Компьютерная геометрия и геометрическое моделирование

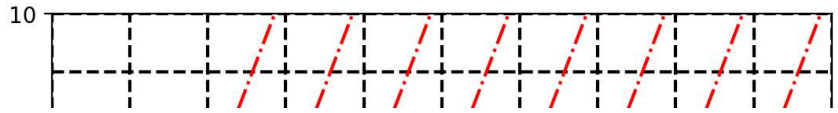
Лабораторная работа №1

- Ф.И.О: **Мухамедияр Адиль**
- Ноиер студ. билета: **1032205725**
- Группа: **НКНбд-01-20**

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon, RegularPolygon
```

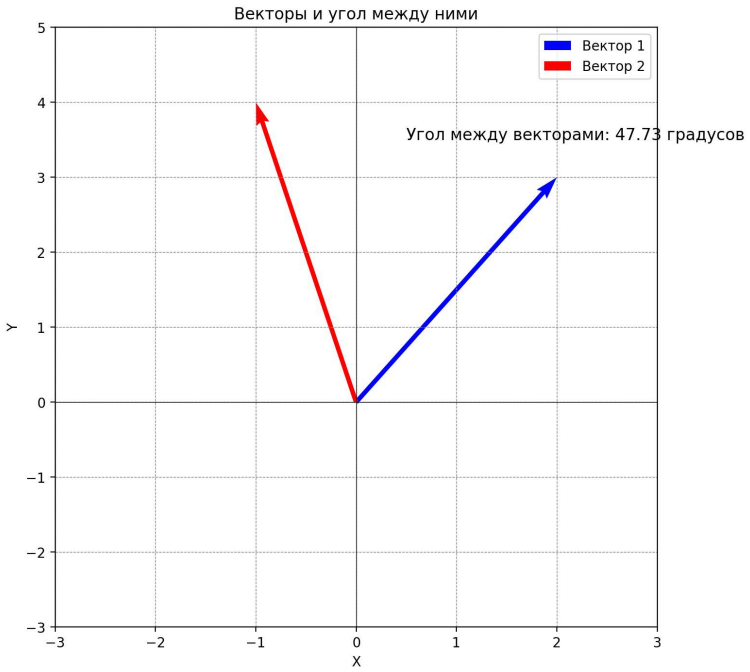
№1

```
# Определяем размер сетки и границы графика
n = 10
# Установили границы по оси X. Определяем ширину графического окна
plt.xlim(0, n)
# Установили границы по оси Y. Определяем высоту графического окна
plt.ylim(0, n)
# Отключаем отображение сетки на графике
plt.grid(False)
# Рисуем сетку вручную
for i in range(n + 1):
    # Вертикальные линии
    plt.vlines(i, 0, n, colors='black', linestyles='dashed')
    # Горизонтальные линии
    plt.hlines(i, 0, n, colors='black', linestyles='dashed')

# Рисуем линию под наклоном
for i in range(n):
    plt.axline((i, 0), slope=3.5, color='red', linestyle='-.')
# Отобразим сетку
plt.show()
```



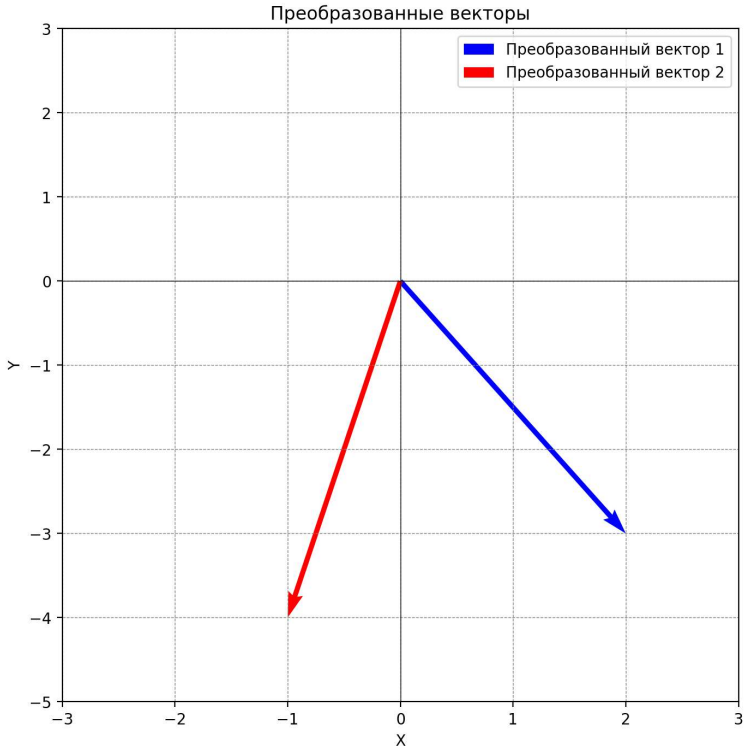
№ 2

