Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Линейная Алгебра

расчетно-графической работа №2 на тему:

Векторы и аналитическая геометрия с в прикладных задачах

Выполнили: студенты группы 10.1
Эллити Мохамед Эмад Ахмед Авад
Махфудх Ахмед Айнин
Рахматов Мирзобаходур
Маюсупов Азимджон
Эркаев Азизджон
Преподаватели:
Кольцова Т.Б.

Санкт-Петербург
2025г.

Практическая работа: Векторы и аналитическая геометрия в прикладных задачах (Задача принадлежности точки тетраэдру)

Содержание

1 Цель

Сравнить различные методы решения задачи принадлежности точки к тетраэдру, определить наиболее удобный и универсальный из них.

2 Постановка задачи

Определить, принадлежит ли заданная точка $P(x, y, z)$ тетраэдру $ABCD$. Исследовать случаи: внутри; на грани; на ребре; совпадение с вершиной; снаружи.

3 Вершины пирамиды

3.1 Таблица координат вершин

Вершина	Координаты
A	$(0; 0; 0)$
B	$(1; 0; 0)$
C	$(0; 1; 0)$
D	$(0; 0; 1)$

3.2 Проверка корректности

Найдем объем V тетраэдра $ABCD$ через смешанное произведение векторов: $\vec{AB}, \vec{AC}, \vec{AD}$.

$$\vec{AB} = (1, 0, 0), \quad \vec{AC} = (0, 1, 0), \quad \vec{AD} = (0, 0, 1)$$

$$\vec{AB} \cdot (\vec{AC} \times \vec{AD}) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = 1$$

$$V_{ABCD} = \frac{1}{6} \cdot 1 = \frac{1}{6} \neq 0$$

Следовательно, тетраэдр невырожденный.

4 Методы решения

4.1 Метод 1: Метод объёмов (через смешанное произведение)

4.1.1 Основная идея

Вычисляем объём пирамиды $ABCD$ и объёмы пирамид с вершиной в P ($PBCD$, $APCD$, $ABPD$, $ABCP$) с помощью смешанного произведения векторов. Сравниваем сумму объёмов пирамид $PBCD$, $APCD$, $ABPD$, $ABCP$ и объём пирамиды $ABCD$.

4.1.2 Математическая реализация

Объём пирамиды $ABCD$:

$$V_{ABCD} = \frac{|\vec{AB} \cdot (\vec{AC} \times \vec{AD})|}{6}$$

Объёмы пирамид с вершиной P:

$$V_1 = V_{PBCD} = \frac{|\vec{PB} \cdot (\vec{PC} \times \vec{PD})|}{6}$$
$$V_2 = V_{APCD} = \frac{|\vec{AP} \cdot (\vec{AC} \times \vec{AD})|}{6}$$
$$V_3 = V_{APBD} = \frac{|\vec{AP} \cdot (\vec{AB} \times \vec{AD})|}{6}$$
$$V_4 = V_{APBC} = \frac{|\vec{AP} \cdot (\vec{AB} \times \vec{AC})|}{6}$$

Пусть $S = V_1 + V_2 + V_3 + V_4$.

- Если $S = V$, то точка находится внутри тетраэдра
- Если $S = V$ и один $V_i = 0$, то точка лежит на грани
- Если $S = V$ и два $V_i = 0$, то точка лежит на ребре
- Если $S = V$ и три $V_i = 0$, то точка лежит на вершине
- Если $S > V$, то точка лежит снаружи

4.1.3 Решение вручную

Точка $P = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ внутри тетраэдра.

