

Coons Surfaces Expressed as B-Spline Surfaces in Python

Urbina Rubén¹; Castillo Yheff²; Santamaria Ronald²; Sosa Henry¹; Jiménez Judith²
Velasquez Felicita²; Sandoval Hugo³

¹Universidad Tecnológica del Perú, Perú, c22910@utp.edu.pe, c20582@utp.edu.pe

²Universidad Nacional de Piura, Perú, 1302018004@alumnos.unp.edu.pe, rsantamarias@egresados.unp.edu.pe, jjimenezv@unp.edu.pe, fvelasquezf@unp.edu.pe, 1302017029@alumnos.unp.edu.pe

³Universidad Nacional de la Frontera, Perú, hsandoval@unf.edu.pe

Abstract– This study analyzes the representation of Coons surfaces using B-Spline, implementing algorithms in Python for their visualization and manipulation. B-Spline basis functions and the Oslo Algorithm are employed to enhance the precision, continuity, and adaptability of surfaces without altering their geometry. The conversion of Coons surfaces to B-Spline optimizes flexibility in Computer-Aided Geometric Design (CAGD). These surfaces are crucial as they allow for smooth surface interpolation from boundary curves, facilitating applications in structural simulation, digital animation, and industrial design. The results demonstrate that the B-Spline-based formulation improves control over surface interpolation without requiring additional structures. Furthermore, these geometric modeling techniques highlight Python as an accessible tool for research and education, as understanding these mathematical aspects enhances the representation of curves and surfaces in computational geometric modeling.

Keywords-- B-Spline Basis Functions, Oslo Algorithm, Coons Surfaces, Ruled Surfaces, Geometric Modeling.

Superficies de Coons expresadas como Superficies B-Spline en Python

Urbina Rubén¹; Castillo Yheff²; Santamaria Ronald²; Sosa Henry¹; Jiménez Judith²; Velasquez Felicita²; Sandoval Hugo³

¹Universidad Tecnológica del Perú, Perú, c22910@utp.edu.pe, c20582@utp.edu.pe

²Universidad Nacional de Piura, Perú, 1302018004@alumnos.unp.edu.pe, rsantamarias@egresados.unp.edu.pe, jjimenezv@unp.edu.pe, fvelasquezf@unp.edu.pe, 1302017029@alumnos.unp.edu.pe

³Universidad Nacional de la Frontera, Perú, hsandoval@unf.edu.pe

Resumen– Este estudio analiza la representación de superficies de Coons mediante B-Spline, implementando algoritmos en Python para su visualización y manipulación. Se emplean funciones base B-Spline y el Algoritmo de Oslo para mejorar la precisión, continuidad y adaptabilidad de las superficies sin alterar su geometría. La conversión de superficies de Coons a B-Spline optimiza la flexibilidad en el diseño geométrico asistido por computadora (CAGD), estas superficies son importantes porque permiten interpolar superficies suavemente a partir de curvas límite, facilitando su aplicación como simulación estructural, animación digital y diseño industrial. Los resultados evidencian que la formulación basada en B-Spline mejora el control sobre la interpolación de superficies sin la necesidad de estructuras adicionales. Además, estas técnicas del modelado geométrico asistido por computa, se destaca su uso de Python como herramienta accesible para la investigación y la educación, ya que entender estos aspectos matemáticos mejora la forma de representar curvas y superficies en el modelado geométrico computacional.

Palabras clave- Funciones Base B-Spline, Algoritmo de Oslo, Superficies Coons, Superficies regladas, Modelado Geométrico.

I. INTRODUCCIÓN

Los astilleros fueron pioneros en la mecanización del modelado matemático de superficies, desarrollando técnicas como las superficies lofted. En la década de 1960, la industria automotriz y aeronáutica adoptó computadoras para automatizar el diseño, sustituyendo modelos en arcilla por métodos digitales [1]. Figuras clave como Pierre Bézier (Renault), Paul de Casteljaeu (Citroën), Steven Coons (Ford), y William Gordon y Richard Riesenfeld (General Motors) impulsaron avances en el modelado de curvas y superficies, junto con James Ferguson en la aeronáutica [2,3]. En 1974, este campo se consolidó como Diseño Geométrico Asistido por Computadora (CAGD). Con el desarrollo del hardware en los años 80, las técnicas de CAGD permitieron en los 90 la creación de efectos visuales en películas como Jurassic Park, evolucionando hasta la animación digital completa en filmes como Toy Story, Buscando a Nemo y Shrek [4].

Las superficies B-spline son esenciales en el modelado geométrico y el procesamiento de imágenes, ya que facilitan representaciones fluidas y continuas de geometrías complejas. En CAD, se utilizan para reconstruir superficies complejas como cascos de barcos mediante técnicas de reparametrización [5] y en el procesamiento de imágenes facilitan la aproximación y el filtrado de datos, mejorando la calidad y eficiencia

computacional [6]. Entre sus ventajas, destacan su continuidad sin restricciones, lo que las hace ideales para diseños de superficies de forma libre [7], y recientes avances en algoritmos garantizan una correcta topología en intersecciones de superficies, abordando desafíos en sistemas CAD [8].

Las superficies de Coons, cuando se expresan como superficies B-Spline, combinan las ventajas de ambas formulaciones para lograr representaciones suaves y flexibles en aplicaciones de CAGD. Existen trabajos de mucha aplicabilidad como la reconstrucción de superficies con curvas B-spline, que se basa en ajustar puntos de datos límite y proyectar puntos internos sobre una superficie de referencia, optimizando el ajuste mediante mínimos cuadrados iterativos, lo que permite modelar superficies de Coons bilineales para datos dispersos [9]. En otro estudio en el contexto de elementos finitos, la interpolación de Coons permite derivar funciones de forma isoparamétricas utilizando estrías B-spline en los bordes de los parches, mejorando la precisión en el análisis estructural [10]. Además, lograr la continuidad G^2 en superficies B-spline es esencial para garantizar transiciones suaves en modelos con mallas cerradas y geometrías complejas, lo que se logra estimando derivadas tangenciales y asegurando compatibilidad de condiciones [11]. Sin embargo, a pesar de sus ventajas en flexibilidad y suavidad, estas superficies pueden generar una mayor complejidad computacional, especialmente cuando se requiere mantener altos niveles de continuidad en geometrías intrincadas.

Boehm [12, 13, 14] introdujo la inserción de nodos en curvas B-spline, lo que permitió una mejor adaptabilidad en el refinamiento de superficies sin alterar su estructura general. Este concepto es crucial en la conversión de superficies de Coons a B-spline, ya que permite la manipulación de mallas de control para ajustarse a geometrías definidas por condiciones de contorno específicas. En este mismo sentido, Boehm [15, 16, 17] abordó la generación de curvas y superficies con continuidad de curvatura, un aspecto esencial para la transición entre distintos tipos de superficies en CAD, asegurando suavidad y precisión en la interpolación de datos geométricos.