```
# Создаем векторы
vector1 = np.array([2, 3])
vector2 = np.array([-1, 4])

# Рисуем векторы
plt.figure(figsize=(8, 8))  # Создание графического окна размером 8x8

# Рисование вектора 1 с началом в (0, 0)
plt.quiver(0, 0, vector1[0], vector1[1], angles='xy', scale_units='xy', scale=1, color='blue', label='Вектор 1')
# Рисование вектора 2 с началом в (0, 0)
plt.quiver(0, 0, vector2[0], vector2[1], angles='xy', scale_units='xy', scale=1, color='red', label='Вектор 2')

# Вычисляем ориентированный угол между векторами
angle_radians = np.arctan2(np.linalg.det([vector1, vector2]), np.dot(vector1, vector2))
# Вычисление ориентированного угла между векторами в радианах
angle_degrees = np.degrees(angle_radians)  # Преобразование угла в градусы

# Вывод значения угла на графике с указанием единиц измерения и округлением до 2 знаков после запятой
plt.text(0.5, 3.5, f'Угол между векторами: {angle_degrees:.2f} градусов', fontsize=12)

# Отображение графика
plt.xlim(-3, 3)  # Границы по оси X от -3 до 3
plt.ylim(-3, 5)  # Границы по оси Y от -3 до 5
plt.axhline(0, color='black',linewidth=0.5)  # Горизонтальная линия через 0
plt.axvline(0, color='black',linewidth=0.5)  # Вертикальная линия через 0
plt.grid(color='gray', linestyle='--', linewidth=0.5)  # Отображение сетки с серым цветом и пунктирным стилем
plt.legend()  # Отображение легенды
plt.title('Векторы и угол между ними')  # Заголовок графика
plt.xlabel('X')  # Подпись оси X
plt.ylabel('Y')  # Подпись оси Y
plt.show()  # Отображение графика
```



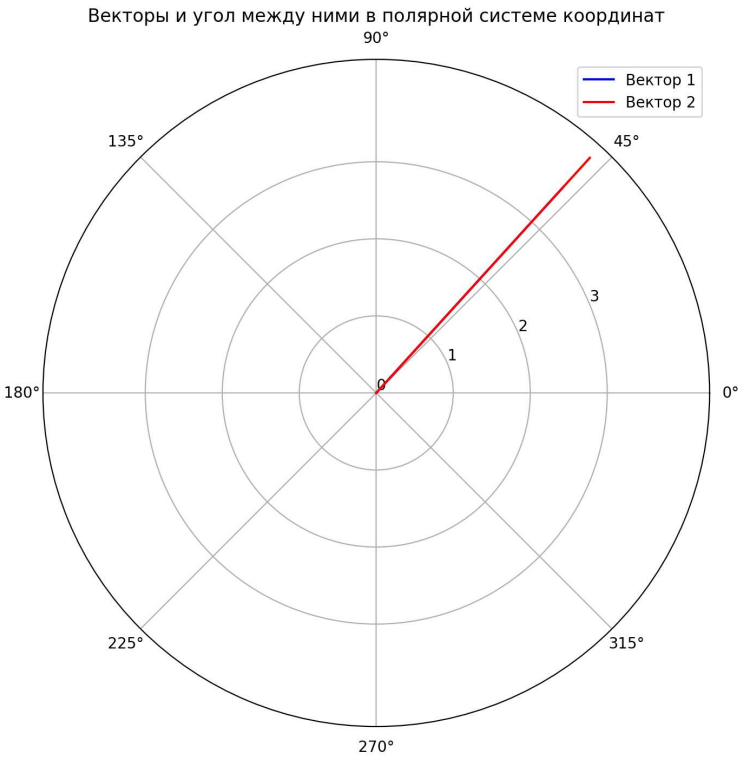
Мы можем видеть, как линейное преобразование изменило ориентированный угол между векторами.

