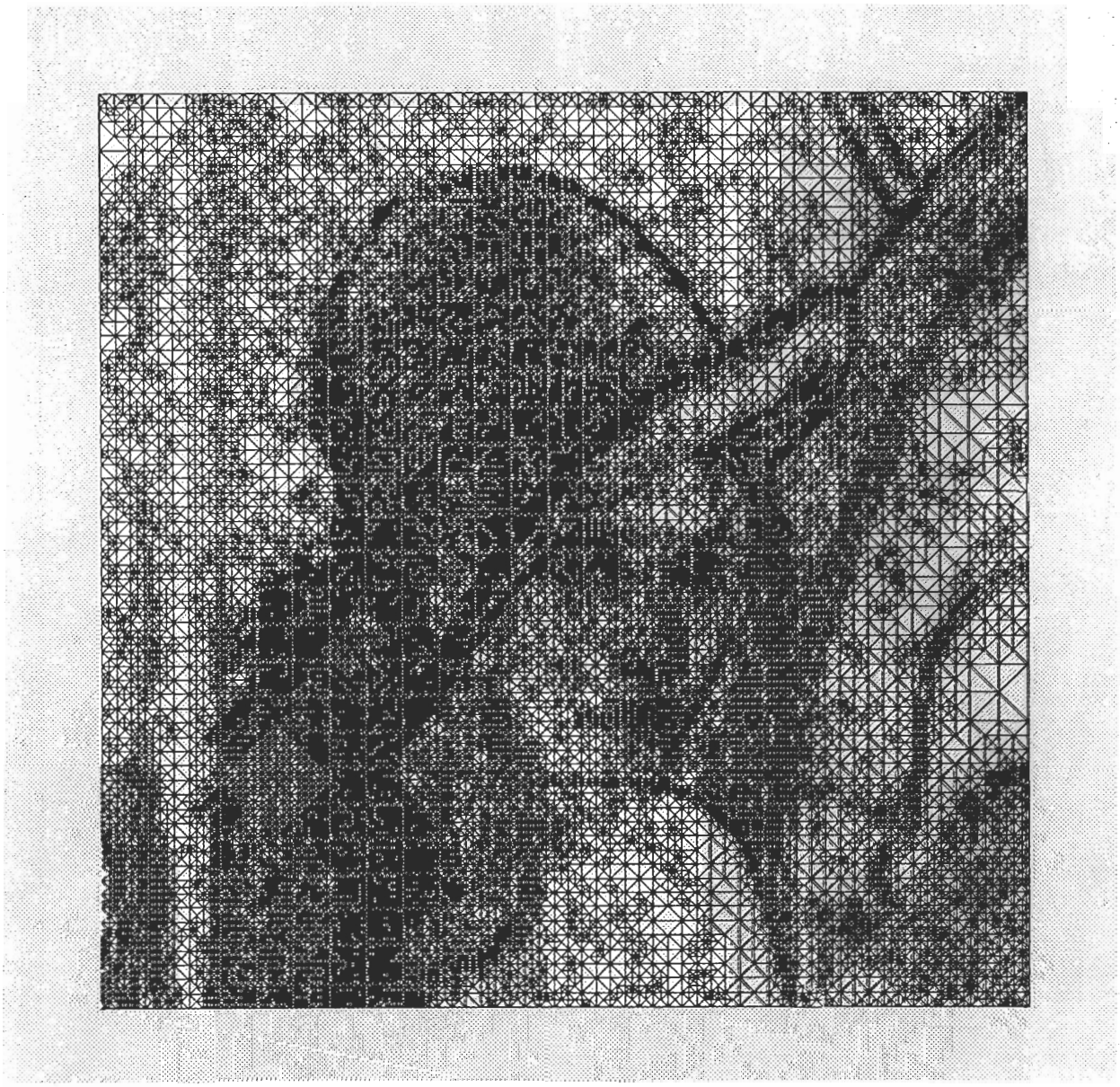


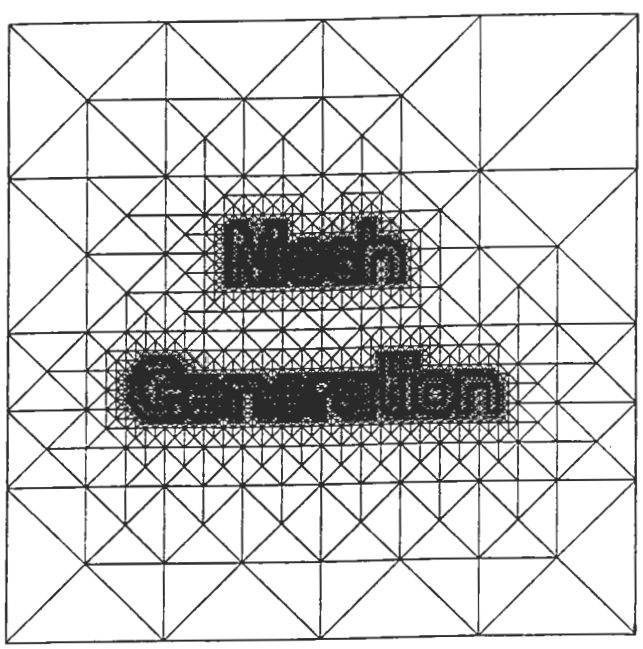
*Imagen y malla para $\varepsilon = 40$: 5.555 nodos, SNR(dB) = 19.95.*

