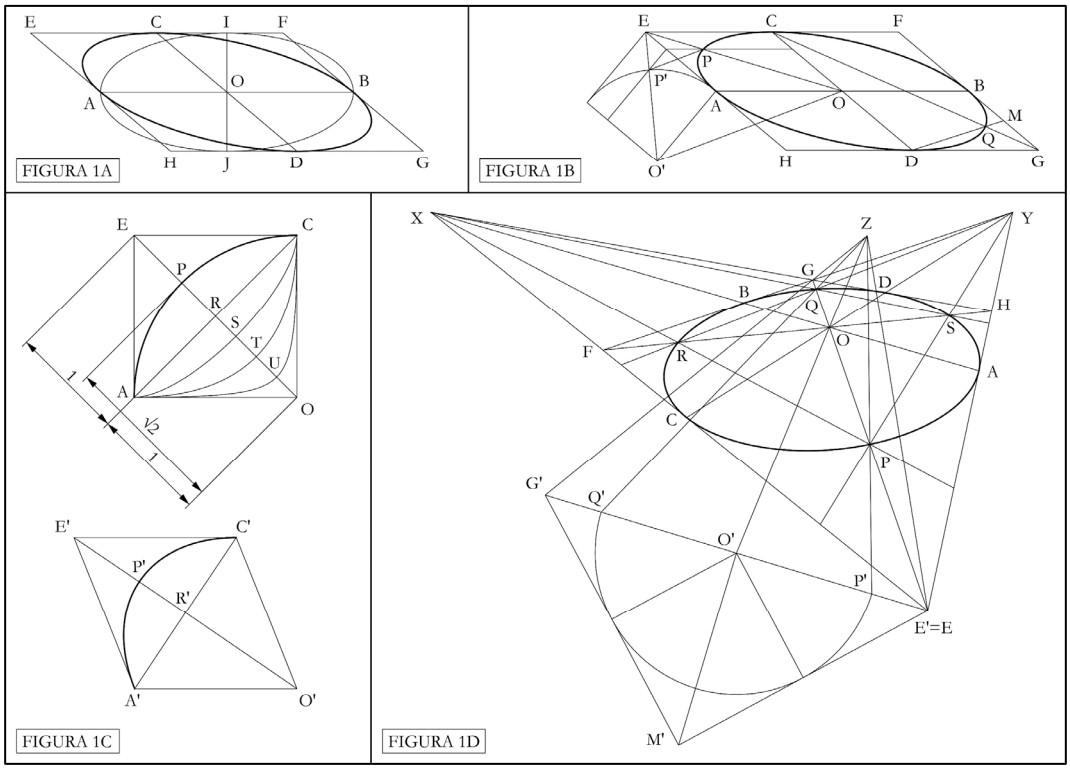
Trazado de elipses a partir del cuadrilátero circunscrito: script de Python para Rhinoceros 7

Pau Natividad-Vivó; Ricardo García-Baño; Macarena Salcedo-Galera; José Calvo-López

Departamento de Arquitectura y Tecnología de la Edificación, Universidad Politécnica de Cartagena;



Figuras 1A, 1B, 1C y 1D. Procedimentos para el trazado de elipses dado el cuadrilátero circunscrito.

Resumen

El presente trabajo revisa, primero, algunos procedimientos de Rhinoceros 7 para el trazado de elipses que representan circunferencias contenidas en planos oblicuos al plano de proyección y, después, expone un procedimiento general válido para todos los casos independientemente de que se emplee un sistema de proyección paralela —sistema diédrico y perspectiva axonométrica— o de proyección cónica —perspectiva cónica—. Los primeros procedimientos permiten dibujar una elipse a partir del paralelogramo circunscrito, mientras que el procedimiento general emplea como dato de entrada un cuadrilátero circunscrito convexo cualquiera. Además, este procedimiento general se ha programado y automatizado en un script de Python para Rhinoceros 7 cuyo código se muestra al final.

Palabras clave

Elipse; CAD; Rhinoceros; script; Python

1. Introducción al problema

Un problema recurrente en el dibujo arquitectónico —donde se usan los sistemas de representación diédrico, axonométrico y cónico fundamentalmente— es el trazado de elipses que representan circunferencias contenidas en planos oblicuos al plano de proyección. Los programas de CAD tienen comandos para dibujar elipses, pero muchos presentan una limitación importante: solo permiten trazar una elipse a partir del centro y los vértices. Si conocemos estos puntos podemos dibujar la elipse; pero si no los conocemos —caso habitual—, tendremos que hallarlos, lo que puede ser complejo y laborioso. En vista de ello, algunos autores han ideado procedimientos sencillos que sortean las limitaciones de los programas de CAD y facilitan el dibujo de estas elipses. En el presente trabajo revisaremos varios procedimientos para Rhinoceros 7 y expondremos un procedimiento general válido para todos los casos, el cual se ha programado en un script.

