УДК 514.144.14

КРИВЫЕ И ПОВЕРХНОСТИ ВТОРОГО ПОРЯДКА

Яковук Н.В., студент

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Примичева З.Н. – канд. физ.-мат. наук, доцент

Аннотация. В работе рассмотрены кривые и поверхности второго порядка, их основные уравнения и виды. Показано их применение при решении инженерных задач: построении переходных кривых, расчете зеркал и сопряжении линий.

Ключевые слова. Кривые второго порядка, поверхности второго порядка, эллипс, гипербола, парабола, эллипсоид, параболоид, гиперболоид.

Введение. Аналитическая геометрия связывает геометрические представления с алгебраическими методами, что позволяет удобно описывать и исследовать различные формы. Среди таких объектов важно место занимают кривые и поверхности второго порядка, задаваемые уравнениями второй степени. Кривые второго порядка включают эллипс, гиперболу и параболу — фигуры, которые возникают при сечении конуса плоскостью и обладают характерными геометрическими свойствами. Их пространственным продолжением являются поверхности второго порядка, используемые для описания более сложных форм. Эти объекты находят применение в задачах моделирования, проектирования и анализа, где требуется точное описание геометрии.

Основная часть. Кривые второго порядка задаются уравнением:

Формулы (включая химические) выносят отдельной строкой, отделяя пробельными строками с обеих сторон и нумеруют. Пояснения к формулам размещают без абзацного отступа начиная со строчной буквы, перечисление идет через точку с запятой:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

Тип кривой определяется дискриминантом:

$$\Delta = B^2 - 4AC$$

Возможны три случая:
1. Δ < 0 — эллипс.
2. Δ = 0 — парабола.
3. Δ > 0 — гипербола.

После поворота и переноса уравнение приводится к каноническому виду:
Эллипс:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Эллипс имеет фокальное расстояние и эксцентриситет равные:

$$c^2 = a^2 - b^2$$

$$e = \frac{c}{a}$$

где 0 < e < 1.

Гипербола:

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$

Гипербола так же имеет фокальное расстояние и эксцентриситет, которые можно получить по следующим формулам:

$$c^2 = a^2 - b^2$$

$$e = \frac{c}{a}$$

где e > 1.

Парабола:

$$y^2 = 2px$$

Или

$$x^2 = 2py$$

.

Фокус параболы располагается в точке (0, p/2). А директриса находится по формуле:

$$y = -\frac{p}{2}$$

Графики кривых второго порядка можно увидеть на рисунке 1:

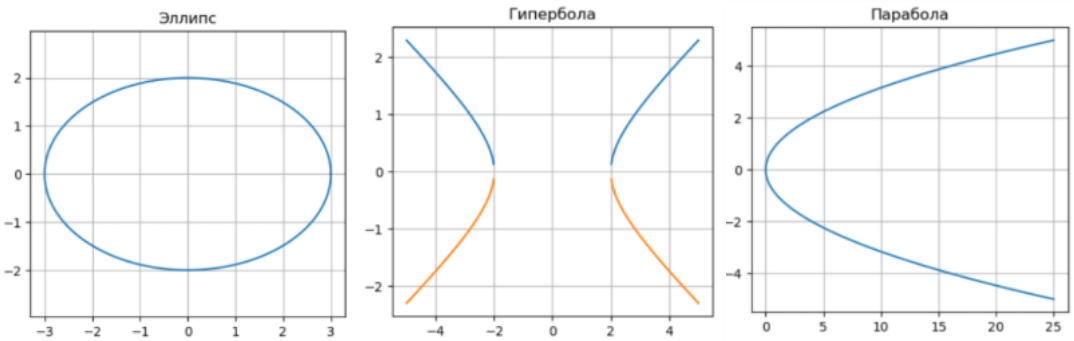

Рисунок 1 – Графики кривых второго порядка, построенных с помощью Python

Поверхности второго порядка имеют общее уравнение:

$$Ax^2 + By^2 + Cz^2 + Dxy + Exz + Fyz + Gx + Hy + Iz + J = 0$$

После преобразований формулы можно получить канонические формы следующих поверхностей:

Эллипсоид:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$$

Однополостный гиперболоид:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$$

Двуполостный гиперболоид:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = -1$$

Эллиптический параболоид:

$$z = \frac{x^2}{a^2} + \frac{y^2}{b^2}$$

Гиперболический параболоид:

$$z = \frac{x^2}{a^2} - \frac{y^2}{b^2}$$

Графики поверхностей второго порядка представлены на рисунке 2.

