Лабораторная работа 4

Правильный многоугольник Рело. Векторизация вычислений в numpy

Компьютерная математика II, ММФ, БГУ

Лаврова О.А., март 2024

Навыки

1. базовый **Python**: оператор `def` для создания функции; списковые включения; `range` ; оператор `assert` для обработки исключений; встроенное исключение `AssertionError`
2. расширение `numpy` : тип данных массив `ndarray` , `array` , `arange` , `linspace` , арифметические операции над массивами, `sin` , `cos` , `transpose` , индексация матриц, `concatenate`
3. модуль `pyplot` из пакета `matplotlib` : `plot` , `axis`
4. **математика**:
   - треугольник Рело, правильный многоугольник Рело
   - окружность, описанная вокруг правильного многоугольника

Основные определения

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

Правильный n-угольник Рело представляет собой область пересечения n кругов радуса r с центрами в вершинах правильного n-угольника с *нечетным числом* сторон длины l. Радиус круга r согласован с длиной стороны l правильного многоугольника таким образом, чтобы окружность радиуса r с центром в вершине правильного n-угольника проходила через две соседние вершины правильного n-угольника, противолежащие центру.

Опорные прямые фигуры -- это две параллельные прямые, которые касаются фигуры, но не пересекают ее.

Фигура постоянной ширины -- это геометрическая фигура, для которой расстояние между любыми опорными прямыми будет всегда одинаковым. Расстояние между опорными прямыми называется **шириной фигуры**.

Примерами фигур постоянной ширины являются *правильный многоугольник Рело* и *круг*. Ширина правильного многоугольника Рело равна радиусу r пересекающихся кругов. Ширина круга равна диаметру круга. Среди всех фигур постоянной ширины треугольник Рело имеет наименьшую площадь, круг имеет наибольшую площадь.

Задание 4.1. Построение треугольника Рело

Постройте представоение границы правильного треугольника Рело в виде матрицы, каждая строка которой содержит координаты точек, описывающих границу.

Полагаем, что заданы координаты центра правильного треугольника Рело `center`, ширина треугольника Рело `r` и количество точек для описания одной стороны треугольника Рело `N`.

Выполнение задания 4.1

Подключим модули

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Определим переменные

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

Этап 1. Вычисление координат вершин правильного треугольника

Вершины правильного многоугольника расположены на окружности, описанной вокруг многоугольника. Радиус R окружности, описанной вокруг правильного n-угольника с длиной стороны l, вычисляется по формуле

$$R = \frac{l}{2\sin\pi/n}.$$

Длина стороны l правильного треугольника, на котором будем строить треугольник Рело, совпадает с шириной треугольника Рело r.

In [3]:
```
l = r # !!! равенство справедливо только для треугольника Рело
```

Вычислим радиус описанной окружности R

In [4]:
```
R = l/(2*np.sin(np.pi/n))
```

Зафиксируем прямоугольную декартову систему координат. Центр треугольника размещаем в точке `center`. Построим матрицу `vertices` с координатами вершин правильного треугольника, используя параметрическое описание точек описанной окружности по формуле $x(t) = x_0 + R\cos(t)$, $y(t) = y_0 + R\sin(t)$ для значений $t \in \{0, 2\pi/3, 4\pi/3\}$.

In [5]:
```
t = np.arange(0,2*np.pi,2*np.pi/n)

vertices = center + R*np.transpose([np.cos(t), np.sin(t)])
vertices
```

Out[5]:
```
array([[ 5.77350269,  0.        ],
       [-2.88675135,  5.        ],
       [-2.88675135, -5.        ]])
```

Обратите внимание: массив `vertices` вычислен без использования циклов. Это пример векторизации вычислений в `numpy`.