2. Procedimientos en diédrico y axonometría

El sistema diédrico y la perspectiva axonométrica se fundamentan en la proyección paralela. Esta propiedad es de gran ayuda a la hora de trazar una elipse que representa una circunferencia contenida en un plano oblicuo al plano de proyección, ya que cualquier par de ejes ortogonales de la circunferencia se proyectará como un par de diámetros conjugados de la elipse. Y con dos diámetros conjugados se puede hallar el centro y los vértices de la elipse, usando, por ejemplo, los métodos de Chasles, Mannheim o Rytz (Izquierdo 2002, p. 121). Además de estos, conocemos otros dos procedimientos para dibujar una elipse mediante Rhinoceros dados dos diámetros conjugados. Estos dos procedimientos usan los comandos “Sesgar” y “Cónico” y su principal interés reside en que son fáciles de recordar y rápidos de ejecutar (Calvo, Alonso y Sanz 2010; García, González y López 2010).

2.1. Procedimiento con el comando “Sesgar”

Queremos dibujar la elipse de diámetros conjugados AB, CD (Fig. 1A). Para ello, trazamos el paralelogramo EFGH de lados paralelos a AB, CD que pasan por A, B, C, D. Dibujamos la perpendicular a AB por O que corta a EF, GH en I, J. Dibujamos la elipse de ejes AB, IJ. Usamos el comando “Sesgar” en la elipse con origen O, punto inicial I y final C. Este comando deforma la elipse en la dirección IC, convirtiendo OI en OC. El resultado es la elipse buscada. Conviene advertir que la transformación es, en términos geométricos, una afinidad entre elipses.

2.2. Procedimiento con el comando “Cónico”

Queremos dibujar la elipse de diámetros conjugados AB, CD (Fig. 1B). Para ello, dibujamos el paralelogramo EFGH y con el comando “Cónico” trazamos la elipse. Este comando no dibuja la elipse completa, sino un arco; en consecuencia, hay que repetir el comando cuatro veces para dibujar los cuatro arcos elípticos AC, CB, BD, DA. Cada arco se dibuja a partir de cuatro datos: el punto inicial; el punto final; el vértice o punto de intersección entre las tangentes al arco en los puntos inicial y final; y un punto de paso del arco o un factor denominado rho.

El punto de paso es un punto del arco cuya posición determina si el arco es elíptico, parabólico o hiperbólico. Una manera de ubicar correctamente el punto de paso dentro del paralelogramo AECO es la siguiente (Fig. 1B): dibujamos un cuadrado a partir de uno de los lados del paralelogramo, por ejemplo AE. Hallamos el punto P’ de intersección entre la diagonal O’E del cuadrado y el arco circular de centro O’ y radio O’A. Trasladamos P’ desde cuadrado hasta el paralelogramo mediante paralelas a los lados y obtenemos P. Hecho esto, ya podemos dibujar el arco elíptico APC. Para ello, empleamos “Cónico” con punto inicial A, final C, vértice E y punto de paso P. Los tres arcos restantes se dibujan igual. Conviene señalar que esta construcción gráfica es, en realidad, una afinidad entre un arco circular auxiliar y el arco elíptico a dibujar.

Otra forma sencilla de obtener puntos de paso es con haces proyectivos (Izquierdo 2002, p. 122). Para ello, se procede como sigue (Fig. 1B): hallamos el punto medio M del lado BG del paralelogramo BGDO. Hecho esto, la intersección entre CG, DM es el punto de paso Q.

Como ya se ha dicho, también podemos usar el factor rho para trazar un arco cónico. Rho es un factor de valor mayor que 0 y menor que 1 que define la posición de un punto de paso. Analicemos cómo funciona (Fig. 1C): sea AECO un cuadrado con centro en R y semidiagonal RO de longitud igual a la unidad. Dividimos RO en cuatro partes iguales y obtenemos S, T, U. Entonces, si trazamos un arco cónico con punto inicial A, final C, vértice O y rho 0.25 se genera el arco elíptico ASC; si repetimos el proceso con rho 0.50 se genera el arco parabólico ATC; y con rho 0.75 se genera el arco hiperbólico AUC. Hecho esto, no es difícil calcular rho para un arco circular como, por ejemplo, APC (Fig. 1C): en este caso rho vale $\sqrt{2} - 1$, es decir, 0.4142…, que es la distancia entre R, P.

