UNIÓN
REVISTA IBEROAMERICANA DE EDUCACIÓN MATEMÁTICA

ISSN: 1815-0640
Número 69. Diciembre 2023
Páginas 1-8

GeoGebra con Python

Mónica Soler Montaner

Resumen	En este artículo revisamos algunas de las opciones que ofrece GeoGebra para representar curvas y mostramos algunas estrategias para representar estas curvas usando el entorno en línea PyGgb. El entorno PyGgb es una herramienta que permite programar en Python sobre objetos de GeoGebra y mostrar el resultado en la vista gráfica de GeoGebra. **Palabras clave**: Python, GeoGebra, Curvas
Abstract	In this article we review some of the options that GeoGebra offers for representing curves and show some strategies for representing these curves using the PyGgb online environment. The PyGgb environment is a tool that allows you to program in Python on GeoGebra objects and display the result in the GeoGebra graphical view. **Keywords**: Python, GoeGebra, Curves
Resumo	Neste artigo revisamos algumas das opções que o GeoGebra oferece para representar curvas e mostramos algumas estratégias para representar essas curvas utilizando o ambiente online PyGgb. O ambiente PyGgb é uma ferramenta que permite programar em Python em objetos GeoGebra e exibir o resultado na visualizaçao gráfica do GeoGebra. **Palavras-chave:** Phyton, GeoGebra, Curvas

1. Introducción

A principios de 2023, GeoGebra anunció el lanzamiento de PyGgb, un entorno de trabajo en línea en el que podemos programar en Python con objetos y comandos de GeoGebra y visualizar el resultado en la vista gráfica de GeoGebra. Además de las ventanas de programación y representación, PyGgb incluye una tercera ventana que permite visualizar otros datos referentes a la construcción. La versión actual de PyGgb es una versión beta.

El acceso al entorno PyGgb se realiza mediante el enlace https://www.geogebra.org/python .En dicho entorno, además de realizar nuestras propias creaciones podemos acceder a doce programas de muestra que pueden servir de inspiración para el diseño de nuestros propios trabajos.

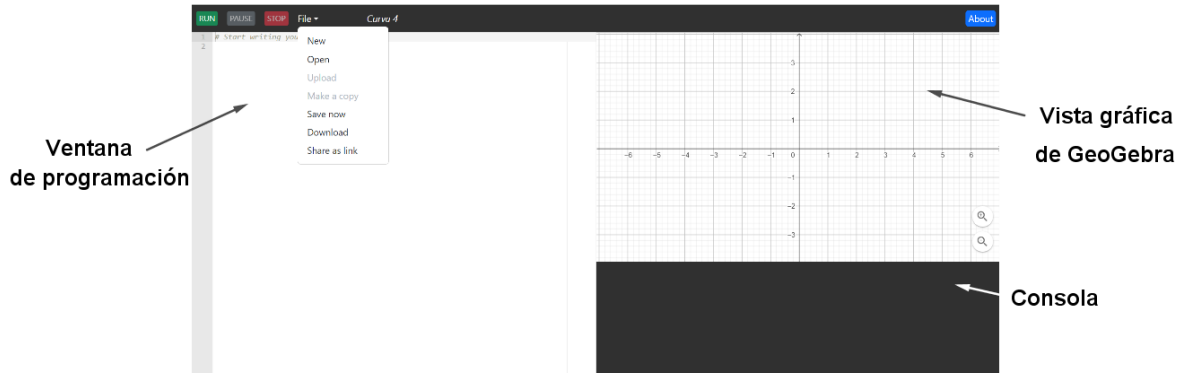


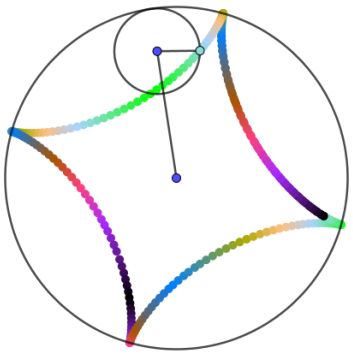
Figura 1. Entorno de trabajo de PyGgb.

En este artículo repasamos algunas de las opciones que nos ofrece GeoGebra para representar curvas y a continuación, mostramos algunas propuestas para hacer representaciones de estas en PyGgb.

