Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Комп'ютерний практикум №3
Побудова сплайнових кривих

предмет «Комп'ютерна графіка»

Роботу виконав:
Студент 3 курсу ФТІ, групи ФІ-91
Цибульник Антон Владиславович

Приймав:
Професор кафедри ІБ
Півень Олег Борисович

2022

Мета

Навчитись будувати різні види сплайнових кривих для заданого масиву опорних точок. Виконати дослідження залежності виду кривої Безьє від порядку слідування опорних точок.

Теоретичні відомості

Сплайн – крива, яка використовується для апроксимації заданих базових (опорних) точок. Існує велика кількість сплайнових кривих, які відрізняються своїми властивостями. Нехай на площині заданий упорядкований набір точок $P_0, \dots, P_m$. Тоді ламана $P_0, \dots, P_m$ називається контрольною ламаною, що породжена заданим масивом $P = \{P_0, P_1, \dots, P_m\}$. Для побудови сплайнової кривої для набору точок $P_0, \dots, P_m$ виконують такий алгоритм:

1) Для побудови кривої на проміжку між точками P_i, P_{i+1} беруть четвірку точок $P_{i-1}, P_i, P_{i+1}, P_{i+2}$;

2) Задають діапазон зміни параметра $0 \leqslant t \leqslant 1$. Значення параметра $t = 0$ відповідає початковій точці, а $t = 1$ – кінцевій точці на ділянці кривої між точками P_i, P_{i+1}. Значення $0 < t < 1$ відповідають внутрішнім точкам даної ділянки;

3) Розбивають діапазон зміни параметра t на n частин (наприклад, $n = 10$);

4) На основі значень відповідних координат четвірки базових точок й значень $t_k, k = \overline{0, n}$, розраховуються n проміжних точок сплайнової кривої між базовими точками P_i, P_{i+1};

5) Розраховані на попередньому кроці точки з'єднуються прямими лініями. Таким чином, чим вище значення n, тим більш точно буде апроксимована сплайнова крива;

Для побудови складної сплайнової кривої, що починається в першій базовій точці та закінчується в останній базовій точці, достатньо доповнити набір копіями першої та останньої точок. Копія першої точки при цьому додається в початок набору, а копія останньої точки – в кінець набору.

Види сплайнових кривих

1. Інтерполяційна крива Catmull-Rom:

$$r(t) = \frac{1}{2}\left(-t(1-t)^2 P_0 + (2 - 5t^2 + 3t^3)P_1 + t(1 + 4t - 3t^2)P_2 - t^2(1-t)P_3\right)$$

2. Кубічна крива Безьє (для чотирьох контрольних точок):

$$r(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

3. Кубічна B-сплайнова крива:

$$r(t) = \frac{1}{6}\left((1-t)^3P_0 + (3t^3 - 6t^2 + 4)P_1 + (-3t^3 + 3t^2 + 3t + 1)P_2 + t^3P_3\right)$$

4. Елементарна B-сплайнова крива:

$$r(t) = b_0(t)P_0 + b_1(t)P_1 + b_2(t)P_2 + b_3(t)P_3, \text{ де}$$
$$b_0(t) = \frac{1}{\delta}\left(2\beta_1^3(1-t)^3\right),$$
$$b_1(t) = \frac{2\beta_1^3t(t^2-3t+3) + 2\beta_1^2(t^3-3t+2) + 2\beta_1(t^3-3t+2) + \beta_2(2t^3-3t^2+1)}{\delta},$$
$$b_2(t) = \frac{1}{\delta}\left(2\beta_1^2t^2(3-t) + 2\beta_1t(3-t^2) + 2\beta_2t^2(3-3t) + 2(1-t^3)\right),$$
$$b_3(t) = \frac{2t^3}{\delta},$$

$$\beta_1 \geqslant 0,\ \beta_2 \geqslant 0,\ \delta = 2\beta_1^3 + 4\beta_1^2 + 4\beta_1 + \beta_2 + 2.$$

Завдання

1. На координатній площині задано масив опорних точок $P_0, P_1, P_2, P_3, P_4, P_5, P_6$. Скласти програму побудови таких сплайнових кривих:

   - інтерполяційна крива Catmull-Rom;
   - кубічна крива Безьє;
   - кубічна B-сплайнова крива;
   - елементарна B-сплайнова крива (поекспереметувати з $\beta_1,\ \beta_2$).