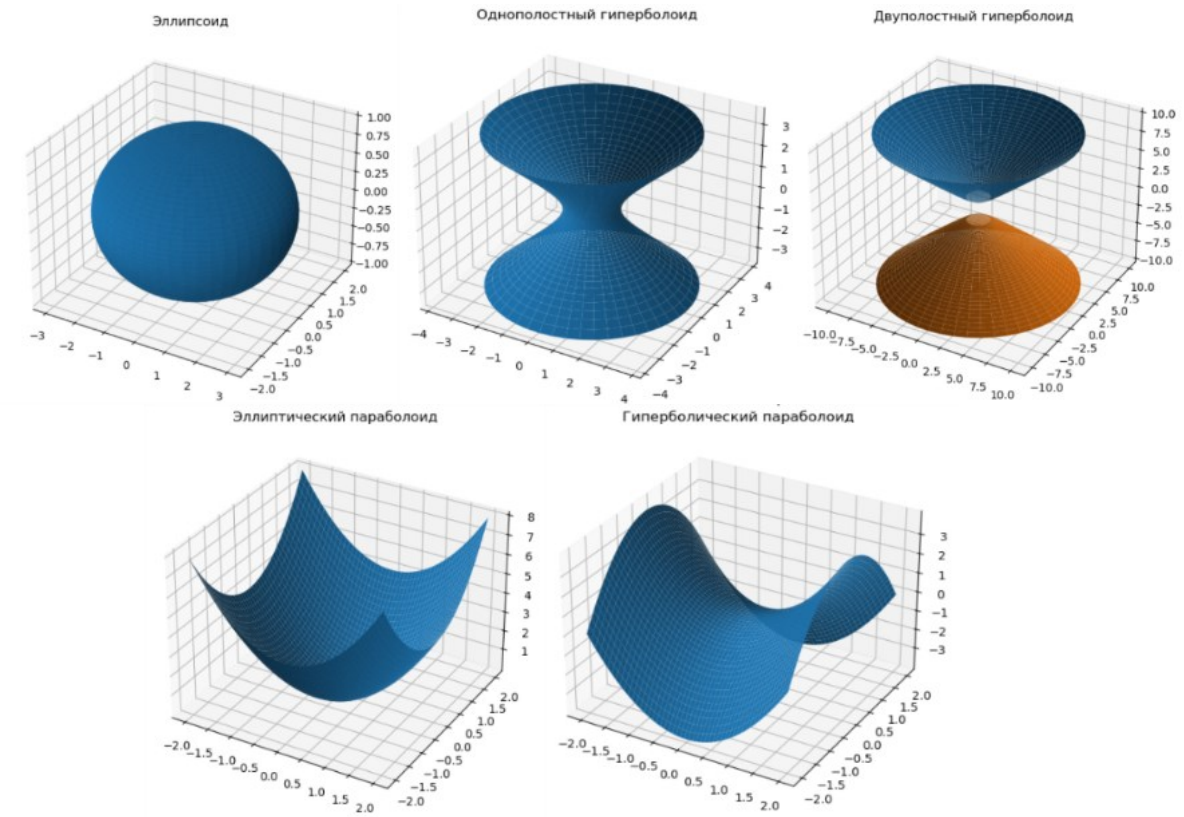

Рисунок 2 – Графики поверхностей второго порядка, построенных с помощью Python

Применение кривых и поверхностей второго порядка.

Постановка задачи. В инженерной практике часто возникает необходимость сопряжения различных криволинейных участков с обеспечением заданных геометрических характеристик: непрерывности кривизны, касания заданного порядка, прохождения через фиксированные точки. В данном разделе рассматриваются три характерные задачи, демонстрирующие применение аппарата кривых второго порядка.

Первая задача связана с проектированием переходной кривой гоночной трассы (клотоиды). Вторая задача посвящена расчету геометрических параметров гиперболического зеркала телескопа. Третья задача рассматривает сопряжение двух окружностей кривой второго порядка, что используется, например, при формировании обводов кузова автомобиля.

Методология исследования. Для решения поставленных задач применяются методы аналитической геометрии. Основными из них являются параметрическое представление кривых, анализ кривизны и условий ее непрерывности, решение систем уравнений, описывающих условия касания и прохождения кривых через заданные точки, а также классификация кривых второго порядка по значению дискриминанта.

Задача 1. Проектирование переходной кривой трассы.

