Лабораторная работа 1

Задача о падении тела. Построение графиков функций

Компьютерная математика II, ММФ, БГУ

Лаврова О.А., Щеглова Н.Л., февраль 2025

Навыки

1. базовые принципы работы с **Jupyter Notebook** (файлы с расширением .ipynb) в **JupyterLab**:
   - типы ячеек: ячейка кода и текстовая ячейка
   - режимы ячеек: режим редактирования и командный режим
   - языки разметки **Markdown** и **Latex** для задания форматирования в тексте
2. базовый **Python**: встроенные типы данных: `int`, `float`, `list`, `tuple`; генераторная функция `range`
3. модуль `math` из стандартной библиотеки модулей: `atan`, `degrees`
4. модуль `pyplot` из пакета `matplotlib`: `figure`, `axes`, `plot`, `title`, `xlabel`, `ylabel`, `axis`, `xlim`, `ylim`
   - двумерные графики явно заданной функции и параметрически заданной функции
   - двумерные графики функций в декартовой и полярной системах координат
5. расширение `numpy`: тип данных массив `ndarray`, `array`, `arange`
6. модуль `constants` пакета расширений `scipy`: `g`, `unit`
7. модуль `sympy`: `symbols`, `solve`
8. **математика**:
   - способы задания линии на плоскости
     - параметрические уравнения $x = x(t), y = y(t)$ в декартовой системе координат (x, y)
     - полярное уравнение $\rho = \rho(\varphi)$ в полярной системе координат (φ, ρ)
   - способы задания прямой на плоскости
     - уравнение по двум точкам
     - векторно-параметрическое уравнение
     - уравнение по точке и угловому коэффициенту

Задание 1.1. Задача о падении тела

Описание процесса. Тело запущено под углом α к горизонту на высоте h_{start} и через время $t = T$ достигает другое тело, расположенное на расстоянии s_{end} и

высоте h_{end}.

Предположения.

- Тело будем считать материальной точкой массы m.
- Движение запущенного тела осуществляется только под действием силы тяжести

$$\mathbf{F} = m(0, -g),$$

  где m — масса тела, g — ускорение свободного падения.
- Сопротивлением среды пренебрегаем.

Данные.

- Задача рассматривается при следующих значениях параметров: $h_{start} = 1$ m, $h_{end} = 3$ m, $T = 3$ s, $s_{end} = 5$ m.
- Угол запуска тела α не известен.
- Начальный момент времени движения тела полагаем равным нулю: $t = 0$.
- Значение массы тела m не повлияет на функции перемещения тела.

Задания для выполнения.

- $\boxed{\color{red}\text{Определите}}$ функцию перемещения тела $(s_x(t), s_y(t))$ под действием силы тяжести, где $s_x(t)$ и $s_y(t)$ — горизонтальная и вертикальная составляющая положения тела в момент времени t, соответственно.
- $\boxed{\color{red}\text{Найдите}}$ угол запуска тела α.
- $\boxed{\color{red}\text{Постройте}}$ график траектории движения тела $(s_x(t), s_y(t))$ для $t \in [0, T]$.

Выполнение Задания 1.1

Этап 1. Импортирование модулей

Импортируем модуль `math` с математическими функциями из стандартной библиотеки:

In [10]:
```
import math
```

Доступ к инструментам модуля осуществляется с применением синтаксиса уточнения `<имя модуля>.<имя атрибута>`. Например,

In [12]:
```
math.sin(math.pi/2)
```

Out[12]:
```
1.0
```

Импортируем расширение `numpy` и создадим псевдоним `np` для доступа к инструментам модуля, не используя имя `numpy`

In [14]:
```
import numpy as np
```

Расширение `numpy` основано на работе со структурой данных массив (`ndarray`). Массив (`ndarray`) не является встроенным типом данных Python, он является базовым типом расширения `numpy` .