Las técnicas de cálculo con Spline [18] y los métodos eficientes de evaluación de Spline [19] han proporcionado herramientas fundamentales para el modelado de superficies B-spline que pueden expresar, con gran fidelidad, la flexibilidad de una superficie de Coons original. Dado que las superficies

de Coons utilizan interpolaciones basadas en curvas de contorno, su representación mediante B-Spline requiere una parametrización cuidadosa para garantizar la correcta transición entre los parches que la conforman. Esta necesidad ha sido abordada en estudios sobre interpolación y refinamiento de superficies en el ámbito del CAGD [20], quienes realizaron una revisión exhaustiva sobre métodos de curvas y superficies aplicados en este contexto.

Otro aspecto importante es la representación de superficies triangulares mediante Spline [17], donde se exploran nuevas estructuras de interpolación para superficies con topologías más complejas. Esta línea de investigación se vincula directamente con la conversión de superficies de Coons en estructuras B-Spline, ya que una de las principales limitaciones de las superficies de Coons es su dependencia de la interpolación en cuadriláteros, mientras que las superficies B-Spline pueden extenderse a estructuras más generales, como mallas triangulares o combinaciones de parches NURBS.

Existen estudios donde se aplican los métodos numéricos aplicados a Spline [21] que ha facilitado la optimización de algoritmos computacionales para la manipulación de superficies en entornos de modelado digital. Estos métodos han mejorado significativamente la precisión y eficiencia de las representaciones geométricas, permitiendo que las superficies de Coons sean expresadas y manipuladas como superficies B-Spline sin perder las propiedades geométricas esenciales que las caracterizan. Así, la convergencia de estos estudios ha permitido la evolución del modelado geométrico, ofreciendo representaciones más flexibles y computacionalmente eficientes en el diseño asistido por computadora.

Este trabajo explora la construcción de Superficies Coons expresadas como Superficies B-Spline en Python, integrando fundamentos matemáticos con aplicaciones prácticas. Implementado en Python mediante Jupyter Notebook, permite modelar y visualizar estas superficies. Más allá del modelado geométrico, también refuerza la comprensión matemática a los estudiantes [22, 23, 24].

Además, su uso en educación fomenta un aprendizaje práctico e interdisciplinario, mostrando su relevancia en áreas como ingeniería, arquitectura y gráficos computacionales, promoviendo Python como herramienta esencial en el ámbito académico y profesional [25, 26, 27, 28].

El desarrollo enfatiza el uso de software libre para destacar las nociones matemáticas y facilitar su implementación en entornos educativos y profesionales. Python, reconocido por su sintaxis clara y versatilidad, ha ganado gran popularidad en los últimos años, siendo una herramienta accesible para modelado geométrico y simulaciones avanzadas [29, 30].

Este trabajo explora la construcción de superficies de Coons representadas como superficies B-Spline, integrando fundamentos matemáticos con aplicaciones prácticas mediante Python en Jupyter Notebook. Esta implementación permite modelar y visualizar superficies de forma intuitiva e interactiva, lo que a su vez refuerza la comprensión matemática de los estudiantes [18, 19, 20].

La elección de Python sobre lenguajes como C o C++ responde a su sintaxis clara, que reduce la curva de aprendizaje y permite centrarse en los conceptos geométricos sin la complejidad de la gestión de memoria o sintaxis verbosa. Aunque C/C++ ofrecen mayor velocidad, Python acelera el desarrollo y resulta más accesible para estudiantes y docentes.

Su robusto ecosistema de bibliotecas científicas como NumPy, SciPy y Matplotlib facilita tanto las operaciones numéricas como la visualización gráfica. Reproducir estas funcionalidades en C/C++ implicaría un esfuerzo considerable. Además, el uso de Jupyter Notebooks promueve un enfoque interactivo y visual del aprendizaje, difícil de lograr en entornos tradicionales [21, 22, 23, 24].

El trabajo también destaca el valor del software libre para facilitar la implementación educativa y profesional. Python, ampliamente adoptado por su versatilidad, es una herramienta clave para el modelado geométrico y simulaciones avanzadas en disciplinas como la ingeniería, la arquitectura y los gráficos computacionales [25, 26].

II. SUPERFICIES DE COONS A TRAVÉS DE B-SPLINE

Los siguientes conceptos matemáticos se han recopiado de [31, 32, 33, 34, 35, 36].

A. Funciones bases B-Spline

Un B-spline, también conocido como base-spline, es una curva que interpola entre un conjunto de puntos de control. Además, incorporamos ciertos valores numéricos, conocidos como *nudos* (knots), para mejorar nuestro control sobre la forma de la curva [37].

Así, las funciones B-spline exhiben un nivel más alto de sofisticación matemática en comparación con otros tipos de spline. Las funciones B-spline se utilizan comúnmente para resolver problemas de valores iniciales y de contorno debido a sus fuertes propiedades geométricas y su capacidad para proporcionar soluciones precisas en lugares fuera de la malla en el dominio espacial [33].

Sea $T = \{t_0, t_1, t_2, \dots, t_{r-1}, t_r\}$ una sucesión no decreciente de números reales llamada nodos. T es llamado vector nodo. La i -ésima función base $N_{i,k}(t)$ de orden k (o equivalentemente, grado $k - 1$) se define mediante la relación recursiva

$$N_{i,0} = \begin{cases} 1 & t_i \leq t \leq t_{i+1} \\ 0 & \text{En otro caso} \end{cases} \quad (1)$$

y

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) \quad (2)$$

Para $k > 1$. En la definición anterior, se aplica la convención $\frac{0}{0} = 0$.

B. Curvas B-Spline

Las curvas B-Spline, su flexibilidad permite transiciones suaves y un control preciso de la forma, lo que las hace ideales para diversas aplicaciones. Se aplican ampliamente en diversas áreas debido a su capacidad para modelar formas complejas con precisión y suavidad. En el diseño asistido por computadora, se utilizan en arquitectura e ingeniería para generar superficies detalladas y estructuras complejas de manera eficiente [38]. En la animación, permiten la creación de movimientos fluidos y naturales, facilitando el control preciso de trayectorias y mejorando la calidad visual en producciones digitales [38]. Además, en el procesamiento de imágenes, las B-spline son esenciales para representar características curvilíneas con alta fidelidad, optimizando la vectorización de imágenes rasterizadas y permitiendo su uso en aplicaciones como el diseño de logotipos y la digitalización de planos [39].

Una curva B-Spline $C(t)$ de orden k se define como

$$C(t) = \sum_{i=0}^n P_i N_{i,k}(t) \quad (3)$$

donde los P_i son puntos de control en 3D y los $\{N_{i,k}(t)\}$ son las funciones base B-spline normalizadas de orden k definidas en la ecuación 2. Como se sabe, para cualquier curva B-spline de orden k , el número de puntos de control $(n + 1)$ y el número de nodos $(r + 1)$, están relacionados mediante la expresión $r = n + k$.

