TP4 : Courbes de Bézier

XIA Qiaoqiao et AIT HAMID Adam

1 Fonction pour le calcul du coefficient binominial

La formule pour le coefficient binomial est donnée par : $C(n,p) = \frac{n!}{p!\cdot(n-p)!}$, grâce à la librairie math de Python on pourra utiliser la fonction factorial pour effectuer une division entière.

```python
def binom(n, p):
    """Calcule le coefficient binomial C(n, p)"""
    return math.factorial(n) // (math.factorial(p) * math.factorial(n - p))
```

2 Fonction pour le calcul du coefficient de Bernstein

$$B_{n,i}(t) = \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i} \tag{1}$$

$$B_{n,i}(t) = C(n,i) \cdot t^i \cdot (1-t)^{n-i} \tag{2}$$

```python
def bernstein(n, i, t):
    """Calcule le coefficient de Bernstein B_n,i(t)"""
    return binom(n, i) * (t ** i) * ((1 - t) ** (n - i))
```

3 Fonction Bezier

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i} \cdot P_i | \text{Bezier}(t) = \sum_{i=0}^{n} \text{Bernstein}(n,i,t) \cdot \text{control\_points}[i] \tag{3}$$

```python
def bezier(control_points, n_points=1000):
    """Calcule les points de la courbe de B zier"""
    n = len(control_points) - 1
    curve_points = []
    for t in np.linspace(0, 1, n_points):
        x, y = 0, 0
        for i in range(n + 1):
            x += bernstein(n, i, t) * control_points[i][0]
            y += bernstein(n, i, t) * control_points[i][1]
        curve_points.append((x, y))
    return curve_points
```

4 Affichage

```python
control_points = [(0, 0), (1, 2), (2, 3), (3, 0)]
curve_points = bezier(control_points)
affichage(control_points, curve_points)
```

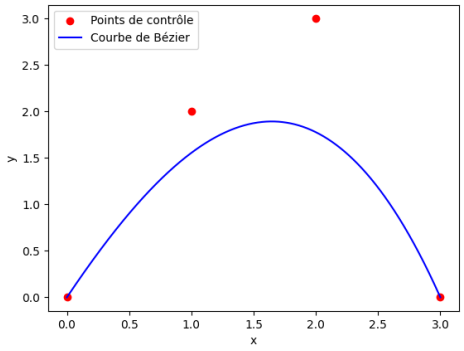

Figure 1: Affichage de la courbe de Bézier