Creating a tube along a trefoil knot

- using Matplotlib, NumPy and scikit-vectors

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

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

In [2]:
```
# This example has been tested with NumPy v1.15.3, Matplotlib v2.1.1 and Jupyter v4.4.0
```

In [3]:
```
# Uncomment one of these to get a Matplotlib backend with interactive plots

# %matplotlib auto
# %matplotlib notebook
```

In [4]:
```
# Get the necessary libraries

import matplotlib.pyplot as plt
import matplotlib.tri as mtri
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
import numpy as np

from skvectors import create_class_Cartesian_3D_Vector
```

In [5]:
```
# Size and resolution for Matplotlib figures

figure_size = (8, 6)
figure_dpi = 100
```

In [6]:

```
# The functions for the trefoil knot curve

def f_x(t):

    return +np.cos(t) + 2 * np.cos(2 * t)


def f_y(t):

    return +np.sin(t) - 2 * np.sin(2 * t)


def f_z(t):

    return +2 * np.sin(3 * t)
```

In [7]:

```
# The first derivatives of the functions for the curve

def d1_f_x(t):

    return -np.sin(t) - 4 * np.sin(2 * t)


def d1_f_y(t):

    return +np.cos(t) - 4 * np.cos(2 * t)


def d1_f_z(t):

    return +6 * np.cos(3 * t)
```

In [8]:
```
# The second derivatives of the functions for the curve

def d2_f_x(t):

    return -np.cos(t) - 8 * np.cos(2 * t)


def d2_f_y(t):

    return -np.sin(t) + 8 * np.sin(2 * t)


def d2_f_z(t):

    return -18 * np.sin(3 * t)
```

In [9]:
```
# Resolutions for plot

nr_of_points_along_curve = 3 * 2**5 + 1
nr_of_points_across_curve = 3 * 2**2 + 1
```

In [10]:

```
# Necessary NumPy functions

np_functions = \
    {
        'not': np.logical_not,
        'and': np.logical_and,
        'or': np.logical_or,
        'all': np.all,
        'any': np.any,
        'min': np.minimum,
        'max': np.maximum,
        'abs': np.absolute,
        'trunc': np.trunc,
        'ceil': np.ceil,
        'copysign': np.copysign,
        'log10': np.log10,
        'cos': np.cos,
        'sin': np.sin,
        'atan2': np.arctan2,
        'pi': np.pi
    }
```

In [11]:

```
# Make a vector class that can hold all the points along the curve

NP_3D_A1 = \
    create_class_Cartesian_3D_Vector(
        name = 'NP_3D_A1',
        component_names = [ 'x', 'y', 'z' ],
        brackets = '<>',
        sep = ', ',
        cnull = np.zeros(nr_of_points_along_curve),
        cunit = np.ones(nr_of_points_along_curve),
        functions = np_functions
    )
```

In [12]:

```python
# Calculate the points along the curve

angles_along_curve = np.linspace(-np.pi, +np.pi, nr_of_points_along_curve, endpoint=True)

p_o = \
    NP_3D_A1(
        x = f_x(angles_along_curve),
        y = f_y(angles_along_curve),
        z = f_z(angles_along_curve)
    )
```

In [13]:

```python
# Vectors from the first derivatives at the points along the curve

v_d1 = \
    NP_3D_A1(
        x = d1_f_x(angles_along_curve),
        y = d1_f_y(angles_along_curve),
        z = d1_f_z(angles_along_curve)
    )
```

In [14]:

```python
# Vectors from the second derivatives at the points along the curve

v_d2 = \
    NP_3D_A1(
        x = d2_f_x(angles_along_curve),
        y = d2_f_y(angles_along_curve),
        z = d2_f_z(angles_along_curve)
    )
```

In [15]:

```python
# Calculate the vectors for all the Frenet frames along the curve

# Tangent vectors at the points along the curve
v_t = v_d1.normalize()

# Binormal vectors at the points along the curve
v_b = v_d1.cross(v_d2).normalize()

# Normal vectors at the points along the curve
v_n = v_t.cross(v_b)
```

In [16]:

```python
# For all the points along the curve, calculate points in a circle across the curve

angles_across_curve = np.linspace(-np.pi, +np.pi, nr_of_points_across_curve, endpoint=True)

tube_radius = 0.3

surface_points = \
    [
        p_o + v_n.axis_rotate(v_t, angle) * tube_radius
        # p_o + v_n.axis_rotate(v_t, angle) * tube_radius * (3 + np.sin(2 * angle)) / 2
        for angle in angles_across_curve
    ]
```

In [17]:

```python
# Function for selecting color for each face

def select_color(i, j):

    if i % 2 == 0:
        color = 'black'
    else:
        if j % 2 == 0:
            color = 'blue'
        else:
            color = 'mediumturquoise'

    return color
```