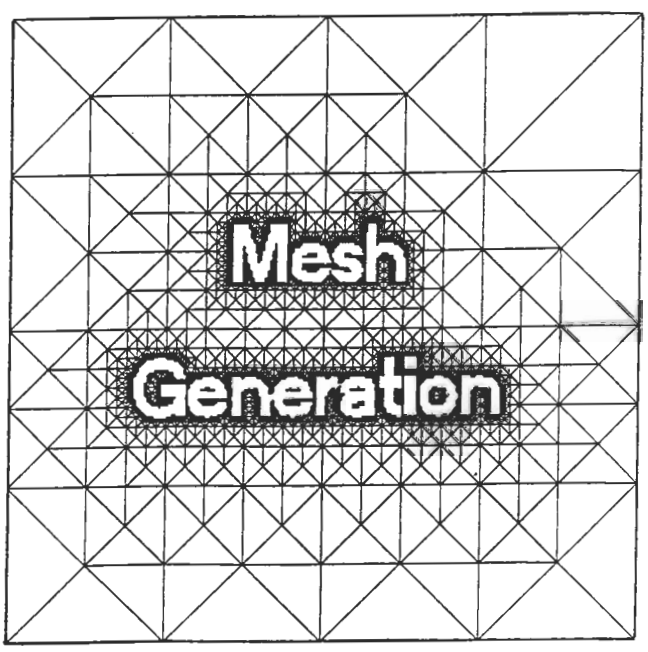




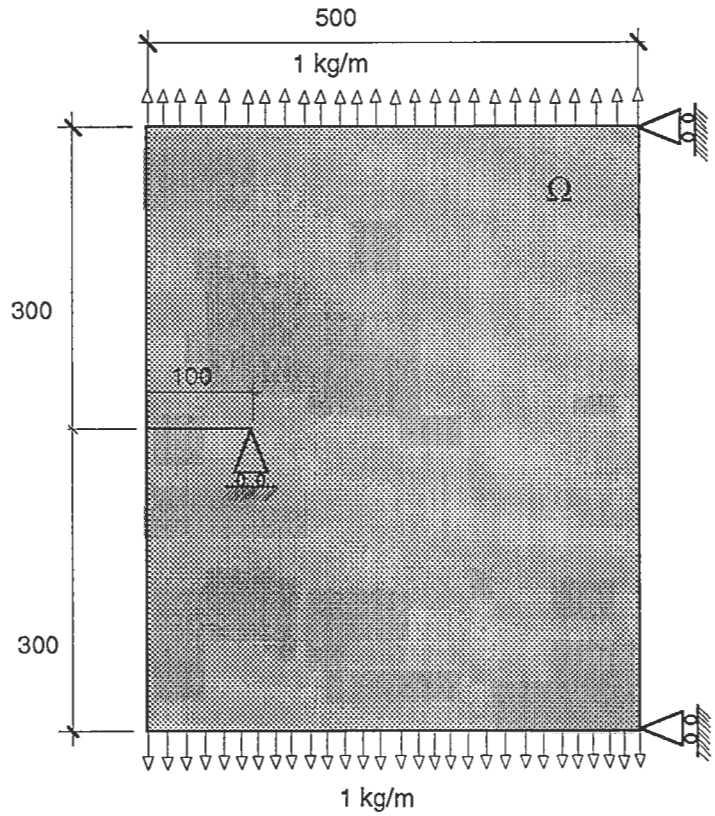


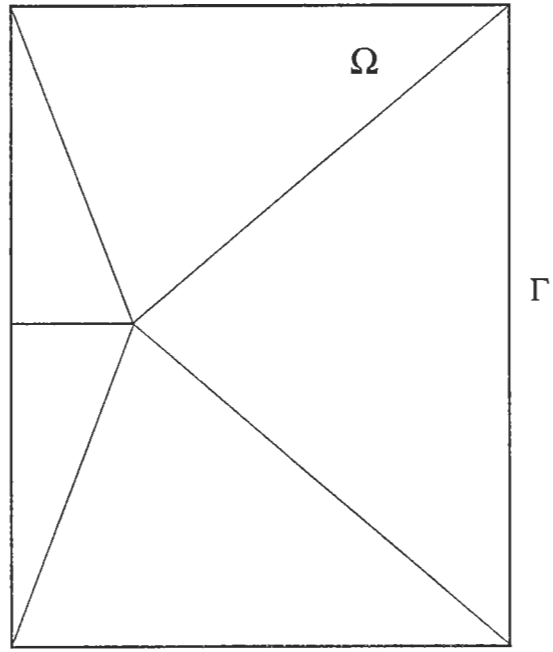
# Mesh Generation



# Un Problema de Elasticidad Lineal



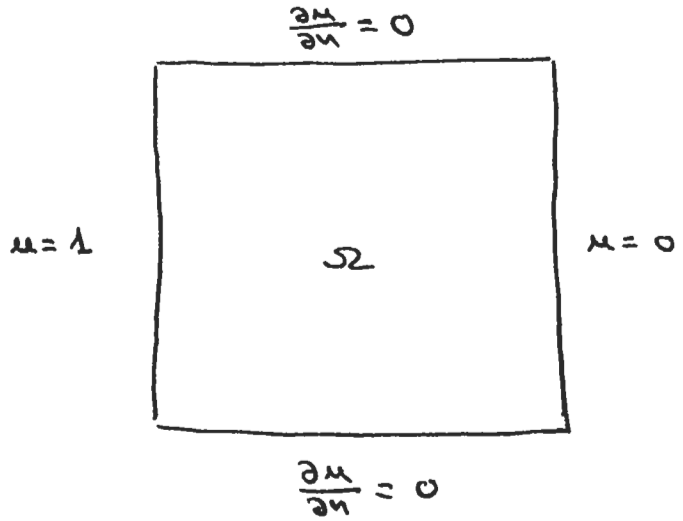



# HISTORIA PASADA

(TESIS - 1989)

Problema Evolutivo cuando solo se disponía del algoritmo de refinamiento

$$\frac{\partial u}{\partial t} + \vec{v} \cdot \nabla u - \nabla \cdot (\underbrace{K}_{1} \nabla u) = 0 \quad \text{en } \Omega$$