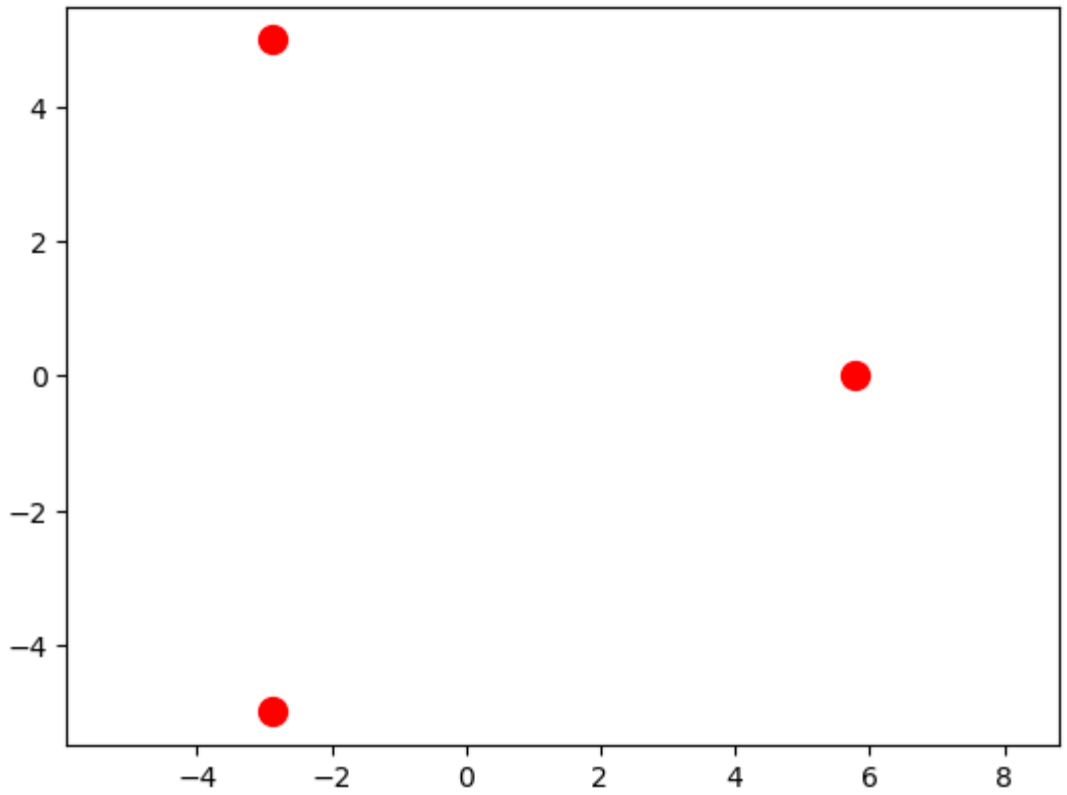
Для доступа к элементам матрицы в `numpy` можно использовать индексацию с двумя индексами

In [6]:
```
vertices[0,0], vertices[0,:], vertices[:,1]
```

Out[6]:
```
(5.773502691896258, array([5.77350269, 0.        ]), array([ 0.,  5., -5.]))
```

Построим изображение вершин треугольника

In [7]:
```
plt.plot(vertices[:,0],vertices[:,1],'r.',markersize=20)
plt.axis('equal');
```



Этап 2. Представление одной стороны треугольника Рело

Обозначим через α центральный угол, соответствующий стороне правильног n-угольника. Тогда

$$\alpha = 2\pi/n.$$

Обозначим через β центральный угол, соответствующий стороне правильного многоугольника Рело, когда центр окружности расположен в вершине правильного многоугольника. Тогда

$$\beta = \alpha/2.$$

Необходимо построить матрицу с координатами точек, описывающих одну сторону треугольника Рело относительно вершины `vertices[0]` . Сторона представляет собой дугу окружности радиуса r с центром в точке `vertices[0]` и значением угла $[\pi - \beta/2, \pi + \beta/2]$.