   Зауваження: координати опорних точок задані в пікселах відносно початку координат, що знаходиться у лівому нижньому куту поверхні екрана, на яку повинна проектуватися площина розміром 200×100 умовних математичних одиниць.

   ## Вимоги до програми

   Кожна програма повинна будувати:

   – контрольну ламану лінію (штрихову), що з'єднує точки в порядку їх проходження;

   – сплайнові криві для відповідної комбінації точок.

   Програма повинна виконувати розмітку та оцифровку області побудови сплайнових кривих з кроком 10 умовних математичних одиниць.

2. Скласти програму побудови кубічних кривих Безьє для кожної з 6-ти комбінацій (порядку слідування) опорних точок: $P_1P_2P_3P_4$, $P_1P_2P_4P_3$, $P_1P_3P_2P_4$, $P_1P_3P_4P_2$, $P_1P_4P_2P_3$, $P_1P_4P_3P_2$.

   Зауваження: координати опорних точок задані в пікселах відносно початку координат, що розміщений у лівому нижньому куті області побудови зображення, на яку повинна проецируватися площина розміром 200×100 умовних математичних одиниць.

Вимоги до програми

Кожна програма повинна будувати:

– кожну криву Безьє в окремому вікні виводу, для чого весь екран слід розбити штриховими лініями на 6 областей (по 3 області в 2-х рядках);

– в кожному випадку програма повинна будувати контрольні відрізки (штриховою лінією), що з'єднують опорні точки в порядку їх проходження в кожній комбінації та відповідні підписи P_1, P_2, P_3, P_4 біля точок.

Програма повинна виконувати розмітку та оцифровку області побудови сплайнових кривих з кроком 10 умовних математичних одиниць.

