Creating Bezier surfaces

- using Matplotlib, NumPy and scikit-vectors

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

In [1]:

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

In [2]:

```
# Uncomment one of these to get a Matplotlib backend with interactive plots

# %matplotlib auto
# %matplotlib notebook
```

In [3]:

```
import operator
from functools import reduce
import matplotlib.colors as colors
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 [4]:

```
# Size and resolution for Matplotlib figures

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

In [5]:

```
class Bicubic_Bezier():

    blend_fns = \
        [
            lambda s: (1 - s)**3,
            lambda s: 3 * s * (1 - s)**2,
            lambda s: 3 * s**2 * (1 - s),
            lambda s: s**3
        ]

    @staticmethod
    def _sum(values):

        return reduce(operator.add, values)


    def __init__(self, points4x4):

        self.points4x4 = points4x4


    def __call__(self, u, v):

        return \
            self._sum(
                self.blend_fns[j](u) *
                self._sum(
                    self.blend_fns[i](v) * self.points4x4[i][j]
                    for i in range(4)
                )
                for j in range(4)
            )
```

In [6]:

```
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,
        'int': np.rint,
        'ceil': np.ceil,
        'copysign': np.copysign,
        'log10': np.log10,
        'cos': np.cos,
        'sin': np.sin,
        'atan2': np.arctan2,
        'pi': np.pi
    }
```

In [7]:

```
control_grid_shape = (4, 4)

ControlGrid3D = \
    create_class_Cartesian_3D_Vector(
        name = 'ControlGrid3D',
        component_names = 'xyz',
        cnull = np.zeros(control_grid_shape),
        cunit = np.ones(control_grid_shape),
        functions = np_functions
    )
```

In [8]:

```
p3d_ctrl = \
    ControlGrid3D(
        x = \
            np.array(
                [
                    [  0.0,  1.0,  2.0,  3.0 ],
                    [  0.0,  1.0,  2.0,  4.0 ],
                    [  0.0,  1.0,  2.0,  2.5 ],
                    [  0.0,  1.0,  2.0,  3.0 ],
                ]
            ),
        y = \
            np.array(
                [
                    [  0.0,  0.0,  1.0,  0.0 ],
                    [  1.0,  1.0,  2.0,  1.0 ],
                    [  2.0,  2.0,  3.0,  2.0 ],
                    [  3.0,  3.0,  5.0,  3.0 ]
                ]
            ),
        z = \
            np.array(
                [
                    [  2.0,  0.0,  0.0, -3.0 ],
                    [ -2.0, -3.0, -2.0,  3.0 ],
                    [  0.0, -4.0,  0.0,  2.0 ],
                    [  2.0,  0.0,  0.0, -3.0 ]
                ]
            )
    )
```

In [9]:

```
surface_shape = nr_u, nr_v = (20, 30)

Surface3D = \
    create_class_Cartesian_3D_Vector(
        name = 'Surface3D',
        component_names = 'xyz',
        cnull = np.zeros(surface_shape),
        cunit = np.ones(surface_shape),
        functions = np_functions
    )
```

In [10]:

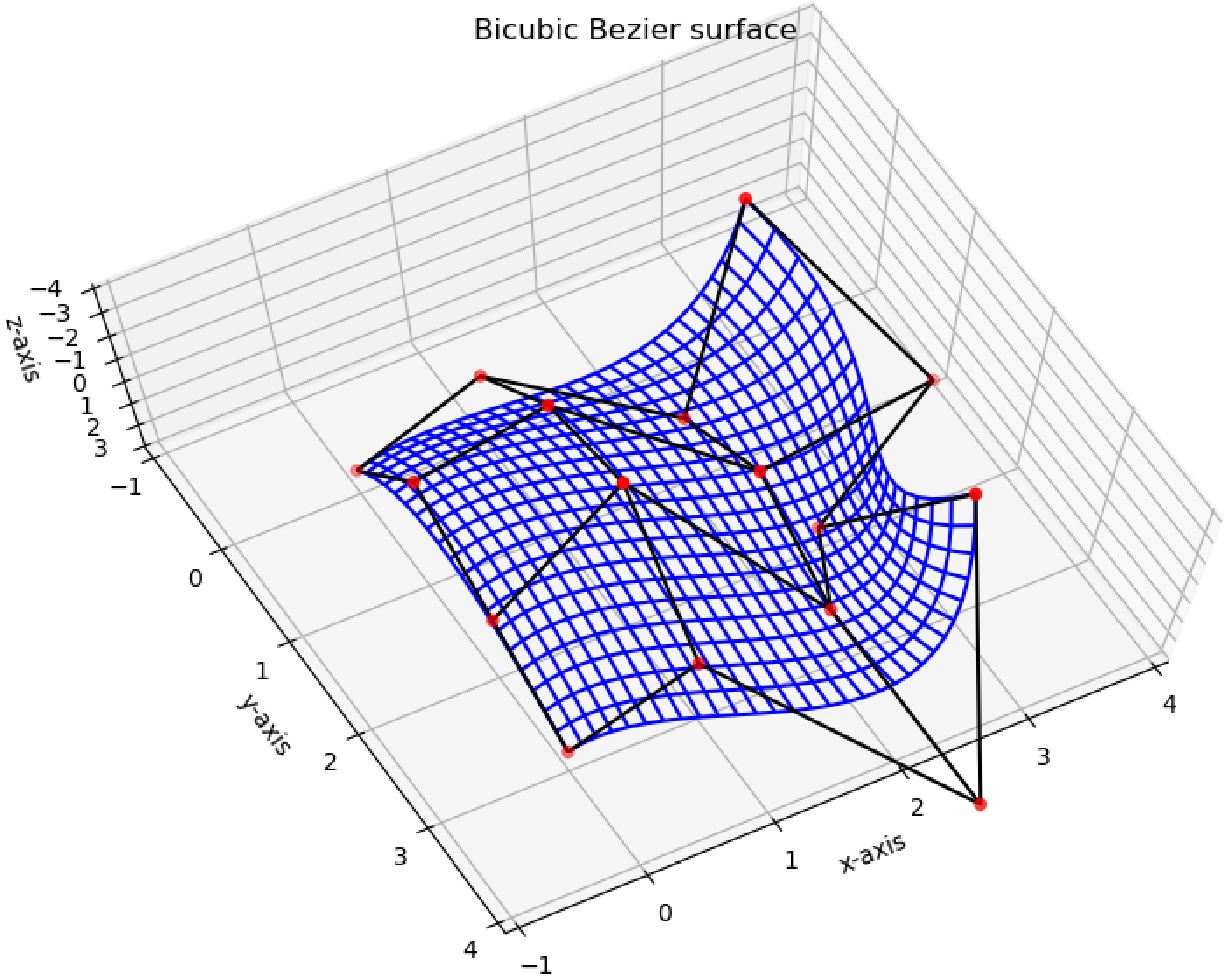
```
bb_x = Bicubic_Bezier(p3d_ctrl.x)
bb_y = Bicubic_Bezier(p3d_ctrl.y)
bb_z = Bicubic_Bezier(p3d_ctrl.z)
```

In [11]:

```
u, v = \
    np.meshgrid(
        np.arange(0, nr_v) / (nr_v - 1),
        np.arange(0, nr_u) / (nr_u - 1)
    )

bezier_points = \
    Surface3D(
        x = bb_x(u, v),
        y = bb_y(u, v),
        z = bb_z(u, v)
    )
```