2. Representación de curvas con GeoGebra

2.1. A partir de la definición mecánica de la curva

La construcción de curvas con GeoGebra puede hacerse a partir de la definición mecánica de la curva. Por ejemplo, la curva astroide puede obtenerse a partir del rastro de un punto de una circunferencia de radio r/4 cuando esta circunferencia gira sin deslizar dentro de una circunferencia de radio r. En



https://www.geogebra.org/m/bzsswznu se puede consultar la construcción.

Figura 2. Curva astroide.

2.2. Como envolvente de una familia de curvas

Algunas curvas pueden obtenerse como envolventes de familias de curvas. Por ejemplo, la curva astroide puede obtenerse como la envolvente de una familia de segmentos de longitud constante cuyos extremos se mueven a lo largo de dos rectas perpendiculares. En https://www.geogebra.org/m/vqkfz9w2 se puede consultar la construcción.

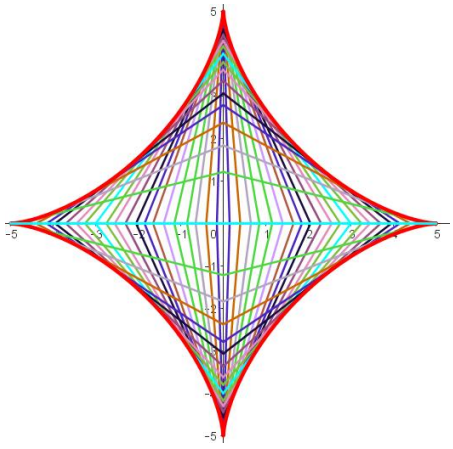


Figura 3. Curva astroide.

2.3. A partir de las ecuaciones de la curva

Las curvas también pueden representarse mediante sus ecuaciones implícitas o paramétricas. El comando de GeoGebra **Curva** permite representar cualquier curva a partir de sus ecuaciones paramétricas. A continuación mostramos la representación de una espiral de Arquímedes mediante el comando de GeoGebra **Curva(0.5 t cos(t),0.5 t sen(t), t, 0, 4π)**. La representación de curvas mediante el comando **Curva** permite aplicar sobre esta cualquier tipo de movimiento, lo que posibilita la creación de imaginativas construcciones. En https://www.geogebra.org/m/zfbqtecf se puede consultar la construcción.

Figura 4. Espiral de Arquímedes.

3. Representación de curvas con PyGgb

3.1. Representación de curvas

Para representar una curva en PyGgb diseñamos un programa en Python que al ejecutarlo nos muestra el resultado en la vista gráfica de GeoGebra. Una propuesta para representar una curva es diseñar un programa que calcule los puntos de esta curva a partir de sus ecuaciones paramétricas. La secuencia a seguir es:

- definir un punto de la curva utilizando las ecuaciones paramétricas de esta;

- mediante un bucle modificar los valores que toma el parámetro.

Como ejemplo representamos la curva conocida como rosa polar. En este caso los puntos de la curva se obtienen a partir de:

punto = Point(4*math.cos(n*t)*math.cos(t), 4*math.cos(n*t)*math.sin(t))

En https://www.geogebra.org/python/index.html?name=Rosa+polar&code=eJx1UEtOxDAM3ecUQSyaDtFQpOkCRO6AxBIhlLaejkV%2BctOphtOTpIVhgxfx58V%2Bz77lr1FT5AthRDfyi5%2BJ936AG4Y2%2BIREtPATWx1PLMSPCb9AHRgb4Mj7mc5aOBnlWRtP9RPjycLsolcvHl0Uh53IjfveT8LtYl3%2FyVMm%2F8MndAUvbBt se puede consultar la construcción.

```
import time
import math
pt_size=4

def curva(n,t,valor):
   punto=Point(4*(math.cos(n*t))*(math.cos(t)),4*(math.cos(n*t))*(math.sin(t)),size=pt_size)
   punto.color=valor

colores=('black','pink','yellow','blue','green','red','orange','purple','brown')
m=0
while m<math.pi:
   for i in range(0,9):
      curva(5,m,colores[i])
      time.sleep(0.05)
      m=m+math.pi/150
```

Figura 5. Código para representar la curva rosa polar.