Постановка задачи. При сопряжении прямолинейного участка трассы с круговым поворотом радиуса R = 50 м возникает скачкообразное изменение центростремительной силы, что недопустимо с точки зрения безопасности и комфорта движения. Требуется построить переходную кривую длины L = 60 м, обеспечивающую линейное нарастание кривизны от 0 до 1/R.

Математическая модель. Я использовал клотоиду (спираль Корню), для которой кривизна определяется выражением:

$$k(s) = \frac{s}{A^2}$$

где A – параметр кривой.

Решение. Я учел, что при s = L кривизна должна совпадать с кривизной окружности. Тогда

$$\frac{L}{A^2} = \frac{1}{R}$$

Отсюда получаем

$$A = \sqrt{L \cdot R} = \sqrt{60 \cdot 50} \approx 54{,}77 \text{ м}$$

Координаты точек клотоиды выражаются через интегралы Френеля:

$$x(t) = \int cos(u^2)du$$

$$y(t) = \int sin(u^2)du$$

Численные значения этих интегралов табулированы используются для практических расчетов.

Результат. В результате я определил параметр клотоиды A ≈ 54,77 м, который обеспечивает плавное изменение кривизны и исключает резкие динамические нагрузки.

Для решения задачи в общем виде была разработана следующая программа на языке C++:

```cpp
#include <iostream>
#include <cmath>

// Интеграл cos(u²)
double fresnel_cos(double t, int n = 1000) {
    double h = t / n;
    double sum = 0.0;

    for (int i = 0; i < n; i++) {
        double u1 = i * h;
        double u2 = (i + 1) * h;
        sum += (cos(u1 * u1) + cos(u2 * u2)) * 0.5 * h;
    }
    return sum;
}

// Интеграл sin(u²)
double fresnel_sin(double t, int n = 1000) {
    double h = t / n;
    double sum = 0.0;

    for (int i = 0; i < n; i++) {
        double u1 = i * h;
        double u2 = (i + 1) * h;
        sum += (sin(u1 * u1) + sin(u2 * u2)) * 0.5 * h;
    }
    return sum;
}

int main() {
    double L, R;
    std::cout << "Введите длину переходной кривой L: ";
    std::cin >> L;

    std::cout << "Введите радиус R: ";
    std::cin >> R;

    if (L <= 0 || R <= 0) {
        std::cout << "Ошибка: параметры должны быть положительными\n";
        return 1;
    }

    double A2 = L * R;
    double A = sqrt(A2);

    std::cout << "\nПараметр клотоиды:\n";
    std::cout << "A² = " << A2 << "\n";
    std::cout << "A = " << A << "\n";

    // Ввод параметра t (нормированная длина)
    double t;
    std::cout << "\nВведите параметр t (например 0..2): ";
    std::cin >> t;

    double x = A * fresnel_cos(t);
    double y = A * fresnel_sin(t);

    std::cout << "\nКоординаты точки:\n";
    std::cout << "x = " << x << "\n";
    std::cout << "y = " << y << "\n";

    return 0;
}
```

Рисунок 3 – Код на языке C++ для решения задачи 1 при параметрах, задаваемых пользователем

Задача 2. Расчет гиперболического зеркала телескопа Кассегрена.

Постановка задачи. В телескопе системы Кассегрена вторичное зеркало выполняется в форме выпуклой гиперболы. Один фокус гиперболы совпадает с фокусом главного параболического зеркала, а второй фокус находится в точке формирования изображения. Расстояние между фокусами гиперболы равно 2c = 1,5 м. Расстояние от вершины гиперболы до ближнего фокуса составляет a = 0,5 м.

Математическая модель. Я использовал каноническое уравнение гиперболы и формулу эксцентриситета.

Решение. Из условий получаем

$$c = \frac{1{,}5}{2} = 0{,}75 \text{ м}$$

Тогда из соотношения a, b и c найдем

$$b^2 = c^2 - a^2 = 0{,}75^2 - 0{,}5^2 = 0{,}3125$$

Эксцентриситет гиперболы равен

$$e = \frac{0{,}75}{0{,}5} = 1{,}5$$

Результат. В результате я получил уравнение гиперболы:

$$\frac{x^2}{0{,}25} + \frac{y^2}{0{,}3125} = 1$$

Для решения задачи в общем виде была разработана следующая программа на языке C++:

```cpp
#include <iostream>
#include <cmath>

int main() {
    double c2, a;

    std::cout << "Введите расстояние между фокусами (2c): ";
    std::cin >> c2;

    std::cout << "Введите параметр a: ";
    std::cin >> a;

    double c = c2 / 2.0;

    if (c <= a) {
        std::cout << "Ошибка: должно выполняться c > a\n";
        return 1;
    }

    double b2 = c * c - a * a;
    double b = sqrt(b2);
    double e = c / a;

    std::cout << "\nРезультаты:\n";
    std::cout << "c = " << c << "\n";
    std::cout << "b² = " << b2 << "\n";
    std::cout << "b = " << b << "\n";
    std::cout << "Эксцентриситет e = " << e << "\n";

    std::cout << "\nУравнение гиперболы:\n";
    std::cout << "x^2 / " << a * a << " - y^2 / " << b2 << " = 1\n";

    return 0;
}
```

