Лабораторная работа 5

Анимация качения треугольника Рело по квадрату

Компьютерная математика II, ММФ, БГУ

Лаврова О.А., март 2025

Навыки

1. базовый **Python**: оператор `def` для создания функции; оператор `global` для изменения области видимости переменной; строки документации для функций; функция как аргумент другой функции
2. расширение `numpy`: тип данных массив `ndarray`, `array`, арифметические операции над массивами, `sin`, `cos`, `pi`, `transpose`, индексация матриц, `linspace`, `dot`
3. модуль `pyplot` из пакета `matplotlib`: `figure`, `plot`, `axis`, `subplot`
4. модуль `animation` из пакета `matplotlib`: `funcAnimation`
5. **математика**:
   - треугольник Рело
   - поступательное движение по заданной траектории
   - вращательное движение, матрица поворота

Основные определения

Треугольник Рело представляет собой область пересечения трех кругов радиуса r с центрами в вершинах равностороннего треугольника с длиной стороны r.

Так как треугольник Рело является фигурой постоянной ширины, то его можно вписать в квадрат с длиной стороны, равной ширине треугольника Рело r таким образом, что треугольник Рело будет касаться всех сторон квадрата. Это связано с тем, что противоположные стороны квадрата, расстояние между которыми равно r, располагаются на опорных прямых к треугольнику Рело.

Поступательное движение фигуры – это движение, при котором траектории движения всех точек фигуры одинаковы.

Вращательное движение фигуры – это движение, при котором траектории движения точек фигуры представляют собой окружности (или дуги окружностей) с центрами, лежащими в одной точке.

Качение треугольника Рело по квадрату – это вращательное движение треугольника Рело относительно своего центра с дополнительным поступательным движением центра треугольника по траектории, близкой к окружности, чтобы обеспечить касание всех сторон квадрата при движении треугольника Рело. Качение треугольника Рело происходит за счет его одновременного поступательного и вращательного движения.

Полагаем, что при качении по квадрату против часовой стрелки центр треугольника Рело совершает *поступательное движение по окружности* против часовой стрелки с центром окружности, расположенным в начале координат. Радиус окружности для поступательного движения является неизвестным и будет вычислен в Задании 5.1.

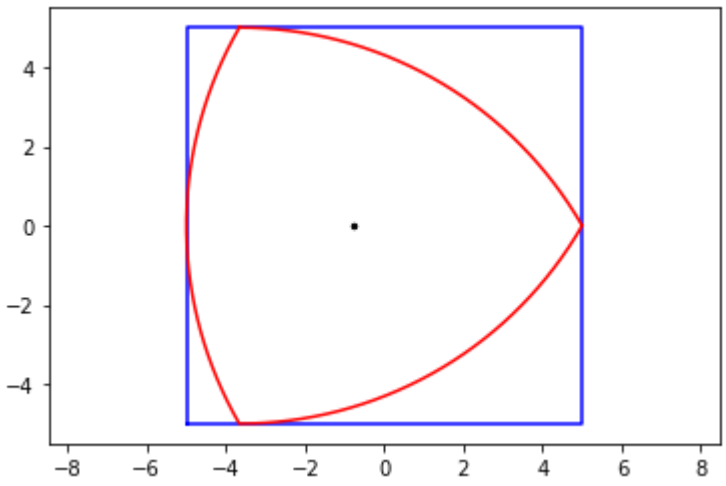
Одновременно с поступательным движением треугольник Рело совершает *вращательное движение* по часовой стрелке. Поступательное и вращательное движения согласованы таким образом, что поступательное движение по всей окружности соответствует вращательному движению треугольника Рело на угол $2\pi/3$.

Задание 5.1. Начальное положение треугольника Рело и квадрата

Изобразите в одной графической области треугольник Рело (сплошная линия красного цвета), точку центра треугольника Рело (маркер черного цвета) и квадрат (сплошная линия синего цвета), в который вписан треугольник Рело.

In [ ]:

Out[ ]: (-5.5, 5.500000000000002, -5.5, 5.500000000000001)



Реализация Задания 5.1

In [1]:
```
import math
import numpy as np
```

```
import matplotlib.pyplot as plt
import relo

# для JupyterLab предварительно нужно установить пакет ipympl
# import matplotlib.animation as anim
# %matplotlib widget
```

Определим исходные данные для построения треугольника Рело

In [2]:
```
n = 3 # количество вершин треугольника Рело
center = np.array([0.,0.]) # центр треугольника Рело
r = 10 # ширина треугольника Рело
N = 100 # количество точек для описания стороны треугольника Рело
```

Для построения матрицы `relo_matrix` координат точек-границ треугольника Рело используем функцию `regular_polygon_Relo(n, center, r, N)` из модуля `relo`, созданного в Лабораторной работе 4.

