GCode Animation with Python: G0-G1, G2-G3, G05*

Umut Özen

Department of Mathematics,
Dokuz Eylül University.

2024-2025 - Fall Semester

*Supervisor: Dr. Başak Karpuz

Abstract

Abstract

In this presentation, we introduce some GCodes for CNC (Computer Numerical Control) machines. Then, we introduce mathematical background related to motion and behavior GCodes of the CNC machine. Finally, we give our `Python` codes that creates an animation for a tangential cutter by reading a `.gcode` file.

Outline of the Talk

Below, we explain the procedure on how we retrieve information from weekly course schedules in `DEBIS`.

1. Examining the target web page on web browser
   1. Playing with the objects
   2. Inspecting the source
2. Reading data from the internet by using `Python`
   1. Getting list of academics
   2. Reading department information
   3. Getting weekly course data
3. Parsing data from the source by using `Python`
   1. Extracting timetable entries
4. Saving data to an `Excel Sheet` by using `Python`
5. Processing data in the excel sheet by using `Python`

GCodes

GCode is a programming language used to control the movements and operations of CNC (Computer Numerical Control) machines. It specifies how the machine should move, how fast it should move, and what operations it should perform.

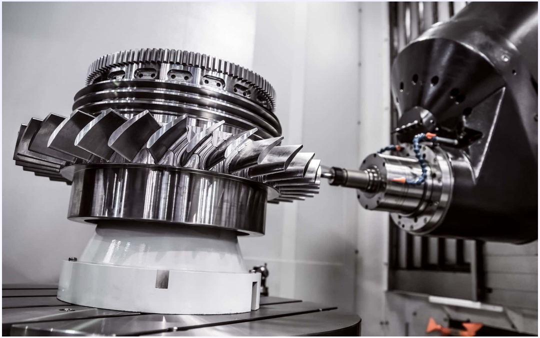

Figure 1: This is an example picture of a CNC machine.

GCodes

Some GCodes: Linear Motion G0-G1

The G0 and G1 commands add a linear move to the queue to be performed after all previous moves are completed.

G-code Examples:

- `G0 X10 Y20` ; Moves the CNC head quickly to the coordinate (X=10, Y=20).
- `G1 X30 Y40 F150:` ; Moves the CNC head to (X=30, Y=40) at a specified feed rate (e.g., F150 units).

GCodes

Some GCodes: Circular Motion G2-G3

G2 adds a clockwise arc move to the planner, while G3 adds a counterclockwise arc. An arc move starts at the current position and ends at the given XYZ, pivoting around a center-point offset specified by I and J.

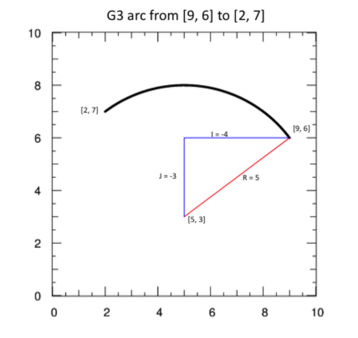

Figure 2: Example of a G3 arc move.

G-code Examples:

- `G0 X9 Y6` ; Rapid move to the starting point
- `G3 X2 Y7 I-4 J-3` ; Counterclockwise arc movement
- `G2 X9 Y6 I3 J-4` ; Clockwise arc movement

GCodes

Some GCodes: Cubic Spline G5

G5 is used to create smooth cubic B-spline curves in the XY plane. These splines are defined using the X and Y axes, along with control parameters P and Q, which determine the control point offsets, and optionally I and J, which specify the starting direction of the spline. The I and J parameters are required for the first G5 command in a series and help establish the initial tangent direction of the curve.

Gcodes Example

```
G0 X2 Y2 Z1 E0;
G1 X17 Y7 Z1 E18.43;
G3 X18.95 Y15.28 I-1.58 J4.74 E135;
G1 X17.54 Y16.69 E135;
G2 X11.68 Y30.84 I14.1423 J14.1453 E90;
G5 X75 Y75 I0 J100 P-15 Q-175 E85.10;
```

Figure 3: This is an example picture of a Gcodes.

Curves Related to the GCodes: G0-G1, G2-G3, G5

Some GCodes: Linear Motion G0-G1

The line segment starting from the point P_0 and ending at the point P_1

$$\alpha(t) := P_0 + t(P_1 - P_0), \quad 0 \leq t \leq 1,$$

whose tangent slope is the constant value

$$\alpha'(t) := (P_1 - P_0), \quad 0 \leq t \leq 1,$$

Curves Related to the GCodes: G0-G1, G2-G3, G5

Some GCodes: Circular Motion G2-G3

The circular arc centered at the point P_0 with radius r, starting at an angle θ_0 and ending at an angle θ_1

$$\alpha(t) := P_0 + r\big(\cos(t), \sin(t)\big), \quad \theta_0 \leq t \leq \theta_1,$$

whose tangent slope is the circular curve

$$\alpha'(t) := r\big(-\sin(t), \cos(t)\big), \quad \theta_0 \leq t \leq \theta_1.$$