Рисунок 4 – Код на языке C++ для решения задачи 2 при параметрах, задаваемых пользователем

Задача 3. Сопряжение двух окружностей кривой второго порядка.

Постановка задачи. При проектировании кузова автомобиля необходимо соединить две дуги окружностей единой кривой второго порядка, обеспечив гладкость перехода (касание) в точках сопряжения.

Исходные данные. Окружность 1 имеет радиус R_1 = 100 мм и центр $O_1(0; 0)$. Точка сопряжения M(70; 71,4) лежит на первой окружности. Производная в этой точке равна

$$y`(M) = -\frac{x}{y}$$

$$y`(M) = -\frac{70}{71{,}4} \approx -0{,}98$$

Окружность 2 имеет радиус R_2 = 60 мм и центр $O_2(180; 40)$. Точка сопряжения N(150; 91,6) лежит на первой окружности. Производная в этой точке равна

$$y`(N) = -\frac{x - x_0}{y - y_0}$$

$$y`(N) = -\frac{150 - 180}{91{,}6 - 40} = -\frac{-30}{51{,}6} \approx 0{,}58$$

Математическая модель. Искомая кривая описывается общим уравнением второго порядка. Для упрощения решения я задал условия нормировки, что позволяет получить единственное решение системы уравнений.

Для определения коэффициентов используются следующие условия:

1. Кривая проходит через точку M.
2. Кривая проходит через точку N.
3. В точке M выполняется условие касания, то есть производная совпадает с производной окружности.
4. В точке N также выполняется условие касания.

Производная неявной функции определяется формулой:

$$\mathrm{y`} = -\frac{2\mathrm{Ax} + \mathrm{By} + \mathrm{D}}{\mathrm{Bx} + 2\mathrm{Cy} + \mathrm{E}}$$

Решение. Я подставил координаты точек M и N, а также значения производных, после чего получил систему уравнений. Решив ее (при фиксированных $A = D = 1$), я определил коэффициенты B, C, E и F:

$$A = 1$$

$$B = -2{,}45$$

$$C = 2{,}48$$

$$D = 1$$

$$E = -217{,}72$$

$$F = 10184{,}30$$

Для определения типа кривой вычисляется дискриминант по формуле. Подставляя найденные значения коэффициентов, получаем

$$\Delta = (-2{,}45)^2 - 4 \cdot 1 \cdot 2{,}48 \approx -3{,}80$$

Так как Δ < 0, я сделал вывод, что полученная кривая является эллипсом.

Результат. Получено уравнение эллипса, проходящего через точки M и N и касающегося заданных окружностей. Подобный метод применяется в системах автоматизированного проектирования для построения гладких обводов сложной формы.

Для решения задачи в общем виде была разработана следующая программа на языке C++:

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    double x1, y1, R1, x2, y2, R2;
    cout << "Введите координаты центра первой окружности (x1; y1): "; cin >> x1 >> y1;
    cout << "Введите радиус первой окружности R1 = "; cin >> R1;
    cout << "Введите координаты центра второй окружности (x2; y2): "; cin >> x2 >> y2;
    cout << "Введите радиус первой окружности R2 = "; cin >> R2;

    double xn, yn, xm, ym;
    const double EPS = 0.5;

    cout << "Введите координаты точки сопряжения, лежащей на первой окружности (xn; yn): "; cin >> xn >> yn;

    double dist1 = hypot(xn - x1, yn - y1);
    if (fabs(dist1 - R1) >= EPS) {
        cout << "Ошибка: точка (" << xn << "; " << yn << ") не лежит на первой окружности!" << endl;
        cout << "Расстояние до центра = " << dist1 << ", радиус = " << R1 << endl;
        return 1;
    }

    cout << "Введите координаты точки сопряжения, лежащей на первой окружности (xn; yn): "; cin >> xm >> ym;

    double dist2 = hypot(xn - x2, yn - y2);
    if (fabs(dist2 - R2) >= EPS) {
        cout << "Ошибка: точка (" << xn << "; " << yn << ") не лежит на второй окружности!" << endl;
        cout << "Расстояние до центра = " << dist2 << ", радиус = " << R2 << endl;
        return 1;
    }