Вычислим значения введенных величин α и β

In [8]:
```
alpha = 2*np.pi/n
beta = alpha/2
```

Вычислим массив `angle` значений угла для построения координат точек стороны треугольника Рело, состоящий из `N` элементов

In [9]:
```
angle = np.linspace(-beta/2, beta/2, N)
```

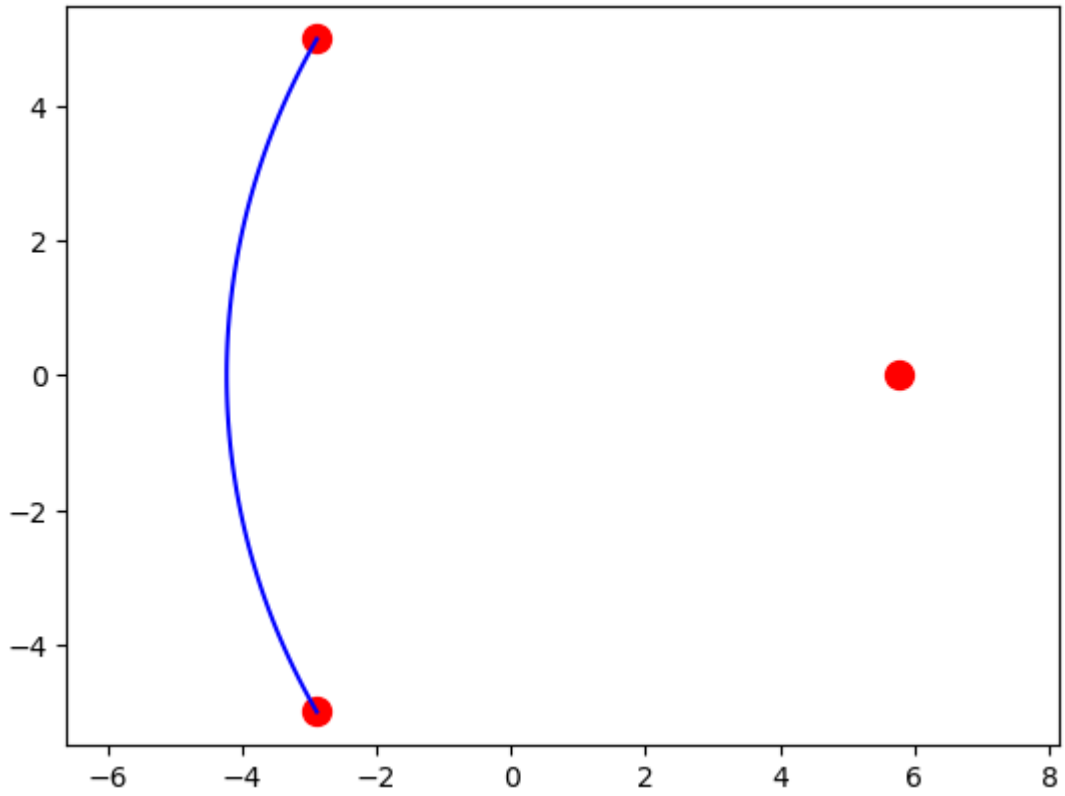
Вычислим матрицу `side0` с координатами точек первой стороны треугольника Рело, используя параметрическое описание окружности с центром в точке `vertices[0]` радиусом `r` для значений угла на отрезке $[\pi - \beta/2, \pi + \beta/2]$.

In [10]:
```
side0 = vertices[0] + r*np.transpose([np.cos(angle + np.pi), np.sin(angle + np.pi)])
```

Обратите внимание: массив `side0` вычислен без использования циклов. Это пример векторизации вычислений в `numpy` .

Построим изображение вершин и стороны треугольника Рело

In [11]:
```
plt.plot(vertices[:,0],vertices[:,1],'r.',markersize=20)
plt.plot(side0[:,0],side0[:,1],'b-',markersize=20)
plt.axis('equal');
```

