REPRESENTACIÓN 3D DE CURVAS Y SUPERFICIES

Francisco Daniel Valbuena Martínez
frankvalbuenam@gmail.com
Universidad Distrital Francisco José de Caldas
Proyecto Curricular de Matemáticas

Abstract: On this paper is shown a simple way to get a 3D representation of curves and surfaces from the linear algebra. In addition to that, is introduced the typical problem that involved the problem of the 3D representation, that is, the visibility problem. The painter's algorithm is used to solve this problem. All of this, developed by Java programming language and in this manner is shown the challenges that involved the implementation.

1. INTRODUCCIÓN

Cuando observamos una gráfica hecha por computador de una función de una variable real, es claro el proceso que está detrás. En los cursos de cálculo se enseña por medio de una tabulación a dibujar estas funciones. Este mismo proceso no es útil cuando se trata del gráfico de una función de dos variables o en general para el gráfico de una superficie, se recurre entonces a los gráficos hechos por computador y comúnmente se pasa por alto la explicación de cómo el computador logra hacer estos gráficos.

Existen muchos métodos que describen cómo hacer este proceso, y las soluciones para este problema varían tanto en su complejidad como en sus resultados. Aquí estamos interesados en lograr representar gráficos de superficies y curvas en 3D, mediante el uso del algebra lineal y haciéndolo de la forma más sencilla que nos sea posible.

Se mostrará como con una sencilla proyección ortogonal se logra encontrar las coordenadas en el plano de un punto del espacio. Esta proyección es una proyección planar paralela.

Usando el lenguaje de programación JAVA se observará como actúa esta proyección y el "problema de visibilidad" que surge cuando se representan superficies mediante esta proyección. Para seguir el enfoque de sencillez de este trabajo se implementa el algoritmo más sencillo que existe para solucionar el "problema de visibilidad", este es, el algoritmo del pintor.

2. REPRESENTACIÓN 3D DE UN PUNTO

Tomaremos a $\mathbf{R}^3$ y $\mathbf{R}^2$ como espacios vectoriales (**R** su campo de escalares). $\mathbf{R}^3$ representa el espacio tridimensional y $\mathbf{R}^2$ representa el plano de proyección. Se desea encontrar una función "adecuada"

$$E: {}^{3}\,3 \quad {}^{2}$$

Tal que cada **v** vector de $\mathbf{R}^3$ sea representado por el vector **π(v)** en $\mathbf{R}^2$. Normalmente se le asigna un punto **O** en $\mathbf{R}^3$ al observador y se toma un plano **P** (el plano **P** reemplaza a $\mathbf{R}^2$) **π** es definida como la función asigna a cada vector **v** el punto de intersección entre dicho plano con la recta que pasa por los puntos **O** y **v**. Las cosas se pueden simplificar mucho si se toman ciertas condiciones:

- El observador **O** se encuentra a una distancia infinita del plano de proyección.
- El plano **P** pasa por el origen.
- Tomamos bases Ortonormales para $\mathbf{R}^3$ y **P**

Un ejemplo sencillo de lo anterior podría ser cuando tomamos $\mathbf{R}^3$ con la base canónica y plano **P** como el plano XY con la base {(1,0,0),(0,1,0)}. Definimos por **π(x,y,z)=(x,y,0).** **π** es una proyección ortogonal de $\mathbf{R}^3$ sobre el subespacio **P**. Se ignora la tercera coordenada y redefinimos **π** por **π(x,y,z)=(x,y).** El problema que surge con esta definición, es que, **π** no nos permite rotar los puntos que están siendo visualizados.

2.1 ROTACIONES

Tomemos el plano XY con la base {(1,0,0),(0,1,0)} y rotémoslo con respecto al eje z un ángulo θ_z y con respecto al eje y un ángulo θ_y.

$$\begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Una base para este nuevo plano **P** es

$$B_P = \{(-\sin\theta_z, \cos\theta_z, 0), (-\cos\theta_z \sin\theta_y, -\sin\theta_y \sin\theta_z, \cos\theta_y)\}$$

Esta es una base ortonormal.

2.2 PROYECCIÓN ORTOGONAL

El último paso para obtener la función **π** es proyectar ortogonalmente todo el espacio $\mathbf{R}^3$ ($\mathbf{R}^3$ con la base canónica) sobre el plano **P** que tiene por base al conjunto

$$B_P = \{(-\sin\theta_z, \cos\theta_z, 0), (-\cos\theta_z \sin\theta_y, -\sin\theta_y \sin\theta_z, \cos\theta_y)\}$$

Es decir dado un vector **v** en $\mathbf{R}^3$ la proyección de este vector sobre el plano P es:

$$(y\cos\theta_z - x\sin\theta_z, z\cos\theta_y - (\sin\theta_y)(x\cos\theta_z + y\sin\theta_z))$$

El anterior vector está expresado en la base $\mathbf{B_P}$ de esta forma tenemos que la función buscada es:

$$e : : {}^{3}\ 3 \quad {}^{2}$$

$$\pi_{\theta_y,\theta_z}(x,y) = (y\cos\theta_z - x\sin\theta_z, z\cos\theta_y - (\sin\theta_y)(x\cos\theta_z + y\sin\theta_z))$$

Es decir para cada par (θ_y, θ_z) se genera una proyección. Esto es útil ya que para rotar la gráfica que se está representando basta variar estos dos parámetros.

