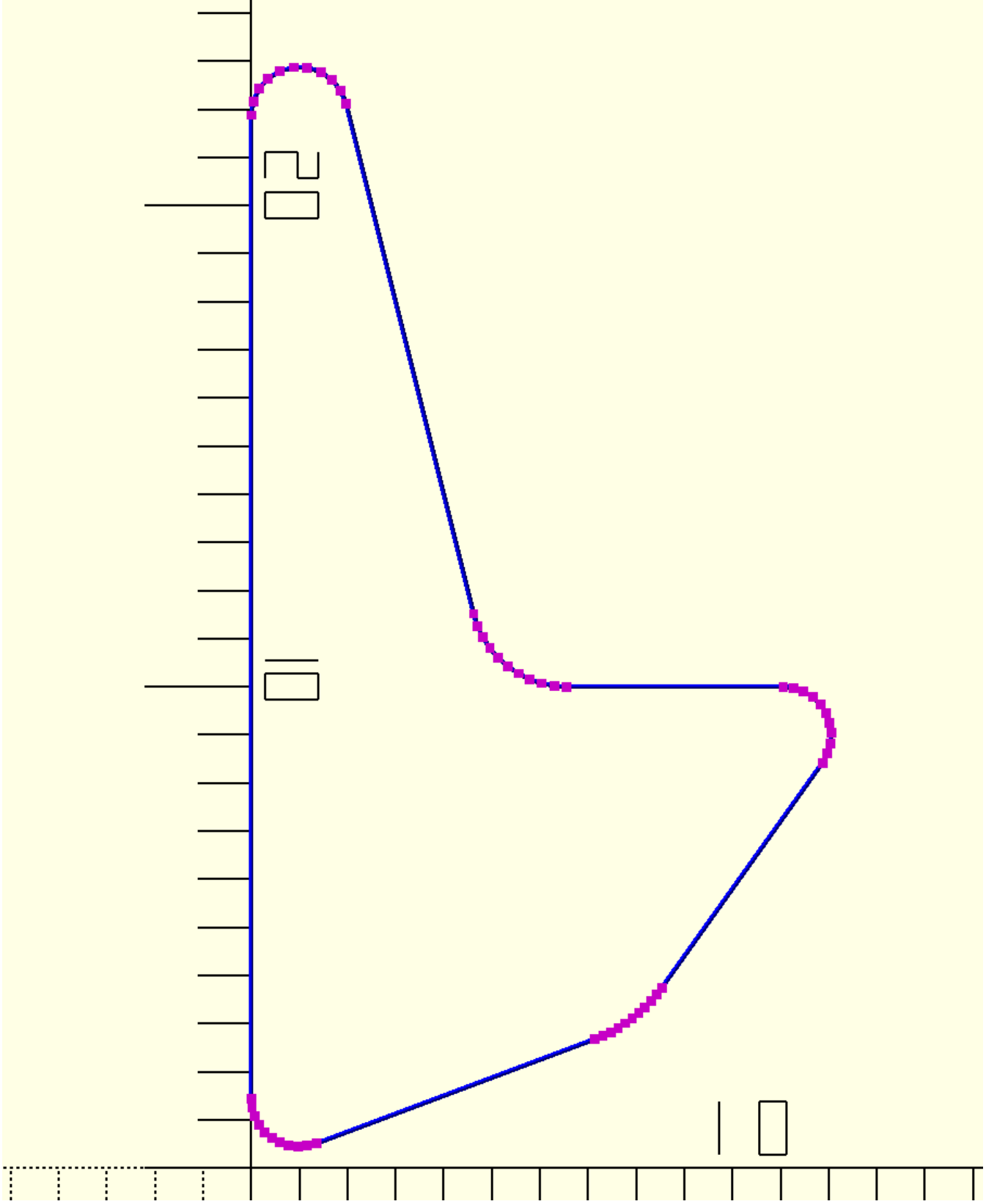
In [15]:
```
from openscad1 import *
```

Steps to compute offset

original section

In [23]:
```
sec=cr(pts1([[0,0,1],[8,3,3],[5,7,1],[-8,0,2],[-5,20,1]]),10)

with open('/users/sanjeevprabhakar/openscad/trial.scad','w+') as f:
    f.write(f'''

    include<dependencies2.scad>
    color("blue")p_line3dc({sec},.1);
    color("magenta")points({sec},.2);

    ''')
```

