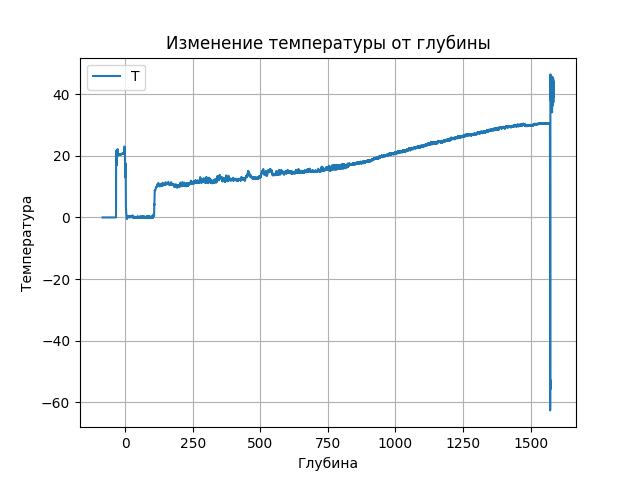
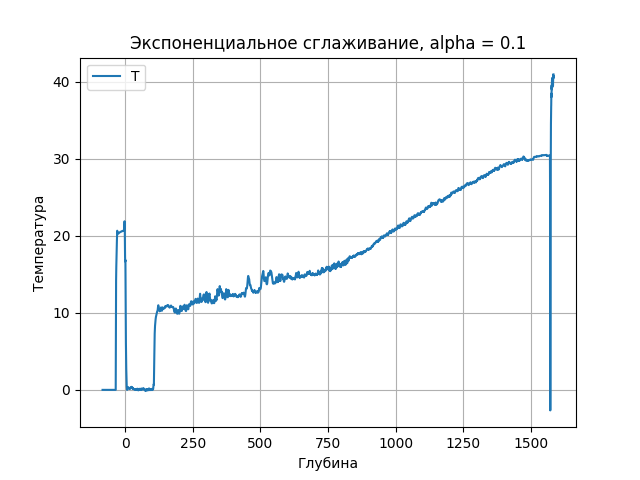
**Задачи:** необходимо построить график изменения T (температуры) от z (глубины) и разработать алгоритм уменьшения шумов в данных.

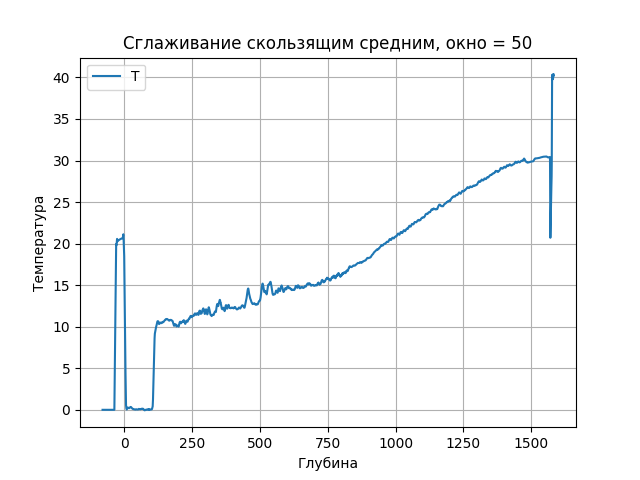
Извлечённые данные передаются в класс DataFrame из Pandas, который использует алгоритмы экспоненциального сглаживания и сглаживания скользящим средним, результаты вносятся в словарь. (Для примера введены alpha = 0.1 и окно = 50.) После в цикле выводятся графики для исходных данных и сглаживаний используя инструментарий matplotlib.

