**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

Лабораторная работа

*по дисциплине «методы численного анализа»*

**«Приближение функций с помощью интерполяционного полинома»**

**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 разбить исходный отрезок [0, 1] на n отрезков (n = 10, 20, 40) и на каждом отрезке приблизить функцию f(x) многочленом Лагранжа (или Ньютона) 3-й степени по узлам Чебышева. Для каждого n построить график интерполяционного многочлена. Использовать любой язык программирования.

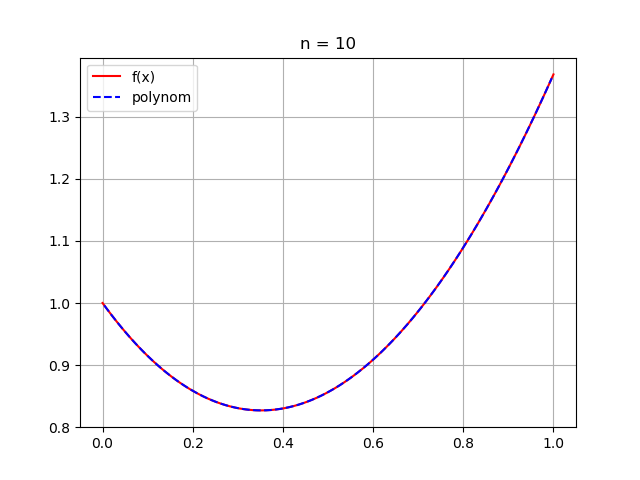
**Решение**

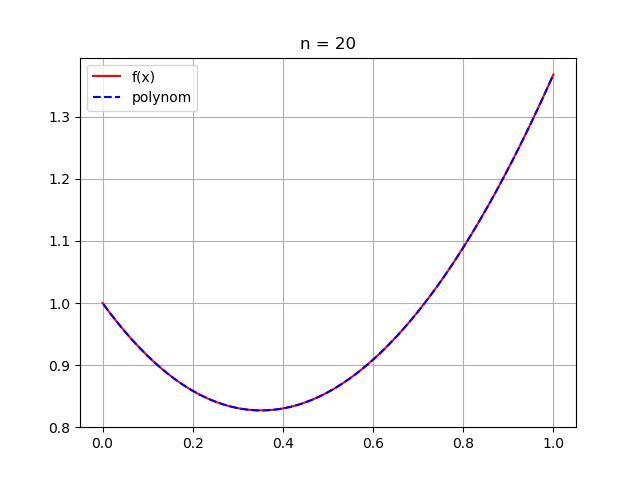
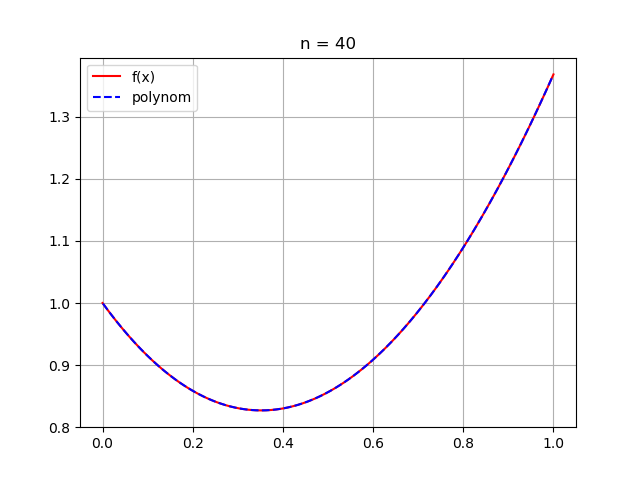
Для решения задачи мною был выбран язык программирования Python. Для построения графиков использовалась библиотека matplotlib. Интерполяция проводилась с помощью многочлена Лагранжа по узлам Чебышева (количество узлов на 1 больше степени полинома). Графики для подотрезков скомбинированы в общий график (для n = 10, 20, 40).

# Исходный код:

**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** pylab **import** \*  
  
*# значения функции (для сравнения с полиномом)*func\_x = np.linspace(0.0, 1.0, 100)  
func\_y = [exp(-x) + x\*\*2 **for** x **in** func\_x]  
  
  
*# вычисление интерполяционного полинома***def** lagrange(x, y, val):  
 res = 0  
 **for** j **in** range(len(y)):  
 num = 1; denom = 1  
 **for** i **in** range(len(x)):  
 **if** i == j:  
 **continue** num \*= val - x[i]  
 denom \*= x[j] - x[i]  
 res += y[j] \* num / denom  
 **return** res  
  
  
*# интерполяция функции по узлам Чебышева***def** interpolate(a, b, nodes\_cnt):  
 nodes\_x = np.asarray([0.5 \* (a + b) + 0.5 \* (b - a) \* cos(pi \* (2 \* k - 1) / (2 \* nodes\_cnt))  
 **for** k **in** range(1, nodes\_cnt + 1)], dtype = float)  
 nodes\_y = np.asarray([(np.exp(-x) + x\*\*2) **for** x **in** nodes\_x], dtype = float)  
 x = np.linspace(np.min(nodes\_x), np.max(nodes\_x), 100)  
 y = [lagrange(nodes\_x, nodes\_y, x\_val) **for** x\_val **in** x]  
 **return** x, y  
  
  
*# построение графика***def** draw\_plot(x, y, n):  
 fig = plt.figure()  
 ax = plt.subplot(111)  
 ax.plot(func\_x, func\_y, **'r-'**, label = **'f(x)'**)  
 ax.plot(x, y, **'b--'**, label = **'polynom'**)  
 plt.grid(**True**)  
 plt.title(**'n = {}'**.format(n))  
 ax.legend()  
 show()  
  
  
num\_segments = [10, 20, 40]  
  
**for** n **in** num\_segments:  
 *# разделяем отрезок на n частей* segments = np.append(np.arange(0.0, 1.0, 1.0 / n), 1.0)  
 x = []; y = []  
  
 *# на каждом подотрезке интерполируем функцию (по 4 узлам)* **for** i **in** range(1, size(segments)):  
 tX, tY = interpolate(segments[i], segments[i - 1], 4)  
 x.extend(tX); y.extend(tY)  
  
 *# строим график* draw\_plot(x, y, n)

# Графики:





# Выводы:

При приближении функции на каждом подотрезке полиномами одинаковой степени суммарная погрешность интерполяции на всем отрезке тем меньше, чем на большее число подотрезков разбивается исходный промежуток. Однако в данной задаче степень интерполяционного полинома достаточно велика, чтобы визуальных отличий при разных n на графиках не было.