ДОДАТОК А

Лістінг програми

```
class Spline:
    """
    This class implements splines
    """

    def __init__(self, spline_base, coord_number, a, b):
        """
        :param spline_base: order of the spline
        :param coord_number: number of coordinate functions
        :param a: a-coordinate
        :param b: b-coordinate
        """

        self.kx = []
        self.ky = []
        i = -spline_base + 4
        j = -spline_base + 4
        while i <= coord_number - 1 - spline_base + 4:
            while j <= coord_number - 1 - spline_base + 4:
                self.ky.append(j)
                self.kx.append(i)
                j += 1
            i += 1
            j = -spline_base + 4
        self.spHx = a / (coord_number - spline_base + 1)
        self.spHy = b / (coord_number - spline_base + 1)

    @staticmethod
    def unit_step(x):
        """
        represents the unit step function
        :param x: x-coordinate
        :return: represents the unit step function, equal to 0 for x<0 and 1 for
x>=0.
        """
        if x < 0:
            return 0

        return 1

    def spline_base(self, x):
        """
        spline of 6 order
        :param x: x-coordinate
        :return: spline of 6 order real value
        """
        return (11 / 20 - x ** 2 / 2 + x ** 4 / 4 - x ** 5 / 12) *
(self.unit_step(x) - self.unit_step(x - 1)) + \
               (17 / 40 + (5 * x) / 8 - (7 * x ** 2) / 4 + (5 * x ** 3) / 4 - (3
* x ** 4) / 8 + x ** 5 / 24) * (
                   self.unit_step(x - 1) - self.unit_step(x - 2)) + \
               (243 / 120 - (81 * x) / 24 + (9 * x ** 2) / 4 - (3 * x ** 3) / 4
+ x ** 4 / 8 - x ** 5 / 120) * (
                   self.unit_step(x - 2) - self.unit_step(x - 3))

    def d1_spline(self, x):
        """
```

```
        first derivative of spline 6 order
        :param x: x-coordinate
        :return: first derivative of spline 6 order real value
        """
        return (-x + x ** 3 - (5 * x ** 4) / 12) * (self.unit_step(x) -
self.unit_step(x - 1)) + \
               (5 / 8 - (7 * x) / 2 + (15 * x ** 2) / 4 - (3 * x ** 3) / 2 + (5
* x ** 4) / 24) * (
                   self.unit_step(x - 1) - self.unit_step(x - 2)) + \
               (-27 / 8 + (9 * x) / 2 - (9 * x ** 2) / 4 + x ** 3 / 2 - x ** 4 /
24) * (
                   self.unit_step(x - 2) - self.unit_step(x - 3))

    def d2_spline(self, x):
        """
        second derivative of spline 6 order
        :param x: x-coordinate
        :return: second derivative of spline 6 order real value
        """
        return (-1 + 3 * x ** 2 - (5 * x ** 3) / 3) * (self.unit_step(x) -
self.unit_step(x - 1)) + \
               (-7 / 2 + (15 * x) / 2 - (9 * x ** 2) / 2 + (5 * x ** 3) / 6) * (
                   self.unit_step(x - 1) - self.unit_step(x - 2)) + \
               (9 / 2 - (9 * x) / 2 + (3 * x ** 2) / 2 - x ** 3 / 6) * (
                   self.unit_step(x - 2) - self.unit_step(x - 3))

    def spline(self, k, x, y):
        """
        coordinate function
        :param k: natural number [1,coord_number]
        :param x: x-coordinate
        :param y: y-coordinate
        :return: coordinate function real value
        """
        return self.spline_base(abs(x / self.spHx - self.kx[k-1])) *
self.spline_base(abs(y / self.spHy - self.ky[k-1]))

    def d1x(self, k, x, y):
        """
        first derivative of coordinate function with respect to x
        :param k:natural number [1,coord_number]
        :param x: x-coordinate
        :param y: y-coordinate
        :return: first derivative of coordinate function with respect to x real
value
        """
        return np.sign(x / self.spHx - self.kx[k-1]) / self.spHx *
self.d1_spline(
            abs(x / self.spHx - self.kx[k-1])) * self.spline_base(abs(y /
self.spHy - self.ky[k-1]))

    def d1y(self, k, x, y):
        """
        first derivative of coordinate function with respect to y
        :param k: natural number [1,coord_number]
        :param x: x-coordinate
        :param y: y-coordinate
        :return: first derivative of coordinate function with respect to y real
value
        """
        return np.sign(y / self.spHy - self.ky[k-1]) / self.spHy *
self.spline_base(
            abs(x / self.spHx - self.kx[k-1])) * self.d1_spline(
            abs(y / self.spHy - self.ky[k-1]))
```

```
    def d2x(self, k, x, y):
        """
        second derivative of coordinate function with respect to x
        :param k: natural number [1,coord_number]
        :param x: x-coordinate
        :param y: y-coordinate
        :return: second derivative of coordinate function with respect to x real
value
        """
        return 1 / self.spHx ** 2 * self.d2_spline(abs(x / self.spHx -
self.kx[k-1])) * self.spline_base(
            abs(y / self.spHy - self.ky[k-1]))

    def d2y(self, k, x, y):
        """
        second derivative of coordinate function with respect to y
        :param k: natural number [1,coord_number]
        :param x: x-coordinate
        :param y: y-coordinate
        :return: second derivative of coordinate function with respect to y real
value
        """
        return 1 / self.spHy ** 2 * self.spline_base(abs(x / self.spHx -
self.kx[k-1])) * self.d2_spline(
            abs(y / self.spHy - self.ky[k-1]))
```

ВІДОМІСТЬ АТЕСТАЦІЙНОЇ РОБОТИ

Позначення	Найменування	Дод. відомості
	Текстові документи	
1	Пояснювальна записка	55 с.
2	Презентаційний матеріал	25 с.
	Інші документи	
3	Роздруківки програм	3 с.
4	Рецензія	2 с.
5	Відгук керівника	1 с.

					Математичне моделювання в'язкої рідини на мові Python з використанням паралельних обчислень			
Змін	**Арк.**	**Номер докум.**	**Підп.**	**Дата**				
Розроб.		**Корсун К.О.**			**(Тема роботи) Відомість атестаційної роботи**		**Аркуш**	**Аркушів**
Перевір.		**Артюх А.В.**						
Н. контр.		**Сидоров М.В.**				**ХНУРЕ Кафедра ПМ**		
Затв.		**Тевяшев А.Д.**						