INTERPOLACION CON ESFERAS DE COBERTURA

TESIS

que para obtener el titulo de
DOCTOR EN CIENCIAS

presenta

LEONARDO TRAVERSONI DOMINGUEZ

DIVISION DE CIENCIAS BASICAS E INGENIERIA
UNIVERSIDAD AUTONOMA METROPOLITANA
IZTAPALAPA
MEXICO D.F.
MEXICO

1 Dedicatoria

A mi padre

2 Agradecimientos

Al Dr Luis Verde por su apoyo, sus consejos, correcciones y paciencia asi como por varias orientaciones e ideas en demostraciones dificiles. Al Dr Oscar Palacios quien me mostró hace ya bastante tiempo la utilidad y aplicaciones de los trabajos de Voronoi y Delaunay y que además me acercó una copia de un trabajo de Delaunay poco conocida y sin embargo muy útil; a él debo agradecer además parte del software que he usado. A los Dres. Octavio Arzate y Richard Wilson por las oportunas observaciones que realizaron sobre las sucesivas versiones de este trabajo. A los miembros del tribunal en general por sus orientaciones. Al Dr Gerald Farin que me hizo notar la posibilidad de que existiera algún nexo entre las esferas de cobertura y el interpolante de Sibson y me facilitó copias de los trabajos de Sibson. Al Dr Larry Schumacker por sus observaciones respecto al número de operaciones del algoritmo de construcción de las esferas. Al Dr Carl de Boor por sus observaciones respecto a la aplicación a Splines. Al Dr Gustavo Fuentes por el apoyo institucional que me brindó.

Interpolación con esferas de cobertura

Leonardo Traversoni Dominguez

Contenido

Capítulo 1

Antecedentes

Existen en la actualidad varias disciplinas que tienen numerosos puntos en común y que giran alrededor de la visualización de curvas y superficies, su aproximación y reconstrucción, así como el reconocimiento de formas y siluetas etc. Entre estas disciplinas se encuentran la teoría de aproximación y la geometría computacional, entre las más importantes.

El tema del presente trabajo tiene nexos con estas disciplinas y sus raíces históricas se podrían ubicar en un contexto puramente geométrico con los trabajos de G. Voronoi a principios de siglo y las aplicaciones prácticas que desarrolló Thiessen en Geografía, pero muy particularmente en un trabajo de B. Delaunay [13] de 1934, que, si bien no es actualmente muy conocido, de alguna manera constituyó un antecedente a partir del cual se ha comenzado a hablar de los triángulos de Delaunay y sus construcciones duales. Por ejemplo, en el Science Citation Index de 1991 se pueden encontrar actualmente más de 80 referencias de artículos que usan esta construcción o la mencionan en el título. Esta eclosión se ha debido en gran parte a la multitud de aplicaciones en geografía, cristalografía y biología que esta construcción tiene. Algo tan usable no tardó en requerir su automatiza ción, y de esta manera han proliferado los algoritmos computacionales para conseguir realizar la construcción automáticamente.

Encontrar los vecinos más cercanos a un punto, elegidos en un conjunto dado de puntos, problema que equivale a realizar la partición de Voronoi, se

convirtió así en un problema de Geometría Computacional de primer orden. Lo mismo sucedió con su construcción dual; los triángulos de Delaunay, y con varias otras construcciones afines, por ejemplo teselizaciones con pesos o penalizadas. Por otro lado, la utilización de estas construcciones en problemas de geociencias de inmediato las constituyó en bases naturales para la interpolación y el método de elemento finito, convirtiéndolas en un problema de teoría de aproximación.

En este trabajo se propone una nueva construcción, las *Esferas de Cobertura*, relacionada con las arriba mencionadas, y se muestra su utilidad y sus propiedades matemáticas.

La primera presentación del concepto se hace en el ámbito de la Geometría Computacional, mostrando el cáracter dual de esta construcción con respecto a la partición de Voronoi. Asimismo, se presenta un algoritmo para realizar computacionalmente esta construcción, el cual puede ser usado inclusive como "atajo" para realizar sus construcciones duales más rápidamente y con menor uso de capacidad de almacenamiento.

Para introducir el concepto a la teoría de aproximación se muestra su relación con el interpolante de Sibson o de vecinos naturales. Como consecuencia directa se presenta un algoritmo para combinar todo lo anterior con el fin de obtener un método de interpolación con interesantes ventajas.

Por último, dado que las esferas de cobertura son fácilmente generalizables a cualquier dimensión, se dan algunas ideas de aplicaciones en dimensiones 3 y 4.

1.1 Los problemas básicos

Cuando se habla del uso de interpolación para la reconstrucción de superficies, la referencia incluye un amplísimo campo generado a partir de necesidades técnicas de la industria y las ciencias aplicadas.

Por esta razón, muchas veces se tienen varias soluciones para un mismo problema, debido a que han sido resueltas sobre la marcha por ingenieros dedicados a algún problema en particular. Por otra parte, la vertiginosa velocidad con la que se va avanzando en la disciplina, propia de una técnica a la que sirve y no tanto de una ciencia, ha provocado dificultades de comunicación, las que han hecho que existan muchos desarrollos independientes, poco diferentes en sus inicios, pero que se han ido diferenciando cada vez

más por razones técnicas.

Sin embargo, han habido constantes y notables esfuerzos de sistematización de todas las áreas que componen la disciplina, que han sido llevados a cabo a veces por los propios ingenieros creadores de las técnicas, como Bèzier, y otras veces por matemáticos interesados en el tema y que ya tenían una óptica mucho más amplia del problema de la interpolación.

Los problemas que clásicamente se han abordado son:

a) La construcción de superficies topográficas o similares, cuya principal característica es la de que exista una relación uno a uno entre los puntos (x, y) del dominio y los puntos (x, y, z) de la superficie.

b) La construcción de superficies cerradas o semicerradas para representar carrocerías o fuselajes, que deben además tener algo que ver con los métodos de cálculos de resistencias a llevar a cabo con dichas superficies, los cuales generalmente usan métodos de elemento finito y pretenden usar al menos la misma discretización; la llamada "malla de alambre" o "wire frame".

c) La construcción de superficies muy complejas, como las que se usan en medicina, como resultado, por ejemplo, de las tomografías computarizadas o de técnicas similares.

d) La construcción de hipersuperficies, como las que describen propiedades, por ejemplo, el calentamiento en una región tridimensional.

e) La representación bidimensional y su interpretación (en la pantalla de un monitor) de las superficies anteriormente mencionadas.

f) La interacción del usuario con el modelo usando la representación que ve en el monitor y modificándola.

1.2 El problema de la discretización del dominio

Para cualquiera de los casos anteriores, y para aplicar cualquier técnica de interpolación, se debe tener una base sobre la cual aplicarla, ya que se usarán técnicas locales de interpolación y no globales, las cuales, por otra parte, tendrían muy poca aplicación para los problemas que se están abordando.

Así, el primer problema es como subdividir el dominio de definición en partes en las que se va a aplicar el método, y luego como asegurar la continuidad entre los resultados locales.

Se pueden distinguir inmediatamente dos formas, la primera usando mallas, generalmente regulares, que son independientes de la localización de puntos con datos. Este tipo de técnicas requieren en realidad una doble interpolación para asignarle valores a los nodos de la malla. Son, sin embargo, las más comunmente usadas por razones que se verán más adelante.

La segunda es crear una malla, generalmente irregular, que tenga como nodos los puntos con datos. De esta técnica es de la que trata este trabajo.

Una vez realizada la opción anterior queda el problema de determinar el criterio para realizar la discretización irregular, obviamente debe cumplir algunos preceptos básicos :

1) Los nodos con datos deben estar en los bordes de las zonas o elementos.

2) Las zonas no deben contener puntos de la malla en su interior, es decir, que no puede haber puntos del conjunto que no formen parte de la malla.

3) La construcción debe ser única, es decir, que para un conjunto dado de puntos debe obtenerse una sola discretización con el método elegido.

La mayoría de los autores considera una cuarta propiedad, que en este caso deliberadamente será omitida, que es que las zonas no se superpongan.

Debe recordarse además que la finalidad de todo ésto es que, dado un punto del dominio del cual no se tienen datos, generarlos, considerando los datos de los puntos donde sí se los tiene.

Debido a que se trata de usar consideraciones locales y no globales, otro problema que se debe resolver, preferiblemente de manera simultánea es el de localizar donde está un punto con relación a la malla, ya que se trata de realizar la construcción de la misma en forma incremental, lo que permite agregar y quitar puntos de la malla en caso de que sea necesario adaptarla según las circunstancias.

Para conseguir todo lo anterior se ha optado por usar la construcción debida a G. Voronoi, que se denomina Teselización o partición de Voronoi. Para fijar ideas llamaremos V a un conjunto de m puntos distintos en el espacio $\mathbf{E}$ de n dimensiones, usando estos criterios $\mathbf{E}$ queda totalmente

subdividido, de tal forma que cualquier punto de $\mathbf{E}$ pertenece a alguna de las divisiones de la teselización, lo cual es ideal para localizar puntos.

El siguiente paso lógico en la construcción de la discretización es relacionar a los puntos de V de acuerdo a la construcción anterior, formando figuras que contengan a los puntos que equidistan de al menos un punto de $\mathbf{E}$, los que se llamarán "vecinos de Voronoi". Del tipo de figuras a que ésto da origen se hablará en el próximo capítulo para no perder generalidad ahora.

1.3 Las superficies generadas por interpolación

Se describen a continuación muy brevemente los métodos más comunmente usados para interpolar.

1.3.1 Productos tensoriales de Bèzier

La base para la generación de superficies por este método es la interpolación bilineal, para crearlo se han aprovechado todas las posibles analogías con la interpolación lineal aplicada a la construcción de curvas, teoría que fue desarrollada previamente por el mismo Bèzier.

Podemos encontrar algunos antecedentes en el trabajo de Ferguson [21] y más reciente en el de L. Nachman [23].

Así como la interpolación lineal describe la curva más sencilla entre dos puntos dados, la interpolación bilineal describe la superficie más sencilla entre cuatro puntos dados en el espacio euclideano tridimensional $\mathbf{E}$.

Para fijar ideas considérense los puntos $P_{0,0}, P_{0,1}, P_{1,0}, P_{1,1}$ pertenecientes al citado espacio, todos ellos distintos entre sí, y considérese al paraboloide hiperbólico que pasa por los cuatro puntos, expresado como el conjunto de puntos x en $\mathbf{E}$ tales que su expresión en forma paramétrica es

$$x(u,v) = \sum_{i=0}^{1}\sum_{j=0}^{1} \mathbf{P}_{i,j} B_i^1(u)\ B_j^1(v),$$

donde los B_i son polinomios de Bernstein de la forma general y los $P_{i,j}$ son los valores de la variable a interpolar en el punto $P_{i,j}$, más propiamente se debiera decir que $\mathbf{P}_{i,j} = f(P_{i,j})$ pero se ha preferido mantener la notación usada por la mayoría de los autores al respecto.

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i},$$

o en forma matricial

$$x(u,v) = (1-u \quad u) \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}.$$

Tanto en u como en v el interpolante es lineal, como se ve, y la superficie x puede ser llamada un interpolante bilineal.

Este interpolante bilineal puede ser visto como un mapeo o correspondencia del cuadrado unidad en el plano u, v, de esta forma, el citado cuadrado sería el dominio del interpolante, mientras que la superficie x es la gráfica. Obsérvese que a una línea paralela a uno de los ejes en el dominio le corresponde una curva en la superficie, la cual es isoparamétrica, y en el caso presente es una recta, dado que los paraboloides hiperbólicos son superficies regladas.

Otra forma de evaluar el interpolante bilineal, debida a De Casteljau [6] puede describirse en dos pasos como sigue.

Tómense dos puntos intermedios

$$P_{0,0}^{0,1} = (1-v)P_{0,0} + vP_{0,1}$$

$$P_{1,0}^{0,1} = (1-v)P_{1,0} + vP_{1,1}$$

obteniéndose

$$x(u,v) = P_{0,0}^{1,1}(u,v) = (1-u)P_{0,0}^{0,1} + uP_{1,0}^{0,1}.$$

Así como las curvas de Bèzier se pueden obtener por interpolación lineal repetida, también se pueden obtener superficies por interpolación bilineal repetida.

Tómense ahora un conjunto de puntos $P_{i,j}$ pertenecientes a un arreglo rectangular tal que $0 < i, j < n$ y unos parámetros (u, v) en $\mathbf{R}^2$. El algoritmo de De Casteljau quedaría

$$P = (1-u \quad u) \begin{pmatrix} P_{i,j}^{r-1,r-1} & P_{i,j+1}^{r-1,r-1} \\ P_{i+1,j}^{r-1,r-1} & P_{i+1,j+1}^{r-1,r-1} \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}$$

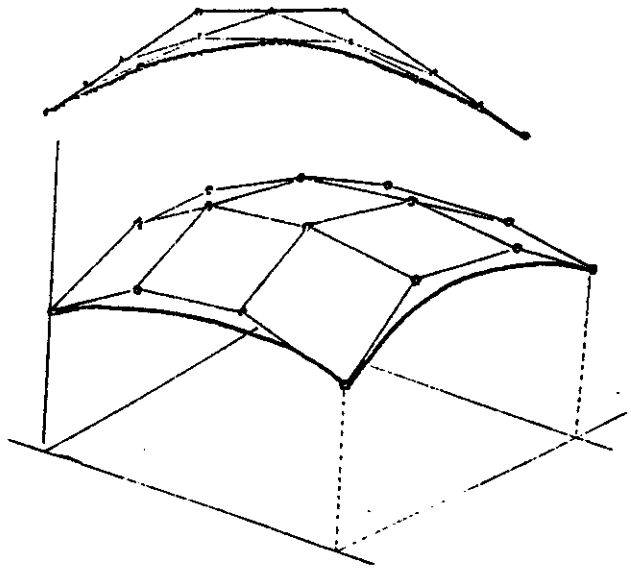

Figura 1

$$r = 1, 2, \ldots, n \qquad i, j = 0, 1, \ldots, n - r.$$

Allí la red de puntos $P_{i,j}$ es la red de control, y lo que se obtiene es una superficie de Bèzier.

En la primer figura se ve como se construye una curva y en la segunda la misma técnica es aplicada para construir una superficie.

Para ésto se puede definir una superficie en el espacio como el lugar de los puntos formados por la trayectoria de una curva que se traslada moviéndose en el espacio.

Para ejemplificarlo considérese primeramente una curva de Bèzier de grado m.

$$P^m(u) = \sum_{i=0}^{m} \mathbf{P}_i B_i^m(u).$$

Ahora considérese que en cada punto P_i la curva original es atravesada por otra curva de Bèzier de grado n

$$P_i = P_i(v) = \sum_{j=0}^{n} \mathbf{P}_{i,j} B_j^n(v).$$

Combinando ambas ecuaciones se obtiene un punto de la superficie

$$P^{m,n}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{P}_{i,j} B_i^m(u) B_j^n(v).$$

Como se ve, por este método se obtiene el mismo resultado que con el método anterior, y las curvas son curvas isoparamétricas que pueden ser obtenidas usando el mismo algoritmo de De Casteljau que se usa para construir curvas en forma iterativa.

1.3.2 Superficies compuestas y Splines

En 1962 De Boor [4] publicó uno de los primeros artículos referentes a Splines bicúbicos, aunque otros autores en forma independiente lo hicieron casi simultáneamente.

A partir de los trabajos iniciales mucho se ha hecho y escrito al respecto, destacaremos el trabajo de W. Bohem [3], y el de Bartels, Beatty y Barsky [1].

La idea consiste básicamente en aplicar lo visto en el inciso anterior, y la teoría de Splines para curvas, a pedazos de superficie.

El primer problema que un planteamiento así presenta es el de la continuidad en las fronteras entre uno y otro pedazo. El problema que para una curva se reduce a solamente asegurar la continuidad en un punto, ahora es el de lograr la continuidad sobre toda la línea de frontera, con una especial dificultad cuando se trata de los vértices en los que concurren varias de estas fronteras, de modo tal que hay que preocuparse de algo que podría llamarse la torsión en dicho punto, para "pegar" los pedazos concurrentes a él.

Sean ahora, por ejemplo, dos pedazos representados por sus vectores normales. La condición que se plantea para que sean continuas en la frontera común las derivadas hasta la de orden n es

$$\frac{\mathrm{d}^n}{\mathrm{d}u^n} x(u,v) = \frac{\mathrm{d}^n}{\mathrm{d}u^n} y(u,v)$$

donde $x(u,v)$ y $y(u,v)$ son las parametrizaciones de las superficies de los dos pedazos y la antecitada relación se pretende que se cumpla en la curva frontera común de ambas.

Como cada pedazo de superficie está definido por una expresión de Bèzier, entonces la condición arriba mencionada solo se puede cumplir si todas las líneas de la malla que atraviesan a ambos lados de la curva común pueden ser consideradas como polígonos de control de Bèzier de orden n. Obsérvese que les llamamos así por analogía con las poligonales de control que se usan para construir curvas.

Obsérvese que, de esta forma, un problema originalmente planteado para superficies se presenta como un problema reiterado de curvas, en el cual se puede aplicar toda la teoría de Splines para curvas; la cual históricamente fue desarrollada antes.

Para simplificar los cálculos se tratará de obtener superficies interpolantes rectangulares con base a lo visto en la sección 1.2.1, es decir, aproximar la superficie con una superficie facetada cuyas facetas son rectángulos, sea así

$$x(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} c_{i,j} A_i(u) B_j(v)$$

una superficie expresada como producto tensorial, donde A y B son polinomios de Bernstein. Escribiendo la ecuación anterior en forma matricial queda

$$x(u,v) = [A_0(u) \ \dots \ A_m(u)] \begin{pmatrix} c_{00} & \dots & c_{0n} \\ c_{10} & \dots & c_{1n} \\ \vdots & \dots & \vdots \\ c_{m0} & \dots & c_{mn} \end{pmatrix} \begin{pmatrix} B_0(v) \\ B_1(v) \\ \vdots \\ B_n(v) \end{pmatrix}.$$

Si se tiene un arreglo $(m+1) \times (n+1)$ de puntos $x_{i,j}$ y en ellos se va a usar la superficie interpolante, entonces se obtienen $(n+1)(m+1)$ ecuaciones de la forma $X = ACB$ donde

$$X = \begin{pmatrix} x_{00} & \dots & x_{0n} \\ x_{10} & \dots & x_{1n} \\ \vdots & \dots & \vdots \\ x_{m0} & \dots & x_{mn} \end{pmatrix}, \quad A = \begin{pmatrix} A_0(u_0) & \dots & A_0(u_m) \\ \vdots & \dots & \vdots \\ A_m(u_0) & \dots & A_m(u_m) \end{pmatrix}$$

$$C = \begin{pmatrix} c_{00} & \dots & c_{0n} \\ \vdots & \dots & \vdots \\ c_{m0} & \dots & c_{mn} \end{pmatrix}, \quad B = \begin{pmatrix} B_0(v_0) & \dots & B_0(v_n) \\ \vdots & \dots & \vdots \\ B_n(v_0) & \dots & B_n(v_n) \end{pmatrix}.$$

Las matrices A y B son matrices de Vandermonde.

Como los $x_{i,j}$ son conocidos y los $c_{i,j}$ son las incógnitas, la interpolación debe resolver el sistema

$$C = A^{-1}XB^{-1}.$$

En esta forma el sistema es difícil de resolver, por lo que se escribe en la forma

$$X = DB,$$

donde $D = AC$.

Esta última se puede interpretar como una interpolación univariada, mientras que todo el proceso consiste en repetir interpolaciones univariadas sucesivas.

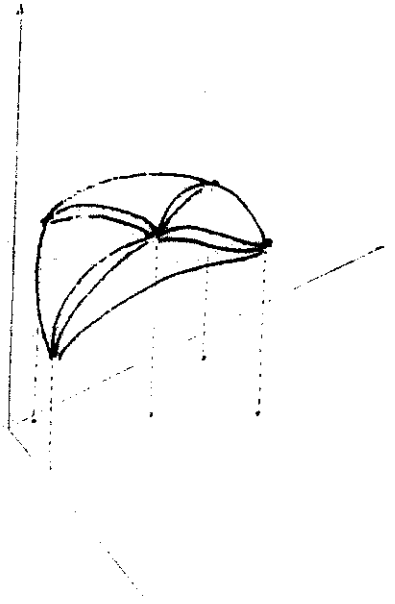

Fig. 2
Pegado de pedazos

Se ha exagerado el efecto a fin de mostrar gráficamente el problema de pegar trozos de superficie que deben pasar por puntos dados y tener una curvatura preestablecida, y hacerlo manteniendo la continuidad.

1.3.3 Triángulos de Bèzier

Este caso es el que representa mayor interés para el presente trabajo debido a su similitud con la teoría cuyo desarrollo es motivo del mismo.

Los primeros trabajos al respecto son de Bèzier [2]. Se puede citar asimismo el trabajo de T. De Rose y T. Hollman [14] y el de Farin [16] y [17].

Para comenzar se definirá un concepto importante, el de las coordenadas baricéntricas.

Sean A, B y C tres puntos en el plano que determinan un triángulo. Las coordenadas de un punto P cualquiera del plano en forma baricéntrica respecto a ABC son :

$$u = \frac{\text{Area}(ABP)}{\text{Area}(ABC)}$$

$$v = \frac{\text{Area}(BCP)}{\text{Area}(ABC)}$$

$$w = \frac{\text{Area}(CAP)}{\text{Area}(ABC)}.$$

Se pueden clasificar los puntos del plano, (eceptuando vértices y aristas del triángulo) mediante sus coordenadas baricéntricas; así los puntos que tienen todas sus coordenadas del mismo signo están "dentro" del triángulo y las que son tales que una de ellas es de distinto signo están "fuera" del mismo. En el primer caso las coordenadas suman 1 cosa que no sucede en el segundo.

Otra propiedad importante es que, cuando P está dentro del triángulo las tres áreas suman la del triángulo, por lo que las coordenadas son siempre iguales o menores que 1. Además se cumple la ecuación vectorial

$$P = uA + vB + wC.$$

Para usar estas coordenadas en interpolación sea ahora un punto P del plano en el interior del triángulo, y sea f una función definida sobre los vértices, entonces podemos asociar a P un valor de f interpolado como sigue

$$f(P) = uf(A) + vf(B) + wf(C).$$

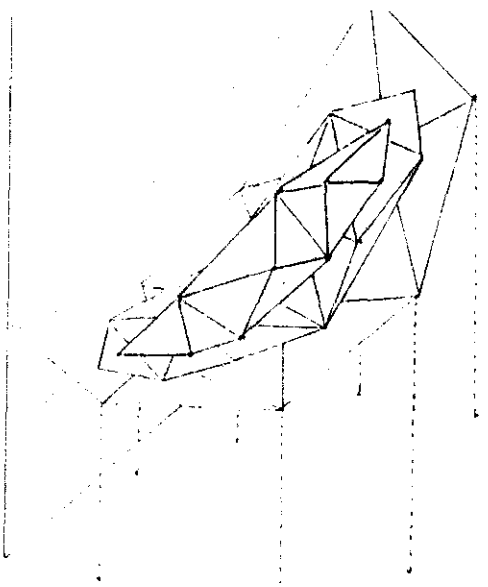

Fig. 3
Triángulos de Bèzier
Se ven tres "capas" de triángulos, la primera genera
con sus baricentros a la siguiente y así sucesivamente.

Si se considera ahora un arreglo triangular de puntos en el plano, y se asocia a cada uno de ellos una tercera coordenada dada por el valor de una función f como más arriba, de modo tal que los puntos correspondientes en el espacio formen una superficie facetada, cuyas facetas serán triángulos, se puede generar con el mecanismo de interpolación de arriba un algoritmo de De Casteljau similar a los que se utilizan en la construcción de curvas.

$$P_i(\mathbf{u}) = uP_{i+e_1}^{n-1} + vP_{i+e_2}^{n-1} + wP_{i+e_3}^{n-1}$$

Con $n = 1, 2, \ldots, N$ y $i = N - n$ donde e_1, e_2, y e_3 son los vectores $(1,0,0), (0,1,0)$ y $(0,0,1)$ respectivamente.

En la figura siguiente se muestra la secuencia de construcción de un punto de la superficie con el algoritmo.

Para expresar el algoritmo también se pueden adaptar los polinomios de Bernstein. Sea uno de ellos

$$B_{\mathbf{i}}^n = \binom{n}{\mathbf{i}} u^i v^j w^k = \frac{n!}{i!j!k!} u^i v^j w^k$$

donde $\mathbf{i} = (i, j, k)$ es un vector tal que $|\ \mathbf{i}\ | = n$.

Es válida la siguiente fórmula recursiva

$$B^n_{\mathbf{iP}}(u, v, w) = B^n_{\mathbf{i}}(\mathbf{u}) = uB_{\mathbf{i}-e_1(\mathbf{u})} + vB_{\mathbf{i}-e_2(\mathbf{u})} + wB_{\mathbf{i}-e_3(\mathbf{u})}.$$

Un punto cualquiera de uno de los triángulos puede ser escrito de la forma

$$P(\mathbf{u}) = \sum P_{\mathbf{i}} B^n_{\mathbf{i}}(\mathbf{u}).$$

1.3.4 Coons

Coons [11] publicó para el M.I.T. un primer trabajo referente a esta construcción, que posteriormente adoptaría su nombre, la idea es como sigue.

Supóngase que se tienen dos curvas en el espacio de las que se supone que pertenecen a una superficie que se quiere obtener por interpolación.

Lo anteriormente descrito puede muy bien ser hecho mediante interpolación lineal tomando pares de puntos uno de cada curva e interpolando entre ellos linealmente, por ejemplo, sean las curvas $C_1(u)$ y $C_2(u)$, entonces una solución al problema de hallar un punto X de la superficie entre ellas puede ser

$$X(u, v) = (1 - v)C_1(u) + vC_2(u).$$

Un pedazo rectangular de superficie en el espacio es, sin embargo, tal que generalmente se conocen cuatro y no solo dos curvas; de esta manera es ideal usar interpolación bilineal.

Sean las curvas

$$X(u, 0) = C_1(u), \qquad X(u, 1) = C_2(u),$$

$$X(0, v) = D_1(v), \qquad X(1, v) = D_2(v).$$

Éstas dan lugar a dos superficies regladas R_c y R_d dadas por

$$R_c(u, v) = (1 - v)X(u, 0) + vX(u, 1)$$

y

$$R_d(u, v) = (1 - u)X(0, v) + uX(1, v).$$

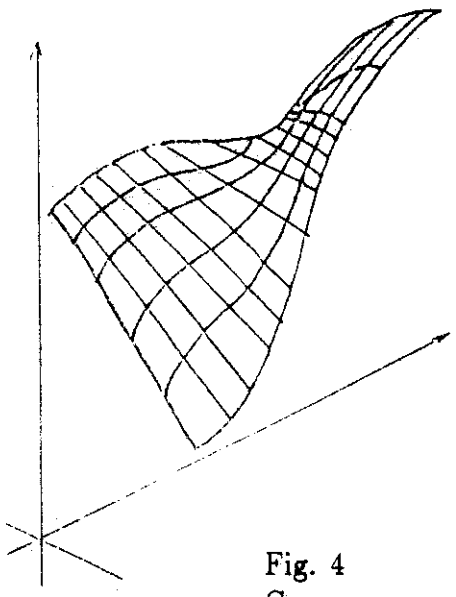

Fig. 4
Coons

Usando interpolación bilineal obtenemos

$$R_{cd} = [1-u \;\; u] \begin{pmatrix} X(0,0) & X(0,1) \\ X(1,0) & X(1,1) \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}.$$

Un "pedazo" queda definido según Coons como

$$X = R_c + R_d - R_{cd}.$$

En forma matricial

$$X(u,v) = [1-u \;\; u] \begin{pmatrix} X(0,v) \\ X(1,v) \end{pmatrix} + [X(u,0) \;\; X(u,1)] \begin{pmatrix} 1-v \\ v \end{pmatrix}$$

$$-[1-u \;\; u] \begin{pmatrix} X(0,0) & X(0,1) \\ X(1,0) & X(1,1) \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}.$$

Lo anterior pudiera perfeccionarse usando interpolación bicúbica, hermitiana u otra, pero la base del razonamiento es la misma, del mismo modo que se puede aplicar la teoría de Splines uniendo "pedazos" de Coons.

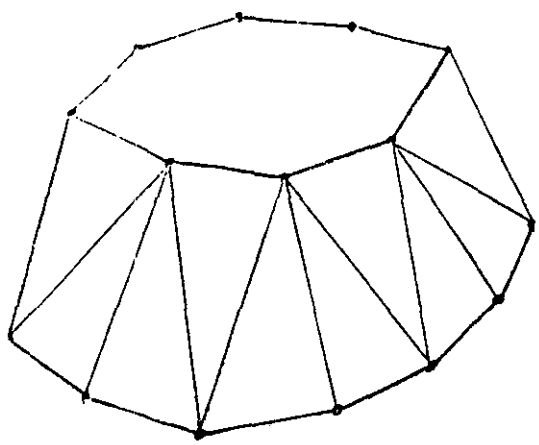

Figura 5
Triangulado entre poligonales

1.3.5 Casos Complicados

Muchas veces, como en el caso de los datos provenientes de tomografías computarizadas, la superficie a representar está dada por una serie de cortes transversales, y estos cortes transversales se presentan como poligonales. Aparentemente este sería un caso a resolver con Coons, sin embargo existe el problema de que las poligonales no están en planos paralelos y no tienen el mismo número de puntos. Ésto dificulta la aplicación de los Coons, dado que no se puede elegir el mismo parámetro para las dos curvas, y menos para tres o más, sin hacer un proceso de interpolación previa.

Como muchas otras veces, el problema se ha resuelto en forma práctica triangulando entre curvas como se puede ver en la figura 5, y posteriormente interpolando sobre los triángulos así obtenidos.

Una descripción detallada de lo anterior se sale un poco del objeto del presente trabajo, sin embargo se consideró ilustrativo de un procedimiento práctico poco ortodoxo. Para más referencias al respecto se puede consultar Y.F. Wang y J.K. Aggarwal [37] o H.N. Christiansen y T.W. Sederberg [8].

