**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

Лабораторная работа

*по дисциплине «методы численного анализа»*

**«Приближение функций с помощью кубического интерполяционного сплайна»**

**Выполнил:**

*Студент 2 курса 2 группы ФПМИ*

*Китиков М.В.*

**Проверил:**

*Доцент кафедры вычислительной математики ФПМИ*

*Никифоров И.В.*

**Минск 2019**

**Постановка задачи**

*Вариант Б*

Задана функция . Необходимо интерполировать функцию кубическим сплайном на отрезке [0, 1] по n узлам (n задается как параметр).

**Решение**

Для решения задачи мною был выбран язык программирования Python. Были использованы библиотеки scipy, numpy и matplotlib. Граничные условия – равенство 0 второй производной на концах промежутка.

# Исходный код:

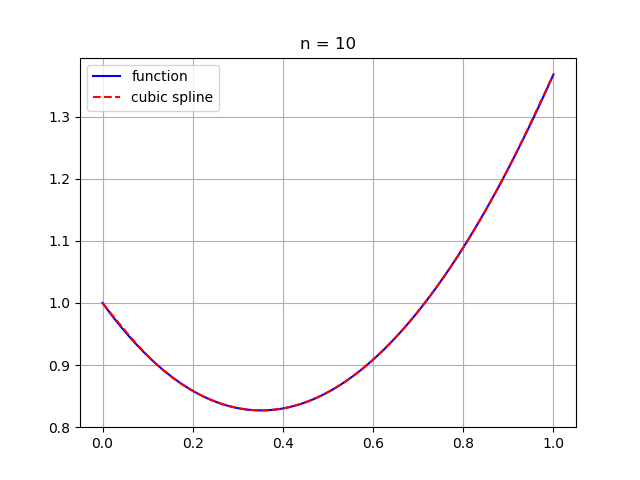
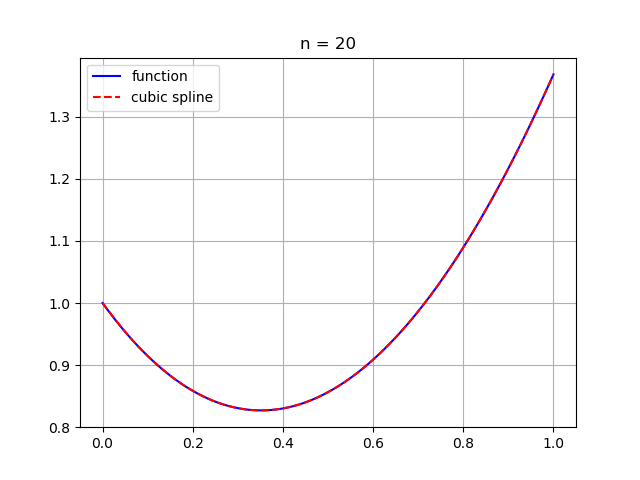
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** scipy.interpolate **import** CubicSpline  
  
  
**def** my\_func(x):  
 **return** np.exp(-x) + x\*\*2  
  
  
**def** draw\_cubic\_spline(x, y):  
 cs = CubicSpline(x, y, bc\_type=**'natural'**)  
 x\_values = np.linspace(0.0, 1.0, 100)  
 plt.grid(**True**)  
 plt.title(**'n = {}'**.format(n))  
 plt.plot(x\_values, my\_func(x\_values), **'b-'**, label=**'function'**)  
 plt.plot(x\_values, cs(x\_values), **'r--'**, label=**'cubic spline'**)  
 plt.show()  
  
  
**for** n **in** [3, 5, 10, 20]:

*# получаем узлы* nodes = np.linspace(0.0, 1.0, n)  
 values = my\_func(nodes)

*# строим кубический сплайн по n узлам* draw\_cubic\_spline(nodes, values)

# Графики:

# 



# Выводы:

Точность интерполяции выше с ростом числа узлов. При n 10 график сплайна визуально совпадает с графиком исходной функции.