Код програми й скріншоти результатів

Чотири різних види сплайнових кривих

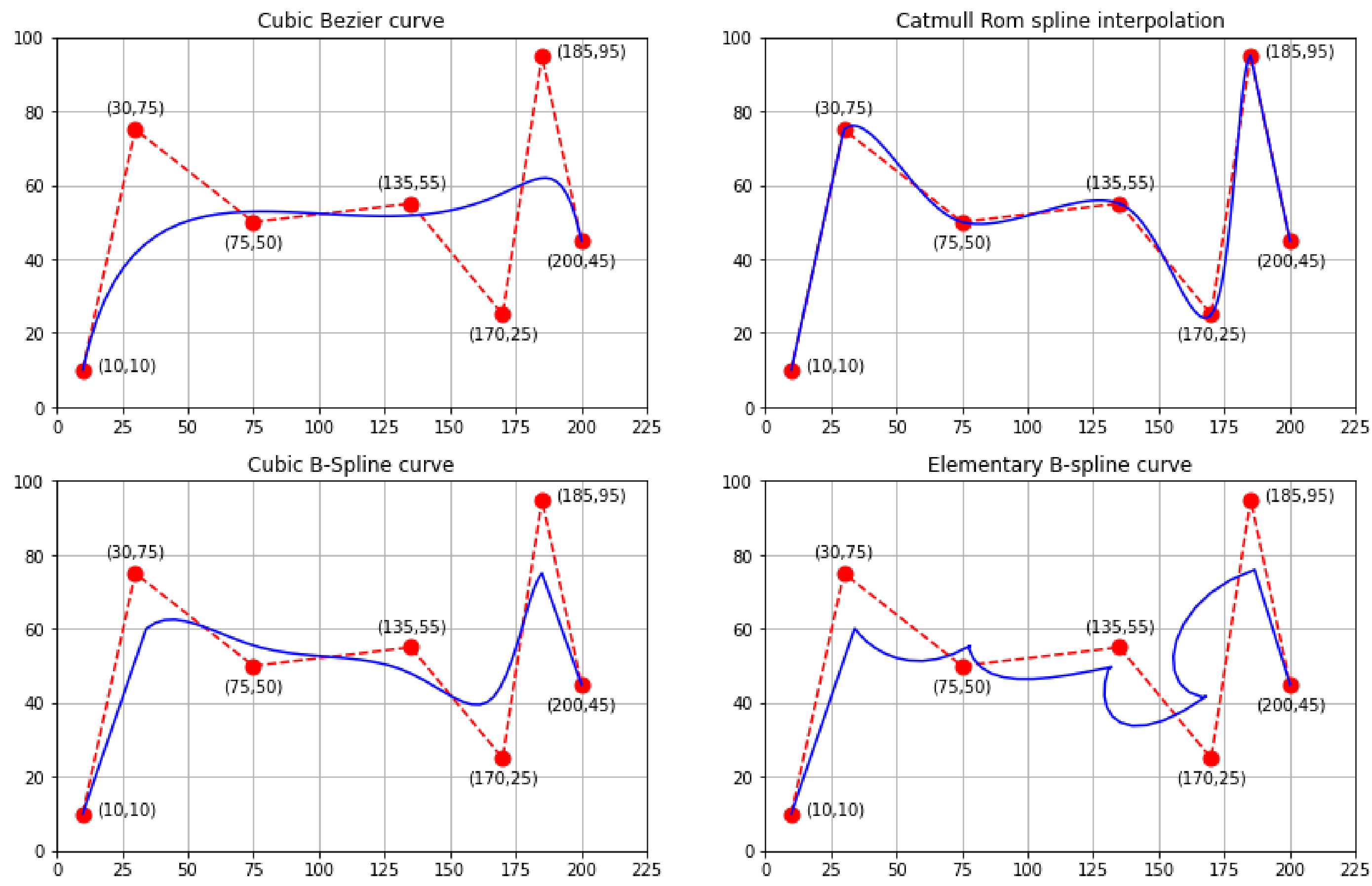
```python
import matplotlib.pyplot as plt

P_x = [10, 30, 75, 135, 170, 185, 200]
P_y = [10, 75, 50, 55, 25, 95, 45]

plt.figure(figsize=(14, 9))

def plot_scatter_7points():
    for x,y in zip(P_x,P_y):
        label = f"({x},{y})"
        if (x == 30 or x == 135):
            plt.annotate(label, # this is the text
                         (x,y), # these are the coordinates to position the label
                         textcoords="offset points", # how to position the text
                         xytext=(0,10), # distance from text to points (x,y)
                         ha="center") # horizontal alignment (left, right or center)
        elif (x == 75 or x == 170 or x == 200):
            plt.annotate(label,
                         (x,y),
                         textcoords="offset points",
                         xytext=(0,-15),
                         ha="center")
```

```
        elif (x == 10 or x == 185):
            plt.annotate(label,
                         (x,y),
                         textcoords="offset points",
                         xytext=(8,0),
                         ha="left")

    plt.grid()

    plt.xlim(0,225)
    plt.ylim(0,100)

    return plt.plot(P_x, P_y, "o--r", markersize = 9)

def Cubic_Bezier_curve_7points(n):
    plot_scatter_7points()

    r_x = []
    r_y = []

    t = []
    t.append(0)

    for i in range(0, n):
        t.append(t[i] + 1/n)

    for i in range(0, len(t)):
        dt = 1-t[i]
        r_x.append(pow(dt,6)*P_x[0] + 6*pow(dt,5)*t[i]*P_x[1] +
            15*pow(dt,4)*pow(t[i],2)*P_x[2] + 20*pow(dt,3)*pow(t[i],3)*P_x[3] +
                15*pow(dt,2)*pow(t[i],4)*P_x[4] + 6*dt*pow(t[i],5)*P_x[5] +
                  pow(t[i],6)*P_x[6])
        r_y.append(pow(dt,6)*P_y[0] + 6*pow(dt,5)*t[i]*P_y[1] +
            15*pow(dt,4)*pow(t[i],2)*P_y[2] + 20*pow(dt,3)*pow(t[i],3)*P_y[3] +
                15*pow(dt,2)*pow(t[i],4)*P_y[4] + 6*dt*pow(t[i],5)*P_y[5] +
                  pow(t[i],6)*P_y[6])

    plt.title("Cubic Bezier curve")
    return plt.plot(r_x,r_y,color="blue")

def Catmull_Rom_spline_interpolation(n):
    plot_scatter_7points()

    r_x = []
    r_y = []

    r_x.append(P_x[0])
    r_y.append(P_y[0])

    t = []
    t.append(0)

    for i in range(0, n):
        t.append(t[i] + 1/n)
```

```
    for i in range(1, 5):
        for j in range(0, len(t)):
            r_x.append(0.5*(-t[j]*pow(1-t[j],2)*P_x[i-1] +
                (2-5*t[j]**2+3*pow(t[j],3))*P_x[i] +
                t[j]*(1+4*t[j]-3*t[j]**2)*P_x[i+1] - t[j]**2*(1-t[j])*P_x[i+2]))
            r_y.append(0.5*(-t[j]*pow(1-t[j],2)*P_y[i-1] +
                (2-5*t[j]**2+3*pow(t[j],3))*P_y[i] +
                t[j]*(1+4*t[j]-3*t[j]**2)*P_y[i+1] - t[j]**2*(1-t[j])*P_y[i+2]))

    r_x.append(P_x[6])
    r_y.append(P_y[6])

    plt.title("Catmull Rom spline interpolation")
    return plt.plot(r_x,r_y,color="blue")

def Cubic_Beta_Spline_curve(n):
    plot_scatter_7points()

    r_x = []
    r_y = []

    r_x.append(P_x[0])
    r_y.append(P_y[0])

    t = []
    t.append(0)

    for i in range(0, n):
        t.append(t[i] + 1/n)

    for i in range(1, 5):
        for j in range(0, len(t)):
            r_x.append((pow(1-t[j],3)*P_x[i-1] + (3*pow(t[j],3)-6*t[j]**2+4)*P_x[i] +
                (-3*pow(t[j],3)+3*pow(t[j],2)+3*t[j]+1)*P_x[i+1] +
                pow(t[j],3)*P_x[i+2])/6)
            r_y.append((pow(1-t[j],3)*P_y[i-1] + (3*pow(t[j],3)-6*t[j]**2+4)*P_y[i] +
                (-3*pow(t[j],3)+3*pow(t[j],2)+3*t[j]+1)*P_y[i+1] +
                pow(t[j],3)*P_y[i+2])/6)

    r_x.append(P_x[6])
    r_y.append(P_y[6])

    plt.title("Cubic B-Spline curve")
    return plt.plot(r_x,r_y,color="blue")

def Elementary_Beta_spline_curve(n,B1,B2):
    plot_scatter_7points()

    r_x = []
    r_y = []

    r_x.append(P_x[0])
    r_y.append(P_y[0])