3. REPRESENTACIÓN 3D DE CURVAS

Con la función **π** se puede representar curvas en 3D. Ahora graficar curvas en $\mathbf{R}^3$ es tan fácil como lo era graficar curvas en $\mathbf{R}^2$, graficando puntos muy cercanos de la grafica, claro está, con la ayuda de un computador.

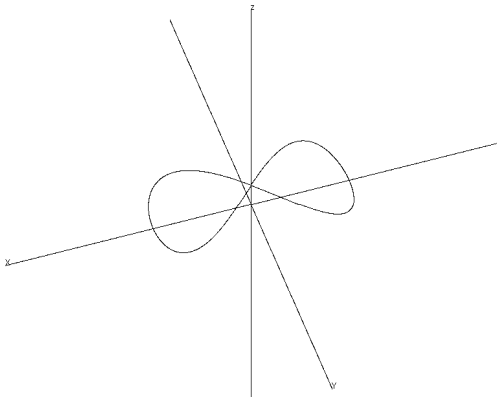

Ilustración 1: Curva hecha en Java usando la función π

Con el gráfico de las curvas podemos tener una aproximación a la gráfica de una superficie, mediante las curvas de nivel como lo muestra la siguiente imagen.

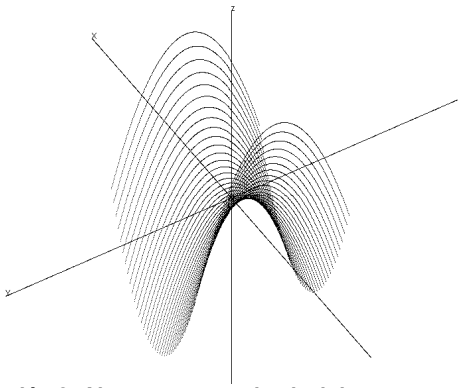

Ilustración 2: Algunas curvas de nivel de una superficie

4. REPRESENTACIÓN 3D DE SUPERFICIES

Teniendo en cuenta que las superficies suaves pueden ser aproximadas por sus planos tangentes, es común encontrar que los programas matemáticos, crean sus gráficas con las imágenes directas de rectángulos en el dominio de la función, los cuales dan la sensación de suavidad sobre la superficie.

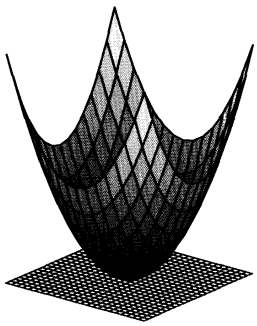

Ilustración 3: fuente John Oprea, Diferential Geometry

En la siguiente gráfica se observa la suavidad aparente de la superficie, así como su solidez.

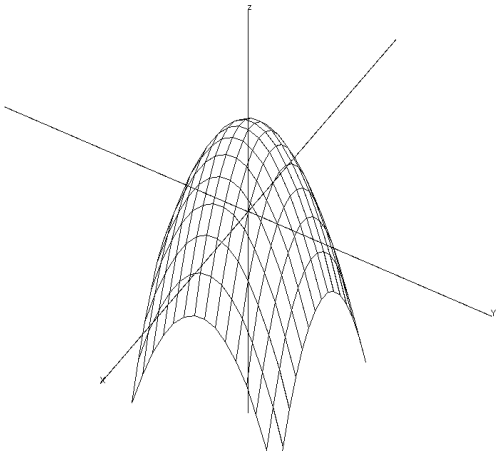

Ilustración 4: Aproximación de una superficie por medio de una malla.

5. EL PROBLEMA DE LA VISIBILIDAD

Después de haber generado la anterior gráfica mediante el programa realizado en Java se rotó el plano de proyección P, rotando el ángulo θ_z y se obtuvo la siguiente imagen:

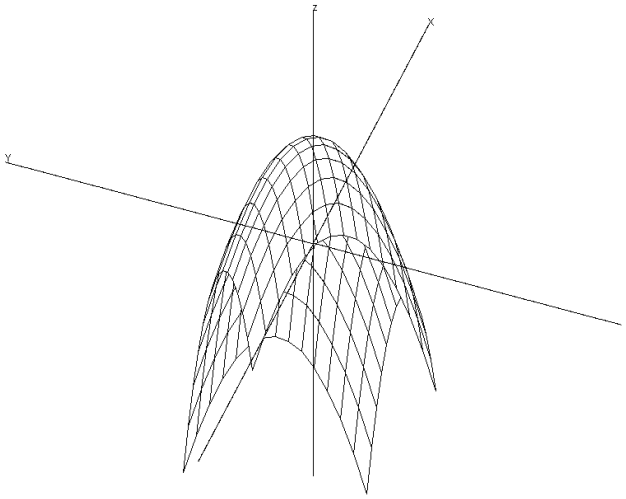

Ilustración 5: Al rotar la imagen la superficie proyectada no luce coherente con la realidad

Notará que la gráfica luce inusual, lo que pasó en realidad fue que la parte que debería verse al fondo paso a estar al frente, de hecho, al correr el programa la gráfica comenzó a girar en sentido contrario al sentido en que giraban los ejes.