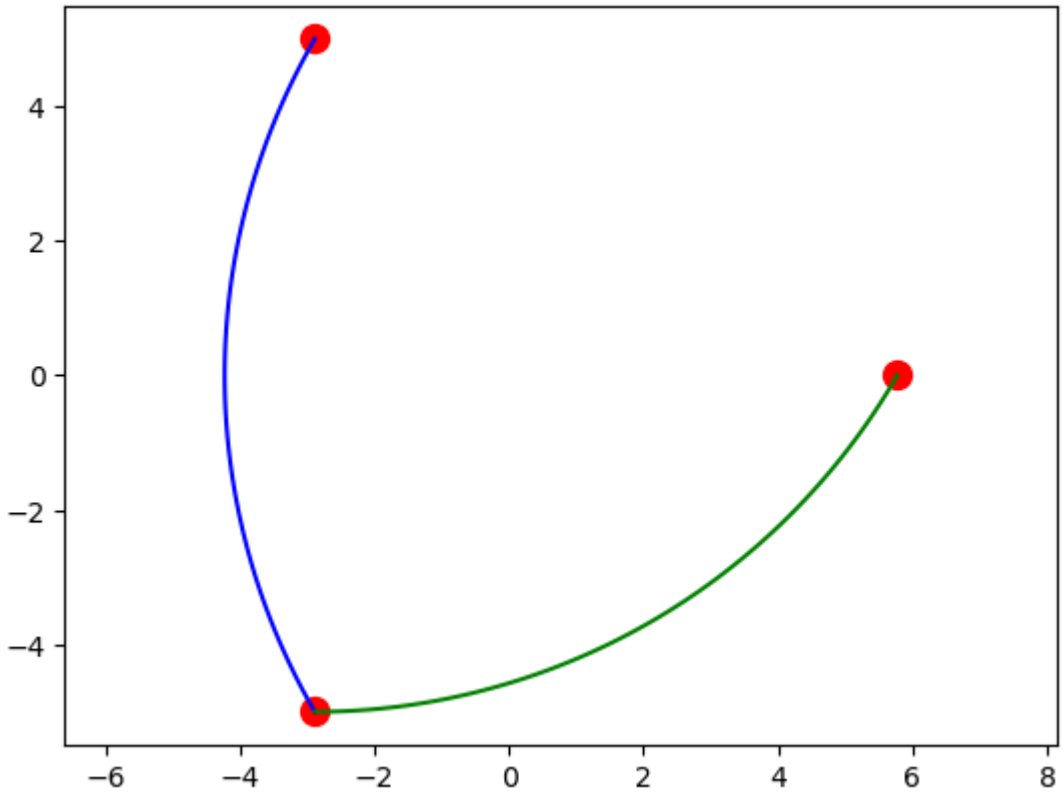


Вычислим матрицу `side1` с координатами точек, описывающих сторону треугольника Рело относительно второй вершины `vertices[1]` . Сторона представляет собой дугу окружности радиуса r с центром в точке `vertices[1]` и значением угла $[\pi - \beta/2 + \alpha, \pi + \beta/2 + \alpha]$

In [12]:
```
side1 = vertices[1] + r*np.transpose([np.cos(angle + np.pi + alpha),
                                      np.sin(angle + np.pi + alpha)])
```

Построим изображение вершин и двух сторон треугольника Рело

In [13]:
```
plt.plot(vertices[:,0],vertices[:,1],'r.',markersize=20)
plt.plot(side0[:,0],side0[:,1],'b-',markersize=20)
plt.plot(side1[:,0],side1[:,1],'g-',markersize=20)
plt.axis('equal');
```



Этап 3. Моделирование границы треугольника Рело

Создадим СПИСОК из матриц, каждая их которых содержит координаты точек одной из сторон треугольника Рело