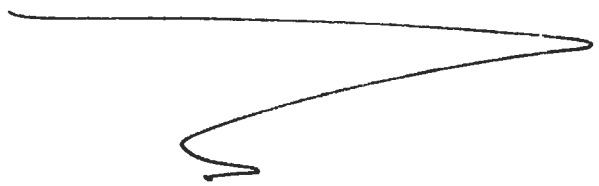

condición inicial : $u(\vec{x}, 0) = 0$

# HISTORIA NEPTUNO

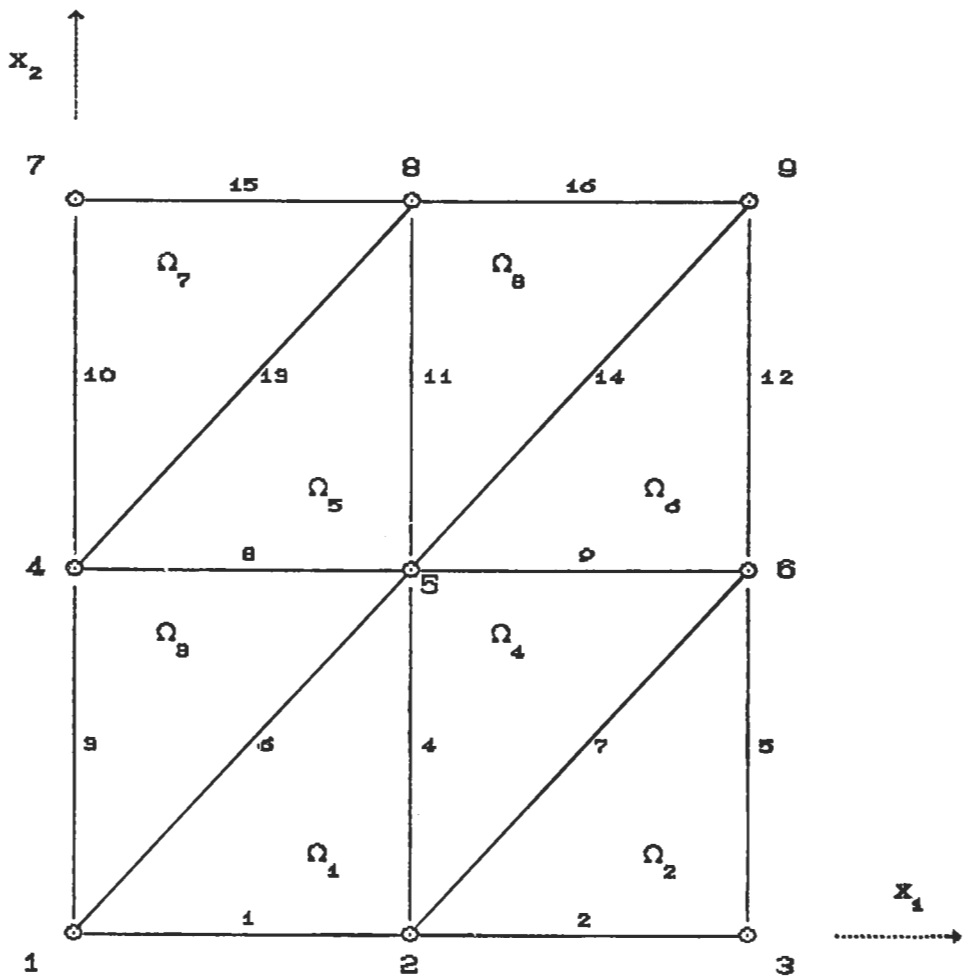
Generación de malla inicial

y

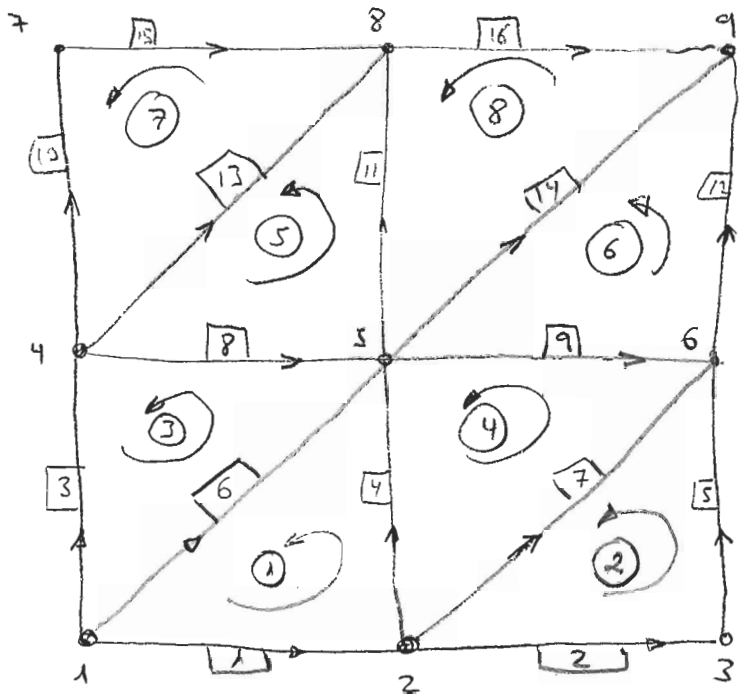
entrada de datos



*MALLADO DE "PARTIDA" ; Nodos, caras y elementos.*



NR = 1

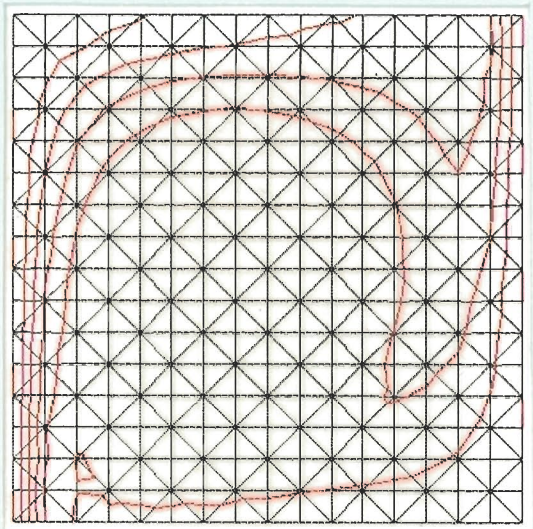



NR = 2

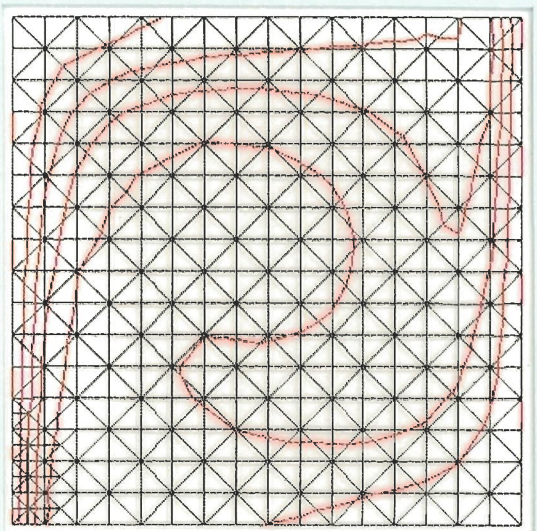
```
NEPT  *** CONVECCION-DIFUSION, CAMPO CIRCULAR, (A) ***
9,16,8,1,2,0,0,5,2,1,2,3,2,3,0,1
COOR
1,1,0.,0.
3,0,1.,0.
4,1,0.,0.5
6,0,1.,0.5
7,1,0.,1.
9,0,1.,1.

FACE
1,0,1,2,1
3,1,1,4,0
4,0,2,5,0
5,2,3,6,0
6,0,1,5,1
8,0,4,5,1
10,1,4,7,0
11,0,5,8,0
12,2,6,9,0
13,0,4,8,1
15,0,7,8,1

ELEM
1,1,1,4,-6,1
3,1,6,-8,-3,1
5,1,8,11,-13,1
7,1,13,-15,-10,1

RNOD
1,3,1
7,0,1
3,3,2
9,0,2

MATE
    1    4
3,0,0,0,4,5
1,0,0.

CCON
1,1
2,2

FUN2
C1-DIRICH-NR=1
1,1.
C2-DIRICH-NR=2
1,0.
C3-CONDUCT.-K
1,1.
C4-VELOC.X-V1
6,1000.
C5-VELOC.Y-V2
7,1000.

NOPR

END
MACR
```

```
TOL             1.E-6
LOOP            4.
NNIV
REFN
NEXT
LOOP            3.
NNIV
MATR
PREC
SEGM
GRCJ                    1.
IMPR            3.
INER                    3.
REFN                    0.4
NEXT
END
PLOT
POSP
6,0.,1.
STOP
```

