Лабораторная работа 3

Анимация движения секущей прямой к заданной линии на плоскости

Компьютерная математика II, ММФ, БГУ

Лаврова О.А., март 2024

Навыки

1. базовый **Python**: оператор `def` и `lambda`-выражение для создания функции; оператор `global` для изменения области видимости переменной; строки документации для функций; функция как аргумент другой функции
2. модуль `pyplot` из пакета `matplotlib`: `figure`, `axes`, `axis`, `plot`; графические объекты типа `Line2D`: методы `get_xdata`, `get_ydata`, `set_data`
3. модуль `animation` из пакета `matplotlib`: `funcAnimation`
4. расширение `numpy`: тип данных массив `ndarray`, `array`, `arange`, `transpose`, `min`, `max`
5. **математика**:
   - способы задания прямой на плоскости
     - векторно-параметрическое уравнение
     - уравнение по точке и угловому коэффициенту

Используем модуль ipyml и функцию FuncAnimation

Информация 1. Для корректного отображения анимации в JupyterLab необходимо установить в Anaconda **модуль `ipympl`** (Matplotlib jupyter extension), так как в стандартную установку модуль `ipympl` не входит. Установить новый модуль можно через **Anaconda Navigator | Environments**. Импортировать модуль `ipympl` в блокноте Jupyter Notebook не нужно.

Информация 2. Для построения анимации будем использовать функцию **`FuncAnimation`** из модуля `animation` пакета `matplotlib`.

```
from matplotlib.animation import FuncAnimation
```

In [1]:
```
# ?FuncAnimation
```

`FuncAnimation(fig, func, frames=None, init_func=None, fargs=None, save_count=None, *, repeat=True, interval=200, **kwargs)`

- обязательный аргумент `fig` : графическое окно, в котором будет отображаться анимация;
- обязательный аргумент `func` : *функция* одного обязательного аргумента, которая будет вызываться в каждом кадре;
- НЕобязательный аргумент `frames` со стандартным значением `None` : последовательность, элементы которой будут определять значение аргумента функции `func` при последовательной индексации последовательности от начала до конца;
- НЕобязательный аргумент `init_func` со стандартным значением `None` : *функция*, которая вызывается перед началом анимации;
- НЕобязательный аргумент `repeat` со стандартным значением `True` : нужно ли повторять анимацию снова после отрисовки всех кадров;
- НЕобязательный аргумент `interval` со стандартным значением `200` : длительность задержки в миллисекундах между кадрами анимации.

Обратите внимание, что значениями аргументов `func` и `init_func` являются функции. Также обратите внимание на использование `*` в качестве аргумента функции `FuncAnimation` .

Задание 3.1. Анимированное построение линии на плоскости

а) Создайте анимацию построения графика некоторой явно заданной аналитической функции $y = y(x)$ по значениям x, последовательно изменяющимся от x_{min} до x_{max} с шагом $step$. Выполните задание для $y(x) = (x-5)^3$, $x_{min} = 0$, $x_{max} = 10$, $step = 0.1$.

б) Создайте анимацию построения графика параметрически заданной функции $x = x(t), y = y(t)$ по значениям t, последовательно изменяющимся от t_{min} до t_{max} с шагом $step$. Выполните задание для функции из Задания 1.3 в Лб1, согласно Вашему варианту, для $t_{min} = 0$, $t_{max} = 10$, $step = 0.1$.

Реализация Задания 3.1а

Сначала импортируем расширение `numpy` , а также модуль `pyplot` из пакета `matplotlib` :

In [2]:
```
import numpy as np
```

In [3]:
```
import matplotlib.pyplot as plt
```

Импортируем функцию `FuncAnimation` из модуля `animation` пакета `matplotlib`

In [4]:
```
from matplotlib.animation import FuncAnimation
```

Вызовем специальную команду JupyterLab, необходимую для корректного отображения анимации в интерактивном документе

In [5]:
```
%matplotlib widget
```

Будем строить анимацию для функции вида $y(x) = (x-5)^3$ по значениям x, изменяющимся от $x_{min} = 0$ до $x_{max} = 10$ с шагом $step = 0.1$.