C. Superficie B-Spline

Las superficies B-spline son una herramienta fundamental en el diseño asistido por ordenador (CAD) y el procesamiento de imágenes, ya que ofrecen flexibilidad y precisión a la hora de modelar geometrías complejas. Estas superficies se definen mediante puntos de control y funciones bases, lo que permite obtener representaciones fluidas y continuas.

Una Superficie B-spline producto tensor se define como

$$S(u, v) = \sum_{i=0}^n \sum_{j=0}^m P_{i,j} N_{i,k}(u) N_{j,l}(v) \quad (4)$$

donde los $P_{i,j}$ forman la malla o red de control en 3D, y $N_{i,k}(u)$, $N_{j,l}(v)$ son las funciones base B-spline normalizadas de orden k y l en las direcciones u y v , respectivamente, definidas sobre los vectores nodo $U = \{u_i\}_{i=0}^r$ y $V = \{v_i\}_{i=0}^s$.

D. Superficies de Coons

Este tipo de superficie se basa en la labor pionera de Steven Anson Coons, en la década de los 60' del siglo pasado, cuando trabajaba para la Compañía Automovilística Ford. Comenzamos con el estudio de la superficie lineal de Coons, que es una generalización de las superficies lofted. Este tipo de superficie, también conocida como parche de Coons, se define por cuatro curvas frontera de la forma $S(u, 0)$, $S(u, 1)$, $S(0, v)$, $S(1, v)$, como se muestra en la Fig. 1.

Naturalmente, las curvas fronteras se cortan en las esquinas, por lo que estos puntos son conocidos implícitamente.

Steven Coons buscó una expresión $S(u, v)$ de una superficie, tal que satisfaga las siguientes condiciones

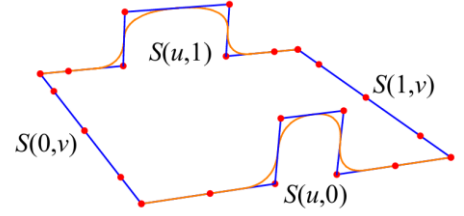


Fig. 1 Curvas frontera de una superficie

1. Que sea simétrica en las variables u y v
2. Que sea una interpolación de $S(u, 0)$, $S(u, 1)$ en la dirección u y de $S(0, v)$, $S(1, v)$ en la otra dirección v .

Coons encontró una solución tan simple en sólo dos pasos. El primer paso, es construir dos superficies regladas con dos pares de curvas frontera, opuestas entre sí: $\{S(u, 0), S(u, 1)\}$ y $\{S(0, v), S(1, v)\}$. Tal que

$$\begin{aligned} P_1 S &= (1 - u)S(0, v) + uS(1, v) \\ &= [1 - u \quad u] \begin{bmatrix} S(0, v) \\ S(1, v) \end{bmatrix} \end{aligned} \quad (5)$$

y

$$\begin{aligned} P_2 S &= S(u, 0)(1 - v) + S(u, 1)v \\ &= [S(u, 0) \quad S(u, 1)] \begin{bmatrix} 1 - v \\ v \end{bmatrix} \end{aligned} \quad (6)$$

El segundo paso es intentar crear provisionalmente la superficie $S(u, v)$ sumando $P_1 S$ y $P_2 S$

$$\begin{aligned} S(u, v) &= P_1 S + P_2 \\ &= (1 - u)S(0, v) + uS(1, v) \\ &\quad + S(u, 0)(1 - v) + S(u, 1)v \end{aligned} \quad (7)$$

Es evidente que ésta no es la expresión que estamos buscando, ya que no converge en las curvas fronteras; es decir, no las interpola.

Por ejemplos, para $u = 0$ queremos que $S(u, v)$ converja a la curva frontera $S(0, v)$. La suma anterior, sin embargo, converge a

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

Por lo tanto, tenemos que restar la expresión $S(0, 0)(1 - v) + S(0, 1)v$. Del mismo modo, para $u = 1$, la suma converge a

$$S(1, v) = S(1, v) + S(1, 0)(1 - v) + S(1, 1)v \quad (9)$$

Por lo tanto, tenemos que restar la expresión $S(1, 0)(1 - v) + S(1, 1)v$. Además, para $v = 0$, se tiene que restar

$S(0,0)(1 - u) + S(0,1)u$ y para $u = 1$, tenemos que restar la expresión $S(0,1)(1 - u) + S(1,1)u$.

Note que $S(0,0), S(0,1), S(1,0), S(1,1)$ son los puntos de intersección de las curvas. Para una mejor lectura se pueden denotar como: $S_{00}, S_{01}, S_{10}, S_{11}$.

En la actualidad, este tipo de superficie se conoce como Superficie Lineal de Coons; se representa como

$$S(u, v) = P_1S + P_2S - P_1P_2S \quad (10)$$

La expresión anterior se puede expresar como suma Boleana

$$\begin{aligned} PS &= P_1S \oplus P_2S \\ &= P_1S + P_2S - P_1P_2S \end{aligned} \quad (11)$$

Donde

$$\begin{aligned} P_1P_2S &= S_{00}(1 - u)(1 - v) + S_{01}(1 - u)v \\ &\quad + S_{10}u(1 - v) + S_{11}uv \\ &= [1 - u \quad u] \begin{bmatrix} S_{00} & S_{01} \\ S_{10} & S_{11} \end{bmatrix} \begin{bmatrix} 1 - v \\ v \end{bmatrix} \end{aligned} \quad (12)$$

Note que P_1S y P_2S son superficies lofted, mientras que P_1P_2S es una superficie bilineal. Finalmente, la expresión es:

$$\begin{aligned} S(u, v) &= P_1S + P_2S - P_1P_2S \\ &= [1 - u \quad u] \begin{bmatrix} S(0, v) \\ S(1, v) \end{bmatrix} + [S(u, 0) \quad S(u, 1)] \begin{bmatrix} 1 - v \\ v \end{bmatrix} \\ &\quad - [1 - u \quad u] \begin{bmatrix} S_{00} & S_{01} \\ S_{10} & S_{11} \end{bmatrix} \begin{bmatrix} 1 - v \\ v \end{bmatrix} \end{aligned} \quad (13)$$

Simplificando y reordenando la ecuación, se obtiene que $S(u, v)$ es

$$[1 - u \quad u \quad 1] \begin{bmatrix} -S_{00} & -S_{01} & S(0, v) \\ -S_{10} & -S_{11} & S(1, v) \\ S(u, 0) & S(u, 1) & (0, 0, 0) \end{bmatrix} \begin{bmatrix} 1 - v \\ v \\ 1 \end{bmatrix} \quad (14)$$