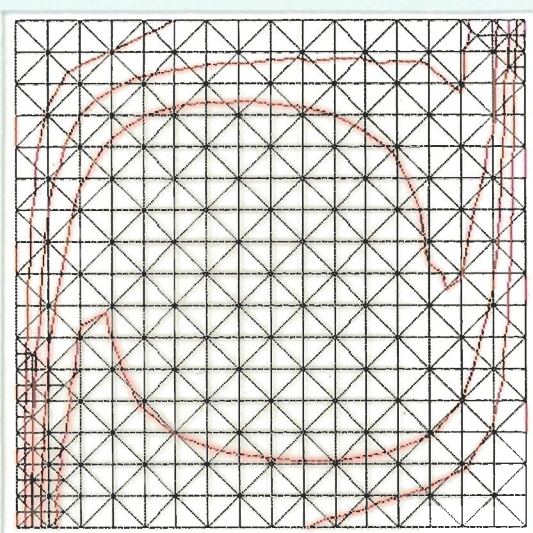
```
      SUBROUTINE ACONTR
C-----Subprograma de control de todo el proceso
C-----Lee el titulo,datos de control iniciales,inicializa la matriz
C-----de indicadores,llama al modulo de entrada de datos,al modulo de ejecucion,
C-----y a los modulos de preproceso y posproceso
C
C     O      :Controla el salto de pagina
C     HEAD   :Titulo
C     NUMNP  :No.de nodos (totales)
C     NUMNF  :No.de caras (totales)
C     NUMEL  :No.de elementos
C     NUMAT  :No.de materiales
C     NUREF  :No.de numeros de referencia
C     NUFN0  :No.de funciones dadas en forma implicita (Fronteras curvas)
C     NUFN1  :No.de funciones de un parametro
C     NUFN2  :No.de funciones de dos parametros
C     NDM    :Dimension del dominio de definicion del problema
C     NDF    :No.de grados de libertad por nodo
C     NDF1   :No.de grados de libertad por nodo variable acoplada
C     NEN    :No.de caras por elemento
C     NEC    :No.de nodos por cara
C     NEL    :No.de nodos por elemento
C     NIN    :No.de nodos internos por elemento
C     NEQ    :No.de ecuaciones
C     NSIM   :si = 0 -> m.simetrica ;si #0 -> m.no simetrica
C-----------------------------------------------------------------------
      COMMON M(1)
      COMMON/DIREC/IM(4,1)
      COMMON/MEMOR/IMAX,IMAXI,NBLOQ,IM1BL,IM2BL,IMEM,IPR
      COMMON /CDATA/ HO,HEAD(20),NUMNP,NUMNF,NUMEL,NUMAT,NUREF,
     1               NUFN0,NUFN1,NUFN2,NEN,NEC,NEL,NIN,NEQ
      COMMON /NDIM/ NDM,NDF,NDF1,NEN1,NEC1,NSIG
      COMMON /NULOG/ NL,NE
      COMMON /MSIM/ NSIML
      LOGICAL PCOMP,NSIML
      DIMENSION TITL(20),WD(8)
      DATA WD/4HNEPT,4HMACR,4HPLOT,4HPOSP,4HVELC,4HFLEC,4HDEFR,4HSTOP/
C
C---  Lee una tarjeta y se dirige al proceso correspondiente
C
```