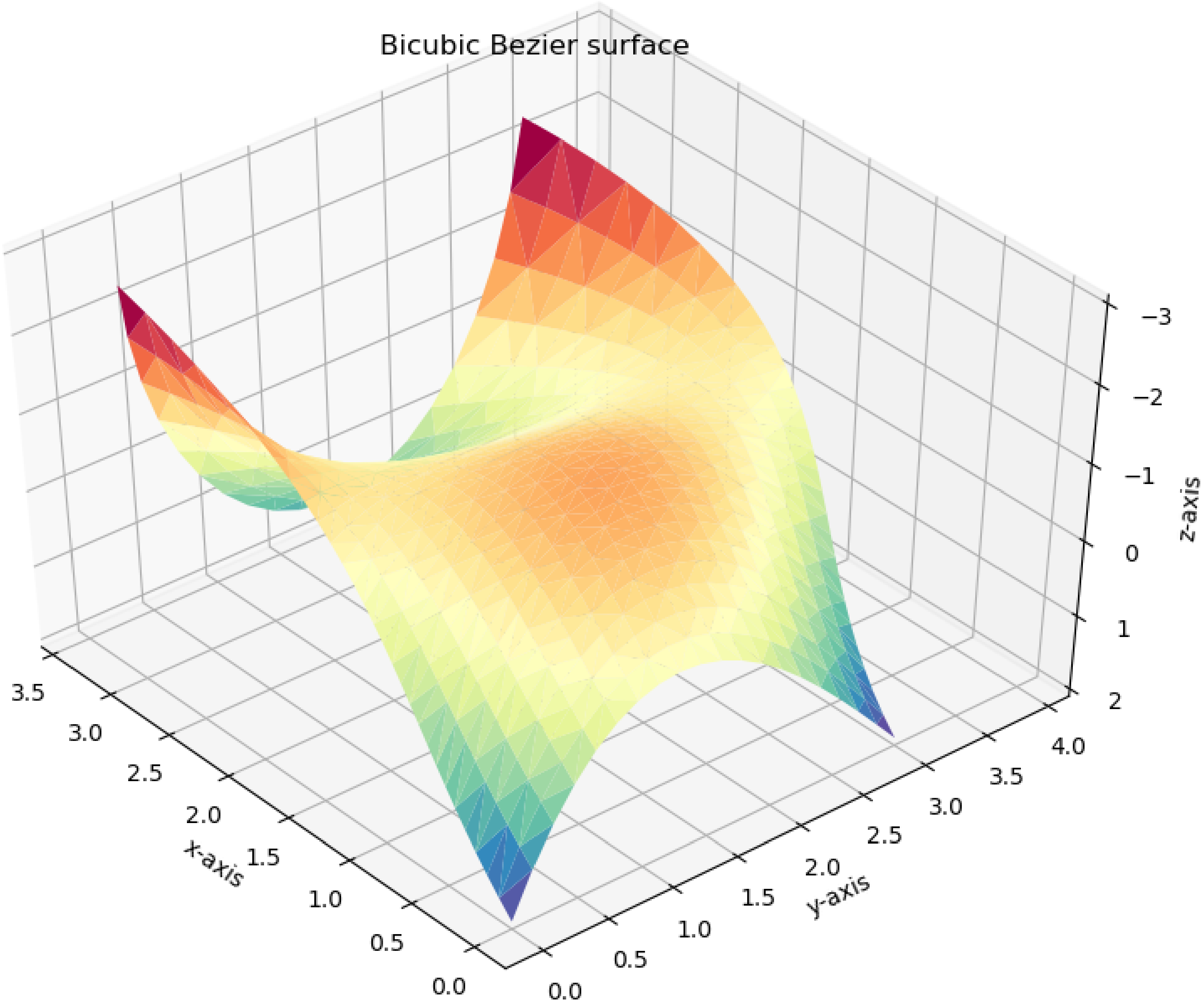
In [12]:

```
fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
ax = Axes3D(fig)
ax.set_title('Bicubic Bezier surface')
ax.plot_wireframe(*p3d_ctrl, color='black')
ax.scatter(p3d_ctrl.x, p3d_ctrl.y, p3d_ctrl.z, c='r', marker='o')
ax.plot_wireframe(bezier_points.x, bezier_points.y, bezier_points.z, color='blue')
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_xlim(-1, +4)
ax.set_ylim(-1, +4)
ax.set_zlim(-4, +3)
ax.view_init(elev=-105, azim=-61)
plt.show()
```


Bicubic Bezier surface
z-axis
−4
−3
−2
−1
0
1
2
3
y-axis
−1
0
1
2
3
4
x-axis
−1
0
1
2
3
4

