ISBN: 978-628-96613-1-6. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LACCEI2025.1.1.1963

Construction of Surfaces Defined from Planar Curves in Python

Urbina Rubén[1]; Castillo Yheff[2]; Santamaria Ronald[2]; Jaramillo Javier[1]; García Manuel[2]; Silupu Vanessa[2]; Torres Jorge[2]
[1]*Universidad Tecnológica del Perú*, *Perú*, *c22910@utp.edu.pe, c20279@utp.edu.pe*
[2]*Universidad Nacional de Piura, Perú,* 1302018004@alumnos.unp.edu.pe, rsantamarias@egresados.unp.edu.pe, mhgarcias@unp.edu.pe, vsilupuo@unp.edu.pe,1302017029@alumnos.unp.edu.pe

Abstract– ***This article presents several techniques for constructing surfaces defined from planar curves. The first surface studied is the bilinear surface, which requires only four points. By generalizing the bilinear surface, we obtain the so-called Ruled Surfaces, which are frequently used in the aerospace industry. A particular case of ruled surfaces is cylindrical surfaces, which are created by translating a planar curve along a straight trajectory. Another method for generating surfaces involves rotating a planar curve around an axis in space; such surfaces are called surfaces of revolution. Swing surfaces are a generalization of surfaces of revolution, as instead of the curve rotating around a circle, it does so along another arbitrary curve***

.

Keywords-- Planar Curves, Surface Techniques, Surfaces of Revolution, Swing Surfaces, Geometric Modeling

Construcción de superficies definidas a partir de curvas planas en Python

Urbina Rubén[1]; Castillo Yheff[2]; Santamaria Ronald[2]; Jaramillo Javier[1]; García Manuel[2]; Silupu Vanessa[2]; Torres Jorge[2]
[1]*Universidad Tecnológica del Perú*, *Perú*, *c22910@utp.edu.pe, c20279@utp.edu.pe*
[2]*Universidad Nacional de Piura, Perú,* 1302018004@alumnos.unp.edu.pe, rsantamarias@egresados.unp.edu.pe, mhgarcias@unp.edu.pe, vsilupuo@unp.edu.pe,1302017029@alumnos.unp.edu.pe

Resumen– ***Este trabajo tiene como objetivo principal implementar y presentar, desde un enfoque pedagógico, cinco técnicas para la construcción de superficies tridimensionales a partir de curvas planas, utilizando el lenguaje de programación Python. La propuesta busca facilitar la comprensión y visualización de conceptos geométricos en cursos de pregrado, fomentando un aprendizaje activo mediante la manipulación directa de parámetros y el uso de herramientas de código abierto. Las superficies estudiadas incluyen la superficie bilineal, las superficies regladas y cilíndricas, las superficies de revolución y las superficies Swung. Cada una ha sido modelada y graficada en Python, permitiendo su aplicación tanto en el aula como en contextos prácticos de diseño y modelado. Además, se discuten los desafíos y limitaciones asociados a estas construcciones, abriendo nuevas posibilidades para la exploración computacional en el modelado geométrico.***

Palabras clave- Curvas planas, Técnicas de superficies, Superficies de revolución, Superficies swing, Modelado Geométrico.

I. Introducción

El dominio de los conceptos básicos de geometría poligonal es esencial para diseñadores, ya que les permite desarrollar modelos tridimensionales adecuados, mejorar proyectos y fomentar la creatividad [1, 2, 3, 4]. Dentro del modelado geométrico, las superficies definidas a partir de curvas planas representan un enfoque clave en el modelado geométrico, con aplicaciones en diversas industrias como la aeronáutica y la naval. En particular, las superficies regladas, desarrollables y alabeadas representan un campo de estudio esencial en la geometría tridimensional debido a su amplia aplicabilidad en áreas como la arquitectura, el diseño industrial, naval y la ingeniería [5, 6 ,7, 8, 9].

La generación de superficies a partir de curvas planas es un tema ampliamente estudiado en la geometría computacional, con aplicaciones en modelado tridimensional y diseño paramétrico. Diversos enfoques han sido explorados, desde el análisis teórico de superficies de revolución en el espacio euclidiano, hasta la implementación de herramientas interactivas para su construcción y su aplicación en optimización acústica [10, 11, 12]. Su modelación no solo permite comprender su comportamiento geométrico, sino también explorar herramientas computacionales, como Python, para su construcción dinámica y visualización. Esto fomenta un aprendizaje más profundo y práctico, conectando la teoría geométrica con aplicaciones reales en diversos campos.

Este trabajo explora las técnicas para construir superficies bilineales, regladas, cilíndricas, de revolución y Swing, utilizando sus fundamentos matemáticos y aplicaciones prácticas. La implementación se realiza en Python utilizando Jupyter Notebook, permitiendo la construcción y visualización de estas superficies.

El estudio de la construcción de superficies mediante curvas planas en Python no solo se basa en aplicar herramientas computacionales para modelar superficies geométricas, sino que también tiene una gran importancia como herramienta de uso pedagógico en cursos de pregrado y permiten un enfoque practico donde los estudiantes pueden manipular parámetros, lo que refuerza su compresión con propiedades matemáticas. Además, su integración con la programación fomenta un aprendizaje más profundo y práctico, facilitando la exploración interactiva de conceptos matemáticos y su aplicación en diversas áreas [13, 14, 15]. Además, permite a los estudiantes comprender su relevancia en contextos aplicados, como el diseño en ingeniería, arquitectura y gráficos computacionales, mientras se promueve el uso de lenguajes de programación como Python, esenciales en el entorno académico y profesional moderno [16, 17, 18, 19]. A través de métodos como interpolación lineal, rotación y combinaciones de curvas, se generan superficies con potencial para diseño geométrico asistido por computadora y simulaciones en 3D [20, 21, 22]. Además, su implementación proporciona una base sólida para entender el comportamiento geométrico y explorar nuevas posibilidades creativas.

