Министерство образования Республики Беларусь
Учреждение образования “Белорусский государственный университет информатики и радиоэлектроники”

Факультет информационных технологий и управления
Кафедра интеллектуальных информационных технологий
Дисциплина: Графический интерфейс интеллектуальных систем

# Отчёт к лабораторной работе №3

Выполнил: Гафаров М.С.
Группа: 221702
Проверила: Жмырко А.В.

Минск 2024

# Лабораторная работа №3
# Интерполяция и аппроксимация кривых

**Цель:** изучить и применить на практике основные алгоритмы интерполяции кривых.

**Задание:** разработать элементарный графический редактор, реализующий построение параметрических кривых, используя форму Эрмита, форму Безье и B-сплайн. Выблр метода задаётся из пункта меню и доступен через панель инструментов “Кривые”. В редакторе должен быть предусмотрен режим корректировки опорных точек и состыковки сегментов. В программной реализации необходимо реализовать базовые функции матричных вычислений.

**Теоретические сведения:**
В лабораторной работе рассмотрено 3 базовых метода для интерполяции и аппроксимации кривых:

**1. Метод Эрмита**

В интерполяционном методе Эрмита по функции $f$ подбирается многочлен, интерполирующий в заданном числе узлов не только саму функцию $f$, но и заданное число последовательных производных $f$. Например, существует только один многочлен третьей степени, который называется кубическим сплайном:

$p_3 = at^3 + bt^2 + ct + d$

**2. Формы Безье**

Кубические сплайны не являются удобными в работе, т.к. пользователю трудно с их помощью задавать сегменты кривой. Построение кривой с помощью форм Безье приводится к решению маричного уравнения:

$$[x(t)\ \ y(t)] = [t^3\ t^2\ t\ \ 1] * \begin{matrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{matrix} * \begin{matrix} P1x & P1y \\ P2x & P2y \\ P3x & P3y \\ P4x & P4y \end{matrix}$$

**3. B-сплайн**

Сглаживание кривых методом B-сплайна подразумевает расширение множества точек путём добавления в конец трёх следующих: $P_{n+1} = P_0$, $P_{n+2} = P_1$, $P_{n+3} = P_2$. В результате расширения полученное множество граничных условий разбивается на сегменты по 4 условия каждый. За счёт того, что что получаемые кривые не проходят точно через заданные точки, B-сплайн обеспечивает построение более гладких кривых по сравнению с другими способами построения.

**Программная реализация:**
**Метод Эрмита:**
**Основной алгоритм:**

```
def ermit(P1, P4, R1, R4):  7 usages
    t = 0.0
    highest_point = 1.0
    x_values, y_values, G = list(), list(), list()  # G - вектор Эрмитовой геометрии
    M = [[2, -2, 1, 1], [-3, 3, -2, -1], [0, 0, 1, 0], [1, 0, 0, 0]]
    P1 = list(map(int, P1))
    P4 = list(map(int, P4))
    R1 = list(map(int, R1))
    R4 = list(map(int, R4))
    G = [P1, P4, R1, R4]
    while t <= highest_point:
        T = [[t**3, t**2, t, 1]]
        P = matrix_multiplication(T, matrix_multiplication(M, G))
        x, y = P[0][0], P[0][1]
        x_values.append(x)
        y_values.append(y)
        t += 0.1
    return x_values, y_values
```

**Построение графика:**

```
def graphic(x_values, y_values):
    plt.figure(figsize=(8, 6))
    plt.plot(x_values, y_values, label="Кривая Эрмита", color="gray", linewidth=2)
    plt.scatter(x_values, y_values, color="red", zorder=5)

    plt.title("График кривой Эрмита", fontsize=16)
    plt.xlabel("x(t)", fontsize=14)
    plt.ylabel("y(t)", fontsize=14)
    plt.grid(True, linestyle="--", alpha=0.6)
    plt.legend(fontsize=12)
    plt.show()
```

**Формы Безье:**
**Основной алгоритм:**

```python
def bezier(P1, P2, P3, P4):  7 usages
    t = 0.0
    highest_point = 1.0
    x_values, y_values = list(), list()
    M = [[-1, 3, -3, 1], [3, -6, 3, 0], [-3, 3, 0, 0], [1, 0, 0, 0]]
    P1 = list(map(int, P1))
    P2 = list(map(int, P2))
    P3 = list(map(int, P3))
    P4 = list(map(int, P4))
    G = [P1, P2, P3, P4]
    while t <= highest_point:
        T = [[t**3, t**2, t, 1]]
        P = matrix_multiplication(T, matrix_multiplication(M, G))
        x = P[0][0]
        y = P[0][1]
        x_values.append(x)
        y_values.append(y)
        t += 0.1
    return x_values, y_values
```

**Построение графика:**

```python
def graphic(x_values, y_values):
    plt.figure(figsize=(8, 6))
    plt.plot(x_values, y_values, label="Кривая Безье", color="gray", linewidth=2)
    plt.scatter(x_values, y_values, color="red", zorder=5)

    plt.title("График кривой Безье", fontsize=16)
    plt.xlabel("x(t)", fontsize=14)
    plt.ylabel("y(t)", fontsize=14)
    plt.grid(True, linestyle="--", alpha=0.6)
    plt.legend(fontsize=12)
    plt.show()
```

**B-сплайн:**

**Основной алгоритм:**

```python
def b_spline(values):  7 usages
    x_values, y_values, segments = list(), list(), list()
    P0, P1, P2 = values[0], values[1], values[2]
    M = [[-1, 3, -3, 1], [3, -6, 3, 0], [-3, 0, 3, 0], [1, 4, 1, 0]]
    values.append(P0)
    values.append(P1)
    values.append(P2)
    values = [list(map(int, sublist)) for sublist in values]
    for i in range(len(values) - 3):
        temp_list = list()
        temp_list.append(values[i])
        temp_list.append(values[i + 1])
        temp_list.append(values[i + 2])
        temp_list.append(values[i + 3])
        segments.append(temp_list)
    for el in segments:
        P0 = el[0]
        P1 = el[1]
        P2 = el[2]
        P3 = el[3]
        G = [P0, P1, P2, P3]
        t = 0.0
        while t <= 1.0:
            T = [[t**3, t**2, t, 1]]
            P = matrix_multiplication(T, matrix_multiplication(M, G))
            P = [[item / 6 for item in sublist] for sublist in P]
            x = P[0][0]
            y = P[0][1]
            x_values.append(x)
            y_values.append(y)
            t += 0.1
    return x_values, y_values
```

**Построение графика:**

```python
def graphic(x_values, y_values):
    plt.figure(figsize=(8, 6))
    plt.plot(x_values, y_values, color="gray", linewidth=2)
    plt.scatter(x_values, y_values, color="red", zorder=5)

    plt.title("График B-сплайна", fontsize=16)
    plt.xlabel("x(t)", fontsize=14)
    plt.ylabel("y(t)", fontsize=14)
    plt.grid(True, linestyle="--", alpha=0.6)
    plt.legend(fontsize=12)
    plt.show()
```

**Вывод:** в результате лабораторной работы был спроектирован элементарный графический редактор для интерполяции и аппроксимации кривых. Были изучены и запрограммированы основные методы для интерполяции и аппроксимации кривых: метод Эрмита, формы Безье, B-сплайн.