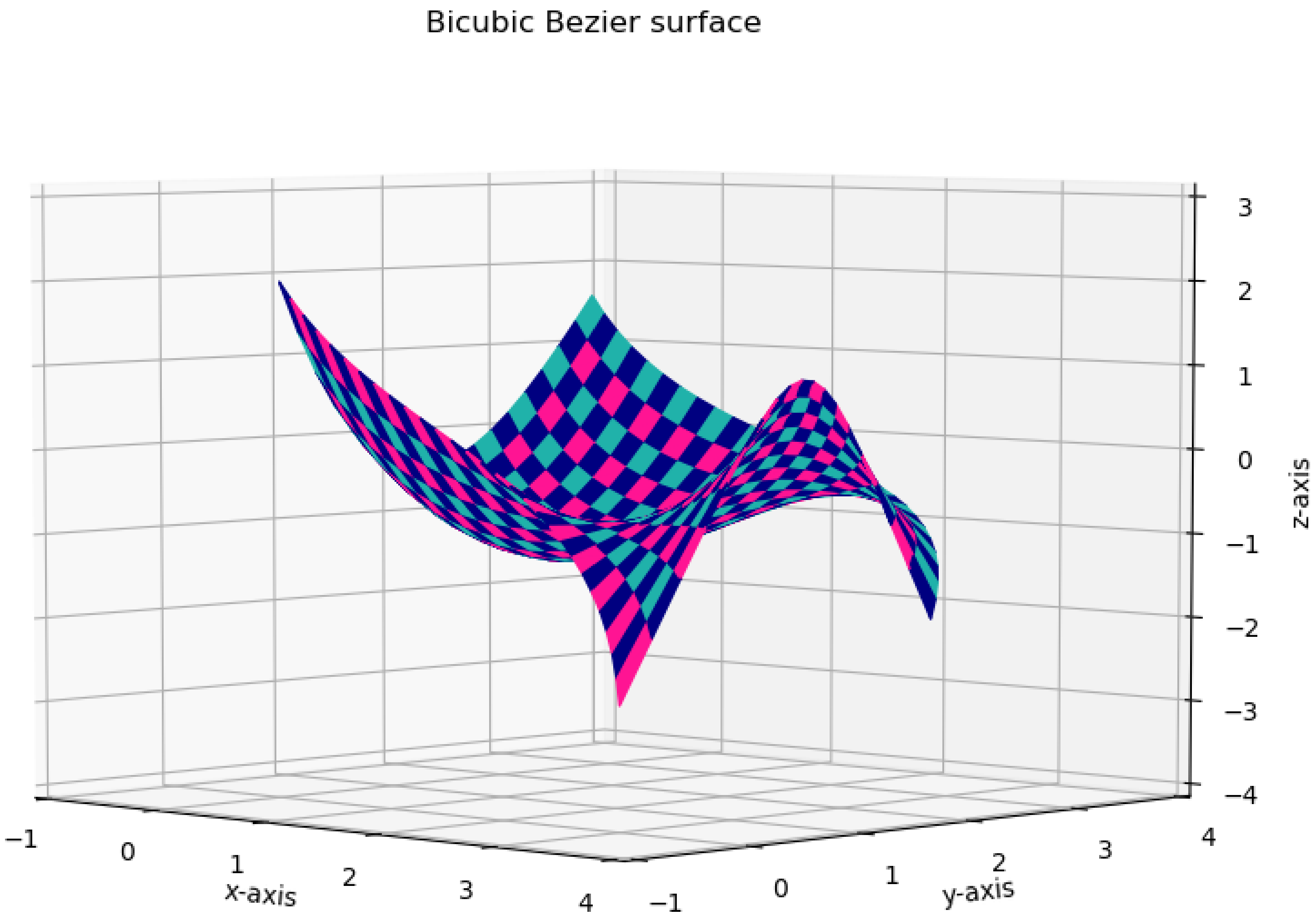
In [13]:

```
tri = \
    mtri.Triangulation(
        u.flatten(),
        v.flatten()
    )

fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
ax = Axes3D(fig)
ax.set_title('Bicubic Bezier surface')
ax.plot_trisurf(
    bezier_points.x.flatten(),
    bezier_points.y.flatten(),
    bezier_points.z.flatten(),
    triangles = tri.triangles,
    cmap = plt.cm.Spectral
)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.view_init(elev=-135, azim=40)
plt.show()
```


Bicubic Bezier surface
3.5
3.0
2.5
2.0
x-axis
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
y-axis
2.5
3.0
3.5
4.0
−3
−2
−1
0
1
2
z-axis

In [14]:

```python
fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
ax = Axes3D(fig)
ax.set_title('Bicubic Bezier surface')
for j in range(nr_v-2):
    for i in range(nr_u-2):
        if (i + j) % 2 == 0:
            color = 'navy'
        else:
            if j % 2 == 0:
                color = 'lightseagreen'
            else:
                color = 'deeppink'
        p00 = bezier_points(lambda cv: cv[i  , j  ])
        p01 = bezier_points(lambda cv: cv[i  , j+1])
        p10 = bezier_points(lambda cv: cv[i+1, j  ])
        p11 = bezier_points(lambda cv: cv[i+1, j+1])
        triangle_a = Poly3DCollection([ [ p00, p10, p11 ] ])
        triangle_a.set_color(color)
        # triangle_a.set_edgecolor('black')
        ax.add_collection3d(triangle_a)
        triangle_b = Poly3DCollection([ [ p11, p01, p00 ] ])
        triangle_b.set_color(color)
        # triangle_b.set_edgecolor('black')
        ax.add_collection3d(triangle_b)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_xlim(-1, +4)
ax.set_ylim(-1, +4)
ax.set_zlim(-4, +3)
ax.view_init(elev=5, azim=-46)
plt.show()
```