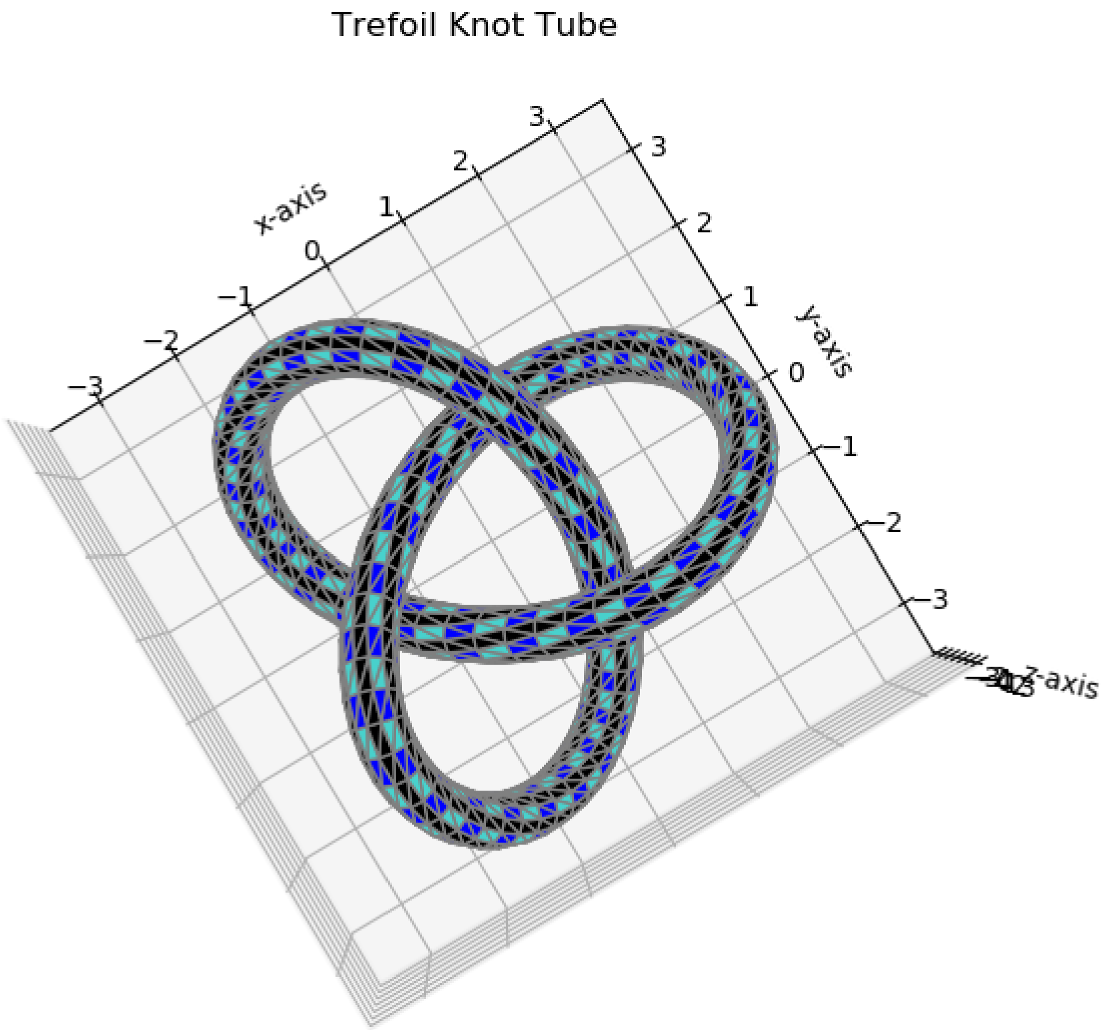
In [18]:

```python
quads_colors = \
    [
        select_color(i, j)
        for i in range(nr_of_points_across_curve-1)
        for j in range(nr_of_points_along_curve-1)
    ]
```

In [19]:

```python
# Function to extract coordinates for the quad patches.
# (It's only needed for the plot below.)

def quads(points_2d):

    sl0 = slice(None, -1)
    sl1 = slice(+1, None)
    for points0, points1 in zip(points_2d[sl0], points_2d[sl1]):
        points00 = zip(points0.x[sl0], points0.y[sl0], points0.z[sl0])
        points01 = zip(points0.x[sl1], points0.y[sl1], points0.z[sl1])
        points10 = zip(points1.x[sl0], points1.y[sl0], points1.z[sl0])
        points11 = zip(points1.x[sl1], points1.y[sl1], points1.z[sl1])
        yield from zip(points00, points01, points10, points11)
```

In [20]:
```python
# Show the trefoil knot tube

fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
fig.text(0.01, 0.01, url)
ax = Axes3D(fig)
ax.set_aspect(1)
ax.set_title('Trefoil Knot Tube')
for (p00, p01, p10, p11), face_color in zip(quads(surface_points), quads_colors):
    triangle_a = Poly3DCollection([ [ p00, p10, p11 ] ])
    triangle_a.set_facecolor(face_color)
    triangle_a.set_edgecolor('grey')
    ax.add_collection3d(triangle_a)
    triangle_b = Poly3DCollection([ [ p11, p01, p00 ] ])
    triangle_b.set_facecolor(face_color)
    triangle_b.set_edgecolor('grey')
    ax.add_collection3d(triangle_b)
ax.set_xlim(-3.5, +3.5)
ax.set_ylim(-3.5, +3.5)
ax.set_zlim(-3.5, +3.5)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.view_init(elev=90, azim=-120)
plt.show()
```



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

In [21]:

```
# Now do it in another way
```

In [22]:
```
# Make a vector class that can hold all the points on the surface of the tube

surface_shape = (nr_of_points_across_curve, nr_of_points_along_curve)
zeros = np.zeros(surface_shape)
ones = np.ones(surface_shape)

NP_3D_A2 = \
    create_class_Cartesian_3D_Vector(
        name = 'NP_3D_A2',
        component_names = [ 'xx', 'yy', 'zz' ],
        brackets = [ '<< ', ' >>' ],
        sep = ', ',
        cnull = zeros,
        cunit = ones,
        functions = np_functions
    )
```

In [23]:
```
# Verify that NumPy's array broadcasting works as needed

A1_cunit = NP_3D_A1.component_unit()
A2_cunit = NP_3D_A2.component_unit()

assert (A2_cunit * A1_cunit).shape == surface_shape
```

In [24]:
```
# Initialize position vectors for the points
# (The 1D arrays are beeing broadcasted to 2D arrays)

pp_o = \
    NP_3D_A2(
        xx = p_o.x,
        yy = p_o.y,
        zz = p_o.z
    )
```

In [25]:
```
# Alternative ways to do the same

# pp_o = NP_3D_A2(*p_o)

# pp_o = \
#     NP_3D_A2(
#         xx = A2_cunit * p_o.x,
#         yy = A2_cunit * p_o.y,
#         zz = A2_cunit * p_o.z
#     )

# tile_size = (nr_of_points_across_curve, 1)
# pp_o = \
#     NP_3D_A2(
#         xx = np.tile(p_o.x, tile_size),
#         yy = np.tile(p_o.y, tile_size),
#         zz = np.tile(p_o.z, tile_size)
#     )
```

