

Offset Curves along a parametric curve

- using Matplotlib, NumPy and scikit-vectors

Copyright (c) 2019 Tor Olav Kristensen, <http://subcube.com> (<http://subcube.com>).

<https://github.com/t-o-k/scikit-vectors> (<https://github.com/t-o-k/scikit-vectors>).

Use of this source code is governed by a BSD-license that can be found in the LICENSE file.

```
In [1]: 1 url = 'https://github.com/t-o-k/scikit-vectors_examples/'
```

```
In [2]: 1 # This example has been tested with NumPy v1.13.3, Matplotlib v2.1.1 and Jupyter v4.4.0.
```

```
In [3]: 1 # Uncomment one of these to get a Matplotlib backend with interactive plots
2
3 # %matplotlib auto
4 # %matplotlib notebook
```

```
In [4]: 1 import matplotlib.pyplot as plt
2 from matplotlib.patches import Polygon
3 from matplotlib.collections import PatchCollection
4 import numpy as np
5
6 from skvectors import create_class_Cartesian_2D_Vector
```

```
In [5]: 1 # Size and resolution for Matplotlib figures
2
3 figure_size = (8, 8)
4 figure_dpi = 100
```

```
In [6]: 1 # Trefoil knot in 2D
2
3 def f_x(t):
4     r = np.sqrt(2 + np.sqrt(3))
5
6     return r * np.cos(2 * t - 3 / 2 * np.pi) - np.sin(t)
7
8
9
10 def f_y(t):
11     r = np.sqrt(2 + np.sqrt(3))
12
13     return r * np.sin(2 * t - 3 / 2 * np.pi) - np.cos(t)
14
```

```
In [7]: 1 no_of_points_along_curve = 3 * 2**8 + 1
```

```
In [8]: 1 # Necessary NumPy functions
2
3 np_functions = \
4     {
5         'not': np.logical_not,
6         'and': np.logical_and,
7         'or': np.logical_or,
8         'all': np.all,
9         'any': np.any,
10        'min': np.minimum,
11        'max': np.maximum,
12        'abs': np.absolute,
13        'trunc': np.trunc,
14        'ceil': np.ceil,
15        'copysign': np.copysign,
16        'log10': np.log10,
17        'cos': np.cos,
18        'sin': np.sin,
19        'atan2': np.arctan2,
20        'pi': np.pi
21    }
```

In [9]:

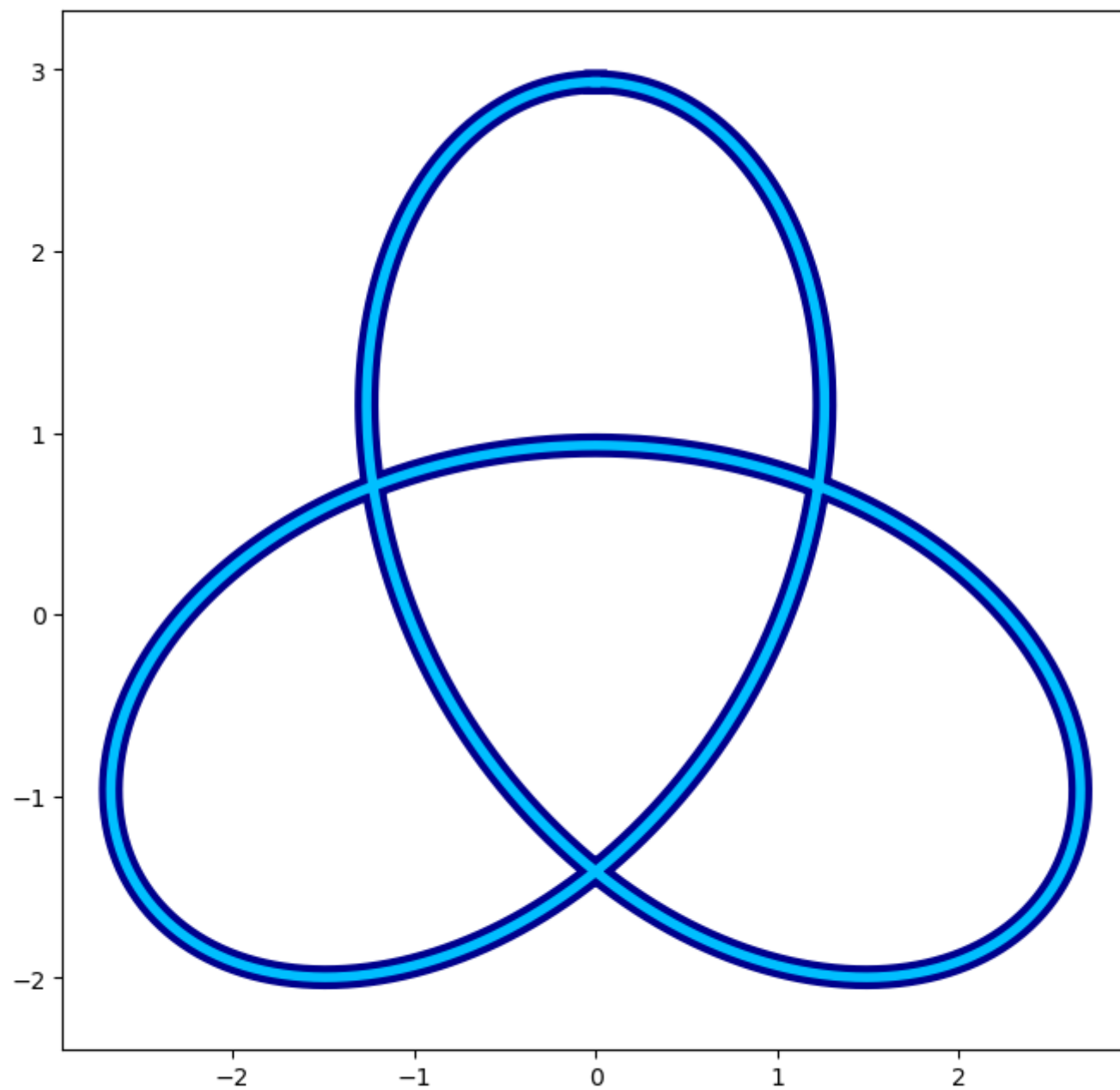
```
1  # Create a vector class that can hold all the points along the curve
2
3  NP2 = \
4      create_class_Cartesian_2D_Vector(
5          name = 'NP2',
6          component_names = 'xy',
7          brackets = '<>',
8          sep = ', ',
9          cnull = np.zeros(no_of_points_along_curve),
10         cunit = np.ones(no_of_points_along_curve),
11         functions = np_functions
12     )
```