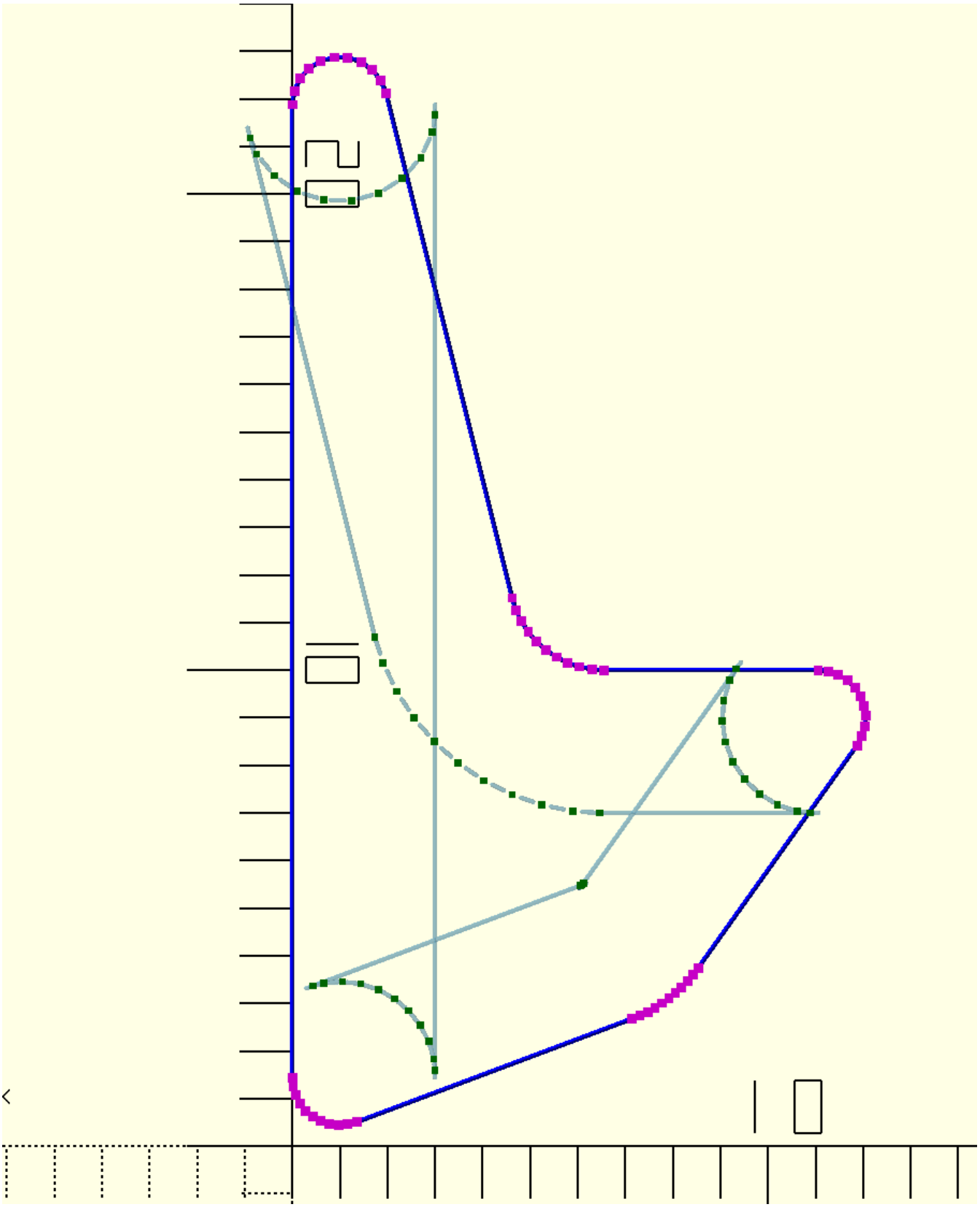


create and offset the line segments and find intersection of adjecent lines

In [22]:

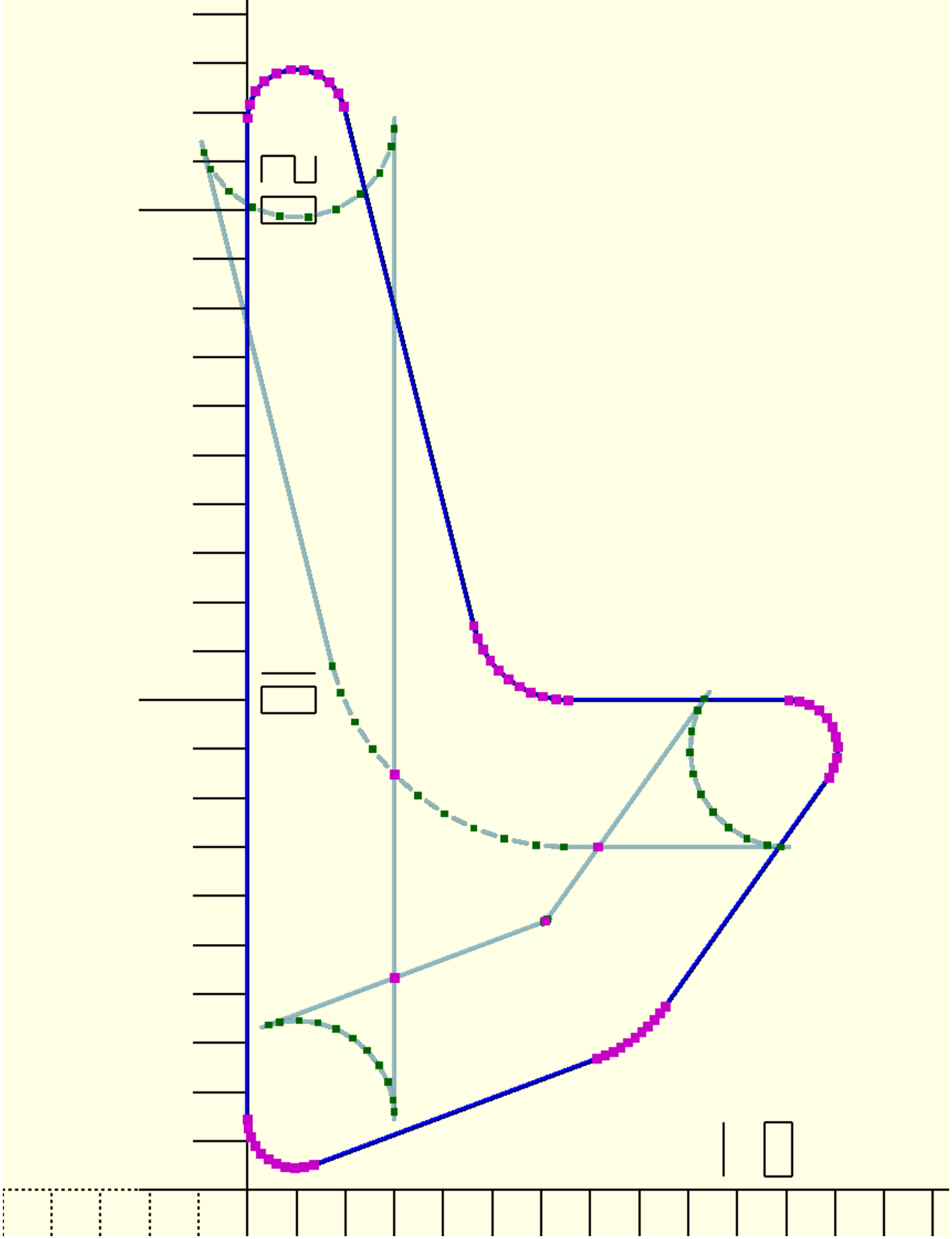
```
sec=cr(pts1([[0,0,1],[8,3,3],[5,7,1],[-8,0,2],[-5,20,1]]),10)

r=-3 # amount of offset required
sec1=offset_segv(sec,r) # create offset line segments
i_p1=intersections(sec1) # this creates intersections even at concave points
```

```
with open('/users/sanjeevprabhakar/openscad/trial.scad','w+') as f:
    f.write(f'''

    include<dependencies2.scad>
    color("blue")p_line3dc({sec},.1);
    color("magenta")points({sec},.2);
    color([.2,.6,.8,.3])for(p={sec1})p_line3d(p,.1);
    color("green")points({i_p1},.15);

    ''')
```