```
# Матрица преобразования, которая отражает векторы относительно оси OY
transformation_matrix = np.array([[1, 0], [0, -1]])

# Применил линейное преобразование к векторам
transformed_vector1 = np.dot(transformation_matrix, vector1)
transformed_vector2 = np.dot(transformation_matrix, vector2)

plt.figure(figsize=(8, 8))  # Создание графического окна размером 8x8
# Рисуем преобразованные векторы с началом в (0, 0)
plt.quiver(0, 0, transformed_vector1[0], transformed_vector1[1], angles='xy', scale_units='xy', scale=1, color='blue', label='Преобразова
plt.quiver(0, 0, transformed_vector2[0], transformed_vector2[1], angles='xy', scale_units='xy', scale=1, color='red', label='Преобразован


# Отображение графика
plt.xlim(-3, 3)  # Границы по оси X
plt.ylim(-5, 3)  # Границы по оси Y
plt.axhline(0, color='black',linewidth=0.5)  # Горизонтальная линия
plt.axvline(0, color='black',linewidth=0.5)  # Вертикальная линия
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.legend()  # Отображение легенды
plt.title('Преобразованные векторы')
plt.xlabel('X')
plt.ylabel('Y')
# Отображение графика
plt.show()
```



Переход в полярную систему координат

```
# Переход в полярную систему координат
plt.figure(figsize=(8, 8))  # Создание графического окна размером 8x8
# Построение графика в полярной системе координат
plt.polar([0, angle_radians], [0, np.linalg.norm(vector1)], label='Вектор 1', color='blue')
plt.polar([0, angle_radians], [0, np.linalg.norm(vector2)], label='Вектор 2', color='red')

# Установка радиальных меток на оси Y
plt.yticks(np.arange(0, np.ceil(np.linalg.norm(vector1)), step=1))

# Отображение легенды
plt.legend()

# Отображение графика
plt.title('Векторы и угол между ними в полярной системе координат')
plt.show()
```



№3

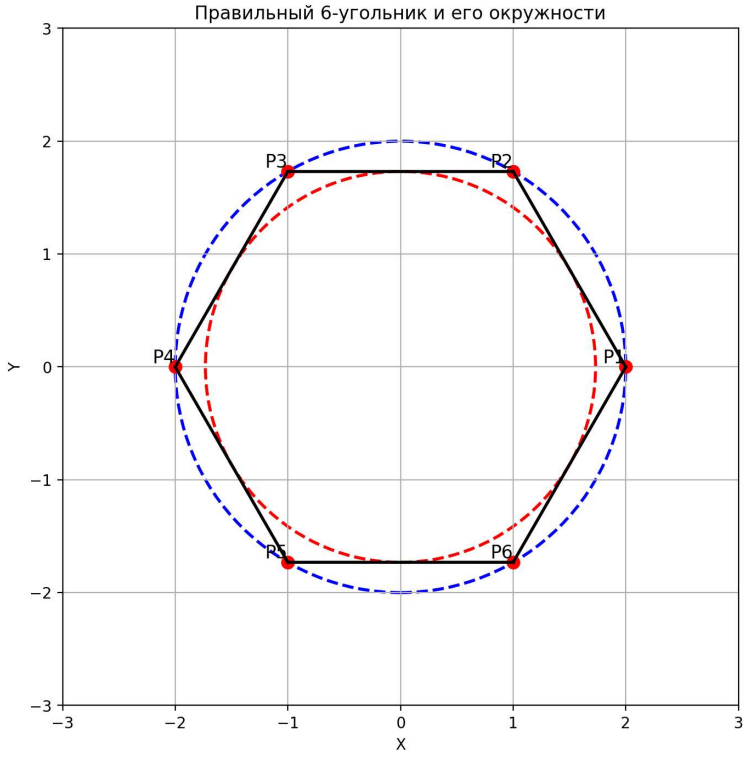
```
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon
import numpy as np

# Установка DPI (точек на дюйм) равным 200
plt.rcParams['figure.dpi'] = 200

# Определение координат вершин многоугольника (любое количество вершин)
vertices = np.array([(1, 1), (2, 3), (4, 2), (3, 1)])
```

```
# Рисование многоугольника
fig, ax = plt.subplots(figsize=(4, 4))  # Регулировка размера графика при необходимости
polygon = Polygon(vertices, closed=True, fill=False, edgecolor='b', linewidth=2)
ax.add_patch(polygon)

# Отображение вершин многоугольника в виде маркеров
x_coords = vertices[:, 0]
y_coords = vertices[:, 1]
ax.plot(x_coords, y_coords, 'ro', markersize=8)  # Красные круглые маркеры