In [10]:

```
1  # Calculate the points along the curve
2
3  angles_along_curve = np.linspace(0, 2*np.pi, no_of_points_along_curve, endpoint=True) + np.pi
4
5  p_o = \
6      NP2(
7          x = f_x(angles_along_curve),
8          y = f_y(angles_along_curve)
9      )
```

In [11]:

```
1  # Show the curve by drawing a line above a thicker line
2
3  fig, ax = plt.subplots(figsize=figure_size, dpi=figure_dpi)
4  fig.text(0.30, 0.05, url)
5  ax.plot(p_o.x, p_o.y, color='darkblue', linewidth=10)
6  ax.plot(p_o.x, p_o.y, color='deepskyblue', linewidth=4)
7  ax.axis('equal')
8  plt.show()
```



https://github.com/t-o-k/scikit-vectors_examples/

```
In [12]: 1 # Numerical approximation of the first derivative of a univariate function
2
3 def first_derivative(fn, h=1e-4):
4     h2 = 2 * h
5
6
7     def d1_fn(t):
8
9         return (fn(t + h) - fn(t - h)) / h2
10
11
12
13     return d1_fn
```

```
In [13]: 1 # Create derivative functions for the curve
2
3 d1_f_x = first_derivative(f_x)
4 d1_f_y = first_derivative(f_y)
```