find global intersections

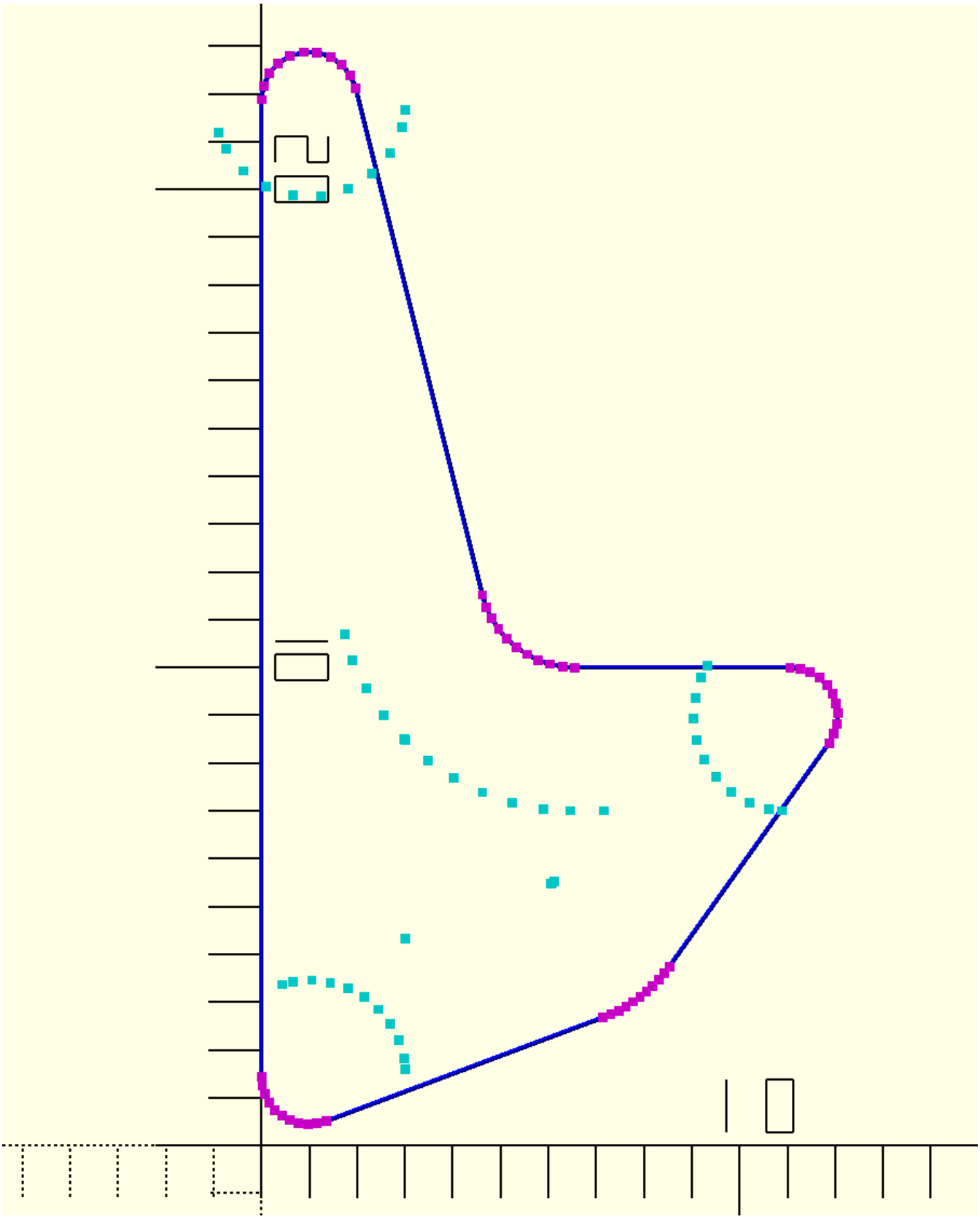
In [26]:
```
sec=cr(pts1([[0,0,1],[8,3,3],[5,7,1],[-8,0,2],[-5,20,1]]),10)

r=-3 # amount of offset required
sec1=offset_segv(sec,r) # create offset line segments
i_p1=intersections(sec1) # this creates intersections even at concave points
g_i=s_int1(seg(i_p1)) # global intersection from the intersection points calculated abov
with open('/users/sanjeevprabhakar/openscad/trial.scad','w+') as f:
    f.write(f'''

    include<dependencies2.scad>
    color("blue")p_line3dc({sec},.1);
    color("magenta")points({sec},.2);
    color([.2,.6,.8,.3])for(p={sec1})p_line3d(p,.1);
    color("green")points({i_p1},.15);
    color("magenta")points({g_i},.2);

    ''')
```