Bicubic Bezier surface
x-axis
−1
0
1
2
3
4
y-axis
−1
0
1
2
3
4
z-axis
3
2
1
0
−1
−2
−3
−4

In [15]:

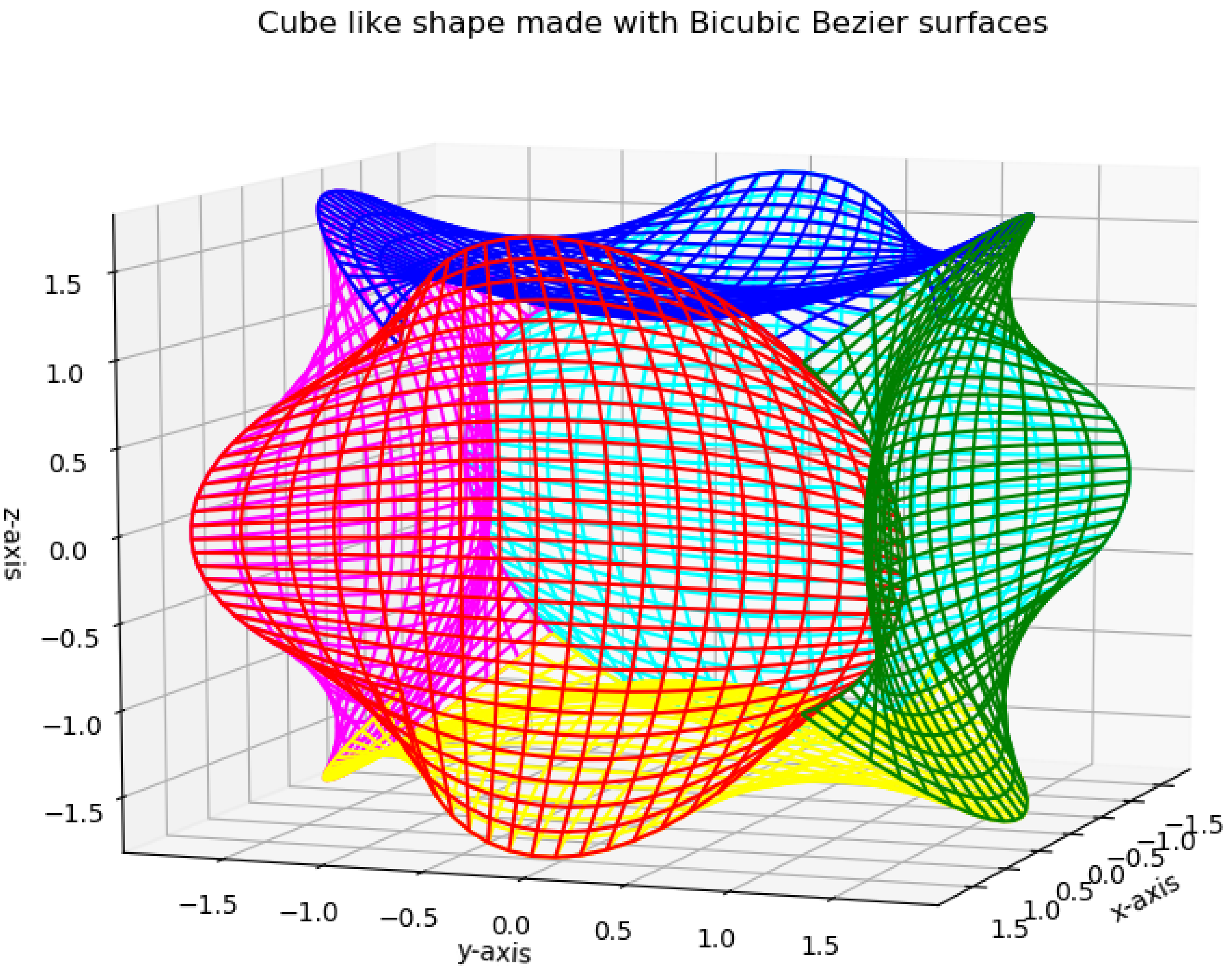
```
p3d_ctrl = \
    ControlGrid3D(
        x = \
            np.array(
                [
                    [  1.0,  2.0,  2.0,  1.0 ],
                    [  2.0,  0.5,  0.5,  2.0 ],
                    [  2.0,  0.5,  0.5,  2.0 ],
                    [  1.0,  2.0,  2.0,  1.0 ]
                ]
            ),
        y = \
            np.array(
                [
                    [ -1.0, -2.0, -2.0, -1.0 ],
                    [ -0.5, -0.5, -0.5, -0.5 ],
                    [  0.5,  0.5,  0.5,  0.5 ],
                    [  1.0,  2.0,  2.0,  1.0 ]
                ]
            ),
        z = \
            np.array(
                [
                    [ -1.0, -0.5,  0.5,  1.0 ],
                    [ -2.0, -0.5,  0.5,  2.0 ],
                    [ -2.0, -0.5,  0.5,  2.0 ],
                    [ -1.0, -0.5,  0.5,  1.0 ],
                ]
            )
    )
```