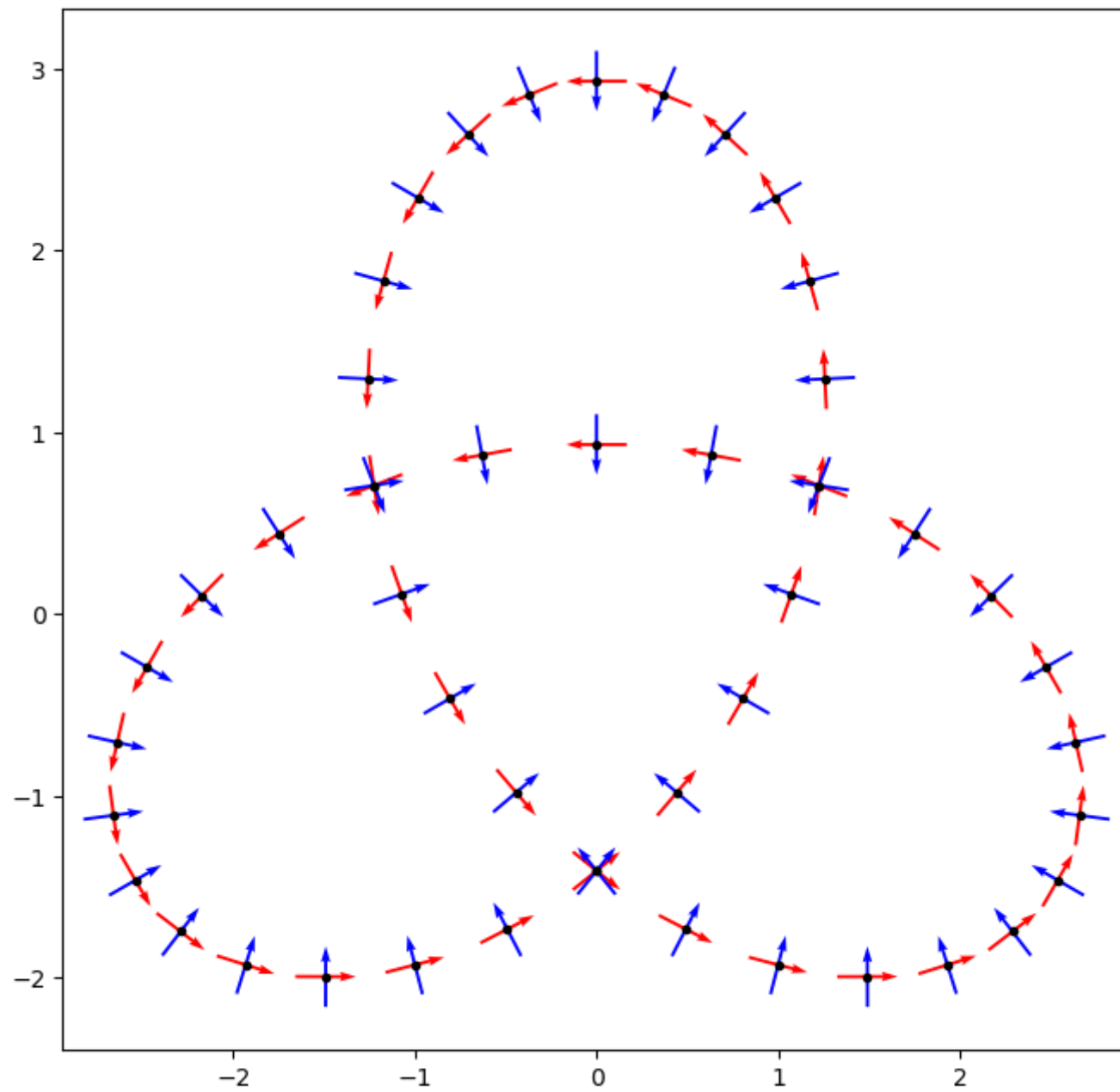
```
In [14]: 1 # Calculate vectors from the first derivatives at the points along the curve
2
3 v_d1 = \
4     NP2(
5         x = d1_f_x(angles_along_curve),
6         y = d1_f_y(angles_along_curve)
7     )
```

```
In [15]: 1 # Calculate tangent vectors at the points along the curve
2
3 v_t = v_d1.normalize()
```

```
In [16]: 1 # Calculate normal vectors at the points along the curve
2
3 v_n = v_t.perp()
```

In [17]:

```
1  # Show some of the tangent vectors (red) and the normal vectors (blue) along the curve
2
3  s = 16 # stride
4
5  sl = slice(None, -1, s)
6
7  fig, ax = plt.subplots(figsize=figure_size, dpi=figure_dpi)
8  fig.text(0.30, 0.05, url)
9  ax.quiver(
10     p_o.x[sl], p_o.y[sl],
11     v_t.x[sl], v_t.y[sl],
12     width = 0.003,
13     color = 'red',
14     scale = 3,
15     scale_units = 'xy',
16     pivot = 'middle'
17 )
18 ax.quiver(
19     p_o.x[sl], p_o.y[sl],
20     v_n.x[sl], v_n.y[sl],
21     width = 0.003,
22     color = 'blue',
23     scale = 3,
24     scale_units = 'xy',
25     pivot = 'middle'
26 )
27 ax.scatter(
28     p_o.x[sl], p_o.y[sl],
29     color = 'black',
30     marker = '.'
31 )
32 ax.axis('equal')
33 plt.show()
```