add intersections and global intersections together

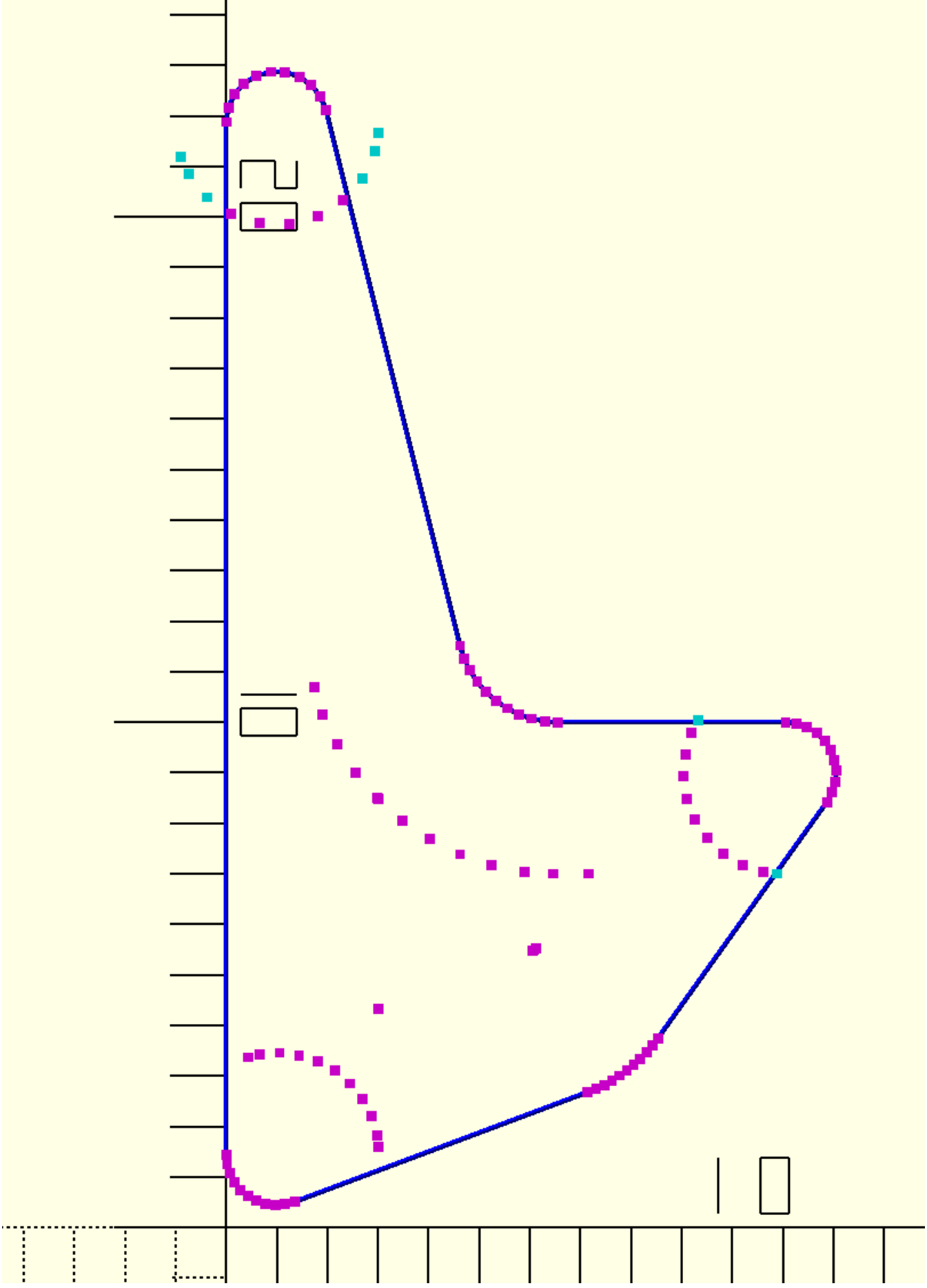
In [27]:
```
sec=cr(pts1([[0,0,1],[8,3,3],[5,7,1],[-8,0,2],[-5,20,1]]),10)

r=-3 # amount of offset required
sec1=offset_segv(sec,r) # create offset line segments
i_p1=intersections(sec1) # this creates intersections even at concave points
```

```
g_i=s_int1(seg(i_p1)) # global intersection from the intersection points calculated abov
sec2=i_p1+g_i

with open('/users/sanjeevprabhakar/openscad/trial.scad','w+') as f:
    f.write(f'''

    include<dependencies2.scad>
    color("blue")p_line3dc({sec},.1);
    color("magenta")points({sec},.2);
    color("cyan")points({sec2},.2);

    ''')
```



identify all the points which are inside the original section and rest of the points can be discarded