La ecuación 13 es más útil que la ecuación 14, ya que muestra como la superficie se define en términos de los dos pares baricéntricos $[1 - u \quad u]$ y $[1 - v \quad v]$. Las cuales son las funciones base de Bernstein de la superficie lineal de Coons. Es posible tomar otros pares de funciones baricéntricas $[f_1(u) \quad f_2(u)]$ y $[g_1(v) \quad g_2(v)]$, que pueden servir como funciones base; con las que se puede construir superficies de Coons más generales. Estas funciones base deben satisfacer las siguientes condiciones:

$$\begin{aligned} f_1(0) &= 1, f_1(1) = 0, f_2(0) = 0 \\ f_2(1) &= 1, f_1(u) + f_2(u) = 1 \\ g_1(0) &= 1, g_1(1) = 0, g_2(0) = 0, \\ g_2(1) &= 1, g_1(v) + g_2(v) = 1 \end{aligned} \quad (15)$$

E. Inserción de Nodos

La inserción de nodos es un proceso mediante el cual se agrega un nuevo nodo a la secuencia de nodos de una B-Spline sin alterar la forma de la curva o superficie. Esta técnica es esencial para operaciones como el refinamiento local y la subdivisión adaptativa, ya que permite aumentar la flexibilidad del modelo sin modificar la geometría original. El procedimiento implica la computación de los puntos de control, lo que permite una transición suave en la estructura de la curva.

Considere un Spline

$$C(u) = \sum_i c_i N_i^n(u) \quad (16)$$

sobre la secuencia de nodos a_i . Sea $\hat{a}_i$ un refinamiento de a_i , esto es, la secuencia de nodos $\hat{a}_i$ contiene a a_i , como una subsecuencia. Ver Fig. 2.

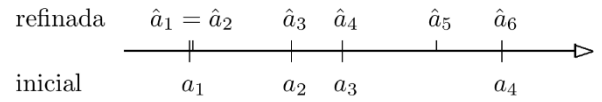


Fig. 2 Refinamiento de una secuencia de nodos.

Sea $\hat{N}_j(u)$ el B-Spline de grado n sobre $\hat{a}_j, \dots, \hat{a}_{j+n+1}$, entonces $s(u)$ se puede describir como

$$C(u) = \sum_j \hat{c}_j \hat{N}_j(u) \quad (17)$$

La mejor manera de calcular los $\hat{c}_j$ es a través de la inserción repetida de nodos simples, esto es: se inserta un nodo en cada paso [13]. Supongamos entonces que $\hat{a}_j$ se obtiene a partir de a_i por inserción de un solo nodo $\hat{a}$. Desplazando el índice de ser necesario, podemos suponer que $\hat{a} = \hat{a}_{n+1}$ y $a_n \leq \hat{a} < a_{n+1}$. Entonces se obtiene

$$\hat{c}_j = \begin{cases} c_j & \text{para } j \leq 0 \\ c_{j-1} \cdot (1 - \alpha_j) + c_j \cdot \alpha_j & \text{para } j = 1, \dots, n \\ c_{j-1} & \text{para } j \geq n + 1 \end{cases} \quad (18)$$

Donde

$$\alpha_j = \frac{\hat{a} - a_j}{a_{j+n} - a_j} \quad (19)$$

Es la coordenada local de $\hat{a}$ con respecto al soporte $[a_j, a_{j+n}]$ de $\hat{N}_j^n$. La Fig. 3 muestra una ilustración para $n = 3$

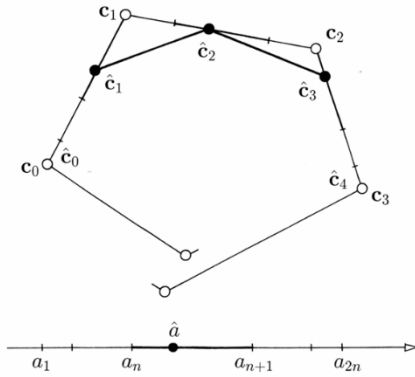


Fig. 4 Inserción de un nuevo nodo

F. Algoritmo de Oslo

El algoritmo de Oslo es un método eficiente para llevar a cabo la inserción de nodos en B-spline. Su ventaja radica en que minimiza el costo computacional al recalcular progresivamente los coeficientes de control sin necesidad de recomputar toda la curva o superficie. Es ampliamente utilizado en aplicaciones de refinamiento de mallas y en la representación precisa de superficies complejas, permitiendo ajustes locales de forma sin comprometer la estructura general. Aunque usualmente la inserción repetida de nodos simples es en general el mejor método [40] para calcular los puntos de control con respecto a cualquier refinamiento $\hat{a}_j$ de a_i , también se puede calcular cada punto c_j a través de la generalización del algoritmo de de Boor

Para calcular c_j se requiere algún intervalo internodal $[a_k, a_{k+1}]$ cuya intersección con el soporte $[\hat{a}_j, \hat{a}_{j+n+1}]$ de $\hat{N}_j^n$ sea no vacía, tal como se ilustra en la Fig. 4. Entonces, cada punto de control

$$\hat{c}_j = s_k[\hat{a}_{j+1}, \dots, \hat{a}_{j+n}] \quad (20)$$

puede calcularse usando el algoritmo generalizado de de Boor. Esta relación de recurrencia para c_j fue descubierta por Cohen, Lyche y Riesenfeld en Oslo en 1980, y se denomina el algoritmo de Oslo [41]. Note que las combinaciones afines en el algoritmo de Oslo en general pueden no ser convexas. En consecuencia, se requieren mejoras adicionales para evitar combinaciones no convexas.

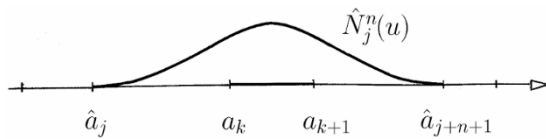


Fig. 3 Elección de a_k para la construcción de c_j

G. Elevación de grado

La elevación de grado es un proceso que incrementa el grado de una B-spline mientras mantiene inalterada su geometría. Esto es especialmente útil para mejorar la continuidad entre diferentes curvas o superficies y para la

conversión de representaciones de menor grado a formatos de mayor compatibilidad en el modelado computacional. Al aumentar el grado, se introducen nuevos puntos de control que preservan la forma original, proporcionando mayor flexibilidad en el refinamiento de modelos.

Todo Spline de grado n

$$s(u) = \sum_i c_i N_i^n(u) \quad (21)$$

Sobre una secuencia de nodos a_i también puede reescribirse en términos de B-Spline N_i^{n+1} de grado $n + 1$

$$s(u) = \sum_i d_i N_i^{n+1}(u) \quad (22)$$

Sobre la secuencia de nodos $\hat{a}_i$ obtenida a partir de a_i elevando la multiplicidad de cada nodo en uno, tal como se ilustra en la Fig. 4

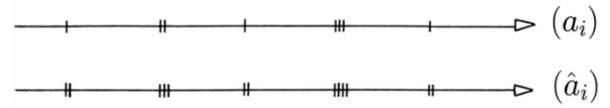


Fig. 5 Secuencia de nodos para la elevación de grado.