```

```
    t = []
    t.append(0)

    for i in range(0, n):
        t.append(t[i] + 1/n)

    d = 2*pow(B1,3) + 4*pow(B1,2) + 4*B1 + B2 + 2

    for i in range(1, 5):
        for j in range(0, len(t)):
            b0 = (2*pow(B1,3)*pow(1-t[j],3))/d
            b1 = (2*pow(B1,3)*t[j]*(pow(t[j],2)-3*t[j]+3) +
                2*pow(B1,2)*(pow(t[j],3)-3*t[j]+2) + 2*B1*(pow(t[j],3)-3*t[j]+2) +
                B2*(2*pow(t[j],3)-3*pow(t[j],2)+1))/d
            b2 = (2*pow(B1,2)*pow(t[j],2)*(3-t[j]) + 2*B1*t[j]*(3-pow(t[j],2)) +
                2*B2*pow(t[j],2)*(3-2*t[j]) + 2*(1-pow(t[j],3)))/d
            b3 = (2*pow(t[j],3))/d

            r_x.append(b0*P_x[i-1] + b1*P_x[i] + b2*P_x[i+1] + b3*P_x[i+2])
            r_y.append(b0*P_y[i-1] + b1*P_y[i] + b2*P_y[i+1] + b3*P_y[i+2])

    r_x.append(P_x[6])
    r_y.append(P_y[6])

    plt.title("Elementary B-spline curve")
    return plt.plot(r_x,r_y,color="blue")

plt.subplot(2, 2, 1)
Cubic_Bezier_curve_7points(50)

plt.subplot(2, 2, 2)
Catmull_Rom_spline_interpolation(50)

plt.subplot(2, 2, 3)
Cubic_Beta_Spline_curve(10)

plt.subplot(2, 2, 4)
Elementary_Beta_spline_curve(10,1,0.1)

plt.show()
```