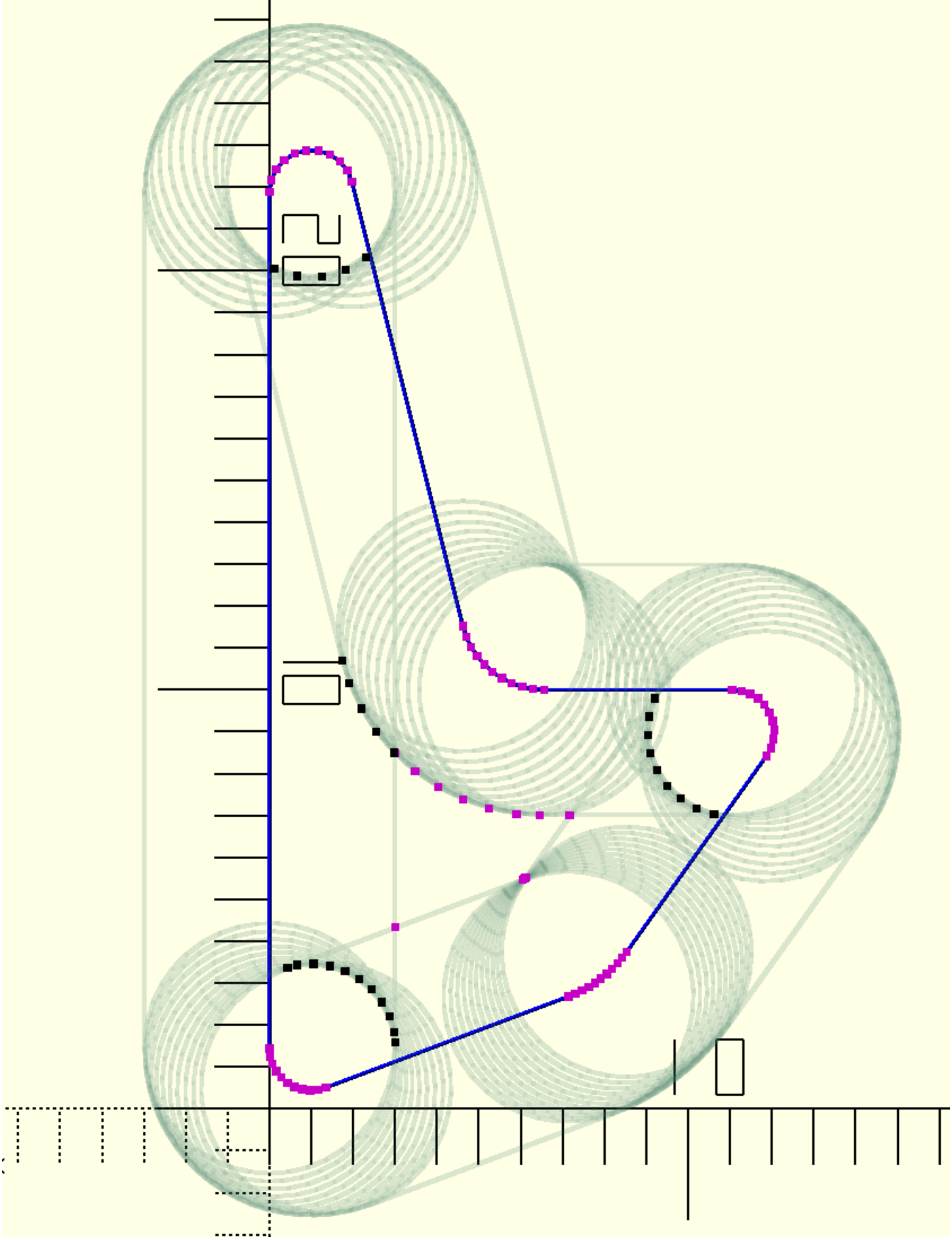
In [28]:
```
sec=cr(pts1([[0,0,1],[8,3,3],[5,7,1],[-8,0,2],[-5,20,1]]),10)

r=-3 # amount of offset required
sec1=offset_segv(sec,r) # create offset line segments
i_p1=intersections(sec1) # this creates intersections even at concave points
g_i=s_int1(seg(i_p1)) # global intersection from the intersection points calculated abov
sec2=i_p1+g_i
p0=pies1(sec,sec2) # only these points are required to be processed further

with open('/users/sanjeevprabhakar/openscad/trial.scad','w+') as f:
    f.write(f'''

    include<dependencies2.scad>
    color("blue")p_line3dc({sec},.1);
    color("magenta")points({sec},.2);
    color("cyan")points({sec2},.2);
    color("magenta")points({p0},.2);

    ''')
```



Draw rounded sections around each line segment of original section

In [35]:

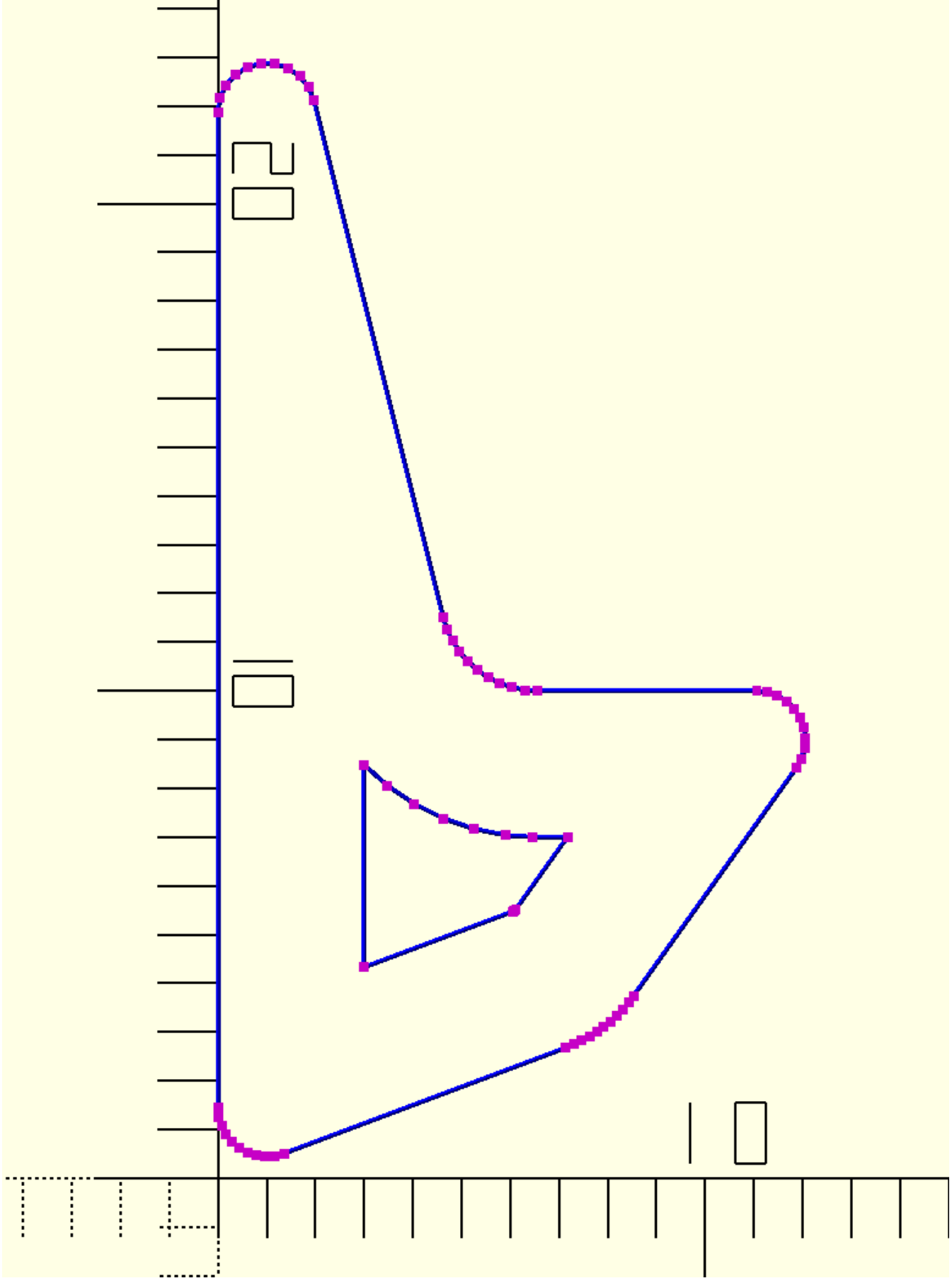
```
sec=cr(pts1([[0,0,1],[8,3,3],[5,7,1],[-8,0,2],[-5,20,1]]),10)

r=-3 # amount of offset required
sec1=offset_segv(sec,r) # create offset line segments
i_p1=intersections(sec1) # this creates intersections even at concave points
g_i=s_int1(seg(i_p1)) # global intersection from the intersection points calculated abov
sec2=i_p1+g_i
p0=pies1(sec,sec2) # only these points are required to be processed further
rounded_sections=cs1(sec,abs(r)-.01)
p1=[pies1(p,p0) for p in rounded_sections if pies1(p,p0)!=[]]
p1=concatenate(p1)
p1=remove_extra_points(p1)
with open('/users/sanjeevprabhakar/openscad/trial.scad','w+') as f:
    f.write(f'''

    include<dependencies2.scad>
    color("blue")p_line3dc({sec},.1);
    color("magenta")points({sec},.2);
    //color("cyan")points({sec2},.2);
    color("magenta")points({p0},.2);
    color([.3,.6,.5,.1])for(p={rounded_sections})p_line3dc(p,.1,rec=1);
    color("black")points({p1},.2);

    ''')
```



remaining points needs to be ordered based on the original points

In [37]:
```
sec=cr(pts1([[0,0,1],[8,3,3],[5,7,1],[-8,0,2],[-5,20,1]]),10)

r=-3 # amount of offset required
```

```
sec1=offset_segv(sec,r) # create offset line segments
i_p1=intersections(sec1) # this creates intersections even at concave points
g_i=s_int1(seg(i_p1)) # global intersection from the intersection points calculated abov
sec2=i_p1+g_i
p0=pies1(sec,sec2) # only these points are required to be processed further
rounded_sections=cs1(sec,abs(r)-.01)
p1=[pies1(p,p0) for p in rounded_sections if pies1(p,p0)!=[]]
p1=concatenate(p1)
p1=remove_extra_points(p1)
p2=exclude_points(p0,p1)
p2=sort_points(sec,p2)
with open('/users/sanjeevprabhakar/openscad/trial.scad','w+') as f:
    f.write(f'''

    include<dependencies2.scad>
    color("blue")p_line3dc({sec},.1);
    color("magenta")points({sec},.2);
    //color("cyan")points({sec2},.2);
    //color("magenta")points({p0},.2);
    //color([.3,.6,.5,.1])for(p={rounded_sections})p_line3dc(p,.1,rec=1);
    //color("black")points({p1},.2);
    color("magenta")points({p2},.2);
    color("blue")p_line3dc({p2},.1);

    ''')
```