Este trabajo utiliza Python y Jupyter Notebook como herramientas para implementar y visualizar superficies bilineales, regladas, cilíndricas, de revolución y Swing. El desarrollo se enfoca en el uso de un Software libre para resaltar las nociones matemáticas de superficies y aplicaciones prácticas, particularmente en diseño geométrico asistido por computadora y simulaciones tridimensionales.

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general que ha ganado una gran popularidad en los últimos años. Es conocido por su sintaxis sencilla y por su legibilidad, lo que facilita su aprendizaje y su comprensión [23, 24].

II. Construcción de superficies

A continuación, se analizarán los conceptos matemáticos relacionados con los métodos de construcción de superficies a partir de curvas planas, los cuales serán fundamentales para su implementación y visualización en Python.

A. Superficie bilineal

Una superficie bilineal es una superficie generada mediante la interpolación entre cuatro puntos en R^3, definida como el producto tensorial de dos interpolaciones lineales independientes [25]. Un polígono plano es un ejemplo simple de superficie. La superficie bilineal es la superficie no plana más simple, porque queda completamente definida por cuatro puntos [26]; es decir, es una superficie tensorial donde las curvas producto son dos rectas. Se emplea con frecuencia para la interpolación de cuatro valores de una tabla (por ejemplo, en antialiasing de texturas) y es la superficie más sencilla que pasa por cuatro puntos en $\mathbb{R}^3$.

Sean los cuatro puntos distintos P_{00}, P_{01}, P_{10}, P_{11} como se muestra en la Fig. 1. Las curvas fronteras son líneas rectas y son fáciles de calcular.

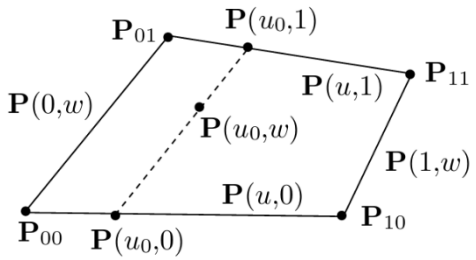

Fig. 1 Superficie Bilineal [26].

Cualquier punto en el interior de la superficie está dado por la interpolación lineal entre fronteras opuestas de un cuadrado unitario como se muestra en la Fig. 2. Esto es,

$$P(u,0) = (P_{10} - P_{00})u + P_{00} \quad (1)$$
$$P(u,1) = (P_{11} - P_{01})u + P_{01} \quad (2)$$

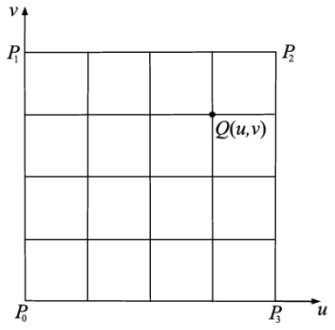

Fig. 2 Interpolación bilineal en el espacio paramétrico

Para realizar la interpolación lineal entre las curvas frontera, primero calculamos los puntos $P(u_0,0)$ y $P(u_0,0)$, y luego los unimos mediante una línea recta $P(u_0,v)$. Los dos puntos son

$$P(u_0,0) = (P_{10} - P_{00})u_0 + P_{00} \quad (3)$$
$$P(u_0,1) = (P_{11} - P_{01})u_0 + P_{01} \quad (4)$$

y la línea recta que los une está dada por

$$\begin{aligned} P(u_0,v) &= \big(P(u_0,1) - P(u_0,0)\big)v - P(u_0,0) \\ &= \big[(P_{11} - P_{01})u_0 + P_{01} - \big((P_{10} - P_{00})u_0 + \\ &\quad P_{00}\big)\big]v + (P_{10} - P_{00})u_0 + P_{00} \end{aligned} \quad (5)$$

Cualquier punto en el interior del cuadrado paramétrico está dado por

$$\begin{aligned} P(u,v) = {} & P_{00}(1-u)(1-v) + P_{01}(1-u)v \\ & + P_{10}u(1-v) + P_{11}uv \end{aligned} \quad (6)$$

En forma matricial

$$P(u,v) = [1-u \quad u]\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}\begin{bmatrix} 1-v \\ v \end{bmatrix} \quad (7)$$

Es necesario que la superficie coincida con los puntos dados. Es fácil verificar que la superficie interpola los puntos dados.

Si los puntos de definición de una superficie bilineal son las diagonales opuestas sobre caras opuestas de un cubo unitario, la superficie resultante es un paraboloide hiperbólico. Es decir, matemáticamente una superficie bilineal es un paraboloide hiperbólico, donde las expresiones paramétricas son lineales respecto a las variables u y v.

B. Superficie Regladas

Una superficie reglada es aquella que puede ser engendrada por el movimiento de una línea recta [27]. Las superficies regladas son frecuentemente usadas en la industria aérea y naval. Por ejemplo, las alas de los aviones son superficies cilíndricas regladas. Una superficie reglada es generada por una línea recta que se mueve a lo largo de un camino con un grado de libertad. Estas superficies son simplemente generalizaciones de la superficie bilineal. Es decir, una superficie reglada se obtiene mediante una interpolación lineal entre dos curvas frontera conocidas asociadas a los lados opuestos de un cuadrado unitario en el espacio paramétrico.