Curves Related to the GCodes: G0-G1, G2-G3, G5

Some GCodes: Cubic Spline G5

Cubic Bézier curve starting at the point P_0, ending at the point P_3 and with the additional control points P_1 and P_2

$$\alpha(t) := \sum_{i=0}^{3} \binom{3}{i} (1-t)^{3-i} t^i P_i, \quad 0 \leq t \leq 1,$$

whose tangent curve is

$$\begin{aligned} \alpha'(t) := & \sum_{i=0}^{2} \binom{3}{i} (3-i)(1-t)^{2-i} t^i P_i \\ & + \sum_{i=1}^{3} \binom{3}{i} i(1-t)^{3-i} t^{i-1} P_i, \quad 0 \leq t \leq 1. \end{aligned}$$

Animation Program Code

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import comb
from scipy.integrate import quad
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation
from scipy.spatial.transform import Rotation as R
from mpl_toolkits.mplot3d.art3d import Poly3DCollection

# G0/G1 [x,y,z,e,f,i,j,p,q]
# [[x1,y1,z1,e1,f1,p1,q1],[x2,y2,z2,e2,f2,p2,q2],...,[xn,yn,zn,en
    ,fn,pn,qn]]
#To Do
#1. Add Plotting Codes for G5 V
#2. Use Partition Norm to Determine the Number of Segments in a
    Curve X
#3. Use Arrows for G05 V
#4. Run the Cursor Along the Curve V
#5. Use Blade for Cursor Instead of Red Ball X
```

Animation Program Code

```
def interpolate_points(x, y, z, steps=100):
    # Find all interpolation points between two points
    x_interp = np.linspace(x[0], x[1], steps)
    y_interp = np.linspace(y[0], y[1], steps)
    z_interp = np.linspace(z[0], z[1], steps)
    return x_interp, y_interp, z_interp
def cubic_bezier_derivative(t, control_points):
    #Calculates the derivative vector at a point on the B zier curve.
    derivative_bernstein_coeffs = np.insert([comb(3, i, exact=True) * (3 - i) * (1 - t) ** (2 - i) * (-1) * t ** i for i in range(0, 3)],3,0)+np.insert([comb(3, i, exact=True) * i * (1 - t) ** (3 - i) * t ** (i - 1) for i in range(1, 4)],0,0)
    return np.dot(derivative_bernstein_coeffs, control_points)
    #return -3 * (1 - t)**2 * control_points[0] + (3 * (1 - t)**2 - 6 * (1 - t) * t) * control_points[1] + (6 * (1 - t) * t - 3 * t**2) * control_points[2] + 3 * t**2 * control_points[3]
def arc_length_integrand(t, control_points):
    #This function calculates the integrand part of an arc length integral of a Cubic B zier curve.
    derivative = cubic_bezier_derivative(t, control_points)
    return np.linalg.norm(derivative)
```

Animation Program Code

```python
def update(frame):
    # Take position
    current_pos = np.array([
        x_interp_all[frame],
        y_interp_all[frame],
        z_interp_all[frame]
    ])

    # Calculate rotation (using e_interp_all only)
    e_angle = e_interp_all[frame]  # Angle in degrees
    rotation_matrix = R.from_euler('z', e_angle, degrees=True).
    as_matrix()

    # Rotate and move the object
    rotated_vertices = (rotation_matrix @ vertices.T).T
    new_vertices = rotated_vertices + current_pos
```

Animation Program Code

```
    # Identify new faces
    new_faces = [
        [new_vertices[0], new_vertices[1], new_vertices[2]],
        [new_vertices[0], new_vertices[1], new_vertices[3]],
        [new_vertices[1], new_vertices[2], new_vertices[3]],
        [new_vertices[2], new_vertices[0], new_vertices[3]]
    ]

    # Update faces
    poly3d.set_verts(new_faces)
    return poly3d,

def convert_np_float64_list_to_ndarray(np_list):

    #Converts a list of type numpy float64 to ndarray.
    return np.array([float(item) for item in np_list])
```

Animation Program Code

```
def cubic_bezier(t, control_points):
    '''
    Computes a point on a cubic B zier curve.
    t: Parameter in [0, 1]
    Control Points = [P0, P1, P2, P3]: Control points (3D)
    '''
    #(1-t)**3 * control_points[0] + 3*(1-t)**2 * t * control_points[1] + 3*(1-t) * t**2 * control_points[2] + t**3 * control_points[3]
    bernstein_coeffs = np.array([comb(3, i, exact=True)*(1-t)**(3-i)*t**i for i in range(4)])
    return np.dot(bernstein_coeffs, control_points)
```

Animation Program Code

```python
def take_coordinate(coordinates, coordinate_name,
    coordinate_previous):
    gcode_u_init = coordinates.find(coordinate_name)
    if gcode_u_init == -1:
        return coordinate_previous
    else:
        gcode_u_end = coordinates.find(" ", gcode_u_init + 1)
        if gcode_u_end != -1:
            gcode_u = float(coordinates[gcode_u_init + 1:
    gcode_u_end])
        else:
            gcode_u = float(coordinates[gcode_u_init + 1:])
        return gcode_u
```

Thank you very much for your interest to our talk.