Rho con valor $\sqrt{2} - 1$ también sirve para dibujar una elipse que representa una circunferencia contenida en un plano oblicuo al plano de proyección. Y esto es así porque la razón simple de tres puntos colineales es un invariante en las proyecciones paralelas (Izquierdo 2002, pp. 9-11). Dicho de otra manera (Fig. 1C): la relación de distancias entre los puntos E, P, R del arco circular se conserva en los puntos E’, P’, R’ del arco elíptico, pues uno se puede obtener del otro por proyección paralela.

3. Procedimiento en perspectiva cónica

El proceso para trazar una elipse que representa una circunferencia contenida en un plano oblicuo al plano de proyección se complica en la perspectiva cónica, ya que esta perspectiva se basa en la proyección cónica, cuyos rayos proyectantes no son paralelos sino convergentes. Esto significa que el centro o cualquier par de ejes ortogonales de una circunferencia ya no tienen por qué coincidir con el centro o un par de diámetros conjugados

de la correspondiente elipse proyectada. Por lo tanto no podemos usar los procedimientos de "Sesgar" o "Cónico" con rho vistos antes; sin embargo, sí podemos emplear el procedimiento de "Cónico" con punto de paso. Es aquí donde queremos a realizar nuestra aportación.

3.1. Objetivo

Exponer un procedimiento general —es decir, válido para el sistema diédrico y las perspectivas axonométrica y cónica— que permita trazar una elipse que representa una circunferencia contenida en un plano oblicuo al plano de proyección. Asimismo, programar dicho procedimiento en un script de Python compatible con Rhinoceros 7.

3.2. Datos de entrada

En los procedimientos anteriores hemos visto que los diámetros conjugados eran el punto de partida para trazar la elipse. Sin embargo, en todos los casos lo que realmente hemos hecho es dibujar un paralelogramo circunscrito y, a partir del mismo, hemos realizado las operaciones gráficas. Ahora emplearemos un cuadrilátero circunscrito convexo cualquiera que, al igual que el paralelogramo anterior, se debe entender como la proyección de un cuadrado.

3.3. Procedimiento con "Cónico" y punto de paso

La geometría proyectiva ofrece varias estrategias para hallar el centro, los vértices y otros puntos notables de las curvas cónicas (Méndez et al. 1997, capítulo IV). Nosotros vamos a exponer un procedimiento sencillo que emplea el comando "Cónico" con punto de paso (Fig. 1D).

Comenzamos el proceso con el cuadrilátero convexo EFGH. Hallamos O como intersección de las diagonales EG, FH. Hallamos X, Y por prolongación de los lados opuestos EF, GH y HE, FG respectivamente. Trazamos las rectas XO, YO y sus intersecciones con el cuadrilátero nos dan los puntos A, B, C, D. Conviene aclarar que si los lados opuestos EF, GH son paralelos, X será un punto en el infinito y la recta XO será una paralela a EF, GH que pasa por O. Lo mismo ocurre con la recta YO. Llegados a este punto, tenemos los vértices E, F, G, H y los puntos iniciales y finales A, B, C, D de los arcos elípticos.

Los puntos de paso se pueden obtener de diferentes maneras, pero en este caso hemos decidido hallarlos en las diagonales del cuadrilátero —luego explicaremos el porqué de esta decisión—. Para ello, dibujamos un segmento E'G' de dirección y magnitud cualesquiera tal que E' coincida con E y G' no coincida con G. Entonces llamamos O' al punto medio de E'G', prolongamos las rectas O'O, G'G y localizamos el punto de intersección Z. Lo que realmente hemos hecho con esta construcción gráfica es establecer una relación proyectiva entre las rectas EG, E'G'. Ahora ambas rectas se pueden entender como secciones de un mismo haz de rectas de vértice en Z, de tal manera que a cada punto en EG le corresponde un homólogo en E'G'. Ahora consideramos que E'G' es la diagonal del medio cuadrado E'G'M' y en su interior trazamos un semicírculo de centro O' que corta a E'G' en los puntos P', Q'. Luego proyectamos P', Q' sobre la diagonal EG mediante rectas convergentes al vértice Z y obtenemos los dos puntos de paso P, Q. Si dibujamos las rectas XP, XQ o YP, YQ podemos hallar los puntos de paso R, S en la diagonal FH. Por último, trazamos los cuatro arcos elípticos utilizando "Cónico". El arco APC, por ejemplo, se dibujaría con punto inicial A, final C, vértice E y punto de paso P.