Capítulo 2

Las esferas de cobertura

2.1 Definiciones básicas

En este capítulo presentamos algunos conceptos geométricos fundamentales, como son los diagramas de Voronoi y los simplejos de Delaunay, e introducimos el concepto de esferas de cobertura, que está íntimamente ligado a los simplejos de Delaunay y que juega un papel principal en el presente trabajo. También demostramos las principales propiedades de las esferas de cobertura. Estas propiedades son fundamentales para las aplicaciones a interpolación que presentaremos en el capítulo 4.

Primeramente enunciamos las definiciones básicas y establecemos la notación que usaremos en este capítulo.

Definición 2.1.1 *Denotaremos por n a un entero positivo fijo, y por $\mathbb{E}$ al espacio $\mathbb{R}^n$ provisto de la distancia euclidiana usual d, definida por*

$$d(x,y) = \left\{ \sum_{i=1}^{n} (x_i - y_i)^2 \right\}^{1/2},$$

donde $x = (x_1, x_2, \ldots, x_n)$ e $y = (y_1, y_2, \ldots, y_n)$.

Definición 2.1.2 *Dados un punto p en $\mathbb{E}$ y un número real positivo r, la esfera abierta con centro en p y radio r es el conjunto*

$$\mathcal{E} = \mathcal{E}(p,r) = \{x \in \mathbb{E} : d(x,p) < r\}.$$

Decimos que $\{x \in \mathbb{E} : d(x,p) = r\}$ *es la frontera de la esfera* $\mathcal{E}$.

Definición 2.1.3 *Si* S *es un subconjunto de* $\mathbb{E}$, *la cerradura convexa de* S, *denotada por* $cc(S)$, *es el menor subconjunto convexo de* $\mathbb{E}$ *que contiene a* S.

Para cada entero positivo m definimos el conjunto

$$I_m = \{1, 2, 3, \ldots, m\}.$$

Si A y B son dos conjuntos, el complemento de A con respecto a B lo denotamos por $B \backslash A$.

Definición 2.1.4 *Sea* $V = \{p_1, p_2, \ldots, p_m\}$ *un conjunto de* m *puntos distintos en* $\mathbb{E}$. *Para cada* i *en* I_m *definimos el conjunto*

$$T_i = \{x \in \mathbb{E} : d(x,p_i) < d(x,p_j), \quad j \in I_m \backslash \{i\}\},$$

llamado el politopo de Voronoi del punto p_i *con respecto a* V.

Es fácil demostrar que T_i es un conjunto abierto y convexo, y que es acotado si y solamente si p_i no está en la frontera de la cerradura convexa de V. Además tenemos que $\{T_1, T_2, \ldots, T_m\}$ es una colección de conjuntos abiertos y ajenos, cuya unión es un conjunto denso en $\mathbb{E}$.

Definición 2.1.5 *Sea* k *un entero tal que* $1 \leq k \leq m$, *y sea* S *un subconjunto de* V *con* k *elementos. Definimos*

$$C(S) = \left\{ x \in \mathbb{E} : \text{ existe } r \geq 0 \text{ tal que } \begin{cases} d(p,x) = r & \text{si } p \in S \\ d(q,x) > r & \text{si } q \in V \backslash S \end{cases} \right\},$$

y

$$\mathcal{D}(V) = \{C(S) : S \subseteq V,\ C(S) \neq \emptyset\}.$$

Si $k < n+1$ y $C(S)$ no es vacío, decimos que $C(S)$ es una cara de dimensión $n+1-k$ del *diagrama de Voronoi* $\mathcal{D}(V)$ de V. Si $k \geq n+1$ y $C(S)$ no es vacío, $C(S)$ es de dimensión cero. Notemos que las caras de dimensión n son los politopos de Voronoi definidos arriba y que $\mathcal{D}(V)$ es una partición del espacio $\mathbb{E}$.

Es fácil comprobar que cada una de las caras de dimensión cero consta de un solo elemento. A estos elementos los llamamos *vértices* del diagrama de Voronoi de V. Es claro que el número de vértices es siempre menor o iqual que $\binom{m}{n+1}$ ya que un vértice equidista de $n+1$ puntos de V por construcción y $\binom{m}{n+1}$ es el máximo número de conjuntos de $n+1$ puntos que se pueden formar con los m y no todos ellos dan origen a vértices.

Definición 2.1.6 *Si p_i y p_j son dos elementos de V tales que $C(\{p_i, p_j\})$ es una cara de dimensión $n-1$ del diagrama de Voronoi, entonces decimos que p_i y p_j son puntos vecinos de Voronoi o vecinos naturales en V. En este caso la cara $C(\{p_i, p_j\})$ es un subconjunto de la intersección de las fronteras de los politopos T_i y T_j.*

Si el conjunto V tiene al menos $n+1$ puntos distintos que no están contenidos en un subespacio afín de $\mathbb{E}$ de dimensión menor que n, entonces es posible demostrar que existen vértices y caras de dimensión k, para cada $k \in I_n$, en el diagrama de Voronoi de V. Véase Delaunay [13]. En lo sucesivo consideraremos que V satisface dichas condiciones.

Cada vértice u del diagrama de V está determinado por un subconjunto S_u de V con al menos $n+1$ elementos, de los cuales equidista. La cerradura convexa de S_u es un politopo en $\mathbb{E}$ cuyos vértices son elementos de V, y que llamaremos *politopo de Delaunay* de u. Si S_u tiene exactamente $n+1$ puntos, entonces su cerradura convexa es un simplejo en $\mathbb{E}$. En algunas ocasiones usaremos el término "triángulo" en vez de simplejo, aun cuando n sea mayor que dos. Notemos que u no tiene que pertenecer a $cc(S_u)$.

Se puede demostrar que la unión de los politopos de Delaunay del diagrama de Voronoi es igual a la cerradura convexa de V. Véase Shamos y Preparata [26].

En la figura 6 mostramos un ejemplo en el plano de un diagrama de Voronoi y los triángulos de Delaunay correspondientes. El triángulo ABC es un ejemplo donde $S_u = ABC$ y su incentro u no pertenece a $cc(S_u)$.

Definición 2.1.7 *Sea $\mathcal{E}$ una esfera abierta con centro en c y radio r. Para cada punto x en $\mathbb{E}$ definimos la potencia de x con respecto a la esfera $\mathcal{E}$, denotada por $\rho(x, \mathcal{E})$, como sigue*

$$\rho(x, \mathcal{E}) = d^2(x, c) - r^2.$$

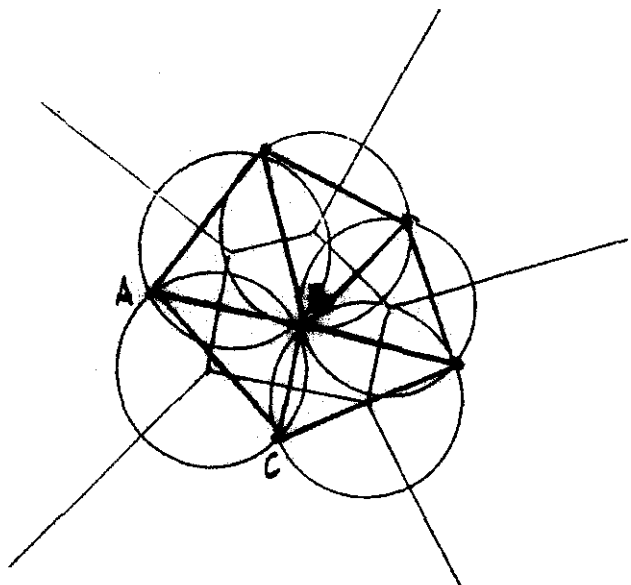

Figura 6
Diagrama de Voronoi con sus correspondientes esferas.
y sus triángulos de Delaunay

Notemos que los puntos de $\mathcal{E}$ tienen potencia negativa, los de su frontera potencia cero, y los puntos exteriores a $\mathcal{E}$ tienen potencia positiva.

Definición 2.1.8 *Sean $\mathcal{E}_1(c_1, r_1)$ y $\mathcal{E}_2(c_2, r_2)$ dos esferas abiertas en* $\mathbf{E}$. *El conjunto*

$$H(\mathcal{E}_1, \mathcal{E}_2) = \{x \in \mathbf{E} : \rho(x, \mathcal{E}_1) = \rho(x, \mathcal{E}_2)\}$$

es llamado el hiperplano radical de las esferas $\mathcal{E}_1$ y $\mathcal{E}_2$.

Notemos que, si $c_1 \neq c_2$ entonces $H(\mathcal{E}_1, \mathcal{E}_2)$ no es vacío. En el caso de $n = 2$ el hiperplano es el eje radical de las circunferencias.

Teorema 2.1.9 *Sean $\mathcal{E}_1$ y $\mathcal{E}_2$ dos esferas abiertas con centros distintos e intersección no vacía y sean F_1 y F_2 los semiespacios abiertos que forman el complemento del hiperplano radical H de las esferas, donde F_k contiene al centro de $\mathcal{E}_k$ para $k = 1, 2$. Entonces tenemos:*

i) Los puntos de la frontera de $\mathcal{E}_1$ que pertenecen a F_2 son puntos interiores de $\mathcal{E}_2$.

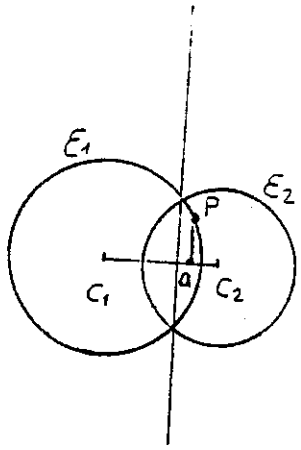

Figura 7

*) *Los puntos de la frontera de $\mathcal{E}_1$ que pertenecen a F_1 son exteriores a la esfera $\mathcal{E}_2$.*

*) *Si x es un punto en $\mathcal{E}_1 \cap \mathcal{E}_2 \cap F_1$ entonces $\rho(x, \mathcal{E}_1) < \rho(x, \mathcal{E}_2)$.*

Las afirmaciones que se obtienen al intercambiar los índices 1 *y* 2 *nbién se cumplen.*

Demostración. i) Sean c_1 y c_2 los centros y r_1 y r_2 los radios de $\mathcal{E}_1$ y $\mathcal{E}_2$ respectivamente. Sea L la recta que pasa por los centros y sea a el punto de intersección de L con el hiperplano radical H. Sea p un punto de la superficie de $\mathcal{E}_1$ que está en el semiespacio F_2. Entonces, la proyección ortogonal q de p sobre la recta L está en la semirrecta que parte de a y pasa por c_2, y además $q \neq a$.

En el plano determinado por p, c_1 y c_2 tenemos la figura 7.

Definimos las distancias $d_1 = d(a, c_1)$, $d_2 = d(a, c_2)$, $e = d(a, q)$ y $f = d(p, c_2)$. Notemos que e es positivo.

Como a está en H tenemos $\rho(a, \mathcal{E}_1) = \rho(a, \mathcal{E}_2)$ y por tanto

$$d_1^2 - r_1^2 = d_2^2 - r_2^2. \tag{2.1.1}$$

De la figura vemos que

$$r_1^2 - (d_1 + e)^2 = f^2 - (d_2 - e)^2, \tag{2.1.2}$$

y por tanto

$$r_1^2 - d_1^2 - 2d_1e = f^2 - d_2^2 + 2d_2e.$$

Usando (2.1.1) obtenemos

$$r_2^2 - f^2 = 2e(d_1 + d_2) > 0,$$

y esto significa que p es un punto interior de la esfera $\mathcal{E}_2$.

La parte ii) se demuestra de manera análoga. En este caso q está en la semirrecta que parte de a y pasa por c_1.

Es bien conocido que tres puntos no colineales en el plano determinan una única circunferencia que pasa por ellos. La proposición correspondiente en dimensión n es el siguiente teorema.

Teorema 2.1.10 *Sean $p_0, p_1, \ldots, p_n$ puntos distintos en $\mathbb{E}$ que no están contenidos en un hiperplano de dimensión menor que n. Entonces existe una única esfera $\mathcal{E}(c, r)$ tal que p_k está en la frontera de $\mathcal{E}$ para $k = 0, 1, \ldots, n$.*

Demostración. Sean $c = (c_1, c_2, \ldots, c_n)$ y r el centro y el radio de la esfera buscada y sea $p_k = (x_{1,k}, x_{2,k}, \ldots, x_{n,k})$. Entonces se deben cumplir las ecuaciones

$$d^2(p_k, c) = \sum_{j=1}^{n} (c_j - x_{j,k})^2 = r^2, \qquad 0 \leq k \leq n. \tag{2.1.3}$$

Si restamos la ecuación con $k = 0$ de las demás obtenemos el sistema de ecuaciones lineales siguiente

$$\sum_{j=1}^{n} (x_{j,k} - x_{j,0}) c_j = \frac{1}{2} \sum_{j-1}^{n} (x_{j,k}^2 - x_{j,0}^2), \qquad 1 \leq k \leq n. \tag{2.1.4}$$

La matriz de coeficientes del sistema es

$$\begin{pmatrix} x_{1,1} - x_{1,0} & x_{2,1} - x_{2,0} & \cdots & x_{n,1} - x_{n,0} \\ x_{1,2} - x_{1,0} & x_{2,2} - x_{2,0} & \cdots & x_{n,2} - x_{n,0} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1,n} - x_{1,0} & x_{2,n} - x_{2,0} & \cdots & x_{n,n} - x_{n,0} \end{pmatrix}.$$

Esta matriz es no singular porque la hipótesis del teorema implica que el conjunto de vectores $\{p_1 - p_0, p_2 - p_0, \ldots, p_n - p_0\}$ es linealmente independiente. Por tanto el sistema (2.1.4) tiene una única solución c. Notemos que r queda determinado por cualquiera de las ecuaciones (2.1.3).

Con las definiciones y la notación dadas arriba podemos ahora introducir el concepto de esferas de cobertura.

Definición 2.1.11 *Sea V un conjunto de m puntos distintos en $\mathbb{E}$, con $m > n$, y que no están contenidos en un subespacio de $\mathbb{E}$ de dimensión menor que n. Una esfera abierta $\mathcal{E}$ en $\mathbb{E}$ se llama esfera de cobertura asociada a V si y solamente si se cumplen las condiciones siguientes:*

i) $\mathcal{E}$ no contiene ningún punto de V.

ii) La frontera de $\mathcal{E}$ contiene al menos $n+1$ puntos de V.

Denotaremos por S *al conjunto de todas las esferas de cobertura asociadas a V.*

El conjunto S es siempre no vacío, finito y su cardinalidad depende del número de puntos en V y de la forma en que éstos están distribuídos en el espacio $\mathbb{E}$. Es claro que $\binom{m}{n+1}$ es una cota superior para el número de esferas de cobertura de V.

2.2 Propiedades de las esferas de cobertura

Definición 2.2.1 *Si $\mathcal{E}$ es una esfera de cobertura asociada a V decimos que el conjunto de al menos $n+1$ puntos que se encuentran en su frontera es el conjunto de generadores de $\mathcal{E}$.*

Observación 2.2.2 *Como se desprende de las definiciones 2.1.4 y 2.1.11, dado V los centros de las esferas de cobertura asociadas son vértices de los politopos de Voronoi definidos por V.*

Observación 2.2.3 *Consideraremos que los hiperplanos que contienen caras de dimensión n de la frontera de $cc(V)$ son también esferas de cobertura de V. Tales hiperplanos son esferas degeneradas con centro en el punto del infinito y la unión de sus interiores es el complemento de la cerradura convexa de V.*

El concepto de esferas de cobertura se puede relacionar históricamente con otros previamente definidos en la literatura, como por ejemplo, la teselización de Voronoi. De hecho, la primera referencia a una entidad similar se hizo por B. Delaunay [13] en un artículo de la revista de la Academia de Ciencias de la URSS de 1934, dedicado a la memoria de G. Voronoi. En este caso la entidad descrita era una esfera que podía pasar, dilatándose o contrayéndose, entre los puntos de un conjunto, sin contener a ninguno, aunque "tocándolos".

Se revisó exhaustivamente la bibliografía de 1934 a la fecha y pese a que todo mundo llama a los triángulos de Delaunay nunca se hace referencia al citado artículo, que es el la fuente original de la idea. Tampoco se ha visto que se retome la idea de la esfera como construcción principal o base de todas las demás. Por ejemplo, en el libro de Shamos y Preparata [26] se dan todas las propiedades de la teselización de Voronoi, entre las que se incluye el que los círculos que circunscriben a los triángulos de Delaunay no contienen ningún punto del conjunto, pero no se avanza más allá, lo qu es más, impone la innecesaria limitante de que no haya cuatro puntos cocirculares.

Es hasta 1980 que en un artículo de Sibson [30] se vuelve a encontrar establecida una propiedad adicional de las esferas, en este caso de los círculos, como es la propiedad de que cubren a la cerradura convexa del conjunto dado de puntos.

En los años 80 han vuelto a aparecer, ya relacionados con CAGD (sigla inglesa para diseño geométrico ayudado por computadora) , artículos relativos a los polígonos de Voronoi, generalizaciones a dimensiones superiores y también acerca de los triángulos de Delaunay; podemos citar en ese sentido el trabajo de Watson [38], el de Bowyer [5], así como el libro de Farin [18], los cuales a su vez son producto de exhaustivas investigaciones bibliográficas. Sin embargo en todos ellos los círculos y esferas son generalmente construcciones auxiliares de poca relevancia, debido al hecho de que los autores tratan de discretizar un dominio y consideran como una de las más importantes propiedades de una discretización que no se traslapen los trozos en los que se divide el dominio. Lógicamente, en este marco, trabajar con círculos o esferas que sí se traslapan resulta absurdo y tal vez por ello los citados autores, teniendo todos los elementos de partida para hacerlo, no desarrollaron el concepto desde el punto de vista que se hará en el presente trabajo.

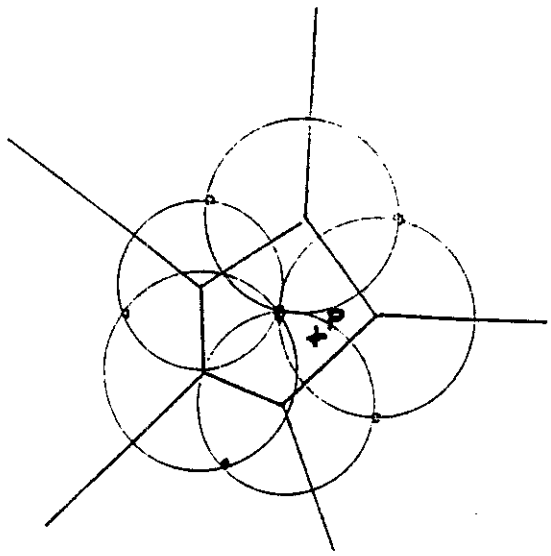

Figura 8
Un politopo acotado, un punto interior y las esferas que lo cubren.

Teorema 2.2.4 *Sea $T_i = C(\{p_i\})$ un politopo de Voronoi de V. Si P es un punto de T_i o de su frontera tal que $P \neq p_i$ entonces P es interior a al menos una de las esferas de cobertura de V que tienen a p_i como generador.*

Demostración: Los politopos de Voronoi pueden ser acotados e interiores a la cerradura convexa o no acotados. En el caso de los politopos acotados, tómese un punto P interior al mismo o en su frontera (ver figura 8) y considérese un vértice del politopo que esté a la distancia mínima de P, entonces el punto P es interior a la esfera con centro en el vértice y que pasa por el centro del politopo.

En el caso de los politopos no acotados, una parte es cubierta por el complemento de la cerradura convexa y la parte del politopo interior a la misma admite la demostración del teorema anterior, como se ilustra en la figura 9.

Usando un razonamiento similar al del teorema 2.1.1 se ve que la superficie (en este caso el arco) UPV es totalmente interior al UVW (que generan el círculo de cobertura).

Observación 2.2.5 *El conjunto* **S** *es el dual de $\{T_i\}$ ya que existe una regla bien definida para generar uno a partir del otro. Usualmente se*

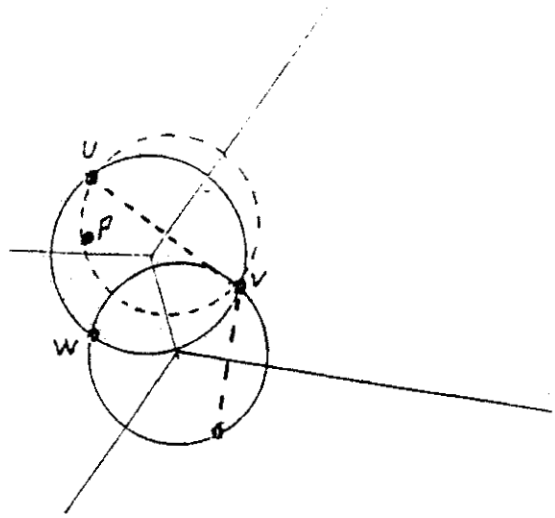

Figura 9
La linea punteada es la cerradura convexa.

considerаba a la triangulación de Delaunay como dual de los politopos de Voronoi, pero existen casos de ambiguedad, ya que cuando hay más de N puntos de V sobre la frontera o superficie de la esfera, hay varias construcciones equivalentes de los triángulos de Delaunay como lo muestra la figura 10.

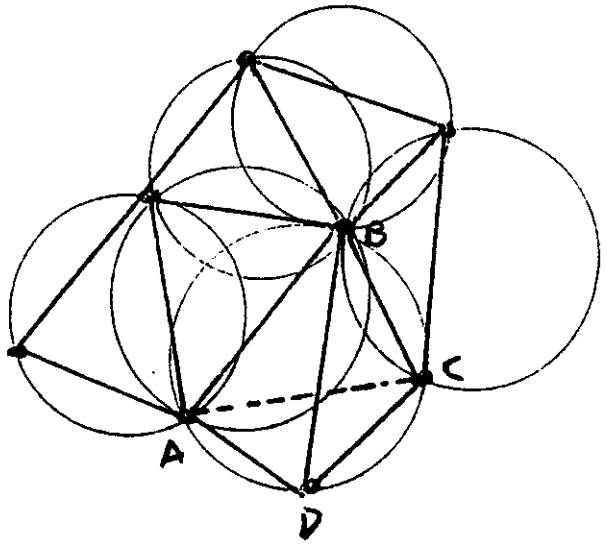

Figura 10
Al haber más de $n + 1$ generadores de la esfera existen más de una triangulaciones válidas. Tanto la triangulación ABD DBC como la ABC ACD (punteada) son válidas.

Capítulo 3

Algoritmos de construcción

Se presentará una breve reseña de varios algoritmos para generar esferas de cobertura. Todos ellos tienen en común una serie de propiedades debidas a que se requiere que el algoritmo que se use debe permitir agregar puntos al conjunto V que se introdujo en el capítulo 2, sin tener que rehacer todo el esquema.

Finalmente se presentará y justificará la elección realizada y el programa resultante.

3.1 El algoritmo en general

Un problema básico que se debe resolver en un algoritmo como el presente, que, como se verá más adelante, debe ser incremental, es el de localizar un punto en el espacio (el nuevo que se está introduciendo) con respecto, ya sea a la teselización de Voronoi o a las esferas de cobertura, o bien a los triángulos de Delaunay.

Normalmente la solución propuesta por la mayoría de los autores ha sido el uso de coordenadas locales respecto a un triángulo o su equivalente en más dimensiones. Esta solución normalmente implica calcular tres áreas (volúmenes o hipervolúmenes, según la dimensión). En la figura 11 se muestra un ejemplo de ubicación de un punto de esta forma.

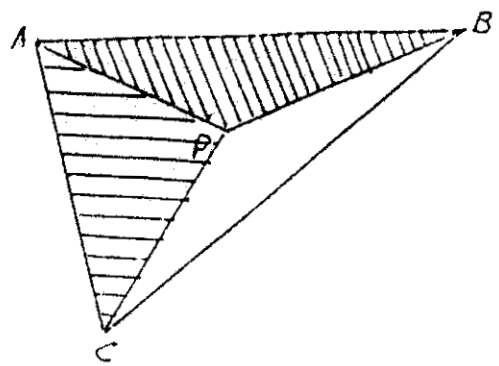

Figura 11
Se muestra un triángulo y las áreas usadas para ubicar P.

Como se ve, la coordenada de P respecto a A es $\frac{PCB}{ABC}$, con respecto a B es $\frac{PCA}{ABC}$ y con respecto a C es $\frac{PAB}{ABC}$.

El número de operaciones en el caso de la figura son las necesarias para evaluar las áreas de los triángulos ABC, ABP y BCP, lo cual implica evaluar determinantes como sigue.

$$\begin{bmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{bmatrix}, \begin{bmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_p & y_p & 1 \end{bmatrix}, \begin{bmatrix} x_b & y_b & 1 \\ x_c & y_c & 1 \\ x_p & y_p & 1 \end{bmatrix}.$$

Puesto que la tercer coordenada se puede calcular por diferencia, no se toma en cuenta.

Este es el caso en dos dimensiones, pero, si por ejemplo se tienen tres dimensiones, entonces los determinantes son de cuatro por cuatro y así sucesivamente.

3.1.1 Posibilidad de basarse solo en la potencia

Si ahora el caso es que se tienen esferas en vez de simplejos, o en dos dimensiones círculos en vez de triángulos, ubicar un punto respecto a una esfera es algo mucho más sencillo; basta con conocer la distancia del punto al centro de la esfera y las características de ésta, que son el centro y el radio.

Debido a una serie de ventajas que se verán en el transcurso del trabajo, se usará el concepto de potencia que se definió en el capítulo 2 (def 2.1.7). De esta forma, ubicar el punto P respecto al círculo C implica calcular

$$(x_c - x_p)^2 + (y_c - y_p)^2 - r^2$$

donde x_c, y_c son las coordenadas del centro, x_p, y_p las coordenadas de P y r es el radio del círculo. Notemos que al aumentar una dimensión solo es necesario agregar otro término.

Observación 3.1.1 *Si el punto P es interior a más de una esfera, se dirá que es "más interior" a aquella respecto a la cual tiene menor potencia. Si P es además interior a $cc(V)$ entonces es interior a cualquier simplejo de Delaunay circunscrito en ella.*

En la figura 12 se ilustra la observación anterior en el caso de dos dimensiones. Como la frontera del triángulo de Delaunay correspondiente es el eje radical de ambas circunferencias, el punto P pertenece a aquella respecto a la cual su potencia es menor.

En efecto, el eje radical tiene igual potencia respecto a ambas, y a cada lado de él la potencia disminuye para la circunferencia respecto a la cual el centro está más cercano, como se vió en el teorema 2.1.1.

Es de destacar el menor número de operaciones que se realizan al calcular la potencia que al calcular varias áreas, y sobre todo el hecho de que la complejidad de los cálculos crece mucho menos al crecer la dimensión.

Así, para determinar las coordenadas baricéntricas de P respecto a ABC se debe calcular

$$\frac{\begin{bmatrix} x_p & y_p & 1 \\ x_a & y_a & 1 \\ x_b & y_b & 1 \end{bmatrix}}{\begin{bmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{bmatrix}}, \quad \frac{\begin{bmatrix} x_p & y_p & 1 \\ x_a & y_a & 1 \\ x_c & y_c & 1 \end{bmatrix}}{\begin{bmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{bmatrix}} \quad y \quad \frac{\begin{bmatrix} x_p & y_p & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{bmatrix}}{\begin{bmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{bmatrix}},$$

lo cual implica calcular los tres numeradores

$$x_p(y_a - y_b) - y_p(x_a - x_b) + (x_a y_b - x_b y_a)$$

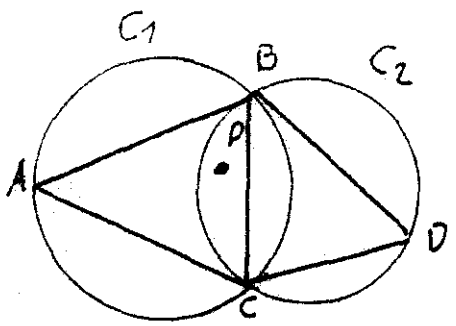

Figura 12
Se puede saber que P pertenece al triángulo ABC
porque P tiene menor potencia respecto a C_1 que a C_2.

$$x_p(y_a - y_c) - y_p(x_a - x_c) + (x_a y_c - x_c y_a)$$

$$x_p(y_b - y_c) - y_p(x_b - x_c) + (x_b y_c - x_c y_b)$$

y el denominador

$$x_a(y_b - y_c) - y_a(x_b - x_c) + (x_b y_c - x_c y_b).$$

Todo esto requiere calcular 20 sumas, 16 multiplicaciones y 3 divisiones. La potencia, en cambio, está dada por

$$(x_p - x_c)^2 + (y_p - y_c)^2 - r^2$$

y requiere solamente 3 productos y 4 sumas.

Como se verá más adelante, el usar la potencia tiene además la ventaja de que no solo permite localizar el nuevo punto dentro del simplejo o triángulo correspondiente, sino que, cuando se usan algoritmos incrementales, también permite localizar todas las esferas y simplejos afectados por el punto sin tener que realizar operaciones extra.

3.1.2 El sistema de búsqueda y su orden de complejidad menor que N^2

Como se trata de un algoritmo incremental, los puntos se van agregando uno a uno, por lo que la hipótesis general es que se tiene un conjunto de esferas de cobertura correspondientes a un determinado conjunto de puntos V, y que entonces se agrega un nuevo punto al conjunto V.

El primer problema que se presenta es entonces localizar en cuales esferas cae el nuevo punto, si es que cae en alguna, a fin de saber a cuáles esferas afecta, para reconstruir el conjunto de esferas teniendo en cuenta el nuevo punto.

El algoritmo que primero viene a la mente es simplemente calcular la potencia del punto respecto a cada una de las esferas de cobertura. El punto estará dentro de todas aquellas para las que tenga potencia negativa.

Los diferentes autores consultados tienen muy variadas definiciones del orden de complejidad de un algoritmo, para el caso presente se considerará que es el número de situaciones que (en el peor de los casos) hay que tener en cuenta. Por ejemplo el número de esferas respecto a las cuales hay que hallar la potencia. Lógicamente este número tiene relación directa con el número de operaciones aritméticas a realizar.

En consecuencia, un algoritmo como el descrito arriba aplicado recursivamente a los N puntos del conjunto V tiene complejidad N^2.

Se trata entonces de encontrar un algoritmo que requiera un menor número de operaciones. Se ha retomado en este sentido la idea de búsqueda orientada que se usa para construir triangulaciones.

Usualmente, cuando se tiene una región triangulada y se quiere ubicar un punto en ella se utilizan coordenadas locales, las cuales sirven también para indicar cual triángulo se debe revisar posteriormente si resultó que el punto no está en el primero con que se probó. En efecto, (ver figura 13) el triángulo elegido será el contiguo al lado respecto del cual el punto tiene coordenada de diferente signo.

En el caso de las esferas suponemos que tenemos el conjunto $\mathbf{S}_f$ de las esferas de cobertura asociadas al conjunto V que tienen radio finito y que para cada una de estas esferas conocemos el conjunto de puntos de V que la generan. Dado un punto cualquiera P que no está en V trataremos de encontrar todas las esferas en $\mathbf{S}_f$ que lo contienen.

La idea básica del algoritmo consiste en tomar una esfera cualquiera S_0

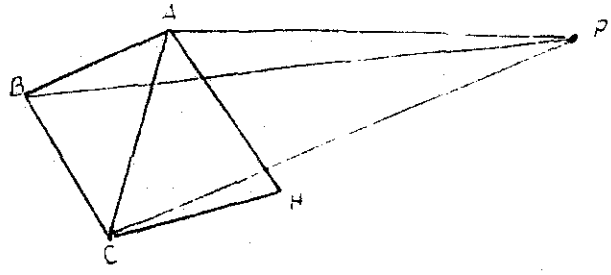

Figura 13
Para proseguir la búsqueda se toma el triángulo ACH,
ya que recorriendo los triángulos APB, BPC y CPA
este último tiene distinto signo.

en $\mathbf{S}_f$ y construir una sucesión alternada de vértices y centros de esferas que se inicia con S_0 y llega a esferas que contienen a P, sin tener que analizar todas las esferas.

Supongamos que P no está en S_0. Sea O el centro de S_0 y sean $v_1, v_2, \ldots, v_r$ los generadores de S_0. Se encuentra v_k, con $1 \leq k \leq r$, tal que su distancia a P es la mínima. Se analizan ahora las esferas tales que v_k pertenece a su lista de generadores. Sea esa lista $E_1, E_2, \ldots, E_h$. De estar P fuera de todas ellas se individualiza a aquella E_j tal que su centro sea el más cercano a P y con ella se repite el procedimiento. De esta manera construimos una sucesión alternada de centros y vértices.

El procedimiento termina cuando hay repetición de un vértice v o de un centro. En el caso de que se repita un vértice, sea éste v, por construcción podemos asegurar que no existe ningún vecino natural de v mas cercano a P. Por tanto P pertenece al politopo de Voronoi de v.

Si se repite un centro c entonces escogemos v como el generador de la esfera con centro en c mas cercano a P.

Otra forma de terminación puede ser cuando la potencia de P respecto a alguna esfera analizada es negativa. Cabe aclarar que, luego de la primera esfera, no es estrictamente necesario calcular la potencia y que el solo criterio de repetición nos da el resultado correcto.

Como todo politopo está totalmente cubierto por las esferas que pasan

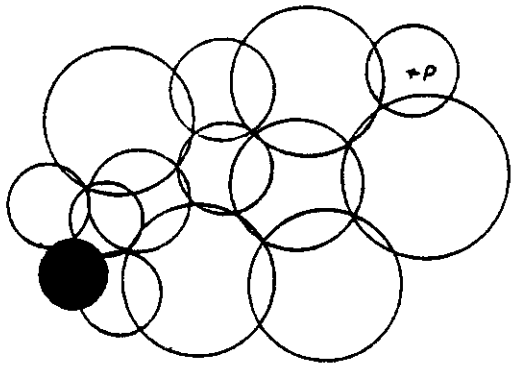

Figura 14
Se quiere localizar el punto P y la primera esfera considerada es la pintada de negro.

por el punto que le da origen, entre ellas debe existir una que contiene a P.

Como solo se guardan las esferas de radio finito pudiera ser que P no perteneciera a ninguna de ellas, entonces se sabe que pertenece a una esfera degenerada de las que forman el complemento de $cc(V)$.

Por otra parte, al terminar el procedimiento, no solo quedan localizadas las esferas en la que cae el punto P, sino que también queda localizado en que politopo de Voronoi y en que simplejo de Delaunay se encuentra.

Observación 3.1.2 *El proceso siempre se completa y en un número finito de operaciones.*

Este procedimiento es muy similar al que publicaron Green y Sibson [32] en 1977, que solo se refería a encontrar el vecino más cercano, lo cual es un problema equivalente al que consideramos aquí.

Obsérvese que al elegir un generador de una esfera se descartan los demás generadores de dicha esfera, junto con todos los vecinos naturales de los generadores descartados que no son vecinos del generador elegido.

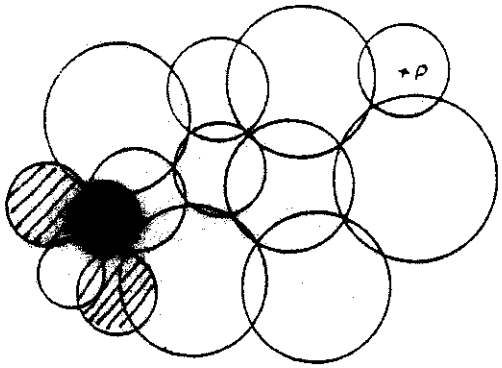

Figura 15
De todas las vecinas se elige la que tiene centro más cercano, pintada en negro.

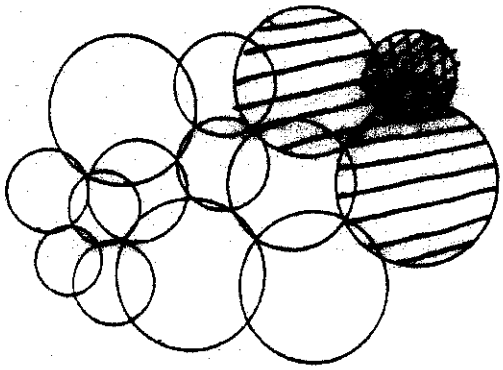

Figura 16a
Se encuentra la que tiene potencia negativa y se analizan sus vecinas.

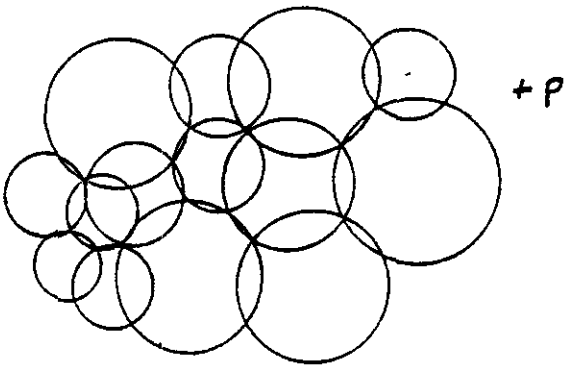

Figura 16b
El punto puede no tener potencia negativa respecto a
ninguna esfera de radio finito.

Obsérvese también que la sucesión de distancias de los vértices a P, o de los centros a P, es decreciente.

La sucesión de los politopos de los vértices que se eligen en cada paso forman un conjunto conexo, ya que avanza sobre politopos tales que cada uno tiene una cara en común con el siguiente. Por tanto, cuando se da la repetición se ha localizado el centro o el vértice o vértices más cercanos a P, ya que si hubiera alguno distinto, mas cercano, se podría continuar el camino hasta él, ya que los politopos forman una partición del espacio y no hay "huecos" posibles, ni en el conjunto de esferas ni en el de politopos.

El hecho de que exista un número finito de puntos en V implica que el resultado se obtiene en un número finito de pasos.

Observación 3.1.3 *El número de operaciones es de orden menor que* N^2.

Como se podrá apreciar, en general no es necesario, salvo cuando se trata de muy pocos puntos, analizar todas las esferas, lo cual hace al orden de complejidad del algoritmo menor de N^2.

El procedimiento es similar a otro que se realiza con triángulos recurriendo a las coordenadas baricéntricas, para el cual se demuestra (Palacios [25]) que es el más corto, por lo que es razonable suponer que éste también lo es, dada la relación que existe entre las esferas de cobertura y los triángulos de Delaunay.

En el citado artículo de Green y Sibson se menciona la posibilidad de efectuar la búsqueda en un orden de $N^{3/2}$.

3.1.3 El algoritmo de construcción incremental de esferas

La continuación lógica del proceso descrito en el apartado anterior es una vez localizado el punto y determinadas las esferas que afecta (se dice que las afecta porque al ser considerado el nuevo punto como parte de V y al quedar éste dentro de ellas, dejan automáticamente de ser esferas de cobertura), se debe proceder a corregir el conjunto de esferas quitando las afectadas y reemplazándolas por nuevas (en el caso 16a) o simplemente generando las esferas que falten (caso 16b).

Caso 1: El punto es interior a una o más esferas

En este caso se ha procedido a generalizar el algoritmo de Watson [38] de la siguiente manera.

En primer lugar se enlistan los puntos que generan a las esferas afectadas. Se probará que tomando conjuntos de N puntos de ellos en forma adecuada, que también se indicará, se obtienen, junto con P las nuevas esferas y éstas son de cobertura.

Una esfera en un espacio de N dimensiones es generada por $N+1$ puntos, uno de ellos es P, si los puntos pertenecientes a esferas afectadas son M con $M > N+1$, se trata de elegir los conjuntos que tienen N de los M puntos entre los $\binom{M}{N}$ posibles.

Para abordar el problema considérese primero el caso aislado de una sola esfera y un punto P interior a ella. Existen dos posibilidades: que P pertenezca a la cerradura convexa de los puntos de V que generan a la esfera o que sea exterior, en cada caso las nuevas esferas se construyen de forma diferente.

Teorema 3.1.4 *Sea P interior a una esfera de cobertura $\mathcal{E}$, sea $\mathcal{G} = \{G_1, G_2, \ldots, G_k\}$ el conjunto de sus puntos generadores, considérese que P está en el interior de $cc(\mathcal{G})$. Entonces las esferas $\mathcal{E}_i$ formadas por P y los conjuntos de N puntos que formen caras o facetas de la cerradura convexa de $\mathcal{G}$ no contienen a ningún punto de $\mathcal{G}$.*

Demostración: La nueva esfera (ver figura 17) queda dividida por el plano o hiperplano (en dos dimensiones la línea) que contiene a la cara o faceta de N puntos de $cc(\mathcal{G})$ en dos partes, una exterior a la cerradura convexa y otra en cuya superficie está P; obviamente ningún punto de los que determinan a la esfera original puede estar dentro de la primera parte, se trata de ver que tampoco puede estar en la segunda.

Cuando dos esferas se cortan quedan divididas en dos conjuntos, uno que tiene potencia positiva (exterior) a la otra y otro que tiene potencia negativa (interior) a la otra. Dado que P es por hipótesis interior a la esfera afectada, el conjunto que lo contiene tiene, al igual que P, potencia negativa respecto a la esfera afectada, por lo que no puede contener a ningún punto de su superficie, ya que los puntos de su interior tienen también potencia negativa y los puntos de la superficie de la afectada tienen, respecto a ella potencia cero.

Teorema 3.1.5 *Si P afecta a una esfera pero no pertenece al interior de la cerradura convexa del conjunto $\mathcal{G}$ de los puntos que la determinan, entonces se cumple el teorema 3.1.4 con la excepción de las facetas que separan a P de la cerradura convexa de $\mathcal{G}$.*

Demostración: En este caso (ver figura 18) la cerradura convexa está, siguiendo el razonamiento de la demostración del teorema 3.1.4, totalmente contenida en la sección de esfera generada por P y los vértices de la faceta exceptuada, que tiene potencia positiva, ya que la otra contiene a P, que tiene potencia negativa.

El razonamiento del teorema 3.1.4 es sin embargo válido para las esferas formadas por el punto P y las restantes figuras de la cerradura convexa, ya que en esos casos P queda "del mismo lado" que la cerradura convexa respecto a la faceta con la que forma la esfera.

Llega ahora el momento de unir los razonamientos anteriores para tratar el caso de que el punto P afecte a varias esferas simultáneamente.

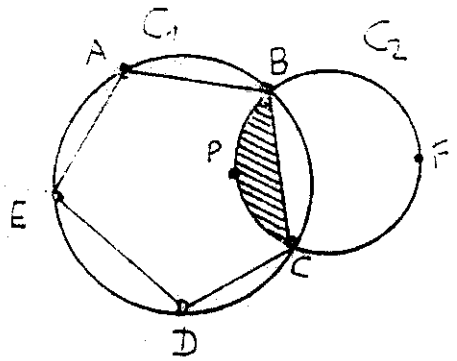

Figura 17
Se ilustra el teorema en el caso de dos dimensiones; el círculo C_1 es generado por los puntos $ABCDE$, P es interior al polígono que forma su cerradura convexa y a C_1 C_2 es el círculo determinado por PBC. La parte rayada de C_2 no puede contener a A, E o D porque tiene potencia negativa respecto a C_1 por ser interior a la cerradura convexa por construcción.

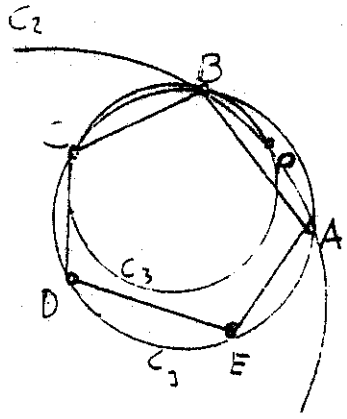

Figura 18
El círculo PAB (C_2) contiene a D y C, sin embargo el PCB (C_3) no contiene a D y A al igual y por las mismas razones que en el teorema anterior.

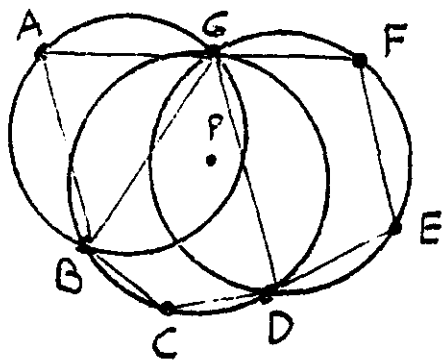

Figura 19
Se ilustra el anterior teorema en dos dimensiones. P afecta a los círculos C_1, C_2 y C_3 generados respectivamente por los puntos $FGDE$, $DGBC$ y GBA, pero solo cae dentro de la cerradura convexa $DGBC$, por lo tanto para C_2 está en las condiciones del teorema 3.1.4 y para C_1 y C_3 en las del 3.1.5

Teorema 3.1.6 *Sea P un punto interior a las esferas de cobertura $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_K$ y sea $\mathcal{G}_i$ el conjunto de generadores de $\mathcal{E}_i$. Sean $G_1, G_2, \ldots, G_N$ puntos en algún $\mathcal{G}_i$ tales que $cc\{G_1, G_2, \ldots, G_N\}$ es una cara de la frontera de $cc\{\mathcal{G}_i\}$ y no está contenida en la frontera de la cerradura convexa de ningún otro $\mathcal{G}_j$. Sea $\mathcal{E}$ la esfera generada por P y los puntos $G_1, G_2, \ldots, G_N$. Entonces $\mathcal{E}$ no contiene a ningún punto de $\bigcup_{i=1}^{K} \mathcal{G}_i$.*

Demostración: El punto P caerá necesariamente dentro de la cerradura convexa de $\mathcal{G}_j$ para una de las esferas $\mathcal{E}_j$, y fuera de las otras, ya que estas cerraduras no pueden traslaparse. Para esta esfera (ver figura 19) se cumple el teorema 3.1.4 y para las restantes el teorema 3.1.5. Puede suceder que P caiga fuera de todas las cerraduras y en ese caso se aplica el teorema 3.1.3. Lo que no puede sucéder es que caiga en más de una.

Teorema 3.1.7 *Dado un conjunto V de M puntos en el espacio de dimensión N sea $\mathbf{S}$ el conjunto de esferas de cobertura que ellos determinan y cuyo número será s. Sea un punto P que se agregará a V y tal que es interior a h esferas de $\mathbf{S}$, con $h < c$. Entonces las esferas generadas por P y los vértices de las facetas de la unión de las cerraduras convexas de los*

generadores de las esferas afectadas (según el teorema 3.1.3) son esferas de cobertura de $V \cup \{P\}$.

Demostración: El teorema 3.1.7 garantiza que las nuevas esferas no contienen a ningún punto de los que generaban a las esferas afectadas, el problema es demostrar que no contienen a ningún otro punto exterior a ellas.

Considérese entonces, por ejemplo, la esfera formada por P y una cualquiera de las figuras antes descritas, la cual además forma una esfera con otros puntos exteriores (ver figura 20).

El punto P es exterior a esa esfera, ya que, de no serlo, la hubiera afectado, contra la hipótesis. De esta manera la esfera formada por el punto exterior y la figura puede ser dividida por ella en dos partes: una de ellas es interior a la esfera de P ya que es interior también a una esfera afectada por P. Allí no puede haber puntos exteriores, como es obvio; la otra parte contiene totalmente a la otra mitad de la esfera de P, ya que P tiene potencia positiva respecto a la esfera del punto interior junto con la mitad que lo contiene, y por lo tanto la otra mitad tiene potencia negativa.

Según se puede apreciar existe aún un caso por estudiar que es cuando el punto P no pertenece a ninguna esfera.

Teorema 3.1.8 *Sea* V *un conjunto de* M *puntos y sea* $\mathbb{S}$ *el conjunto de esferas de cobertura que ellos determinan. Sea* P *un punto exterior a todas las esferas. Entonces, las esferas determinadas por* P *y las figuras de* N *puntos totalmente contenidas en la frontera de la cerradura convexa de* V*, tales que todos sus puntos vértice son "vistos" por* P*, es decir, que la recta que une* P *con el vértice no corta a la cerradura convexa de* V*, son esferas de cobertura del conjunto* $V \cup \{P\}$.

Demostración: Obsérvese que, tal como está planteado el problema, (ver figura 21) las figuras "vistas" por P separan al espacio de modo tal que de un lado queda la cerradura convexa y del otro P.

Hecha esta observación, considérese por un lado la esfera formada por P y una de dichas figuras y la otra formada por la figura y algún otro punto de V. Como P tiene potencia positiva respecto a esta última, la sección de la esfera que por él pasa que no lo contiene, necesariamente tiene potencia negativa respecto a la otra esfera por lo que mal puede contener a algún punto de V siendo interior a una esfera de cobertura.

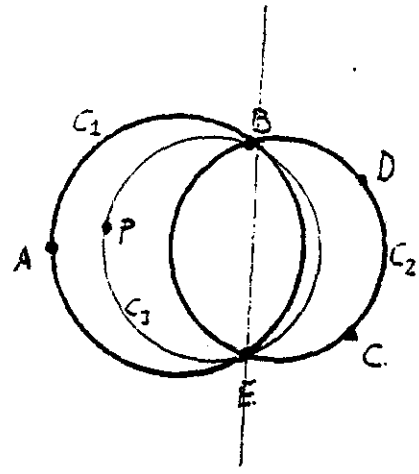

figura 20
Los círculos originales son C_1 y C_2, P afecta a C_1 pero no a C_2. De esta manera el círculo C_3 generado por PBE queda dividido en dos partes por EB. Una, la representada por el arco EPB tiene al menos en ese arco potencia positiva respecto a C_2, la otra totalmente negativa, motivo por el cual no puede afectar a D o C por ejemplo.

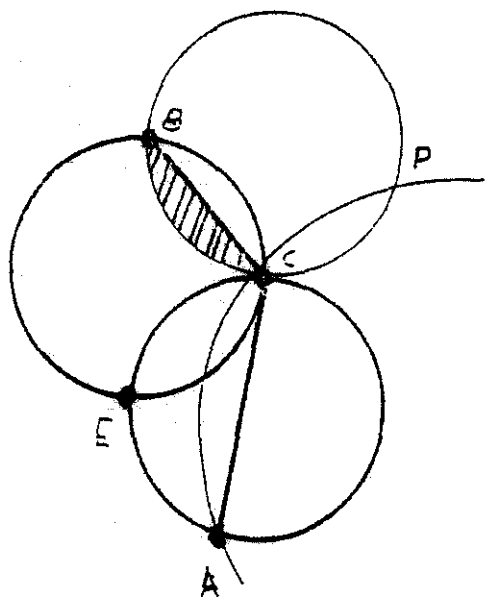

Figura 21
El círculo C_2 generado por PBC tiene una parte totalmente exterior a la cerradura convexa y la interior (rayada) tiene en su totalidad potencia negativa respecto a C_1, por lo que no afecta a E.

Con lo presentado anteriormente se tiene un esbozo de las partes que se utilizan para formar un algoritmo incremental que genere esferas de cobertura. Queda solamente un pequeno problema a resolver: cómo saber si una figura es o no parte de la frontera de la cerradura convexa. Para saberlo basta repetir el razonamiento hecho en los teoremas 3.1.3 y 3.1.4 considerando todas las posibles figuras que son frontera de las cerraduras convexas correspondientes a los generadores de cada esfera de cobertura y eliminando a todas aquellas que pertenezcan a más de una.

3.2 Algoritmos incrementales

Se trata ahora de analizar una por una las posibles alternativas para implementar prácticamente un algoritmo como el esbozado anteriormente.

Una condición que se impondrá a cualesquiera de los algoritmos es la de que debe manejarse con consideraciones locales, de modo tal que cualquier modificación, como agregar un nuevo punto, no implique realizar todo el trabajo desde el principio, sino que las modificaciones se reduzcan a la zona de influencia del nuevo punto.

Es importante destacar lo anterior, ya que ello descarta algunos algoritmos muy usados para realizar triangulaciones, los que pudieran tener su contrapartida con las Esferas de Cobertura, por ejemplo el algoritmo de Lawson y muchos esquema del tipo "divide y vencerás", en particular estos últimos han sido recientemente muy citados como los más rápidos para realizar triangulaciones.

Hecha la salvedad anterior existen dos posibles aproximaciones al problema, la es primera considerar la existencia de una superesfera auxiliar que cubre a todo el conjunto de puntos V. De esta manera al ir introduciendo los puntos de V uno por uno siempre resultarán ser interiores a una esfera por lo menos de modo tal que solo se usará el algoritmo para puntos interiores.

La otra posibilidad es tomar los primeros $N+1$ puntos de V y hacer que éstos generen a la primera esfera y luego aplicar cualquiera de los dos algoritmos (para puntos interiores y para puntos exteriores) según se vayan presentando los sucesivos puntos.

3.2.1 Con esfera auxiliar inicial

El primer paso de este tipo de algoritmos es determinar $N+1$ puntos auxiliares de modo tal que ellos generen una esfera que cubra totalmente a todo el conjunto V.

Un primer requisito es entonces conocer completamente el conjunto V y en particular sus límites superior e inferior en cada una de la dimensiones o bien predeterminar una zona donde es válido insertar puntos del conjunto.

Hecha esta consideración con la observación de que en realidad es una limitante, la hipótesis de trabajo es que todo punto P perteneciente a V que se inserte será siempre interior a alguna esfera de cobertura, por lo que el algoritmo de construcción será siempre el del teorema 3.1.4. Para poder usar el teorema en forma recursiva es necesario demostrar el siguiente:

Teorema 3.2.1 *Una esfera E afectada por un punto P interior a ella es totalmente cubierta por las nuevas esferas formadas con los puntos que la generaban y P.*

Demostración: Sea H un punto de E diferente de P y sea

$$\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$$

el conjunto de los generadores de E.

Entonces existen dos posibles situaciones: H está en el interior de $cc(\mathcal{G})$ o H está en la frontera o en el exterior de $cc(\mathcal{G})$.

Si H es interior a la cerradura convexa entonces está cubierto por las nuevas esferas, porque las cerraduras convexas de los generadores de las nuevas esferas cubren a $cc(\mathcal{G})$.

Si H es exterior a $cc(\mathcal{G})$ también está cubierto, ya que, si se toma la esfera original E y una cualesquiera de las nuevas, todas las partes de E fuera de $cc(\mathcal{G})$ pertenecen a la parte que tiene potencia negativa respecto a alguna de las nuevas esferas, porque todas las partes dentro de la cerradura convexa tienen potencia positiva respecto a alguna de las nuevas esferas.

Este teorema permite entonces asegurar que si el primer punto que se inserte es interior a la esfera inicial, entonces los puntos sucesivos van a pertenecer a esferas que van cubriendo a la inicial y así sucesivamente.

Al culminar el proceso se tendrá que algunas esferas quedan determinadas usando puntos del conjunto de $N+1$ puntos auxiliares con el que se

comenzó, estas esferas deben ser eliminadas y al hacerlo, dependiendo de la disposición original de los datos, puede ser necesario completar la cerradura convexa con algún procedimiento del tipo de la marcha de Jarvis.

3.2.2 Sin esfera auxiliar inicial

Para comenzar este algoritmo se requieren los primeros $N+1$ puntos cualesquiera de V en un orden arbitrario; un nuevo punto P quedará entonces en las condiciones del teorema 3.1.4 o del 3.1.5, por lo que falta demostrar algo parecido al teorema 3.1.6 para el caso de que P sea exterior.

Teorema 3.2.2 *Dado un punto P exterior al conjunto de esferas de cobertura que cubren a un conjunto dado V, las nuevas esferas que lo contienen, construídas según el teorema 3.1.6, tienen intersección no vacía con el conjunto original de esferas.*

Demostración: Por la propia construcción del teorema 3.1.7 que forma las nuevas esferas con P y una cara de la cerradura convexa, esta cara divide a la nueva esfera en una parte con potencia negativa y otra con potencia positiva respecto a alguna esfera del conjunto original, la parte negativa es interior a esta última, por lo que la intersección la contiene y no es vacía.

De esta manera, con los teoremas 3.1.6 y 3.1.7 se puede asegurar la aplicación reiterada de los teoremas 3.1.4, 3.1.5 y 3.1.6 para construir todo el conjunto de esferas. En este caso no hay puntos ni esferas auxiliares, por lo que el proceso de borrado y la marcha de Jarvis no son necesarios.

3.3 Algoritmos jerárquicos

Para mejorar los procesos de búsqueda y localización es posible usar una estructura jerárquica de datos.

La idea de usar este tipo de estructura de datos ha sido usada para realizar triangulaciones en el plano; partiendo de un triángulo auxiliar inicial se genera toda una estructura jerárquica en la que el orden de precedencia es establecido por la relación geométrica de inclusión. En otras palabras los triángulos son "hojas" en un árbol en el cual las ramas son los triángulos que los contienen.

Traducido a un esquema incremental de generar, por ejemplo, triángulos de Delaunay, ésto significa que al agregar un punto, los triángulos que lo tienen como vértice son "hojas" de los triángulos donde cayó el punto.

Hay otros antecedentes de algoritmos jerárquicos en problemas relacionados al que constituye la parte central de este trabajo, por ejemplo, para construir teselizaciones de Voronoi, T. Ohya, M. Iri y K. Murota [24] ofrecen dos diferentes algoritmos incrementales con estructuras jerárquicas, uno considerando el dominio a ser dividido como un cuadrado unidad y dividiéndolo en cuadrados de lado $1/k$, con k arbitrario, y otro formando un árbol cuaternario con cuadrados similares.

Para el caso de triángulos de Delaunay, O. Palacios y B. Cuevas [25] desarrollaron una estructura jerárquica similar a un árbol cuaternario.

Los orígenes de la idea se pueden atribuir al trabajo de H. Samet [29] sobre los árboles cuaternarios pensados para usarse en técnicas de tipo "raster" de dibujo computacional, tanto para el propio dibujo como para el almacenamiento de la información.

En la figura 22 se puede ver cual es la idea básica, se tiene el plano dividido en cuartos y cada uno de ellos en cuartos y así hasta que cada cuarto es un pixel. Para ubicar un punto se averigua en cual de los cuartos está y al hacerlo la búsqueda se reduce solo a esa parte y así sucesivamente, lo que disminuye notablemente el tiempo de búsqueda.

Lo importante para poder aplicar un esquema como este es entonces poder establecer la relación de precedencia (lo que hace a un elemento rama y a otro hoja). En el caso de la figura es simplemente estar contenido en, un cuadro es hoja de otro, lo sucede, si está contenido en él.

Pasar de cuadrados a triángulos no es ningún problema, ya que ambas figuras llenan la región del plano, inclusive los triángulos lo hacen con mayor facilidad. Algo similar se puede decir de la generalización a varias dimensiones.

Sin embargo para hacer aún más eficiente el algoritmo, por ejemplo el algoritmo de generación de triángulos de Delaunay incremental, lo ideal es que la jerarquía sea también "cronológica", es decir, que los triángulos generados al principio, en vez de ser destruídos sirvan como árbol. Para que esto suceda la relación de precedencia ya no puede ser simplemente "estar dentro de" sino "ser afectado por"; en otras palabras, que la inclusión del punto a localizar provoque que el triángulo ya no sea más de Delaunay.

En este punto es cuando se llega a un nexo con las esferas de cobertura,

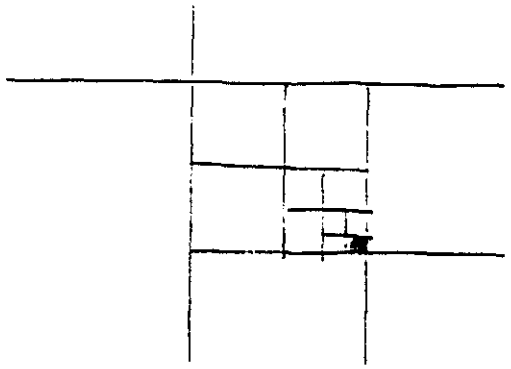

Figura 22

ya que la forma de saber si el punto afecta o no al triángulo es ver si afecta o no a su círculo circunscripto, que no es otro que la versión en dos dimensiones de la correspondiente esfera de cobertura.

De esta manera, si se quiere tener una estructura jerárquica con las esferas de cobertura, basta con poner la condición de que la relación de precedencia de una esfera respecto a otra sea compartir puntos de generación y "cronológicamente" haber sido generada a partir de que uno de sus puntos de generación "afecta" a su predecesora. Un primer intento en este sentido ha sido ya realizado por Traversoni [34], aunque el programa no ha sido aún optimizado.

Como es posible imaginar, utilizar una técnica como la expuesta solo tiene ventajas cuando se trabaja con un gran número de datos, y tiene la desventaja de utilizar más memoria (en el caso análogo de los triángulos, el triple) debido a que todas las esferas generadas durante el proceso deben ser guardadas.

La forma en la que se establece la relación de precedencia con las esferas no obliga a utilizar una esfera inicial, como era el caso con los triángulos, sino que el razonamiento es igualmente válido para esquemas sin esfera auxiliar inicial.

Para simplificar aún más el procedimiento de búsqueda es conveniente agregar a las características que se almacenan de cada esfera (centro, radio y puntos que la generan) dos indicadores, uno de si la esfera ha sido afectada o no y otro que indique el lugar o nivel en el árbol jerárquico.

Es importante destacar el papel que puede desempeñar el hecho de

usar algoritmos de múltiple aplicación, como es un árbol cuaternario tal como el que se presenta, ya que se puede en el mismo programa aplicar el mismo algoritmo para varios fines, por ejemplo en este caso para realizar la ubicación de un punto, pero también para sombrear superficies o denotar gráficamente cualquier característica deseada. En estos casos el aumento de velocidad en el algoritmo puede ser muy considerable, así como en simplicidad del programa.

3.4 La solución adoptada

Se ha intentado construir un híbrido de las opciones anteriores incorporando las ventajas más sobresalientes de cada una. La justificación del mismo, así como alguno de los teoremas precedentes fueron ya publicados en 1990 Traversoni [33] y varias versiones computacionales han sido implementadas para aplicaciones concretas.

Aún cuando la teoría es general para las dimensiones que se quiera se han generado dos programas, uno para dos y otro para tres dimensiones, que son los casos más usados.

3.4.1 El caso bidimensional

Se trata en este caso de generar círculos de cobertura a partir de un conjunto dado de puntos en el plano. Se puede partir de dos hipótesis iniciales; en la primera se conoce de antemano todo el conjunto de puntos y en la segunda éstos se van conociendo uno por uno.

Solo en la primera hipótesis se puede usar un algoritmo jerárquico o algoritmos que usen un círculo cobertor de todo el conjunto; es de aclarar que podría haber una hipótesis intermedia en la cual se predeterminara la zona donde pueden caer los puntos del conjunto, aún cuando no se los conociera en particular a cada uno por adelantado.

El primer paso del programa para construir círculos de cobertura varía entonces según estas dos posibilidades:

En el primer caso se toma el punto con máxima x (maxx), el punto con mínima x (minx) y el punto con máxima y (maxy) para generar la primera circunferencia tomando puntos alejados del baricentro del conjunto por ejemplo el doble que los mencionados anteriormente. En el segundo

caso simplemente se toman los tres primeros puntos, siempre que no estén alineados.

La razón de la elección en el primer caso es que, si bien ese primer círculo no cubre a todo el conjunto, sí lo hace casi totalmente, y son necesarios pocos círculos más para completar la cobertura.

3.4.2 El ciclo principal: la búsqueda

El ciclo principal consta de dos ciclos anidados, el primero simplemente introduce los puntos del conjunto uno a uno, excluyendo los tres usados en la primera circunferencia.

El segundo halla la potencia del punto respecto a la última circunferencia considerada, existiendo tres posibilidades:

a) La potencia es menor que cero. En ese caso se aplica el algoritmo de construcción de nuevas circunferencias que llamaremos algoritmo A.

b) La potencia es cero. Se agrega el punto a la lista de generadores de la circunferencia y se pasa al siguiente punto.

c) La potencia es mayor que cero. En este caso se procede como se relató más arriba usando el algoritmo de búsqueda orientada

Hay dos formas de salir del ciclo, una es que se aplique el algoritmo A, la otra es que no sea posible aplicarlo por no darse en ningún caso la condición de potencia negativa. En esta eventualidad se aplica otro algoritmo de construcción de círculos que llamaremos el algoritmo B.

3.4.3 El algoritmo A

La hipótesis de partida en este caso es que ya ha sido encontrado un círculo afectado por el punto, se debe buscar entonces como primer paso otros posibles círculos afectados por el mismo punto.

Teorema 3.4.1 *Si un punto P afecta a un círculo, entonces todo otro círculo afectado por él tiene en común con el original al menos un punto generador, o existe un círculo intermedio con puntos generadores comunes a ambos.*

Demostración: Sean, ver figura 23, ABC y DEF los generadores de los círculos afectados considerando aislado el conjunto $ABCDEF$ su cerradura convexa está por lo demostrado anteriormente totalmente cubierta por círculos de cobertura, con la particularidad de que la unión de las figuras circunscritas en ellos, tales que sus vértices son generadores, es la propia cerradura convexa. ABC y DEF forman un conjunto no conexo por lo que existen figuras intermedias entre ambos, cuyos vértices son algunos de los seis puntos mencionados que al unirse a ellas forman una figura conexa y convexa. Los círculos que cubren a esas figuras son los que pide el teorema.

Par poder realizar eficientemente la búsqueda anterior es necesario almacenar a cuáles circunferencias pertenece cada punto, y cuáles puntos a cada circunferencia (generadores de los círculos de cobertura) lo cual se hace m diante arreglos de listas ligadas.

Esto es útil también para los pasos subsecuentes del algoritmo y si bien es costoso en memoria, agilita el procesamiento, lo cual, dado el crecimiento constante de la capacidad de memoria de las máquinas, es la opción más conveniente.

Mediante un algoritmo que se describirá más tarde se deben almacenar también las aristas que forman la cerradura convexa del polígono inscrito en cada uno de los círculos.

Disponiendo de tales listas el segundo paso del algoritmo A es formar una lista de las aristas no repetidas pertenecientes a los círculos afectados por P.

Los nuevos círculos son generados en un tercer paso que consiste en generarlos con P y los extremos de las aristas de la lista fabricada en el paso 2.

3.4.4 Rutina para hacer lista de aristas

El primer caso que se puede presentar es muy sencillo, se trata de cuando el círculo está determinado por solo tres puntos, en tal caso cualquier combinación de dos puntos forma una arista del polígono que sirve de cerradura convexa a los tres puntos.

No es tan sencillo cuando el círculo está definido por más de tres puntos, caso en el cual lo anterior no se da, ya que algunos de los segmentos que se obtendrían no son aristas sino diagonales, es necesario entonces recurrir a una sencilla rutina para determinar las aristas.

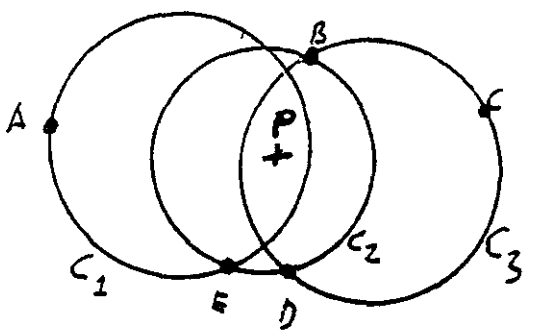

figura 23
Se representa el caso mas desfavorable, tanto C_1 como C_3 son afectados por P, sin embargo sus puntos de corte no son generadores, existe entonces C_2 que sí tiene los puntos en común dentro de sus generadores.

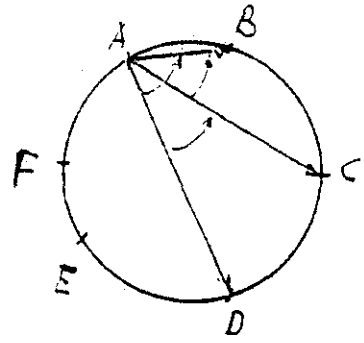

Figura 24

Para ello se toma uno cualquiera de los puntos y se halla el más cercano a él, entre los dos forman siempre una arista, podría repetirse el procedimiento varias veces hasta completar la cerradura convexa, sin embargo, es más práctico recurrir a hallar el producto vectorial entre el segmento y todos los otros que se forman con uno de sus extremos y un punto de los generadores del círculo; ordenando los antedichos puntos utilizando sus productos vectoriales, se obtiene la secuencia de la poligonal que forma la cerradura convexa (ver figura 24). Si el orden en que se tienen los puntos de la figura es por ejemplo $ADCBEF$, se calcula $AD \wedge AC$ y $AD \wedge AB$ como tiene el mismo signo se hace $AC \wedge AB$ como tiene el mismo signo se sabe que $DCBA$ es un orden correcto y así se sigue.

Se toma A, el punto más cercano es B, se realizan los productos vectoriales entre las direcciones (normalizando los vectores) AB y AC, AD,... y se ordenan en forma creciente los resultados obteniéndose la sucesión de puntos que da la poligonal $ABCDEF$ Otra opción es considerar todas las posibles esferas construíbles con todos los puntos tomados de $n+1$ en $n+1$. Posteriormente se eliminan aquellos que contengan a alguno de los puntos A, B, C, D, o F, un procedimiento similar a aplicar el algoritmo de Lawson para construir triángulos de Delaunay.

3.4.5 Algoritmo B

Este algoritmo se aplica cuando el punto P ha resultado exterior a todos los círculos.

El algoritmo de búsqueda proporciona la información de que círculo

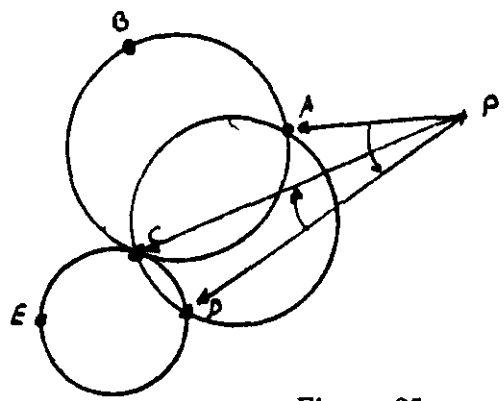

Figura 25
El producto $AP \wedge PD$ es de distinto signo al $PD \wedge PC$

tiene menor potencia respecto al punto; dentro de los puntos que generan a este círculo y en la cerradura convexa de todo el conjunto se encuentra también el punto más cercano del conjunto al punto P. Este punto es seguramente "visto" por P.

El primer paso del algoritmo B es entonces, encontrar ese punto.

A partir de allí se aplica a cada lado del punto la marcha de Jarvis, o una versión adaptada de ella para estas circunstancias, que consiste en hallar los productos vectoriales de la distancia vértice de una arista-punto P con la arista, cuando estos productos cambian de signo se está en la parte oculta.

La lista de aristas con las cuales probar se obtiene de las respectivas listas del círculo más cercano y sus vecinos; si la condición aún no se ha cumplido al agotar sus aristas, se pasa a los vecinos de los vecinos y así sucesivamente; lógicamente solo se consideran las aristas no repetidas, es decir las que forman parte de la frontera de la cerradura convexa.

Se recurre con tanta frecuencia al producto vectorial debido a que se puede realizar en forma muy económica en cuanto a número de operaciones involucradas; en la figura 25 se ilustra el algoritmo B.

En este caso, al igual que en el anterior, es posible, si no se desea usar el producto externo o vectorial, generar todas las esferas que se pueden y eliminar las que contienen a algún punto de V.

Principales variables utilizadas

Para ahorrar en lo posible memoria y usar en mejor medida las posibilidades del Pascal y el C (lenguajes en los que se ha implementado el algoritmo) se hace frecuente uso de listas ligadas.

De esta manera, existe un arreglo de listas ligadas en el que a cada punto se le asignan los círculos que por él pasan.

Otro arreglo recíproco del anterior asigna a cada círculo los puntos que lo generan, ésto, que es una redundancia, ahorra sin embargo varias costosas búsquedas durante el transcurso del proceso. El uso de listas ligadas, por otra parte, permite tener mayor flexibilidad y ahorrar memoria, cosa que no se haría de usarse arreglos de reales o enteros.

En un arreglo fijo se guardan las coordenadas del centro de cada círculo, su radio y una bandera provisional que marca en el algoritmo de búsqueda si ha sido considerado y que por lo tanto se reinicializa cada vez que se reinicia el proceso de búsqueda, y otra bandera que en el caso de algoritmos jerárquicos indica el nivel en el árbol.

De no usarse algoritmos jerárquicos, este arreglo, así como los demás, deben ser actualizados, borrando los círculos que van siendo afectados en el proceso y compactando el arreglo; para este proceso es particularmente ventajoso tener listas ligadas.

Cuando se usan algoritmos jerárquicos no se hace este proceso, pero en contrapartida existe un algoritmo que realiza la búsqueda jerárquica previo al descrito anteriormente.

Algoritmo de búsqueda jerárquica

Dado el punto P, busca primero en las de primer nivel, en la que resulta negativa la potencia busca en el siguiente nivel y así sucesivamente hasta llegar al último nivel de círculos que no han sido afectados; en él, si alguno tiene potencia negativa respecto a P, aplica el algoritmo A, de no ser así aplica el B.

Para hallar en que caso se encuentra, aplica el algoritmo de búsqueda a los círculos dependientes del último negativo del nivel anterior.

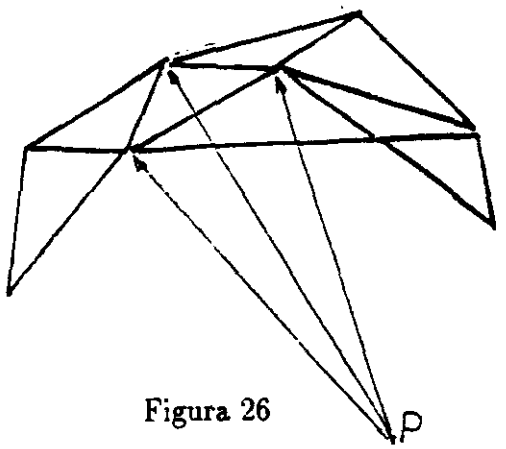

Figura 26

3.4.6 El caso tridimensional

Generalizar el algoritmo anterior para el caso tridimensional (esferas) no ofrece mayor dificultad.

Para hacerlo se debe recordar que las esferas son determinadas por al menos cuatro puntos en vez de tres, como ocurre con los círculos. El equivalente de las aristas es ahora las caras, de modo que cambiando aristas por caras se tiene la versión tridimensional de los algoritmos anteriores.

El problema más importante a resolver para poder hacer lo anterior es adecuar la rutina que usa productos vectoriales.

Hay dos formas de solucionarlo, una es proyectar sobre un plano los vectores normales a las caras y proceder como si fueran las aristas del caso anterior, y otra es considerar los volúmenes determinados por los módulos de los vectores resultantes del triple producto vectorial formado (ver figura 26) por tres aristas del tetraedro que contiene a P y una cara, sumarlos acumulativamente en la marcha y la disminución de la suma tiene la misma interpretación que el cambio de signo en el plano.

Capítulo 4

Interpolantes de Sibson y las esferas de cobertura

4.1 Introducción

El concepto de vecino natural y el de interpolante de Sibson que se introducen a continuación se relacionan muy naturalmente con las esferas de cobertura, como se demostrará en el presente capítulo. Esta relación puede ser aprovechada de dos formas, una para simplemente hacer más fácil la puesta en práctica del interpolante de Sibson y la otra, reformulando el interpolante como polinomio de Bernstein, la cual abre una gran gama de posibilidades, relacionando todo lo anterior con la teoría de splines y el algoritmo de De Casteljau.

4.2 La partición generada por las esferas de cobertura

Recordemos algunas definiciones y notación introducidas en el Capítulo 2. El conjunto $V = \{v_1, v_2, \ldots, v_m\}$ es una colección dada de puntos en el espacio euclidiano $\mathbf{E}$ de dimensión n.

Denotamos por $\mathbf{S} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_s\}$ a la colección de esferas de cobertura de V. Cada $\mathcal{E}_j$ es una esfera abierta que contiene al menos $n+1$ puntos de V en su frontera y que no contiene a ningún punto de V en su

interior.

En el conjunto $\mathbb{S}$ están incluidos los semiespacios abiertos que contienen en su frontera a alguna cara de dimensión $n-1$ de la cerradura convexa de V. Tales semiespacios son considerados como esferas de radio infinito con centro en el punto del infinito de $\mathbb{E}$.

Las esferas de cobertura no son conjuntos ajenos, pero podemos construir a partir de ellas una partición $\mathcal{P}$ del espacio $\mathbb{E}$ de la manera siguiente. Para cada subconjunto J de I_s definimos

$$A_J = \{x \in \mathbb{E} : x \in \mathcal{E}_j \text{ si } j \in J, \text{ y } x \notin \mathcal{E}_k \text{ si } k \in I_s \backslash J\}$$

y

$$\mathcal{P} = \{A_J : J \subseteq I_s \text{ y } A_J \neq \emptyset\}.$$

Notemos que $A_\emptyset = V$, porque los elementos de V son los únicos puntos que no están contenidos en el interior de ninguna de las esferas de cobertura.

Teorema 4.2.1 *La colección de conjuntos $\mathcal{P}$ definida arriba es una partición del espacio $\mathbb{E}$.*

La demostración es trivial, porque la unión de las esferas de cobertura es el complemento del conjunto V.

Notemos que algunos elementos de $\mathcal{P}$ son conjuntos abiertos y que otros de ellos contienen una parte de su frontera y, por tanto, no son ni abiertos ni cerrados.

Es claro que 2^s es una cota superior para el número de elementos de $\mathcal{P}$, pero es una cota muy sobrada. En la práctica se observa que el número de elementos de $\mathcal{P}$ es más bien una función lineal del número s de esferas de cobertura de V.

Como $\mathcal{P}$ es una partición de $\mathbb{E}$, para cada x en $\mathbb{E}$ existe un único $J = J(x) \subseteq I_s$ tal que A_J no es vacío y que $x \in A_J$.

Definimos el conjunto N_x de *puntos vecinos* de x en V como sigue:

$$N_x = \{v_i \in V : v_i \text{ es generador de } \mathcal{E}_j \text{ para algún } j \in J(x)\}.$$

Es claro que dos puntos del espacio que pertenecen a un mismo conjunto A_J tienen el mismo conjunto de vecinos en V.

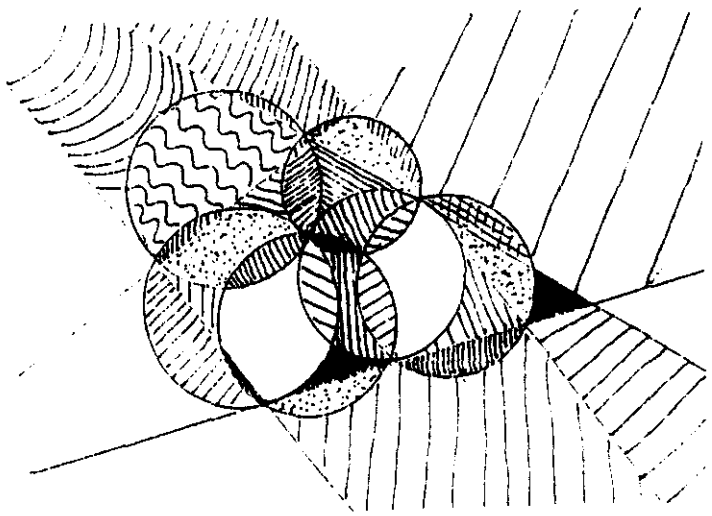

Figura 27
La partición en el plano.

4.3 Funciones interpolantes de Sibson

Sea p un punto que no pertenece a V. Como vimos en el Capítulo 2 el diagrama de Voronoi de $V \cup \{p\}$ se obtiene de la manera descrita a continuación.

Sea T_p el politopo de Voronoi de p en el diagrama de $V \cup \{p\}$ y sean $v_1, v_2, \ldots, v_r$ sus vecinos. Entonces los nuevos politopos de los puntos v_i, denotados por T_i', están relacionados con sus correspondientes politopos T_i en $\mathcal{D}(\mathcal{V})$ por medio de

$$T_i' = \{x \in T_i : x \notin \text{ cerr}(T_p)\}$$

y por tanto tenemos que T_p es la unión de los conjuntos $P_1, P_2, \ldots, P_r$ definidos por

$$P_i = \text{ cerr}(T_i) \setminus \text{ cerr}(T_i)'.$$

Definimos las *coordenadas de Sibson* $c_1, c_2, \ldots, c_r$ de p respecto a sus vecinos en V por medio de

$$c_i = \frac{\text{Vol}(P_i)}{\text{Vol}(T_p)}, \qquad 1 \leq i \leq r,$$

en donde $\mathrm{Vol}(P_i)$ denota el volumen generalizado del politopo P_i. Notemos que

$$\sum_{i=0}^{r} c_i = 1,$$

porque los P_i se intersecan en conjuntos de volumen nulo.

El siguiente resultado básico, debido a Sibson [?], nos dice que los números c_i generalizan a las coordenadas baricéntricas.

Teorema 4.3.1 *Las coordenadas de Sibson del punto p satisfacen*

$$p = \sum_{i=0}^{r} c_i \; v_i.$$

Cuando el número de vecinos de p es igual a la dimensión del espacio más uno, las coordenadas de Sibson son similares a las coordenadas baricéntricas.

Al igual que para las coordenadas baricéntricas, si p tiende a v_i entonces c_i tiende a 1 y las demás coordenadas tienden a cero.

Supongamos que f es una función con valores numericos definida en los elementos de V. Dado un punto p que no está en V sean $v_1, v_2, \ldots, v_r$ los vecinos de p en V. Definimos la *función de interpolación de Sibson*, denotada por S_f, o simplemente S, por medio de la fórmula

$$S(p) = S_f(p) = \sum_{i=0}^{r} c_i \; f(v_i),$$

donde las c_i son las coordenadas de Sibson de p con respecto a sus vecinos en el diagrama de Voronoi de V.

4.4 La función de Sibson en la teoría de Splines

Si en lugar de utilizar los enteros $1, 2, \ldots, r$ como índices de los vecinos de p en el diagrama de V, usamos los vectores (o multi-índices) $e_1, e_2, \ldots, e_r$, donde $e_k = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$ con el 1 en la k-ésima posición. entonces podemos expresar la función interpolante de Sibson como un polinomio

de Bernstein en tantas variables como vecinos la definan. La expresión correspondiente es

$$S_f(p) = \sum_{|\mathbf{i}|=1} B^r_{\mathbf{i}}(\mathbf{p}) f(v_{\mathbf{i}}),$$

donde $B^r_{\mathbf{i}}(\mathbf{p})$ es un polinomio de Bernstein de grado r en r variables, una por cada vecino natural de p, y las p_i son las coordenadas de Sibson de p. Es claro que $\mathbf{i}$ solamente puede ser alguno de los e_k, debido a la condición $|\mathbf{i}| = 1$.

La expresión de $B^r_{\mathbf{i}}(\mathbf{p})$ será

$$B^r_{\mathbf{i}}(\mathbf{p}) = \binom{r}{\mathbf{i}} p_1^{i_1} p_2^{i_2} \cdots p_r^{i_r},$$

que se puede también poner como

$$\frac{r!}{i_1! i_2! \cdots i_r!} p_1^{i_1} p_2^{i_2} \cdots p_r^{i_r}.$$

Como se ve, en este caso los polinomios son simplemente otra forma de expresar las coordenadas de Sibson. De esta manera $p_1 + p_2 + \cdots + p_r = 1$, también $|\mathbf{i}| = r$ y $p_1, \ldots p_r$ son las coordenadas de Sibson respecto a una colección de puntos vecinos $v_1, v_2, \ldots, v_r$. La expresión es además trivial, ya que en este caso el coeficiente binomial es uno, ya que los p_i son iguales a los c_i.

Pese a la trivialidad, expresar de esta manera el interpolante no es en vano, ya que nos permite generalizar posteriormente la idea, tanto en teoría de Splines como para construir un algoritmo de De Casteljau. Véase, por ejemplo, que si $r(p) \geq 1$ entonces más de un punto v_{e_j} intervendría en cada sumando, lo cual sería equivalente a agregar un punto intermedio entre ellos, o mejor aún, lo que se propone más adelante es agregar una condición a la derivada direccional en dicha dirección, que es justamente en donde "falla" el algoritmo de Sibson.

El hecho de que exista una expresión polinomial de Bernstein para la función de Sibson es un resultado muy importante, ya que permite usar mucho de lo que se ha elaborado en la teoría de Splines. De esta manera para cada conjunto A en la partición $\mathcal{P}$ se tiene un polinomio definido en él en función de las coordenadas locales de los puntos de V que son vecinos naturales de los puntos de A. Se puede así definir un espacio de splines

formado por los polinomios definidos en cada parte de la partición $\mathcal{P}$, cuya expresión es la anteriormente citada.

En la sección siguiente se enumeran algunas propiedades del interpolante, una de ellas, la conformabilidad en los puntos de V, es importante como base para la aplicación a teor´ ia de Splines que se mencionaba más arriba.

Para fijar ideas se definirá el espacio de splines como un espacio de splines de vértices.

Sean $-1 \leq r_1 \leq \ldots \leq r_s$ y $g \geq 0$ enteros y sea $\pi_g = \pi_g^s$ la colección de todos los polinomios con grado g en s variables. Nos referiremos a algunas funciones contenidas en la colección $S_g^s = S_g^s(\mathcal{P})$ tales que la restricción de cada función a cada elemento de $\mathcal{P}$ pertenece a π_g y de manera que, para $k = 0, \ldots, s-1$, en cada punto x de cada A en $\mathcal{P}$ todas las derivadas parciales de orden r_{s-k}

$$D^\alpha f(x) = (\frac{\partial}{\partial x_1})^{\alpha_1} \cdots (\frac{\partial}{\partial x_s})^{\alpha_s} f(x),$$

donde $\alpha = (\alpha_1, \ldots, \alpha_s)$ y $\mid \alpha \mid = \alpha_1 + \ldots + \alpha_s = r_{s-k}$, existen. De esta manera, toda función en S_g^r es llamada spline de grado g y suavidad de orden r en la partición $\mathcal{P}$, y será un super spline si los r_i no son idénticos. $f \in S_g^r$ es un spline de vértices con partición $\mathcal{P}$ si $supp(f)$ contiene por lo menos un vértice de V y el interior de supp(f) contiene a lo sumo un vértice de A.

En el presente caso el soporte será un punto de V y la unión de todos los $A \in \mathcal{P}$ que lo tienen en su frontera. Las funciones son obviamente las funciones interpolantes de Sibson, en este caso en su expresión de polinomios de Bernstein, solo que en este caso $r \neq 1$, y los vectores i pueden tener más de una componente diferente de cero. Se trata justamente de imponer la condición de que en cada punto v de V donde, como se recordará, existían problemas con la continuidad, el plano tangente a la superficie aproximada contenga a las derivadas direccionales que apuntan a los respectivos vecinos naturales de v.

4.5 Algunas propiedades

i) Diferenciabilidad. Las derivadas direccionales de la función inter-

polante de Sibson se pueden expresar como

$$D(S(u)) = \sum_{i=1}^{r} \partial^i f(u) B_i^r(d)$$

donde $d = u_2 - u_1$, los u_j son puntos del dominio A y

$$\partial^i f(u) = \frac{\partial^{|i|}}{\partial u_1^i \cdots u_r^i} F(u)$$

ii) Continuidad. Farin [20] demostró que el interpolante de Sibson es C^1 continuo excepto en los puntos donde hay datos, en las direcciones de sus vecinos naturales, en donde solamente es C^0 continuo.

iii) Idempotencia. El interpolante no es idempotente, usaremos esta propiedad más adelante, ya que al agregar un punto a V el interpolante cambia en su vecindad.

iv) Conformabilidad. Se trata de que tanto los valores de la función interpolante como los de sus derivadas direccionales en un punto de V coincidan.

4.6 Algoritmos derivados a partir de Sibson

Farin [20] presentó un interpolante derivado del de Sibson, haciendo que los puntos de V sean proyecciones de simplejos de Bèzier, de esta manera se puede resolver el problema siguiente: dados los puntos de V y los planos tangentes en ellos, encontrar una superficie que los interpole. Como resultado se obtiene una superficie C^1 continua con precisión cuadrática.

Se puede también considerar que los puntos de V forman una malla de control de esta manera, y aprovechando que el interpolante de Sibson se puede expresar como polinomio de Bernstein, se puede plantear un algoritmo de De Casteljau para aproximar una superficie, con un procedimiento similar al que se usa con triángulos de Bezièr. Una descripción detallada de como implementar este algoritmo se encuentra en el capítulo siguiente.

Por último, como se vió más arriba, es posible incorporar todo a la teoría de Splines. Conviene recalcar que en todos los casos el incorporar las esferas de cobertura al interpolante simplifica los cálculos y además es

teóricamente más correcto, ya que con ellas se conforma la partición en la que quedan definidos las diferentes versiones del interpolante.

Merece una mención también la posibilidad de usar el interpolante en las aplicaciones del método de elemento finito, la cual ha sido mencionada por nosotros en Traversoni [35]. El resultado es muy parecido a usar triángulos de Delaunay como elementos, puesto que el sistema de ecuaciones queda muy similar.

Capítulo 5

Algoritmos de interpolación

En este capítulo se explican los diferentes modos de implementar computacionalmente las ideas expresadas en el capítulo anterior, con la finalidad de construir superficies uniformes en 3D a partir de una partición en 2D, generada con un conjunto arbitrario de puntos. Las superficies no uniformes, cerradas o de otro tipo, serán tema del siguiente capítulo.

La función interpolante de Sibson, como se explicó en el capítulo anterior, genera valores punto a punto. Para poder visualizar los resultados se deberá posteriormente ubicar esos puntos, de modo de poder dibujar la superficie que los contiene.