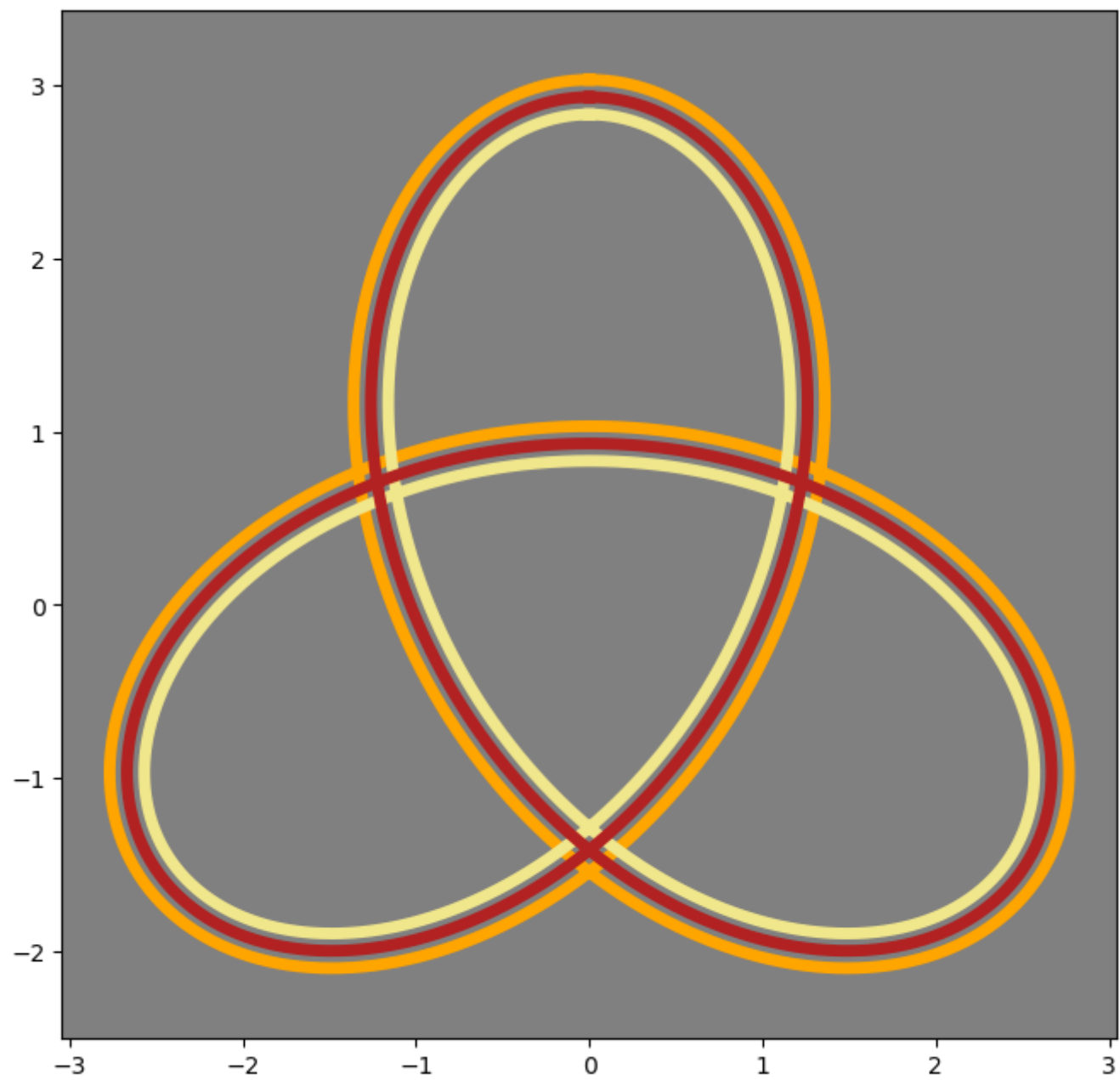


https://github.com/t-o-k/scikit-vectors_examples/


```
In [18]: 1 # Calculate points for two offset curves
          2
          3 d = 0.1
          4
          5 p_dm = p_o - d * v_n
          6 p_dp = p_o + d * v_n
```