Sean las curvas paramétricas opuestas $P(u,0)$ y $P(u,1)$. La superficie reglada está dada por

$$S(u,v) = P(u,0)(1 - v) + P(u,1)v \quad (8)$$

O de forma matricial esta dado por

$$\begin{aligned} S(u,v) &= [x(u,v) \quad y(u,v) \quad z(u,v)] \\ &= [1-v \quad v]\begin{bmatrix} P(u,0) \\ P(u,1) \end{bmatrix} \end{aligned} \quad (9)$$

Ahora note, que dos de los extremos de la superficie interpolante coincide con las curvas dadas; es decir, $Q(u,0) = P(u,0)$ y $Q(u,1) = P(u,1)$.

C. Superficie Cilíndricas

Las líneas rectas en una superficie reglada se llaman sus generatrices. Si todas las generatrices son paralelas, nuestra superficie reglada es cilíndrica [28]. Recordemos que una superficie reglada queda definida conociendo las curvas fronteras opuestas $P(u,0)$ y $P(u,1)$. De modo que, la superficie reglada está dada por [26]

$$P(u,v) = P(u,0)(1-v) + P(u,1)v \tag{10}$$

Como caso particular de superficies regladas, están las denominadas cilindros generalizados, que se obtienen al desplazar una curva $S(u,0)$ a través del plano siguiendo una trayectoria recta [25].

Sea el vector W que indica la dirección de desplazamiento de la curva $P(u,0)$, y sustituyendo en la ecuación (15), por la curva de traslación $P(u,1) = P(u,0) + W$, se obtiene

$$\begin{aligned} P(u,v) &= (1-v)P(u,0) + v[P(u,0) + W] \\ &= P(u,0) + vW \end{aligned} \tag{11}$$

D. Superficie Revolución

Otro método simple para generar superficies tridimensionales es rotando una curva 3D alrededor de un eje en el espacio [25]. Este tipo de superficies son llamadas superficies de revolución. Las superficies de revolución se obtienen por rotación de la generatriz respecto del eje de giro, la directriz [29]. Sin perder generalidad considere como eje de rotación el eje positivo x y la curva a rotar contenida en el plano xy. Sea la curva generatriz $C(u)$ con ecuación paramétrica.

$$c(u) = [x(u) \quad y(u) \quad 0] \tag{12}$$

Una rotación del ángulo φ alrededor del eje x viene dada por la transformación

$$[T(v)] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(v) & sen(v) \\ 0 & -sen(v) & \cos(v) \end{bmatrix} \tag{13}$$

Con lo cual la superficie es:

$$S(u,v) = C(u)[T(v)] \tag{14}$$

$$\begin{aligned} &= [x(u) \quad y(u) \quad 0]\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(v) & sen(v) \\ 0 & -sen(v) & \cos(v) \end{bmatrix} \\ &= [x(u) \quad y(u)\cos(v) \quad y(u)sen(v)] \end{aligned}$$

E. Superficie Swung

Una generalización de las superficies de revolución es las denominadas superficies Swung (del inglés Swung: balancear, oscilar, abatir) [25]. Esto es, en lugar que la curva $C_1(u)$ gire alrededor de una superficie, considere una curva arbitraria $C_2(v) = [0 \quad c(v) \quad s(v)]$.

Considere el producto de dos curvas, contenido respectivamente en los planos XY y XZ.

$$C_1(u) = [x(y) \quad y(u) \quad 0] \text{ y } C_2(u) = [0 \quad c(v) \quad s(v)] \tag{15}$$

Por analogía con el procedimiento constructivo para las superficies de revolución, se obtiene la superficie

$$\begin{aligned} S(u,v) &= [x(u) \quad y(u) \quad 0]\begin{bmatrix} 1 & 0 & 0 \\ 0 & c(v) & s(v) \\ 0 & -s(v) & c(v) \end{bmatrix} \\ &= [x(u) \quad y(u)c(v) \quad y(u)s(v)] \end{aligned} \tag{16}$$

Donde, ahora la matriz no tiene por qué ser ortogonal, pues en general $c^2(v) + s^2(v) \neq 1$.

III. Implementación de superficies en Python

A continuación, se mostrarán las clases definidas en Python para definir las superficies que se estudiaron.

A. Clase para superficie bilineal

La clase *Bilinearsurface* permite graficar este tipo de superficies, donde lo único que necesita son los puntos P_{00}, P_{01}, P_{10} y P_{11}. A continuación, se muestra cómo funciona esta clase donde *Surfacebilineal_graph* es un atributo de la clase en mención y los parámetros: puntos, líneas y superficie toman valores boolenos, que permitirá controlar los objetos a mostrar en el gráfico.

```
Bilinearsurface().Surfacebilineal_graph(p00,p01,p10,p1
1, puntos=True, lineas=True, superficie=True)
```

Esos son los parámetros principales (obligatorios) del atributo *Surfacebilineal_graph*, pero se le pueden pasar más, los mismos parámetros que usa *go.surface()* de Plotly para graficar superficies. Entre los principales se tiene: *colorscale='viridis'*, *opacity =0.5* y *showscale=False*.

Si consideramos los puntos ubicados en un cubo unitario, como se muestra en la Fig. 3. Sea $P_{00} = [1 \quad 0 \quad 0]$, $P_{01} = [0 \quad 1 \quad 0]$, $P_{10} = [0 \quad 0 \quad 1]$ y $P_{11} = [1 \quad 1 \quad 1]$, donde la interpolación de los puntos genera un paraboloide hiperbólico.