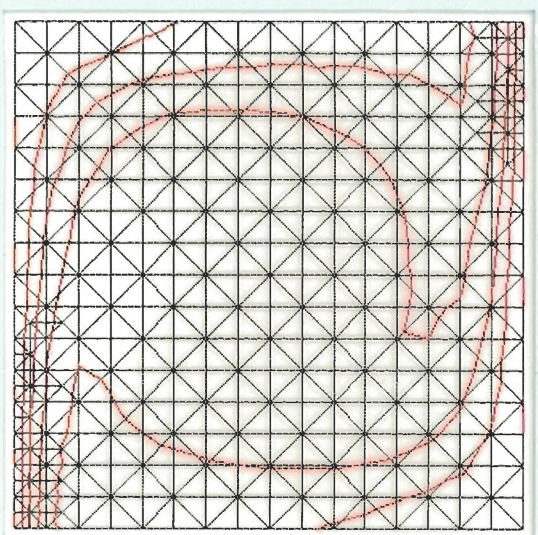
* MALLA 1 (Tiempo=0.05511) *



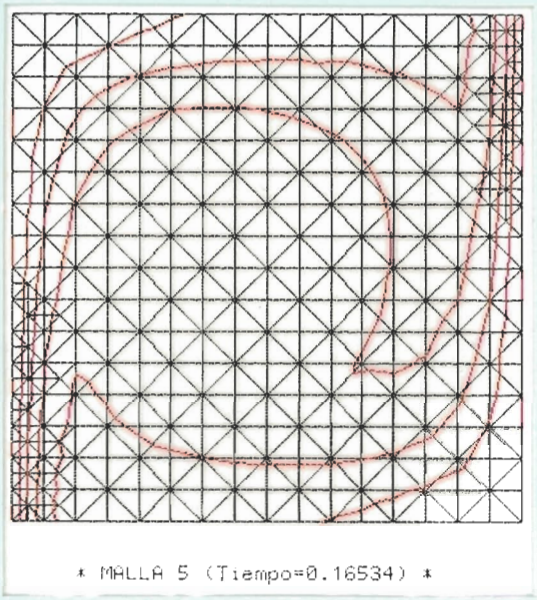
* MALLA 2 (Tiempo=0.08267) *



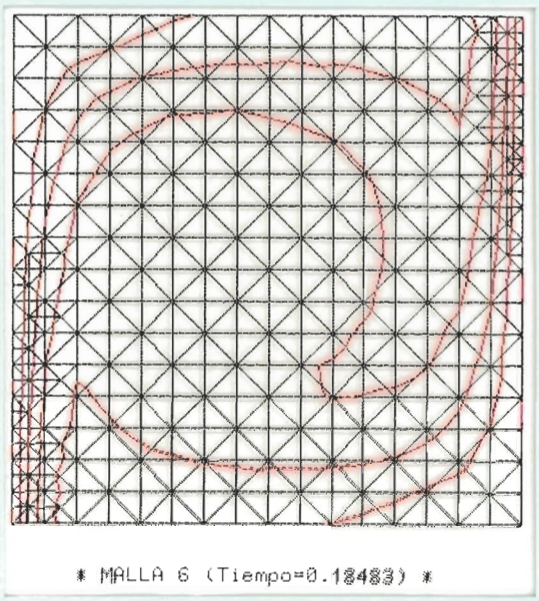
* MALLA 3 (Tiempo=0.11023) *



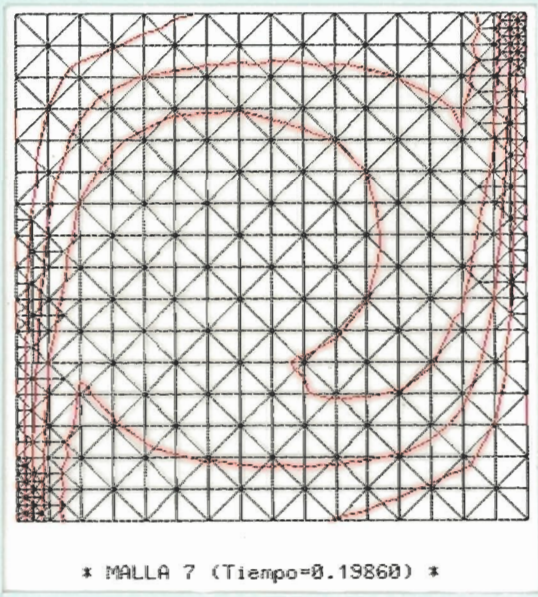
* MALLA 4 (Tiempo=0.13778) *



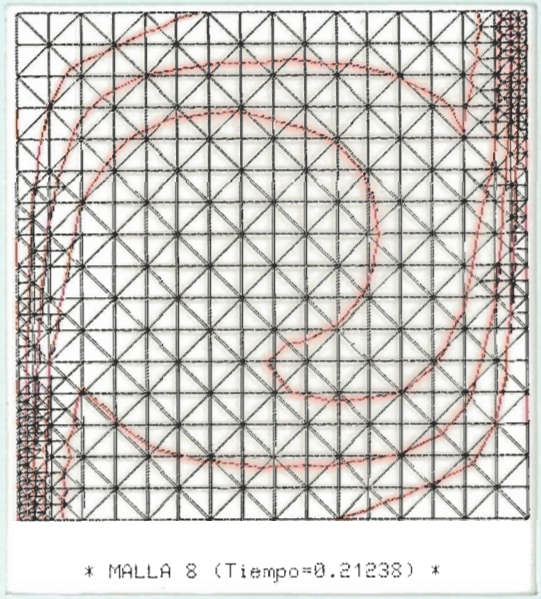
* MALLA 5 (Tiempo=0.16534) *



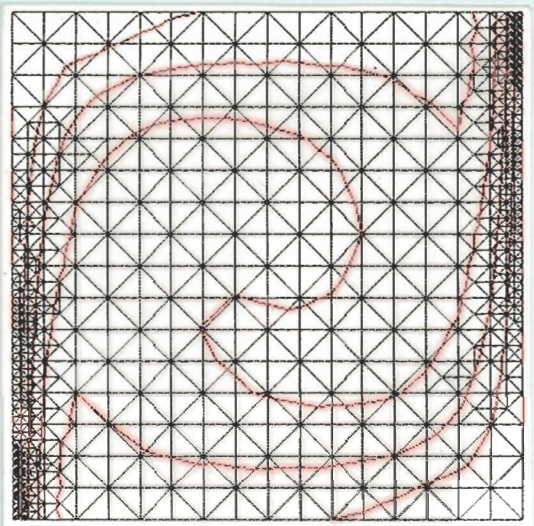
* MALLA 6 (Tiempo=0.18483) *