El teorema fundamental [31] se obtiene que

$$\begin{aligned} d_j &= s_r[\hat{a}_{j+1}, \dots, \hat{a}_{j+n+1}] \\ &= \frac{1}{n+1} \sum_{k=1}^{n+1} s_r[\hat{a}_{j+1}, \dots, \hat{a}_k^*, \hat{a}_{j+n+1}] \end{aligned} \quad (23)$$

Donde $s_r[u_1, \dots, u_r]$ y $s_r[u_1, \dots, u_{n+1}]$ representan las formas polares en n y $n + 1$ variables, respectivamente, de un segmento polinómico del Spline $s(u)$ que depende de d_j . Cada punto

$$s_r[\hat{a}_{j+1}, \dots, \hat{a}_k^*, \hat{a}_{j+n+1}] \quad (24)$$

Se puede calcular por medio del algoritmo de Boor generalizado, requiriéndose de la inserción de a lo sumo $\left\lceil \frac{n-1}{2} \right\rceil$ nodos. Para el caso de un Spline cubico uniforme, este algoritmo fue descrito por primera vez en [42] para dos segmentos y en [43] para cinco segmentos.

III. IMPLEMENTACIÓN DE PYTHON

A continuación, se mostrarán las clases definidas en Python.

A. Bases B-Spline

La base para construir curvas y superficies B-Spline se deriva de las funciones base B-Spline, como se ha definido matemáticamente. Se utiliza la clase "Bases_curves", que posee dos atributos principales: N_0 y N_p . El primero define la base B-Spline de grado 0, mientras que el segundo permite la

```

algoritmo().vector(n, k)
algoritmo().a_0ij(i, j, x, y)
algoritmo().a_kij(i, j, p, x, y)
algoritmo().elevation(n, k, x)

```

construcción de bases B-Spline de grado p . El funcionamiento de estos atributos se expresa de la siguiente manera:

Donde:

- t representa la secuencia de nodos.
- i es el índice del nodo correspondiente.
- u es el valor de la variable en la función B-Spline.
- p indica el grado de la base.

Esta estructura permite la representación y manipulación de curvas y superficies B-Spline con flexibilidad y precisión en aplicaciones de modelado geométrico.

B. Curvas B-Spline

La clase "BSpline_curve" permite obtener los puntos de una curva B-Spline a través de sus dos atributos: *curva* y *curva1*. A continuación, se muestra el funcionamiento de la clase con sus atributos:

Ambos atributos dependen de cuatro parámetros:

- pts representa los puntos de control.
- nod corresponde a los nodos.
- p es el grado de la curva.
- u es el valor de la variable en la curva B-Spline.

Aunque los atributos *curva* y *curva1* parecen similares, presentan una diferencia clave: en *curva*, el parámetro u solo

```

generation_data(P1, P2, P0, p, q).data_directionU()
generation_data(P1, P2, P0, p, q).data_directionV()
generation_data(P1, P2, P0, p, q).data_bilineal()
generation_data(P1, P2, P0, p, q).data_coons()

```

acepta un valor numérico, mientras que en *curva1*, dicho parámetro puede ser una lista de valores numéricos.

C. Superficies B-Spline

La clase BSpline_surface permite obtener los puntos de una superficie B-Spline, con un funcionamiento similar al de la clase BSpline_curve, como se muestra a continuación

La definición de superficies B-Spline se basa en la suma del producto entre los puntos de control y dos funciones base B-Spline, una asociada a la variable u y otra a v . Para ello, la clase requiere el parámetro *data*, que representa los puntos de control organizados en una matriz donde cada elemento es un punto.

Asimismo, *nod1* y *nod2* corresponden a los nodos de cada función base, mientras que *p1* y *p2* indican los grados de las funciones B-Spline. El atributo *superficie* acepta los parámetros u y v como valores numéricos individuales, mientras que en

procesar, estos parámetros se manejan como arreglos bidimensionales, es decir, matrices.

D. Algoritmos Adicionales

La clase algoritmo contiene cuatro atributos clave en la generación de mallas para superficies de Coons: *vector*, *a_0ij*,

```

Bases_curves(t).N_0(i, u)
Bases_curves(t).N_p(i, u, p)

```

a_kij y *elevation*.

El atributo *vector* acepta dos parámetros, n y k , donde $k \leq n$, y se utiliza para construir nodos abiertos. Los atributos *a_0ij* y *a_kij* están relacionados con el algoritmo de Oslo, en el cual i representa el índice del nodo x y j el del nodo y .

Finalmente, el atributo *elevation* tiene tres parámetros: n , que representa el grado al que se elevará la base, k , que es el índice del nodo abierto, y x , que define la posición del nodo en la estructura.

E. Generación de Data para las superficies de Coons

Las superficies de Coons se obtienen mediante interpolaciones

```

BSpline_curve().curva(pts, nod, p, u)
BSpline_curve().curva1(pts, nod, p, u)

```

lineales entre pares de curvas, combinando dos superficies regladas y restando la superficie bilineal formada por los cuatro puntos de intersección. Para su representación, se emplea la clase *generation_data*, que cuenta con cuatro atributos clave:

- *data_directionU()*: Matriz de puntos para la superficie reglada en dirección u y sus nodos.
- *data_directionV()*: Similar al anterior, pero en dirección v .
- *data_bilineal()*: Matriz de puntos para la superficie bilineal con sus nodos.
- *data_coons()*: Retorna la malla de puntos para graficar la superficie de Coons con sus respectivos nodos.

Su implementación se expresa de la siguiente forma

Donde $P1$ y $P2$ son listas de curvas en función de los parámetros u y v , respectivamente, mientras que $P0$ contiene los cuatro puntos de intersección de las curvas fronterizas. Los parámetros p y q representan los grados de las curvas en u y v .

Las matrices generadas por *data_directionU*, *data_directionV*, *data_bilineal* y *data_coons* tienen dimensiones proporcionales al número de puntos en las curvas de entrada, estructurándose como: (tamaño de $P2 \times$ tamaño de

```

BSpline_surface(data, nod1, nod2, p1, p2).superficie(u, v)
BSpline_surface(data, nod1, nod2, p1, p2).procesar(u, v)

```

$P1$) en todas sus combinaciones.

F. Clase para graficar Superficies B-Spline

Con las clases definidas, es posible graficar superficies de Coons utilizando curvas B-Spline. A continuación, se detallan los parámetros requeridos por la clase para su visualización:

```
my_grafica1 = BSpline_superficie3D_graph(data, nod1,
nod2, p1, p2)
my_grafica1.grafica(enmallar, pts_ctrl, superficie,
cur, **kwargs)
```

Primero, se inicializa la clase `BSpline_superficie3D_graph`, donde los parámetros representan la data o malla de la superficie B-Spline, junto con los nodos y sus respectivos grados que definen cada función base. Luego, el atributo `grafica` permite visualizar la superficie.