Скріншоти сплайнових кривих

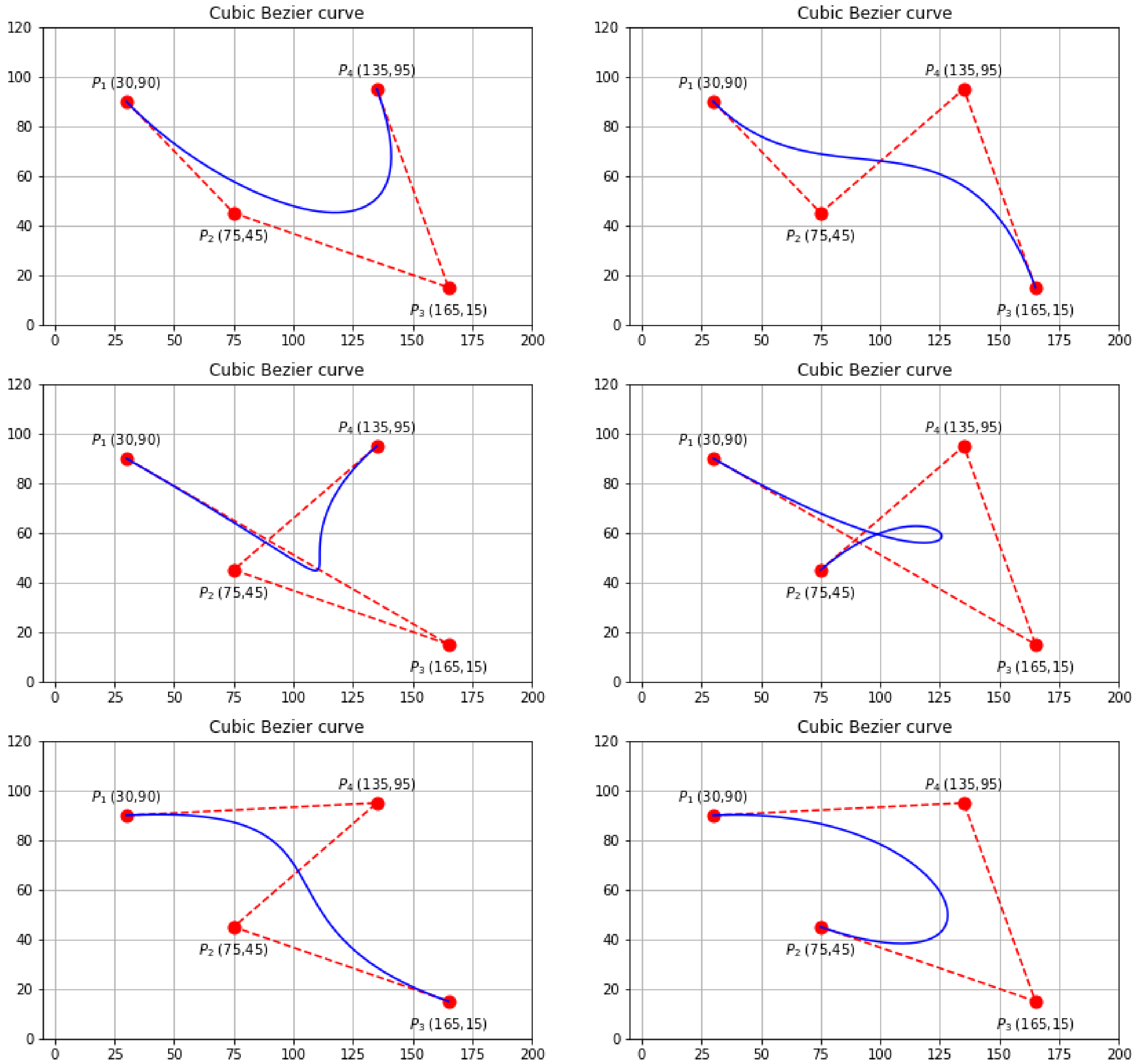


Дослідження залежності виду кривої Безьє

```
import matplotlib.pyplot as plt

plt.figure(figsize=(14, 13.5))

def plot_scatter_4points(P_x,P_y,a,b,c,d):
    i = 1
    for x,y in zip(P_x,P_y):
        label = f"$P_{i}$ ({x},{y})"
        if (i == 1 or i == 4):
            plt.annotate(label,
                        (x,y),
                        textcoords="offset points",
                        xytext=(0,10),
                        ha="center")
        if (i == 2 or i == 3):
            plt.annotate(label,
                        (x,y),
                        textcoords="offset points",
                        xytext=(0,-20),
                        ha="center")
    i = i + 1

