Elliptic curves

January 26, 2024

1 Elliptic curves

https://www.math.wustl.edu/~acuna/content/Elliptic%20curves.html

An elliptic curve over the complex numbers can be written in the form $V(y^2 - x^3 - ax - b)$, with $a, b \in \mathbb{C}$. And it can be visualized using it's associated Weierstrass P-function $\wp$, this is because it satisfies the differential equation,

$$(y')^2 = 4y^3 - g_2 y - g_3 \iff \left(\frac{y'}{2}\right)^2 = y^3 - \frac{g_2}{4}y - \frac{g_3}{4}$$

Since $g_2, g_3 \in \mathbb{C}$ are numbers such that $a = g_2/4$,$b = g_3/4$, if $x := \wp(z)$, and $y := \wp'(z)/2$ then x, and y satisfy the equation of the elliptic curve $V(y^2 - x^3 - ax - b)$, so they give coordinates on it.

To visualize it we can take the real, and imaginary parts of $\wp$ as the first two coordinates, the real part of $\wp'(z)$ as the third, and the imaginary part of $\wp'$ as the color.

[ ]:
```
def minus_floor(x):
    return x-floor(x)

def elliptic_curve_graph(e,s,r,precision):
    E = EllipticCurve(e)
    p = E.weierstrass_p(prec=precision).truncate(precision)
    pp = p.derivative()/2
    x = lambda u,v: p(u+i*v).real()
    y = lambda u,v: p(u+i*v).imag()
    z = lambda u,v: pp(u+i*v).real()
    w = lambda u,v: pp(u+i*v).imag()

    cf = lambda u,v: minus_floor(w(u,v))
    cm = colormaps.hsv

    E = parametric_plot3d([x,y,z], (-s,s),(-s,s), aspect_ratio=1,color=(cf,cm))
    E = E.add_condition(lambda x, y, z:  x^2+y^2+z^2 < r^2)
    return E
```

1.1 $y^2 = x^3 + x$

[ ]:
```
elliptic_curve_graph([1,0],14/8,2,300).show(frame=false,viewpoint =[[-0.5864,-0.
↪5682,-0.5772],118.44])
```

Graphics3d Object

1.2 $y^2 = x^3 - x + 1$

[ ]:
```
elliptic_curve_graph([-1,1],19/16,2,300).show(frame=false,viewpoint =[[-0.
↪5864,-0.5682,-0.5772],118.44])
```

Graphics3d Object

1.3 $y^2 = x^3 + 4$

Fun fact: this is a curve with Mordel-Weil group $\mathbb{Z}/3\mathbb{Z}$, what that means is it's group of rational points is torsion of order 3.

[ ]:
```
elliptic_curve_graph([0,4],1,4,300).show(frame=false, viewpoint =[[-0.5652,0.
↪5741,0.5924],119.5])
```

Graphics3d Object

1.4 $y^2 = x^3 - 7x + 10$

[ ]:
```
elliptic_curve_graph([-7,10],3/4,8,300).show(frame=false,viewpoint =[[-0.
↪0587,-0.6777,-0.733],172.73])
```

Graphics3d Object

1.5 $y^2 = x^3 - x/4$

[ ]:
```
elliptic_curve_graph([-1/4,0],2,1,300).show(frame=false,viewpoint =[[-0.9909,0.
↪0937,0.0967],88.73])
```

Graphics3d Object

But, we can also use Donu Arapura's idea of interpolating between the real and imaginary parts of to get the following animation. Here, I'm using the same curve he uses in his website,

$$y^2 = x^3 - \frac{x}{4}$$

[ ]:
```
def elliptic_curve_frame(w,s,r,precision,t):
    E = EllipticCurve(w)
    p = E.weierstrass_p(prec=precision).truncate(precision)
    pp = p.derivative()/2
    x = lambda u,v: p(u+i*v).real()
```

```
    y = lambda u,v: p(u+i*v).imag()
    z = lambda u,v,t: cos(t)*pp(u+i*v).real() + sin(t)*pp(u+i*v).imag()

    cf = lambda u,v: minus_floor(sin(t)*pp(u+i*v).real() + cos(t)*pp(u+i*v).
 ↪imag())
    cm = colormaps.hsv

    E = parametric_plot3d([x,y,lambda u,v: z(u,v,t)], (-s,s),(-s,s),␣
 ↪aspect_ratio=1,color=(cf,cm))
    E = E.add_condition(lambda x, y, z:  x^2+y^2+z^2 < r^2)
    return E

build_frame = lambda t: elliptic_curve_frame([-1/4,0],2,1.5,300,t)
frames = [build_frame(t) for t in srange(0,6.28,0.08)+[6.28]]
plot = animate(frames).interactive()
plot = plot.show(delay=5, auto_play=False, projection='orthographic',␣
 ↪frame=false
                 ,viewpoint =[[0.0834,-0.7046,-0.7046],189.53])
```

```
Graphics3d Object
```

[ ]:

```

```