Los parámetros `enmallar`, `pts_ctrl`, `superficie` y `cur` son booleanos que determinan si se incluyen en la gráfica el mallado, los puntos de control, la superficie y la curva, respectivamente. Además, el parámetro `kwargs` permite agregar opciones adicionales para la visualización, como `colscale='viridis'` o `showscale=False`, entre otras.

III. CONSTRUCCIÓN DE SUPERFICIES DE COONS

A. Primera superficie de Coons

A continuación, se presentan los puntos de control de las curvas en los parámetros `u` y `v`, almacenados en `P1` y `P2`, respectivamente. Los cuatro puntos de intersección de las curvas están definidos en `P0`:

```
P1 = np.array([[0, 0, 0], [1/2, 0, 0], [3/2, 0, 0],
[3/2, 0, 1], [7/2, 0, 1], [7/2, 0, 0], [9/2, 0, 0],
[5, 0, 0]], [[0, 5, 0], [1, 5, 0], [2, 5, 0], [2, 5,
1], [3, 5, 1], [3, 5, 0], [4, 5, 0], [5, 5, 0]])
P2 = np.array([[0, 0, 0], [0, 1, 1/10], [0, 5/2,
3/10], [0, 4, 1/10], [0, 5, 0]], [[5, 0, 0], [5, 1,
1/10], [5, 5/2, 3/10], [5, 4, 1/10], [5, 5, 0]])
P0 = np.array([[0, 0, 0], [5, 0, 0]], [[0, 5, 0], [5,
5, 0]])
```

A continuación, se llama a la clase `generation_data` con su atributo `data_directionU` para generar la malla de puntos:

```
mi_data, nod1, nod2 = generation_data(P1, P2, P0, 2,
2).data_directionU()
```

Para graficar la superficie reglada definida por las curvas del parámetro `u`, se emplea el siguiente código:

```
curval = BSpline_superficie3D_graph(data=mi_data,
nod1=nod1, nod2=nod2, p1=2, p2=2)
curval.grafica(enmallar=False, pts_ctrl=False,
superficie=True, cur=True, frontera=True,
colscale='viridis', showscale=False)
```

A continuación, vamos a mostrar las curvas fronteras, ver Fig. 6 que se utilizarán para la construcción de la superficie, donde estas curvas su importancia se explica desde la Fig. 1.

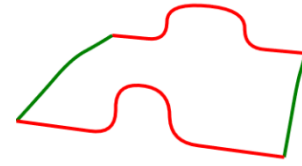


Fig. 6 Curvas Fronteras para la primera superficie de Coons

Este procedimiento permite visualizar la superficie reglada generada a partir de las curvas de control.

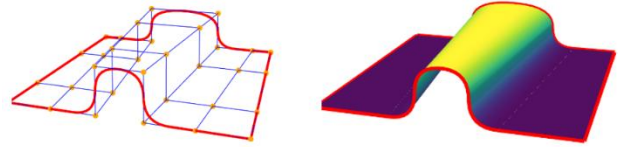


Fig. 7 Primera generación de malla de puntos y su superficie reglada

Para graficar la segunda superficie reglada, se sigue un procedimiento similar al anterior. Se llama a la clase `generation_data` con su atributo `data_directionV`:

```
mi_data, nod1, nod2 = generation_data(P1, P2, P0, 2,
2).data_directionV()
```

Luego, se grafica la superficie reglada definida por las curvas en el parámetro `v` utilizando el siguiente código:

```
Curvav = BSpline_superficie3D_graph(data=mi_data,
nod1=nod1, nod2=nod2, p1=2, p2=2)
Curvav.grafica(enmallar=False, pts_ctrl=False,
superficie=True, cur=True,
colscale='viridis', showscale=False)
```

Este proceso permite visualizar la segunda superficie reglada generada a partir de las curvas de control en la dirección `v`.

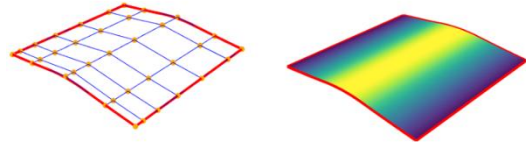


Fig. 8 Segunda generación de malla de puntos y su superficie reglada

Para graficar la superficie de Coons, se utiliza el atributo `data_coons` de la clase `generation_data` para obtener el mallado:

```
mi_data, nod1, nod2 = generation_data(P1, P2, P0, 2,
2).data_coons()
```

Luego, se emplea la siguiente instrucción para generar la visualización de la superficie:

```
Coons1 = BSpline_superficie3D_graph(data=mi_data,
nod1=nod1, nod2=nod2, p1=2, p2=2)
Coons1.grafica(enmallar=False, pts_ctrl=False,
superficie=True, cur=True,
colscale='viridis', showscale=False)
```

Este procedimiento permite obtener la representación final de la superficie de Coons con el mallado generado a partir de los datos de control.

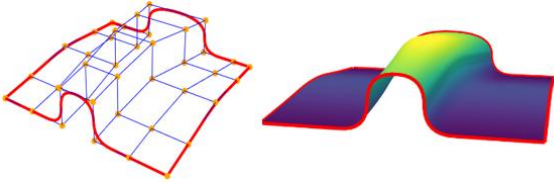


Fig. 9 A partide del enmallado de puntos se obtiene la superficie Coons

B. Segunda superficie de Coons

A partir del ejemplo anterior, consideremos ahora la siguiente Data

```
P1 = np.array([[0, 0, 0], [1/2, 0, 0], [3/2, 0, 0],
[3/2, 0, 1], [7/2, 0, 1], [7/2, 0, 0], [9/2, 0, 0],
[5, 0, 0]], [[0, 5, 0], [1, 5, 0], [2, 5, 0], [2, 5,
1], [3, 5, 1], [3, 5, 0], [4, 5, 0], [5, 5, 0]])
P2 = np.array([[0, 0, 0], [0, 1/2, 1/10], [0, 3/2,
3/10], [0, 3/2, 13/10], [0, 7/2, 13/10], [0, 7/2,
3/10], [0, 9/2, 1/10], [0, 5, 0]], [[5, 0, 0], [5, 1,
1/10], [5, 2, 3/10], [5, 2, 13/10], [5, 3, 13/10], [5,
3, 3/10], [5, 4, 1/10], [5, 5, 0]])
P0 = np.array([[0, 0, 0], [5, 0, 0]], [[0, 5, 0], [5,
5, 0]])
```

Sus curvas fronteras a partir de estos puntos son, ver Fig. 10



Fig. 10 Curvas fronteras de la segunda superficie de Coons

Para graficar las dos superficies regladas y la superficie de Coons, se sigue el mismo procedimiento detallado anteriormente.