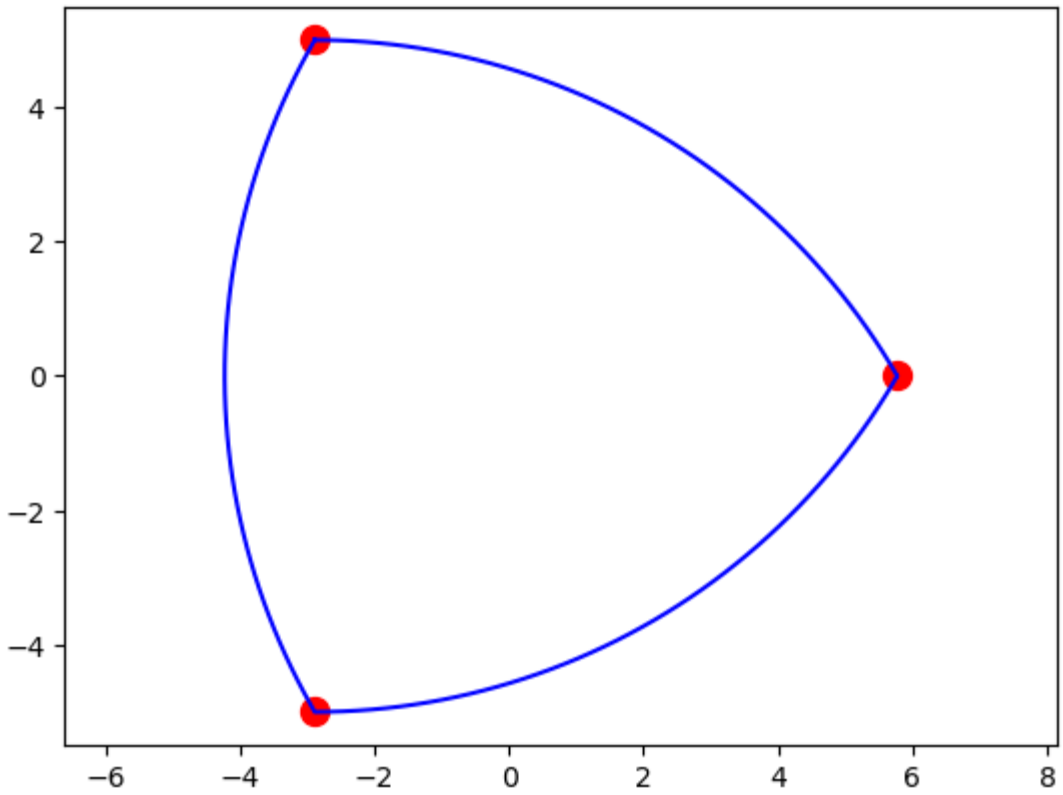
In [14]:
```
list_sides = [vertices[i] +
              r*np.transpose([np.cos(angle + np.pi + i*alpha),
                              np.sin(angle + np.pi + i*alpha)])
              for i in range(n)]
```

С помощью фунции `concatenate` из расширения `numpy` объединим массивы, созданные для каждой из сторон треугольника Рело, в единый массив `sides`

In [15]:
```
sides = np.concatenate(list_sides)
```

Построим изображение вершин и всех сторон треугольника Рело

In [16]:
```
plt.plot(vertices[:,0],vertices[:,1],'r.',markersize=20)
plt.plot(sides[:,0],sides[:,1],'b-',markersize=20)
plt.axis('equal');
```



Задание 4.2. Построение правильного многоугольника Рело

Создайте матричное описание границы правильного многоугольника Рело для произвольных значений переменных `n`, `center`, `r`, `N`, следуя аналогичным рассуждениям из Задания 4.1 (вычисление координат вершин, представление одной стороны, моделирование всей границы). Помните, что количество вершин многоугольника n должно быть нечетным.

Обратите внимание, что длина стороны l правильного многоугольника, на основании которого строится многоугольник Рело, является неизвестной величиной и должна быть выражена через количество вершин n и ширину r многоугольника Рело. Напишите подробно, как получена аналитическая зависимость l от n и r. Объяснения оформите в тексте документа с лабораторной работой.

Задание 4.3. Результирующая пользовательская функция

а) Напишите пользовательскую функцию `regular_polygon_Relo(n, center, r, N)` на основании кода из Задания 4.2. Функция должна возвращать матрицу, каждая строка которой содержит координаты точек, описывающих границу правильного многоугольника Рело.