Sustituyendo los puntos dados en (6), tenemos:

$$\begin{aligned} P(u,w) = {} & [1 \quad 0 \quad 0](1-u) + [0 \quad 1 \quad 0](1-u) \\ & + [0 \quad 0 \quad 1]u(1-w) + [1 \quad 1 \quad 1]uw \end{aligned} \tag{17}$$

cuya ecuación es

$$P(u,w) = [1 - w + u(2w-1) \quad w \quad u] \tag{18}$$

Ahora, con la implementación de la clase Bilinearsurface en Python, mostraremos su uso. Para este caso solo necesitamos 4 puntos que son $p_{00}, p_{01}, p_{10}, p_{11}$

```
Bilinearsurface().Surfacebilineal_graph(p00=[1,0,0],p0
1=[0,1,0], p10=[0,0,1], p11=[1,1,1], puntos=True,
lineas=False, superficie=True, colorscale='viridis',
showscale=False)
```

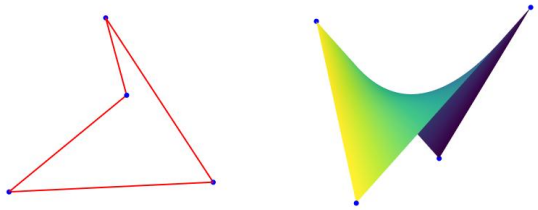

Fig. 3 (Izquierda) Cubo unitario mostrando los puntos a interpolar; (derecha) Superficie bilineal resultante: Paraboloide hiperbólico

B. *Clase para superficie Regladas*

Rulersurface permite graficar superficies regladas utilizando su método *Surfaceruler_graph*. Este requiere dos curvas (curva_1 y curva_2), que deben ser listas de puntos que representan las curvas, fácilmente generadas con la librería numpy. El parámetro j define un rango asociado a la variable v de la superficie, mientras que los parámetros booleanos curvas y superficie permiten controlar la visualización de las curvas y la superficie, respectivamente.

```
Rulersurface().Surfaceruler_graph(curva_1, curva_2,
j=[v_1,v_2], curvas=True, superficie=True)
```

A modo de ejemplo, considere las curvas

$$P(u,0) = [\cos(u) \quad \text{sen}(u) \quad 5] \tag{19}$$
$$P(u,1) = [2\cos(u) \quad 3\,\text{sen}(u) \quad 0] \tag{20}$$

una circunferencia y una elipse, respectivamente. La ecuación de la superficie reglada es

$$S(u,v) = [(1+v)\cos(u) \quad (1+2v)sen(u) \quad 5-5v] \tag{21}$$

que representa a un cono truncado, de base superior la circunferencia y base inferior la elipse, ver Fig. 4.

Ahora, con la implementación de la clase *Rulersurface* en Python, mostraremos su uso. Primero, definiremos los parámetros $curva_1$, $curva_2$ y j que es el rango de la variable v

```
u = np.linspace(0, 2*np.pi, 30)
C_u1 = [np.cos(u), np.sin(u), 5+0*u]
C_u2 = [2*np.cos(u), 3*np.sin(u), 0*u]
Rulersurface().Surfaceruler_graph(curva_1=C_u1,
curva_2=C_u2, j=[0,1], curvas=True, superficie=True,
colorscale='viridis', opacity=0.95, showscale=False)
```

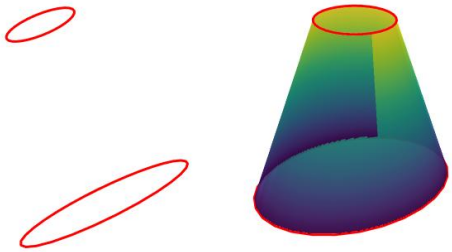

Fig. 4 (Izquierda) curvas fronteras cuyas bases son una elipse y una circunferencia; (derecha) Superficie reglada resultante: cono truncado

C. *Clase para superficies cilíndricas*

Esta clase es similar a la de las superficies regladas, solo que aquí necesita como entrada una curva y el vector dirección W. Los otros parámetros cumplen la misma función como en las superficies regladas.

.

```
Cylindricalsurface().Surfacecylindrical_graph(curva_1,
W=[a, b, c], j=[v_1, v_2], curvas=True,
superficie=True)
```

En la Fig. 5 se muestra una superficie cilíndrica, donde la curva base es una rosa de cuatro pétalos, cuya ecuación paramétrica está dada por

$$P(u,0) = [\cos(2u)\cos(u) \quad \cos(2u)\,sen(u) \quad 0] \tag{22}$$

que descansa sobre el plano XY; siguiendo la dirección del vector $W = [0 \quad 1 \quad 1]$.

se obtiene la curva de traslación

$$\begin{aligned} P(u,1) &= P(u,0) + W \\ &= [\cos(2u)\cos(u) \quad \cos(2u)\,sen(u) \quad 0] \\ &\quad +[0 \quad 1 \quad 1] \\ &= [\cos(2u)\cos(u) \quad \cos(2u)\,sen(u)+1 \quad 1] \end{aligned} \tag{23}$$

De donde

$$P(u,1) = [\cos(2u)\cos(u) \quad \cos(2u)\,sen(u)+1 \quad 1] \tag{24}$$

La superficie cilíndrica generada tiene un ángulo de inclinación de $\varphi = \frac{\pi}{4}$, dado por el vector W. De le ecuación (11), se tiene que le ecuación de la superficie cilíndrica está dada por

$$\begin{aligned} P(u,w) &= [\cos(2u)\cos(u) \quad \cos(2u)\,sen(u) \quad 0] \\ &\quad +w[0 \quad 1 \quad 1] \\ &= [\cos(2u)\cos(u) \quad \cos(2u)\,sen(u)+w \quad w] \end{aligned} \tag{25}$$

Ahora, con la implementación de la clase Cylindricalsurface en Python, mostraremos su uso. Primero,

definiremos los valores: u (parámetro de la curva), $C1$ (curva) y W (vector dirección).

```
u = np.linspace(0, 2*np.pi,50)
C1= [np.cos(2*u)*np.cos(u), np.cos(2*u)*np.sin(u), 0*u]
W = [0,1,1]
Cylindricalsurface().Surfacecylindrical_graph(curva_1=
C1, W=[0,1,1], j=[0,1], curvas=True, superficie=True,
colorscale='viridis', showscale=False)
```

Donde la curva base que es una rosa de cuatro pétalos se ha graficado con 50 puntos y le vector dirección es $w = [0 \quad 1 \quad 1]$.