In [16]:

```
bb_x = Bicubic_Bezier(p3d_ctrl.x)
bb_y = Bicubic_Bezier(p3d_ctrl.y)
bb_z = Bicubic_Bezier(p3d_ctrl.z)

vxp = Surface3D(x=+1, y= 0, z= 0)
vxn = Surface3D(x=-1, y= 0, z= 0)
vyp = Surface3D(x= 0, y=+1, z= 0)
vyn = Surface3D(x= 0, y=-1, z= 0)
vzp = Surface3D(x= 0, y= 0, z=+1)
vzn = Surface3D(x= 0, y= 0, z=-1)

bezier_points_xp = \
    Surface3D(
        x = bb_x(u, v),
        y = bb_y(u, v),
        z = bb_z(u, v)
    )
bezier_points_yp = bezier_points_xp.reorient(vxp, vyp)
bezier_points_yn = bezier_points_xp.reorient(vxp, vyn)
bezier_points_zp = bezier_points_xp.reorient(vxp, vzp)
bezier_points_zn = bezier_points_xp.reorient(vxp, vzn)
bezier_points_xn = bezier_points_yp.reorient(vyp, vxn)

```

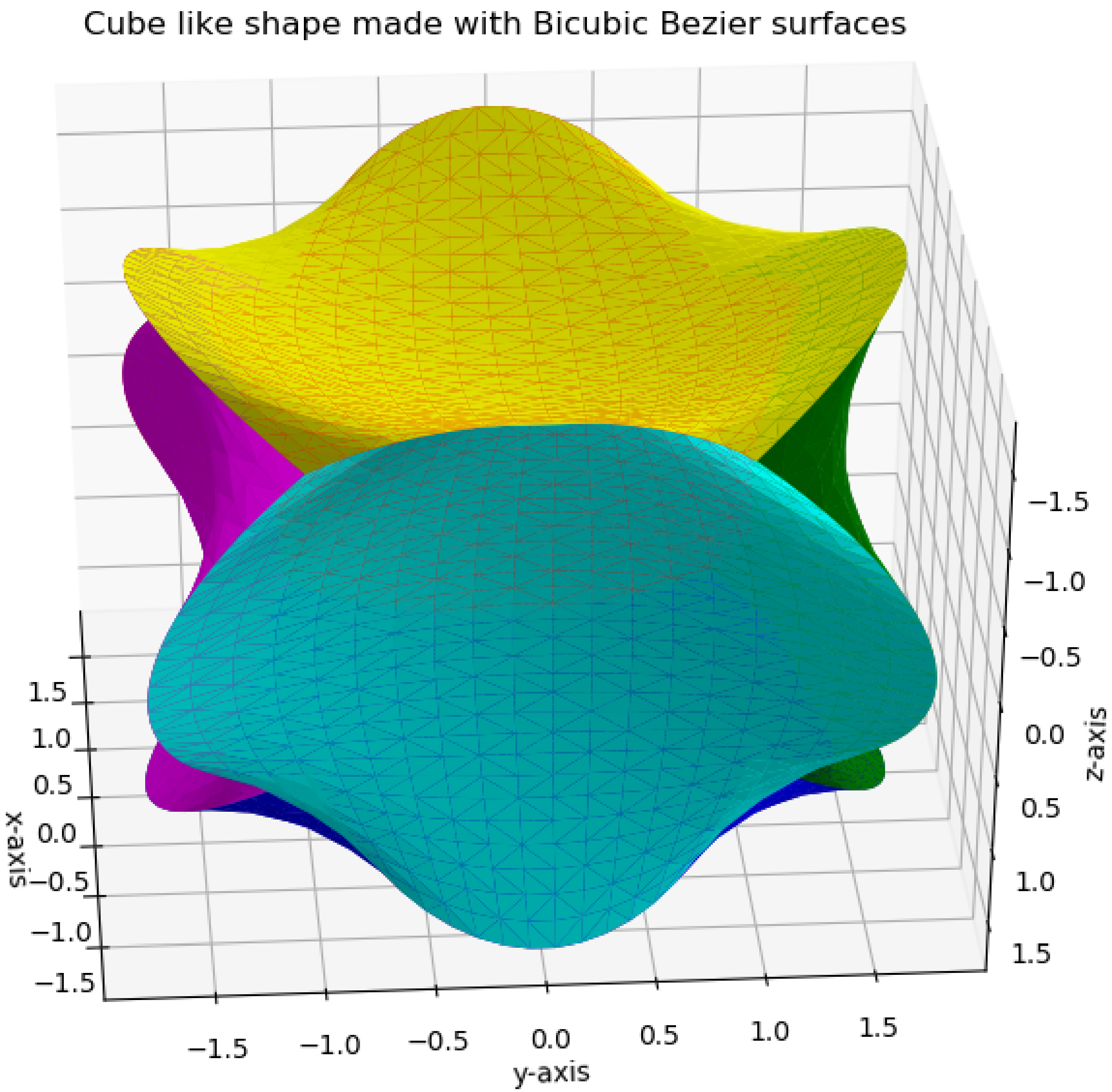
In [17]:

```
fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
ax = Axes3D(fig)
ax.set_title('Cube like shape made with Bicubic Bezier surfaces')
ax.plot_wireframe(*bezier_points_xp, color='red')
ax.plot_wireframe(*bezier_points_xn, color='cyan')
ax.plot_wireframe(*bezier_points_yp, color='green')
ax.plot_wireframe(*bezier_points_yn, color='magenta')
ax.plot_wireframe(*bezier_points_zp, color='blue')
ax.plot_wireframe(*bezier_points_zn, color='yellow')
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
# ax.set_xlim(-1, +5)
# ax.set_ylim(-4, +3)
# ax.set_zlim(-1, +4)
ax.view_init(elev=10, azim=20)
plt.show()
```


Cube like shape made with Bicubic Bezier surfaces
z-axis