5.1 Implementación del algoritmo de interpolación

Como se vió en el capítulo anterior, se puede realizar la interpolación de diversas formas, desde simplemente aplicar el algoritmo de Sibson hasta casos más sofisticados, usando polinomios de mayor grado.

En el caso de que la interpolación se realice simplemente con el algoritmo de Sibson, queda aún la pregunta de como implementarlo, para ello distinguiremos dos etapas, la aplicación del interpolante de Sibson y la repetición de la misma en base a las consideraciones relatadas en el capítulo anterior. Esto es generar un algoritmo del tipo del de De Casteljau introduciendo nuevos puntos y repitiendo el proceso.

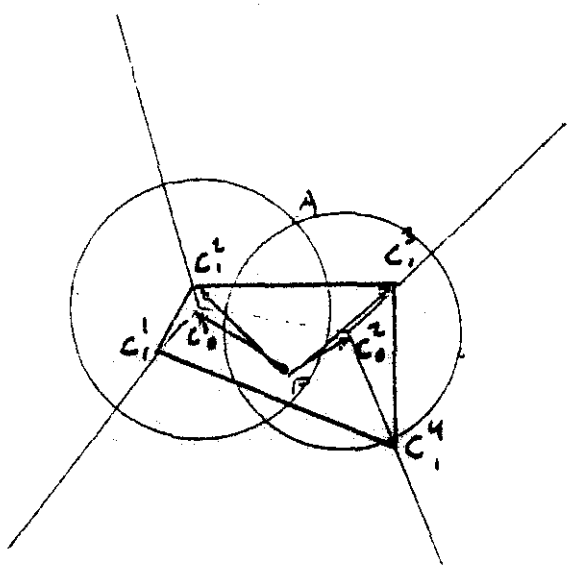

Figura 28
Cálculo del peso de A usando triángulos

Se debe remarcar el hecho de que en el caso de la utilización del algoritmo de De Casteljau se debe elegir los puntos de V, no como puntos de la superficie sino como puntos de control, al igual que en otras aplicaciones del citado algoritmo.

Queda entonces un tercer caso que es utilizar polinomios de orden mayor para aproximar la superficie, suponiendo que los puntos de V son de la superficie a aproximar. En este caso lo que se puede agregar al algoritmo de Sibson es su utilización en el marco de la teoría de Splines, imponiendo la condición de que desaparezcan las discontinuidades en las primeras derivadas que tiene en algunas direcciones el algoritmo de Sibson.

En el primer caso, los pesos con los que se realiza el promedio ponderado usado en la interpolación, se definen como cocientes de áreas. Los polígonos cuyas áreas se usan son tales que todos sus vértices son centros de algún círculo de cobertura, este hecho se usa para hallar las áreas.

Existen dos formas de hallar las áreas, una es el método de suma de trapecios y la otra considerando las áreas como sumas y restas de triángulos tales que uno de sus vértices sea P. En la figura siguiente se ilustra un ejemplo. Sean ahora C_i los centros que se usan y P el nuevo punto, cada

triángulo usado se calcula de la forma

$$A = (C_i - P) \bigwedge (C_{i+1} - P).$$

donde A es el área del triángulo.

El área total del nuevo polígono de Voronoi en la figura 28 $C_1^1C_1^2C_1^3C_1^4$ se encuentra como suma de los triángulos $PC_1^2C_1^2$, $PC_1^2C_1^3$, $PC_1^3C_1^4$, $PC_1^4C_1^1$. Para saber como calcularlos se usa la preordenación que en la radiación realiza el algoritmo que construye los círculos. En otras palabras, al insertar el punto P, el algoritmo de construcción encontró los círculos que éste afectó (en la figura 28 los dos dibujados) y lógicamente, al tener los círculos se tienen los puntos que los generan (en la figura A, B, CyD). Para construir los nuevos círculos el algoritmo encuentra las parejas adecuadas de vecinos que junto con P los generan. Se puede aprovechar entonces esta circunstancia para ordenar los centros $C_1^1, C_1^2, C_1^3, C_1^4$, poniendo como condición que si los círculos que les corresponden tienen un punto generador en común, entonces los centros son consecutivos en la radiación con centro en P.

El siguiente problema a resolver es hallar las áreas de los trozos de intercepciones de polígonos con los que se va a realizar la ponderación, tal como sería en la figura 28 el $C_0^1, C_0^2, C_1^1, C_1^2$ y hacerlo con un sistema similar.

El primer paso es identificar cuales son los vértices de cada polígono, lo cual servirá de paso para ordenar los vértices de cada polígono en la radiación. Los vértices de un polígono tienen como característica común ser centros de círculos que pasan por un punto de V. La consecutividad se obtiene con la misma técnica que en el caso anterior, teniendo en cuenta además que, si hay dos centros con subíndice cero, éstos son siempre consecutivos.

Para hallar el área basta con hallar las áreas de los triángulos correspondientes y sumarlas, y por diferencia se obtendrá el area cerrada por el polígono.

Obsérvese que el razonamiento es totalmente general, que aparte de la operación del producto externo emplea solo operaciones lógicas y que en ningún momento halla intersecciones de rectas o distancias especiales que no sean parte de lo ya hallado en el algoritmo de construcción de los propios círculos.

Obsérvese además que solo se usan las coordenadas de los centros de los círculos, y que éstas ya han sido previamente calculadas y almacenadas por el programa, ésto, aunado a lo anterior, permite realizar los cálculos muy rapidamente, ya que se hacen sobre todo operaciones lógicas, y las operaciones aritméticas se minimizan.

Esto nos permite en forma general calcular la expresión $S_f(P)$ del teorema 3.4.1, lo cual se hará ahora en forma reiterada eligiendo al igual que en otras aplicaciones del algoritmo de De Casteljau, puntos intermedios; en este caso los baricentros de los generadores de las esferas afectadas por P.

En la figura se ilustra un paso intermedio en una aplicación bidimensional.

5.2 Algoritmos locales sin memoria

El propio nombre indica las dos principales características de estos algoritmos, la primera es que trabajan en base a consideraciones locales, es decir, que si se quiere aproximar en un punto se usan solo los puntos que determinan a los círculos afectados por ese punto. Sea así C el conjunto de círculos de cobertura que cubren al plano y P un punto de ese plano. Los puntos cuyos valores serán usados en la interpolación serán los de los círculos C_i tales que $P \in C_i$.

La segunda característica es el trabajar "sin memoria", es decir, que los círculos generados en el proceso de aproximación son deshechados cuando se cambia de punto P.

La aplicación de este algoritmo presupone que se dispone de un conjunto arbitrario de puntos V y su correspondiente conjunto de círculos de cobertura C y lógicamente de los valores asignados a la variable a interpolar (por ejemplo una altura) en cada uno de los puntos de V. Se dispone además del número de aproximaciones que se quieren realizar. Con estos datos, y a criterio del usuario, se determinan los puntos donde se desea que el algoritmo provea de resultados de interpolación.

Al insertar un punto P se localiza, usando el algoritmo de construcción de círculos, los círculos C_i a los que pertenece; sin embargo, no se generan los círculos que él determina, sino que se localizan los baricentros B_i de las figuras descritas en el capítulo 4 formadas con los círculos C_i afectados

por P , estos puntos sí son usados para construir nuevos círculos, posteriormente se verifica en cuales de los nuevos círculos de encuentra P. Sean éstos los C_i donde el supraíndice indica que se está trabajando con círculos provenientes de una primera interpolación.

El proceso se continúa el número de veces predeterminado, obteniéndose una sucesión $C_1, \ldots, C_m$, donde m es el número predeterminado de aproximaciones. En la última sí se incluye a P y se obtiene el valor correspondiente. En todos los casos los valores son obtenidos con el procedimiento descrito en 4.1.2.

Puede darse el caso, sin embargo, que no se tenga un especial interés en un punto determinado, sino que se quiera la superficie completa, en ese caso bastará con aplicar a todas las zonas $A \in \mathcal{P}$ el proceso descrito en el capítulo anterior.

En la figura 29 se ven sucesivos pasos del proceso. En el primer dibujo se ve el punto P localizado y en la segunda los puntos B_1 y B_2 baricentros y los círculos a que dan origen.

Cuando se toma un nuevo punto P los puntos B_i y los círculos C_i son olvidados y se reinicia el proceso con el nuevo punto con el fin de ahorrar memoria y tiempo.

En este tipo de algoritmos, la aproximación es igual en todos los P, ya que los círculos auxiliares no son almacenados. Como se lleva un orden de inserción de puntos, la localización de los círculos afectados se hace mucho más sencilla que si se agregaran en desorden.

En el programa, como se sigue una aproximación tipo barrido, se determina una malla de puntos en los que se quiere tener valores y luego se dibuja en perspectiva la linea que une dichos puntos; la citada malla es dada por una ley de formación y para ahorrar espacio se realiza el dibujo al mismo tiempo que se calcula, de modo de ni guardar los puntos de la malla ni más de dos de los que van formando el dibujo.

5.2.1 Entrada de datos

Lo primero que se ingresa son las coordenadas en 3D (x, y, z) formadas con las coordenadas x y y de los puntos de V, y z que es el valor correspondiente de la función que se desea interpolar, éstos son almacenados en un arreglo de $m \times 3$ que se denomina a efectos del programa como PTS. Con estos puntos se corre el programa de construcción de los círculos de cobertura

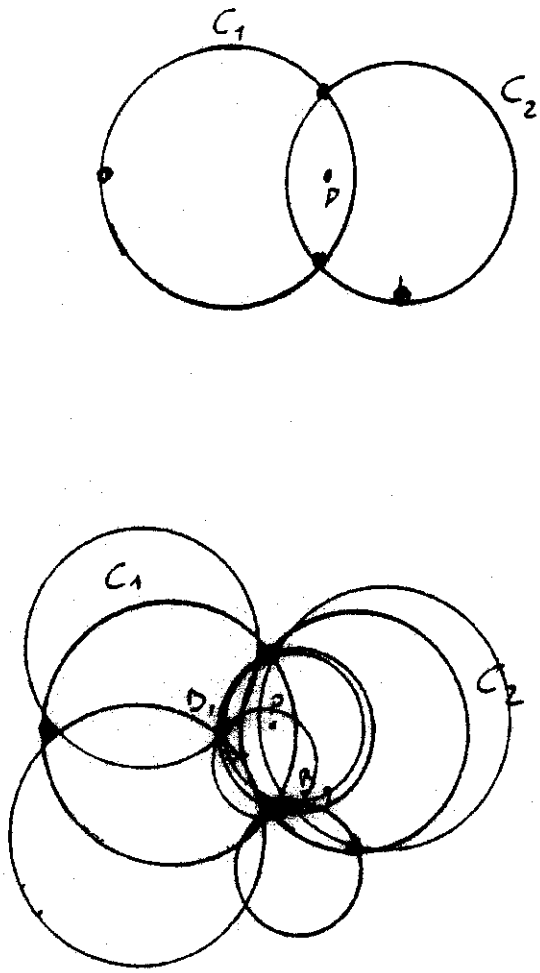

Figura 29

(usando las coordenadas x y y, en el arreglo $PTS[i,1]$ y $PTS[i,2]$). Se tiene de esa manera el conjunto de círculos base que son los que se han denominado hasta ahora C y que a efectos del programa se encuentran en el arreglo $CIRC$ el cual contiene centro y radio de los círculos.

Adicionalmente se crean dos arreglos de listas ligadas que contienen los círculos que pasan por un punto, y los puntos que están en la circunferencia de cada círculo. En el programa éstos son los arreglos $CFPUNT$ y CFA.

Para la opción jerárquica se tienen banderas en $CIRC$, adicionalmente a las otras características, las cuales indican si la esfera ha sido afectada o no.

Los otros datos que se piden al usuario en forma interactiva son el grado de aproximación que va a desear, es decir, cuántas veces quiere que se repita en cada punto el proceso de aproximación; este dato se guarda en el entero $APROX$. También se requiere de los datos necesarios para crear la malla de puntos en los cuales se obtendrán resultados de la interpolación. Estos datos son simplemente la separación entre los puntos, al darlo, el programa irá generando puntos cuya separación, tanto en x como en y será el número indicado, comenzando por el extremo inferior de la zona en estudio (en la y mínima) y terminando en el extremo superior de la misma, respetando del mismo modo los límites en x. Este dato se encuentra almacenado en la variable SEP.

5.2.2 Proceso de interpolación

Tomando los puntos que se generan (pero no se almacenan) con el proceso de cuadriculado anterior, se procede a interpolar para asignar valores en ellos. Para no almacenar todo el cuadriculado solo se va guardando el punto en el que se va y el anterior (los cuales se van generando uno a uno).

Dado un punto de la malla se determina a que círculos pertenece, luego se encuentran los baricentros de cada una de las figuras inscritas en cada uno de ellos y se los agrega provisionalmente a PTS, creando por lo tanto modificaciones a $CIRC$, $CFPUNT$ y CFA. Se dice provisionalmente, ya que al terminar el trabajo con ese punto se dejan los citados arreglos igual como estaban al comienzo, antes de agregar los puntos.

El proceso descrito más arriba se repite tantas veces como las indicadas por $APROX$ ($APROX$ veces) y se repite también para cada punto de la malla. Al terminar el proceso con un punto y obtener su valor en z se

lo une con el anterior dibujando la correspondiente línea en perspectiva isométrica u otra, para ello se tienen los vectores de 3 elementos $PACT$ y $PANT$ por punto actual y punto anterior, las posiciones 1, 2 y 3 de esos vectores son los correspondientes valores de x, y y z.

En la figura 30 se ve el diagrama de flujo del proceso.

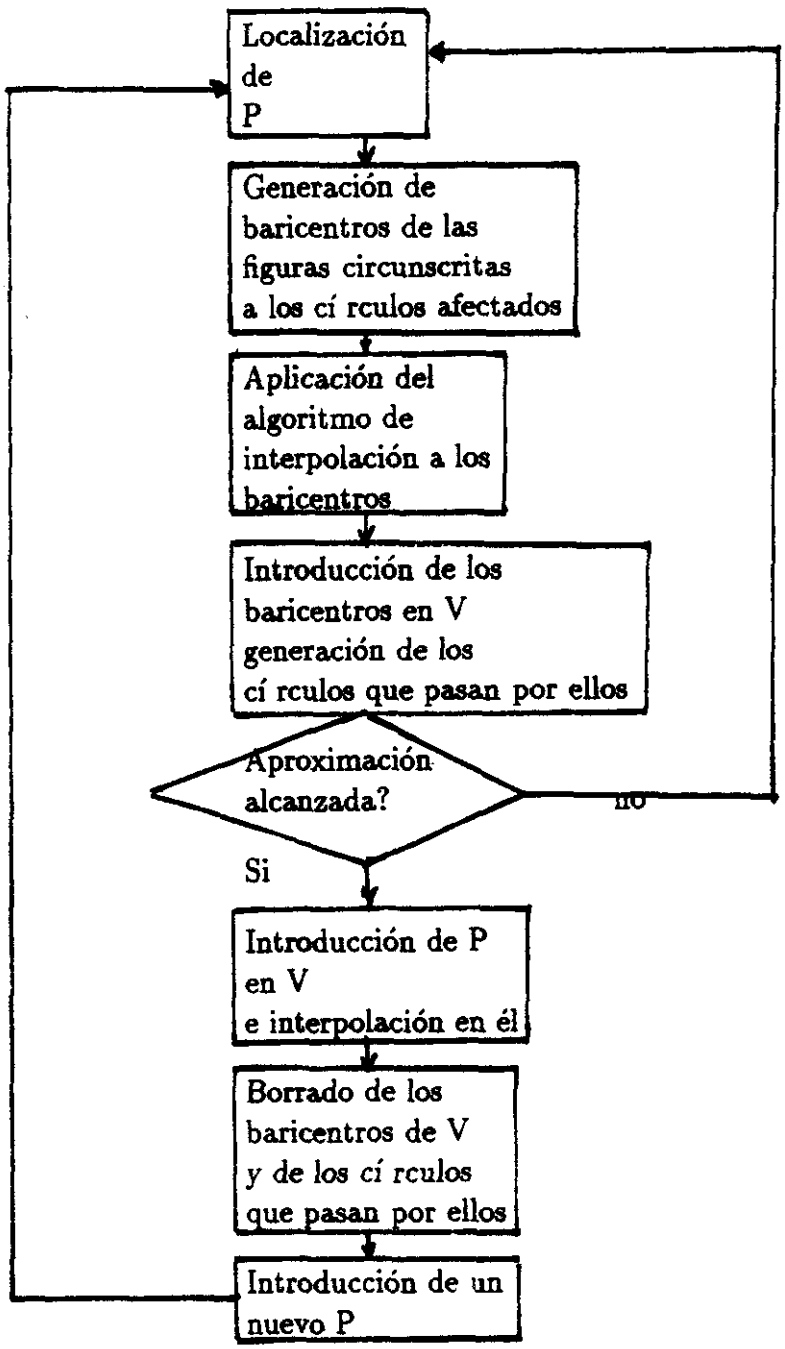
Localización de P
Generación de baricentros de las figuras circunscritas a los cí rculos afectados
Aplicación del algoritmo de interpolación a los baricentros
Introducción de los baricentros en V generación de los cí rculos que pasan por ellos
Aproximación alcanzada?
no
Si
Introducción de P en V e interpolación en él
Borrado de los baricentros de V y de los cí rculos que pasan por ellos
Introducción de un nuevo P

Figura 30

5.3 Algoritmos con memoria

Usando este tipo de algoritmos se obtienen las mayores ventajas del método, sin embargo, dado el tiempo y cantidad de memoria que consumen, no son recomendables ecepto para máquinas con varios procesadores en paralelo, siendo que los algoritmos anteriores podían usarse aún en una PC.

Tal como lo dice el título, se trata de guardar los círculos auxiliares usados para aproximar cada punto agregando los baricentros y el propio punto al arreglo principal, así como todos los círculos. Hacer lo anterior tiene dos particularidades muy destacables :

1) Van a existir puntos que alteren círculos recién creados para el punto anterior, lo cual va a aumentar la aproximación en una o varias veces, puede darse el caso que la aumenten $APROX$ veces dependiendo del círculo que afecten, ésto dependerá de lo cerrado de la malla de puntos en la que se está interpolando.

2) El grado de aproximación será diferente en el centro de la zona que en los bordes (mayor en el centro y menor en los bordes). Esto se debe a que lo planteado en el inciso 1 es más probable que suceda en el centro, donde un punto tiene muchos vecinos ya procesados, que en la periferia que tiene menos. Se pueden lograr alteraciones en lo anterior variando el orden de inserción de los puntos.

En la figura 31 se ilustra como pueden darse los casos anteriores.

Los puntos X_i generan los círculos de la figura al realizar la interpolación, en los puntos $P_1, P_2, \ldots, P_{12}$ es distinta la aproximación, ya que al introducir P_1, éste forma círculos con X_1, X_2 y X_2, X_3, que son afectados por P_2, y ese efecto es cada vez mayor al acercarse al centro de la orradura convexa.

5.3.1 El ingreso de datos

En este programa los datos son los mismos que en el anterior, con la diferencia en como son almacenados, ya que como se ingresan los baricentros al arreglo principal la dimensión de éste debe ser mucho mayor. Es más, no solo los baricentros, sino todos los puntos en los que se aproxima quedan incluidos en dicho arreglo.

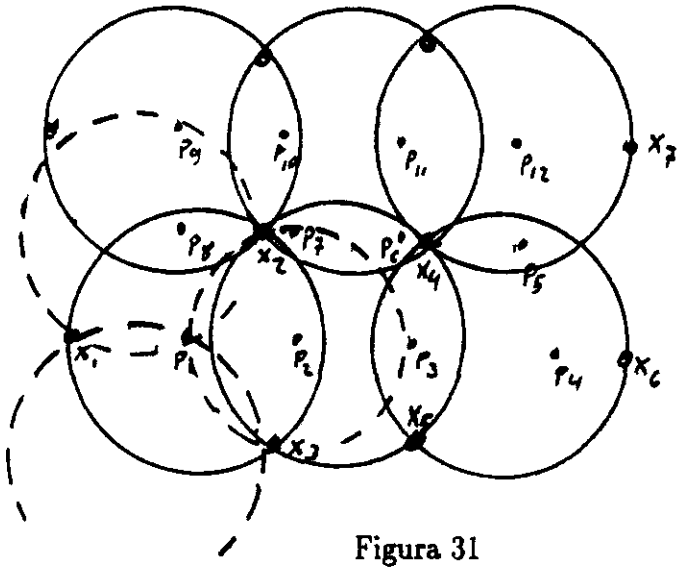

Figura 31

Lo mismo pasa con los círculos, ya que deben ser guardados todos, pero además cada punto, siendo que se enfrenta a una densidad mayor de círculos afectará en promedio a más de ellos, lo cual a su vez aumentará la cantidad de círculos a almacenar.

5.3.2 El proceso de interpolación

Como se dijo anteriormente, al usar este programa todos los círculos pasan a formar parte del arreglo general, de modo tal que, cuando se agrega un punto, éste y todos los baricentros usados para aproximarlo entran al arreglo PTS.

Cuando se agrega el siguiente punto, si se los ha ordenado correctamente o simplemente si se encuentran con un espaciamiento adecuado, afectará algunos de los círculos recién creados para aproximar el punto anterior, lo cual mejorará la aproximación del nuevo punto sin realizar operaciones adicionales.

Lo anterior puede ser visto como una ventaja del método, ya que al hacerlo se están ahorrando pasos de computación. Para lograr ésto, los círculos tienen una bandera que indica a que "generación" pertenecen, el programa verifica que todos los círculos involucrados en la "presente"

aproximación pertenezcan al menos al grado de aproximación deseada, que igual que en el programa anterior fue especificado previamente.

Como se puede apreciar, no todos los círculos serán de la "generación" especificada, sino que para el entorno de cada punto aproximado serán al menos de esa generación, siendo la mayoría de una o dos generaciones posteriores y aún más.

Ambos métodos aseguran la continuidad por las razones expuestas en el capítulo anterior, pero la suavización lograda con éste es sensiblemente mayor, ya que es justamente donde se yuxtaponen los entornos de cada punto donde se está logrando la mejor aproximación.

Al requerir menos pasos, el proceso es en este caso más rápido que en el anterior, ambos se prestan dado que trabajan con consideraciones locales para ser programados en paralelo y al hacerlo las ventajas son sensiblemente mayores. En el primer método se pueden establecer los procesos que se desee en paralelo mientras que en este último se debe establecer un límite que se puede fijar cuando la cercanía de dos puntos tratados simultáneamente rebase el doble del radio promedio de los entornos de los puntos, lo cual, y esto es un defecto, debe ser establecido previamente sin realmente ser un dato conocido.

En la figura 32 se muestra el diagrama de flujo del método.

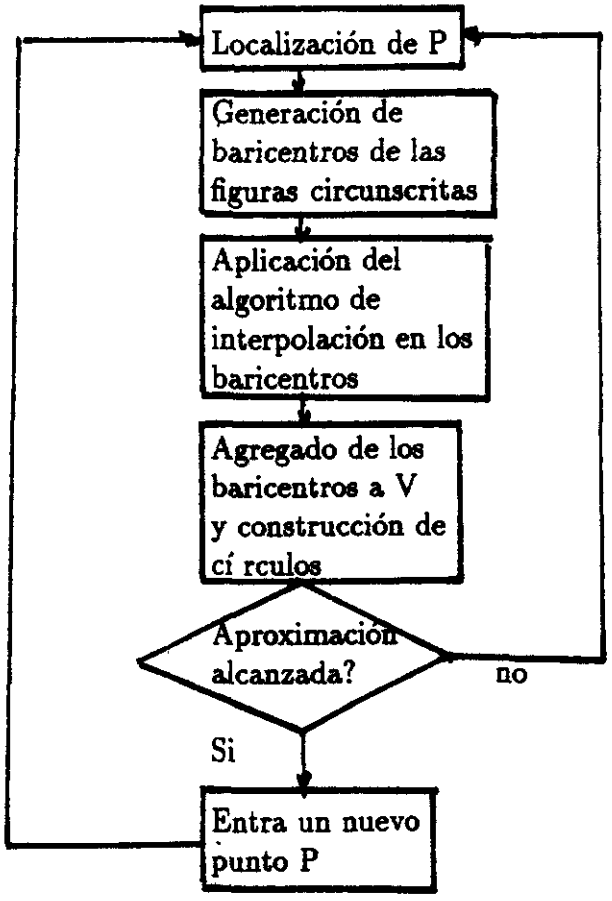

Figura 32
Diagrama de flujo

5.4 Algoritmos con resultados en puntos no prefijados

El hecho de poder usar los baricentros de las figuras que intervienen en cada sección $A \in \mathcal{P}$ nos permite realizar en forma automática la aproximación general de toda la superficie, agregando todos los baricentros de las partes $A \in \mathcal{P}$, construyendo la nueva partición en círculos de cobertura y volviendo a repetir el proceso, realizando la interpolación en cada paso.

El problema práctico a resolver es entonces determinar automáticamente los baricentros de las $A \in \mathcal{P}$, para ello proponemos un algoritmo similar al de búsqueda utilizado para construir los círculos. Esta vez, sin

embargo, lo que se buscaría sería el punto equipotencial respecto a las esferas involucradas en la definición de cada $A \in \mathcal{P}$.

5.5 Aplicación a funciones Spline

En este caso se debe plantear un sistema de ecuaciones cuya matriz principal es enrarecida y cada ecuación es la expresión numérica de la ecuación diferencial en derivadas parciales correspondiente a imponer suavidad en cada vértice.

Cada renglón de la matriz corresponde a un vértice de V, para plantear la ecuación se debe hallar la lista de sus vecinos naturales, posteriormente calcular los cosenos directores de cada una de las direcciones que lo unen con sus vecinos. Finalmente se calculan las primeras derivadas direccionales en dichas direcciones.

La ecuación a plantear consiste en igualar los productos externos o vectoriales de dichas derivadas, tomadas dos a dos, lo cual es igual a imponer que pertenezcan todas a un mismo plano tangente a la superficie a aproximar. El polinomio de Bernstein sobre el que se hace lo anterior es el que se obtiene con la fórmula del capítulo 4, pero con $n = 2$.

En otras palabras:

$$S_f(P) = \sum_{|\mathbf{i}|=1} B^r_{\mathbf{i}}(P) f(v_{\mathbf{i}})$$

aqui $B^r_{\mathbf{i}}(P)$ es de grado 2 en r variables entonces:

$$\left(\sum_{i=1}^{r} \partial^{i-1} f(u-1) B^r_i(d) \bigwedge \sum_{i=1}^{r} \partial^i f(u) B^r_i(d) \right)$$

$$\bigwedge \left(\sum_{i=1}^{r} \partial^i f(u) B^r_i(d) \bigwedge \sum_{i=1}^{r} \partial^{i+1} f(u+1) B^r_{i+1}(d) \right) = 0$$

es la condición de suavidad.

Capítulo 6

Superficies en 3D y 4D

Hasta ahora solamente hemos mostrado aplicaciones para el caso relativamente particular de funciones definidas en una región del espacio de 2D, cuya gráfica es una superficie en el espacio de 3D, utilizable, por ejemplo, para reconstruir superficies topográficas a partir de datos de campo. No obstante, como se ha mencionado anteriormente, las reales ventajas del método son más evidentes cuando aumenta la dimensión.

En efecto, para el problema que se ha tratado hasta ahora existen, como se ha visto, muchas otras soluciones y también existe la implementación computacional de las mismas. Inclusive se puede afirmar que estas implementaciones son muy eficientes, dado el tiempo y la cantidad de personas que se han dedicado a las mismas, llegando muchas de ellas a ser verdaderas obras de arte.

Cuando se trata de reconstruir o interpolar funciones definidas en un espacio de 3D también existen soluciones, pero ya son más escasas y más recientes, y muchas veces se basan más en adelantos en el "hardware" que en el "software", lo cual suele ser, aunque eficaz, una aproximación muy brutal.

Es muchísimo más escaso lo escrito para 4D y prácticamente nulo lo hecho en cuanto a implementación, sin hablar de más dimensiones, donde ya no hay referencias.

Generalizar los triángulos de Bézier, por ejemplo a tetraedros, no es

muy simple y además existen complicaciones adicionales cuando se trata de interpretar los resultados, aunque sea para algo tan simple como dibujar algún resultado. Algo similar sucede usando los tetraedros como base de splines, ya que en ambos casos se debe llevar el conteo de las caras, aristas, etc., a más de construir previamente la red de tetraedros o simplejos de más dimensiones. Un intento en tal sentido fué hecho por nosotros en Traversoni [36].

Si se acepta la *hipótesis* de que se parte de una malla de puntos arbitrarios en 3D por ejemplo, el tomar mallas "regulares" de tetraedros implica realizar un paso adicional de interpolación para asignar valores en los vértices de esa malla y, por el contrario, construir algo similar a los triángulos de Delaunay puede resultar muy complicado en más de 2D.

Construir la malla y refinarla basándose en esferas de cobertura implica muy poco más trabajo que construir círculos de cobertura, y todas sus propiedades son igualmente generalizables, como se vió en el Capítulo 2, a 3 y más dimensiones.

En este capítulo solo abordaremos dos problemas que son los más comunes en este tópico: la construcción de superficies no uniformes en 3D, por ejemplo, superficies cerradas, en principio superficies convexas, pero se indicará alguna forma de proceder en el caso de concavidades. Se abordará también el problema de como hallar puntos en 4D o más, es decir la generalización estricta de lo visto en los Capítulos 4 y 5.

6.1 Interpolación en superficies de 3D

Planteamiento del problema.

Sea V un conjunto finito arbitrario de puntos en el espacio euclidiano de dimensión 3 pertenecientes a una superficie S no plana, convexa y conexa, así como cerrada. Se trata de conseguir una representación suave de S, para lo cual se deben construir nuevos puntos de S o aproximados a ellos dentro de un margen preestablecido.

Este planteamiento tiene algunas consecuencias interesantes de destacar:

1) La superficie a aproximar cubre (contiene) a la cerradura convexa de V.

2) Toda suavización del conjunto original de esferas de cobertura cubre también a la cerradura convexa de V.

Teniendo en cuenta las anteriores consideraciones se puede enfocar el problema en dos vertientes. Una consiste en triangular la superficie, teniendo como vértices los puntos conocidos. Esto tiene una doble función, cuando la superficie no se conoce la idea es usar la triangulación para, a partir de ella, reconstruir la superficie; sin embargo existe también el problema, cuya solución puede ser muy útil, de triangular la superficie conociéndola. Este último problema ya ha sido abordado por algunos autores como Renka [27] que ofrece algunas soluciones reduciendo el problema a triangular una esfera.

Las esferas de cobertura presentan una posibilidad mucho más sencilla y rápida para resolver el citado problema, solución que el autor presentó por primera vez en la "SIAM Conference on Geometric Modeling", que se realizó en Tempe, Arizona en Noviembre de 1989 y que describimos a continuación.

Se propone, dado el conjunto V de puntos perteneciente a S constuir el correspondiente conjunto $\mathbf{S}$ de esferas de cobertura.

Teorema 6.1.1 *Sea P un poliedro inscrito en una de las esferas de $\mathbf{S}$ y con vértices en los puntos de V. Entonces P es un tetraedro de Delaunay o es divisible en un número finito de ellos, y todas sus caras o son triángulos de Delaunay o son divisibles en un número finito de ellos.*

Demostración: La primera parte del teorema se sigue de que dichos poliedros cumplen con la propiedad de que en su esfera circunscripta no está incluido ningún punto de V, lo cual está asegurado por la construcción de las esferas de cobertura; la segunda parte fue ya demostrada por Delaunay [15] en 1934.

De esta manera, construídas las esferas, para tener la triangulación de la superficie basta con considerar las caras de los poliedros que no están repetidas (dado que éstas son interiores) para tener de una sola vez la triangulación y la cerradura convexa de V. En la figura 33 se aprecia el dibujo de un ejemplo.

Se ven remarcadas las aristas que forman parte de la cerradura convexa.

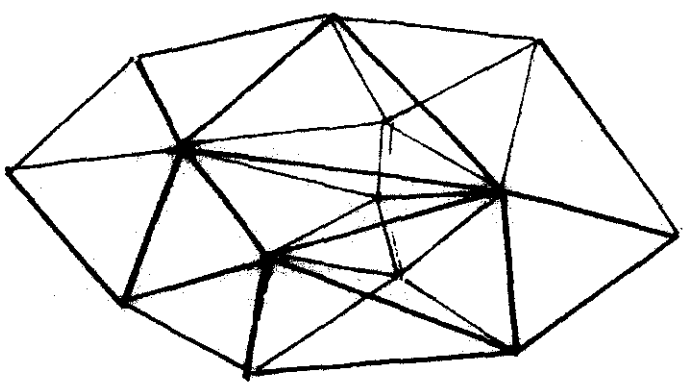

Figura 33

Realizar lo anterior es muy fácil con tetraedros, (dado que cualquier combinación de tres puntos es una cara) por lo que se debe previamente subdividir los poliedros en tetraedros.

Para subdividir un poliedro en tetraedros se puede aprovechar algo de lo que ya se había realizado usando productos vectoriales. La idea es usar la subrutina que calcula los productos vectoriales aprovechándola al máximo. Para hacerlo, dado un poliedro P inscrito en la esfera E (ver figura 34) y tal que tiene m vértices, con $m > 4$, se construye primero un tetraedro cualquiera con cuatro de los m vértices; posteriormente se toma una cara cualquiera de dicho tetraedro y se la caracteriza con el vector v producto vectorial de dos de sus aristas y se decide con los restantes $m-4$ puntos si están del mismo lado de la cara elegida que el cuarto vértice del tetraedro.

Si el producto vectorial de v por el vector que une al vértice común de las dos aristas con el cuarto punto del tetraedro tiene el mismo signo que el producto de v por el vector que une al punto con dicho vértice, entonces están del mismo lado de la cara y de diferente lado en caso contrario. Si están en lados distintos se construye el segundo tetraedro con la cara y el nuevo punto y se continúa el proceso; si están todos los $m-4$ puntos del mismo lado se cambia de cara. En la figura 34 se ilustra el proceso.

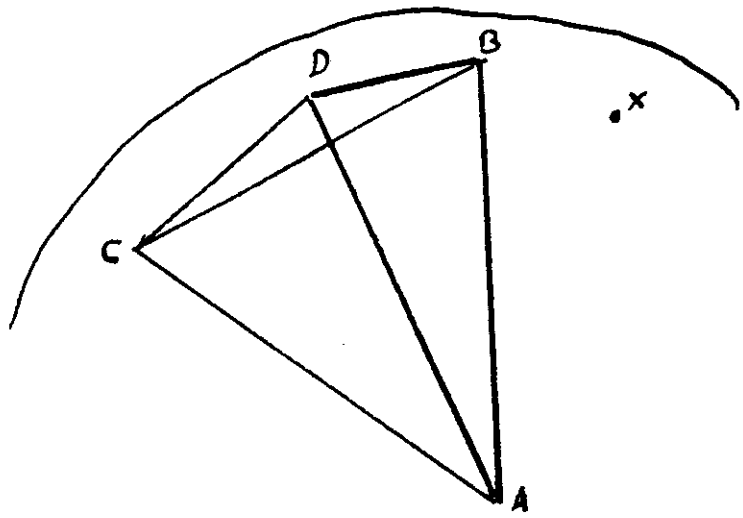

Figura 34

El poliedro se muestra abierto para facilitar el dibujo. El tetraedro elegido es el $ABCD$ (marcado más obscuro) X está en distinto semiespacio que C respecto a ABD, entonces el tetraedro $ABDX$ es válido.

De esta manera se usan las esferas de cobertura como un paso intermedio para obtener el "wire frame" del conjunto V, pero la finalidad última es realizar una triangulación, para luego, apoyándose en ella, recurrir a otros métodos para construir la superficie S.

La construcción propuesta, gracias a las esferas es la más sencilla y rápida, ya que la construcción de las mismas puede ser completada en $n \log n$ operaciones, a las que hay que agregar las necesarias para individualizar las caras útiles de los tetraedros que forman la cerradura convexa. Realizar ésto implica sobre todo operaciones lógicas mucho menos costosas que las operaciones aritméticas asociadas a construcciones geométricas, a las que solo hay que recurrir en casos como el descrito más arriba, en el cual el poliedro inscrito en la esfera no es un tetraedro.

Un subproducto no poco importante es también la tetraedronalización del espacio contenido en la cerradura convexa de V, tarea sobre la que prácticamente no existen referencias de algoritmos operativos.

Se decía más arriba de dos vertientes posibles para enfocar el problema

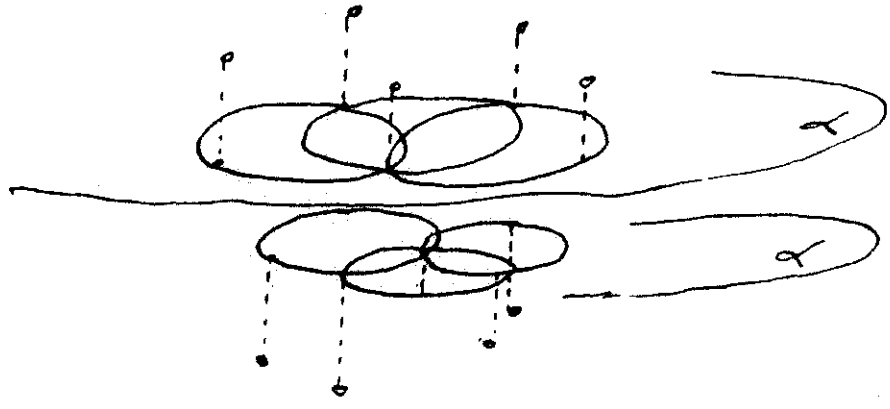

Figura 35

original de las que la anterior es solo una. La otra sería reconstruir la superficie a partir de los puntos de V. Para ello, el método que se propone es reducir el problema a uno del estilo del presentado en el capítulo 4.

Realizar lo anterior requiere el uso de un plano auxiliar α tal que pase por el baricentro de V y los puntos extremos de V en cualesquiera de las tres dimensiones del espacio de 3D (x, y o z). Un plano así partirá a una superficie como la que se ha descrito en dos superficies uniformes. Proyéctese ahora los puntos de cada una de las partes sobre α, obteniéndose dos conjuntos de puntos auxiliares V_1 y V_2 sobre α.

Procédase ahora a construir los círculos de cobertura correspondientes a V_1 y V_2, sean éstos C_1 y C_2. Con ellos se puede proceder como en el Capítulo 4. En la figura 35 se ilustra un ejemplo.

Se muestra el plano y el conjunto de puntos separados por claridad. En realidad se trata de un solo plano.

Proceder de la forma descrita nos asegura tener dos superficies convexas.

Teorema 6.1.2 *Sea V un conjunto de puntos en 3D ?? convexo y sea V_1 su proyección sobre un plano α. Entonces el resultado de aplicar el algoritmo de 4.1.2. a V_1 da una superficie convexa.*

?? *Demostración:* Los puntos obtenidos con el algoritmo 4.1.2. pertenecerán, según el caso, al plano que une tres puntos de V_1 (y es parte de su cerradura convexa) o a alguna cuádrica que une a más puntos de V_1 la cual, al igual que el plano, será convexa.

Lo que no se puede asegurar es que la unión de las superficies obtenidas para V_1 y V_2 sea suave, por lo cual será necesario aplicar algún tipo de suavización particular a esa zona, la cual puede muy bien ser volver a aplicar el método, ahora sobre un plano β perpendicular a α, de modo tal de restringir dicha aplicación a la zona afectada.

6.2 Interpolación en superficies de 4D y más

Supóngase ahora que lo que se tiene es, por ejemplo, un conjunto de puntos en 3D, munidos cada uno con una propiedad que presenta un valor determinado para cada uno. Si se considera a dicha propiedad como una cuarta dimensión se puede plantear el problema de construir una superficie en 4D mapeando desde 3D y usando una generalización casi inmediata del algoritmo de 4.1.2.

Existen infinidad de ejemplos en los que lo anterior puede ser de utilidad, dependiendo de lo que sea la "propiedad", como sería el caso del calentamiento de superficies (fuselajes por ejemplo), la simulación de mezclas de líquidos con distintas propiedades, como por ejemplo, distinto contenido de sal, o inclusive para aplicaciones en cálculos relativistas de Regge de los llamados 3+1 donde la cuarta dimensión es el tiempo, sustituyendo un problema de Elemento Finito por uno de interpolación.

A continuación se analizan las posibles dificultades en la generalización de lo visto en 4.1.2. a más dimensiones.

En primer lugar véase que lo planteado en el capítulo 4 es general y se aplica en cualquier dimensión. Los ejemplos de dos dimensiones con los que se ha ilustrado todo lo anterior han sido usados fundamentalmente porque es fácil dibujarlos.

Para fijar ideas se repetirá el proceso del capítulo 4 en lo relativo al algoritmo de De Casteljau, ahora para más dimensiones, nos apoyaremos para ello en la figura 36, análoga a la 28.

El punto P afecta a las esferas E_1 y E_2 determinadas respectivamente por los puntos A, B, C, D, E y E, C, D, F. Se aplica el algoritmo de Sibson,

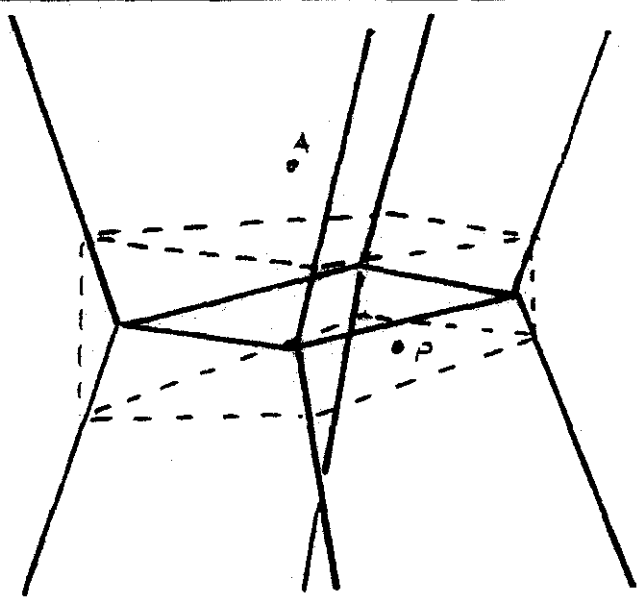

Figura 36

ahora con volúmenes.

Ahora en vez de hablar de triángulos hablaremos de tetraedros, si se trata de $3D$, si no de simplejos, lo importante es que, si se trata de un espacio de dimensión n, tienen $n+1$ vértices y son la mínima figura inscriptible en una esfera de cobertura correspondiente al espacio que sea.

Las consideraciones que se hacían en la figura 28 son ahora también válidas en cuanto al número de vecinos de un *punto* $P \in V$ afectante.

Para implementar el algoritmo de Sibson tal como se hizo en el capítulo 4 se debe poder hallar de la misma forma el volumen de un tetraedro formado con P y, por ejemplo, C_0^1 , C_0^2 , C_0^3 , centros de las esferas afectadas y algunas nuevas.

Sea ahora P el punto afectante y C_0^1, C_0^2, C_0^3 la cara de un poliedro afectado, a C_0^1, C_0^2, C_0^3 lo caracterizaremos por su área, dada por el vector

$$A = \| \, (C_0^1 - C_0^2) \wedge (C_0^3 - C_0^2) \, \| \, .$$

Se rebate ese vector 90 grados sobre el plano de $C_0^1 \, C_0^2 \, C_0^3$ obteniéndose el vector A'. Ahora el volumen del tetraedro $PC_0^1 \, C_0^2 \, C_0^3$ quedará dado por

$$V = \| \, A' \wedge (P - C_0^2) \, \| \, .$$

Es importante notar que en aras de la generalización a más dimensiones se ha optado por no usar el producto mixto, que hubiera dado fácilmente el volumen, de modo de poder usar siempre solo el producto vectorial. El rebatimiento de 90 grados se debe hacer $N-2$ veces a medida que N va aumentando con la dimensión.

La coordenada de Sibson de P respecto a uno de sus vecinos queda entonces

$$P_i = \frac{\sum V_i'}{\sum V_i}.$$

Donde V_i' representa la suma de todos los productos vectoriales con los que se forma el volumen intersección de poliedros de Voronoi y V_i es la suma de los productos que dan el volumen del poliedro de Voronoi de P.

Y el valor interpolado de la variable será como siempre

$$PI = \sum P_i H_i.$$

Donde los H_i son los valores en cada uno de los vecinos de P.

Bibliografía

[1] R. Bartels, J. Beatty and B. Barsky, *An introduction to splines for use in Computer Graphics and Geometric Modelling*, Morgan Kaufmann, 1987.

[2] P. Bèzier, *Definition numerique des courbes et surfaces*, I. Automatisme, 11:625–632, 1966.

[3] W. Boehm, *Cubic B–Spline curves and surfaces in computer aided geometric design*, Computing 19:29–34, 1977.

[4] C. de Boor, *Bicubic Spline Interpolation*, J. Math. Phys. 41:212–218, 1962.

[5] A. Bowyer, *Computing Dirichlet tessellations*, Comput. J. 24(2) 162-166. 1981.

[6] P. de Casteljau, *Outillages méthodes Calcul*, Technical report, A. Citroen, Paris, 1959.

[7] P. de Casteljau, *Shape mathematics and CAGD*, Kogan page, Londres, 1986.

[8] H.N. Christiansen and T.W. Sederberg, *Conversion of complex contour line definitions into polygonal element mosaics*, Computer Graphics, 13: 187–197, 1986.

[9] C. K. Chui, *Vertex Splines and their approximation power of multivariate splines*, Computation of Curves and Surfaces, NATO ASI series, Dahmen, Gasca and Micchelli eds., Kluwer, 1990.

[10] C. K. Chui Lai, *Vertex Splines*, Curves and Surfaces in computer vision and graphics, Proceedings of the SPIE SPSE Simposium on Electronic Imaging, 1990.

[11] S.A. Coons, *Surfaces for Computer Aided Design*, Technical report M.I.T. 1964.

[12] P. J. Davis, *Interpolation and approximation*, Dover, New York, 1975.

[13] B. Delaunay, *Sur la sphere vide*, Academia de Ciencias de la URSS, 1934.

[14] T. DeRose and T. Hollman, *The triangle; a multiprocessor architecture for fast curve and surface generation*, Technical report, Computer Science Department, Univ. of Washington, 1987.

[15] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Berlin, 1987.

[16] G. Farin, *Bezier polynomials over triangles and the construction of piecewise C^r polynomials*, Technical report TR/91, Brunel University, Uxbridge, Inglaterra, 1980.

[17] G. Farin, *Triangular Bernstein Bezier patches*, Computer Aided Geometric Design, 3(2):83–128, 1986.

[18] G. Farin, *Curves and Surfaces for CAGD*, Arizona State University, Course 24, 1988.

[19] G. Farin, *Curves and Surfaces for Computer Aided Design*, Academic Press, New York, 1988.

[20] G. Farin, *Surfaces over Dirichlet tessellations*, Computer Aided Geometric Design 7 N 1–4, 1990.

[21] J. Ferguson, *Multivariable curve interpolation*, JACM II 2:221–228, 1964.

[22] R. N. Goldman, *Recursive Triangles*, Computation of Curves and Surfaces, NATO ASI series, Dahmen, Gasca and Micchelli, eds., Kluwer, 1990.

[23] L. Nachman, *A blended tensor product B-spline surface*, Computer Aided Design, 20, 1988.

[24] Takao Ohya, Masao Iri and Kazuo Murota, *Improvement of the incremental method for the Voronoi diagram with computational comparison of various algorithms*, Journal of the Operations Research Society of Japan, 27 N24, 1984.

[25] O. Palacios Velez and Baltazar Cuevas, *A dynamic hierarchical subdivision algorithm for computing Delaunay triangulations and closest point problems*, ACM Trans. Math. Software, 16 N 3, 275–292, 1990.

[26] F.P. Preparata and M.I. Shamos, *Computational Geometry: an Introduction*, Springer-Verlag, N.Y., Berlin, 1985.

[27] R. Renka, *Interpolation of data on the surface of a sphere*, ACM Trans. on Math. Software, 10 N4, 1984.

[28] M. Sabin, *Recursive division*, In J. Gregory Editor, The mathematics of surfaces, pp 269–281, 1982.

[29] Hanan Samet, *The quadtree and related hierarchical data structures*, Computing Surveys, 16 n2, 1984.

[30] R. Sibson, *A vector identity for the Dirichlet tessellation*, Math. Proc. Cambridge Philos. Soc. 87 151–155, 1980.

[31] R. Sibson, *A brief description of the natural neighbor interpolant*, Technical report, Math. Dept. Univ. of Bath., 1980.

[32] R. Sibson and P.J. Green, *Computing Dirichlet Tesselations in the Plane*, The Computer Journal, 21, N 2, 1977.

[33] L. Traversoni, *Algunos aspectos de la triangulación de Delaunay en el plano*, Memorias de la Academia de Ciencias de Zaragoza, España, N 2, 1990.

[34] L. Traversoni y O. Palacios, *Hierarchical Covering Spheres of a given set of points*, Curves and Surfaces I, P.J. Laurent, A. Le Mehauté, L. Schumaker Eds. Academic Press, New York, 1991.

[35] L. Traversoni, ***Modified Natural Neighbor Interpolant***, SPIE– Proceedings V. 1824, Curves and Surfaces III, J. Warren. Ed., Boston, 1992.

[36] L. Traversoni, ***Modelling man made channels between sea and coastal lagoons***, Computational Modelling of seas and coastal regions, Elsiever, Computational Mechanics 1992.

[37] Y.F. Wang and J.K. Aggarwal, *Surface reconstruction and representation of 3D scenes*, Pattern Recognition, 19, N 3, 197–207, 1986.

[38] D. Watson, *Computing the n–dimensional Delaunay tessellation with application to Voronoi polytopes*, Comput. J. 24: 167–172, 1981.

1 ANEXO A

Se presenta un listado de un programa preliminar que realiza los círculos de cobertura y los dibuja. El programa requiere aún ser perfeccionado y optimizado.

```
program esco;
uses
    crt,graph,ventan;
    var
       tik,k1,k2,k3,i,j,k,icon,x1,y1,x2,y2,ix,iy,xrp,yrp,gd,gm,icon1,ncfa,xc1,yc1,rc1,
       k7,xb,yb,pv,marca,icon2,ex,ey,imaxx,imaxy,iminx,err,icon3:integer;
       xr1,xr2,yr1,yr2,rel,rely,ya,xmax,ymax,xmin,ymin,
       pw,xi,yi,xj,yj,xk,yk,maxx,maxy,minx,xnr,ynr:real;
       pts:array[1..100,1..3] of real;
       cfa:array[1..200,1..4] of real;
       cfpunt:array[1..200,1..5] of integer;
       pot:array[1..100] of real;
       puaf,indir,x,y:array[1..100] of integer;
       afect:array[1..50] of integer;
       cxu,cuy,cu,ch,ch1:char;
       ls,lr,cota,ai:array[1..30] of real;
       nomb,cade,crx,cry,six,siy:string[10];
       entra,sale:text;
    procedure dibujo;
        begin
         for i:=1 to icon do
          begin
    line(x[i]-3,y[i],x[i]+3,y[i]);
    line(x[i],y[i]-2,x[i],y[i]+2);
          end;
         end;
         procedure blsup;
         var
            ik:integer;
          begin
                           setcolor(0);
         for ik:=5 to 24 do
         line(10,ik,550,ik);
         setcolor(15);
         moveto (5,5);
          end;
         procedure convert;
         begin
```

```
  xrp:=ix;
          yrp:=(y2-y1)-iy;
          ynr:=(yr2-yr1)*yrp/(y2-y1);
          ynr:=ynr+yr1;
          xnr:=(xr2-xr1)*xrp/(x2-x1);
          xnr:=xnr+xr1;
 end;
procedure leenomb;
 begin
 nomb:='';
 cu  := readkey;
while cu <> #13 do
begin
 nomb:=nomb+cu;
 outtextxy(ex,ey,nomb);
 cu:=readkey;
end;
 end;
 procedure guarda;
  begin

nomb:= '';
leenomb;
if nomb <> ''
then
  begin
  assign(sale,nomb+'.dat');
  rewrite(sale);
  writeln(sale,icon);
  for i:=1 to icon do
   begin
    writeln(sale,pts[i,1],' ',pts[i,2],' ',pts[i,3]);
   end;
  close(sale);
end;
  end;
 procedure leetex;
 begin
 nomb:='';
 cu  := readkey;
while cu <> #13 do
begin
 nomb:=nomb+cu;
 gotoxy(20,12);write(nomb);
```

```
  cu:=readkey;
 end;
  end;

       procedure capin1;

begin
clrscr;
venta(8);
 gotoxy(2,2);write('De las coordenadas reales de su pantalla');
 gotoxy(2,3);write('De x1 : ');read(xr1);
 gotoxy(2,4);write('de y1 : ');read(yr1);
 gotoxy(2,5);write('de x2 : ');read(xr2);
 gotoxy(2,6);write('de y2 : ');read(yr2);
 gotoxy(2,8);write('Declare el disco de bgi ');cuy:=readkey;
 write(cuy);
 gotoxy(2,10);write('Tiene una pantalla de 640x400 pixeles s/n');
 cu:=readkey;write(cu);
 maxx:=0;maxy:=0;minx:=xr2+100;
 gd:=detect;
 initgraph(gd,gm,cuy+':\tp6\bgi');
 if cu='n' then begin
 setviewport(30,10,600,200,clipon);
 line(1,25,569,25);line(569,25,569,189);
 line(569,189,1,189);line(1,189,1,25);
 x1:=1;x2:=569;y1:=1;y2:=189;
 end;
 if cu='s' then begin
 setviewport(30,10,600,400,true);
 line(1,1,569,1);line(569,1,569,389);
 line(569,389,1,389);line(1,389,1,1);
 x1:=1;x2:=569;y1:=1;y2:=389;
 end;
  settextstyle(defaultfont,horizdir,1);
 moveto (5,5);
 outtext('Oprima segun el caso : <1>Despliegue archivo <2>Captura');
 ch:=readkey;
 if ch='1' then begin
 blsup;
  outtext('Archivo a desplegar ?');ex:=150;leenomb;
   if nomb <> ''
 then
   begin
   assign(entra,nomb+'.dat');
```

```
      reset(entra);
      readln(entra,icon);
      for i:=1 to icon do
       begin
        readln(entra,pts[i,1],pts[i,2],pts[i,3]);
       end;
      close(entra);
        rel:=(x2-x1)/(xr2-xr1);
        rely:=(y2-y1)/(yr2-yr1);
        for i:=1 to icon do
         begin
         x[i]:=round((pts[i,1]-xr1)*rel);
         ya:=(yr2-yr1)-pts[i,2]+yr1;
         y[i]:=round(ya*rely);
        end;
      end;
      dibujo;delay(3000);
     end;
    if ch='2' then begin
    blsup;
outtext('Elija : <1> Datos de mapa <2> Coordenadas');
ch:=readkey;
blsup;
     case ch of
     '1' : begin marca:=1;moveto (5,5);
     outtext('Ubique los puntos con el cursor, con home los fija');end;
     '2' : begin marca:=2; moveto(5,5);
     outtext('Oprima : <1> Para poligonales <2> Detalle cortes');
     ch1:=readkey; if ch1='2' then marca:=4;end;
     end;

    end;
    icon2:=0;

    icon1:=0;
    ix:=50;iy:=50;
    if (marca=1) or (marca=3) then begin
    icon:=0;
    repeat

    str(ix,six);str(iy,siy);

    moveto (5,15);outtext(' Posicion del cursor : ');
    outtext(six);outtext('  ');outtext(siy);
```

```
moveto (24,15);six:='   ';siy:='   ';
outtext(six);outtext('  ');outtext(siy);
setcolor(15);

 line(ix,iy,ix+2,iy);
ch:=readkey;
setcolor(0);
for k:=15 to 21 do
begin
line(190,k,250,k);
end;
for k:=15 to 21 do
line(304,k,570,k);
line(ix,iy,ix+2,iy);
        icon1:=icon1+1;

 if icon1=10 then begin
          setcolor(15);
 dibujo;
 icon1:=0;
 end;
 if ch='8' then iy:=iy-10;
 if ch='2' then iy:=iy+10;
 if ch='6' then ix:=ix+10;
 if ch='4' then ix:=ix-10;
case ord(ch)  of
77 : ix:=ix+2;
75 : ix:=ix-2;
72 : iy:=iy-1;
80 : iy:=iy+1;
71 : begin
     setcolor(blue);

line(ix-3,iy,ix+3,iy);line(ix,iy-2,ix,iy+2);
icon:=icon+1;
x[icon]:=ix;y[icon]:=iy;
convert;
pts[icon,1]:=xnr;
pts[icon,2]:=ynr;
      if marca=1 then begin
       blsup;
     outtext('De la cota en metros : ');ex:=200;ey:=5;
     leenomb;val(nomb,pts[icon,3],err);if err<>0 then write('error');
     end;
```

```
    if maxx<=pts[icon,1] then begin maxx:=pts[icon,1];imaxx:=icon;end;
    if maxy<=pts[icon,2] then begin maxy:=pts[icon,2];imaxy:=icon;end;
    if minx>=pts[icon,1] then begin minx:=pts[icon,1];iminx:=icon;end;
    end;
            end;
        setcolor(15);
        convert;
    str(xnr:8:2,crx);str(ynr:8:2,cry);
            moveto(256,15);outtext('coord reales ');outtext(crx);
            outtext('  ');outtext(cry);
        line(ix,iy,ix+2,iy);
        until ch='f';
      end;
      if (marca=2) or (marca=4) then delay(3000);
       blsup;
         outtext('Quiere archivar los datos s/n ? ');ch:=readkey;
         if ch='s' then begin
          blsup;
         outtext('Con que nombre archiva ? ');ex:=200;ey:=5;
      guarda;
      end;

end;
       procedure poner(i1,j1,k1:integer);
        begin
        xi:=pts[i1,1];
        yi:=pts[i1,2];
        xj:=pts[j1,1];
        yj:=pts[j1,2];
        xk:=pts[k1,1];
        yk:=pts[k1,2];
       end;
PROCEDURE Circulo(VAR Xi,Yi,Xj,Yj,Xk,Yk,Xc,Yc,Rc : REAL);
(*************************************************************************
 *                                                                       *
 * DESCRIPCION : Procedimiento para calcular las coordenadas del centro y *
 *               el radio de la circunferencia que pasa por tres puntos  *
 *               dados.                                                  *
 * NOTA : Cuando los puntos son colineales se regresa Rc = 0             *
 *                                                                       *
 * AUTORES: Dr. Oscar Palacios Velez                                     *
 *          M. C. Baltasar Cuevas Romaud                                 *
 *                                                                       *
 * DESARROLLO INICIAL: octubre de 1987                                   *
```

```
*                                                                              *
* ULTIMA REVISION: abril de 1988                                               *
*                                                                              *
********************************************************************************)

VAR
  Ipen1, Ipen2                    : BOOLEAN;
  Numcaso                         : CHAR;
  X1,Y1,X2,Y2,Pen1,Pen2,Coc       : REAL;

BEGIN
  X1  := (Xi + Xj) / 2.0;
  Y1  := (Yi + Yj) / 2.0;
  Ipen1 := FALSE;
  IF Yi = Yj THEN Ipen1 := TRUE
  ELSE
  Pen1 := -(Xi-Xj)/(Yi-Yj);
  X2 := (Xi+Xk)/2.0;
  Y2 := (Yi+Yk)/2.0;
  Ipen2 := FALSE;
  IF Yi = Yk THEN Ipen2 := TRUE
  ELSE
  Pen2 := -(Xi -Xk)/(Yi-Yk);
  IF (Ipen1 = FALSE) AND (Ipen2 = FALSE) THEN
    IF Pen2 <> 0.0 THEN  Coc := Pen1/Pen2 ELSE Coc := 2.0;
  IF (Ipen1 = FALSE) AND (Ipen2 = FALSE) AND (Pen1 <> Pen2) THEN
     Numcaso := 'A';
  IF (Ipen1 = FALSE) AND (Ipen2 = FALSE) AND (Pen1 = Pen2) THEN
     Numcaso := 'B';
  IF (Ipen1 = FALSE) AND (Ipen2 = FALSE) AND (Coc > 0.9999) AND
     (Coc < 1.0001) THEN Numcaso := 'B';
  IF (Ipen1 = FALSE) AND (Ipen2 = TRUE) THEN
     Numcaso := 'C';
  IF (Ipen1 = TRUE) AND (Ipen2 = FALSE) THEN
     Numcaso := 'D';
  IF (Ipen1 = TRUE) AND (Ipen2 = TRUE) THEN
     Numcaso := 'E';
  CASE Numcaso OF
      'A'     : BEGIN
                Xc := Y1 -Pen1*X1-Y2+Pen2*X2;
                Xc := Xc/(Pen2-Pen1);
                Yc := Y1 + Pen1*(Xc-X1);
                END;
      'B','E': BEGIN
```

```
                Rc := 0;
                EXIT;
                END;
        'C'     : BEGIN
                Xc :=X2;
                Yc := Y1+Pen1*(Xc-X1);
                END;
        'D'     : BEGIN
                Xc := X1;
                Yc := Y2+Pen2*(Xc-X2);
                END;
   END;
   Rc := SQRT(SQR(Xi-Xc)+SQR(Yi-Yc));
END;

procedure potencia(i1,i2:integer);
 begin
 pw:=(sqr(pts[i1,1]-cfa[i2,1])+sqr(pts[i1,2]-cfa[i2,2]))-sqr(cfa[i2,3]);
 blsup;
 moveto(15,15);str(pw,six);outtext(six);cxu:=readkey;
 end;
       begin
       clrscr;
       for i:=1 to 200 do
        begin
        for j:=1 to 5 do
         begin
         cfpunt[i,j]:=0;
        end;
       end;
       for i:=1 to 200 do
        begin
        for j:=1 to 4 do
         begin
         cfa[i,j]:=0.0;
         end;
        end;
       venta(5);
       gotoxy(5,5);write('PROGRAMA PARA CALCULAR Y DIBUJAR');
       gotoxy(5,6);write('CIRCULOS DE COBERTURA');
       gotoxy(5,8);write('Leonardo Traversoni 1992');
       gotoxy(5,10);write('oprima una tecla para comenzar');
       repeat until keypressed;
       capin1;
```

```
str(icon,six);
blsup;moveto(15,15);outtext(six);cxu:=readkey;
xmax:=pts[1,1];xmin:=pts[1,1];ymax:=pts[1,2];ymin:=pts[1,2];
for i:=2 to icon do
 begin
 if xmax < pts[i,1] then xmax:=pts[i,1];
 if xmin > pts[i,1] then xmin:=pts[i,1];
 if ymax < pts[i,2] then ymax:=pts[i,2];
 if ymin > pts[i,2] then ymin:=pts[i,2];
end;
pts[icon+1,1]:=2*xmax;
pts[icon+1,2]:=(ymax+ymin)/2;
pts[icon+1,3]:=0;
pts[icon+2,1]:=(xmax+xmin)/2;
pts[icon+2,2]:=2*ymax;
pts[icon+2,3]:=0;
pts[icon+3,1]:=xmin-2*xmax;
pts[icon+3,2]:=(ymax+ymin)/2;
pts[icon+3,3]:=0;
cfpunt[1,1]:=icon+1;
cfpunt[1,2]:=icon+2;
cfpunt[1,3]:=icon+3;
ncfa:=1;
poner(icon+1,icon+2,icon+3);
circulo(xi,yi,xj,yj,xk,yk,cfa[1,1],cfa[1,2],cfa[1,3]);
  rel:=(x2-x1)/(xr2-xr1);
     rely:=(y2-y1)/(yr2-yr1);
      xc1:=round((cfa[1,1]-xr1)*rel);
      ya:=(yr2-yr1)-cfa[1,2]+yr1;
      yc1:=round(ya*rely);
      rc1:=round(cfa[1,3]*rel);
      circle(xc1,yc1,rc1);
      str(icon,six);
blsup;moveto(15,15);outtext(six);cxu:=readkey;
blsup;
moveto(15,15);outtext('<1> Algoritmo lento <2> Rapido');
      cu:=readkey;outtext(cu);
{algoritmo lento}
if cu='1' then begin

for i:=1 to icon do
 begin
  icon1:=0;
  blsup;moveto(15,15);
```

```
outtext('Doy ncfa antes busqueda  ');str(ncfa,six);
outtext(six);outtext('  punto  ');str(i,six);
outtext(six);cxu:=readkey;
for j:=1 to ncfa do
 begin
 if cfa[j,4]<>1 then
  begin
  blsup;moveto(15,15);outtext('calculo potencia 1');cxu:=readkey;
   potencia(i,j);
   if pw=0 then
    begin
     if cfpunt[j,4]=0 then cfpunt[j,4]:=i;
     if cfpunt[j,4]<>0 then cfpunt[j,5]:=i;
    end;
   if pw<0 then
    begin
     blsup;moveto(15,15);outtext('pasa por aqui');cxu:=readkey;
     cfa[j,4]:=1;
     icon1:=icon1+1;
     afect[icon1]:=j;
    end;
  end;
end;{fin del do del j}
k1:=0;tik:=0;
for k:=1 to icon1 do
 begin
  for k2:=1 to 5 do
   begin
    if (k=1) and (k2=1) then
     begin
     blsup;moveto(15,15);outtext('paso por k1');cxu:=readkey;
      k1:=1;
      puaf[1]:=cfpunt[afect[1],1];
     end;
    for k3:=1 to k1 do
     begin
      if (cfpunt[afect[k],k2]=puaf[k3]) and (k3<>1)  then tik:=1;
     end;
    if (tik<>1) and (cfpunt[afect[k],k2]<>0) then
     begin
      k1:=k1+1;
      puaf[k1]:=cfpunt[afect[k],k2];
     end;
   end;
```

```
      end;{fin del do de k}
      tik:=0;
      if k1<>0 then
       begin
        k1:=k1+1;
        puaf[k1]:=puaf[1];
       end;
      str(k1,six);
      blsup;moveto(15,15);outtext(six);outtext(' numero de puntos para cfa ');cxu:=
      for k:=2 to k1 do
       begin
        poner(puaf[k-1],puaf[k],i);
        circulo(xi,yi,xj,yj,xk,yk,cfa[200,1],cfa[200,2],cfa[200,3]);
        tik:=0;
        for k2:=1 to k1-1 do
         begin
         blsup;moveto(15,15);outtext('calculo potencia 2');cxu:=readkey;
          potencia(puaf[k2],200);
          if pw<0 then tik:=1;
         end;
         if tik<>1 then
         begin
         blsup;moveto(15,15);outtext('pasa ncfa');cxu:=readkey;
          ncfa:=ncfa+1;
          cfa[ncfa,1]:=cfa[200,1];
          cfa[ncfa,2]:=cfa[200,2];
          cfa[ncfa,3]:=cfa[200,3];
          cfa[ncfa,4]:=0;
          cfpunt[ncfa,1]:=puaf[k-1];
          cfpunt[ncfa,2]:=puaf[k];
          cfpunt[ncfa,3]:=i;
         end;
       end;
    end;{fin del do del i}
blsup;
moveto(15,15);
str(ncfa,six);
blsup;moveto(15,15);outtext('Lista de circulos son ');outtext(six);
for i:=1 to ncfa do
 begin
  str(cfa[i,1],six);moveto(15,15+8*i);outtext(six);outtext('  ');
  str(cfa[i,2],six);outtext(six);outtext('  ');
  str(cfa[i,3],six);outtext(six);outtext('  ');
  str(cfa[i,4],six);outtext(six);outtext('  ');
```

```
 end;
 cxu:=readkey;
outtext('voy a empezar a dibujar');outtext(six);outtext('ncfa');
cu:=readkey;
rel:=(x2-x1)/(xr2-xr1);
     rely:=(y2-y1)/(yr2-yr1);
for i:=1 to ncfa do
 begin
  if cfa[i,4]<>1 then begin
      xc1:=round((cfa[i,1]-xr1)*rel);
      ya:=(yr2-yr1)-cfa[i,2]+yr1;
      yc1:=round(ya*rely);
      rc1:=round(cfa[i,3]*rel);
      circle(xc1,yc1,rc1);
      blsup;moveto(15,15);outtext('dibuja otra cfa');cxu:=readkey;
  end;
 end;
 end;{fin algoritmo lento}
     blsup;moveto(15,15);
     outtext('oprima una tecla para terminar');cxu:=readkey;
              closegraph;
end.
```

1 ANEXO B

Se presenta un listado del programa que realiza la interpolación

```
program intesco;
    uses
        graph,crt;
    const
        c1=100;
        c2=20;
    type
      {en este registro se guardan los puntos que pasan por
       la circunferencia}
      cadena=string[10];
      nodo=^reg_cir;
      reg_cir=Record
                  cir:integer;
                  nump:array[1..3] of integer;
                  elimina:string[2];
                  conec:nodo;
              End;

      {en este registro se guardan las circunferencias que
       pasan por cada punto}
      indi=^Reg_pto;
      Reg_pto=Record
                   npto:integer;
                   numc:integer;
                   elimina:string[2];
                  direc:indi;
              End;

      {en este registro se guardan las coordenadas de cada
       circunferencia}
      puntero=^reg_cor;
      reg_cor=Record
                ncirc:integer;
                h,k,r:real;
                nump:array[1..3] of integer;
                enlace:puntero;
                elimina:string[2];
              end;

  {    indica=^salvar;
```

```
      salvar= record
                numpe:array[1..4] of integer;
                elimina:string[2];
                pega:indica;
              end; }

      arreglo=array[1..c1,1..4] of real;
      arre=array[1..c2,1..2] of integer;
      lis_ar=array[1..15,1..2] of real;

      var
     {  marcir:indica; }
       juan:char;
       rd:string[2];
       lista,lista1,lista2,sal_lis,sal_ins,sal_cor:puntero;
       sal_cir:nodo;
       sal_pto:indi;
       datins,datos:arreglo;
       integra:arre;
       xpi,ypi,arr,arv,distpot,xi,xj,xk,yi,yj,yk,xc,yc,radio:real;
       borracir,np,ncir,pi,ii:integer;
         nombre:string;
         xw,resp,re:char;
         entra,guarda:text;
       sp,i,j,k,cont,ncmax,i1,i2,i3:integer;
       ptox,ptomax,pti,numero,cmar,cmax,pmax,pmin:integer;
       cirbor:arre;
       cbm:integer;
       crx:cadena;
       xf,yf,xam,yam:real;
       puntosarea:lis_ar;
{ lee los datos de entrada que son cuantos puntos y pide}
{sus coordenadas de cada uno de ellos}
   procedure lee_datos;
         var
           j:integer;
         begin
            clrscr;
            write(' Dame el numero de puntos de la red:');
            readln(np);
            for j:=1 to np do
              begin
                writeln('  Punto: ',j);
                write('      X: ');
```

```
                read(datos[j,1]);
                write('      Y: ');
                read(datos[j,2]);
                write('      Z: ');
                read(datos[j,3]);
              end;
    end; {procedimiento lee datos}


  {guarda en archivo los datos de entrada}
      procedure salvadatos;

        var
          i:integer;

        begin
          assign(guarda,nombre);
          rewrite(guarda);
          writeln(guarda,np);
          for i:=1 to np do
            writeln(guarda,datos[i,1],' ',datos[i,2],' ',datos[i,3]);
          close(guarda);
        end;


{lee los datos de disco}
    procedure leedisco;

       var
         i:integer;

       begin
         assign(entra,nombre);
          reset(entra);
         readln(entra,np);
         for i:=1 to np do
           readln(entra,datos[i,1],datos[i,2],datos[i,3]);
         close(entra);
       end;


{si lee datos de disco este procedimiento los presenta en pantalla}
    procedure escribedatos;
```

```
var
 i:integer;

begin
   clrscr;
   writeln('   LOS DATOS SON:');
   writeln('      Punto           X          Y          Z');
   for i:=1 to np do
      writeln(i,datos[i,1]:11:2,datos[i,2]:11:2,datos[i,3]:11:2);
end;


{en este procedimientos se incluyen todos los anteriores a este}
   procedure opcion;

      var
       respi:char;

      procedure leenomb;
         var
           cu:char;
          begin
            nombre:='';
            cu:=readkey;
            write(cu);
            while cu<>#13 do
              begin
                 nombre:=nombre+cu;
                { outtextxy(ex,ey,nombre); }
                 cu:=readkey;
                 write(cu);
              end;
          end;


      begin
         clrscr;
         gotoxy(20,11);
         write(' QUIERES LEER DATOS DE DISCO (S/N):');
         respi:=readkey;
         if (respi='S') or (respi='s') then
           begin
             gotoxy(20,12);
```

```
        write('   Dame drive, nombre y extension del archivo:');
        leenomb;
        leedisco;
        escribedatos;
       end;
      if (resp1='N') or (resp1='n') then
       begin
          lee_datos;
          write('  QUIERES GUARDAR ARCHIVO EN DISCO?');
          resp1:=readkey;
          if (resp1='S') or (resp1='s') then
            begin
              write(' DAME DRIVE, NOMBRE Y EXTENSION PARA SALVAR ARCHIVO:');
              leenomb;
              salvadatos;
             end;
       end;
  end;


{este procedimiento determina las coordenadas (xc,yc,r) de la
 circunferencia que pasa por tres puntos}
procedure determinahkr(xi,xj,xk,yi,yj,yk:real; var xc,yc,radio:real);

    var
      coc:real;
      x1,x2,y1,y2,m1,m2:real;
      Im1,Im2:boolean;
      caso:char;
      i:integer;

    begin
      x1:=(xi+xj)/2;
      y1:=(yi+yj)/2;
      Im1:=false;
      if yi=yj then
       Im1:=true
       else
          m1:=-(xi-xj)/(yi-yj);
      x2:=(xi+xk)/2;
      y2:=(yi+yk)/2;
      Im2:=false;
      if yi=yk then
         Im2:=true
```

```
 else
   m2:=-(xi-xk)/(yi-yk);
if (im1=false) and (im2=false) then
    if m2<>0 then
        coc:=m1/m2
      else
        coc:=2.0;
if (im1=false) and (im2=false) and (m1<>m2) then
    caso:='A';
if (im1=false) and (im2=false) and (m1=m2)
  then
     caso:='B';
if (im1=false) and (im2=false) and (coc>0.999) and (coc<1.0001)
   then
     caso:='B';
if (im1=false) and (im2=true)
 then
    caso:='C';
if (im1=true) and (im2=false)
  then
     caso:='D';
if (im1=true) and (im2=true)
  then
     caso:='E';

CASE caso of
     'A':begin
          xc:=y1-(m1*x1)-y2+(m2*x2);
          xc:=xc/(m2-m1);
          yc:=y1+m1*(xc-x1);
         end;
'B','E':begin
          radio:=0;
          writeln('LOS PUNTOS SON COLINEALES');
          delay(1000);
          EXIT;
        end;
    'C':begin
          xc:=x2;
          yc:=y1+m1*(xc-x1);
        end;
    'D':begin
          xc:=x1;
          yc:=y2+m2*(xc-x2);
```

```
            end;
        END; {fin del caso}
        radio:=sqrt(sqr(xi-xc)+sqr(yi-yc));
    end; {procedimiento que calcula el h,k,r de la circunferencia}


function detarea(xi,xj,xk,yi,yj,yk:real):real;

    var
      aux1,aux2,aux3:real;

    begin
       aux1:=xj*yk-yj*xk;
       aux2:=xi*(yk-yj);
       aux3:=yi*(xk-xj);
       detarea:=abs(0.5*(aux1-aux2+aux3));
    end;

function potencia(radio,xc,yc,xpi,ypi:real):real;

   var
     aux1,aux2:real;

   begin
     aux1:=sqr(xc-xpi)+sqr(yc-ypi);
     aux2:=sqrt(aux1);
     potencia:=aux2-radio;
   end;

function corvar(xi,xj,xk,yi,yj,yk,xpi,ypi:real):real;

   var
    ar1,ar2,ar3:real;

   begin
     ar1:=detarea(xpi,xi,xj,ypi,yi,yj);
     ar2:=detarea(xpi,xj,xk,ypi,yj,yk);
     ar3:=detarea(xpi,xk,xi,ypi,yk,yi);
     ar1:=abs(ar1);
     ar2:=abs(ar2);
     ar3:=abs(ar3);
     corvar:=ar1+ar2+ar3;
   end;
```

```
{   procedure inicial1;
     begin
       marcir:=nil;
     end;}

{   procedure libera1;

     var
       aux:indica;

     begin
       aux:=marcir;
       while aux<>nil do
          begin
           aux^.elimina:='si';
           aux:=aux^.pega;
          end;
     end;
               }

{   procedure inser1;

      var
        aux:indica;

      begin
         new(aux);
         aux^.numpe[1]:=borracir;
         aux^.numpe[2]:=i1;
         aux^.numpe[3]:=i2;
         aux^.numpe[4]:=i3;
         aux^.elimina:='no';
         aux^.pega:=marcir;
         marcir:=aux;
      end;     }

{    procedure escribeq;
       var
         aux:indica;
       begin
         aux:=marcir;
         while aux<>nil do
           begin
```

```
            writeln('bc:',aux^.numpe[1]);
            writeln('i1:',aux^.numpe[2]);
            writeln('i2:',aux^.numpe[3]);
            writeln('i3:',aux^.numpe[4]);
            xw:=readkey;
            aux:=aux^.pega;
          end;
    end;  }


procedure inicializar1;
    begin
      sal_cor:=nil;
    end;


 procedure insertar1(i1,i2,i3:integer; xc,yc:real; var lista:puntero);

    var
      aux:puntero;

    begin
      new(aux);
      aux^.ncirc:=ncir;
      aux^.h:=xc;
      aux^.k:=yc;
      aux^.r:=radio;
      aux^.nump[1]:=i1;
      aux^.nump[2]:=i2;
      aux^.nump[3]:=i3;
      aux^.elimina:='no';
      aux^.enlace:=lista;
      lista:=aux;
    end;


 procedure inicializar2;
   begin
     sal_cir:=nil;
   end;
```

```
procedure insertar2;
   var
     aux:nodo;

   begin
     new(aux);
     aux^.cir:=ncir;
     aux^.nump[1]:=i1;
     aux^.nump[2]:=i2;
     aux^.nump[3]:=i3;
     aux^.elimina:='no';
     aux^.conec:=sal_cir;
     sal_cir:=aux;
   end;


procedure inicializar3;
   begin
     sal_pto:=nil;
   end;


procedure insertar3;

    var
      aux:indi;
      ap:integer;

procedure insaux;
   begin
      new(aux);
      aux^.numc:=ncir;
      aux^.npto:=ap;
      aux^.elimina:='no';
      aux^.direc:=sal_pto;
      sal_pto:=aux;
   end;

    begin
      ap:=i1;
      insaux;
      ap:=i2;
      insaux;
```

```
     ap:=i3;
     insaux;
   end;


procedure insertatodo;

    begin
     if radio<>0 then
      begin
        insertar1(i1,i2,i3,xc,yc,sal_cor);
        insertar3;
        insertar2;
      end;
  end;




procedure escribe1(lista:puntero);

  var
    aux:puntero;

  begin
    aux:=lista;
    while aux<>nil do
      begin
         if aux^.elimina<>'si' then
          begin
             write('  Circunferencia num.:');
             writeln(aux^.ncirc);
             writeln('  Las coordenadas del centro y el radio son:');
             writeln(' X          Y         radio');
             writeln(aux^.h:4:2,aux^.k:11:2,aux^.r:11:2);
             writeln('  Los puntos que forman esta circunferencia son:');
             writeln('     punto 1:',aux^.nump[1]);
             writeln('     punto 2:',aux^.nump[2]);
             writeln('     punto 3:',aux^.nump[3]);
             rw:=readkey;
          end;
       aux:=aux^.enlace;
      end;
```

```
    end;

  procedure escribe2;

    var
      aux:nodo;
      ia:integer;

    begin
     aux:=sal_cir;
      while aux<>nil do
        begin
          if aux^.elimina='no' then
            begin
              writeln(' En la circunferencia:',aux^.cir,' estan los puntos:');
              for ia:=1 to 3 do
              writeln('    punto:',aux^.nump[ia]);
              xw:=readkey;
            end;
          aux:=aux^.conec;
        end;
    end;

{   procedure escribe3;

    begin
      while sal_pto<>nil do
        begin
          writeln(' En el punto:',sal_pto^.npto,' esta la circunferencia:',sal_pto
          sal_pto:=sal_pto^.direc;
          xw:=readkey;
        end;
    end;     }


   procedure graba;

    var
      archi:text;
      aux:puntero;
      gr:char;

      procedure leenomb;
       var
```

```
        cu:char;
      begin
        nombre:='';
        cu:=readkey;
        write(cu);
        while cu<>#13 do
          begin
             nombre:=nombre+cu;
            { outtextxy(ex,ey,nombre); }
             cu:=readkey;
             write(cu);
          end;
      end;

  begin
    aux:=sal_cor;
    write('Quieres guardar resultados en archivo s/n ?');
    gr:=readkey;
    if (gr='s') or (gr='S') then
     begin
        write('Dame nombre y drive del archivo:');
        leenomb;
        assign(archi,nombre);
        rewrite(archi);
        while aux<>nil do
          begin
               writeln(archi,aux^.ncirc);
               writeln(archi,aux^.h:11:3,' ',aux^.k:11:3,' ',aux^.r:11:3);
               writeln(archi,'       ',aux^.nump[1]);
               writeln(archi,'       ',aux^.nump[2]);
               writeln(archi,'       ',aux^.nump[3]);
               writeln(archi,'                       ');
              aux:=aux^.enlace;
          end;
       close(archi);
     end;
  end;

procedure borraia(var lista:puntero);

    var
      aux2:puntero;

    begin
```

```
    aux2:=lista;
    while aux2<> nil do
      begin
       aux2^.elimina:='si';
       aux2:=aux2^.enlace;
      end;
  end;


procedure borra1(var lista:puntero);

  var
    aux2:puntero;

  begin
    aux2:=lista;
    while aux2<> nil do
      begin
        if aux2^.ncirc=borracir then
          aux2^.elimina:='si';
       aux2:=aux2^.enlace;
      end;
  end;

procedure desaloja;

  var
    aux2:puntero;

  begin
    aux2:=sal_ins;
    while aux2<> nil do
      begin
       aux2^.elimina:='si';
       aux2:=aux2^.enlace;
      end;
  end;

procedure borra2;

  var
    aux2:nodo;
```

```
    begin
      aux2:=sal_cir;
      while aux2<> nil do
         begin
            if aux2^.cir=borracir then
              aux2^.elimina:='si';
          aux2:=aux2^.conec;
         end;
    end;


  procedure borra3;

    var
      aux2:indi;

    begin
      aux2:=sal_pto;
      while aux2<> nil do
         begin
           if aux2^.numc=borracir then
             aux2^.elimina:='si';
           aux2:=aux2^.direc;
         end;
    end;


  procedure borratodo;

    begin
      borra1(sal_cor);
      borra2;
      borra3;
    end;


procedure detcir;

  var
    auc:integer;
    poa,poa1:real;
    xpa,ypa:real;

    begin
```

```
 borratodo;
 determinahkr(xpi,xj,xk,ypi,yj,yk,xc,yc,radio);
 ncmax:=ncmax+1;
 ncir:=ncmax;
 i1:=cont;
 i2:=j;
 i3:=k;
 poa1:=1;
 for auc:=cont-1 downto 1 do
    begin
      xpa:=datos[auc,1];
      ypa:=datos[auc,2];
      poa:=potencia(radio,xc,yc,xpa,ypa);
      if poa<-0.000001 then
        poa1:=poa;
    end;
if poa1>-0.000001 then
    insertatodo;

 determinahkr(xpi,xk,xi,ypi,yk,yi,xc,yc,radio);
 ncmax:=ncmax+1;
 ncir:=ncmax;
 i1:=cont;
 i2:=k;
 i3:=i;
 poa1:=1;
 for auc:=cont-1 downto 1 do
   begin
     xpa:=datos[auc,1];
     ypa:=datos[auc,2];
     poa:=potencia(radio,xc,yc,xpa,ypa);
     if poa<-0.000001 then
       poa1:=poa;
   end;
if poa1>-0.000001 then
    insertatodo;

 determinahkr(xpi,xi,xj,ypi,yi,yj,xc,yc,radio);
 ncmax:=ncmax+1;
 ncir:=ncmax;
 i1:=cont;
 i2:=i;
 i3:=j;
 poa1:=1;
```

```
        for auc:=cont-1 downto 1 do
          begin
            xpa:=datos[auc,1];
            ypa:=datos[auc,2];
            poa:=potencia(radio,xc,yc,xpa,ypa);
            if poa<-0.000001 then
              poa1:=poa;
          end;
        if poa1>-0.000001 then
          insertatodo;

    end;

procedure detcir1;

    var
      ar1,ar2,ar3:real;
      xpa,ypa:real;
      auc:integer;
      poa,poa1:real;

     function detar(xi,xj,xk,yi,yj,yk:real):real;

        var
          aux1,aux2,aux3:real;

        begin
           aux1:=xj*yk-yj*xk;
           aux2:=xi*(yk-yj);
           aux3:=yi*(xk-xj);
           detar:=0.5*(aux1-aux2+aux3);
        end;

    begin
      borratodo;
      ar1:=detar(xpi,xj,xk,ypi,yj,yk);
      ar2:=detar(xpi,xk,xi,ypi,yk,yi);
      ar3:=detar(xpi,xi,xj,ypi,yi,yj);
      if (ar1>0) and (ar2>0) and (ar3<0)
        then
         begin
            ncmax:=ncmax+1;
            ncir:=ncmax;
            i1:=cont;
```

```
        i2:=j;
        i3:=k;
        determinahkr(xpi,xj,xk,ypi,yj,yk,xc,yc,radio);
        poa1:=1;
        for auc:=cont-1 downto 1 do
          begin
           xpa:=datos[auc,1];
           ypa:=datos[auc,2];
           poa:=potencia(radio,xc,yc,xpa,ypa);
           if poa<-0.000001 then
             poa1:=poa;
          end;
        if poa1>-0.000001 then
            insertatodo;

        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=cont;
        i2:=k;
        i3:=i;
        determinahkr(xpi,xk,xi,ypi,yk,yi,xc,yc,radio);
        poa1:=1;
        for auc:=cont-1 downto 1 do
          begin
           xpa:=datos[auc,1];
           ypa:=datos[auc,2];
           poa:=potencia(radio,xc,yc,xpa,ypa);
           if poa<-0.000001 then
             poa1:=poa;
          end;
        if poa1>-0.000001 then
            insertatodo;

     end;
if  (ar1<0) and (ar2<0) and (ar3>0)
    then
     begin
        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=cont;
        i2:=j;
        i3:=k;
        determinahkr(xpi,xj,xk,ypi,yj,yk,xc,yc,radio);
        poa1:=1;
```

```
        for auc:=cont-1 downto 1 do
          begin
           xpa:=datos[auc,1];
           ypa:=datos[auc,2];
           poa:=potencia(radio,xc,yc,xpa,ypa);
           if poa<-0.000001 then
             poa1:=poa;
          end;
        if poa1>-0.000001 then
            insertatodo;

        ncir:=ncmax;
        i1:=cont;
        i2:=k;
        i3:=i;
        determinahkr(xpi,xk,xi,ypi,yk,yi,xc,yc,radio);
        poa1:=1;
        for auc:=cont-1 downto 1 do
          begin
           xpa:=datos[auc,1];
           ypa:=datos[auc,2];
           poa:=potencia(radio,xc,yc,xpa,ypa);
           if poa<-0.000001 then
             poa1:=poa;
          end;
        if poa1>-0.000001 then
            insertatodo;

    end;
if (ar1>0) and (ar2<0) and (ar3>0)
  then
    begin
       ncmax:=ncmax+1;
       ncir:=ncmax;
       i1:=cont;
       i2:=j;
       i3:=k;
       determinahkr(xpi,xj,xk,ypi,yj,yk,xc,yc,radio);
       poa1:=1;
       for auc:=cont-1 downto 1 do
         begin
          xpa:=datos[auc,1];
          ypa:=datos[auc,2];
          poa:=potencia(radio,xc,yc,xpa,ypa);
```

```
            if poa<-0.000001 then
              poa1:=poa;
            end;
          if poa1>-0.000001 then
             insertatodo;

          ncmax:=ncmax+1;
          ncir:=ncmax;
          i1:=cont;
          i2:=i;
          i3:=j;
          determinahkr(xpi,xi,xj,ypi,yi,yj,xc,yc,radio);
          poa1:=1;
          for auc:=cont-1 downto 1 do
            begin
             xpa:=datos[auc,1];
             ypa:=datos[auc,2];
             poa:=potencia(radio,xc,yc,xpa,ypa);
             if poa<-0.000001 then
               poa1:=poa;
            end;
          if poa1>-0.000001 then
             insertatodo;

      end;
  if (ar1<0) and (ar2>0) and (ar3<0)
    then
      begin
         ncmax:=ncmax+1;
         ncir:=ncmax;
         i1:=cont;
         i2:=j;
         i3:=k;
         determinahkr(xpi,xj,xk,ypi,yj,yk,xc,yc,radio);
         poa1:=1;
         for auc:=cont-1 downto 1 do
           begin
            xpa:=datos[auc,1];
            ypa:=datos[auc,2];
            poa:=potencia(radio,xc,yc,xpa,ypa);
            if poa<-0.000001 then
              poa1:=poa;
           end;
         if poa1>-0.000001 then
```

```
            insertatodo;

          ncmax:=ncmax+1;
          ncir:=ncmax;
          i1:=cont;
          i2:=i;
          i3:=j;
          determinahkr(xpi,xi,xj,ypi,yi,yj,xc,yc,radio);
          poa1:=1;
          for auc:=cont-1 downto 1 do
            begin
             xpa:=datos[auc,1];
             ypa:=datos[auc,2];
             poa:=potencia(radio,xc,yc,xpa,ypa);
             if poa<-0.000001 then
               poa1:=poa;
            end;
          if poa1>-0.000001 then
              insertatodo;

      end;
  if (ar1<0) and (ar2>0) and (ar3>0)
    then
      begin
          ncmax:=ncmax+1;
          ncir:=ncmax;
          i1:=cont;
          i2:=k;
          i3:=i;
          determinahkr(xpi,xk,xi,ypi,yk,yi,xc,yc,radio);
          poa1:=1;
          for auc:=cont-1 downto 1 do
            begin
             xpa:=datos[auc,1];
             ypa:=datos[auc,2];
             poa:=potencia(radio,xc,yc,xpa,ypa);
             if poa<-0.000001 then
               poa1:=poa;
            end;
          if poa1>-0.000001 then
              insertatodo;

          ncmax:=ncmax+1;
          ncir:=ncmax;
```

```
            i1:=cont;
            i2:=i;
            i3:=j;
            determinahkr(xpi,xi,xj,ypi,yi,yj,xc,yc,radio);
            poa1:=1;
            for auc:=cont-1 downto 1 do
              begin
               xpa:=datos[auc,1];
               ypa:=datos[auc,2];
               poa:=potencia(radio,xc,yc,xpa,ypa);
               if poa<-0.000001 then
                 poa1:=poa;
              end;
            if poa1>-0.000001 then
                insertatodo;

        end;
    if (ar1>0) and (ar2<0) and (ar3<0)
       then
         begin
            ncmax:=ncmax+1;
            ncir:=ncmax;
            i1:=cont;
            i2:=k;
            i3:=i;
            determinahkr(xpi,xi,xk,ypi,yi,yk,xc,yc,radio);
            poa1:=1;
            for auc:=cont-1 downto 1 do
              begin
               xpa:=datos[auc,1];
               ypa:=datos[auc,2];
               poa:=potencia(radio,xc,yc,xpa,ypa);
               if poa<-0.000001 then
                 poa1:=poa;
              end;
            if poa1>-0.000001 then
                insertatodo;

            ncmax:=ncmax+1;
            ncir:=ncmax;
            i1:=cont;
            i2:=i;
            i3:=j;
            determinahkr(xpi,xi,xj,ypi,yi,yj,xc,yc,radio);
```

```
            poa1:=1;
            for auc:=cont-1 downto 1 do
              begin
               xpa:=datos[auc,1];
               ypa:=datos[auc,2];
               poa:=potencia(radio,xc,yc,xpa,ypa);
               if poa<-0.000001 then
                 poa1:=poa;
              end;
            if poa1>-0.000001 then
                insertatodo;

        end;
    end;


procedure  ubica1;

  var
    acv,ar:real;
    aux:nodo;

    begin
       xi:=datos[i,1];
       xj:=datos[j,1];
       xk:=datos[k,1];
       yi:=datos[i,2];
       yj:=datos[j,2];
       yk:=datos[k,2];
       acv:=corvar(xi,xj,xk,yi,yj,yk,xpi,ypi);
       ar:=detarea(xi,xj,xk,yi,yj,yk);
       writeln('acv=',acv,'    ar=',ar);
       xw:=readkey;
       if acv<=ar then
        BEGIN
          clrscr;
          WRITELN('ESTOY EN DETCIR CON EL PUNTO:',CONT);
          XW:=READKEY;
          detcir;
          {libera1;}
        END;
       if acv>ar then
         begin
          clrscr;
```

```
          WRITELN('ESTOY EN DETCIR1 CON EL PUNTO:',CONT);
          XW:=READKEY;
          detcir1;
          {libera1;}
        end;
    end;



procedure donde_pto;

   var
     poa:real;
     op:boolean;
     aux1,aux:puntero;

   begin
        op:=true;
        aux:=sal_cor;
        while aux<>nil do
          begin
             ncir:=aux^.ncirc;
             xc:=aux^.h;
             yc:=aux^.k;
             radio:=aux^.r;
             i:=aux^.nump[1];
             j:=aux^.nump[2];
             k:=aux^.nump[3];
             if aux^.elimina<>'si' then
               begin
                   poa:=potencia(radio,xc,yc,xpi,ypi);
                   writeln(' potencia:',poa);
                   xw:=readkey;
                   if poa<0 then
                     begin
                       borracir:=ncir;
                       ubica1;
                       op:=false;
                     end;
               end;
            aux:=aux^.enlace;
          end;
```

```
  end;


procedure recorrido;

   begin
     CONT:=0;
     xi:=datos[1,1];
     xj:=datos[2,1];
     xk:=datos[3,1];
     yi:=datos[1,2];
     yj:=datos[2,2];
     yk:=datos[3,2];
     ncir:=1;
     i1:=1;
     i2:=2;
     i3:=3;
     determinahkr(xi,xj,xk,yi,yj,yk,xc,yc,radio);
     {inicial1;}
     inicializar1;
     inicializar2;
     inicializar3;
     insertar1(i1,i2,i3,xc,yc,sal_cor);
     insertar2;
     insertar3;
     ncmax:=ncir;
     for cont:=4 to np do
       begin
         writeln('PROCESANDO EL PUNTO:',cont);
         xpi:=datos[cont,1];
         ypi:=datos[cont,2];
         xw:=readkey;
         donde_pto;

       end;
   end;


procedure escptos;

    var
      ir:integer;

    begin
```

```
      writeln(' Los puntos seleccionados son:');
      xw:=readkey;
      for ir:=np+1 to pi do
        begin
          write('x:',datos[ir,1]:2:2);
          writeln('   y:',datos[ir,2]:2:2);
          xw:=readkey;
        end;
    end;

procedure grafica(var datos:arreglo);

    var
      ycormax,xcormax:real;
      conver1,conver2:real;
      conver:real;
      Directorio:string;
      CodError,GraphDriver:integer;
      cu:char;
      xr1,xr2,yr1,yr2:real;
      x1,x2,y1,y2:real;
      rel,rely:real;
      col,pmx,pmy:integer;


     procedure InicializaSistemaGrafico(Directorio:string;
                                        Var CodError:integer;
                                        Var Graphdriver:integer);

         var
           GraphMode:integer;

         begin
           GraphDriver:=Detect;
           InitGraph(GraphDriver,GraphMode,directorio);
           CodError:=GraphResult;
         end;

     procedure ponpuntos;

        var
          posx,posy,iw:integer;
          color:word;
```

```
      begin
        conver1:=round((x2-x1)/(xr2-xr1));
        conver2:=round((y2-y1)/(yr2-yr1));
        for iw:=1 to np do
          begin
            posx:=round(datos[iw,1]*conver1);
            posy:=round(pmy-(datos[iw,2]*conver2));
            putpixel(posx,posy,color);
            writeln('punto:',iw,'  x:',posx,'  y:',posy);
          end;
      end;

procedure conviertetexto(var palabra:real; var crx:cadena);


    begin
      str(palabra:2:2,crx);
    end;



procedure muevepunto(var datos:arreglo);

  var
    kte,px,py:real;
    pm:char;
    pia,pxa1,pya1:integer;
    pxa,pya:real;
    color:word;


 begin
   pia:=np;
   kte:=pmy/conver2;
   color:=white;
   px:=50;
   py:=50;
   pxa:=(px/conver1);
   pya:=(kte-(py/conver2));
   pxa1:=round(px);
   pya1:=round(py);
   putpixel(pxa1,pya1,color);
   repeat
    pm:=readkey;
```

```
    setcolor(black);
    outtextxy(100,2,'X:');
    convierteteXto(pxa,crx);
    outtextxy(120,2,crx);
    outtextxy(200,2,'Y:');
    convierteteXto(pya,crx);
    outtextxy(220,2,crx);
    color:=black;
    putpixel(pxa1,pya1,color);
case ord(pm) of

  77:px:=px+2;
  75:px:=px-2;
  72:py:=py-1;
  80:py:=py+1;
  81: begin
       pxa:=(px/conver1);
       pya:=(kte-(py/conver2));
       pia:=pia+1;
       datos[pia,1]:=pxa;
       datos[pia,2]:=pya;
       setcolor(white);
       line(pxa1-5,pya1,pxa1+5,pya1);
       line(pxa1,pya1+5,pxa1,pya1-5);
       outtextxy(40,10,' Puntos seleccionados:');
       outtextxy(50,20,'x:');
       convierteteXto(pxa,crx);
       outtextxy(65,20,crx);
       outtextxy(50,30,'y:');
       convierteteXto(pya,crx);
       outtextxy(65,30,crx);
      end
     else
      begin
       setcolor(black);
       outtextxy(40,10,' Puntos seleccionados:');
       outtextxy(50,20,'x:');
       convierteteXto(pxa,crx);
       outtextxy(65,20,crx);
       outtextxy(50,30,'y:');
       convierteteXto(pya,crx);
       outtextxy(65,30,crx);
      end;
   end;
```

```
      setcolor(white);
      pxa:=(px/conver1);
      pya:=(kte-(py/conver2));
      outtextxy(100,2,'X:');
      conviertetexto(pxa,crx);
      outtextxy(120,2,crx);
      outtextxy(200,2,'Y:');
      conviertetexto(pya,crx);
      outtextxy(220,2,crx);
      pxa1:=round(px);
      pya1:=round(py);
      color:=white;
      putpixel(pxa1,pya1,color);

    until
       pm='n';
   pi:=pia;
end;

      procedure poncirculos;

         var
           aux1,aux:puntero;
           xcen,ycen,radio:integer;
           color:word;

         begin
           aux:=sal_cor;
           conver1:=round((x2-x1)/(xr2-xr1));
           conver2:=round((y2-y1)/(yr2-yr1));
           while aux<>nil do
             begin
              if aux^.elimina<>'si' then
                begin
                  xcen:=round(conver1*(aux^.h));
                  ycen:=round(pmy-(conver2*aux^.k));
                  radio:=round(conver1*aux^.r);
                 {putpixel(xcen,ycen,color); }
                  rw:=readkey;
                  circle(xcen,ycen,radio);
                  gotoxy(45,3);
                  rw:=readkey;
                end;
              aux:=aux^.enlace;
```

```
      end;
    aux1:=sal_ins;
    while aux1<>nil do
      begin
         xcen:=round(conver1*aux1^.h);
          ycen:=round(350-(conver2*aux1^.k));
          radio:=round(conver1*aux1^.r);
          {putpixel(xcen,ycen,color); }
          xw:=readkey;
          circle(xcen,ycen,radio);
          gotoxy(45,3);
          xw:=readkey;
          aux1:=aux1^.enlace;
      end;


    muevepunto(datos);
  end;


begin
  clrscr;
  write('Dame directorio del Archivo GRAPH.TPU:');
  readln(Directorio);
  writeln('De las cordenadas reales de su pantalla:');
  write('  De x1:'); readln(xr1);
  write('  De y1:'); readln(yr1);
  write('  De x2:'); readln(xr2);
  write('  De y2:'); readln(yr2);
  write('Tiene una pantalla de 640x400 pixeles s/n');
  cu:=readkey;
  inicializaSistemaGrafico('c:\tp6\bgi',CodError,Graphdriver);
  if CodError=0 then
    begin
       if cu='n' then
        begin
          setviewport(30,10,600,200,clipon);
          line(1,25,569,25);
          line(569,25,569,189);
          line(569,189,1,189);
          line(1,189,1,25);
          x1:=1; x2:=569; y1:=1; y2:=189;
          col:=15;
          pmy:=189;
```

```
            ponpuntos;
            xw:=readkey;
            poncirculos;
            xw:=readkey;
            closegraph
          end;
        if cu='s' then
          begin
            setviewport(30,10,600,400,true);
            line(1,1,569,1);
            line(569,1,569,389);
            line(569,389,1,389);
            line(1,389,1,1);
            x1:=1; x2:=569; y1:=1; y2:=389;
            pmy:=389;
            col:=1;
            ponpuntos;
            xw:=readkey;
            poncirculos;
            xw:=readkey;
            closegraph
          end;

      end

    else
      writeln('  ERROR EN EL SISTEMA GRAFICO');
end;



procedure circunferenciasnvas;

    var
      poa,poa1:real;
      cirden,ix:integer;
  {   mar:indica;    }


 procedure checaptos;

    var
      aux1:puntero;
```

```
    xpa,ypa:real;

 begin
      poa1:=1;
      aux1:=sal_cor;
      while aux1<>nil do
        begin
          poa:=potencia(aux1^.r,aux1^.h,aux1^.k,datos[ix,1],datos[ix,2]);
          if (poa<0.0) and (aux1^.elimina<>'si') then
            begin
              xpa:=datos[aux1^.nump[1],1];
              ypa:=datos[aux1^.nump[1],2];
              poa:=potencia(radio,xc,yc,xpa,ypa);
              if poa<-0.000001 then
                  poa1:=poa;
              xpa:=datos[aux1^.nump[2],1];
              ypa:=datos[aux1^.nump[2],2];
              poa:=potencia(radio,xc,yc,xpa,ypa);
              if poa<-0.000001 then
                 poa1:=poa;
              xpa:=datos[aux1^.nump[3],1];
              ypa:=datos[aux1^.nump[3],2];
              poa:=potencia(radio,xc,yc,xpa,ypa);
              if poa<-0.000001 then
                poa1:=poa;
            end;
         aux1:=aux1^.enlace;
      end;

  end;

procedure creaptos1;


   begin
      determinahkr(xpi,xj,xk,ypi,yj,yk,xc,yc,radio);
      ncmax:=ncmax+1;
      ncir:=ncmax;
      i1:=ix;
      i2:=j;
      i3:=k;
      checaptos;
     if poa1>-0.000001 then
      insertar1(i1,i2,i3,xc,yc,sal_ins);
```

```
        determinahkr(xpi,xk,xi,ypi,yk,yi,xc,yc,radio);
        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=k;
        i3:=i;
        checaptos;
      if poa1>-0.000001 then
         insertar1(i1,i2,i3,xc,yc,sal_ins);

      determinahkr(xpi,xi,xj,ypi,yi,yj,xc,yc,radio);
      ncmax:=ncmax+1;
      ncir:=ncmax;
      i1:=ix;
      i2:=i;
      i3:=j;
      checaptos;
      if poa1>-0.000001 then
         insertar1(i1,i2,i3,xc,yc,sal_ins);

  end;


procedure creaptos2;

  var
     ar1,ar2,ar3:real;
     xpa,ypa:real;
     auc:integer;
     poa,poa1:real;

  function detar(xi,xj,xk,yi,yj,yk:real):real;

     var
       aux1,aux2,aux3:real;

     begin
        aux1:=xj*yk-yj*xk;
        aux2:=xi*(yk-yj);
        aux3:=yi*(xk-xj);
        detar:=0.5*(aux1-aux2+aux3);
    end;
```

```
begin
  ar1:=detar(xpi,xj,xk,ypi,yj,yk);
  ar2:=detar(xpi,xk,xi,ypi,yk,yi);
  ar3:=detar(xpi,xi,xj,ypi,yi,yj);
  if (ar1>0) and (ar2>0) and (ar3<0)
    then
     begin
        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=j;
        i3:=k;
        determinahkr(xpi,xj,xk,ypi,yj,yk,xc,yc,radio);
        checaptos;
        if poai>-0.000001 then
            insertar1(i1,i2,i3,xc,yc,sal_ins);

        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=k;
        i3:=i;
        determinahkr(xpi,xk,xi,ypi,yk,yi,xc,yc,radio);
        checaptos;
        if poai>-0.000001 then
            insertar1(i1,i2,i3,xc,yc,sal_ins);

     end;
 if  (ar1<0) and (ar2<0) and (ar3>0)
    then
     begin
        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=j;
        i3:=k;
        determinahkr(xpi,xj,xk,ypi,yj,yk,xc,yc,radio);
        checaptos;
        if poai>-0.000001 then
            insertar1(i1,i2,i3,xc,yc,sal_ins);

        ncir:=ncmax;
        i1:=ix;
```

```
        i2:=k;
        i3:=i;
        determinahkr(xpi,xk,xi,ypi,yk,yi,xc,yc,radio);
        checaptos;
        if poa1>-0.000001 then
            insertar1(i1,i2,i3,xc,yc,sal_ins);

    end;
if (ar1>0) and (ar2<0) and (ar3>0)
  then
    begin
        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=j;
        i3:=k;
        determinahkr(xpi,xj,xk,ypi,yj,yk,xc,yc,radio);
        checaptos;
        if poa1>-0.000001 then
            insertar1(i1,i2,i3,xc,yc,sal_ins);

        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=i;
        i3:=j;
        determinahkr(xpi,xi,xj,ypi,yi,yj,xc,yc,radio);
        checaptos;
        if poa1>-0.000001 then
            insertar1(i1,i2,i3,xc,yc,sal_ins);

    end;
if (ar1<0) and (ar2>0) and (ar3<0)
  then
    begin
        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=j;
        i3:=k;
        determinahkr(xpi,xj,xk,ypi,yj,yk,xc,yc,radio);
        checaptos;
        if poa1>-0.000001 then
            insertar1(i1,i2,i3,xc,yc,sal_ins);
```

```
        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=i;
        i3:=j;
        determinahkr(xpi,xi,xj,ypi,yi,yj,xc,yc,radio);
        checaptos;
        if poa1>-0.000001 then
            insertar1(i1,i2,i3,xc,yc,sal_ins);

     end;
 if (ar1<0) and (ar2>0) and (ar3>0)
   then
     begin
        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=k;
        i3:=i;
        determinahkr(xpi,xk,xi,ypi,yk,yi,xc,yc,radio);
        checaptos;
        if poa1>-0.000001 then
            insertar1(i1,i2,i3,xc,yc,sal_ins);

        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=i;
        i3:=j;
        determinahkr(xpi,xi,xj,ypi,yi,yj,xc,yc,radio);
        checaptos;
        if poa1>-0.000001 then
            insertar1(i1,i2,i3,xc,yc,sal_ins);

     end;
 if (ar1>0) and (ar2<0) and (ar3<0)
    then
     begin
        ncmax:=ncmax+1;
        ncir:=ncmax;
        i1:=ix;
        i2:=k;
        i3:=i;
```

```
            determinahkr(xpi,xi,xk,ypi,yi,yk,xc,yc,radio);
            checaptos;
            if poa1>-0.000001 then
                insertar1(i1,i2,i3,xc,yc,sal_ins);

            ncmax:=ncmax+1;
            ncir:=ncmax;
            i1:=ix;
            i2:=i;
            i3:=j;
            determinahkr(xpi,xi,xj,ypi,yi,yj,xc,yc,radio);
            checaptos;
            if poa1>-0.000001 then
                insertar1(i1,i2,i3,xc,yc,sal_ins);

        end;
end;


procedure  analizaforma;

var
  acv,ar:real;

 begin
    xi:=datos[i,1];
    xj:=datos[j,1];
    xk:=datos[k,1];
    yi:=datos[i,2];
    yj:=datos[j,2];
    yk:=datos[k,2];
    acv:=corvar(xi,xj,xk,yi,yj,yk,xpi,ypi);
    ar:=detarea(xi,xj,xk,yi,yj,yk);
    if acv<=ar then
     begin
      xw:=readkey;
      creaptos1;
     end;
   if acv>ar then
     begin
      xw:=readkey;
      creaptos2
    end;
end;
```

```
procedure analizapto;


   var
     poa:real;
     aux:puntero;

   begin
      aux:=sal_cor;
      while aux<>nil do
       begin
          ncir:=aux^.ncirc;
          cirden:=ncir;
          xc:=aux^.h;
          yc:=aux^.k;
          radio:=aux^.r;
          i:=aux^.nump[1];
          j:=aux^.nump[2];
          k:=aux^.nump[3];
          if aux^.elimina<>'si' then
            begin
               poa:=potencia(radio,xc,yc,xpi,ypi);
               if poa<-0.000001 then
                  BEGIN
                  { inscir;    }
                   analizaforma;
                  END;
             end;
           aux:=aux^.enlace;
        end;

  end;


procedure crealista(lista,listai:puntero; numero:integer;
                    var lista2:puntero);


  var
   aux,aux2:puntero;
```

```
begin
  aux:=lista;  {sal_cor}
  while aux<>nil do
      begin
        if aux^.elimina<>'si' then
          begin
          ncir:=aux^.ncirc;
          cirden:=ncir;
          xc:=aux^.h;
          yc:=aux^.k;
          radio:=aux^.r;
          i1:=aux^.nump[1];
          i2:=aux^.nump[2];
          i3:=aux^.nump[3];
               poa:=potencia(radio,xc,yc,xpi,ypi);
               if poa<-0.000001 then
                   if (i1=numero) or (i2=numero) or (i3=numero) then
                       insertar1(i1,i2,i3,xc,yc,sal_lis);
          end;
          aux:=aux^.enlace;
       end;

   aux:=lista1;  {sal_ins;}
   while aux<>nil do
      begin
         if aux^.elimina<>'si' then
          begin
          ncir:=aux^.ncirc;
          cirden:=ncir;
          xc:=aux^.h;
          yc:=aux^.k;
          radio:=aux^.r;
          i1:=aux^.nump[1];
          i2:=aux^.nump[2];
          i3:=aux^.nump[3];
               poa:=potencia(radio,xc,yc,xpi,ypi);
               if (i1=numero) or (i2=numero) or (i3=numero) then
                   insertar1(i1,i2,i3,xc,yc,sal_lis);
          end;
          aux:=aux^.enlace;
       end;
   end;
```

```
procedure puntosfijos(lista:puntero; var xf,yf:real);

  var
   aux:puntero;
   yb:real;

   begin
     yb:=30000;
     aux:=lista;
     while aux<>nil do
       begin
        if aux^.elimina<>'si' then
         if yb>aux^.k then
           begin
             yf:=aux^.k;
             xf:=aux^.h;
             yb:=yf;
           end;
         aux:=aux^.enlace;
       end;

   end;



procedure angulos(xf,yf:real; cont:integer; puntosarea:lis_ar;
                  lista:puntero; var xam,yam:real);

      var
        xm,ym,a1:real;
        amin:real;
        aux1,aux2:puntero;
        apunt:integer;
        relog:boolean;

      procedure primercuadrante;

         begin

            a1:=(ym-yf)/(xm-xf);
            a1:=arctan(a1);
         end;

       procedure segundocuadrante;
```

```
    begin

      a1:=(ym-yf)/(xm-xf);
      a1:=arctan(a1);
      a1:=3.1416+a1;
    end;

procedure tercercuadrante;

    begin

       a1:=(ym-yf)/(xm-xf);
       a1:=arctan(a1)+3.1416;
    end;

procedure cuartocuadrante;

    begin

      a1:=(ym-yf)/(xm-xf);
      a1:=arctan(a1);
      a1:=6.283185307+a1;
    end;

begin {ANGULOS}
   amin:=6.283185307;
   aux1:=lista;
   while aux1<>nil do
     begin
      if aux1^.elimina<>'si' then
       begin
        xm:=aux1^.h;
        ym:=aux1^.k;
        relog:=true;
        for apunt:=1 to cont do
          if (puntosarea[apunt,1]=xm) and (puntosarea[apunt,2]=ym)
           then
              relog:=false;
        if relog=true then
          begin
           if (xf<xm) and (yf<=ym)
             then
               begin
```

```
      if ym=yf then
        a1:=0;
      if (ym<>yf) and (xm<>xf) then
        primercuadrante;
      if a1<amin then
        begin
          amin:=a1;
          xam:=xm;
          yam:=ym;
        end;
    end;
if (xm<=xf) and (ym>yf)
   then
     begin
        if xm=xf then
          a1:=1.570796327;
        if (xm<>xf) and (ym<>yf) then
          segundocuadrante;
        if a1<amin then
          begin
            amin:=a1;
            xam:=xm;
            yam:=ym;
          end;
     end;

if (xf>xm) and (yf>=ym)
    then
      begin
         if ym=yf then
            a1:=3.141592654;
         if (ym<>yf) and (xm<>yf) then
            tercercuadrante;
         if a1<amin then
           begin
             amin:=a1;
             xam:=xm;
             yam:=ym;
           end;
      end;

 if (xm>=xf) and (ym<yf)
    then
      begin
```

```
                        if xf=xm then
                          a1:=2.35619449;
                        if (xf<>xm) and (ym<>yf) then
                          cuartocuadrante;
                        if a1<amin then
                           begin
                              amin:=a1;
                              xam:=xm;
                              yam:=ym;
                           end;
                     end;
                  end;
               end;
                aux1:=aux1^.enlace;
            end; {WHILE}
         end;{PROCEDIMIENTO}

procedure det_arreglo(lista:puntero; xf,yf:real;
                      var puntosarea:lis_ar);

    var
      aux:puntero;
      cont,cont1:integer;


    begin
      aux:=lista;
      cont:=1;
      puntosfijos(aux,xf,yf);
      puntosarea[cont,1]:=xf;
      puntosarea[cont,2]:=yf;
      repeat
        angulos(xf,yf,cont,puntosarea,aux,xam,yam);
        cont:=cont+1;
        puntosarea[cont,1]:=xam;
        puntosarea[cont,2]:=yam;
        xf:=xam;
        yf:=yam;
      until
         (xf=puntosarea[cont-1,1]) and (yf=puntosarea[cont-1,2]);
         puntosarea[cont,1]:=puntosarea[1,1];
         puntosarea[cont,2]:=puntosarea[1,2];
         ptomax:=cont;
         writeln(' Los puntos para determinar el area son:');
```

```
      for cont1:=1 to cont do
        begin
          write('  x=',puntosarea[cont1,1]:2:2);
          writeln('   y=',puntosarea[cont1,2]:2:2);
        end;
      xw:=readkey;

end;


   procedure busca1;
      var
        aux1:puntero;
        ptt,ca:integer;
        pre:boolean;


      begin
        for ca:=1 to 20 do
          begin
           integra[ca,1]:=0;
           integra[ca,2]:=0;
          end;
        aux1:=sal_ins;
        ptt:=1;
        integra[1,1]:=0;
        while aux1<>nil do
           begin
             pre:=false;
             clrscr;
             for ca:=1 to ptt  do
               if (aux1^.nump[1]=integra[ca,1]) or (aux1^.nump[1]=ix) then
                  pre:=true;
             if pre=false then
               begin
                integra[ptt,1]:=aux1^.nump[1];
                ptt:=ptt+1;
               end;
             pre:=false;
             for ca:=1 to ptt  do
               if (aux1^.nump[2]=integra[ca,1]) or (aux1^.nump[2]=ix) then
                  pre:=true;
             if pre=false then
               begin
```

```
                    integra[ptt,1]:=aux1^.nump[2];
                    ptt:=ptt+1;
                   end;
                 pre:=false;
                 for ca:=1 to ptt  do
                   if (aux1^.nump[3]=integra[ca,1]) or (aux1^.nump[3]=ix) then
                      pre:=true;
                 if pre=false then
                   begin
                    integra[ptt,1]:=aux1^.nump[3];
                    ptt:=ptt+1;
                   end;
               aux1:=aux1^.enlace;
          end;

          pt1:=ptt-1;
     end;


procedure carga;

  var
   ca:integer;

  begin
    clrscr;
    writeln('  DAME LOS:');
    for ca:=1 to pt1 do
      begin
       write('P',ca,'.-');
       readln(datos[integra[ca,1],3]);
      end;
  end;

procedure calculaarea(ptomax,ptox:integer; puntosarea:lis_ar;
                      var integra:arre);

   var
     decont:integer;
     sum1,sum2,sum3:real;

   begin
     sum1:=0;
     for decont:=1 to ptomax-1 do
```

```
        begin
          sum2:=puntosarea[decont,1]*puntosarea[decont+1,2];
          sum3:=puntosarea[decont+1,1]*puntosarea[decont,2];
          sum1:=sum1+((sum2-sum3)/2);
        end;
      datos[integra[ptox,1],4]:=sum1;
      write(' El area para el punto:',integra[ptox,1]);
      writeln('  A=',sum1:2:2);
     end;

procedure interpola;

  var
   sumat,sum1,sum2:real;


  begin
    busca1;
    carga;
    writeln(' PUNTOS QUE INTERVIENEN PARA EL PUNTO ',ix,':');
    for ptox:=1 to pt1 do
      writeln('  ',integra[ptox,1]);
    xw:=readkey;
    sal_lis:=nil;
    for ptox:=1 to pt1 do
      begin
        numero:=integra[ptox,1];
        crealista(sal_cor,sal_ins,numero,sal_lis);
       { escribe1(sal_lis);}
       writeln('PARA EL PUNTO ',NUMERO,':');
        det_arreglo(sal_lis,xf,yf,puntosarea);
        calculaarea(ptomax,ptox,puntosarea,integra);
        borra1a(sal_lis);
      end;

    sumat:=0;
    for ptox:=1 to pt1 do
      begin
        sum1:=datos[integra[ptox,1],4];
        sumat:=sumat+sum1;
      end;
    for ptox:=1 to pt1 do
      begin
        sum1:=datos[integra[ptox,1],4]/sumat;
```

```
          datos[ix,3]:=sum1*datos[integra[ptox,1],3]+datos[ix,3];
        end;


  end;



 begin
   {inic;}
   sal_ins:=nil;
   for ix:=np+1 to pi do
     begin
      xpi:=datos[ix,1];
      ypi:=datos[ix,2];
      analizapto;
      writeln(' LOS CIRCULOS QUE SE CREAN CON EL PUNTO SON ',ix,' SON:');
      escribe1(sal_ins);
      INTERPOLA;
      WRITELN('*****************************************************');
      writeln(' EL RESULTADO DE LA INTERPOLACION PARA EL PUNTO INSERTADO:');
      writeln('     X:',xpi:2:2,'      Y:',ypi:2:2);
      writeln('     RESULTDO:',datos[ix,3]);
      delay(3000);
     end;
end;

  begin
    opcion;
    recorrido;
    escribe1(sal_cor);
    graba;
    sal_ins:=nil;
    escribe2;
    xw:=readkey;
    grafica(datos);
    escptos;
    circunferenciasnvas;
  end.
```