Импортируем модуль `pyplot` из пакета `matplotlib` для построения графиков. Для этого применим синтаксис `<имя пакета>.<имя модуля в пакете>` и создадим псевдоним имени модуля `plt`

In [17]:
```
import matplotlib.pyplot as plt
```

Импортируем модуль `constants` из пакета расширений `scipy` с псевдонимом `const` . Модуль `constants` предоставляет доступ к значениям физических констант

In [19]:
```
import scipy.constants as const
```

Импортируем модуль `sympy` для символьных вычислений в Python

In [21]:
```
import sympy
```

Этап 2. Создание переменных

Переменные создаются при первом присваивании им значений. Слева от оператора присваивания (`=`) записывается переменная, справа — выражение, результатом вычисления которого является объект: `переменная = объект` . Имена переменных рекомендовано задавать в нижнем регистре с использованием знака нижнего подчеркивания для смыслового разделения имени на части, например `first_variable = 1` . Объявлять переменные перед их инициализацией не нужно.

После присваивания всякий раз, когда переменная появляется в коде, она заменяется на объект, на который ссылается. Нельзя использовать переменную, которой не присвоено значение.

In [24]:
```
h_start = 1 # вертикальная составляющая положения тела в момент запуска
h_end = 3.0E+0 # вертикальная составляющая положения тела в конечный момент движ
T = 3.0 # время полета
s_end = 5.0e0 # горизонтальное перемещение тела за время полета
```

Отобразить значение переменной можно, указав ее имя в ячейке ввода в последней строке или с использованием встроенной функции `print`

In [26]:
```
h_start # переменная не отобразится
h_end # переменная отобразится
```

Out[26]:
```
3.0
```

In [27]:
```
print(h_start, h_end)
```

```
1 3.0
```

Определим значение ускорения свободного падения для дальнейших расчетов:

In [29]:
```
g = const.g
g, const.unit('standard acceleration of gravity')
```

Out[29]:
```
(9.80665, 'm s^-2')
```

\color{red}\text{Приведите} несколько примеров физических констант из модуля `constants` пакета расширений `scipy`.

Временной отрезок $[0, T]$ представим в виде последовательности чисел. Создадим последовательность равномерно распределенных чисел на отрезке $[0, T]$ с шагом 0.01 двумя способами: на основе встроенного типа список (`list`) и на основе массива (`ndarray`) из расширения `numpy`

In [32]:
```
step = 0.01
t_list = [0 + i*step for i in range(int(T/0.01))]
t_array = np.arange(0,T,step)
```

\color{red}\text{Сформулируйте} спецификации функций `range` и `arange`, приведите примеры.

In [34]:
```
#?range
```

In [35]:
```
#?np.arange
```

Запомните, что после создания списка (`list`) его размер изменить можно, после создания массива (`ndarray`) его размер изменить нельзя.

Этап 3. Определение функции перемещения и нахождение угла запуска тела

Полагаем, что начало прямоугольной декартовой системы координат соответствует уровню земли. Тогда в начальный момент движения тела $t = 0$ имеем, что

$$s_x(0) = 0, \quad s_y(0) = h_{start}.$$

Для построения функции перемещения $(s_x(t), s_y(t))$ воспользуемся вторым законом Ньютона

$$m\mathbf{a} = \mathbf{F},$$

где m — масса тела, ускорение движения тела $\mathbf{a} = (s_x''(t), s_y''(t))$ и приложенная сила $\mathbf{F} = m(0, -g)$. В результате имеем два уравнения, что

$$s_x''(t) = 0, \quad s_y''(t) = -g.$$

На основании уравнений с учетом начальных условий можно записать аналитический вид для функций $s_x(t)$ и $s_y(t)$

$$s_x(t) = s_x(0) + v0_x t = 0 + v0_x t,$$

$$s_y(t) = s_y(0) + v0_y t - \frac{g}{2}t^2 = h_{start} + v0_y t - \frac{g}{2}t^2,$$

где $\mathbf{v0} = (v0_x, v0_y)$ обозначает неизвестную скорость движения тела в начальный момент времени $t = 0$.