In [26]:
```
# Initialize tangent, binormal and normal vectors

vv_t = NP_3D_A2(xx=v_t.x, yy=v_t.y, zz=v_t.z)
vv_b = NP_3D_A2(xx=v_b.x, yy=v_b.y, zz=v_b.z)
vv_n = NP_3D_A2(xx=v_n.x, yy=v_n.y, zz=v_n.z)
```

In [27]:
```
# Set up 2D arrays for angles along and across the curve

angles_along, angles_across = np.meshgrid(angles_along_curve, angles_across_curve)
```

In [28]:
```
# Calculate all the vectors along and across the curve towards the surface of the tube

vv_s = vv_n.axis_rotate(vv_t, angles_across)
```

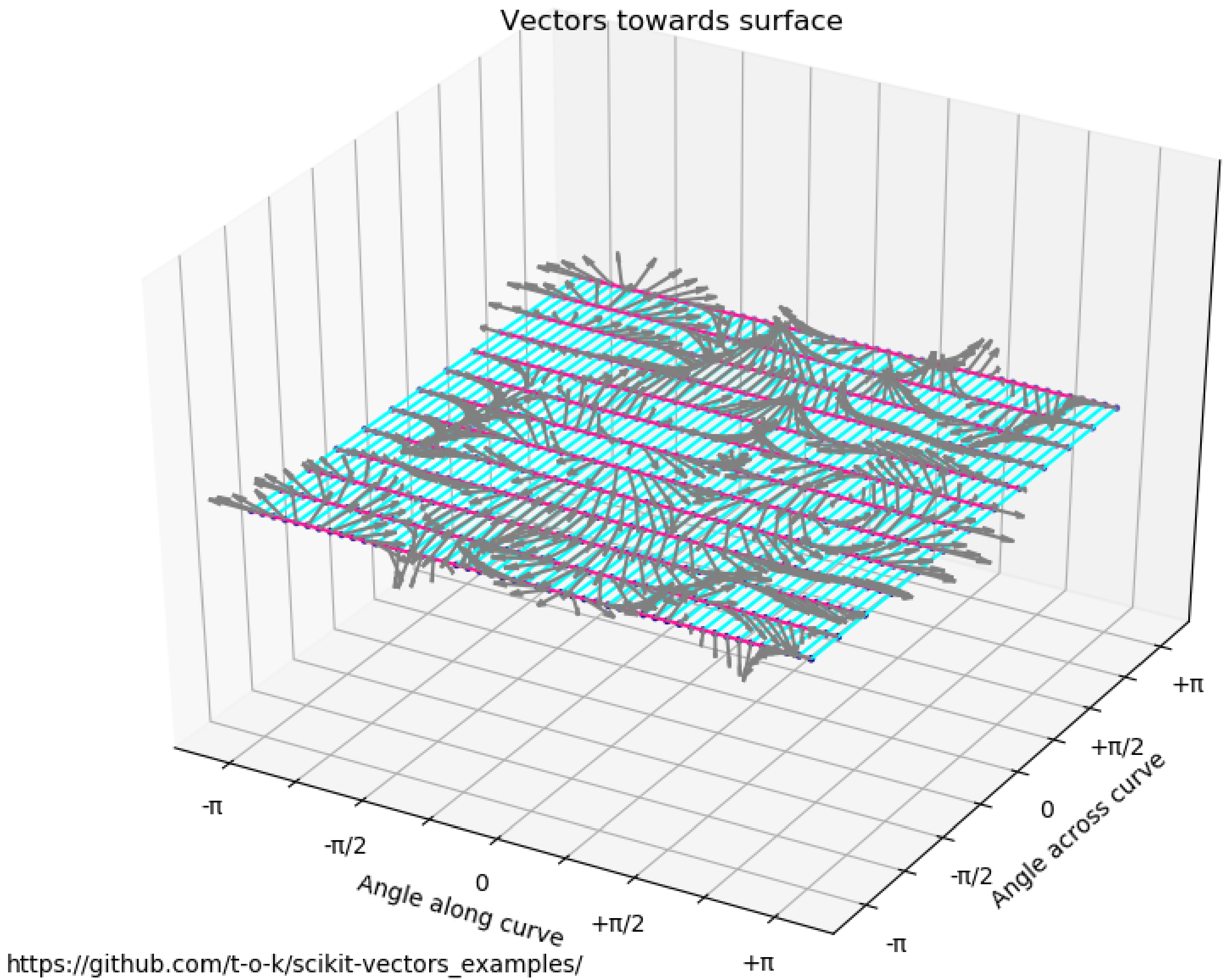
In [29]:

```python
# Prepare some variables for plotting

no_labels = [ ]
no_ticks = [ ]

pi_labels = [ '-π', '', '-π/2', '', '0', '', '+π/2', '', '+π' ]
pi_ticks = \
    [
        n / 4 * np.pi
        for n in [ -4, -3, -2, -1, 0, +1, +2, +3, +4 ]
    ]

vector_length = 0.5

stride_along = 2
stride_across = 1

sl_along = slice(None, None, stride_along)
sl_across = slice(None, None, stride_across)
sl = (sl_across, sl_along)
```

In [30]:

```
# Show some of the vectors calculated above

fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
fig.text(0.01, 0.01, url)
ax = Axes3D(fig)
ax.set_title('Vectors towards surface')
ax.scatter(
    angles_along[sl], angles_across[sl], zeros[sl],
    color = 'darkblue',
    marker = '.',
    edgecolors = 'face'
)
ax.plot_wireframe(
    angles_along, angles_across, zeros,
    rstride = 0,
    cstride = stride_along,
    color = 'cyan'
)
ax.plot_wireframe(
    angles_along, angles_across, zeros,
    rstride = stride_across,
    cstride = 0,
    color = 'deeppink'
)
ax.quiver(
    angles_along[sl], angles_across[sl], zeros[sl],
    vv_s.xx[sl], vv_s.yy[sl], vv_s.zz[sl],
    length = vector_length,
    pivot = 'tail',
    color = 'gray'
)
ax.set_xlim(-np.pi-0.5, +np.pi+0.5)
ax.set_ylim(-np.pi-0.5, +np.pi+0.5)
ax.set_zlim(-np.pi-0.5, +np.pi+0.5)
ax.set_xlabel('Angle along curve')
ax.set_ylabel('Angle across curve')
ax.set_xticklabels(pi_labels)
ax.set_yticklabels(pi_labels)
ax.set_zticklabels(no_labels)
ax.set_xticks(pi_ticks)
ax.set_yticks(pi_ticks)
ax.set_zticks(no_ticks)
ax.view_init(elev=36, azim=-60)
```

```
plt.show()
```