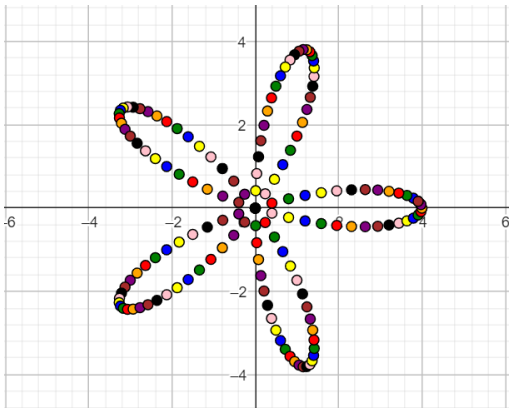


Figura 6. Curva rosa polar.

3.2. Rotaciones de curvas

De nuevo, para representar curvas y las curvas rotadas de esta, diseñamos un programa en Python que al ejecutarlo nos muestra el resultado en la vista gráfica. La propuesta es diseñar un programa que calcule puntos de cada una de las curvas que deseamos representar. La secuencia a seguir es:

- definir un punto de la curva utilizando las ecuaciones paramétricas de esta;
- aplicar una rotación que depende del parámetro ángulo de rotación, al anterior punto;
- definir un bucle que recorra todos los ángulos de rotación deseados y un segundo bucle dentro del anterior que recorra todos los valores del parámetro de la ecuación paramétrica.

En el siguiente ejemplo rotamos una espiral de Bernoulli. Definimos los puntos de la curva a partir de sus ecuaciones paramétricas:

punto = Point(0.5*2t*math.cos(t), 0.5*2**t*math.sin(t))**

Aplicamos una rotación de ángulo m al punto definido anteriormente.

$$\begin{pmatrix} \cos(m) & -sen(m) \\ sen(m) & \cos(m) \end{pmatrix} \begin{pmatrix} 0.5 2^t \cos(t) \\ 0.5 2^t sen(t) \end{pmatrix} = \begin{pmatrix} \cos(m)(0.5 2^t \cos(t)) - sen(m)(0.5 2^t sen(t)) \\ sen(m)(0.5 2^t \cos(t)) + \cos(m)(0.5 2^t sen(t)) \end{pmatrix}$$

Los puntos obtenidos después de aplicar la rotación en función de los parámetros t y m son:

punto = Point (math.cos(m)*(0.5*2t*math.cos(t))-math.sin(m)*(0.5*2**t*math.sin(t)), math.sin(m)*(0.5*2**t*math.cos(t))+math.cos(m)*(0.5*2**t*math.sin(t)))**

En https://www.geogebra.org/python/index.html?name=Espiral+de+Bernuilli&code=eJytUFsOwiAQ%2FOcUq%2F4U1Nqa6IeRO5h4AOOD6CZQSLum8fZCUaPGmvjYn4WdYYadHixpXRLUJRIWezjZYwlbu1MdhsZZjxAadT2bNR0YYwYkZKw%2BoFZgYA5jEYDU4YyBL5JZ0yOB5s2zGxrKWSxolctF6EnS4FtbJYZzkWTpRIyFoCg se puede consultar la construcción.

```
import time
import math

m = 0
while m < 2*math.pi:
   t=0
   while t< math.pi:
      point_1=Point((math.cos(m))*(0.5*2**t*math.cos(t))-math.sin(m)*(0.5*2**t*math.sin(t)),
math.sin(m)*(0.5*2**t*math.cos(t))+math.cos(m)*(0.5*2**t*math.sin(t)))
      point_1.color="yellow"
      t=t+0.1
   time.sleep(0.1)
   m=m+math.pi/6
time.sleep(0.5)
m = 0
while m < 2*math.pi:
   t=0
   while t< math.pi:
      point_1=Point(math.cos(m)*(0.5*2**t*math.cos(t))-math.sin(m)*(0.5*2**t*math.sin(t)), -
(math.sin(m)*(0.5*2**t*math.cos(t))+math.cos(m)*(0.5*2**t*math.sin(t))))
      point_1.color="blue"
      t=t+0.1
   time.sleep(0.1)
   m=m+math.pi/6
```

Figura 7. Código para representar la rotación de las espirales de Bernoulli.

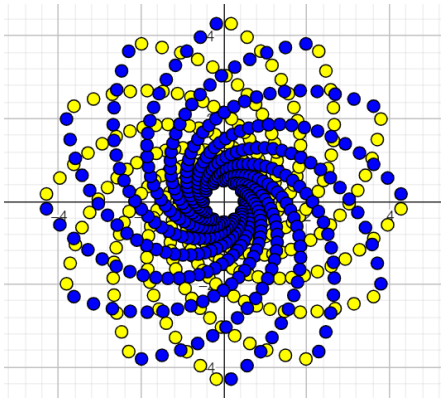


Figura 8. Espirales de Bernoulli.