In [19]:

```
1  # Show the curve together with the two offset curves
2
3  lw = 5
4
5  fig, ax = plt.subplots(figsize=figure_size, dpi=figure_dpi)
6  fig.text(0.30, 0.05, url)
7  ax.plot(*p_dm, c='orange', linewidth=lw)
8  ax.plot(*p_dp, c='khaki', linewidth=lw)
9  ax.plot(*p_o, c='firebrick', linewidth=lw)
10 ax.set_facecolor('gray')
11 ax.axis('equal')
12 plt.show()
```



https://github.com/t-o-k/scikit-vectors_examples/

```

In [20]: 1 # Prepare for plotting with quad patches (between pairs of offset curves) instead of lines
2
3 def Patches(p_e, p_f, color='black'):
4     no_of_points = len(p_e.cnull)
5
6     return \
7     [
8         Polygon(
9             [
10                [ p_e.x[j], p_e.y[j] ],
11                [ p_f.x[j], p_f.y[j] ],
12                [ p_f.x[k], p_f.y[k] ],
13                [ p_e.x[k], p_e.y[k] ]
14            ],
15            closed = True,
16            color = color
17        )
18        for j, k in zip(range(0, no_of_points-1), range(1, no_of_points))
19    ]
20

```

```

In [21]: 1 # Calculate points for the offset curves
2
3 d = 0.04
4 p_cm = p_o - d * v_n
5 p_cp = p_o + d * v_n
6
7 a = 2 * d
8 p_am = p_o - a * v_n
9 p_ap = p_o + a * v_n
10
11 b = 4 * d
12 p_bm = p_o - b * v_n
13 p_bp = p_o + b * v_n

```

```

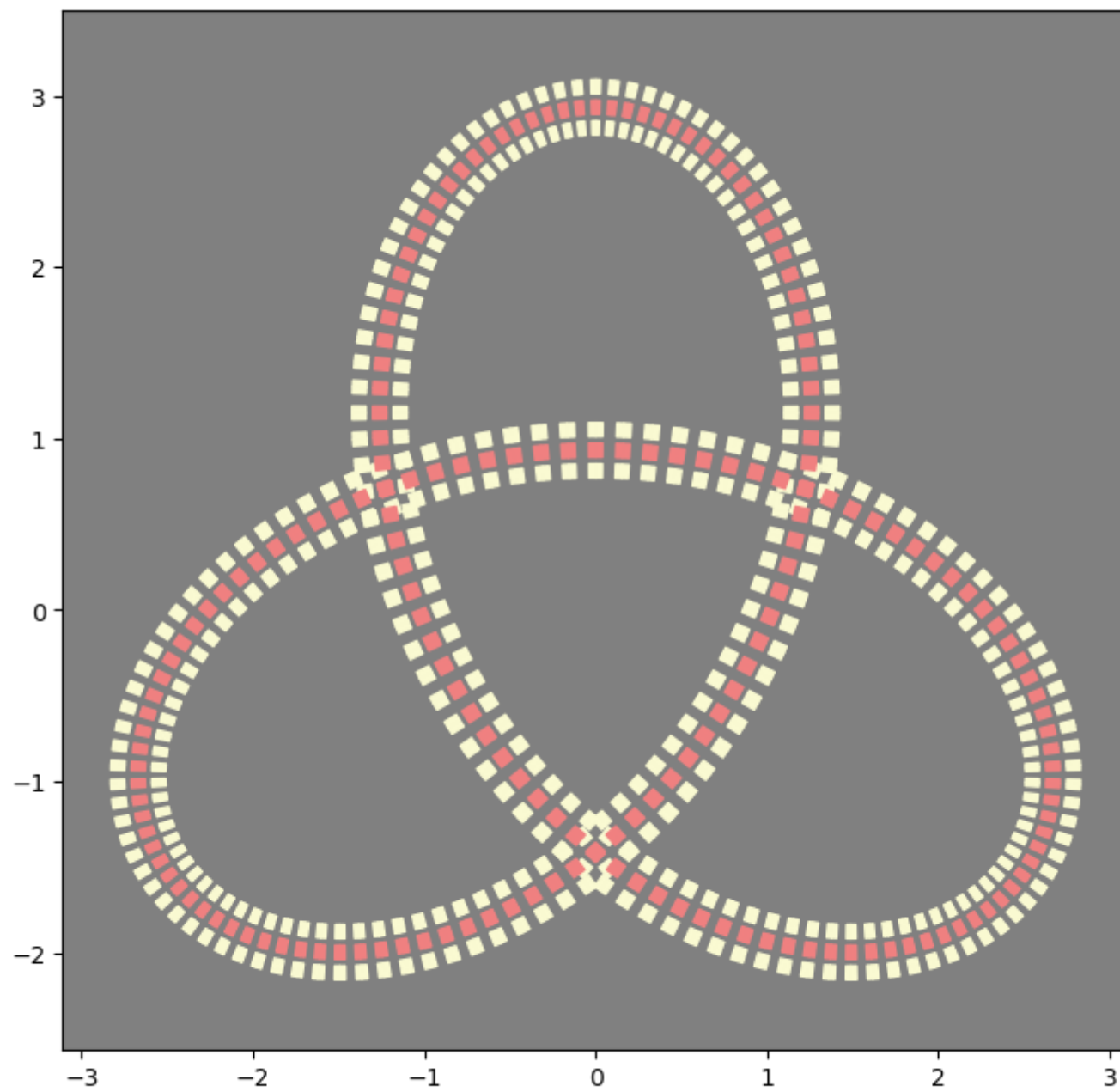
In [22]: 1 # Create the patches
2
3 patches_outer = Patches(p_am, p_bm)
4 patches_inner = Patches(p_ap, p_bp)
5 patches_center = Patches(p_cm, p_cp)