```

```
    plt.grid()
    plt.xlim(-5,200)
    plt.ylim(0,120)

    P_x = [P_x[a], P_x[b], P_x[c], P_x[d]]
    P_y = [P_y[a], P_y[b], P_y[c], P_y[d]]

    return plt.plot(P_x, P_y, 'o--r', markersize = 9)

def Cubic_Bezier_curve_4points(P_x,P_y,n,a,b,c,d):
    plot_scatter_4points(P_x,P_y,a,b,c,d)

    r_x = []
    r_y = []

    t = []
    t.append(0)

    for i in range(0, n):
        t.append(t[i] + 1/n)

    for i in range(0, len(t)):
        dt = 1-t[i]
        r_x.append(pow(dt,3)*P_x[a] + 3*pow(dt,2)*t[i]*P_x[b] +
            3*dt*pow(t[i],2)*P_x[c] + pow(t[i],3)*P_x[d])
        r_y.append(pow(dt,3)*P_y[a] + 3*pow(dt,2)*t[i]*P_y[b] +
            3*dt*pow(t[i],2)*P_y[c] + pow(t[i],3)*P_y[d])

    plt.title("Cubic Bezier curve")
    return plt.plot(r_x,r_y,color="blue")

P_x = [30, 75, 165, 135]
P_y = [90, 45, 15, 95]

plt.subplot(3, 2, 1)
Cubic_Bezier_curve_4points(P_x,P_y,50,0,1,2,3)

plt.subplot(3, 2, 2)
Cubic_Bezier_curve_4points(P_x,P_y,50,0,1,3,2)

plt.subplot(3, 2, 3)
Cubic_Bezier_curve_4points(P_x,P_y,50,0,2,1,3)

plt.subplot(3, 2, 4)
Cubic_Bezier_curve_4points(P_x,P_y,50,0,2,3,1)

plt.subplot(3, 2, 5)
Cubic_Bezier_curve_4points(P_x,P_y,50,0,3,1,2)

plt.subplot(3, 2, 6)
Cubic_Bezier_curve_4points(P_x,P_y,50,0,3,2,1)

plt.show()
```

Скріншоти кривих Безьє



Висновки

У лабораторному практикумі я навчився будувати такі чотири різних типи сплайнових кривих: інтерполяційна крива Catmull-Rom, кубічна крива Безьє, кубічна B-сплайнова крива та елементарна B-сплайнова крива. Здобув практичні навички кодування побудови кривих на мові python. Крім того, засвоїв чимало нових елементів графічної бібліотеки побудови графіків `matplotlib`. Намалював графіки сплайнових кривих й зобразив усі проміжні результати на скріншотах.

Контрольні питання

1. *Що таке сплайни? Де застосовуються сплайни?*

   Як і алгоритм цифрового диференціального аналізатора, алгоритм Брезенхейма теж використовують для пошуку оптимальних растрових координат для представлення відрізку між двома точками.

2. *Які сплайнові криві ви знаєте? Яким умовам повинні задовольняти сплайни?*

   Як і алгоритм цифрового диференціального аналізатора, алгоритм Брезенхейма теж використовують для пошуку оптимальних растрових координат для представлення відрізку між двома точками.

3. *Як побудувати складену сплайнову криву?*

   Як і алгоритм цифрового диференціального аналізатора, алгоритм Брезенхейма теж використовують для пошуку оптимальних растрових координат для представлення відрізку між двома точками.

4. *Назвіть властивості складеної сплайнової кривої Catmull-Rom.*

   Як і алгоритм цифрового диференціального аналізатора, алгоритм Брезенхейма теж використовують для пошуку оптимальних растрових координат для представлення відрізку між двома точками.

5. *Назвіть властивості кривих Безьє. Які переваги і недоліки кривих Безьє?*

   Як і алгоритм цифрового диференціального аналізатора, алгоритм Брезенхейма теж використовують для пошуку оптимальних растрових координат для представлення відрізку між двома точками.

6. *Назвіть властивості кубічних B-сплайнів.*

   Як і алгоритм цифрового диференціального аналізатора, алгоритм Брезенхейма теж використовують для пошуку оптимальних растрових координат для представлення відрізку між двома точками.

7. *Назвіть властивості складених елементарних B-сплайнових кривих.*

   Як і алгоритм цифрового диференціального аналізатора, алгоритм Брезенхейма теж використовують для пошуку оптимальних растрових координат для представлення відрізку між двома точками.