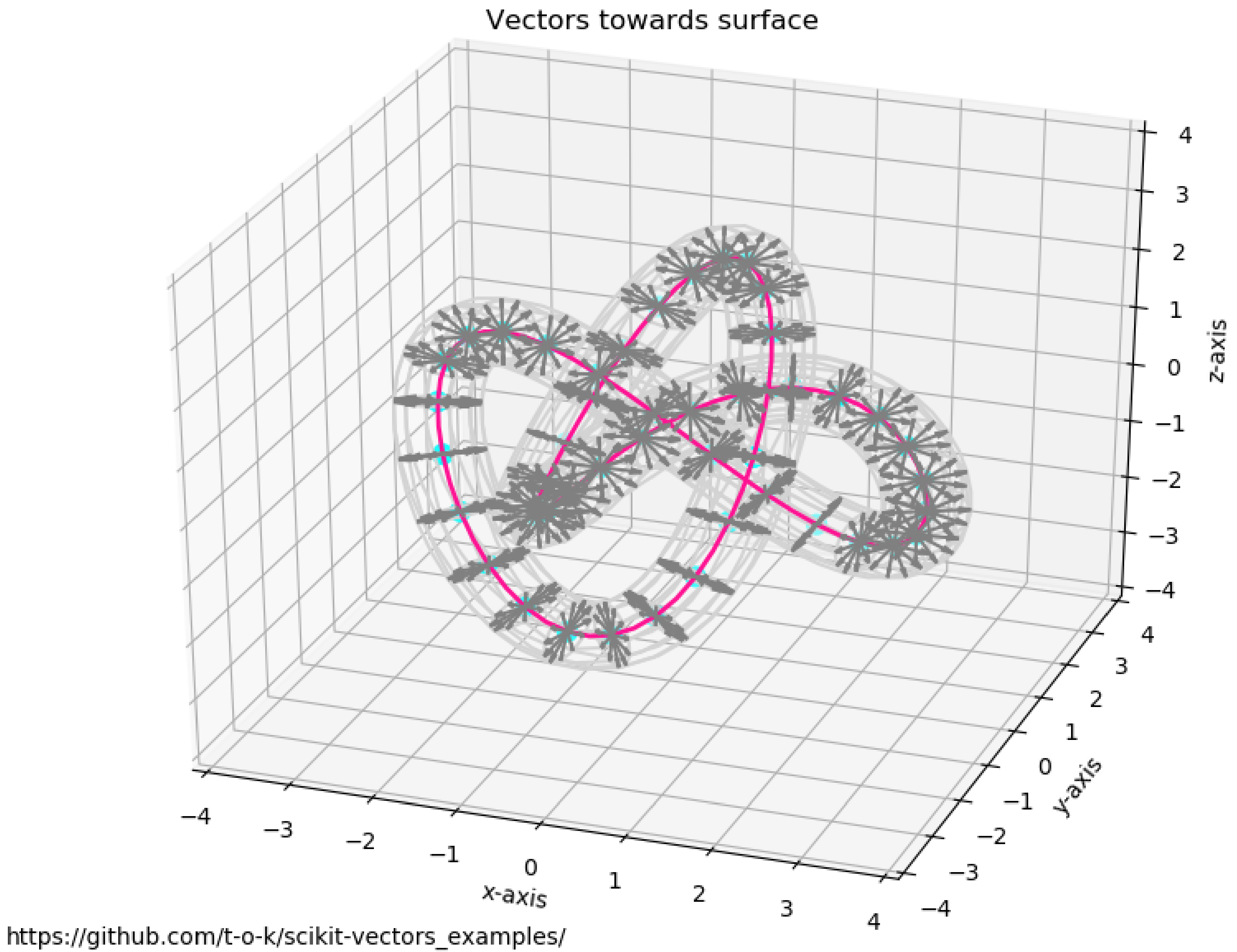


In [31]:

```
# Calculate all the position vectors for the points on the surface of the tube

pp_w = pp_o + vv_s * vector_length
```

In [32]:
```
# Show some of the vectors calculated above

fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
fig.text(0.01, 0.01, url)
ax = Axes3D(fig)
ax.set_title('Vectors towards surface')
ax.plot_wireframe(
    pp_w.xx, pp_w.yy, pp_w.zz,
    rstride = 0,
    cstride = stride_along,
    color = 'lightgray'
)
ax.plot(
    p_o.x, p_o.y, p_o.z,
    color = 'deeppink',
    linewidth = 2
)
ax.plot_wireframe(
    pp_w.xx, pp_w.yy, pp_w.zz,
    rstride = stride_across,
    cstride = 0,
    color = 'lightgray'
)
ax.quiver(
    pp_o.xx[sl], pp_o.yy[sl], pp_o.zz[sl],
    vv_s.xx[sl], vv_s.yy[sl], vv_s.zz[sl],
    length = vector_length,
    pivot = 'tail',
    color = 'gray'
)
ax.scatter(
    p_o.x[sl_along], p_o.y[sl_along], p_o.z[sl_along],
    color = 'cyan',
    marker = 'o',
    linewidth = 5
)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_xlim(-4, +4)
ax.set_ylim(-4, +4)
ax.set_zlim(-4, +4)
ax.view_init(elev=30, azim=-70)
```

```
plt.show()
```