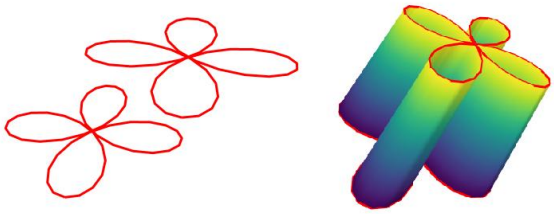

Fig. 5 Superficie con curvas fronteras:
Rosa de cuatro pétalos. $w = [0 \quad 1 \quad 1]$.

De manera similar, podemos utilizar como curva base la ecuación de la espiral [30]. Esta ecuación, expresada en coordenadas polares, tiene la forma $r = \theta^2$. Al convertirla a ecuaciones paramétricas y realizar algunos ajustes para que el parámetro sea $u = [0, 2\pi]$, obtenemos la siguiente ecuación

$$C_1(u) = \left[\left(\frac{u}{4}\right)^2 \cos 4u\,, \left(\frac{u}{4}\right)^2 \sin 4u\right] \tag{26}$$

Entonces de forma práctica ya podemos hacer uso de nuestra clase en Python definiendo sus parámetros

```
u = np.linspace(0, 2*np.pi, 100)
C1 = [(u/4)**2*np.cos(4*u), (u/4)**2*np.sin(4*u), u*0]
W = [0, 0, 2]
Cylindricalsurface().Surfacecylindrical_graph(curva_1=
C1, W=[0,1,2], j=[0,1], curvas=True, superficie=True,
colorscale='viridis', showscale=False)
```

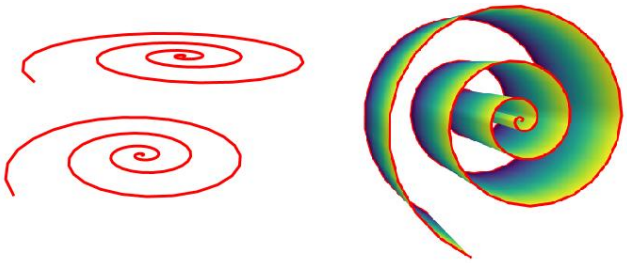

Fig. 6 Superficie generada por curvas de espirales

D. Clase para superficies de revolución

La clase *Revolutionsurface*, a través de su método *Surfacerevolution_graph*, permite graficar superficies de revolución. Solo se requiere la curva generatriz definida en el plano xy. El parámetro $j = [0, 2\pi]$ determina que la curva realice una rotación completa alrededor de la circunferencia. No obstante, este rango es ajustable, para una rotación parcial de media circunferencia, se puede establecer como $j = [0, \pi]$.

```
Revolutionsurface().Surfacerevolution_graph(curv
a_1, j=[0, 2*np.pi], curva=True,
superficie=True)
```

La Fig. 7 muestra dos ejemplos de superficies de revolución. La primera es el resultado de rotar un segmento de la función seno

$$C(u) = [u \quad 1.5 + sen(u) \quad 0] \tag{27}$$

con $u \in \left(0, \frac{3\pi}{2}\right)$ alrededor del eje X. La ecuación de la superficie resultante es

$$P(u,v) = c(u)\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(v) & sen(v) \\ 0 & -sen(v) & \cos(v) \end{bmatrix} = \left[u \quad \frac{(2\sin(u)+3)\cos(v)}{2} \quad \frac{(2\sin(u)+3)\sin(v)}{2}\right] \tag{28}$$

La visualización de esta superficie en Python usando la clase se muestra a continuación

```
u = np.linspace(0, 3*np.pi/2, 30)
C1 = [u, 1.5+np.sin(u), 0*u]
Revolutionsurface().Surfacerevolution_graph(curva_1=C1
, j=[0, 2*np.pi], curva=True, superficie=True,
colorscale='viridis', showscale=False)
```

La segunda superficie que se muestra en la Fig. 7 es el resultado de rotación de la circunferencia centrada en $[0, 1, 0]$ y radio $1/2$ contenido en el plano xy.

$$C(u) = \left[\frac{\cos(u)}{2} \quad 1 + \frac{sen(u)}{2} \quad 0\right] \text{ tal que } u \in (0, 2\pi). \tag{29}$$

La implementación en Python viene dada por

```
u = np.linspace(0, 2*np.pi, 50)
C1 = [(1-2*np.cos(u))*np.sin(u),4+(1-2*np.cos(u))
*np.cos(u),0*u]
Revolutionsurface().Surfacerevolution_graph(curva_1=C1,
j=[0, 2*np.pi], curva=True, superficie=True,
colorscale='viridis', showscale=False)
```

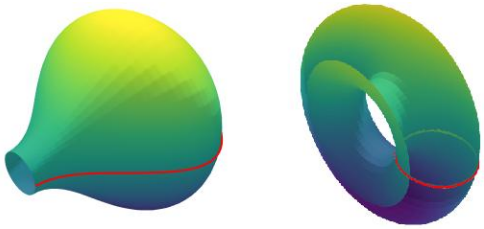

Fig. 7 (Izquierda) Rotación de la función seno.
(Derecha) Rotación de una circunferencia

