La Columna de Matemática Computacional

Sección a cargo de

Tomás Recio

La Columna

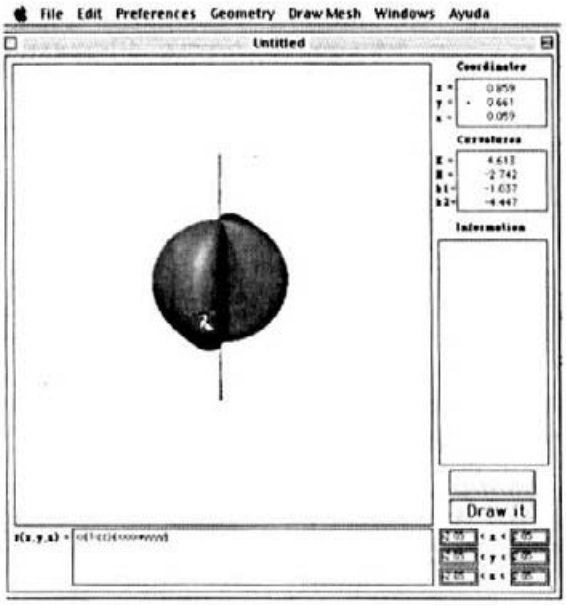
El objetivo de esta columna es presentar de manera sucinta, en cada uno de los números de La Gaceta, alguna cuestión matemática en la que los cálculos, en un sentido muy amplio, tengan un papel destacado. Para cumplir este objetivo el editor de la columna (sin otros méritos que su interés y sin otros recursos que su mejor voluntad) quisiera contar con la colaboración de los lectores, a los que anima a remitirle (a la dirección que se indica al pie de página[1] *) los trabajos y sugerencias que consideren oportunos.*

En este Número ...

Para este número de *La Gaceta* hemos solicitado la colaboración del profesor de la Universidad de Valencia, Angel Montesinos Amilibia, que nos ha enviado una breve introducción a su programa **Superficies**. Las líneas siguientes son un simple "aperitivo" visual al artículo del profesor Montesinos, sin ninguna pretensión de explorar todas las posibilidades de su programa; en particular debe observarse que el tipo de superficie de las figuras que incluímos es sólo unc de los dos posibles.

Para empezar supongamos, por ejemplo, que deseamos representar la superficie implícita $x^2(1-z^2)-(x^4+y^4)=0$. Nótese que esta superficie singular real tiene un "mango" (el eje z) cuya existencia puede pasar inadvertida para otros programas gráficos (compárese con el resultado de aplicar, a la misma ecuación, la instrucción *implicitplot3d* de Maple V-R2). Abrimos una ventana en *Superficies* y tecleamos $x^2(1-z^2)-(x^4+y^4)$ (de esta forma o en alguna variante como $xx(1-zz)-(xxxx+yyyy)$) en el recuadro correspondiente –véase figura:

[1]Tomás Recio. Departamento de Matemáticas. Facultad de Ciencias. Universidad de Cantabria. 39071 Santander. recio@matesco.unican.es

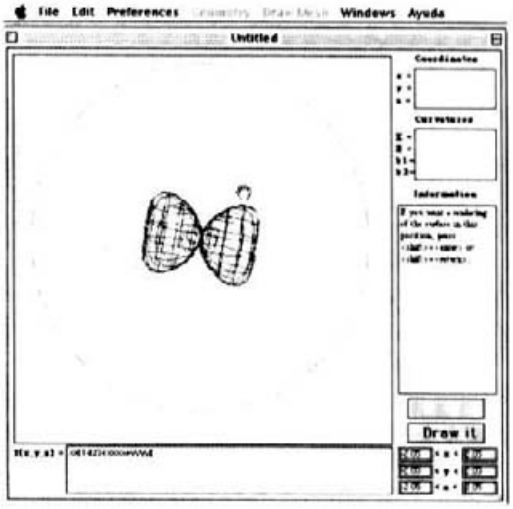


Aparece, al cabo de algunos segundos, una vista de la superficie (incluyendo el eje z) en una gama de 256 colores (desgraciadamente, estos no se aprecian, por razones de economía, en el formato de *La Gaceta*). Además una pajarita –que hace, en este programa, el papel del cursor controlado por el ratón– señala con el pico un punto cuyas coordenadas y curvaturas se visualizan en los cuadros de datos, a la derecha de la imagen. Si elegimos la opción *líneas de curvatura* en uno de los menús del programa, el pico de la pajarita traza las correspondientes al punto dado, como se observa en esta figura. Moviendo la pajarita sobre la superficie se actualizan, instantáneamente, los datos correspondientes.