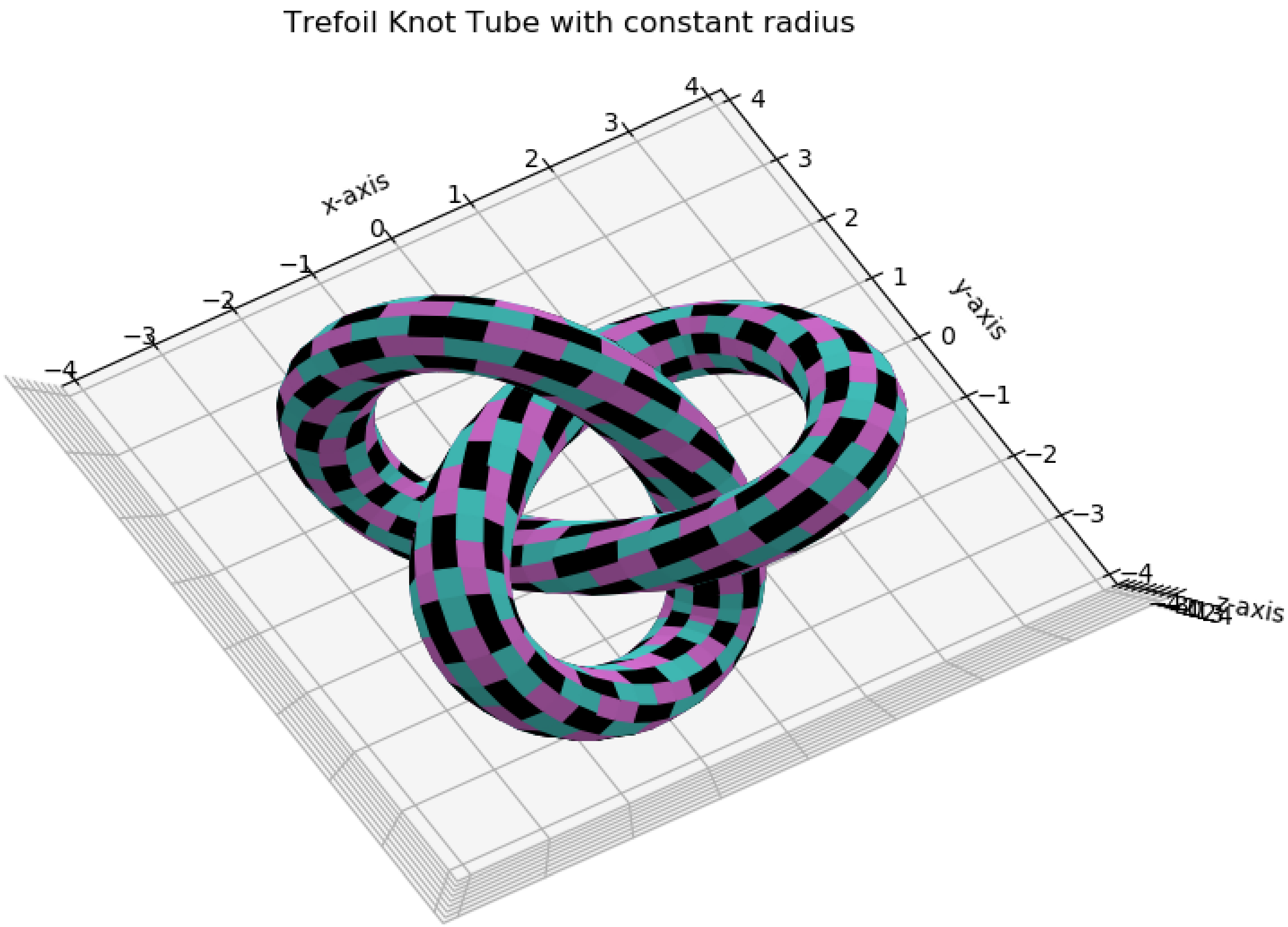
Vectors towards surface
x-axis
y-axis
z-axis
−4
−3
−2
−1
0
1
2
3
4
https://github.com/t-o-k/scikit-vectors_examples/

In [33]:

```python
# Select colors for the faces

def select_color(i, j):

    k = (i + 3 * j) % 6
#     k = (3 * i + 2 * j) % 6
    if k < 2:
        color = 'mediumturquoise'
    elif k < 4:
        color = 'black'
    else:
        color = 'orchid'

    return color


face_colors = \
    [
        [
            select_color(i, j)
            for i in range(nr_of_points_along_curve - 1)
        ]
        for j in range(nr_of_points_across_curve - 1)
    ]
```

In [34]:

```python
# Show the trefoil knot tube

fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
fig.text(0.01, 0.01, url)
ax = Axes3D(fig)
ax.set_title('Trefoil Knot Tube with constant radius')
ax.plot_surface(
    pp_w.xx, pp_w.yy, pp_w.zz,
    rstride = 1, cstride = 1,
    facecolors = face_colors,
    # cmap = plt.cm.inferno,
    # shade = False
)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_xlim(-4, +4)
ax.set_ylim(-4, +4)
ax.set_zlim(-4, +4)
# ax.view_init(elev=30, azim=-70)
ax.view_init(elev=90, azim=-120)
plt.show()
```