Определим функцию $y(x)$ с применением оператора `def`

In [6]:
```
def y(x):
    return (x-5)**3
```

Функцию $y(x)$ также можно определить с помощью `lambda` -выражения

In [7]:
```
y = lambda x: (x-5)**3
```

Создадим переменные

In [8]:
```
x_min = 0.; x_max = 10.; step = 0.1
```

Создадим массив `x_array` равномерно распределенных чисел на отрезке $[x_{min}, x_{max}]$ с шагом $step$ с помощью функции `arange` . Создадим массив `y_array` соответствующих значений y с помощью пользовательской функции `y`

In [9]:
```
x_array = np.arange(x_min, x_max, step)
y_array = y(x_array)
```

Из двух массивов `x_array` и `y_array` создадим матрицу `matrix` координат точек графика функции $y(x)$. Матрица состоит из двух столбцов. Первый столбец матрицы содержит x-координаты точек, второй столбец -- y-координаты точек

In [10]:
```
matrix = np.transpose([x_array, y_array])
```

С помощью функции `figure` из модуля `pyplot` создадим графическое окно `fig1` , в котором в дальнейшем будет отображаться анимация

In [11]:
```
fig1 = plt.figure()
```

Figure

С помощью функции `axes` из модуля `pyplot` создадим графическую область `ax1` и зададим для нее пределы по осям

In [12]:
```
y_min, y_max = np.min(y_array), np.max(y_array)

ax1 = plt.axes()
plt.axis([x_min-1, x_max+1, y_min-1, y_max+1])
```

Out[12]:
```
(-1.0, 11.0, -126.0, 118.64900000000003)
```

С помощью функции `plot` из пакета `pyplot` создадим в графической области `ax1` графический объект типа `Line2D` синего цвета, координаты которого пока не определены

In [13]:
```
line1, = ax1.plot([],[],'b')
print(type(line1))
line1.get_xdata(), line1.get_ydata()
```

Out[13]:
```
<class 'matplotlib.lines.Line2D'>
(array([], dtype=float64), array([], dtype=float64))
```

In [14]:
```
?line1.get_xdata
```

```
Signature: line1.get_xdata(orig=True)
Docstring:
Return the xdata.

If *orig* is *True*, return the original data, else the
processed data.
File:      c:\users\olga\anaconda3\lib\site-packages\matplotlib\lines.py
Type:      method
```

Определим пользовательскую функцию одного аргумента `at_frame1` , которая будет вызываться в каждом кадре анимации.

Единственный аргумент функции `at_frame1` является массивом из x и y координат точки графика функции. Функция `at_frame1` добавляет к графическому объекту `line1` точку с координатами (x, y)

In [15]:
```
def at_frame1(point):
    """добавляет к объекту line1 точку с координатами (point[0],point[1])

    Arguments :

    point : массив из двух элементов

    Returns : None
    """
    x_coord = list(line1.get_xdata())
    y_coord = list(line1.get_ydata())

    x_coord.append(point[0])
    y_coord.append(point[1])

    line1.set_data(x_coord, y_coord)
```

In [16]:
```
?line1.set_data
```

```
Signature: line1.set_data(*args)
Docstring:
Set the x and y data.

Parameters
----------
*args : (2, N) array or two 1D arrays
File:      c:\users\olga\anaconda3\lib\site-packages\matplotlib\lines.py
Type:      method
```

С помощью функции `help` и команды `?` извлекем строки документации для пользовательской функции `at_frame1`

In [17]:
```
help(at_frame1)
```

```
Help on function at_frame1 in module __main__:

at_frame1(point)
    добавляет к объекту line1 точку с координатами (point[0],point[1])

    Arguments :

    point : массив из двух элементов

    Returns : None
```

In [18]:
```
?at_frame1
```

```
Signature: at_frame1(point)
Docstring:
добавляет к объекту line1 точку с координатами (point[0],point[1])

Arguments :

point : массив из двух элементов

Returns : None
File:      c:\users\olga\appdata\local\temp\ipykernel_18080\3908975406.py
Type:      function
```

Атрибут `__doc__` объекта-функции содержит строки документации

In [19]:
```
print(at_frame1.__doc__)
```

```
добавляет к объекту line1 точку с координатами (point[0],point[1])

    Arguments :

    point : массив из двух элементов

    Returns : None
```