\color{red}\text{Напишите} подробно, как из второго закона Ньютона и начальных условий получен аналитический вид функции перемещения $(s_x(t), s_y(t))$. Объяснения оформите в тексте документа с лабораторной работой.

\color{red}\text{Задание* необязательное}: как изменится вид функций перемещения, если начальный момент времени будет равен не $t = 0$, а $t = t^*$?

Из условия задачи имеем, что $s_x(T) = s_{end}$. Из этого соотношения находим значение для $v0_x$

In [44]:
```
v0_x = s_end/T
v0_x
```

Out[44]:
```
1.6666666666666667
```

Из условия задачи имеем, что $s_y(T) = h_{end}$. Из этого соотношения находим значение для $v0_y$

In [46]:
```
v0_y = (h_end-h_start+g/2*T**2)/T
v0_y
```

Out[46]:
```
15.376641666666666
```

Значение угла запуска тела α определяем из соотношения $\tan\alpha = \frac{v0_y}{v0_x}$

In [48]:
```
alpha = math.atan(v0_y/v0_x)
alpha, math.degrees(alpha)
```

Out[48]:
```
(1.462828312588189, 83.81388846354714)
```

Результирующая функция перемещения имеет следующий вид

In [50]:
```
f's_x(t) = {v0_x} t'
```

Out[50]:
```
's_x(t) = 1.6666666666666667 t'
```

In [51]:
```
f's_y(t) = {h_start} + {v0_y:.6} t - {g:.6}/2*t\N{superscript two}'
```

Out[51]:
```
's_y(t) = 1 + 15.3766 t - 9.80665/2*t²'
```

Этап 4. Построение графика траектории движения тела

Построение графиков функций осуществляется по координатам точек графика. Для этого нужно создать последовательность значений x-координат точек графика

функции и последовательность соответствующих значений y-координат точек графика функции.

На Этапе 2 было создано две последовательности равномерно распределенных чисел на отрезке $[0, T]$ с шагом 0.01 для переменной времени t

In [55]:
```
t_list, t_array; # символ ; в конце выражения предотвращает вывод результата вып
```

Определим списки чисел, соответствующие горизонтальным и вертикальным составляющим положения тела в моменты времени `t_list` с использованием формул, полученных на Этапе 3:

In [57]:
```
s_x_list = [v0_x*t for t in t_list]
s_y_list = [h_start+v0_y*t-g*t**2/2 for t in t_list]
```

Определим массивы чисел, соответствующие горизонтальным и вертикальным составляющим положения тела в момент времени `t_array` :

In [59]:
```
s_x_array = v0_x*t_array
s_y_array = h_start+v0_y*t_array-g*t_array**2/2
```

Обратите внимание, что арифметические действия ($+$, $-$, $*$, $/$, $**$) с массивами `ndarray` выполняются **поэлементно** без использования дополнительного синтаксиса.

Строим график траектории движения тела, используя инструменты модуля `pyplot` пакета `matplotlib` . Смотрите **Help | Matplotlib Reference** для справочной информации по используемым функциям: `figure` , `plot` , `title` , `xlabel` , `ylabel` , `axis` , `xlim` , `ylim` .

In [62]:
```
plt.figure()
plt.plot(s_x_list,s_y_list,'b-',lw=5)
plt.title('Вертикальное перемещение тела')
plt.xlabel('x (m)'); plt.ylabel('y (m)') # символ ; позволяет записывать несколь

plt.axis([0, s_end, 0, h_end+11])
```

Out[62]:
```
(0.0, 5.0, 0.0, 14.0)
```

При вызове функции `plot` из `pyplot` можно использовать как списки, так и массивы.

In [64]:
```
plt.figure()
plt.plot(s_x_array,s_y_array,color='green',lw=3)
plt.title('Вертикальное перемещение тела')
plt.xlabel('x (m)'); plt.ylabel('y (m)')

# альтернативный синтаксис для установки пределов по осям
plt.xlim(0, s_end)
plt.ylim(0, h_end+11);
```

Задание 1.2. Определение момента времени касания телом земли

Для задачи о падении тела из Задания 1.1 найдите момент времени касания телом земли (время нахождения тела в воздухе).