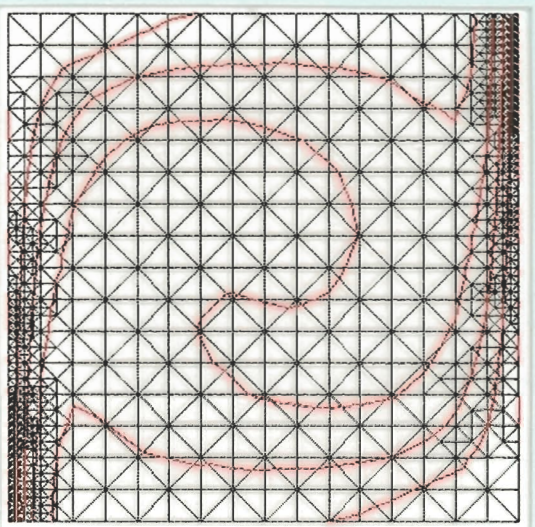
* MALLA 7 (Tiempo=0.19860) *



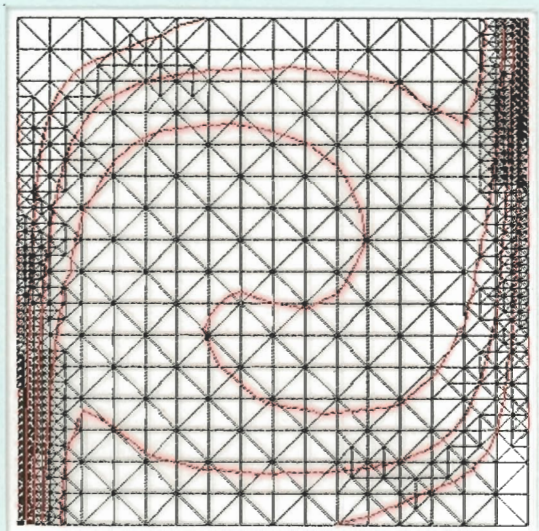
* MALLA 8 (Tiempo=0.21238) *



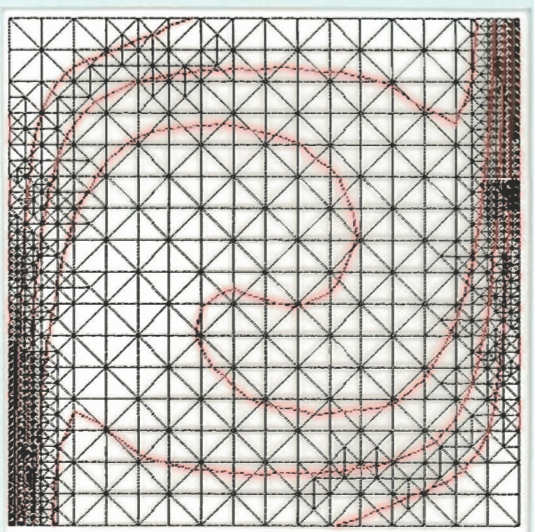
* MALLA 13 (Tiempo=0.26631) *



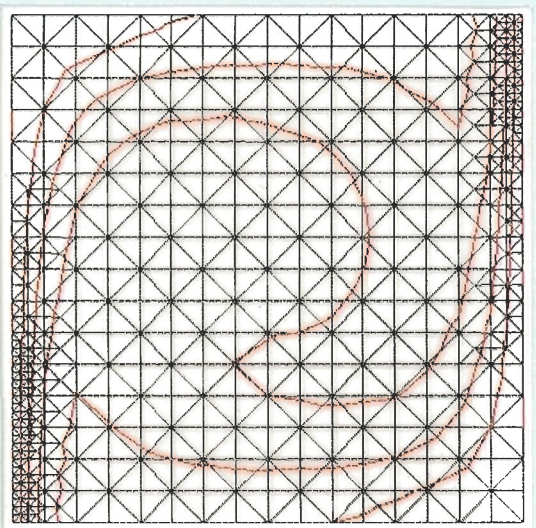
* MALLA 14 (Tiempo=0.27320) *



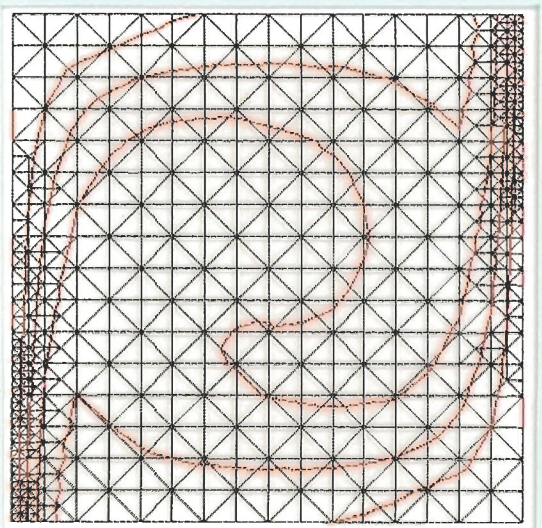
* MALLA 15 (Tiempo=0.28009) *



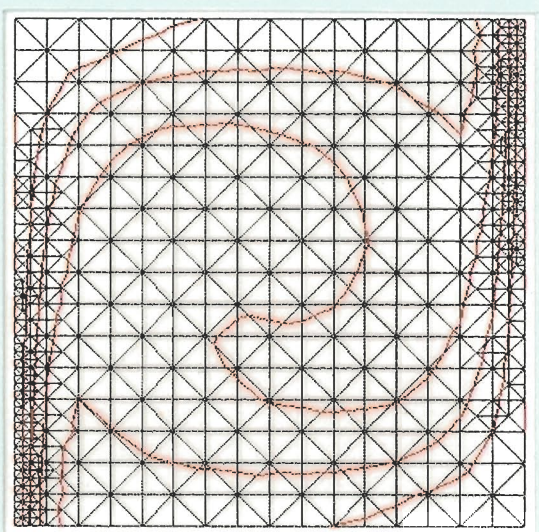
* MALLA 16 (Tiempo=0.28698) *