Pero hay otras muchas opciones. Por ejemplo, podemos trazar geodésicas, indicando, mediante un movimiento del ratón, el origen, la dirección y velocidad inicial, como se aprecia en la figura siguiente:

En este rápido repaso visual al programa, dibujemos la misma superficie en otra posición (hemos elegido un punto casi en el infinito del eje z, es decir,

viendo la superficie desde arriba). Para ello se representa la superficie en modo *wireframe*, una de las opciones del meú, y se rota sobre la pantalla. Al final se puede, de nuevo, representar coloreada en la nueva posición elegida, como indica el cuadro de información que aparece a la derecha de la ventana.



Para terminar, creo necesario advertir al lector de un problema que sucede trabajando con este programa: un problema sobre el que no encontrará ninguna referencia en el artículo del profesor Montesinos. Y es lo difícil que resulta darse cuenta del enorme trabajo que esconde **Superficies**, programado "desde cero" a pesar del enorme bagaje de matemáticas y de cálculos matemáticos optimizados involucrados. Gracias, Ángel, por poner esta hermosa herramienta a nuestra disposición.

Superficies

por

Ángel Montesinos Amilibia

INTRODUCCIÓN

En la enseñanza de la Geometría Diferencial Clásica se tropieza el profesor con el obstáculo de la visualización de curvas y superficies. Todos sabemos que la visión espacial es algo que se va educando lentamente, y que resulta siempre costoso establecer relaciones intuitivas entre formas y fórmulas. El problema es agudo cuando se trata de curvas alabeadas, pero se convierte en formidable al pasar al estudio de las superficies. Y si dentro del campo de las superficies se trata de representar conceptos como líneas de curvatura o geodésicas, la dificultad suele superar las posibilidades de dibujo a mano alzada de casi cualquier profesor.

Por eso, desde hace ocho años, vengo trabajando en desarrollar un programa de ordenador que responda a esa necesidad didáctica, independientemente de que pueda también utilizarse en otros campos más profesionales.

En 1989 desarrollé un programa, *Siluetas*, que dibuja superficies dadas de forma explícita, es decir, mediante una función $z = f(x, y)$. El programa dibuja la superficie de la manera más económica, es decir delineando sólo su contorno y sus límites (o sea, la silueta de la superficie tal como se la dibujaría con plumilla).

Con la popularización de ordenadores más potentes, he visto recientemente la posibilidad de remodelar el programa para incluir superficies mucho más generales. El programa, que se llama ahora *Superficies* y voy a describir a continuación, dibuja superficies en $\mathbb{R}^3$ definidas en forma implícita y en forma paramétrica, y también la geometría de un rectángulo de $\mathbb{R}^2$ dotado de una métrica pseudo-riemanniana. El programa pretende ser una herramienta didáctica, lo que no quita que se pueda usar para otras cosas, desde el diseño de un jarrón hasta el estudio de una singularidad.

DESCRIPCIÓN DEL PROGRAMA

Para obtener el dibujo de una superficie, el usuario escribe en la ventana las ecuaciones que definen la superficie y el dominio o el rango en que desea representarla. Entonces, *Superficies* la dibuja en una gama continua de colores verdes por una de sus caras, y en una gama de colores dorados

por la otra. Se supone que hay un foco de luz que arroja sombras y produce reflejos, aunque la textura de la superficie es un poco mate para evitar reflejos demasiado duros.

Una vez dibujada, se puede interaccionar con ella para obtener información geométrica. Es decir, según se mueve el ratón, *Superficies* presenta en tiempo real las coordenadas y las curvaturas en el punto de la superficie sobre el que está el ratón en ese instante. Según la opción elegida del menú, se pueden dibujar las líneas y direcciones asintóticas y de curvatura, las geodésicas, bolas geodésicas, aplicación exponencial, líneas de curvatura geodésica constante y (con un poco de suerte) una geodésica entre dos puntos fijados por el usuario, aunque no se garantiza que minimice la distancia. También se puede pedir un dibujo "en alambre" y moverlo en tiempo real. Esto permite colocar la superficie en la posición deseada para obtener una nueva vista más conveniente.