Выполнение задания осуществите на основе двух подходов.

Подход 1 (**точное решение**): время касания телом земли находим как точное решение уравнения $s_y(t) = 0$;

Подход 2 (**приближенное решение**)

- Постройте последовательность s_y на временном отрезке $[0, t_{stop}]$, чтобы момент касания тела земли t^* попадал в выбранный отрезок, т.е. $t^* \in [0, t_{stop}]$. Отметим, что $t_{stop} > T$, где значение T задается в Задании 1.1.
- Найдите в построенной последовательности s_y два соседних элемента последовательности, значения которых меняют знак. Т. е. найдите значение индекса k, для которого $s_y[k]s_y[k+1] < 0$. Это будет означать, что $0 \in (s_y[k], s_y[k+1])$.
- Постройте прямую линию $s_{lin}(t)$, проходящую через две точки $(t[k], s_y[k])$ и $(t[k+1], s_y[k+1])$ и найдите значение $t^* \in (t[k], t[k+1])$ из условия $s_{lin}(t^*) = 0$. Найденное значение t^*, будет соответствовать точке $(t^*, 0)$ на прямой $s_{lin}(t)$. \color{red}\text{Приведите} в документе формулы для прямой $s_{lin}(t)$ и для вычисления t^*.

\color{red}\text{Сравните} точное решение (*Подход 1*) и приближенное решение (*Подход 2*), вычислив относительную ошибку.

Выполнение Задания 1.2

Рассмотрим *Подход 1* выполнения Задания 1.2.

Для решения алгебраического уравнения $s_y(t) = 0$ воспользуемся функцией `solve` из модуля `sympy`, которая предназначена для решения алгебраических уравнений и систем уравнений. Для выполнения функции `solve` предварительно с помощью функции `symbols` необходимо создать **символьную переменную**, относительно которой будет определено уравнение.

In [69]:
```
t = sympy.symbols('t')
result = sympy.solve(h_start+v0_y*t-g*t**2/2, t)
result
```

Out[69]:
```
[-0.0637382309488101, 3.19970039267921]
```

In [70]:
```
t_star_exact = result[-1]
t_star_exact
```

Out[70]:
$$3.19970039267921$$

\color{red}\text{Сформулируйте} спецификации функций `symbols` и `solve` из модуля `sympy`. \color{red}\text{Приведите} два собственных примера решения алгебраических уравнений.

In [72]:
```
# ?sympy.solve
```

\color{red}\text{Выполните} Задание 1.2 на основе *Подхода 2*:

- для поиска индекса k, для которого $s_y[k]s_y[k+1] < 0$, используйте цикл `for` по последовательности s_y в сочетании с `enumerate`;
- для вычисления t^* используйте построенную формулу.

\color{red}\text{Задание* необязательное}: предложите подход для вычисления дальности полета до момент касания телом земли.

Задание 1.3. Построение секущей, касательной и нормали к графику функции

\color{red}\text{Постройте} в одной графической области:

- график траектории движения тела $(s_x(t), s_y(t))$ для $t \in [0, T]$ из Задания 1.1;

- график секущей прямой к траектории движения тела, проходящей через начальную точку при $t = 0$ и конечную точку при $t = T$;
- график касательной прямой к траектории движения тела в начальной точке при $t = 0$;
- график нормальной прямой к траектории движения тела в начальной точке при $t = 0$.

Построение графиков функций осуществляется по координатам точек графика. Для построения графиков прямых линий (секущая, касательная, нормаль) используйте только две точки.

Выполнение Задания 1.3

Подготовим необходимые данные для построения секущей прямой к траектории движения тела.

Воспользуемся *векторно-параметрическим уравнением прямой* для заданной точки p_{start}, лежащей на прямой, и заданного направляющего вектора a

$$p(t) = p_{start} + at.$$

Направляющий вектор можно определить как $a = p_{end} - p_{start}$, где точка p_{end} лежит на прямой и отлична от точки p_{start}.