$$V_{ABCD} = \frac{1}{6}$$

$$\vec{PB} = (\frac{3}{4}, -\frac{1}{4}, -\frac{1}{4}), \quad \vec{PC} = (-\frac{1}{4}, \frac{3}{4}, -\frac{1}{4}), \quad \vec{PD} = (-\frac{1}{4}, -\frac{1}{4}, \frac{3}{4})$$

$$\vec{PB} \cdot (\vec{PC} \times \vec{PD}) = \begin{vmatrix} \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} \end{vmatrix} = \frac{1}{4}$$

$$V_{PBCD} = \frac{|1/4|}{6} = \frac{1}{24}$$

Аналогично:

$$V_{APCD} = \frac{1}{24}, \quad V_{APBD} = \frac{1}{24}, \quad V_{APBC} = \frac{1}{24}$$

$$S = \frac{1}{24} + \frac{1}{24} + \frac{1}{24} + \frac{1}{24} = \frac{1}{6} = V_{ABCD}$$

Ни один из объемов не равен 0, значит точка P лежит внутри тетраэдра.

4.1.4 Решение с помощью кода

```
import math

def getVector(v1, v2):
    return (v1[0]-v2[0], v1[1]-v2[1], v1[2]-v2[2])

def vector_product(v1, v2):
    return (v1[1]*v2[2]-v1[2]*v2[1],
            v1[2]*v2[0]-v1[0]*v2[2],
            v1[0]*v2[1]-v1[1]*v2[0])

def scalar_product(v1, v2):
    return v1[0]*v2[0] + v1[1]*v2[1] + v1[2]*v2[2]

def getVolume(w,x,y,z):
    wx = getVector(x, w)
    wy = getVector(y, w)
    wz = getVector(z, w)
    det = scalar_product(wx, vector_product(wy, wz))
    return abs(det) / 6.0

def classify_point_in_tetra(A,B,C,D,P, error_margin=10**(-9)):
    V = getVolume(A,B,C,D)
    if V <= error_margin:
        return "                                                "
    V1 = getVolume(P,B,C,D)
    V2 = getVolume(A,P,C,D)
    V3 = getVolume(A,B,P,D)
    V4 = getVolume(A,B,C,P)
    S = V1 + V2 + V3 + V4

    zeros = sum(1 for v in (V1,V2,V3,V4) if v <= error_margin)
    outside = abs(S - V) > error_margin

    if outside:
        return "                                "

    if zeros == 0:
        return "                                "
    elif zeros == 1:
        return "                                  "
    elif zeros == 2:
        return "                                  "
    elif zeros == 3:
        return "                                      "
    else:
        return "                                            "

A=(0,0,0); B=(1,0,0); C=(0,1,0); D=(0,0,1)
P=(0.25,0.25,0.25)
print(classify_point_in_tetra(A,B,C,D,P))
```

4.1.5 Выводы

Метод объёмов позволяет определить положение точки относительно тетраэдра для всех случаев. Метод хорошо реализуется как вручную, так и программно.

4.2 Метод 2: Метод знаков смешанных произведений

4.2.1 Основная идея

Для каждой грани сравниваем знаки смешанных произведений для точки P и противоположной вершины.

4.2.2 Математическая реализация

Для каждой грани:

- Грань ABC (противоположная D): сравниваем знаки M_{ABCP} и M_{ABCD}
- Грань ABD (противоположная C): сравниваем знаки M_{ABDP} и M_{ABDC}
- Грань ACD (противоположная B): сравниваем знаки M_{ACDP} и M_{ACDB}
- Грань BCD (противоположная A): сравниваем знаки M_{BCDP} и M_{BCDA}

Точка P принадлежит тетраэдру если для всех граней смешанные произведения имеют одинаковый знак или равны нулю.

Классификация:

- Внутри: все M_P и M_{opp} имеют одинаковый ненулевой знак
- Снаружи: хотя бы для одной грани знаки разные
- На грани: ровно одно $M_P = 0$
- На ребре: два $M_P = 0$
- На вершине: три $M_P = 0$

4.2.3 Решение вручную

Точка $P = (1, 0, 0)$ (вершина B).

$$M_{ABCP} = \det \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = 0$$

$$M_{ABCD} = \det \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 1$$

$$M_{ABDP} = \det \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = 0$$

$$M_{ABDC} = \det \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = -1$$

$$M_{ACDP} = \det \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = 1$$

$$M_{ACDB} = \det \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = 1$$

$$M_{BCDP} = \det \begin{bmatrix} 1 & -1 & 0 \\ 0 & -1 & 1 \\ 1 & -1 & 0 \end{bmatrix} = 0$$

$$M_{BCDA} = \det \begin{bmatrix} 1 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & -1 & 0 \end{bmatrix} = 1$$

Нулевых значений: 3. Точка на вершине.