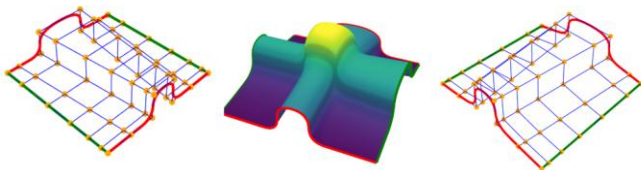


Fig. 11 Superficie de Coons a través de dos enmallados de puntos

C. Tercera superficie de Coons

Ahora consideremos la siguiente lista de puntos

```
P1 = np.array([[0, 0, 0], [1/2, 0, 0], [3/2, 0, 0],
[3/2, 0, 1], [7/2, 0, 1], [7/2, 0, 0], [9/2, 0, 0],
[5, 0, 0]], [[0, 5, 0], [1, 5, 0], [2, 5, 0], [2, 5, -1],
[3, 5, -1], [3, 5, 0], [4, 5, 0], [5, 5, 0]])
P2 = np.array([[0, 0, 0], [0, 1/2, 1/10], [0, 3/2,
3/10], [0, 3/2, 13/10], [0, 7/2, 13/10], [0, 7/2,
3/10], [0, 9/2, 1/10], [0, 5, 0]], [[5, 0, 0], [5, 1,
-1/10], [5, 2, -3/10], [5, 2, -13/10], [5, 3, -13/10],
[5, 3, -3/10], [5, 4, -1/10], [5, 5, 0]])
P0 = np.array([[0, 0, 0], [5, 0, 0]], [[0, 5, 0], [5,
5, 0]])
```

Las curvas fronteras generadas son, ver Fig. 12

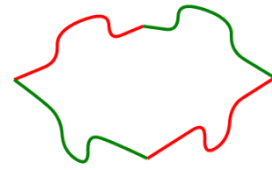


Fig. 12 Curvas fronteras para la tercera superficie de Coons

El enmallado y la superficie de Coons resultante es, ver Fig. 13

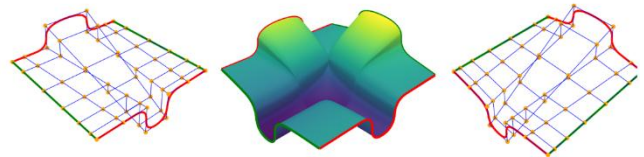


Fig. 13 Superficie de Coons a través de dos enmallados de puntos

D. Cuarta superficie de Coons

Finalmente, se considera los puntos

```
P1 = np.array([[0, 0, 0], [1/2, 0, 0], [3/2, 0, 0],
[3/2, 0, 1], [7/2, 0, 1], [7/2, 0, 0], [9/2, 0, 0],
[5, 0, 0]], [[0, 5, 0], [1, 5, 0], [2, 5, 0], [2, 5,
1], [3, 5, 1], [3, 5, 0], [4, 5, 0], [5, 5, 0]])
P2 = np.array([[0, 0, 0], [0, 1/2, -1/10], [0, 3/2, -
3/10], [0, 3/2, -13/10], [0, 7/2, -13/10], [0, 7/2, -
3/10], [0, 9/2, -1/10], [0, 5, 0]], [[5, 0, 0], [5, 1,
-1/10], [5, 2, -3/10], [5, 2, -13/10], [5, 3, -13/10],
[5, 3, -3/10], [5, 4, -1/10], [5, 5, 0]])
P0 = np.array([[0, 0, 0], [5, 0, 0]], [[0, 5, 0], [5,
5, 0]])
```

Donde las curvas fronteras son, ver Fig. 14

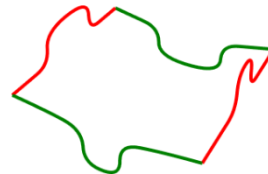


Fig. 14 Curvas fronteras para la cuarta superficie de Coons

El enmallado y la superficie de Coons resultante es, ver Fig. 15

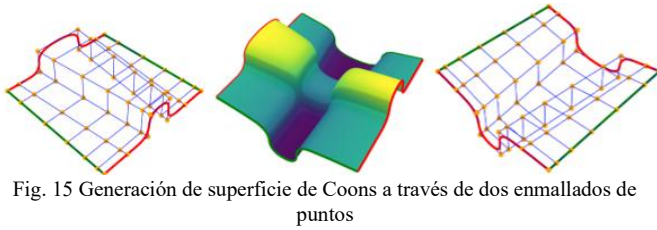


Fig. 15 Generación de superficie de Coons a través de dos enmallados de puntos

IV. DISCUSIÓN

En este estudio se evidencia la efectividad de la representación de superficies de Coons mediante B-Spline en Python, permitiendo una mayor flexibilidad y continuidad en el modelado geométrico. La implementación en Jupyter Notebook facilitó la visualización y manipulación de estas superficies, proporcionando una plataforma accesible para el análisis de interpolaciones y la construcción de mallas de control.

Desde el punto de vista computacional, la utilización de funciones base B-Spline mejora la precisión en la aproximación de superficies, optimizando su continuidad sin necesidad de estructuras adicionales. No obstante, el principal desafío radica en la complejidad computacional al incrementar el grado de las funciones base, lo que puede afectar el rendimiento en modelos de gran escala. Sin embargo, técnicas como la inserción de nodos y la elevación de grado permiten un control más refinado sobre la forma y suavidad de la superficie, mitigando estos efectos.

En términos aplicados, la conversión de superficies de Coons a B-Spline abre posibilidades en áreas como el diseño geométrico asistido por computadora (CAGD), la simulación en ingeniería y la animación digital. La compatibilidad con herramientas de modelado paramétrico y la capacidad de interpolar datos dispersos refuerzan su aplicabilidad en contextos industriales y académicos. Además, se propone un modelo de integración entre conceptos matemáticos avanzados y herramientas computacionales accesibles, promoviendo así un aprendizaje interactivo y práctico, ya que, no solo facilita la enseñanza de conceptos de programación, sino que también mejora la preparación profesional de los estudiantes al integrarse con tecnologías innovadoras.

V. CONCLUSIONES

En nuestro estudio, la representación de superficies de Coons mediante B-Spline en Python se presenta como una alternativa eficiente y flexible para el modelado geométrico. Esta metodología permite mejorar la continuidad y precisión de las superficies, facilitando su implementación en diseño geométrico asistido por computadora, simulación y enseñanza. Los resultados evidencian que la conversión a B-Spline optimiza la interpolación y el control sobre la geometría sin requerir estructuras adicionales. Sin embargo, el aumento en la complejidad computacional sigue siendo un desafío, mitigable mediante técnicas como la inserción de nodos y la elevación de grado. Se sugiere optimizar el rendimiento computacional y

explorar la integración con NURBS para ampliar su aplicabilidad en modelado avanzado.