Введем новые переменные для описания начальной точки p_{start} и конечной точки p_{end} через их координаты. Для новых переменных используем тип массив (`ndarray`) из `numpy` . Представление координат точек в виде массивов иногда эффективнее, чем представление в виде списков, для дальнейшей работы с координатами.

In [82]:
```
p_start = np.array([s_x_array[0],s_y_array[0]])
p_end = np.array([s_x_array[-1],s_y_array[-1]])
p_start, p_end
```

Out[82]:
```
(array([0., 1.]), array([4.98333333, 3.13994275]))
```

Вычислим направляющий вектор:

In [84]:
```
a = p_end-p_start
```

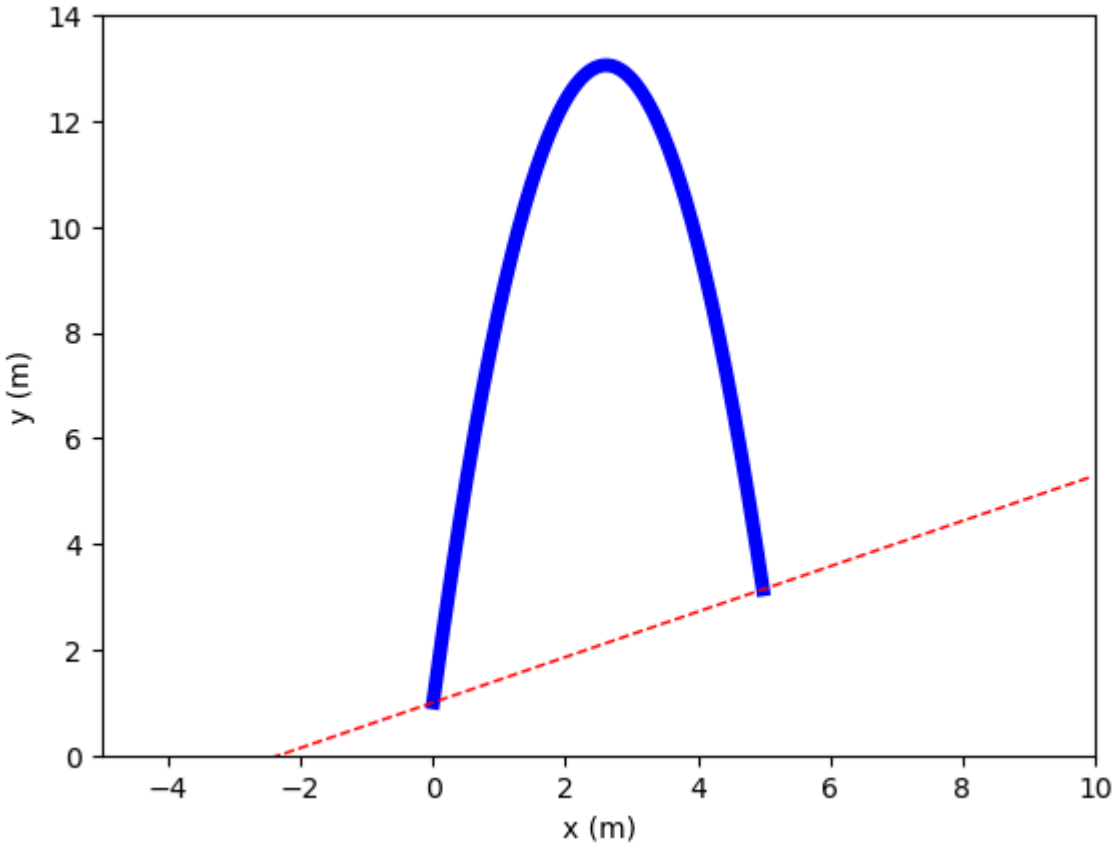
Для построения графика прямой достаточно задания двух точек, лежащих на прямой:

In [86]:
```
p = [p_start + a*t for t in [-1, 2]]
p = np.array(p)
```

Построим график траектории движения тела и график секущей прямой в одной графической области:

In [88]:
```
fig = plt.figure()
plt.plot(s_x_array,s_y_array,'b-',lw=5)
plt.plot(p[:,0],p[:,1],'--',color='red',lw=1) # обратите внимание не специальную
plt.xlabel('x (m)'); plt.ylabel('y (m)')

plt.axis([-5, s_end+5, 0, h_end+11]);
```



\color{red}\text{Постройте} график касательной прямой к траектории движения тела в одной графической области с траекторией движения и секущей прямой. Для построения касательной прямой воспользуйтесь уравнением прямой по точке $p_{start}(x_{start}, y_{start})$ и угловому коэффициенту k:

$$y = k(x - x_{start}) + y_{start}.$$

\color{red}\text{Поясните} в документе, как Вы определяете угловой коэффициент k .

\color{red}\text{Сформулируйте} в документе уравнение нормальной прямой к траектории движения тела в момент времени $t = 0$, используя свойство перпендикулярности каксательной и нормальной прямых.

\color{red}\text{Постройте} график нормальной прямой к траектории движения тела в момент времени $t = 0$ в одной графической области с траекторией движения, секущей прямой и касательной прямой.

Задание 1.4. Построение графиков функций

$\color{red}\text{Постройте}$ графики функций на плоскости по координатам точек согласно варианту. Функция задана:

а) полярным уравнением $\rho = \rho(\varphi)$ в полярной системе координат (φ, ρ);

б) параметрическими уравнениями $x = x(t), y = y(t)$ в декартовой системе координат (x, y).

Построение графиков функций осуществляется по координатам точек графика. Для этого нужно создать последовательность значений x (φ)-координат точек графика функции и последовательность соответствующих значений y (ρ)-координат точек графика функции.

Варианты задания:

1. а) *Декартов лист* $\rho = \frac{3a\sin\varphi\cos\varphi}{\sin^3\varphi+\cos^3\varphi}$, $a \in \mathbb{R}$

   б) $x = \frac{3t^2+1}{3t^3}$, $y = \sin\left(\frac{t^3}{3} + t\right)$.

2. а) *Роза* $\rho = a\sin\left(\frac{m}{n}\varphi\right)$, $a \in \mathbb{R}$, m, n — натуральные нечетные, $m \neq n$

   б) $x = \sqrt{1-t^2}$, $y = \tan\sqrt{1+t}$.

3. а) *Кардиоида* $\rho = 2a(1-\cos\varphi)$, $a \in \mathbb{R}$

   б) $x = \arcsin(\sin t)$, $y = \arccos(\cos t)$.

4. а) *Роза* $\rho = a\sin(2k\varphi)$, $a \in \mathbb{R}$, $k \in \mathbb{N}$

   б) $x = \ln\left(t + \sqrt{t^2+1}\right)$, $y = t\sqrt{t^2+1}$.

5. а) *Роза* $\rho = a\sin\left(\frac{m}{n}\varphi\right)$, $a \in \mathbb{R}$, $m \in \mathbb{N}$, $n \in \mathbb{N}$, m либо n — четное, $m \neq n$

   б) $x = \sqrt{2t-t^2}$, $y = \arcsin(t-1)$.

6. а) *Улитка Паскаля* $\rho = 2a\cos\varphi \pm \ell$, $a \in \mathbb{R}$, $\ell \in \mathbb{R}$

   б) $x = \cot\left(2e^t\right)$, $y = \ln\left(\tan e^t\right)$.

7. а) *Конхоида Никомеда* $\rho = \frac{a}{\sin\varphi} + \ell$, $a \in \mathbb{R}$, $\ell \in \mathbb{R}$

   б) $x = \ln(\cot t)$, $y = \frac{1}{\cos^2 t}$.

8. а) *Роза* $\rho = a\sin((2k+1)\varphi)$, $a \in \mathbb{R}$, $k \in \mathbb{N}_0$

   б) $x = \arctan e^{t/2}$, $y = \sqrt{e^t+1}$.