Ниже представлена функция, исполняющая всю логику и отрисованные графики.

def draw\_plots(  
 z: list[float],  
 t: list[float],  
 alpha: float,  
 window: int  
) -> None:  
 *"""  
 Обработка данных и отрисовка графиков  
  
 Parameters:  
 z (list[float]): массив данных глубины  
 t (list[float]): массив данных температуры  
 alpha (float): коэффициент альфа экспоненциального сглаживания  
 window (int): окно для сглаживания скользящим средним  
 """* df = pd.DataFrame(dict(z=z, T=t))  
 plots = {  
 'Изменение температуры от глубины': df,  
 f'Экспоненциальное сглаживание, alpha = {alpha}': df.ewm(alpha=alpha).mean(),  
 f'Сглаживание скользящим средним, окно = {window}': df.rolling(window=window).mean()  
 }  
 for title, data in plots.items():  
 data.plot(x='z', y='T', kind='line')  
 plt.title(title)  
 plt.xlabel('Глубина')  
 plt.ylabel('Температура')  
 plt.grid()  
 plt.show()





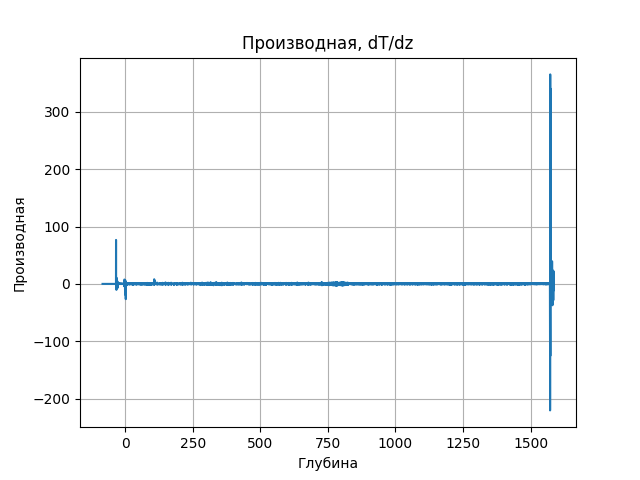


**Задача**: определение dT/dz на промежутках.

Производная для каждой точки предоставлена в документе. Для поиска производной в промежутке пишется функция, принимающая массив из производных, абсцисс (глубины) и границы промежутка.

Вначале проводится проверка на верность границ, после чего из массива глубины извлекается индекс, соответствующий каждой границе, если же границы была введена примерно, то он ищет ту, с которой наименьшая разность и берёт её в качестве границы. В конце выводит производную на промежутке и выводит график производной, отрисованный в matplotlib.

def get\_derivative(  
 x1: float,  
 x2: float,  
 z: list[float],  
 derivative: list[float]  
) -> None:  
 *"""  
 Расчёт среднего арифметического производной в промежутке [x1;x2]  
  
 Parameters:  
 x1 (float): начальная точка абсциссы  
 x2 (float): конечная точка абсциссы  
 z (list[float]): массив с данными глубины  
 derivative (list[float]): массив с данными производной  
 """* statements = [  
 x1 >= x2,  
 x1 < min(z),  
 x1 > max(z),  
 x2 < min(z),  
 x2 > max(z),  
 ]  
 if statements.count(True) > 0:  
 raise ValueError('Неверно заданные x1 или x2')  
 if x1 in z:  
 index\_x1 = z.index(x1)  
 else:  
 temp = [abs(x-x1) for x in z]  
 index\_x1 = temp.index(min(temp))  
 if x2 in z:  
 index\_x2 = z.index(x2)  
 else:  
 temp = [abs(x-x2) for x in z]  
 index\_x2 = temp.index(min(temp))  
 drv = sum(derivative[index\_x1:index\_x2]) / len(derivative[index\_x1:index\_x2])  
 print(f'Производная на промежутке [{x1};{x2}]: {drv}')