# Вычисление ориентированной площади многоугольника
def oriented_area(vertices):
    n = len(vertices)
    area = 0
    for i in range(n):
        x1, y1 = vertices[i]
        x2, y2 = vertices[(i + 1) % n]
        area += (x1 * y2 - x2 * y1)
    return 0.5 * abs(area)

area = oriented_area(vertices)
print(f"Ориентированная площадь многоугольника: {area}")

# Получение координат вершин и координат i-ой вершины
x_of_vertices = vertices[:, 0]
y_of_vertices = vertices[:, 1]

i = 2  # Пример получения координат i-ой вершины
x_i = x_of_vertices[i]
y_i = y_of_vertices[i]

# Отображение графика
plt.xlim(0, 5)  # Границы по оси X
plt.ylim(0, 5)  # Границы по оси Y
plt.grid(True)  # Отображение сетки
plt.title('Произвольный многоугольник с вершинами')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

```
Ориентированная площадь многоугольника: 3.5
```

№4

```
# Задание количества сторон (n) и радиусов вписанной (r) и описанной (R) окружностей
n = 6  # Количество сторон
R = 2.0  # Радиус описанной окружности
r = R * np.cos(np.pi / n)  # Радиус вписанной окружности

# Вычисление координат вершин правильного многоугольника
theta = np.linspace(0, 2 * np.pi, n, endpoint=False)
x_coords = R * np.cos(theta)
y_coords = R * np.sin(theta)

# Создание фигуры с DPI равным 200
fig = plt.figure(figsize=(8, 8), dpi=200)

# Рисование описанной окружности
circle_outside = plt.Circle((0, 0), R, color='blue', fill=False, linestyle='dashed', linewidth=2)
plt.gca().add_patch(circle_outside)

# Рисование вписанной окружности
circle_inside = plt.Circle((0, 0), r, color='red', fill=False, linestyle='dashed', linewidth=2)
plt.gca().add_patch(circle_inside)

# Рисование вершин многоугольника
plt.plot(x_coords, y_coords, 'ro', markersize=8)

# Подпись вершин многоугольника
for i, (x, y) in enumerate(zip(x_coords, y_coords), start=1):
    plt.text(x, y, f'P{i}', fontsize=12, ha='right', va='bottom')

# Рисование сторон многоугольника
for i in range(n):
    plt.plot([x_coords[i], x_coords[(i + 1) % n]], [y_coords[i], y_coords[(i + 1) % n]], color='black', linewidth=2)

# Настройка представления
plt.xlim(-R - 1, R + 1)  # Ограничение для оси X
plt.ylim(-R - 1, R + 1)  # Ограничение для оси Y
plt.gca().set_aspect('equal', adjustable='box')  # Масштабирование без искажения
plt.grid(True)  # Отображение сетки

# Отображение графика
plt.title(f'Правильный {n}-угольник и его окружности')  # Заголовок графика
plt.xlabel('X')  # Подпись оси X
plt.ylabel('Y')  # Подпись оси Y
plt.show()  # Отображение графика
```



№5

```
# Ввод начальных данных:
r = 5
t = np.linspace(-2 * np.pi, 2 * np.pi, 100)

# Инициализация функции:
x = r * (t - np.sin(t))
y = r * (1 - np.cos(t))

# Вычисление производной:
dx = r - r * np.cos(t)
dy = r * np.sin(t)

fig03 = plt.figure(num=3, dpi=200)
ax03 = fig03.add_subplot(1, 1, 1)

# Отображение синего круга:
l = np.linspace(0, 2 * np.pi, 100)
X_circle = [(r * np.cos(i) + x[25]) for i in l]
Y_circle = [(r * np.sin(i) + r) for i in l]
ax03.plot(X_circle, Y_circle, marker="", color="blue", linewidth=0.6)
ax03.plot(x[25], r, marker=".", ms=3, color="purple")

# Отображение касательного поля и циклоиды (с помощью quiver):
ax03.quiver(x[::6], y[::6], dy[::6], dx[::6], units='xy', angles='xy', color='blue', width=0.1, scale=2.5, label='Горизонтальные касатель
ax03.quiver(x[::6], y[::6], dx[::6], dy[::6], units='xy', angles='xy', color='red', width=0.1, scale=2.5, label='Вертикальные касательные
ax03.plot(x, y, linewidth=1, color='black', label=r'Циклоида')

ax03.set_xlabel(r'x $Ox$')
ax03.set_ylabel(r'y $Oy$')

legend = ax03.legend(loc=1)
for text in legend.get_texts():
    text.set_fontsize('small')

ax03.set_aspect('equal')

plt.show()
```