Выполнил Бабенко Роман

Код для полинома лагранжа был выполнен и показан прямо на паре, поэтому в этом отчёте не будет его объяснения. Но его и все другие программы можно найти в приложении к отчёту.
также всё выложено на github: https://github.com/skrabik/hw/
Задача 1.

Нужно найти оченку погрешности:

$$|f(x) - L_n(x)| \leq \frac{M_n |w_n(x)|}{n!}$$

Найдём сначала значение M_n. С помощью sympy найдём четвёртую производную от $\frac{1}{x}$ и M_n с помошью такого кода:

```
x = sympy.Symbol('x')
f = 1/x
fourth_derivative = sympy.diff(f, x, 4)

def four_f(x):
    return 24 / (x**5)

Mn = max([four_f(x) for x in Xxx])
```

Далее функцию w_n и функцию выкториала

$$w_n(x) = \prod_{j=1}^{n}(x - x_j)$$

```
def wk (x):
    res = 1
    for j in range(4):
        res *= (x - Xi[j])
    return res

def fact(n):
    if n == 1 or n == 0:
        return 1
    return fact(n-1)*n

res = list()
```

Теперь проверим утверждение. Пройдёмся по всем x, и запишем значение выражения в каждой точке.

```
for x in Xxx:
    expression = ((abs(f(x) - float(lag_pol(x)))) <= (Mn
    * abs(wk(x) / fact(4))))
    if not expression:
        print(x)
    res.append(expression)
print(res.count(True), res.count(False))
```

Далее смотрим количестов значаний "True" и "False": 800 0. Наше выражение верно! Аналогичный результат получаем для многочленов лагранжа, посроенных на 3-х и 2-х точках. (Весть подробный код есть в ipyng файле) Производные для вычисления M_n получатся:

```
def three_f(x):
    return -6 / (x**4)

def two_f(x):
    return 2 / (x**3)
```

Задача 2

Необходимо найти оценку погрешности с помощью разделённых разностей.

$$f(x) - L_n(x) = f(x_1; x_2; ...; x_n)w_n(x)$$

Для вычисления w_n мы воспользуемся уже написанной функцией из прошлой задачи. Для фукции вычислления раздёлённой разности я написал такую рекурсивную функцию:

```
def f_razd (xvalues = list()):
    if len(xvalues) == 2:
        return ((f(xvalues[1]) - f(xvalues[0]) ) / (xvalues[1] -
xvalues[0]))
    return f_razd(xvalues[1:]) - f_razd(xvalues[:-1]) /
     (xvalues[-1] / xvalues[0])
```

Соответственно также пробегаемся по всем x, и проверяем значение выражения:

```
res = list()
for x in Xxx:
    expression = ((f(x) - lag_pol(x)) == (f_razd(Xi)*wn(x)))
    res.append(expression)

print(res.count(True), res.count(False))
```

Мы получили ответ: 800 0!

Домашнее заданте 3

1. Найдём корни многочлена чебышева 4 степени

```
from numpy import cos, pi
n = 4
a = 0.1
b = 1.3
racines = list()
for m in range(0, 4):
    Xm = float((b+a) / 2 + ((b-a)/2)*cos(((pi*(2*m+1))/(2*n))))
    racines.append(Xm)
racines.sort()
```

2. Построим многочлен Лагранжа по этим точкам

```
Xi = racines
Yi = [f(i) for i in Xi]

lag_pol = create_polinom(Xi, Yi)
Xxx = np.linspace(0.1, 1.3, 500)
Yyy = []
for x in Xxx:
Yyy.append(lag_pol(x))
```

3. Теперь проведём оценку погрешности

$$max|f(x) - L_n(x)| \leqslant \frac{M_n}{n!}(b-a)^n 2^{1-2n}$$

```
left_part = max([abs(f(x)-lag_pol(x)) for x in Xxx])

x = sympy.Symbol('x')
f = 1/x
derivative = sympy.diff(f, x, 4)
print(derivative)

def der(x):
    return 24/x**5

Mn = max([der(x) for x in Xxx])

def fact(n):
    if n == 1 or n == 0:
        return 1
    return fact(n-1) * n

rigth_part = Mn *  (b-a)**n * 2**(1-2*n)

print(left_part <= rigth_part)
```

Результат получаем: TRUE.

ps. подробные программы можно найти в прилагаемом файле lagrange.ipynb