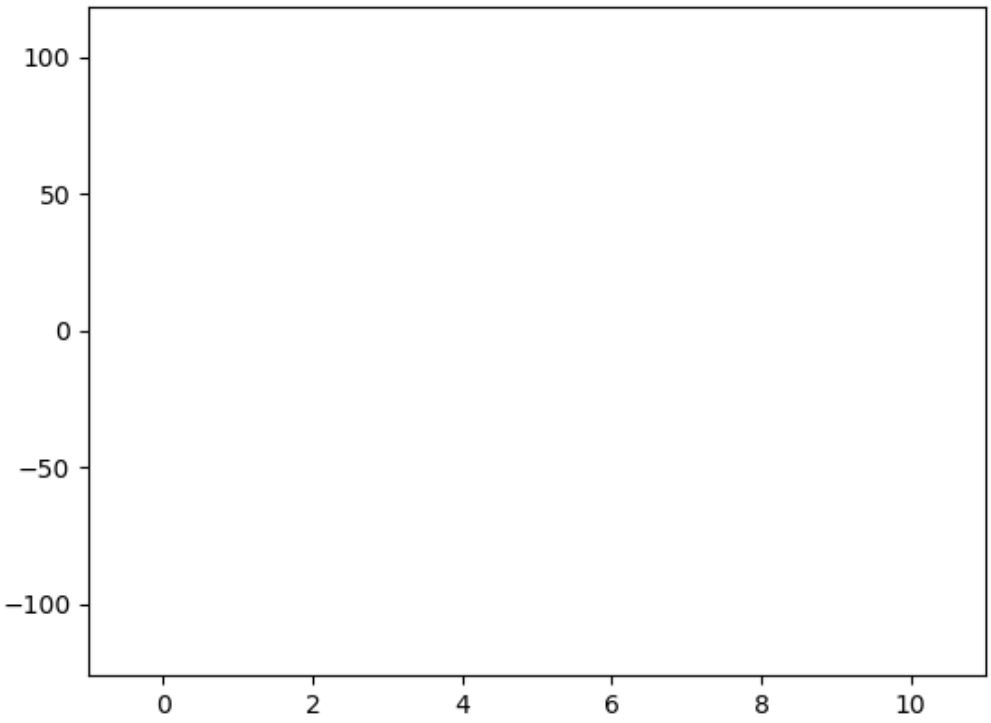
Для построения анимации полагаем аргумент `frames` для функции `FuncAnimation` следующим образом: `frames=matrix` . Это означает, что количество кадров анимации будет совпадать с количеством строк матрицы `matrix` . При этом для каждого кадра анимации будет вызываться функция `at_frame1` со значением аргумента, равным массиву со значениями строки матрицы `matrix` , соответствующей номеру кадра.

Важная информация для построения анимации: вызов функций `figure` , `axes` , `plot` , необходимых для анимации, а также вызов функции `FuncAnimation` должны располагаться в одной ячейке кода.

In [20]:
```
fig1 = plt.figure()
ax1 = plt.axes()
plt.axis([x_min-1, x_max+1, y_min-1, y_max+1])

line1, = ax1.plot([], [],'b')

FuncAnimation(fig1, at_frame1, frames=matrix, repeat=False, interval=15)
```

Out[20]: <matplotlib.animation.FuncAnimation at 0x1605da30810>

Figure



Для повторного воспроизведения анимации нужно запустить предыдущую ячейку кода снова.

Реализация Задания 3.1b

Создайте аналогичную анимацию построения графика параметрически заданной функции $x = x(t), y = y(t)$ по значениям t, последовательно изменяющимся от t_{min} до t_{max} с шагом $step$. Выполните задание для функции из Задания 1.3 в Лб1, согласно Вашему варианту, для $t_{min} = 0$, $t_{max} = 10$, $step = 0.1$.

Задание 3.2. Движение точки по линии на плоскости

Создайте анимацию движения точки по графику функции $x = x(t), y = y(t)$ по значениям t, последовательно изменяющимся от t_{min} до t_{max} с шагом $step$. Функция $x = x(t), y = y(t)$ и значения переменных t_{min}, t_{max}, $step$ задаются такими же, как и при выполнении Задания 3.1b.