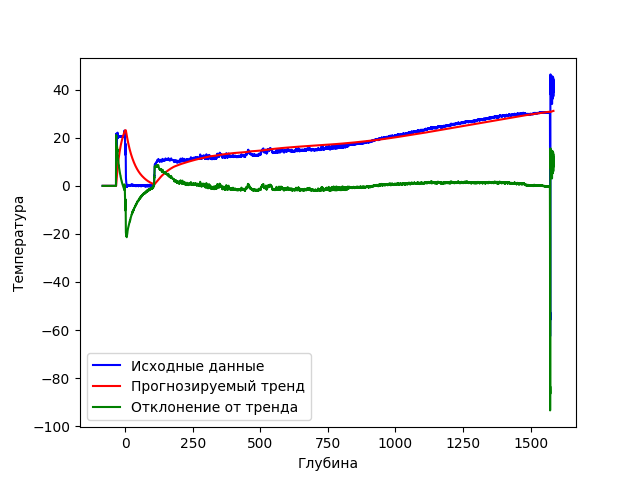
**Задачи**: отклонение данных от тренда и определение начала и конца промежутка отклонения данных и амплитуды отклонения.

Используя модель линейной регрессии из scikit-learn ищется тренд для каждой точки, а затем в отдельный список заносится его разность с реальными данными. После отрисовываются три графика: исходных данных, тренда, отклонения от тренда. Функция возвращает массив с данными по отклоениями.

def predict\_trand(z: list[float], t: list[float]) -> list[float]:  
 *"""  
 Расчёт тренда, отклонения от него и отрисовка графиков  
  
 Parameters:  
 z (list[float]): массив данных глубины  
 t (list[float]): массив данных температуры  
  
 Returns:  
 list[float]: массив с данными отклонения от тренда  
 """* trand\_x, trand\_y, deflection = [], [], t[:9]  
 for i in range(9, len(z)-1):  
 x = np.array(z[:i]).reshape(-1, 1)  
 y = np.array(t[:i])  
 model = LinearRegression()  
 model.fit(x, y)  
 a = model.intercept\_  
 b = model.coef\_[0]  
 future\_x = np.array([z[i+1]]).reshape(-1, 1)  
 future\_y = model.predict(future\_x)  
 trand\_x.append(future\_x[0])  
 trand\_y.append(future\_y[0])  
 deflection.append(float(t[i+1]-future\_y[0]))  
 plt.plot(z, t, color='blue', label='Исходные данные')  
 plt.plot(trand\_x, trand\_y, color='red', label='Прогнозируемый тренд')  
 plt.plot(z[:-1], deflection, color='green', label='Отклонение от тренда')  
 plt.xlabel('Глубина')  
 plt.ylabel('Температура')  
 plt.legend()  
 plt.show()  
 return deflection

В другой функции на основе полученных данных происходит сбор всех идущих подряд точек, чьё отклонения от тренда больше 5. После объединяются все интервалы, которые слишком близко друг к другу. В конце интервалы выводятся в консоль после преобразования из номера точки в её координаты.

def get\_anomalies(z: list[float], deflection: list[float]) -> None:  
 *"""  
 Поиск промежутков из данных по отклонения  
  
 Parameters:  
 z (list[float]): массив данных глубины  
 deflection (list[float]): массив с отклонениями от тренда  
 """* trand = [1 if x >= 5 else 0 for x in deflection]  
 anomalies = []  
 start, stop = 0, 0  
 for p in range(len(trand)-1):  
 if trand[p] == 1 and trand[p-1] == 0:  
 start = p  
 elif trand[p] == 1 and (trand[p+1] == 0 or p == len(trand)-1):  
 stop = p  
 anomalies.append((start, stop))  
 filtered = []  
 for a in range(len(anomalies)-1):  
 start1, stop1 = anomalies[a]  
 start2, stop2 = anomalies[a+1]  
 if abs(stop1 - start2) < 11:  
 filtered.append((start1, stop2))  
 else:  
 filtered.append((start1, stop1))  
 anomalies, ban = set(), set()  
 for f in range(len(filtered)-1):  
 start1, stop1 = filtered[f]  
 start2, stop2 = filtered[f+1]  
 if stop1 in range(start2, stop2+1):  
 anomalies.add((start1, stop2))  
 ban.add((start2, stop2))  
 else:  
 anomalies.add((start1, stop1))  
 result = 'Промежутки отклонения данных: '  
 for x1, x2 in anomalies.difference(ban):  
 result += f'[{z[x1]};{z[x2]}], '  
 print(result[:-2])



Консоль в конце работы программы.