4.2.4 Решение с помощью кода

```python
import numpy as np

def point_in_tetrahedron(P, A, B, C, D):
    vAB = B - A
    vAC = C - A
    vAP = P - A
    vAD = D - A
    M_P_ABC = np.linalg.det([vAB, vAC, vAP])
    M_D_ABC = np.linalg.det([vAB, vAC, vAD])
    M_P_ABD = np.linalg.det([vAB, vAD, vAP])
    M_C_ABD = np.linalg.det([vAB, vAD, vAC])
    M_P_ACD = np.linalg.det([vAC, vAD, vAP])
    M_B_ACD = np.linalg.det([vAC, vAD, vAB])
    vCB = B - C
    vCD = D - C
    vCP = P - C
    vCA = A - C
    M_P_BCD = np.linalg.det([vCB, vCD, vCP])
    M_A_BCD = np.linalg.det([vCB, vCD, vCA])
```

```
    epsilon = 1e-9

    def check_sign(Mp, Mopp):
        if Mopp == 0:
            return False, "                                                    "
        if Mp * Mopp < -epsilon:
            return False, "                "
        return True, "                                    "

    results = [
        check_sign(M_P_ABC, M_D_ABC),
        check_sign(M_P_ABD, M_C_ABD),
        check_sign(M_P_ACD, M_B_ACD),
        check_sign(M_P_BCD, M_A_BCD)
    ]

    if not all(res[0] for res in results):
        return "                "

    zero_counts = 0
    if abs(M_P_ABC) < epsilon: zero_counts += 1
    if abs(M_P_ABD) < epsilon: zero_counts += 1
    if abs(M_P_ACD) < epsilon: zero_counts += 1
    if abs(M_P_BCD) < epsilon: zero_counts += 1

    if zero_counts == 0: return "              "
    elif zero_counts == 1: return "                  "
    elif zero_counts == 2: return "                  "
    elif zero_counts >= 3: return "
                        "


A = np.array([0, 0, 0])
B = np.array([1, 0, 0])
C = np.array([0, 1, 0])
D = np.array([0, 0, 1])

P_inside = np.array([1/4,1/4,1/4])
P_on_face = np.array([1/3, 1/3, 1/3])
P_on_edge = np.array([0.5, 0, 0])
P_on_vertex = np.array([0, 0, 0])
P_outside = np.array([1, 1, 1])

print(f"P_inside: {point_in_tetrahedron(P_inside, A, B, C, D)}")
print(f"P_on_face: {point_in_tetrahedron(P_on_face, A, B, C, D)}")
print(f"P_on_edge: {point_in_tetrahedron(P_on_edge, A, B, C, D)}")
print(f"P_on_vertex: {point_in_tetrahedron(P_on_vertex, A, B, C, D)}")
print(f"P_outside: {point_in_tetrahedron(P_outside, A, B, C, D)}")
```

4.2.5 Выводы

Метод знаков смешанных произведений — надёжный инструмент анализа положения точки относительно тетраэдра. Обеспечивает точное разделение всех возможных случаев.

4.3 Метод 3: Метод уравнений плоскостей

4.3.1 Основная идея

Каждая грань тетраэдра задаётся своей плоскостью. Если точка P находится по ту же сторону каждой плоскости, что и противоположная вершина, то она внутри тетраэдра.

4.3.2 Математическая реализация

Для каждой грани находим уравнение плоскости $Ax + By + Cz + D = 0$.