```

In [23]:

```
1  # Show every second of the patches along the curves
2
3  s = 2
4
5  sl1 = slice(None, 1 - s * 2, s * 2)
6  sl2 = slice(s * 2 - 1, None, s * 2)
7
8  fig, ax = plt.subplots(figsize=figure_size, dpi=figure_dpi)
9  fig.text(0.30, 0.05, url)
10 ax.add_collection(
11     PatchCollection(
12         patches_inner[sl1],
13         # match_original = True,
14         color = 'lightgoldenrodyellow'
15     )
16 )
17 ax.add_collection(
18     PatchCollection(
19         patches_inner[sl2],
20         # match_original = True,
21         color = 'lightgoldenrodyellow'
22     )
23 )
24 ax.add_collection(
25     PatchCollection(
26         patches_outer[sl1],
27         # match_original = True,
28         color = 'lightgoldenrodyellow'
29     )
30 )
31 ax.add_collection(
32     PatchCollection(
33         patches_outer[sl2],
34         # match_original = True,
35         color = 'lightgoldenrodyellow'
36     )
37 )
38 ax.add_collection(
39     PatchCollection(
40         patches_center[sl1],
41         # match_original = True,
42         color = 'lightcoral'
43     )
44 )
```

```
44 )
45 ax.add_collection(
46     PatchCollection(
47         patches_center[sl2],
48         # match_original = True,
49         color = 'lightcoral'
50     )
51 )
52 ax.set_facecolor('gray')
53 ax.axis('equal')
54 plt.show()
```