In [3]:
```
relo_matrix = relo.regular_polygon_Relo(r=r, center=center, N=N)
```

Длина стороны квадрата равна ширине треугольника Рело r, центр квадрата располагаем в начале координат.

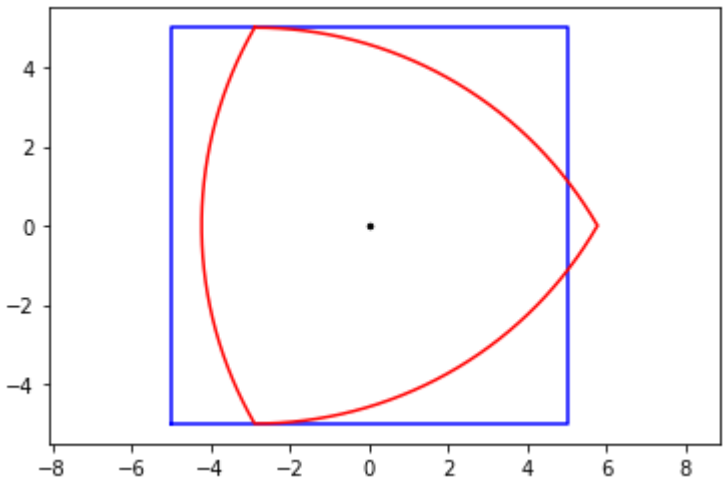
\color{red}\text{Постройте} матрицу `square` для покоординатного описания границы квадрата.

\color{red}\text{Изобразите} в одной системе координат квадрат, треугольник Рело и точку центра треугольника Рело.

In [4]:
```
plt.plot(square[:,0],square[:,1],'b-')
plt.plot(relo_matrix[:,0],relo_matrix[:,1],'r-')
plt.plot(center[0],center[1],'k.',markersize=5)

plt.axis('equal')
```

Out[4]:
```
(-5.538675134594813, 6.312177826491073, -5.5, 5.500000000000001)
```



Треугольник Рело необходимо сдвинуть влево таким образом, чтобы он оказался вписанным в квадрат. Определите величину вектора сдвига `shift`. Поясните в тексте документа формулу для расчета вектора сдвига.

In [5]:
```
relo_matrix_shifted = relo_matrix - shift
center -= shift
```

\color{red}\text{Изобразите} в одной графической области квадрат, треугольник Рело после сдвига и точку центра треугольника Рело после сдвига.

Задание 5.2. Анимация поступательного движения треугольника Рело по окружности

Полагаем, что при качении треугольника Рело по квадрату против часовой стрелки центр треугольника Рело совершает поступательное движение по окружности с центром окружности в начале координат и радиусом, равным `shift[0]`, против часовой стрелки.

Зададим количество кадров анимации `N_frames`, необходимое для поступательного движения треугольника с полным обходом окружности

In [7]:
```
N_frames = 100
```

Создадим матрицу `center_frame` из `N_frames` строк и двух столбцов для описания координат точек окружности, необходимой для реализации поступательного движения, с центром в начале координат и радиусом `shift[0]` при обходе окружности против часовой стрелки от $-\pi$ до π. В первом столбце матрицы расположены x-координаты точек, во втором столбце -- y координаты.

Подумайте, почему обход окружности осуществляется от $-\pi$ до π.

In [8]:
```
t_center = np.linspace(-np.pi, np.pi, N_frames)
center_frame = shift[0]*np.transpose([np.cos(t_center), np.sin(t_center)])
```

Изобразим первый кадр анимации. Для этого в одной графической области изображаем квадрат (сплошная линия синего цвета), вписанный треугольник Рело (сплошная линия красного цвета), центр треугольника Рело (линия по точкам красного цвета) и окружность, по которой будет двигаться центр треугольника Рело (пунктирная линия черного цвета).