Для грани ABC (противоположная вершина D):

$$\vec{n} = \vec{AB} \times \vec{AC}$$

$$A(x - x_A) + B(y - y_A) + C(z - z_A) = 0$$

$$D = -(Ax_A + By_A + Cz_A)$$

Вычисляем значение для точки P:

$$F(P) = Ax_P + By_P + Cz_P + D$$

Точка P внутри тетраэдра если для всех граней:

$$\text{sign}(F(P)) = \text{sign}(F(\text{противоположная вершина}))$$

или $F(P) = 0$ (точка на плоскости).

4.3.3 Решение вручную

Точка $P = (0.25, 0.25, 0.25)$.

1. Грань ABC (противоположная D):

$$\vec{AB} = (1, 0, 0), \quad \vec{AC} = (0, 1, 0)$$

$$\vec{n} = (0, 0, 1), \quad D = 0$$

Уравнение: $z = 0$

$$F(P) = 0.25 > 0, \quad F(D) = 1 > 0$$

Знаки совпадают.

2. Грань ABD (противоположная C):

$$\vec{AB} = (1, 0, 0), \quad \vec{AD} = (0, 0, 1)$$

$$\vec{n} = (0, -1, 0), \quad D = 0$$

Уравнение: $-y = 0$ или $y = 0$

$$F(P) = 0.25 > 0, \quad F(C) = 1 > 0$$

Знаки совпадают.

3. Грань ACD (противоположная B):

$$\vec{AC} = (0, 1, 0), \quad \vec{AD} = (0, 0, 1)$$

$$\vec{n} = (1, 0, 0), \quad D = 0$$

Уравнение: $x = 0$

$$F(P) = 0.25 > 0, \quad F(B) = 1 > 0$$

Знаки совпадают.

4. Грань BCD (противоположная A):

$$\vec{CB} = (1, -1, 0), \quad \vec{CD} = (0, -1, 1)$$

$$\vec{n} = (-1, -1, -1), \quad D = 1$$

Уравнение: $-x - y - z + 1 = 0$ или $x + y + z - 1 = 0$

$$F(P) = 0.25 + 0.25 + 0.25 - 1 = -0.25 < 0$$

$$F(A) = 0 + 0 + 0 - 1 = -1 < 0$$

Знаки совпадают.

Все условия выполняются, точка внутри.

4.3.4 Решение с помощью кода

```python
import numpy as np

def plane_equation(v1, v2, v3):
    u = np.array(v2) - np.array(v1)
    w = np.array(v3) - np.array(v1)
    n = np.cross(u, w)
    A, B, C = n
    D = -np.dot(n, v1)
    return A, B, C, D

def check_point_in_tetrahedron(P, A, B, C, D, epsilon=1e-9):
    planes = [
        (plane_equation(A, B, C), D),  #             ABC
        (plane_equation(A, B, D), C),  #             ABD
        (plane_equation(A, C, D), B),  #             ACD
        (plane_equation(B, C, D), A)   #             BCD
    ]

    zero_count = 0
    for (A_coef, B_coef, C_coef, D_coef), opposite_vertex in planes:
        F_P = A_coef*P[0] + B_coef*P[1] + C_coef*P[2] + D_coef
        F_opp = A_coef*opposite_vertex[0] + B_coef*opposite_vertex[1]
           + C_coef*opposite_vertex[2] + D_coef

        if abs(F_P) < epsilon:
            zero_count += 1
            continue

        if F_P * F_opp < -epsilon:
            return "               "

    if zero_count == 0:
        return "            "
    elif zero_count == 1:
```

```
        return "                  "
    elif zero_count == 2:
        return "                  "
    elif zero_count == 3:
        return "                      "
    else:
        return "                                          "

A = (0, 0, 0)
B = (1, 0, 0)
C = (0, 1, 0)
D = (0, 0, 1)

test_points = [
    ((0.25, 0.25, 0.25), "             "),
    ((1/3, 1/3, 1/3), "                BCD"),
    ((0.5, 0, 0), "                AB"),
    ((0, 0, 0), "                  A"),
    ((1, 1, 1), "              ")
]

for point, expected in test_points:
    result = check_point_in_tetrahedron(point, A, B, C, D)
    print(f"           {point}: {result} (                  :
       {expected})")
```

4.3.5 Выводы

Метод уравнений плоскостей интуитивно понятен и легко реализуем. Позволяет определить положение точки относительно тетраэдра через проверку знаков значений в уравнениях плоскостей.