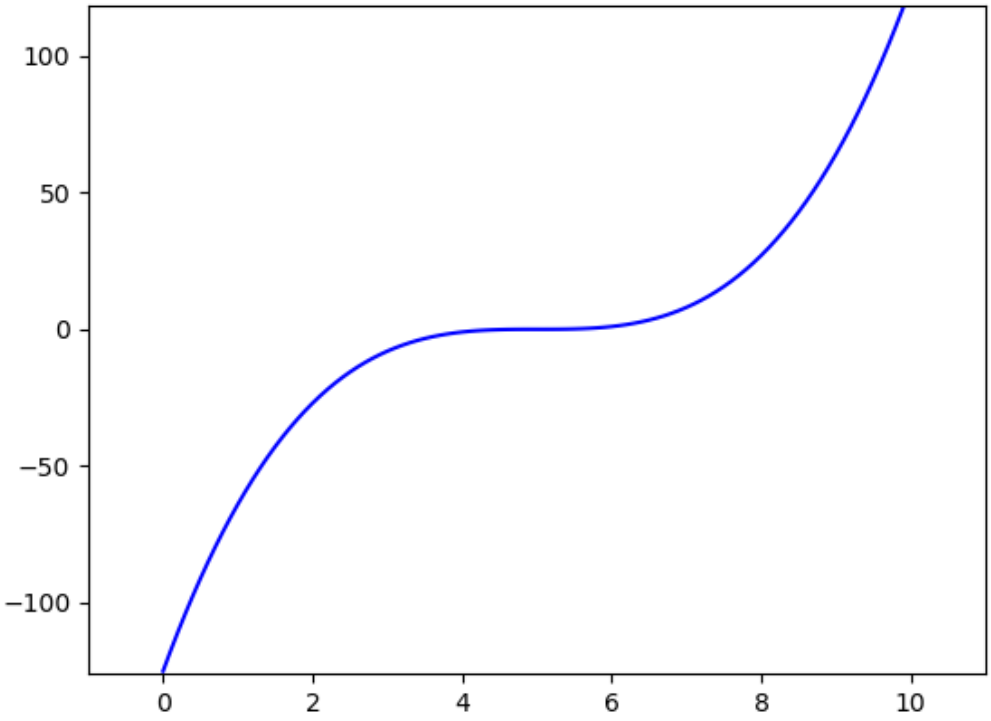
Реализация Задания 3.2 для явно заданной функции

Последовательно выполним следущие шаги с использованием функций из модуля `pyplot` :

- с помощью функции `figure` создадим графическое окно, в котором в дальнейшем будет отображаться анимация
- с помощью функции `axes` создадим графическую область и зададим для нее пределы по осям с помощью функции `axis`
- с помощью функции `plot` создадим в графической области графический объект типа `Line2D` синего цвета `'b'` , координаты которого описывают аналитическую функцию
- с помощью функции `plot` создадим в графической области графический объект типа `Line2D` зеленого цвета с маркером в виде кружка `'go'` , координаты которого пока неизвестны

In [21]:
```
fig2 = plt.figure()

ax2 = plt.axes()
plt.axis([x_min-1, x_max+1, y_min-1, y_max+1])

line1, = ax2.plot(x_array, y_array,'b')  # объект для графика функции
line2, = ax2.plot([], [],'go') # объект для точки
```

Figure



Определим пользовательскую функцию одного аргумента `at_frame2` , которая будет вызываться в каждом кадре анимации.

Единственный аргумент функции `at_frame2` является массивом из x и y координат точки графика функции. Функция `at_frame2` задает графический объект `line2` единственной точкой с координатами (x, y)