REFERENCIAS

- [1] E. R. G. Shankar, "Computer- Aided Design in the Aerospace Industry," *International Journal For Multidisciplinary Research*, vol. 5, no. 4, Aug. 2023, doi: 10.36948/ijfmr.2023.v05i04.5307.
- [2] A. P. Michel, "Pierre Bézier et les machines transfert Renault (1940-1960). Quand le cinéma ne suffit pas à consacrer un héros de l'innovation industrielle," vol. 4, no. 4, Sep. 2019, doi: 10.21494/ISTE.OP.2019.0410.
- [3] A. Tolok, N. B. Tolok, and A. Sycheva, "Construction of the Functional Voxel Model for a Spline Curve," Dec. 2020, doi: 10.51130/GRAPHICON-2020-2-3-52.
- [4] Y. Huang, "3D special effects modelling based on computer graphics technology," *Applied and Computational Engineering*, Mar. 2024, doi: 10.54254/2755-2721/50/20241280.
- [5] X. Jiang and Y. Lin, "Reparameterization of B-spline surface and its application in ship hull modeling," *Ocean Engineering*, Oct. 2023, doi: 10.1016/j.oceaneng.2023.115535.
- [6] R. Świta and Z. Suszyński, "B-splines image approximation using resampled chordal parameterization," *Iet Image Processing*, Jun. 2024, doi: 10.1049/ipr.2.13150.
- [7] S. Zhang and X. Liu, "Design of Biquintic B-spline Surface Construction Algorithm," *Journal of applied mathematics and computation*, vol. 7, no. 2, pp. 298–303, Jul. 2023, doi: 10.26855/jamc.2023.06.011
- [8] J. Yang, X. Jia, and D.-M. Yan, "Topology Guaranteed B-Spline Surface/Surface Intersection," *ACM Transactions on Graphics*, vol. 42, pp. 1–16, Dec. 2023, doi: 10.1145/3618349.
- [9] W. Wang, Z. Guo, and G. Zhao, "B spline curved surface reconstruction method," Sep. 06, 2019.
- [10] C. G. Provatidis, "COONS' Interpolation as a Vehicle to Derive Large Isoparametric Elements," *Springer, Cham*, 2019, pp. 93–174. doi: 10.1007/978-3-030-03889-2_3.
- [11] K.-L. Shi, S. Zhang, H. Zhang, J.-H. Yong, J. Sun, and J.-C. Paul, "G 2 B-spline interpolation to a closed mesh," *Computer-aided Design*, vol. 43, no. 2, pp. 145–160, Feb. 2011, doi: 10.1016/J.CAD.2010.10.004.
- [12] W. Boehm, "Über die Konstruktion von B-Spline-Kurven," *Computing*, vol. 18, pp. 161–166, 1977.
- [13] W. Boehm, "Inserting new knots into B-Spline-Curves," *Computer-Aided Design*, vol. 12, no. 4, pp. 199–201, 1980.
- [14] W. Boehm, "Generating the Bézier points of triangular splines," in *Surfaces in Computer Aided Geometric Design*, R. E. Barnhill and W. Boehm, Eds. North-Holland: Elsevier Science Publishers B.V., 1983, pp. 77–91.
- [15] W. Boehm, "Curvature continuous curves and surfaces," *Computer Aided Geometric Design*, vol. 2, pp. 313–323, 1985.
- [16] W. Boehm, "Smooth curves and surfaces," in *Geometric Modeling: Algorithms and New Trends*, G. Farin, Ed. SIAM, 1987, pp. 175–184.
- [17] W. Boehm, H. Prautzsch, and P. Arner, "On triangular splines," *Constructive Approximation*, vol. 3, pp. 157–167, 1987.
- [18] W. Boehm, "Calculating with box splines," *Computer Aided Geometric Design*, vol. 1, pp. 149–162, 1984.
- [19] W. Boehm, "Efficient evaluation of splines," *Computing*, vol. 33, pp. 171–177, 1984.
- [20] W. Boehm, G. Farin, and J. Kahmann, "A survey of curve and surface methods in CAGD," *Computer Aided Geometric Design*, vol. 1, no. 1, pp. 1–60, 1984.
- [21] W. Boehm and H. Prautzsch, *Numerical Methods*. Vieweg, Wiesbaden, and AK Peters, Wellesley, Mass., 1993.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.
 - [26] 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>.
 - [27] 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.
 - [28] 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.
 - [29] R. Duque, "Python para todos". Disponible en: <http://mundogeek.net/tutorial-python/>, 2011.
 - [30] 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.
 - [31] H. Prautzsch, M. Paluszny, and W. Böhm, Métodos de Bézier y B-splines. KIT Scientific Publishing, 2005.
 - [32] Cordero Juan y Cortés José, Curvas y Superficies para Modelado Geométrico, Alfaomega, 2002.
 - [33] Salomon David, Curves and Surfaces for Computer Graphics, Springer, 2006.
 - [34] Fengqiang Lin y W. T. Hewitt, "Expressing Coons-Gordon surface as NURBS," Computer-Aided Design, vol. 26, pp. 145–155, 1994.
 - [35] Coons, Steven A, Surfaces for Computer-Aided Design of Space Figures, Cambridge, MA, MIT Project MAC, report MAC-M-253, January 1964.
 - [36] Farin, Gerald, Curves and Surfaces for CAGD (Computer Aided Graphics and Design), San Diego, Academic Press, 2001.
 - [37] A. R. Khlefha, "Comprehensive Review of Interpolation Techniques Using B-Spline," International Academic Journal of Science and Engineering, vol. 11, no. 1, pp. 304–311, Jun. 2024, doi: 10.9756/iajse/v11i1/iajse1135.
 - [38] Li, L. (2020). Application of Cubic B-spline Curve in Computer-Aided Animation Design. Computer-Aided Design and Applications, 18, 43–52. <https://doi.org/10.14733/CADAPS.2021.S1.43-52>
 - [39] P. J. Kulkarni and U. B. Hatwar, "Scaling of Color Image using B-spline Curves," no. 1, pp. 5–8, Aug. 2015, [Online]. Available: <https://research.ijcaonline.org/ncsc2015/number1/ncsc7323.pdf>
 - [40] E. Cohen, T. Lyche, and L.L. Schumaker, "Algorithms for degree raising of splines," ACM Transactions on Graphics, vol. 4, no. 3, pp. 171–181, 1985.
 - [41] E. Cohen, T. Lyche, and R.F. Riesenfeld, "Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics," Computer Graphics and Image Processing, vol. 14, pp. 87–111, 1980.
 - [42] L. Ramshaw, "Blossoming: a Connect-the-Dots Approach to Splines," Technical report, Digital Systems Research Center, Palo Alto, California, June 21, 1987.
 - [43] H.-P. Seidel, "Computing B-spline control points," in Theory and Practice of Geometric Modeling, W. Strasser and H.-P. Seidel, Eds. Springer, Berlin, 1989, pp. 17–32.