9. а) $\rho = \frac{a}{m+n\sin\varphi}$, $a \in \mathbb{R}$, $m \in \mathbb{N}$, $n \in \mathbb{Z}$

   б) $x = \ln\sqrt{\frac{1-t}{1+t}}$, $y = \sqrt{1-t^2}$.

10. а) *Строфоида* $\rho = \frac{a}{\cos\varphi} + a\tan\varphi$, $a \in \mathbb{R}$

    б) $x = \ln\frac{1}{\sqrt{1-t^4}}$, $y = \arcsin\frac{1-t^2}{1+t^2}$.

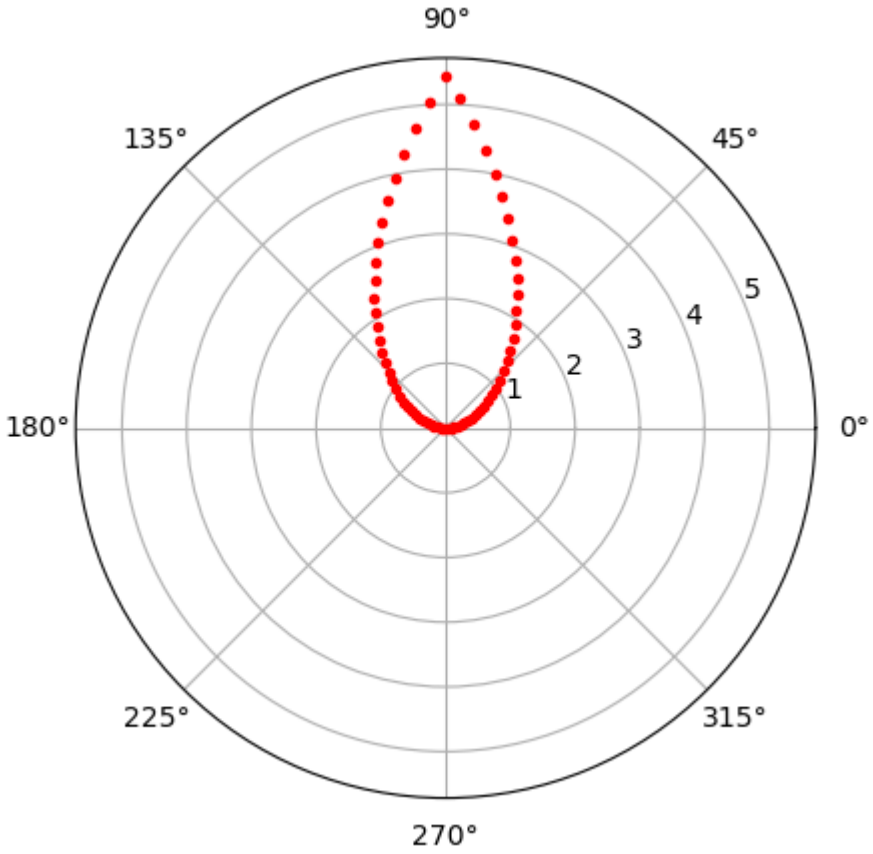
Пример построения графика функции в полярной системе координат

Приведем пример построения графика функции $\rho(\varphi) = \varphi^3 + \varphi$, $\varphi \in [-\pi/2, \pi/2]$ в полярной системе координат.

Отметим, что полярный радиус ρ может принимать только неотрицательные значения, поэтому график функции строится для функции $|\rho|$ с дополнительной корректировкой значения угла φ на $\varphi + \pi$ для отрицателных значений радиуса

```
plt.plot(phi+(r<0)*np.pi, abs(r), 'r.')
```

In [96]:
```
plt.figure()
# установка способа отображения осей
plt.axes(projection='polar')
# подготовка данных
phi = np.arange(-np.pi/2, np.pi/2, 0.05)
r = phi**3+phi
# построение графика
plt.plot(phi + (r<0)*np.pi,abs(r),'r.');
```


90°
135°
45°
180°
0°
225°
315°
270°
1
2
3
4
5