3.4. Justificación del procedimiento

Para hallar los puntos de paso P, Q en la diagonal EG hemos realizado una construcción gráfica —el semicírculo dentro del medio cuadrado E'G'M'— que puede resultar más laboriosa de dibujar que otro tipo de trazados, como, por ejemplo, los haces proyectivos. Sin embargo, hemos decidido hallar los puntos de paso de esta manera porque su implementación en el código es muy sencilla. Esto es así porque usamos el concepto de razón doble de cuatro puntos colineales, un invariante en proyecciones paralelas y cónicas (Méndez et al. 1997, pp. 23-29; Izquierdo 2002, pp. 9-11). En el caso que nos ocupa, sabemos que la razón doble de los puntos E', O', G', P' de la diagonal E'G' es constante para cualquier cuadrado con una circunferencia inscrita. Y puesto que la razón doble se conserva en las proyecciones, entonces podemos hallar la posición del punto P sobre la diagonal EG conocidos E, O, G.

Es más, la razón doble se conserva en la proyección de cualquier cuaterna de puntos colineales. Por ejemplo, conocidos E, H, Y se puede hallar A o conocidos E, F, X se puede hallar C. Esto significa que es posible programar una función basada en la razón doble y usarla para hallar todos los puntos necesarios en el procedimiento. De esta manera se ahorra código en el script, ganando claridad y legibilidad.

3.5. Programación en un script

Todo el procedimiento anterior se ha programado en un script de Python para Rhinoceros 7 (Fig. 2) que usa la biblioteca de comandos RhinoScriptSyntax disponible en: https://developer.rhino3d.com/api/RhinoScriptSyntax/.

El script es un archivo de texto plano convencional con extensión PY que se puede abrir y editar en cualquier editor de texto. Para ejecutarlo se escribe en la línea de comandos "EjecutarScriptDePython". Aparece entonces una ventana que permite buscar el archivo y seleccionarlo. Otro modo de ejecutar el script rápidamente es asociarlo a un botón. Una vez que se ejecuta, se solicita al usuario que introduzca los cuatro vértices del cuadrilátero circunscrito. Los vértices deben introducirse ordenadamente en sentido horario o antihorario. Entonces el script comprueba que los vértices son válidos y dibuja la elipse. El script también se puede editar dentro del propio programa Rhinoceros. Para ello, se escribe "EditarScriptDePython" en la línea de comandos y aparece la ventana del editor.

Por último, queremos indicar que esperamos poder subir el archivo a la web para que esté a disposición de los