3.3. Envolventes de familias de curvas

También podemos obtener las curvas como envolventes de familias de curvas. Como muestra representamos la envolvente de una familia de segmentos que pasan por puntos que se mueven con velocidad constante sobre dos semirrectas que tienen un punto en común y cuyo resultado es una parábola. En este programa representamos puntos equidistantes sobre los ejes de coordenadas y después definimos segmentos entre estos puntos. De nuevo, al ejecutar el programa obtenemos la representación en la vista gráfica de GeoGebra.

```
import time
import math
colores=['blue','orange','red']
for i in range (1,50):
    D=Point(0,i,is_visible=False)
    E=Point(50-i,0,is_visible=False)
    k=Segment(D,E)
    r=i%3
    k.color=colores[r]
    time.sleep(0.1)
```

Figura 9. Código para obtener la parábola como envolvente de curvas.

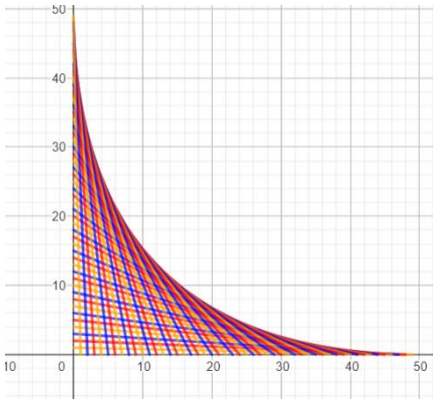


Figura 10. La parábola como envolvente.

4. Otras construcciones

Python puede ser de gran ayuda para automatizar tareas repetitivas en nuestros diseños. A continuación, mostramos un ejemplo de estas características.

En primer lugar, creamos una lista vacía denominada *puntos*. Después representamos un conjunto de puntos sobre una circunferencia de radio 4 unidades mediante un bucle while que actúa sobre el parámetro de las ecuaciones paramétricas de la circunferencia como hemos descrito antes. Cada punto que se calcula, se añade a la lista *puntos*. El programa finaliza con dos bucles for anidados que dibujan los segmentos entre los elementos de la lista *puntos*.

```
import time
import math
puntos=[]
m=0
while m<2*math.pi:
    P=Point(4*math.cos(m),4*math.sin(m))
    P.size=4
    puntos.append(P)
    m=m+math.pi/6
for j in range (0,12):
    for i in range (1,12):
        k=Segment(puntos[i],puntos[j])
        k.color='black'
        time.sleep(0.05)
```

Figura 11. Código de la construcción.

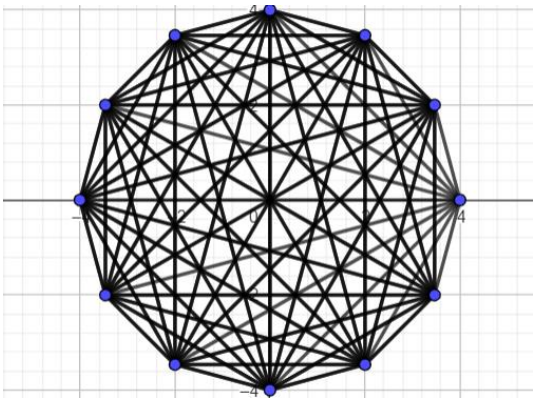


Figura 12. Construcción geométrica.

4. Conclusiones

La nueva herramienta PyGgb presentada por el equipo de GeoGebra amplía las posibilidades que ofrece GeoGebra para mejorar el aprendizaje y la enseñanza de las matemáticas. La integración del entorno de programación Python con GeoGebra facilita el diseño de actividades retadoras y consecuentemente estimula el aprendizaje y ayuda al desarrollo de estrategias para resolver problemas.

Bibliografía

Mathcurve.com. (2023) Último acceso el 15 de diciembre de 2023.

Soler Montaner Mónica: Licenciada en CC Físicas por la Universitat de València y graduada en Matemáticas por la UNED. Profesora de enseñanza secundaria en el IES Josep de Ribera de Xàtiva. Ha participado en distintas actividades relacionadas con GeoGebra. Correo electrónico: msoler40@gmail.com