4.4 Метод 4: Метод барицентрических координат

4.4.1 Основная идея

Любую точку внутри тетраэдра можно представить как взвешенную сумму вершин с коэффициентами (барицентрическими координатами) $\alpha, \beta, \gamma, \delta$, такими что:

$$P = \alpha A + \beta B + \gamma C + \delta D$$

где $\alpha + \beta + \gamma + \delta = 1$.

Точка P принадлежит тетраэдру тогда и только тогда, когда все барицентрические координаты неотрицательны.

4.4.2 Математическая реализация

Выразим барицентрические координаты через объёмы тетраэдров:

$$\alpha = \frac{V_{PBCD}}{V_{ABCD}}, \quad \beta = \frac{V_{APCD}}{V_{ABCD}}, \quad \gamma = \frac{V_{ABPD}}{V_{ABCD}}, \quad \delta = \frac{V_{ABCP}}{V_{ABCD}}$$

Где V_{ABCD} — объём исходного тетраэдра.

Условия принадлежности:

- Все $\alpha, \beta, \gamma, \delta \geq 0$
- $\alpha + \beta + \gamma + \delta = 1$

Классификация положения:

- Внутри: все координаты > 0
- На грани: одна координата $= 0$, остальные > 0
- На ребре: две координаты $= 0$, остальные > 0
- На вершине: три координаты $= 0$, одна $= 1$
- Снаружи: хотя бы одна координата < 0

4.4.3 Решение вручную

Точка $P = (0.25, 0.25, 0.25)$.

Из метода 1 мы уже вычислили:

$$V_{ABCD} = \frac{1}{6}, \quad V_{PBCD} = \frac{1}{24}, \quad V_{APCD} = \frac{1}{24}, \quad V_{ABPD} = \frac{1}{24}, \quad V_{ABCP} = \frac{1}{24}$$

Барицентрические координаты:

$$\alpha = \frac{1/24}{1/6} = \frac{1}{4}, \quad \beta = \frac{1/24}{1/6} = \frac{1}{4}, \quad \gamma = \frac{1/24}{1/6} = \frac{1}{4}, \quad \delta = \frac{1/24}{1/6} = \frac{1}{4}$$

Проверяем условия:

- Все координаты $\frac{1}{4} > 0$ — выполняется
- Сумма: $\frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = 1$ — выполняется

Точка внутри тетраэдра.

Проверим точку $P = (0.5, 0, 0)$ (на ребре AB):

$$V_{PBCD} = 0 \quad \text{(точка на плоскости BCD)}$$

$$V_{APCD} = \frac{|\vec{AP} \cdot (\vec{AC} \times \vec{AD})|}{6} = \frac{|0.5 \cdot 1|}{6} = \frac{0.5}{6} = \frac{1}{12}$$

$$V_{ABPD} = 0 \quad \text{(точка на плоскости ABD)}$$

$$V_{ABCP} = \frac{|\vec{AP} \cdot (\vec{AB} \times \vec{AC})|}{6} = \frac{|0.5 \cdot 1|}{6} = \frac{0.5}{6} = \frac{1}{12}$$

$$\alpha = \frac{0}{1/6} = 0, \quad \beta = \frac{1/12}{1/6} = 0.5, \quad \gamma = \frac{0}{1/6} = 0, \quad \delta = \frac{1/12}{1/6} = 0.5$$

Две координаты равны 0, точка на ребре.