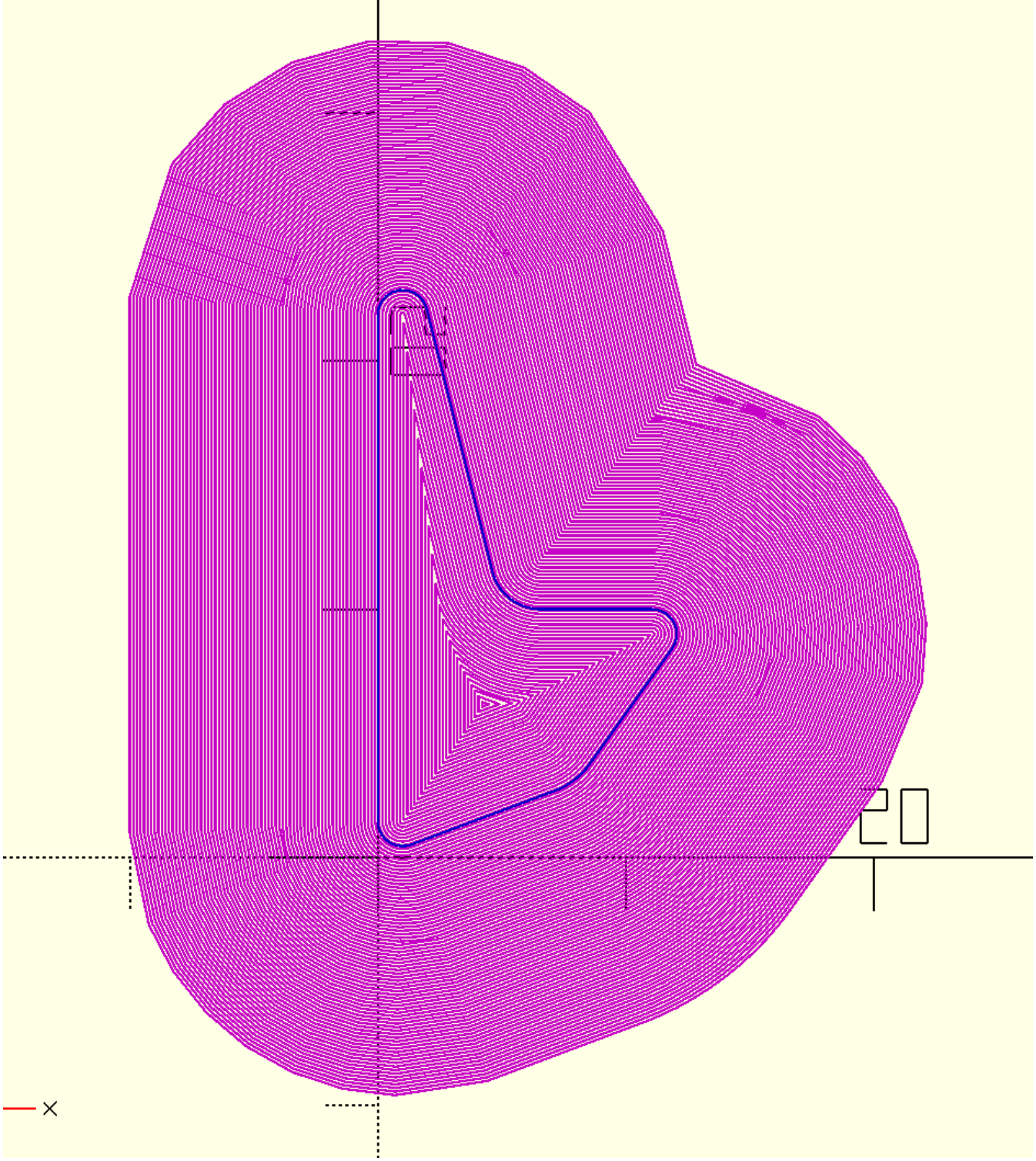


with this method a 100 offsets takes around 4.5 s

In [40]:
```
# example of function offset(sec,r)

t0=time.time()
```

```
sec=cr(pts1([[0,0,1],[8,3,3],[5,7,1],[-8,0,2],[-5,20,1]]),10) # 10 segments at each corn
# if the corner radiuses are increased to 30 it takes around 23 sec to calculate 100 off

os=linspace(-4.2,10,100)
sec1=[offset(sec,i) for i in os]
with open('/users/sanjeevprabhakar/openscad/trial.scad','w+') as f:
    f.write(f'''
include<dependencies2.scad>

color("magenta")for(p={sec1})p_line(p,.1);
color("blue")p_line({sec},.1);

''')
t1=time.time()
t1-t0
```

Out[40]:
```
4.391197204589844
```


20
X