5.1 ALGORITMO DEL PINTOR

El problema de visibilidad surge por la incapacidad de determinar la profundidad con la que está un objeto con respecto al plano de proyección **P**, este problema al igual que la representación 3d tiene múltiples soluciones aquí he implementado la que a mi criterio es la más sencilla de comprender.

El algoritmo del pinto hace la analogía de cómo debería el computador dibujar los objetos, en base a como un pintor realiza un cuadro de un paisaje, sin moverse del lugar donde está observando dicho paisaje.

Lo que hace el artista, es pintar primero los objetos más lejanos y después pinta encima de ellos los objetos más cercanos, esta apreciación tan simple soluciona el anterior problema. Entonces lo que hay que hacer es tomar los paralelogramos tomar el centroide y medir a que distancia está del plano de proyección es decir:

$$d(v, P) = \|v - \pi(v)\|$$

Organizar los paralelogramos de tal forma que los que sean dibujados primeros sean aquellos que están a mayor distancia de **P**. (Debido a que el plano pasa por el origen tal vez aplicar una traslación para evitar problemas con la simetría de algunas superficies).

6. COLOR Y RESULTADO FINAL

Finalmente, aprovechando que al implementar el algoritmo del pintor hay que ordenar los paralelogramos, le he asignado una escala de color que indique mediante una degradación a medida que el objeto está más lejano al plano de proyección.

La colorida imagen de abajo corresponde a una superficie generada por la función

$$f(x, y) = \sin^2(x) + \cos^2(y)$$

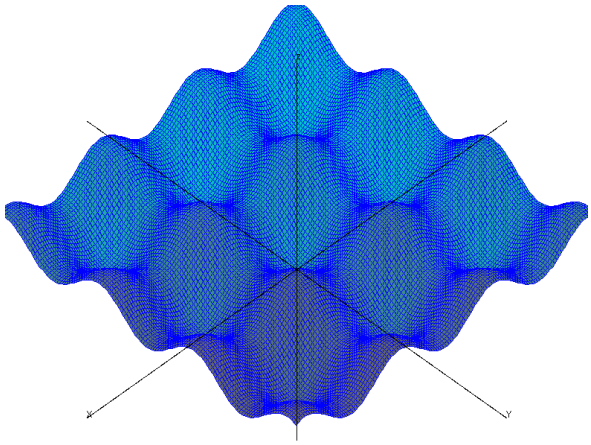

Ilustración 6: Imagen con degradado de color, se han incrementado el numero de paralelogramos y la malla se ha coloreado de azul para que luzca más estética

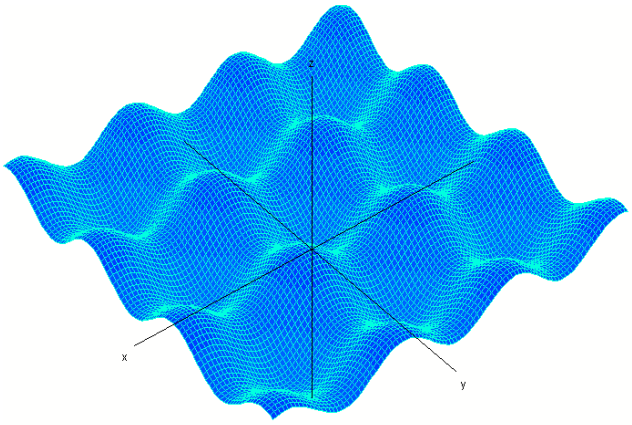

Ilustración 7: Gráfica hecha en Derive 6.0 de la misma función anterior

Bueno, finalmente para medir el resultado obtenido se tiene una gráfica hecha por el paquete matemático Derive, el cual como todos sabemos es un reconocido programa.

7. BIBLIOGRAFIA

John Oprea, (1997), Diferential Geometry and its applications.

Ingrid Carlbom, Joseph Paciorek (1978). *Planar Geometric Projections and Viewing Transformations*. v.10 n.4.

Tom M Apóstol, (1973) Calculus Segunda Edición.

Stanley Grossman, (1996), Algebra Lineal, Mc Graw Hill.

Harvey M. Deitel Y Paul J. Deitel, Java How to Program, 7th Edition