usuarios interesados.

```
import rhinoscriptsyntax as rs

def GetPoints():
    pts,obj,txt=[],[],'Select vertex {}'
    for i in range(4):
        gpt=rs.GetPoint(txt.format(i+1))
        if not gpt:
            if obj!=[]: rs.DeleteObjects(obj)
            exit()
        pts.append(gpt)
        obj.append(rs.AddPoint(gpt))
    rs.DeleteObjects(obj)
    return pts

def CheckPoints(pts):
    #Points must be coplanar
    if not rs.PointsAreCoplanar(pts): return
    #Points must be different (not repeated)
    for i in range(0,3):
        for j in range(i+1,4):
            if rs.PointCompare(pts[i],pts[j]):
                return
    #Points must form a convex quadrilateral
    ipt=rs.LineLineIntersection((pts[0],pts[2]),
                                (pts[1],pts[3]))
    if ipt==None: return
    if not rs.PointCompare(ipt[0],ipt[1]): return
    for i,j in [[0,2],[2,0],[1,3],[3,1]]:
        distance1=rs.Distance(pts[i],ipt[0])
        distance2=rs.Distance(pts[i],pts[j])
        if distance1>=distance2: return
    return pts

def CrossRatioPoint(a,b,c,value):
    # Returns d which (a,b,c,d)=value
    def Sign(p,q):
        for i in range(3):
            # Positive sense from a to b
            if (b[i]-a[i])*(q[i]-p[i])<0:
                return -1
        return +1
    ab=rs.Distance(a,b)
    #c = infinity point
    if c==None:
        ad=ab/2
    #c = real point
    else:
        ac=Sign(a,c)*rs.Distance(a,c)
        bc=Sign(b,c)*rs.Distance(b,c)
        ad=(ab*ac)/(ac-(value*bc))
    vector=rs.VectorCreate(b,a)
    vector=rs.VectorUnitize(vector)
    vector=rs.VectorScale(vector,ad)
    d=rs.PointAdd(a,vector)
    return d

def DrawEllipse(pts):
    list1=[pts[i] for i in [0,1,2,3]]
    list2=[pts[i] for i in [1,2,3,0]]
    list3=[pts[i] for i in [2,3,0,1]]
    list4=[pts[i] for i in [3,0,1,2]]
    for E,F,G,H in [list1,list2,list3,list4]:
        X=rs.LineLineIntersection((E,F),(G,H))
        if X!=None: X=X[0]
        Y=rs.LineLineIntersection((E,H),(F,G))
        if Y!=None: Y=Y[0]
        O=rs.LineLineIntersection((E,G),(F,H))[0]
        A=CrossRatioPoint(E,H,Y,-1)
        C=CrossRatioPoint(E,F,X,-1)
        P=CrossRatioPoint(E,O,G,-2/((2**0.5)-1))
        rs.Command('_Conic '+\
                   str(A)+' '+str(C)+' '+\
                   str(E)+' '+str(P),False)

if __name__=="__main__":
    pts=GetPoints()
    if CheckPoints(pts):
        rs.EnableRedraw(False)
        DrawEllipse(pts)
        rs.EnableRedraw(True)
```

Figura 2. Código del script en el editor de Rhinoceros.

Referencias

Calvo López, J., Alonso Rodríguez, M. Á. y Sanz Alarcón, J. P., 2010. Algunos problemas planteados por el trazado de elipses y la representación de circunferencias en el dibujo por ordenador. (Póster). *Actas del XIII Congreso Internacional EGA*. Valencia: UPV.

García Reig, C., González Uriel, A. y López Mozo, A., 2010. *Cónicas. Elementos principales y manejo con Rhinoceros 4.0. Apuntes de la asignatura Geometría y Dibujo de Arquitectura 1, Curso 2010-11*. Madrid: ETSAM.

Izquierdo Asensi, F., 2002. *Construcciones geométricas*. Madrid: Francisco Javier Izquierdo Ruiz de la Peña.

Méndez Valentín, L., Martínez Simón, J. M., González Gámez, F., Gordo Murillo, C., Martínez Marín, R., 1997. *Geometría proyectiva. Tomo 1. Formas geométricas fundamentales*. Madrid: ETSICCP.

Datos biográficos de los autores

Pau Natividad-Vivó

Universidad Politécnica de Cartagena (UPCT); pau.natividad@upct.es

Profesor contratado doctor de la UPCT. Arquitecto y Máster en conservación del patrimonio arquitectónico. Su docencia e investigación se centran en la Expresión gráfica arquitectónica y la Historia de la construcción.

Ricardo García-Baño

Universidad Politécnica de Cartagena (UPCT); ricardo.garcia@upct.es

Profesor asociado de la UPCT. Doctor arquitecto y Máster en investigación y gestión del patrimonio histórico artístico y cultural. Compagina la investigación en cantería y estereotomía con la actividad profesional en su propio estudio. Autor de numerosas publicaciones en congresos y revistas especializadas, así como de trabajos profesionales premiados.

Macarena Salcedo-Galera

Universidad Politécnica de Cartagena (UPCT); macarena.salcedo@upct.es

Profesora contratada doctora de la UPCT. Arquitecta y Máster en patrimonio arquitectónico. Está especializada en geometría, levantamiento arquitectónico y análisis estereotómico. Autora de numerosas publicaciones en revistas y congresos nacionales e internacionales.

José Calvo-López

Universidad Politécnica de Cartagena; jose.calvo@upct.es

Catedrático de universidad de la UPCT. Arquitecto. Ha impartido docencia en Geometría descriptiva, Dibujo por ordenador, Fotografía arquitectónica, Historia de la arquitectura y de la construcción y otras materias afines. Ha publicado numerosos trabajos sobre construcción en piedra y ladrillo, proporción arquitectónica y perspectiva axonométrica y lineal.