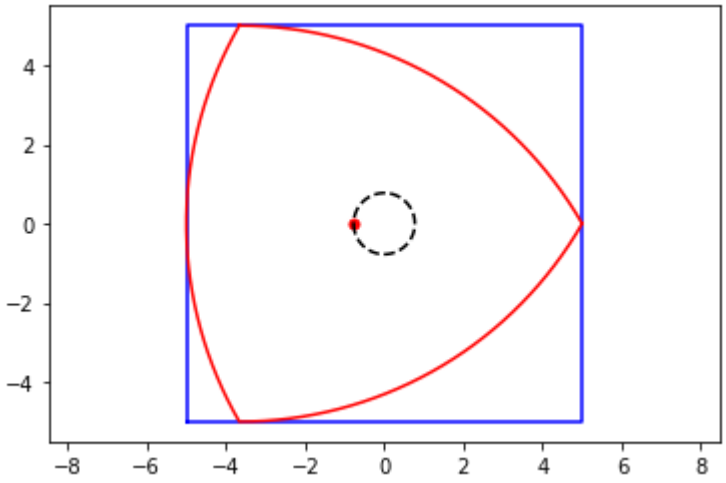
In [9]:
```
frame = 0
relo_matrix_shifted = relo_matrix + center_frame[frame]
```

In [10]:
```
plt.figure()
plt.plot(square[:,0],square[:,1],'b-')
plt.plot(relo_matrix_shifted[:,0],relo_matrix_shifted[:,1],'r-')
plt.plot(center_frame[frame,0],center_frame[frame,1],'r.',markersize=10)
```

```
plt.plot(center_frame[:,0],center_frame[:,1],'k--')

plt.axis('equal')
```

Out[10]: (-5.5, 5.500000000000002, -5.5, 5.500000000000001)



Напишите пользовательскую функцию трех аргументов `draw_frame(r, N_frames, frame)`, которая изображает кадр с номером `frame` при поступательном движении центра треугольника Рело по окружности против часовой стрелки.

- Аргумент `r` задает ширину треугольника Рело.
- Аргумент `N_frames` задает количество кадров для прохождения центром треугольника Рело полной окружности.
- Аргумент `frame` задает номер кадра; по умолчанию `frame=0`.
- Функция осуществляет визуализацию и не возвращает объекты.

Для функции `draw_frame` \color{red}\text{укажите} аннотации типов и \color{red}\text{оформите} строки документации. **Переменные из глобальной области видимости в теле функции использовать нельзя**!

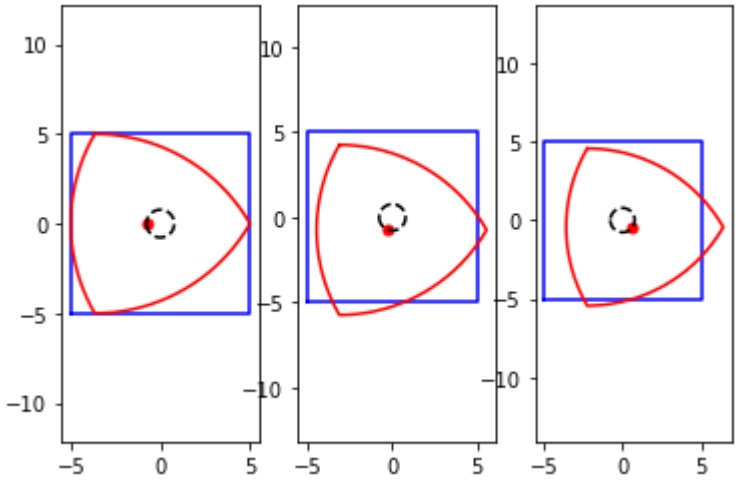
Изобразим несколько кадров анимации, расположив их в одном графическом окне по горизонтали. Для этого используем функцию `subplot(m,n,k)` из модуля `pyplot` пакета `matplotlib`, которая позволяет расположить графики в виде таблицы из `m` строк, `n` столбцов на `k`-ой позиции при последовательной нумерации графиков слева-направо и сверху-вниз, начиная с 1.

In [12]:
```
plt.figure()
N_frames = 100

plt.subplot(1,3,1)
draw_frame(r=r, N_frames=N_frames)

plt.subplot(1,3,2)
draw_frame(r=r, N_frames=N_frames, frame=20)

plt.subplot(1,3,3)
draw_frame(r=r, N_frames=N_frames, frame=40)
```



Реализуйте анимацию поступательного движения треугольника Рело по окружности против часовой стрелки. При построении анимации возьмите за основу функцию `draw_frame`, но не используйте ее напрямую в качестве аргумента функции `FuncAnimation` (модуль `animation` библиотеки `matplotlib`).