4.4.4 Решение с помощью кода

```python
import numpy as np

def barycentric_coordinates(P, A, B, C, D, epsilon=1e-9):
    def tetra_volume(v1, v2, v3, v4):
        v12 = v2 - v1
        v13 = v3 - v1
        v14 = v4 - v1
        det = np.linalg.det([v12, v13, v14])
        return abs(det) / 6.0

    V_total = tetra_volume(A, B, C, D)

    if V_total < epsilon:
        return None, "                                                    "

    alpha = tetra_volume(P, B, C, D) / V_total
    beta = tetra_volume(A, P, C, D) / V_total
    gamma = tetra_volume(A, B, P, D) / V_total
    delta = tetra_volume(A, B, C, P) / V_total

    return [alpha, beta, gamma, delta], None

def classify_by_barycentric(P, A, B, C, D, epsilon=1e-9):
    coords, error = barycentric_coordinates(P, A, B, C, D, epsilon)

    if error:
        return error

    alpha, beta, gamma, delta = coords

    negative_count = sum(1 for c in coords if c < -epsilon)
    zero_count = sum(1 for c in coords if abs(c) < epsilon)
    positive_count = sum(1 for c in coords if c > epsilon)

    if negative_count > 0:
        return "               "

    if zero_count == 0:
        return "            "
    elif zero_count == 1:
        return "               "
    elif zero_count == 2:
        return "                "
    elif zero_count == 3:
        return "                   "
    else:
        return "                                        "

def point_location_barycentric(P, A, B, C, D, epsilon=1e-9):
    coords, error = barycentric_coordinates(P, A, B, C, D, epsilon)

```

```
    if error:
        return error, None

    alpha, beta, gamma, delta = coords

    location = classify_by_barycentric(P, A, B, C, D, epsilon)

    return location, {
        'alpha': alpha,
        'beta': beta,
        'gamma': gamma,
        'delta': delta,
        'sum': alpha + beta + gamma + delta
    }

A = np.array([0, 0, 0])
B = np.array([1, 0, 0])
C = np.array([0, 1, 0])
D = np.array([0, 0, 1])

test_points = [
    (np.array([0.25, 0.25, 0.25]), "            "),
    (np.array([1/3, 1/3, 1/3]), "               "),
    (np.array([0.5, 0, 0]), "                "),
    (np.array([0, 0, 0]), "                  "),
    (np.array([1, 1, 1]), "               "),
    (np.array([0.1, 0.2, 0.3]), "             "),
]

print("
                      :")
print("-" * 50)

for P, expected in test_points:
    location, info = point_location_barycentric(P, A, B, C, D)

    if info:
        print(f"           {P}:")
        print(f"                      : {location} (                  :
           {expected})")
        print(f"                        :   ={info['alpha']:.3f},
             ={info['beta']:.3f},   ={info['gamma']:.3f},
             ={info['delta']:.3f}")
        print(f"             : {info['sum']:.3f}")
        print()
    else:
        print(f"           {P}: {location}")
        print()
```

4.4.5 Выводы

Метод барицентрических координат обладает следующими преимуществами:

1. Прямая геометрическая интерпретация через объёмы

2. Легко определить положение точки (внутри, на грани, на ребре, на вершине)

3. Коэффициенты могут быть использованы для интерполяции свойств (цвет, текстура) в компьютерной графике

4. Устойчив к вычислительным погрешностям при использовании относительных объёмов

Недостатки:

1. Требует вычисления 5 объёмов (4 маленьких и 1 полного)

2. Менее эффективен чем метод уравнений плоскостей для множественных проверок

3. Чувствителен к вычислительным погрешностям при малых объёмах

4.5 Тестовые точки

Координаты точки P	Расположение	Примечание
$(0.25, 0.25, 0.25)$	Внутри	Центр масс
$(0.33, 0.33, 0.33)$	На грани	На плоскости $x + y + z = 1$
$(0.5, 0, 0)$	На ребре	На ребре AB
$(0, 0, 0)$	Вершина	Вершина A
$(1, 1, 1)$	Снаружи	За пределами тетраэдра
$(0, 0.5, 0.5)$	На грани	На плоскости $x = 0$
$(0.5, 0.5, 0)$	На грани	На плоскости $z = 0$
$(0, 0, 0.5)$	На ребре	На ребре AD
$(0.1, 0.2, 0.3)$	Внутри	Произвольная внутренняя точка
$(0.7, 0.1, 0.1)$	Снаружи	За гранью $x + y + z = 1$

4.6 Сравнение методов

4.7 Вывод

Наиболее удобным и универсальным является **Метод уравнений плоскостей**, так как он:

1. Легко обобщается на произвольные выпуклые многогранники

2. Интуитивно понятен геометрически

3. Широко применяется в компьютерной графике для отсечения

4. Позволяет легко определить расстояние до граней при необходимости

Метод барицентрических координат особенно полезен в компьютерной графике для интерполяции цветов, текстур и других свойств вершин.