https://github.com/t-o-k/scikit-vectors_examples/

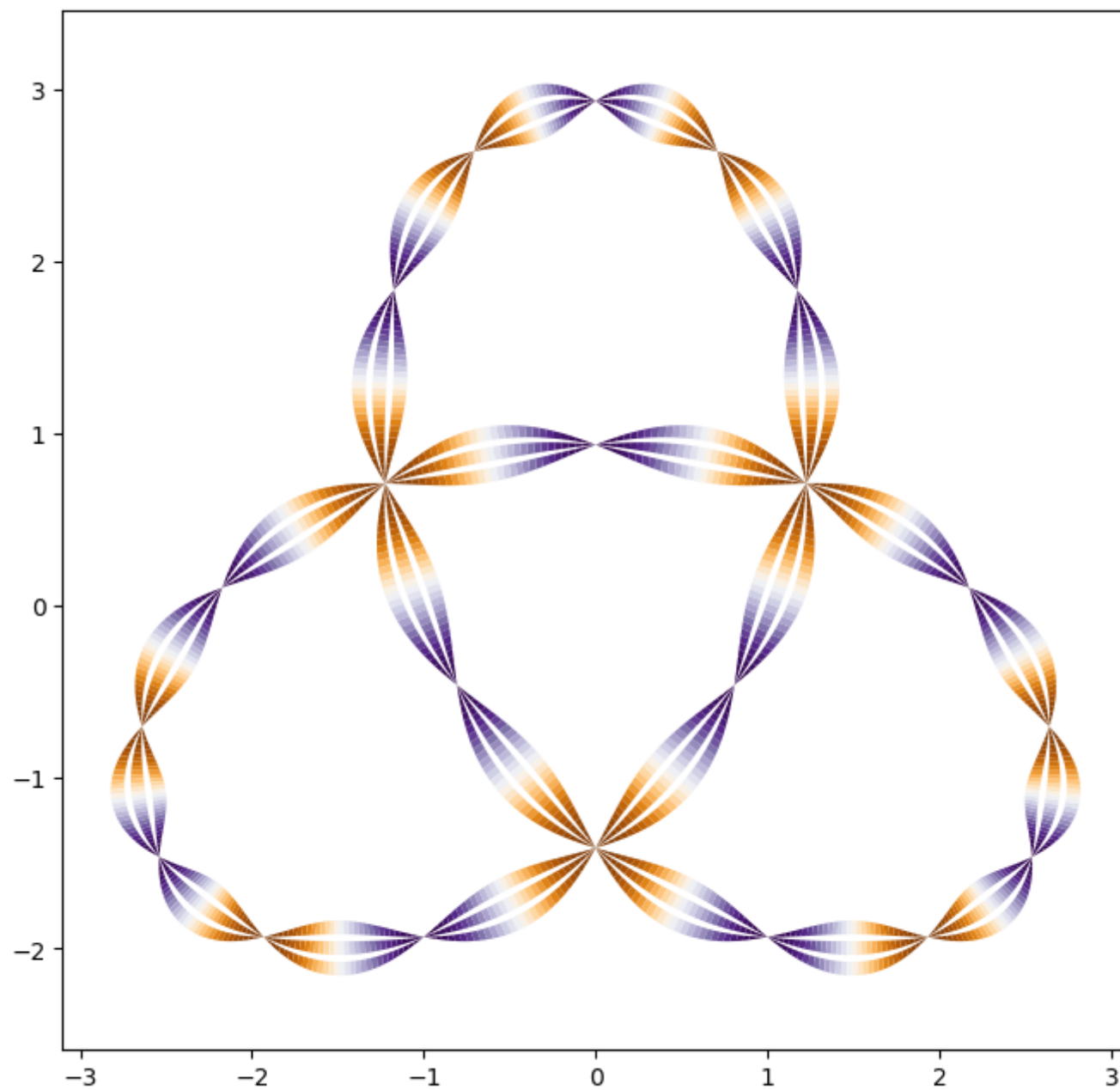
```
In [24]: 1 # Calculate points for more offset curves
2
3 # NB: The order of the operands in the first multiplication matters here
4 v_n_w = v_n * np.sin(12 * angles_along_curve)
5
6 d = 0.04
7 p_cm = p_o - d * v_n_w
8 p_cp = p_o + d * v_n_w
9
10 a = 2 * d
11 p_am = p_o - a * v_n_w
12 p_ap = p_o + a * v_n_w
13
14 b = 4 * d
15 p_bm = p_o - b * v_n_w
16 p_bp = p_o + b * v_n_w
```

```
In [25]: 1 # Create more patches
2
3 patches_outer_w = Patches(p_am, p_bm)
4 patches_inner_w = Patches(p_ap, p_bp)
5 patches_center_w = Patches(p_cm, p_cp)
```

```
In [26]: 1 # Prepare values for choosing colors from a color map
2
3 phase_shift = np.pi / (no_of_points_along_curve - 1)
4 angles_for_color = 12 * (angles_along_curve + phase_shift)
5 values_for_color = (np.cos(angles_for_color) + 1) / 2
```


In [27]:

```
1  # Show the curves with colors cycling
2
3  fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
4  fig.text(0.30, 0.05, url)
5  ax = fig.add_subplot(1, 1, 1)
6  ax.add_collection(
7      PatchCollection(
8          patches_inner_w,
9          array = values_for_color,
10         cmap = plt.cm.PuOr
11     )
12 )
13 ax.add_collection(
14     PatchCollection(
15         patches_outer_w,
16         array = values_for_color,
17         cmap = plt.cm.PuOr
18     )
19 )
20 ax.add_collection(
21     PatchCollection(
22         patches_center_w,
23         array = values_for_color,
24         cmap = plt.cm.PuOr
25     )
26 )
27 # ax.set_facecolor('grey')
28 ax.axis('equal')
29 plt.show()
```