Смотрите Лабораторную работу 3 для примеров построения анимации.

Задание 5.3. Анимация качения треугольника Рело

Дополнительно к поступательному движению треугольника Рело против часовой стрелки добавим вращательное движение по часовой стрелке на угол θ. При этом будем учитывать, что полный оборот центра треугольника на угол 2π соответствует вращению треугольника Рело на угол $2\pi/3$.

Определим пользовательскую функцию `rotate_matrix(theta)` для создания **матрицы поворота по часовой стрелке** на угол `theta` с применением синтаксиса `lambda`-функций

In [13]:
```
rotate_matrix = lambda theta: np.array([[np.cos(theta), np.sin(theta)],
                                        [-np.sin(theta), np.cos(theta)]])
rotate_matrix(np.pi)
```

Out[13]:
```
array([[-1.0000000e+00,  1.2246468e-16],
       [-1.2246468e-16, -1.0000000e+00]])
```

Для поворота треугольника Рело на заданный угол необходимо матрицу поворота умножить на каждую точку в представлении треугольника Рело `relo_matrix` *с центром в начале координат*. Для умножения матрицы на вектор используем функцию `dot` из расширения `numpy`.

Например, повернем треугольник Рело `relo_matrix` с центром в начале координат на угол $\pi/3$ по часовой стрелке

In [14]:
```
theta = np.pi/3
matrix = rotate_matrix(theta)
```

```
relo_matrix_rotated = np.array([np.dot(matrix,row) for row in relo_matrix])
```

Задание* (необязательное): предложите альтернативную реализацию кода `np.array([np.dot(matrix,row) for row in relo_matrix])` с использованием матричного умножения в виде `matrix1 @ matrix2`.

Изобразим в одной графической области исходный треугольник Рело (сплошная линия красного цвета) и повернутый треугольник Рело (пунктирная линия зеленого цвета). При этом выделим маркером первую точку в матричном представлении треугольника Рело, чтобы проиллюстрировать эффект вращения по часовой стрелке на угол $\pi/3$.

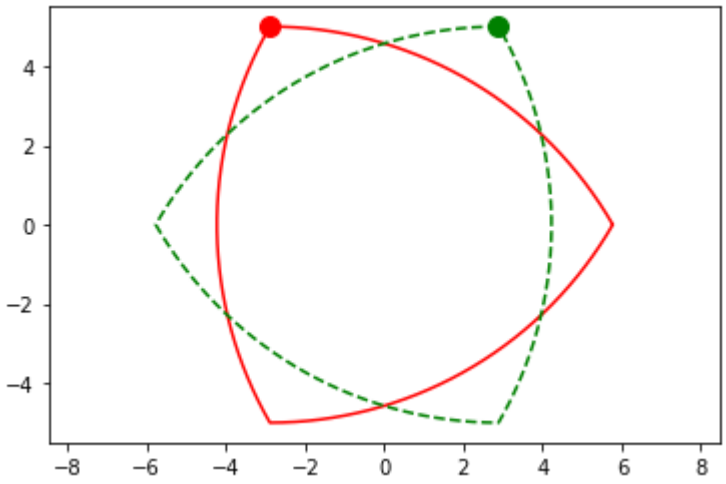
In [15]:
```
plt.figure()

plt.plot(relo_matrix[:,0],relo_matrix[:,1],'r-')
plt.plot(relo_matrix[0,0],relo_matrix[0,1],'r.',markersize=20)

plt.plot(relo_matrix_rotated[:,0],relo_matrix_rotated[:,1],'g--')
plt.plot(relo_matrix_rotated[0,0],relo_matrix_rotated[0,1],'g.',markersize=20)

plt.axis('equal')
```

Out[15]:
```
(-6.3508529610858835, 6.350852961085886, -5.500000000000005, 5.500000000000001)
```



Определим массив `theta` со значениями угла поворота при вращательном движении, согласованный с полным оборотом центра треугольника Рело за `N_frames` кадров. **Напомним, что полный оборот центра треугольника на угол 2π соответствует вращению треугольника Рело на угол $2\pi/3$.**

In [16]:
```
theta = np.linspace(0, 2*np.pi/3, N_frames)
```

Для совмещения вращательного и поступательного движения треугольника Рело в кадре `frame` сначала будем поворачивать треугольник Рело с центром в начале координат на угол `theta[frame]`, затем будем перемещать треугольник Рело на вектор `center_frame[frame]`.