Аргументы пользовательской функции:

- `n` : количество вершин правильного многоугольника Рело; является *нечетным* целым числом большим 2; стандартное значение $n = 3$;
- `center` : массив координат центра правильного многоугольника Рело; стандартное значение $center = np.array([0, 0])$;
- `r` : ширина правильного многоугольника Рело; является положительным числом; стандартное значение $r = 1$;
- `N` : количество точек для описания одной стороны правильного многоугольника Рело; является натуральным числом; стандартное значение $N = 100$.

Код функции не должен использовать переменные из глобальной области видимости модуля. Внутри функции не должно осуществляться отображение многоугольника Рело.

б) Осуществите контроль за значениями аргументов при вызове функции с помощью оператора `assert` .

в) Укажите аннотации типов и оформите строки документации. Осуществите доступ к аннотации типов и к строкам документации через атрибуты объекта функции.

г) Создайте модуль `relo.py` , в котором будет содержаться пользовательская функция `regular_polygon_Relo` .

Комментарии по использованию оператора `assert`

Для контроля за передаваемыми значениями аргументов при вызове функции можно использовать оператор `assert` , вызываемый в начале тела функции. Вызов

```
assert condition
```

генерирует исключение `AssertionError` , если условие `condition` ложно, и выполнение функции прерывается. Например,

In [17]:
```
def f(r):
    assert r>0, 'radius r should be positive'
    ...
```

Если условие $r > 0$, указанное после оператора `assert` , истинно, то код, следующий за оператором `assert` , выполняется:

In [18]:
```
f(1)
```

Если условие $r > 0$, указанное после оператора `assert`, ложно, то оператор `assert` генерирует исключение `AssertionError`, прерывает процесс выполнения функции и выдает сообщение об ошибке с использованием строкового объекта `message`, который может быть указан при вызове оператора `assert`:

```
assert condition, message
```

In [19]:
```
f(0)
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Cell In[19], line 1
----> 1 f(0)

Cell In[17], line 2, in f(r)
      1 def f(r):
----> 2     assert r>0, 'radius r should be positive'
      3     ...

AssertionError: radius r should be positive
```

Задание 4.4. Тестирование функции

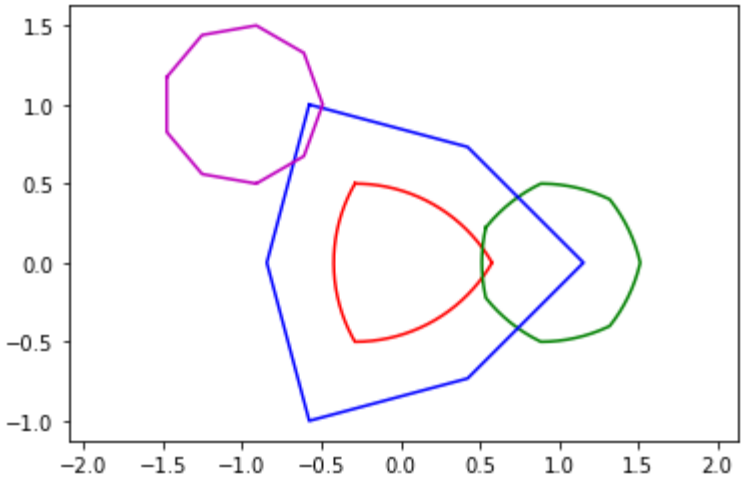
а) Протестируйте функцию `regular_polygon_Relo` из модуля `relo` для различных значений аргументов в предположении, что корректность вводимых данных НЕ гарантируется. Предварительно очистите ядро Python.

б) Постройте в одной системе координат правильные многоугольники Рело для различных значений вершин `n`, положений центра `center`, ширины `r` и количества точек для представления одной стороны `N`. При этом вызовите функцию `regular_polygon_Relo` с различными способами указания аргументов:

- все аргументы позиционные,
- все аргументы ключевые,
- аргументы позиционные и ключевые,
- все аргументы определяются стандартными значениями.

In [ ]:
```

```


1.5
1.0
0.5
0.0
−0.5
−1.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0