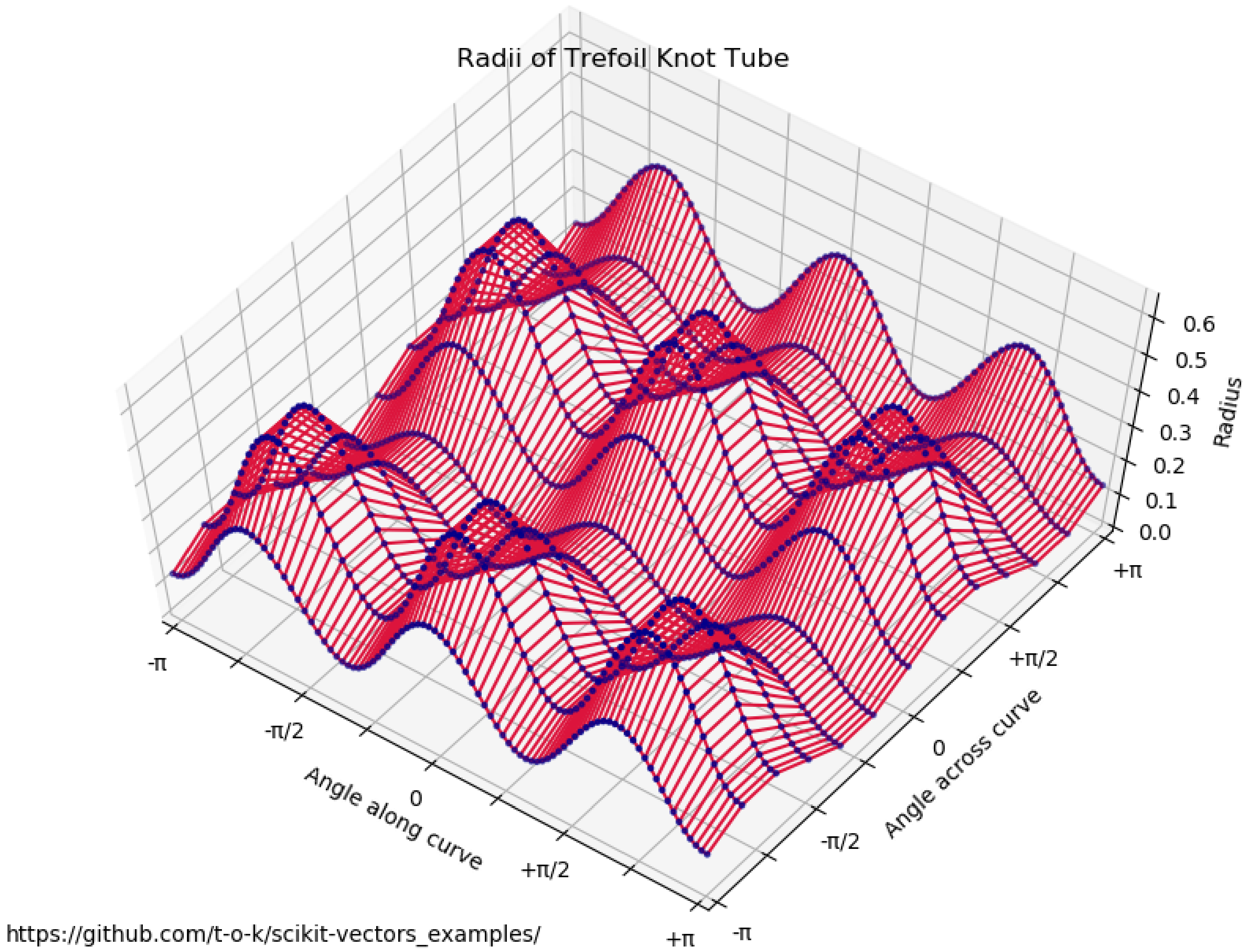
https://github.com/t-o-k/scikit-vectors_examples/

In [35]:

```
# Calculate all the radii for the tube along and across the curve

rr = (4 + 2 * np.sin(2 * angles_across)) * (6 + 3 * np.cos(3 * angles_along)) / 90

assert rr.shape == surface_shape
```

In [36]:

```
# Show the varying radii calculated above

fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
fig.text(0.01, 0.01, url)
ax = Axes3D(fig)
ax.set_title('Radii of Trefoil Knot Tube')
ax.plot_wireframe(angles_along, angles_across, rr, rstride=1, cstride=1, color='crimson')
ax.scatter(angles_along, angles_across, rr, color='darkblue', marker = '.')
# ax.plot_surface(angles_along, angles_across, rr, rstride=1, cstride=1, cmap=plt.cm.Spectral)
ax.set_xlabel('Angle along curve')
ax.set_ylabel('Angle across curve')
ax.set_zlabel('Radius')
ax.set_xlim(-np.pi, +np.pi)
ax.set_ylim(-np.pi, +np.pi)
ax.set_zlim(0.00, 0.65)
ax.set_xticklabels(pi_labels)
ax.set_yticklabels(pi_labels)
ax.set_xticks(pi_ticks)
ax.set_yticks(pi_ticks)
ax.view_init(elev=64, azim=-53)
plt.show()
```



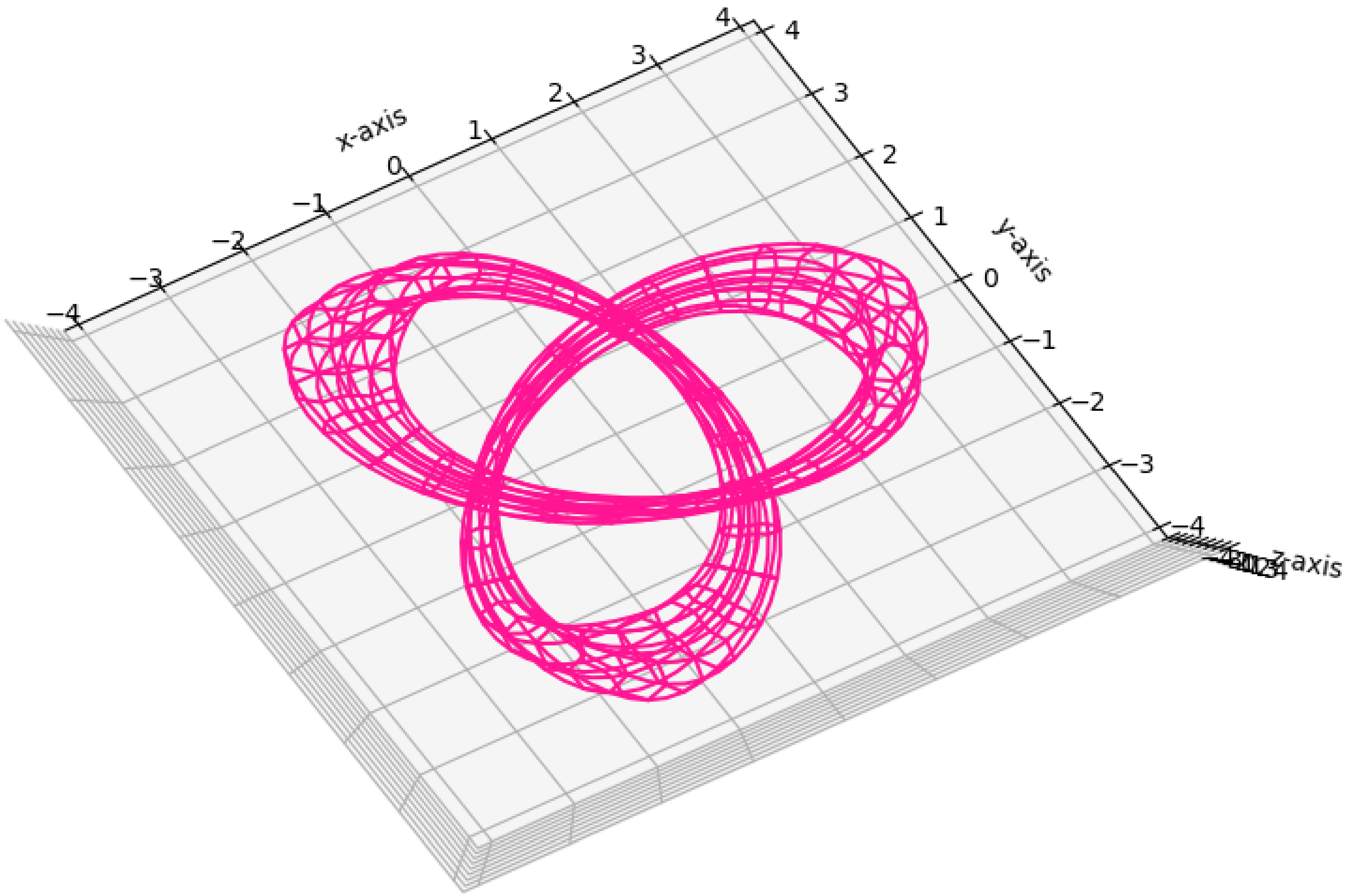
In [37]:

```
# Calculate all the position vectors for the points on the surface of the tube

pp_s = pp_o + vv_s * rr
```

