Листинг кода (исх. стр. 5-7)

```
# Таблица значений функции
table = pd.DataFrame({"x_i": x_vals, "f(x_i)": f_vals})
table_transposed = table.T

def compute_newton_coefficients(x_vals, y_vals):
    """
    Возвращает список коэффициентов интерполяционного многочлена Ньютона
    с использованием рекурсивного определения разделённых разностей.
    """
    n = len(x_vals)
    # Создаём таблицу размером n x n
    dd_table = [y_vals.copy()]  # f[x_i]

    for level in range(1, n):
        prev_column = dd_table[-1]
        curr_column = []
        for i in range(len(prev_column)-1):
            numerator = prev_column[i + 1] - prev_column[i]
            denominator = x_vals[i + level] - x_vals[i]
            curr_column.append(numerator / denominator)
        dd_table.append(curr_column)
    # Коэффициенты Ньютона — это верхние элементы каждого столбца
    return dd_table, [dd_table[i][0] for i in range(n)]



def extend_divided_difference(dd_table, x_vals, x_star, f_star):
    """
    Расширяет таблицу разделённых разностей на одну точку x_star, f_star
    и возвращает f[x0, ..., xn, x*] (верхний элемент новой диагонали).
    """
#    n = len(x_vals)
    column = [f_star.copy()]
    for k in range(len(x_vals)):
        numerator = column[-1] - dd_table[k][-1]
        denominator = x_star - x_vals[k]
        column.append(numerator / denominator)
    return column[-1]




dd_table, newton_coeffs = compute_newton_coefficients(x_vals, f_vals)
def newton_interpolation(x_vals, y_vals, x, coef):
    """
    Вычисляет значение интерполяционного многочлена Ньютона в точке x
    с использованием рекурсивной формулы:
    P_{n+1}(x) = P_n(x) + alpha_{n+1} * omega_{n+1}(x)
    """
```

```
    result = coef[0]
    omega = 1.0
    for i in range(1, len(coef)):
        omega *= (x - x_vals[i - 1])
        result += coef[i] * omega
    return result

def omega(x_vals, x_point):
    res = 1
    for x in x_vals:
        res *= (x_point - x)
    return res
omegas = [omega(x_vals, x_point) for x_point in x_star]

P_x_star = [newton_interpolation(x_vals, f_vals, x_st, newton_coeffs) for x_st in x_star]

omega_frame = pd.DataFrame(omegas, index=[f'omega_{i}' for i in range(len(omegas))])
# Результаты интерполяции
data = {
    "Точка": ["x*", "x**", "x***"],
    "Значение x": x_star,
    "f(x)": f_star,
    "P(x) (полином)": P_x_star
}

n = len(newton_coeffs)
dd_frame = pd.DataFrame(dd_table).T
dd_frame.columns = [f"f[x0..x{i}]"
                    for i, n in zip(range(len(dd_table)), reversed(range(len(dd_table))))]

dd_frame.insert(0, "x_i", x_vals)
coeff_frame = pd.DataFrame(newton_coeffs, index=[f"a_{i}" for i in range(n)]).T
df = pd.DataFrame(data)

# Истинная погрешность
r_x_stars = np.abs(f_star - P_x_star)

error_bound_stars = []

for i in range(len(x_star)):
    error_bound = abs(extend_divided_difference(dd_table, x_vals, x_star[i], f_star[i]) * omegas[i])
    error_bound_stars.append(error_bound)

# Проверка выполнения неравенства
is_error_bound_stars_valid = [
    abs(r_x_stars[i]) <= error_bound_stars[i] for i in range(3)
]

# Таблица ошибок
error_table = pd.DataFrame({
    "Точка": ["x*", "x**", "x***"],
```

```
    "Значение x": x_star,
    "r истинная": r_x_stars,
    "оценка погрешности": error_bound_stars,
    "Неравенство выполняется?": is_error_bound_stars_valid
})

# Вывод таблиц
display(table_transposed)
display(df)
display(error_table)
display(dd_frame)
display(coeff_frame)
display(omega_frame)
```

Результаты (исх. стр. 7)

	Точка	Значение x	r истинная	оценка погрешности	Неравенство выполняется?
0	x*	0.766667	1.021405e-13	2.652478	True
1	x**	1.250000	3.108624e-15	3.836909	True
2	x***	1.666667	2.433609e-13	5.375227	True

Интерполирование многочленом Ньютона на Чебышёвской сетке (исх. стр. 9)

Пусть теперь x_i заданы на отрезке [0.7,1.7] следующим образом :

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} cos\left(\frac{\pi(2i+1)}{2n+1}\right), \ где$$

$i = \overline{0,n}$,

$a = 0{,}7$,

$b = 1{,}7$,

$n = 10$, *т. е.*

$$x_i = 1.2 + \frac{1}{2} cos\left(\frac{\pi(2i+1)}{21}\right), \ i = \overline{0,\ 10},$$

Точки восстановления те же:

- $x^* = 0.766667$,
- $x^{**} = 1.25$,
- $x^{***} = 1.66667$

Остаток интерполирования в точках $x^*,\ x^{**}, x^{***}$ оценим по следующим формулам:

$$|r_n(x)| \le \frac{M}{(n+1)!}|\omega_{n+1}(x)| \le \frac{M}{(n+1)!}\cdot 2\left(\frac{b-a}{4}\right)^{n+1}, \quad (\text{v.}1),$$

$$|r_n(x)| \le |f[x_0,\ \dots,\ x_n,\ x]| \cdot |\omega_{n+1}(x)| \le$$

$$\le |f[x_0,\ \dots,\ x_n,\ x]| \cdot 2\left(\frac{b-a}{4}\right)^{n+1}, (\text{v.}2), \text{где}$$

$x \in \{x^*,\ x^{**}, x^{***}\}$

$M = \left\|f^{(n+1)}\right\|_{C[a,b]} = \max\limits_{a\le x\le b}\left|f^{(n+1)}(x)\right| = 3.870417,$

n = 10,

a = 0.7, b = 1.7.

Анализ (исх. стр. 13)

Нетрудно видеть, что для $x^*,\ x^{**}, x^{***}$ $r_{\text{ист}}$ не превосходит оценки сверху. Также можно заметить, что истинная погрешность в контрольных точках на чебышевской сетке улучшилась по сравнению с равномерной сеткой. Однако для x^{**} это не так, поскольку x^{**} находится в середине отрезка, где сетка Чебышева имеет более разреженные узлы. Поэтому для точек в середине отрезка погрешность на чебышевской сетке будет выше, чем на равномерной.

Результаты (исх. стр. 20)

	Точка	Значение x	f(x)	φ(x)
0	x*	0.766667	1.714927	1.714929
1	x**	1.250000	2.727935	2.727936
2	x***	1.666667	4.004765	4.004767

Анализ (исх стр 20)

В ходе интерполяции с применением метода наименьших квадратов была проведена оценка точности аппроксимации на ряде контрольных точек. Полученные результаты оказались лучше теоретических оценок, что указывает на эффективность выбранного подхода и точное воспроизведение исходной функции.

Замечено, что значения погрешности для всех тестовых точек имеют сопоставимый масштаб, что говорит о равномерном распределении ошибки по всему интервалу. Это подчеркивает не только точность, но и стабильность метода наименьших квадратов при аппроксимации. Такое поведение объясняется тем, что аппроксимирующей функции не обязательно проходить через все контрольные точки, достаточно лишь минимизировать среднеквадратичное отклонение.

Выводы

Среди всех рассмотренных способов приближения функций наилучший результат продемонстрировала интерполяция многочленом Ньютона (ошибка порядка e-14). Это объясняется тем, что многочлен Ньютона стремится пройти через все контрольные точки, увеличивая точность. Отдельно следует отметить влияние эффекта Рунге на интерполяцию многочленом Ньютона – именно оно является причиной увеличения ошибки при интерполяции на равномерной сетке (в нашем случае вплоть до порядка e-13).

Второе место – МНК (ошибка порядка e-06), т.к. не стремится пройти через все точки, а лишь уменьшает суммарную ошибку. Ошибка равномерно распределяется во всех точках и в каждой конкретной невысока.

Худший результат – у кубического сплайна (ошибка порядка e-04), объясняется тем, что сплайн локален по построению и к тому же ограничен 3 степенью.