De manera similar, construiremos una superficie a partir de la ecuación de la cardioide. En coordenadas polares, esta se define por $r^2 = 2r - 2r\cos(\theta)$ [30]. Al convertirla a ecuaciones paramétricas y realizar un pequeño ajuste para trasladarla 4 unidades en el eje y, además de generar un punto doble, obtenemos la siguiente ecuación

$$C_1(u) = [(1 - 2\cos(u))\sin(u), 4 + (1 - 2\cos(u))\cos(u)] \quad (30)$$

Entonces de forma práctica, podemos hacer uso de nuestra clase Revolutionsurface() en Python definiendo los parámetros

```
u = np.linspace(0, 2*np.pi, 50)
C1=[(1-2*np.cos(u))*np.sin(u), 4+(1-
2*np.cos(u))*np.cos(u), 0*u]
Revolutionsurface().Surfacerevolution_graph(curva_1=C1,
j=[0, 2*np.pi], curva=True, superficie=True,
colorscale='viridis', opacity=0.8, showscale=False)
```

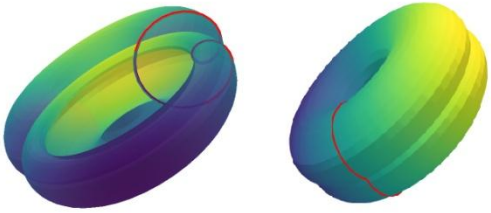

Fig. 8 Superficie de revolución generada por la curva cardioide

E. Clase para superficies swung

Por último, la clase *Swungsurface* incluye el método *Surfaceswung_graph*, que requiere como parámetros dos curvas: *curva_1*, que actúa como la curva generatriz en el plano xy, y *curva_2*, que es la curva directriz en el plano yz.

```
Swungsurface().Surfaceswung_graph(curva_1, curva_2,
curvas=True, superficie=True)
```

Es importante destacar que, todas las clases mencionadas esperan que las curvas se ingresen como listas de puntos, no de forma simbólica. Además, estas clases permiten incorporar parámetros adicionales, compatibles con los utilizados en *go.Surface()* de la librería Plotly, como se detalló previamente en el apartado de superficies bilineales.

La Fig. 8 muestra una superficie Swung. Donde la curva directriz es la función seno de ecuación

$$C_1(u) = [u \quad 1.5 + sen(u) \quad 0] \text{ tal que } u \in \left[0, \frac{3\pi}{2}\right]. \quad (31)$$

que descansa en el plano XY; y la curva generatriz es una astroide que descansa en el plano YZ.

$$C_2(v) = [0 \quad \cos^3(v) \quad sen^3(v)] \text{ tal que } v \in [0, 2\pi] \quad (32)$$

Donde la ecuación de la superficie está dada por

$$S(u,v) = C_1(u)\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos^3(v) & sen^3(v) \\ 0 & -sen^3(v) & \cos^3(v) \end{bmatrix} \quad (33)$$
$$= \begin{bmatrix} u & (1.5 + sen(v))\cos^3(v) & (1.5 + sen(u))sen^3(v) \end{bmatrix}$$

Su implementación en Python es de la siguiente forma

```
u = np.linspace(0, 3*np.pi/2, 50)
v = np.linspace(0, 2*np.pi, 50)
C1 = [u, 1.5+np.sin(u), 0*u]
C2 = [0*u, np.cos(v)**3, np.sin(v)**3]
Swungsurface().Surfaceswung_graph(curva_1=C1,
curva_2=C2, curvas=True, superficie=True)
```

Fig. 9 Superficie generada por una directriz de función seno

De la misma forma utilizaremos en coordenadas polares la ecuación de estrella de tres puntas. Su ecuación es $r = 2 + \cos(3\theta)$ [30]. En ecuaciones paramétricas ajustándolo para reducir su tamaño es

$$C(v) = [(1.5 + \cos(3v))\cos(v), (1.5 + \cos(3v))\sin(v)] \quad (34)$$

Para la ecuación de estrella de tres puntas tenemos

```
u = np.linspace(0, np.pi, 50)
v = np.linspace(0, 2*np.pi, 100)
C1 = [u, 1.5+np.sin(u), 0*u]
C2 = [0*v, (1.5+np.cos(3*v))*np.cos(v),
(1.5+np.cos(3*v))*np.sin(v)]
```

```
Swungsurface().Surfaceswung_graph(curva_1=C1,
curva_2=C2, curvas=True, superficie=True,
colorscale='viridis', opacity=0.95, showscale=False)
```

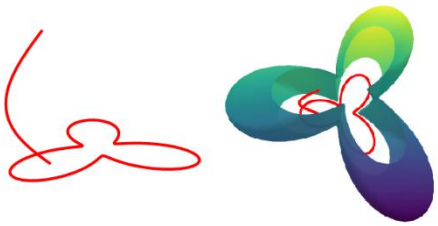

Fig. 10 Superficie Swing a través de la ecuación estrella de tres puntas

IV. DISCUSIONES

En este estudio, se propone un modelo de integración entre conceptos matemáticos avanzados y herramientas computacionales accesibles, promoviendo así un aprendizaje interactivo y práctico. La implementación en Python facilita la visualización y manipulación de superficies, lo que representa una ventaja significativa tanto en el ámbito educativo como en el profesional. Además, no solo facilita la enseñanza de conceptos de programación, sino que también mejora la preparación profesional de los estudiantes al integrarse con tecnologías innovadoras. Además, su enfoque práctico, como experimentos en reconocimiento de imágenes, aprendizaje automático y análisis de datos, demuestra la aplicabilidad de Python en diversos campos, fomentando el aprendizaje activo y la investigación científica. La combinación de teoría y práctica, junto con la integración de la programación en un entorno experimental, convierte a Python en una herramienta educativa inclusiva, versátil y fundamental para la formación en ingeniería y tecnología [31, 32 ,33].