Для простых задач проверки принадлежности точки тетраэдру достаточно **Метода объёмов**, но для работы с более сложными полиэдрами и в прикладных задачах компьютерной графики предпочтительнее метод уравнений плоскостей или барицентрических координат.

5 Формулировка прикладной задачи

5.1 Область: Компьютерная графика

В компьютерной графике при работе с виртуальной камерой часто используется понятие **пирамиды видимости** (view frustum). Это усечённая пирамида, определяющая область пространства, которая видна через камеру. В упрощённом случае для точечного источника света или для некоторых алгоритмов определения видимости используется полная пирамида (тетраэдр).

Задача: Определить, находится ли объект (или его часть) внутри пирамиды видимости камеры. Это необходимо для:

1. **Отсечения невидимых объектов:** Если объект полностью вне пирамиды видимости, его не нужно отрисовывать
2. **Определения уровня детализации:** Объекты ближе к камере требуют более детальной прорисовки
3. **Оптимизации физических вычислений:** Взаимодействие рассчитывается только для объектов в поле зрения

Решение: Для каждого объекта вычисляется его ограничивающий параллелепипед (bounding box) или сфера. Проверяем принадлежность вершин этого объёма к пирамиде видимости (тетраэдру). Если хотя бы одна вершина внутри — объект потенциально видим.

5.2 Область: Робототехника

В робототехнике при планировании траектории манипулятора важно обеспечить безопасность работы. Определяются **запретные зоны** — области пространства, куда не должен заходить манипулятор.

Задача: Проверить, не заходит ли рабочий орган робота в запретную зону, заданную в виде тетраэдра (например, зона вокруг опасного оборудования или хрупких объектов).

Решение: В реальном времени отслеживается положение конечного эффектора манипулятора. Проверяется принадлежность этой точки к запретным зонам. При обнаружении пересечения система безопасности останавливает робота или корректирует траекторию.

6 Визуализация результатов

Для визуализации использован **Python** с библиотеками **matplotlib** и **numpy**.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def plot_tetrahedron(ax, A, B, C, D):
    vertices = np.array([A, B, C, D])

    edges = [
        [0, 1], [0, 2], [0, 3],
        [1, 2], [1, 3], [2, 3]
    ]

    for edge in edges:
```

```
        ax.plot3D(*zip(vertices[edge[0]], vertices[edge[1]]), 'b-',
            linewidth=2)

    ax.scatter(*zip(A, B, C, D), c='red', s=100, marker='o')

    labels = ['A', 'B', 'C', 'D']
    for i, label in enumerate(labels):
        ax.text(vertices[i][0], vertices[i][1], vertices[i][2], label,
            fontsize=12)

def plot_test_points(ax, points, colors, markers):
    for point, color, marker in zip(points, colors, markers):
        ax.scatter(*point, c=color, s=150, marker=marker, alpha=0.8)

