**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

**Лабораторная работа №4**

по дисциплине: Вычислительная математика

тема: «Интерполяция и экстраполяция для таблично-заданных функций»

Выполнил: ст. группы ВТ-231

Масленников Даниил

Проверили:

Островский Алексей Мичеславович

Белгород 2025 г

Л а б о р а т о р н а я р а б о т а № 4

**Цель работы**: Изучить методы интерполяции и экстраполяции для таблично-заданных

функций, особенности их алгоритмизации.

**Задачи**:

1) Рассмотреть теоретические основы интерполяции и экстраполяции для таблично-

заданных функций.

2) Научиться выбирать методы и алгоритмизировать интерполяцию, а также экстраполяцию

для таблично-заданных функций в зависимости от численной ситуации с вниманием к

проблемам точности, численной стабильности и релевантности поставленной задачи.

3) Выполнить индивидуальное задание, закрепляющее на практике полученные знания и

практические навыки (номер задания соответствует номеру студента по журналу; если этот

номер больше, чем максимальное число заданий, тогда вариант задания вычисляется по

формуле: номер по журналу % максимальный номер задания, где % — остаток от деления).

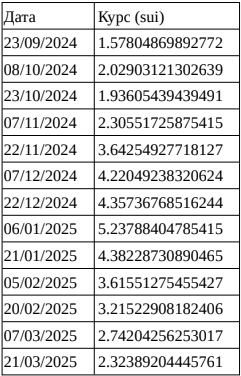
Алгоритм выполнения индивидуального задания находится в разделе «Ход выполнения

практической части лабораторной работы».

4) Отразить в отчете все полученные результаты, включая графики, тексты программ.

Сделать выводы.

**Вариант 14**



**Интерполяция:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from datetime import datetime, timedelta**

**from scipy.interpolate import interp1d**

**import pandas as pd**

**# Исходные данные (Sui)**

**data = {**

**'Дата': ["23/09/2024", "08/10/2024", "23/10/2024", "07/11/2024", "22/11/2024",**

**"07/12/2024", "22/12/2024", "06/01/2025", "21/01/2025", "05/02/2025",**

**"20/02/2025", "07/03/2025", "21/03/2025"],**

**'Курс': [1.57804869892772, 2.02903121302639, 1.93605439439491, 2.30551725875415,**

**3.64254927718127, 4.22049238320624, 4.35736768516244, 5.23788404785415,**

**4.38228730890465, 3.61551275455427, 3.21522908182406, 2.74204256253017,**

**2.32389204445761]**

**}**

**df = pd.DataFrame(data)**

**df['Дата'] = pd.to\_datetime(df['Дата'], format='%d/%m/%Y')**

**df = df.sort\_values('Дата').reset\_index(drop=True)**

**# Преобразование дат в дни**

**start\_date = df['Дата'].iloc[0]**

**df['Дни'] = (df['Дата'] - start\_date).dt.days**

**# Функции для интерполяции Ньютона**

**def calculate\_divided\_differences(x, y):**

**n = len(y)**

**coef = np.zeros([n, n])**

**coef[:,0] = y**

**for j in range(1, n):**

**for i in range(n - j):**

**coef[i][j] = (coef[i+1][j-1] - coef[i][j-1]) / (x[i+j] - x[i])**

**return coef[0, :]**

**def newton\_interpolate(x, xi, coefs):**

**result = coefs[-1]**

**for i in range(len(coefs)-2, -1, -1):**

**result = result \* (x - xi[i]) + coefs[i]**

**return result**

**# Добавление узлов Чебышёва**

**def add\_chebyshev\_nodes(x, y, num\_nodes=5):**

**a, b = min(x), max(x)**

**k = np.arange(1, num\_nodes+1)**

**cheb\_nodes = 0.5\*(a + b) + 0.5\*(b - a)\*np.cos((2\*k - 1)\*np.pi/(2\*num\_nodes))**

**interp = interp1d(x, y, kind='cubic', fill\_value="extrapolate")**

**cheb\_values = interp(cheb\_nodes)**

**new\_x = np.sort(np.concatenate([x, cheb\_nodes]))**

**new\_y = interp(new\_x)**

**return new\_x, new\_y**

**# Расчет коэффициентов**

**x = df['Дни'].values**

**y = df['Курс'].values**

**coefs = calculate\_divided\_differences(x, y)**

**# Добавление узлов Чебышёва и пересчет**

**new\_x, new\_y = add\_chebyshev\_nodes(x, y)**

**new\_coefs = calculate\_divided\_differences(new\_x, new\_y)**

**# Интерполяция для тестовой даты (21/03/2025)**

**test\_date = datetime.strptime("21/03/2025", "%d/%m/%Y")**

**test\_day = (test\_date - start\_date).days**

**interp\_value = newton\_interpolate(test\_day, x, coefs)**

**actual\_value = df['Курс'].iloc[-1]**

**error = abs(interp\_value - actual\_value)**

**# Визуализация**

**plt.figure(figsize=(12, 6))**

**plt.plot(df['Дата'], df['Курс'], 'bo', label='Исходные данные')**

**x\_vals = np.linspace(min(x), max(x), 365)**

**y\_vals = [newton\_interpolate(day, x, coefs) for day in x\_vals]**

**dates = [start\_date + timedelta(days=int(day)) for day in x\_vals]**

**plt.plot(dates, y\_vals, 'r--', label='Полином Ньютона')**

**y\_vals\_cheb = [newton\_interpolate(day, new\_x, new\_coefs) for day in x\_vals]**

**plt.plot(dates, y\_vals\_cheb, 'g-', label='С узлами Чебышёва')**

**plt.plot(test\_date, interp\_value, 'k\*', markersize=10, label=f'Ошибка: {error:.2f}')**

**plt.title('Интерполяция курса Sui')**

**plt.xlabel('Дата')**

**plt.ylabel('Курс')**

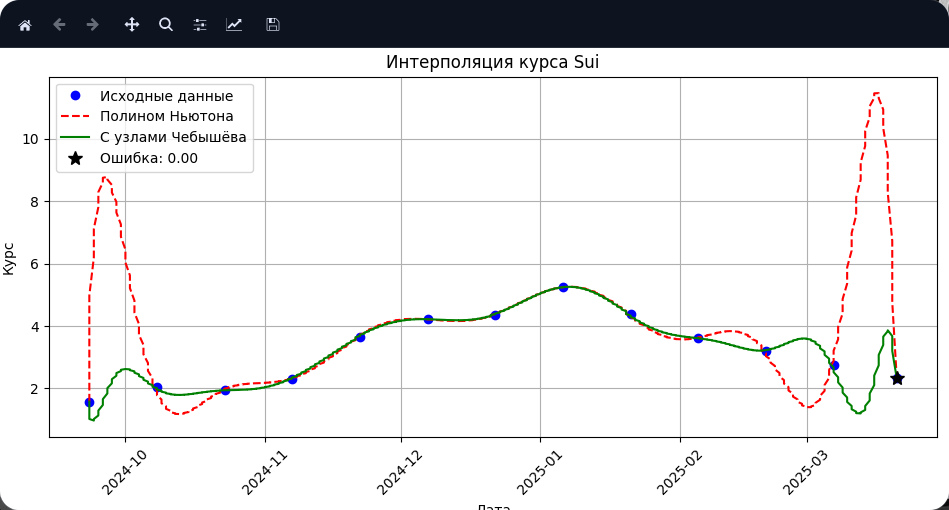
**plt.legend()**

**plt.grid(True)**

**plt.xticks(rotation=45)**

**plt.tight\_layout()**

**plt.show()**

**Вывод программы:**

**Выполнив интерполяцию, можно сделать вывод, что узлы Чебышёва используются в интерполяции для минимизации максимальной ошибки аппроксимации на заданном интервалею.**

**Экстраполяция:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from datetime import datetime**

**import pandas as pd**

**from prophet import Prophet**

**from statsmodels.tsa.arima.model import ARIMA**

**from statsmodels.tsa.holtwinters import ExponentialSmoothing**

**# Исходные данные**

**data = {**

**'Дата': ["23/09/2024", "08/10/2024", "23/10/2024", "07/11/2024", "22/11/2024",**

**"07/12/2024", "22/12/2024", "06/01/2025", "21/01/2025", "05/02/2025",**

**"20/02/2025", "07/03/2025", "21/03/2025"],**

**'Курс': [1.57804869892772, 2.02903121302639, 1.93605439439491, 2.30551725875415,**

**3.64254927718127, 4.22049238320624, 4.35736768516244, 5.23788404785415,**

**4.38228730890465, 3.61551275455427, 3.21522908182406, 2.74204256253017,**

**2.32389204445761]**

**}**

**df = pd.DataFrame(data)**

**df['Дата'] = pd.to\_datetime(df['Дата'], format='%d/%m/%Y')**

**df = df.sort\_values('Дата').reset\_index(drop=True)**

**# Целевая дата и реальное значение**

**test\_date = datetime(2025, 4, 1)**

**actual\_value = 2.38**

**# Прогнозирование с помощью Prophet**

**prophet\_df = df.rename(columns={'Дата': 'ds', 'Курс': 'y'})**

**model\_prophet = Prophet()**

**model\_prophet.fit(prophet\_df)**

**future = model\_prophet.make\_future\_dataframe(periods=30)**

**forecast\_prophet = model\_prophet.predict(future)**

**# Прогнозирование с помощью ARIMA**

**model\_arima = ARIMA(df['Курс'], order=(1,1,1)).fit()**

**forecast\_arima = model\_arima.forecast(steps=30)**

**# Прогнозирование с помощью ETS**

**model\_ets = ExponentialSmoothing(df['Курс']).fit()**

**forecast\_ets = model\_ets.forecast(30)**

**# Сбор прогнозов**

**prophet\_pred = forecast\_prophet[forecast\_prophet['ds'] == test\_date]['yhat'].values[0]**

**arima\_pred = forecast\_arima.iloc[-1]**

**ets\_pred = forecast\_ets.iloc[-1]**

**# Сравнение результатов**

**results = pd.DataFrame({**

**'Метод': ['Prophet', 'ARIMA', 'ETS'],**

**'Прогноз': [prophet\_pred, arima\_pred, ets\_pred],**

**'Ошибка': [abs(p - actual\_value) for p in [prophet\_pred, arima\_pred, ets\_pred]]**

**})**

**# Визуализация**

**plt.figure(figsize=(12, 6))**

**plt.plot(df['Дата'], df['Курс'], 'ko-', label='Исторические данные')**

**plt.axvline(test\_date, color='r', linestyle='--', label='Целевая дата')**

**# Получаем последнюю историческую точку**

**last\_date = df['Дата'].iloc[-1]**

**last\_value = df['Курс'].iloc[-1]**

**# Цвета для методов**

**colors = ['blue', 'green', 'purple']**

**# Рисуем линии прогнозов и точки**

**for i, (method, pred) in enumerate(zip(results['Метод'], results['Прогноз'])):**

**# Линия от последней точки к прогнозу (теперь сплошная)**

**plt.plot([last\_date, test\_date], [last\_value, pred],**

**'-', color=colors[i], alpha=0.7, linewidth=1)**

**# Точка прогноза**

**plt.plot(test\_date, pred, 'o', markersize=10, color=colors[i],**

**label=f'{method}: {abs(pred-actual\_value):.2f}')**

**# Линия и точка реального значения (теперь сплошная)**

**plt.plot([last\_date, test\_date], [last\_value, actual\_value],**

**'k-', alpha=0.7, linewidth=1)**

**plt.plot(test\_date, actual\_value, 's', color='black', markersize=10,**

**label=f'Реальное значение: {actual\_value}')**

**plt.title(f'Сравнение методов экстраполяции')**

**plt.xlabel('Дата')**

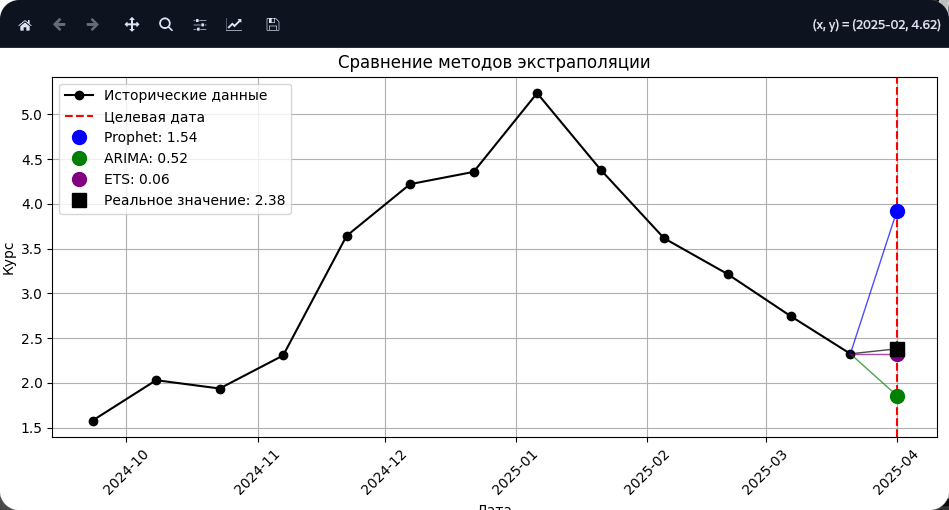
**plt.ylabel('Курс')**

**plt.legend()**

**plt.grid(True)**

**plt.xticks(rotation=45)**

**plt.tight\_layout()**

**plt.show()  
  
Вывод программы:** **Выполнив экстраполяцию, можно сделать вывод, что наиболее точным оказался метод ETS.  
  
  
Вывод:** Изучили методы интерполяции и экстраполяции для таблично-заданных функций, особенности их алгоритмизации.