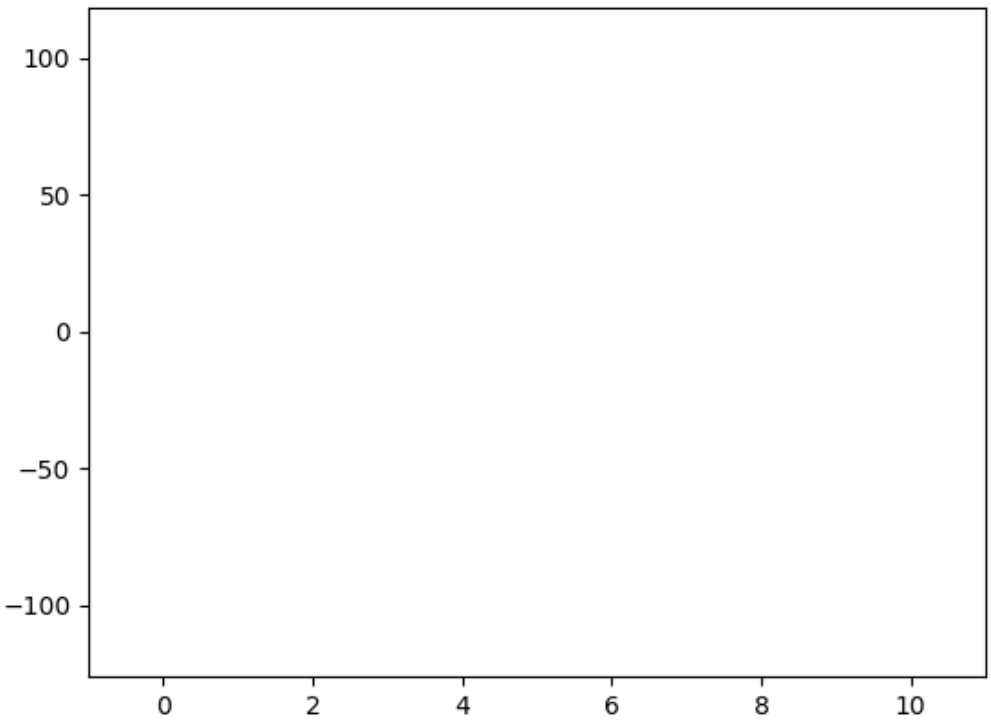
In [22]:
```
def at_frame2(point):
    """задает объект line2 точкой с координатами (point[0],point[1])

    Arguments :

    point : массив из двух элементов

    Returns : None
    """
    line2.set_data(point)
```

Полагаем аргумент `init_func` функции `FunAnimation` равным пользовательской функции `init` : `init_func=init` . Функция `init` будет вызываться перед началом анимации.

In [23]:
```
fig2 = plt.figure()
ax2 = plt.axes()
plt.axis([x_min-1, x_max+1, y_min-1, y_max+1])

def init():
    """создает начальное состояние графической области"""
    global line2 # переменная сделана глобальной, чтобы она могла изменяться внутри

    line1, = ax2.plot(x_array, y_array,'b')
    line2, = ax2.plot([], [],'go')
    plt.legend(['$y(x) = (x-5)^3$','Movable point'])

FuncAnimation(fig2, at_frame2, frames=matrix, init_func=init, repeat=False, interva
```

Out[23]: <matplotlib.animation.FuncAnimation at 0x1605d99b250>

Figure



Создайте аналогичную анимацию для функции $x = x(t), y = y(t)$ из Задания 3.1b для $t_{min} = 0, t_{max} = 10, step = 0.1$.

Задание 3.3. Движение секущей прямой к заданной линии на плоскости

Линия на плоскости задана графиком параметрической функции $x = x(t)$, $y = y(t)$ для $t \in [t_{min}, t_{max}]$ из Задания 3.1b. Начальная точка A с координатами $(x(t_{min}), y(t_{min}))$ является неподвижной точкой. Точка B движется последовательно по линии от конечной точки кривой с координатами $(x(t_{max}), y(t_{max}))$ к неподвижной точке A.

Создайте анимацию движения секущей прямой, проходящей через точки A и B до момента совпадении координат точек A и B, когда секущая прямая становится касательной прямой к заданной линии в начальной точке A.

Реализация Задания 3.3 для явно заданной функции

Перед началом анимации графическая область должна содержать следующие графические объекты:

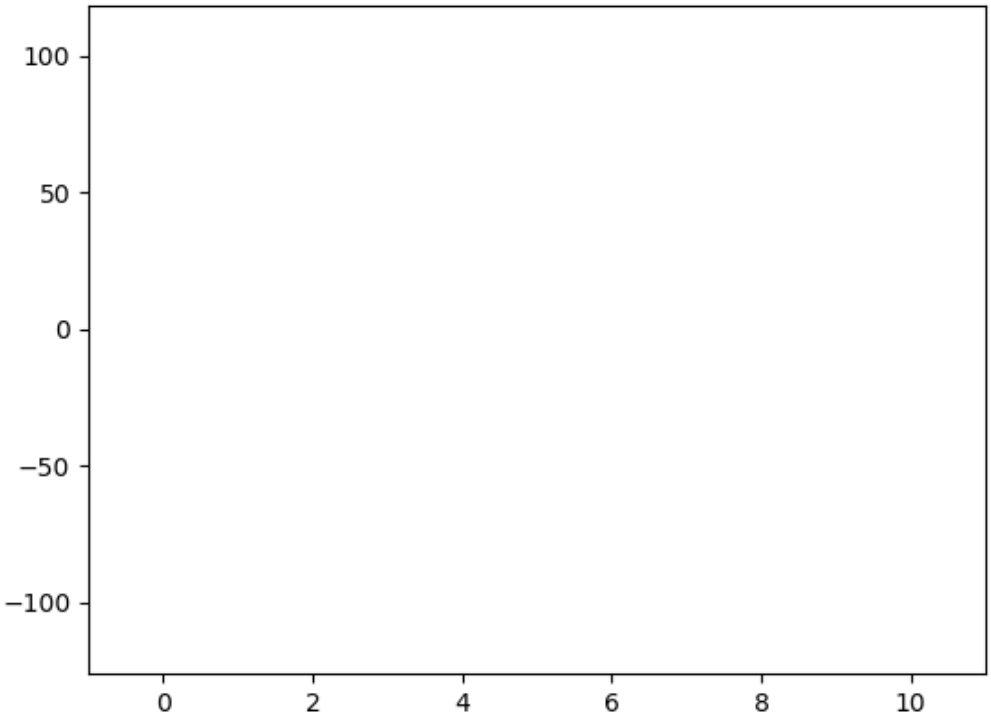
- график заданной функции, которая определяет траекторию движения подвижной точки B;
- неподвижную точку A;
- начальное положение подвижной точки B;
- секущую прямую, проходящую через точки A и B.

Начальное состояние графической области реализуем с помощью пользовательской функции `init` .

In [24]:
```
fig3 = plt.figure()
ax3 = plt.axes()
plt.axis([x_min-1, x_max+1, y_min-1, y_max+1])

def init():
    """создает начальное состояние графической области перед началом анимации и воз
    global point_B, secant_line # переменные сделаны глобальными, чтобы они были до

    curve, = ax3.plot(x_array, y_array,'b') # графический объект для исходной функц

    A = np.array([x_array[0], y_array[0]])
    point_A, = ax3.plot(A[0],A[1],'ro') # графический объект для неподвижной точки

    B = np.array([x_array[-1], y_array[-1]])
    point_B, = ax3.plot(B[0],B[1],'go') # графический объект для подвижной точки

    secant_p = [A + (B - A)*t for t in [-2, 2]] # две точки секущей прямой, проходя
    secant_p = np.array(secant_p)
    secant_line, = ax3.plot(secant_p[:,0], secant_p[:,1],'g') # графический объект

    plt.legend(['$y(x) = (x-5)^3$','Unmovable point A','Movable point B','Secant li

def at_frame3(point):
    """do ...

    Arguments :

    point : массив из двух элементов, задающий координаты подвижной точки

    Returns : None
    """
    ...

FuncAnimation(fig3, at_frame3, frames=matrix[::-1], init_func=init, repeat=False, i
```

Out[24]:
```
<matplotlib.animation.FuncAnimation at 0x1605d9e3f50>
```

Figure



Напишите пользовательскую функцию `at_frame3(t)`, которая будет вызываться в каждом кадре анимации, полагая, что аргумент `point` является массивом координат подвижной точки B. Напишите строки документации для пользовательской функции `at_frame3`.

Обратите внимание, что при совпадении координат неподвижной точки A и подвижной точки B уравнение для задания секущей прямой через две точки возвращает только точку A. В этом случае векторно-параметрическое уравнение прямой должно быть заменено на уравнение касательной прямой в точке A. Вычисление производной осуществите с помощью возможностей модуля `sympy`. Касательную прямую изобразите красным цветом.

Задание 3.4. Движение секущей прямой (версия 2)

Линия на плоскости задана графиком параметрической функции $x = x(t)$, $y = y(t)$ для $t \in [t_{min}, t_{max}]$ из Задания 3.1b. Точка A с координатами $(x(t_{min}), y(t_{min}))$ является подвижной точкой на заданной линии, точка B с координатами $(x(t_{max}), y(t_{max}))$ является неподвижной точкой. Точка A движется последовательно по кривой до неподвижной точки B.

Создайте анимацию движения секущей прямой, проходящей через точки A и B до момента совпадении координат точек A и B, когда секущая прямая становится касательной прямой к заданной линии в конечной точке B.