Uno de los principales aportes de esta investigación es su capacidad para conectar la teoría geométrica con herramientas computaciones como Python, ya que estos conceptos matemáticos en particular se pueden conectar con aplicaciones reales, particularmente en disciplinas como la ingeniería, la arquitectura y el diseño industrial. Las superficies regladas y cilíndricas, por ejemplo, tienen un papel fundamental en el diseño de estructuras arquitectónicas y componentes aeroespaciales, lo que evidencia la aplicabilidad y relevancia del estudio en contextos prácticos.

Además, este estudio trabaja con el concepto de superficies Swung, una generalización de las superficies de revolución, en las que la curva generatriz no solo rota alrededor de un eje fijo, sino que también se desplaza a lo largo de otra curva arbitraria. Esta ampliación de las posibilidades geométricas abre nuevas perspectivas en el modelado tridimensional y fomenta la creatividad en el diseño computacional.

El objetivo de este estudio es integrar la teoría geométrica, en particular la construcción de diversos tipos de superficies, con herramientas de programación modernas como Python. Esto no solo permite una comprensión más profunda de las propiedades geométricas, sino que también incentiva a los estudiantes y profesionales a familiarizarse con un lenguaje de programación versátil y ampliamente utilizado en la academia y la industria [34, 35]. Más allá de la enseñanza, este trabajo abre nuevas posibilidades para la investigación y el desarrollo en áreas como la ingeniería, la arquitectura y la educación.

Sin embargo, el estudio puede expandirse aún más mediante una comparación detallada con otras herramientas de modelado geométrico, como MATLAB o software especializado en el diseño geométrico asistido por computadora (CAGD).

V. CONCLUSIONES

Este estudio implementó cinco técnicas clásicas de generación de superficies a partir de curvas planas, utilizando Python como herramienta pedagógica para facilitar su visualización y comprensión en el contexto de la educación superior. La propuesta demostró que el uso de lenguajes de programación accesibles permite no solo representar superficies complejas, sino también explorar sus propiedades geométricas de forma interactiva.

Los algoritmos desarrollados permitieron la construcción dinámica de superficies bilineales, regladas, cilíndricas, de revolución y Swung, brindando a los estudiantes una experiencia práctica en el modelado geométrico. Esta integración entre geometría y programación constituye un recurso eficaz para fortalecer el aprendizaje y promover nuevas aplicaciones en diseño, arquitectura e ingeniería.

Los resultados obtenidos evidencian que la integración de herramientas computacionales con el modelado geométrico permite no solo una representación eficiente de superficies, sino también un análisis detallado en conceptos matemáticos. Este trabajo ha contribuido al campo del modelado geométrico asistido por computadora al proporcionar metodologías y herramientas que facilitan la generación y análisis de superficies, abriendo nuevas oportunidades para futuras investigaciones en optimización geométrica, diseño generativo y aplicaciones en ingeniería avanzada.

Como proyección futura, se sugiere ampliar estas metodologías hacia superficies más complejas y evaluar su aplicación comparativa con otras plataformas de modelado geométrico asistido por computadora.

REFERENCIAS

[1] A. Suárez, P. Villanueva, A. Flores and D. Hernández, "Interpretación y manejo de la geometría poligonal para la creación de superficies y modelos digitales 3D," Diseño en Síntesis, vol. 58, pp. 44-59, Otoño 2017.

[2] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, "Polygon Mesh Processing", 2010. [Online]. Available: https://www.amazon.com/Polygon-Mesh-Processing-Mario-Botsch/dp/1568814267.

[3] M. Fedele, A. Quarteroni, and A. Quarteroni, "Polygonal surface processing and mesh generation tools for the numerical simulation of the cardiac function," International Journal for Numerical Methods in Biomedical Engineering, vol. 37, no. 4, Jan. 2021, doi: 10.1002/CNM.3435.

[4] G. Yngve and G. Turk, "Robust creation of implicit surfaces from polygonal meshes," IEEE Transactions on Visualization and Computer Graphics, vol. 8, no. 4, pp. 346–359, Oct. 2002, doi: 10.1109/TVCG.2002.1044520. Available: https://www.cs.jhu.edu/~misha/Fall05/Papers/yngve02.pdf.

[5] L. Arias, M. Zorro and M. Patarroyo, "Superficies regladas desarrollables y alabeadas, un estudio histórico y su modelación". Universidad Pedagógica y Tecnológica de Colombia (UPTC), Colombia, 2015.

[6] V. N. Ivanov, В. Н. Иванов, V. I. Rynkovskaya, and М. И. Рынковская, "Application of circular surfaces to the architecture of the buildings, structures and products," no. 3, pp. 111–119, Dec. 2015.

[7] S. Flöry and H. Pottmann, "Ruled Surfaces for Rationalization and Design in Architecture," pp. 103–109, Jan. 2010. [Online]. Available: https://research.kaust.edu.sa/en/publications/ruled-surfaces-for-rationalization-and-design-in-architecture.

[8] V. Nikolić, O. Nikolić, J. Tamburic, S. Spasić Đorđević, and S. Janković, "Higher-Order Ruled Surfaces and their Possible Use in Architectural Design," Nexus Network Journal, vol. 21, no. 1, pp. 129–147, Jan. 2019, doi: 10.1007/S00004-018-00425-0.