In [38]:

```
# Show the trefoil knot tube

fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
fig.text(0.01, 0.01, url)
ax = Axes3D(fig)
ax.set_title('Trefoil Knot Tube with varying radius')
ax.plot_wireframe(*pp_s, color='deeppink')
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_xlim(-4, +4)
ax.set_ylim(-4, +4)
ax.set_zlim(-4, +4)
ax.view_init(elev=90, azim=-120)
plt.show()
```

Trefoil Knot Tube with varying radius



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

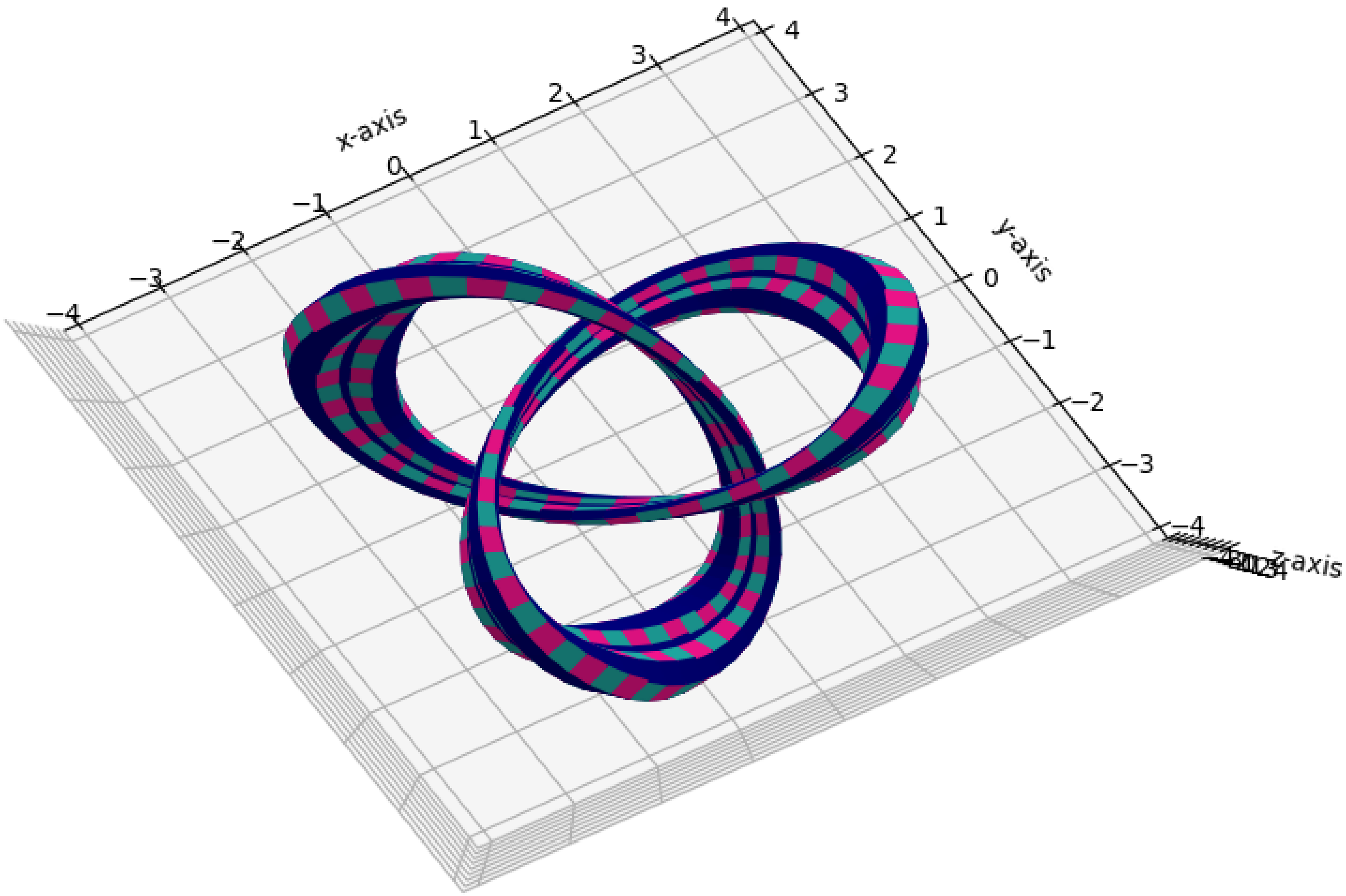
In [39]:

```
# Select colors for all the faces

def select_color(i, j):

    if j % 2 == 0:
    # if (i + j) % 2 == 0:
        color = 'navy'
    else:
        if i % 2 == 0:
            color = 'lightseagreen'
        else:
            color = 'deeppink'

    return color


face_colors = \
    [
        [
            select_color(i, j)
            for i in range(nr_of_points_along_curve - 1)
        ]
        for j in range(nr_of_points_across_curve - 1)
    ]
```