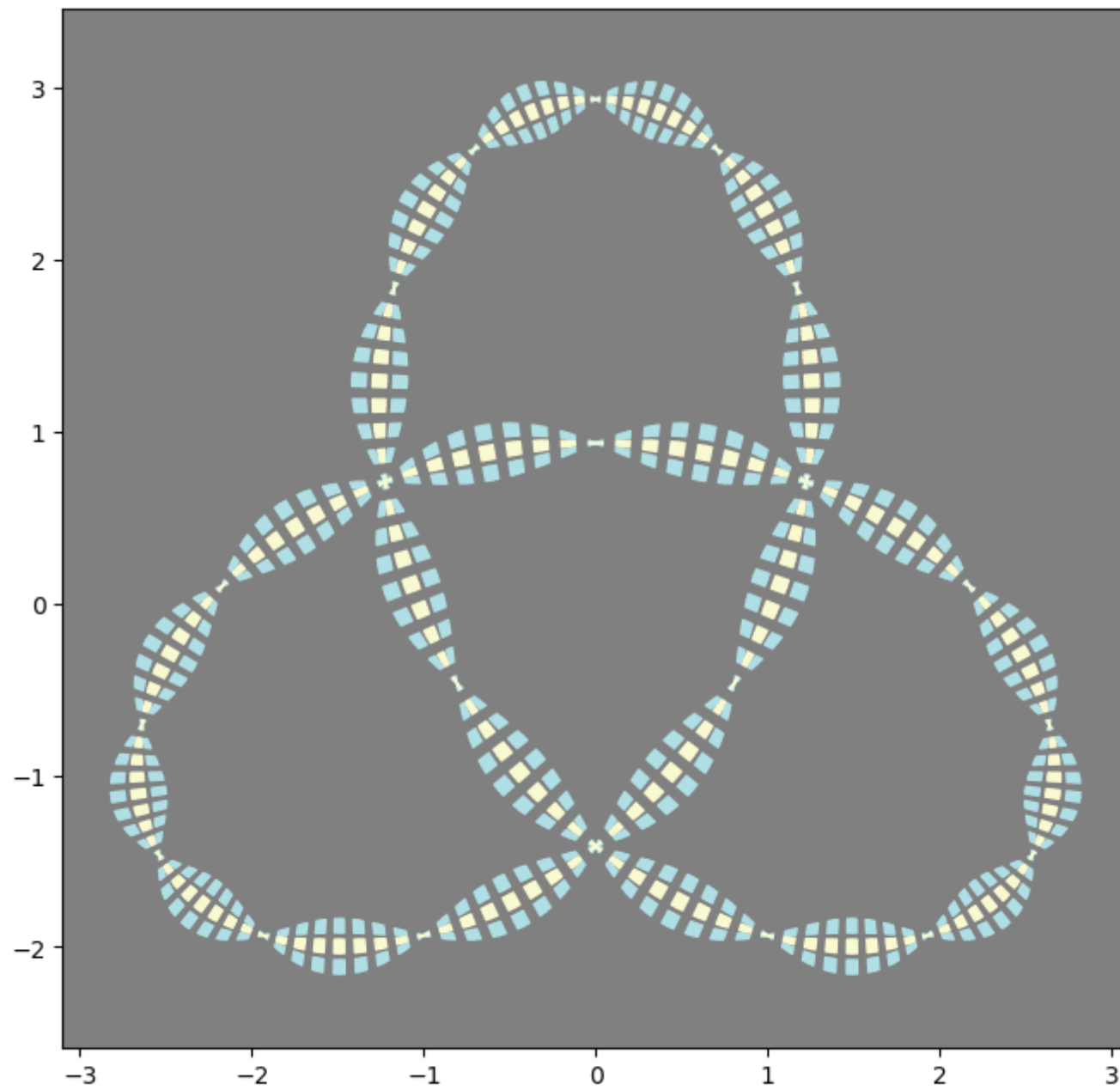


https://github.com/t-o-k/scikit-vectors_examples/

In [28]:

```
1  # Show every second of the patches along the curves
2
3  s = 2
4
5  sl1 = slice(None, 1 - s * 2, s * 2)
6  sl2 = slice(s * 2 - 1, None, s * 2)
7
8  fig, ax = plt.subplots(figsize=figure_size, dpi=figure_dpi)
9  fig.text(0.30, 0.05, url)
10 ax.add_collection(
11     PatchCollection(
12         patches_inner_w[sl1],
13         color = 'powderblue'
14     )
15 )
16 ax.add_collection(
17     PatchCollection(
18         patches_inner_w[sl2],
19         color = 'powderblue'
20     )
21 )
22 ax.add_collection(
23     PatchCollection(
24         patches_outer_w[sl1],
25         color = 'powderblue'
26     )
27 )
28 ax.add_collection(
29     PatchCollection(
30         patches_outer_w[sl2],
31         color = 'powderblue'
32     )
33 )
34 ax.add_collection(
35     PatchCollection(
36         patches_center_w[sl1],
37         color = 'lightgoldenrodyellow'
38     )
39 )
40 ax.add_collection(
41     PatchCollection(
42         patches_center_w[sl2],
43         color = 'lightgoldenrodyellow'
```

```
44     )
45 )
46 ax.set_facecolor('grey')
47 ax.axis('equal')
48 plt.show()
```



https://github.com/t-o-k/scikit-vectors_examples/

In []: 1