[9] D. Brander and J. Gravesen, "Surfaces Foliated by Planar Geodesics: A Model for Curved Wood Design," pp. 487–490, Jan. 2017.

[10] M. Soliman, H. N. Abd-Ellah, and M. Basuney, "Some New Results of Revolution Surfaces in Euclidean 3-Space E3," Assiut University Journal of Multidisciplinary Scientific Research, Sep. 2023, doi: 10.21608/aunj.2023.214982.1055.

[11] J. L. Vergara Ibarra, "Revolution solids with GeoGebra: Sólidos de revolución con GeoGebra," vol. 22, no. 1, Jul. 2021, doi: 10.18845/RDMEI.V22I1.5735.

[12] B. Peters, "Parametric Acoustic Surfaces," Jan. 2009. [Online]. Available: http://papers.cumincad.org/data/works/att/acadia09_174.content.pdf.

[13] J. Qiu, A. Moeller, J. Zhen, and N. Adelstein, "Teaching Undergraduate Students Heterogeneous Electrocatalytic Water Oxidation Using Cyclic Voltammetry and Python Simulation," Meeting abstracts, vol. MA2024-01, no. 54, p. 2901, Aug. 2024, doi: 10.1149/ma2024-01542901mtgabs.

[14] G. V. Ferreira, W. D. Pizzol Maria, and A. R. de Melo, "Introdução à Geometria Fractal no Ensino Médio Técnico: Uma Abordagem com Programação Python," Apr. 2021, doi: 10.14210/COTB.V12.P543-546.

[15] C. M. Păcurar, "Jointly Learning Python Programming and Analytic Geometry," World Academy of Science, Engineering and Technology, International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering, vol. 11, no. 2, pp. 62–66, Jan. 2017, doi: 10.5281/ZENODO.1128977.

[16] A. M. Khan, W. S. Alaloul, M. A. Musarat, and M. H. F. Khan, "Python: An Automation Tool for Unlocking Innovation and Efficiency in the AEC Sector," pp. 89–94, Oct. 2023, doi: 10.1109/icdabi60145.2023.10629478.

[17] R. Labib, "Is computer programming beneficial to architects and architecture students for complex modeling and informed performative design decisions," Oct. 2017. [Online]. Available: https://escholarship.org/uc/item/0203m6dj.

[18] L. Nan, W. Chengxi, C. Zhang, T. Xu, and J. Fenglei, "A multi-dimensional and hierarchical comprehensive experimental platform based on Python," vol. 1, no. 1, Sep. 2024, doi: 10.59782/sidr.v1i1.51.

[19] N. A. Otahanov, "Teaching the Python programming language in higher education institutions," Informatika i obrazovanie, Nov. 2023, doi: 10.32517/0234-0453-2023-38-5-78-86.

[20] J. Gulanová, M. Vereš, and L. Gulan, "Surface interpolation and procedure used in the generative engineering design of surface-based automotive components," International Journal of Vehicle Design, vol. 77, no. 4, p. 211, Jan. 2018, doi: 10.1504/IJVD.2018.10021449.

[21] H. Bidnichenko, "Features of geometric surfaces and methods of their computer modeling," Прикладная геометрия и инженерная графика, no. 102, pp. 13–26, Jun. 2022, doi: 10.32347/0131-579x.2022.102.13-26.

[22] N. Vukašinović and J. Duhovnik, "Creating Complex CAD Models with Freeform Surfaces," Springer, Cham, 2019, pp. 81–109. doi: 10.1007/978-3-030-02399-7_4.

[23] R. Duque, "Python para todos". Disponible en: http://mundogeek.net/tutorial-python/, 2011.

[24] L. Cuervo, N. Cuervo, and J. Cuervo, "Iniciando a programar con Python: Guía básica de programación". Editorial de la Universidad Pedagógica y Tecnológica de Colombia - UPTC, 2024.

[25] J. Cordero and J. Cortés, "Curvas y Superficies para Modelado Geométrico", México: Alfaomega, 2002.

[26] D. Salomon, "Curves and Surfaces for Computer Graphics, Springer, 2006.

[27] C. Lehmann, "Geometría Analítica", México: Editorial Limusa, 1989.

[28] G. Farin, "Curves and Surfaces for CAGD (Computer Aided Graphics and Design)", San Diego: Academic Press, 2001.

[29] J. Gutiérrez, "Superficies como curvas en movimiento", España, UCrea, pp 11-32, 2016.

[30] J. Stewart, "Cálculo de varias variables: Trascendentes tempranas", 8.ª ed., Cengage Learning, 2016.

[31] N. A. Otahanov, "Teaching the Python programming language in higher education institutions," Informatika i obrazovanie, Nov. 2023, doi: 10.32517/0234-0453-2023-38-5-78-86.

[32] G. J. S, S. Doshi, A. K. Alex, Y. U, and F. D. J.L, "A Game-Based Approach to Teach Basic Python Programming," Journal of Engineering Education Transformations, Oct. 2024, doi: 10.16920/jeet/2024/v38i2/24186.

[33] L. Nan, W. Chengxi, C. Zhang, T. Xu, and J. Fenglei, "A multi-dimensional and hierarchical comprehensive experimental platform based on Python," vol. 1, no. 1, Sep. 2024, doi: 10.59782/sidr.v1i1.51.

[34] S. Koch, T. Schneider, F. Williams, and D. Panozzo, "Geometric computing with python," International Conference on Computer Graphics and Interactive Techniques, Jul. 2019, doi: 10.1145/3305366.3328067.

[35] N. Miolane et al., "Introduction to Geometric Learning in Python with Geomstats," pp. 48–57, Jul. 2020, doi: 10.25080/MAJORA-342D178E-007.