In [40]:

```
# Show the trefoil knot tube

fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
fig.text(0.01, 0.01, url)
ax = Axes3D(fig)
ax.set_title('Trefoil Knot Tube with varying radius')
ax.plot_surface(
    *pp_s,
    rstride = 1, cstride = 1,
    facecolors = face_colors,
    # cmap=plt.cm.inferno,
    # shade = False
)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_xlim(-4, +4)
ax.set_ylim(-4, +4)
ax.set_zlim(-4, +4)
ax.view_init(elev=90, azim=-120)
plt.show()
```

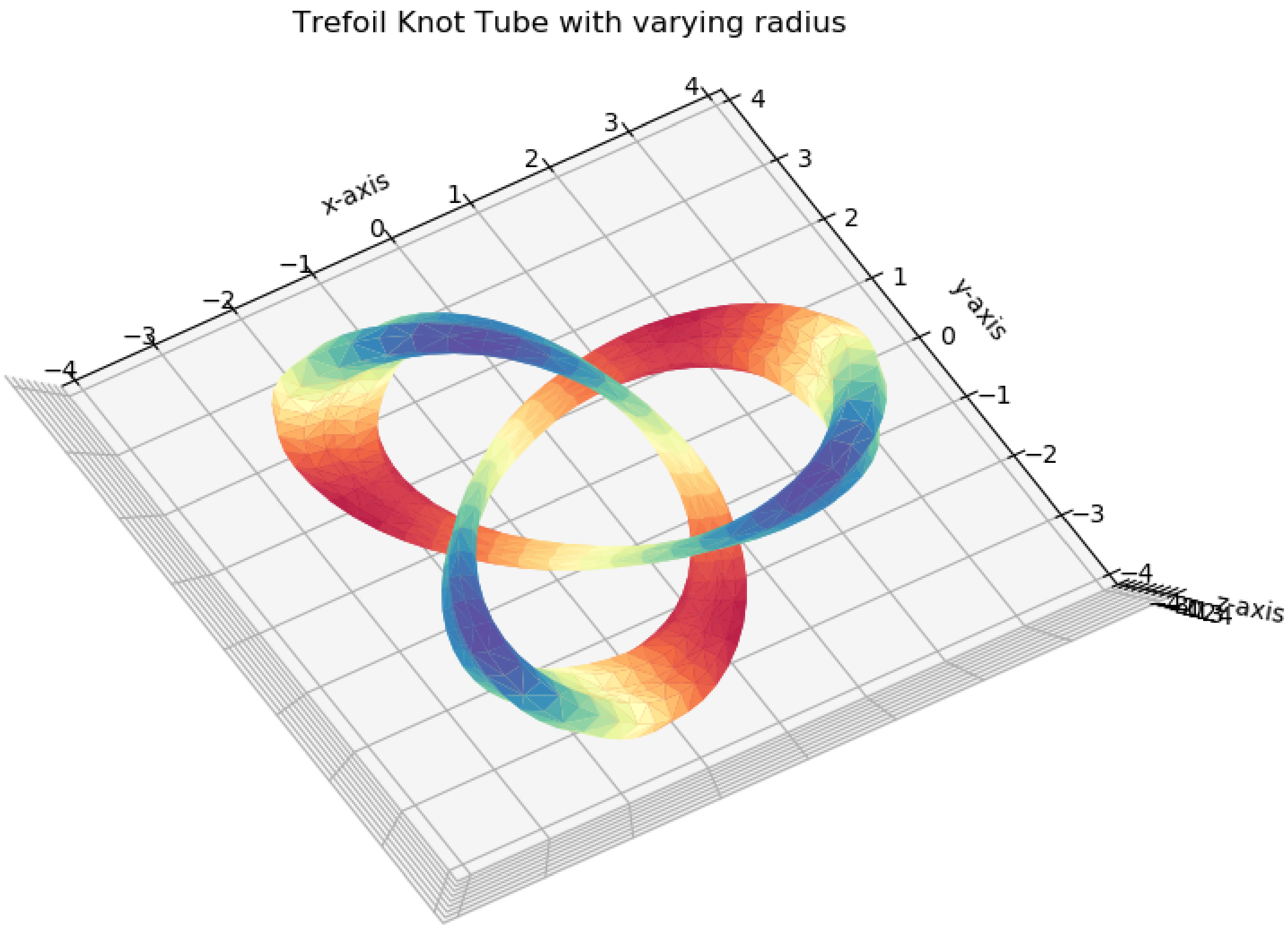
Trefoil Knot Tube with varying radius



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

In [41]:

```
# Show the trefoil knot tube

tri = \
    mtri.Triangulation(
        angles_along.flatten(),
        angles_across.flatten()
    )

fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
fig.text(0.01, 0.01, url)
ax = Axes3D(fig)
ax.set_title('Trefoil Knot Tube with varying radius')
ax.plot_trisurf(
    pp_s.xx.flatten(),
    pp_s.yy.flatten(),
    pp_s.zz.flatten(),
    triangles = tri.triangles,
    cmap = plt.cm.Spectral
)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_xlim(-4, +4)
ax.set_ylim(-4, +4)
ax.set_zlim(-4, +4)
ax.view_init(elev=90, azim=-120)
plt.show()
```



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

In [ ]: 1