A = np.array([0, 0, 0])
B = np.array([1, 0, 0])
C = np.array([0, 1, 0])
D = np.array([0, 0, 1])

test_points = [
    np.array([0.25, 0.25, 0.25]),  #              -

    np.array([1/3, 1/3, 1/3]),     #                -

    np.array([0.5, 0, 0]),         #                -

    np.array([0, 0, 0]),           #               -

    np.array([1, 1, 1]),           #                -

]

colors = ['green', 'yellow', 'orange', 'red', 'black']
markers = ['o', 's', '^', '*', 'x']

fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')

plot_tetrahedron(ax, A, B, C, D)
plot_test_points(ax, test_points, colors, markers)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('
                              ')
ax.view_init(elev=20, azim=45)

plt.legend(['                ', '             ', '                ',
    '               ', '                    ', '              '],
           loc='upper right')
plt.tight_layout()
plt.show()
```

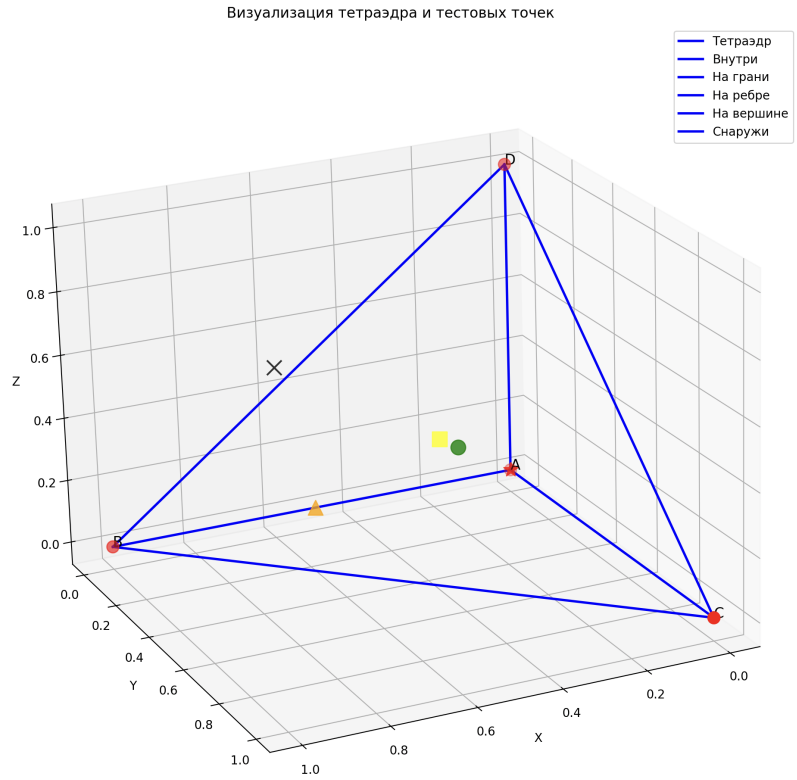

Рис. 1: Визуализация тетраэдра и тестовых точек различных типов

Обозначения на рисунке:

- Синие линии — рёбра тетраэдра
- Красные точки — вершины тетраэдра (A, B, C, D)
- Зелёный круг — точка внутри тетраэдра
- Жёлтый квадрат — точка на грани
- Оранжевый треугольник — точка на ребре
- Красная звезда — точка на вершине
- Чёрный крест — точка снаружи тетраэдра

Метод	Плюсы	Минусы
Метод объёмов	• Прямая геометрическая интерпретация • Работает для всех случаев • Легко считать вручную для простых точек	• Чувствителен к вычислительным погрешностям • Требует вычисления 5 определителей • Сложнее для невыпуклых многогранников
Метод знаков смешанных произведений	• Чёткая математическая основа • Позволяет определить положение на грани/ребре/вершине • Эффективная реализация	• Требует аккуратной работы с знаками • Сложнее для понимания начинающими • Чувствителен к ориентации тетраэдра
Метод уравнений плоскостей	• Интуитивно понятный • Легко расширяется на произвольные многогранники • Хорошо работает с отсечениями в графике	• Требует предварительного вычисления уравнений плоскостей • Нужно правильно определять направление нормалей • Больше вычислений для проверки
Метод барицентрических координат	• Естественная геометрическая интерпретация • Коэффициенты могут использоваться для интерполяции • Позволяет легко определить положение точки	• Требует вычисления 5 объёмов • Менее эффективен для множественных проверок • Чувствителен к погрешностям при малых объёмах

Таблица 1: Сравнение методов проверки принадлежности точки тетраэдру