Подумайте, почему нужно делать сначала поворот, а потом перенос.

Изобразим 20-ый кадр анимации качения треугольника Рело

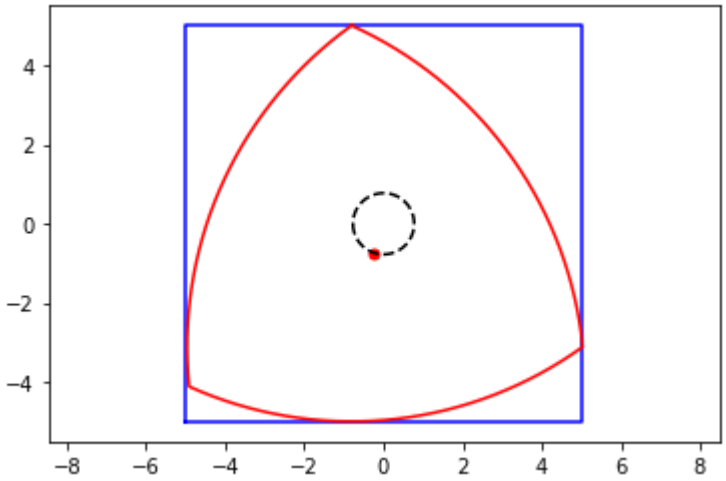
In [17]:
```
frame = 20

matrix = rotate_matrix(theta[frame])
relo_matrix_changed = (np.array([np.dot(matrix,row) for row in relo_matrix])
                       + center_frame[frame])

plt.figure()
plt.plot(square[:,0],square[:,1],'b-')
plt.plot(relo_matrix_changed[:,0],relo_matrix_changed[:,1],'r-')
plt.plot(center_frame[frame,0],center_frame[frame,1],'r.',markersize=10)
plt.plot(center_frame[:,0],center_frame[:,1],'k--')

plt.axis('equal')
```

Out[17]:
```
(-5.501735269668723, 5.536440663043177, -5.5002878716042405, 5.506045303689057)
```



Измените пользовательскую функцию `draw_frame(r, N_frames, frame)`, чтобы создавалось изображение кадра с номером `frame` для анимации качения треугольника Рело по квадрату. **Переменные из глобальной области видимости в теле функции использовать нельзя**!

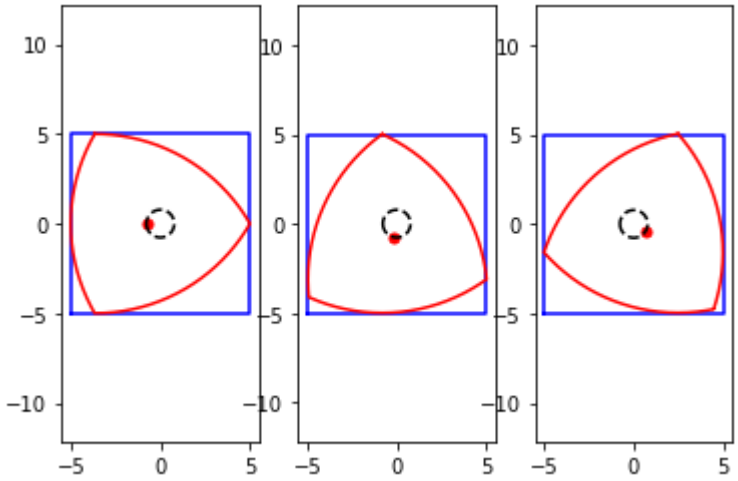
Изобразим несколько кадров анимации, расположив их в одном графическом окне по горизонтали с использованием функции `subplot`.

In [19]:
```
plt.figure()

plt.subplot(1,3,1)
draw_frame(r=r, N_frames=N_frames)

plt.subplot(1,3,2)
draw_frame(r=r, N_frames=N_frames, frame=20)

plt.subplot(1,3,3)
draw_frame(r=r, N_frames=N_frames, frame=40)
```



Реализуйте анимацию качения треугольника Рело против часовой стрелки. При построении анимации возьмите за основу функцию `draw_frame`, но не используйте ее напрямую в качестве аргумента функции `FuncAnimation` (модуль `animation` библиотеки `matplotlib`).