    // Производные
    double k1 = -(xm - x1) / (ym - y1);
    double k2 = -(xn - x2) / (yn - y2);

    cout << "y'(M) = " << k1 << endl;
    cout << "y'(N) = " << k2 << endl;

    // Система для B, C, D, E
    double a[4][5];

    // (2)-(1) - убрали F
    a[0][0] = xn * yn - xm * ym;      // B
    a[0][1] = yn * yn - ym * ym;      // C
    a[0][2] = xn - xm;                // D
    a[0][3] = yn - ym;                // E
    a[0][4] = -(xn * xn - xm * xm);   // RHS

    // касание в M
    a[1][0] = xm * k1 + ym;
    a[1][1] = 2 * k1 * ym;
    a[1][2] = 1;
    a[1][3] = k1;
    a[1][4] = -2 * xm;

    // касание в N
    a[2][0] = xn * k2 + yn;
    a[2][1] = 2 * k2 * yn;
    a[2][2] = 1;
    a[2][3] = k2;
    a[2][4] = -2 * xn;

    // дополнительное уравнение (например D = 1 - нормировка)
    a[3][0] = 0;
    a[3][1] = 0;
    a[3][2] = 1;
    a[3][3] = 0;
    a[3][4] = 1;

    int n = 4;

    // Метод Гаусса
    for (int i = 0; i < n; i++) {
        int maxRow = i;
        for (int k = i + 1; k < n; k++) {
            if (abs(a[k][i]) > abs(a[maxRow][i]))
                maxRow = k;
        }

        // swap
        for (int j = i; j <= n; j++) {
            double tmp = a[i][j];
            a[i][j] = a[maxRow][j];
            a[maxRow][j] = tmp;
        }

        double pivot = a[i][i];

        for (int j = i; j <= n; j++)
            a[i][j] /= pivot;

        for (int k = 0; k < n; k++) {
            if (k == i) continue;

            double factor = a[k][i];
            for (int j = i; j <= n; j++)
                a[k][j] -= factor * a[i][j];
        }
    }

    double B = a[0][4];
    double C = a[1][4];
    double D = a[2][4];
    double E = a[3][4];

    // Находим F
    double F = -(xm * xm + B * xm * ym + C * ym * ym + D * xm + E * ym);

    cout << fixed << setprecision(4);

    cout << "\nКоэффициенты:\n";
    cout << "A = 1\n";
    cout << "B = " << B << endl;
    cout << "C = " << C << endl;
    cout << "D = " << D << endl;
    cout << "E = " << E << endl;
    cout << "F = " << F << endl;

    double delta = B * B - 4 * C;

    cout << "\nΔ = " << delta << endl;

    if (delta < 0)
        cout << "Тип: эллипс\n";
    else if (delta > 0)
        cout << "Тип: гипербола\n";
    else
        cout << "Тип: парабола\n";

    return 0;
}
```

Рисунок 5 – Код на языке C++ для решения задачи 3 при параметрах, задаваемых пользователем

Заключение. В работе рассмотрены основные виды кривых и поверхностей второго порядка, их уравнения и геометрические свойства. Показана эффективность применения аналитической геометрии для решения инженерных задач: построения переходных кривых, расчета зеркал и сопряжения линий. Разработанные программные реализации на C++ подтверждают практическую ценность рассмотренных методов.

Список использованных источников:
1. Атанасян Л.С. Геометрия. – Москва: Просвещение, 2019.
2. Погорелов А.В. Аналитическая геометрия. – Мосва: Наука, 1968.
3. Ильин В.А., Позняк Э.Г. Аналитическая геометрия. – Москва: Физмалит, 2005.
4. Киселев А.П. Геометрия. – Москва: Физмалит, 2004.

UDC 514.144.14

CURVES AND SURFACES OF THE SECOND ORDER

Yakovuk N.V., student

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Primicheva Z.N. – PhD in Physics and Mathematics

Annotation. The article discusses curves and surfaces of the second order, their basic equations and types. It also shows their application in solving engineering problems, such as the construction of transition curves, the calculation of mirrors and matching lines.

Keywords. Second-order curves, second-order surfaces, ellipse, hyperbola, parabola, ellipsoid, paraboloid, hyperboloid.