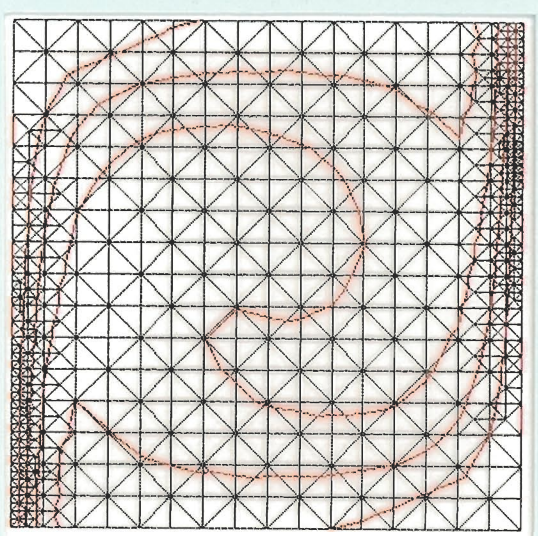
* MALLA 9 (Tiempo=0.22616) *



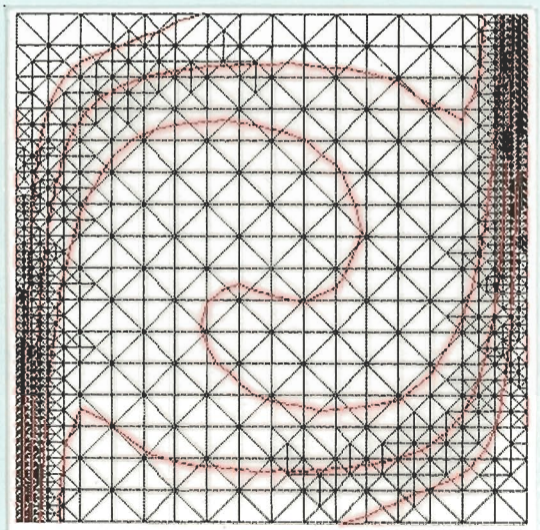
* MALLA 10 (Tiempo=0.23994) *



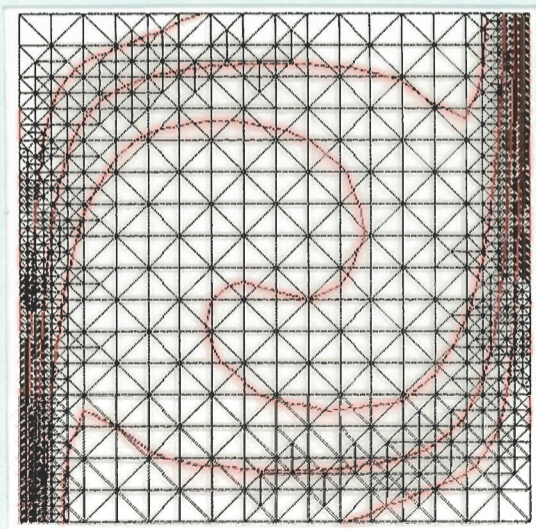
* MALLA 11 (Tiempo=0.24968) *



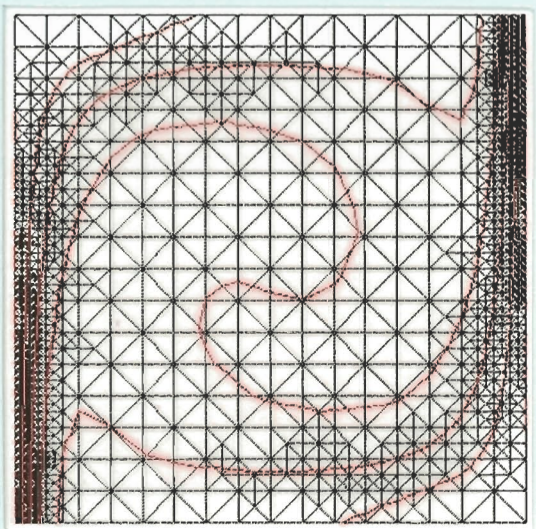
* MALLA 12 (Tiempo=0.25943) *



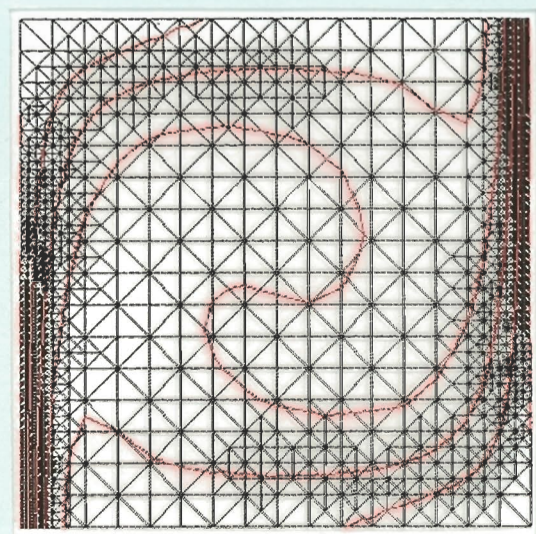
* MALLA 17 (Tiempo=0.29185) *



* MALLA 18 (Tiempo=0.29672) *



* MALLA 19 (Tiempo=0.30017) *



* MALLA 20 (Tiempo=0.30361) *