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y-axis
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x-axis
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

In [18]:

```
tri = \
    mtri.Triangulation(
        u.flatten(),
        v.flatten()
    )

fig = plt.figure(figsize=figure_size, dpi=figure_dpi)
ax = Axes3D(fig)
ax.set_title('Cube like shape made with Bicubic Bezier surfaces')
ax.plot_trisurf(*bezier_points_xp(np.ndarray.flatten), triangles = tri.triangles, color = 'red')
ax.plot_trisurf(*bezier_points_xn(np.ndarray.flatten), triangles = tri.triangles, color = 'cyan')
ax.plot_trisurf(*bezier_points_yp(np.ndarray.flatten), triangles = tri.triangles, color = 'green')
ax.plot_trisurf(*bezier_points_yn(np.ndarray.flatten), triangles = tri.triangles, color = 'magenta')
ax.plot_trisurf(*bezier_points_yn(np.ndarray.flatten), triangles = tri.triangles, color = 'magenta')
ax.plot_trisurf(*bezier_points_zp(np.ndarray.flatten), triangles = tri.triangles, color = 'blue')
ax.plot_trisurf(*bezier_points_zn(np.ndarray.flatten), triangles = tri.triangles, color = 'yellow')
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.view_init(elev=-145, azim=4)
plt.show()
```


Cube like shape made with Bicubic Bezier surfaces
x-axis
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y-axis
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
z-axis
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

In [ ]: