1 Digitale Medien Blatt 2

1.1 Aufgabe 1

$p(t) = p_3^3 = (1-t)p_2^2 + tp_3^2$

$= (1\text{-}t)[(1\text{-}t)\ p_1^1 + tp_2^1] + t[(1-t)p_2^1 + tp_3^1]$

$= (1\text{-}t)[(1\text{-}t)[(1\text{-}t)p_0 + tp_1] + t[(1-t)p_1 + tp_2]] + t[(1-t)[(1-t)p_1 + tp_2] + t[(1-t)p_2 + tp_3]]$

$= (1\text{-}t)[(1\text{-}t)^2 p_0 + t(1-t)p_1 + t(1-t)p_1 + t^2 p_2] + [(1-t)^2 p_1 + t(1-p)p_2 + t(1-t)p_2 + t^2 p_3]$

$= (1\text{-}t)^3 p_0 + t(1-t)^2 p_1 + t(1-t)^2 p_1 + t^2(1-t)p_2 + t(1-t)^2 p_1 + t^2(1-t)p_2 + t^2(1-t)p_2 + t^3 p_3$

1.2 Aufgabe 2

n = Ordnung

$$\sum_{i=0}^{n}((n-k+1)t^k(1-t)^{(n-k)}p^k) - (n+1)$$

1.3 Aufgabe 3

Algorithm 1 Einfacher Algorithmus

```
t_start = 0
n = Dimension
while(t_start < 1){
    for(k = 0, k<n, i++){
        P = P + (n–k+1) t_start^k (1 − t_start)^(n−k) p^k
    }
    P − (n+1)
    draw P(t_start)
    t_start += ε
}
```

Algorithm 2 Bezier Algorithmus

```
function devide(point P0, point P1, point P2, point P3){
        if(isLine(P0,P1,P2,P3) == true){
                drawLine(point p0, point p1)
        }else{
                devide(point P0, point P1)
                devide(point P2, point P3)
        }
}
```

1.4 Aufgabe 4

Algorithmus 1:

- (+) einfach zu implementieren
- (-) viele Berechnungen: lange Laufzeit
- (-) trotzdem können große Abstände zwischen manchen Punkten liegen, sodass die Linie nicht durchgängig ist.

Algorithmus 2:

- (+) keine Pixel-Lücken in der Kurve
- (+) bessere Laufzeit wegen divide and conquer