Se puede cambiar la geometría del espacio $\mathbb{R}^3$ ambiente eligiendo "Minkowski Space" en el menú. Desde ese momento, *Superficies* asume que $\mathbb{R}^3$ tiene la métrica $dx \otimes dx + dy \otimes dy - dz \otimes dz$, o en términos físicos, que $\mathbb{R}^3$ es un espacio-tiempo en que la coordenada z representa el tiempo. En tal caso, se puede pedir al programa que dibuje los "rayos de luz", es decir las líneas sobre la superficie cuya tangente tiene cuadrado nulo.

La superficie se puede definir de dos maneras, implícita o paramétrica. En el primer caso, se proporciona al programa una función en $\mathbb{R}^3$, $f(x, y, z)$, y se fija un paralelepípedo en $\mathbb{R}^3$ de lados paralelos a los ejes en el cual *Superficies* ha de buscar los puntos en que f se anula y dibujarlos.

En forma paramétrica, se debe escribir en la ventana las tres funciones $x(u, v), y(u, v), z(u, v)$ que definen la aplicación $\mathbf{x} : U \subset \mathbb{R}^2 \to \mathbb{R}^3$, y fijar U como un rectángulo de lados paralelos a los ejes en el plano u, v.

Cuando se desea visualizar la geometría de un rectángulo en $\mathbb{R}^2$ dotado de una métrica riemanniana o pseudo-riemanniana, se escriben en la ventana las funciones $E(x, y), F(x, y), G(x, y)$, y se fija el rectángulo U, que tendrá lados paralelos a los ejes. Entonces, *Superficies* considera que en U hay una métrica

$$g = E\,dx \otimes dx + F(dx \otimes dy + dy \otimes dx) + G\,dy \otimes dy,$$

y colorea U de acuerdo con la curvatura de cada punto, de modo que se puedan distinguir fácilmente los puntos de curvatura positiva y negativa, y los puntos en que g es riemanniana de aquellos en que es pseudo-riemanniana. Como en el caso de las superficies, el programa suministra en tiempo real las coordenadas y la curvatura del punto señalado por el ratón, y se le puede pedir que dibuje los rayos de luz, las geodésicas, etc. En este caso, claro está, no se dispone de las líneas asintóticas o de curvatura, que corresponden a la geometría extrínseca. En cambio, se puede visualizar la "esfera unidad" en cada punto, y también el transporte paralelo.

Se puede abrir varias ventanas simultáneamente, guardar las superficies ya dibujadas como documentos que se abren al picar en ellos dos veces, guardar los dibujos en formato PICT, imprimirlos, etc.

Requerimientos:

Se necesita un PowerMac con un mínimo de 4.5 Mb de RAM libres, y un monitor con al menos 256 colores o 256 tonos de gris. El programa sólo corre en una profundidad de color de 8 bits (256 colores o grises), de modo que si al comenzar se encuentra con una profundidad diferente, la cambia a 8 bits. Si el Mac tiene tiene una frecuencia menor que 100 MHz o carece de una caché de nivel 2 (L2 cache), entonces *Superficies* tarda un minuto o dos en dibujar una superficie implícita corriente, pero entonces se le puede dejar trabajando en el transfondo.

Nota. *Superficies* junto con su manual en inglés y algunos ejemplos se puede obtener mediante ftp en el servidor

<ftp://topologia.geomet.uv.es/pub/montesin/Superficies_folder>.

ALGUNAS PECULIARIDADES INFORMÁTICAS

Una de las cosas que más satisfacción producen al programador es conseguir la "transparencia" informática, es decir que las ficciones que el programa presenta al usuario las tome éste de forma inconsciente como si fueran la cosa más natural del mundo. Esto llevó a uno de los mejores programadores que conozco, el australiano Peter N. Lewis, a decir que el imponente esfuerzo de programar algo de modo que funcione a la primera con toda suavidad se paga con una total falta de reconocimiento, porque es precisamente esa suavidad la que conduce a pensar que debió ser muy fácil programarlo.

A mí, la verdad, me gustaría que quienes usen *Superficies* disfruten de él sin preocuparse de los problemas informáticos que se han tenido que resolver, pero ya que el editor de esta columna me pide alguna información sobre el asunto, describiré dos o tres curiosidades.

La primera es el problema del "señalar": se ha dibujado ya una superficie en la ventana, y el ratón está en un punto de la ventana. En el caso implícitas, ¿cuáles son las coordenadas del punto de la superficie donde parece estar el ratón?; en el caso de paramétricas, ¿cuáles son los valores de los parámetros u, v que definen el punto de la superficie donde está el ratón? *Superficies* necesita resolver este problema para dos cosas: 1) Poder suministrar en tiempo real las coordenadas y las curvaturas del punto de la superficie donde está el ratón. 2) Servir como condición inicial para dibujar las líneas de curvatura, rayos de luz, etc. Además, el éxito consiste en encon-

trar ese punto en un tiempo brevísimo, digamos menos que media décima de segundo, de modo que puedan aparecer en pantalla en tiempo real las coordenadas y curvaturas (que habrán de calcularse una vez conocidas las coordenadas o los parámetros).

En ambos casos la solución se obtiene empleando la inmensa fuerza bruta del ordenador en ejecutar adaptaciones de algoritmos que fueron inventados hace ya siglos. En el caso de implícitas, sea $\mathbf{m} \in \mathbb{R}^3$ el punto desde el cual se está observando la superficie; la ventana del ordenador se supone que es un plano interpuesto entre la superficie y $\mathbf{m}$, de modo que la posición del ratón es un punto $\mathbf{r}$ de ese plano; éstos son los datos que podemos suponer conocidos. Lo que se trata es de calcular el primer cero de la función f en la recta de ecuación $t \mapsto \mathbf{m} + t(\mathbf{r} - \mathbf{m})$ dentro del paralelepípedo P en que se ha dibujado la superficie $f = 0$; así pues, lo primero es encontrar los dos valores $t_1 < t_2$ en que la recta en cuestión entra y sale del paralelepípedo, si los hay. Esto es trivial, pero hay que hacerlo muy rápidamente y entonces no es tan trivial y requiere su correspondiente truco, que no describiré. A continuación se emplea una adaptación del método de Newton-Raphson combinado con un método de subdivisión para hallar el primer cero de la función $h(t) = f(\mathbf{m} + t(\mathbf{r} - \mathbf{m}))$ en el intervalo $[t_1, t_2]$. Téngase en cuenta que *Superficies* es capaz de dibujar correctamente superficies implícitas singulares en cuyos puntos la función f no cambia de signo, sino que simplemente se anula. Por ejemplo, si $f = x^2 + y^2 + z^2 - 1$, tendremos una esfera. Si ponemos en cambio $f = (x^2 + y^2 + z^2 - 1)^2$, entonces f no cambia de signo, pero $f = 0$ sigue representando la misma esfera. Un método basado simplemente en los cambios de signo de f fracasaría aquí estrepitosamente, mientras que *Superficies* pasa fácilmente la prueba.

En el caso de superficies paramétricas, el problema del "señalar" es más difícil. En efecto, ahora tenemos que resolver no una sola ecuación $h(t) = 0$ como antes, sino las tres ecuaciones simultáneas contenidas en las tres componentes de la ecuación vectorial:

$$\mathbf{m} + t(\mathbf{r} - \mathbf{m}) = \mathbf{x}(u, v),$$

para las incógnitas t, u, v, todas ellas en intervalos conocidos, y seleccionar la solución con menor t. El primer paso consiste en reducir a dos el número de incógnitas. Para ello se considera la función "cuadrado de la distancia de un punto a la recta $\mathbf{m} + t(\mathbf{r} - \mathbf{m})$", que es fácil de expresar y vamos a denotar d_r^2. Tendremos entonces que buscar los ceros de la función $W(u, v) = (d_r^2 \circ \mathbf{x})(u, v)$ en el rectángulo U. Se supone que mientras se dibujaba la superficie se ha construido el llamado "z-buffer", que acopia, para cada pixel de la ventana, la distancia a $\mathbf{m}$ del punto de la superficie que, al dibujarla, va a parar a ese pixel. Así, una vez encontrado un cero de W, por ejemplo (u_0, v_0), se calcula la distancia de $\mathbf{x}(u_0, v_0)$ a $\mathbf{m}$, y el pixel en que se proyecta el punto $\mathbf{x}(u_0, v_0)$. Si esa distancia es casi igual a

la dada por el "z-buffer"para ese pixel, se acepta $\mathbf{x}(u_0, v_0)$ como solución, y se termina el proceso.

Para encontrar los ceros de W se divide U en una red de pequeños rectángulos. Sea p el centro de uno de ellos. Se calculan por diferencias finitas las derivadas parciales primeras y segundas de W en p, se simula que W tiene la forma $au^2 + 2buv + cv^2$ en las proximidades de p (lo que será aproximadamente cierto si p es cercano a un cero de W y se supone que ese cero es genérico), y se determina en qué punto se anularía esa función cuadrática, suponiendo que en p esa función tuviera como derivadas parciales primeras y segundas las calculadas. Si el punto así calculado cae fuera del pequeño rectángulo, se rechaza, con la idea de que se encontrará una mejor aproximación al llegar a ese rectángulo (si es que no se ha encontrado ya). Si no, se repite el proceso de aproximación al cero mientras se vea que converge; si en cierta iteración deja de converger, se rechaza; si converge y se alcanza una suficiente aproximación a $W = 0$, se da por bueno. La convergencia se entiende en el plano u, v. Se examinan sucesivamente de este modo todos los pequeños rectángulos, a menos que se encuentre antes la solución buscada.

El proceso no es tan rápido como para implícitas, pero visualmente no se aprecia la diferencia.

La segunda curiosidad se refiere al realismo de los dibujos, y la necesidad de hacer algo en ese sentido me la hizo ver mi compañero de Departamento, Prof. Francisco Carreras. Cuando se pide al programa que dibuje una línea sobre la superficie (geodésica o de otro tipo), *Superficies* elige un color para pintarla, que para geodésicas es el morado. Pero no se limita el programa a calcular la línea y a dibujar sólo su parte visible; es preciso que la "pintura" de la línea responda también a la "iluminación" de la superficie, o sea que imite un reflejo del foco de luz o el paso por una zona de sombra. Por ello, antes de dibujar uno de los pequeños segmentos que forman la línea, *Superficies* comprueba cómo es la iluminación de la zona donde se va a pintar el segmento y elige el color de la línea entre una gama de colores morados; color morado claro si el segmento está fuerte iluminado y refleja luz hacia el observador, o color morado oscuro si el segmento está en una zona más oscura. Así se consigue que las propias líneas contribuyan a dar realismo a la superficie.

Y la tercera es para que las superficies paramétricas no envidien a las implícitas. Cuando el programa dibuja curvas en una superficie implícita (líneas de curvatura, por ejemplo), lo que hace es resolver numéricamente la ecuación diferencial definida por las direccciones principales en cada punto del cubo por la función f. La propia definición de esas direcciones las hace tangentes a las superficies f = constante, de modo que, si esas curvas empiezan en un punto de la superficie, continuarán sobre la superficie. Sólo hay que preocuparse de evitar que se alcance un punto en que no haya

esas direcciones (punto umbilical) o que esté fuera del cubo. Por eso, si la superficie es compacta el programa continua pintando indefinidamente las curvas mientras el usuario no lo pare, o se pare él solo por alguna de las razones apuntadas.

Para superficies dadas en paramétricas, lo que hace *Superficies* es resolver la ecuación diferencial definida en el espacio de parámetros u, v por las funciones $x(u,v), y(u,v), z(u,v)$. Como antes, debe preocuparse de que la curva no se salga del rectángulo u, v fijado por el usuario. Pero ¿qué pasa si esas ecuaciones paramétricas quieren representar una superficie compacta, digamos un elipsoide? Tendríamos U definido por ejemplo como $0 < u < 2\pi,\ 0 < v < \pi$. En cuanto la curva llegara a $u = 0$ ó a $u = 2\pi$, el programa debería pararla, de modo que desde el punto de vista del usuario, parecería haber una invisible barrera para esas curvas en el meridiano cero. Lo que hace *Superficies* es lo siguiente. Supongamos que los parámetros u_1, v_1 del próximo punto a pintar caen fuera de U, y sea $\mathbf{p_1} = \mathbf{x}(u_1, v_1)$. El programa busca, del modo descrito anteriormente, las intersecciones de la superficie con la recta $\mathbf{m} + t(\mathbf{p_1} - \mathbf{m})$, es decir los valores de los parámetros u, v para los cuales $\mathbf{x}(u,v)$ pertenece a esa recta. Sea $\mathbf{p_2} = \mathbf{x}(u_2, v_2)$ uno de ellos, y sea $\mathbf{p_0}$ el último punto legítimo calculado de la curva, cuyos parámetros se suponen muy cercanos a la frontera de U. Si la distancia entre $\mathbf{p_0}$ y $\mathbf{p_2}$ no es muy pequeña, se rechaza $\mathbf{p_2}$. Si lo es, se reinicia el cálculo de la ecuación diferencial a partir de ese punto, tomando como condición inicial de la velocidad aquella velocidad que proyectada en pantalla mantenga la continuidad con la proyección en pantalla de la velocidad que traía la curva en $\mathbf{p_0}$. De ese modo resulta inapreciable el cambio tanto en posición como en velocidad.