

Альтернативная версия гайда по графикам в python.

Показаны:

- загрузка таблиц из .els, .ods форматов
- аппроксимация данных с погрешностями произвольной функцией (χ^2).
- отображение доверительной области аппроксимации (вкл/выкл переменной visApproxError)
- автоматическая генерация легенды графика. Параметры аппроксимации отображаются с погрешностью (с соблюдением правил округления)

Баг-репорты, последние обновления, и аппроксимация с учетом погрешности измерений и по оси y, и по оси x [сюда](#) и [тут](#). Там же теор обоснование построения доверительной области по матрице ковариаций и вообще ликбез по тому, как этой матрицей ковариаций пользоваться.

Импорты и объявление функций

```
import os
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.optimize import curve_fit

from datetime import datetime
def time_string():
    # return datetime.now().strftime("%d %B %Y %X")
    return datetime.now().strftime("%X")
impath = 'trash_img'      # путь к папке, в которой будут сохраняться
картинки
#switching latex fonts
plt.rcParams.update({
    "text.usetex" : True,
    "font.family" : "serif",
    "font.serif" : "Computer Modern",
    "text.latex.preamble" : r'\usepackage{amsmath} \
usepackage{mathtext} \usepackage[english, russian]{babel}'
})
plt.rcParams.update({
    "figure.figsize": (8, 6),
    'figure.dpi': 200
})

# необходимые функции

def err_pres_formatter(value, error):
```

```

"""Rounds value and it's error according to scientific
conventions"""
    first_digit = (np.array(error/10**np.floor(np.log10(error)),
dtype="int"))
    first_digit_position = np.array(np.floor(np.log10(error)),
dtype="int")
    ndigits = 1
    if first_digit == 1:
        ndigits += 1
    power = np.log10(abs(value))
    if abs(power//3) >= 1:
        power = 3*int(power//3)
        value /= 10**power
        error /= 10**power
        digit_number = -first_digit_position + ndigits+power-1
        if digit_number <= 0:
            return f"({round(int(value), digit_number)} \u00B1
{round(int(error), digit_number)})\cdot 10^{{{power}}}"
            return f"({round(value, digit_number)} \u00B1 {round(error,
digit_number)})\cdot 10^{{{power}}}"
        digit_number = -first_digit_position + ndigits-1
        if digit_number <= 0:
            return f"{round(int(value), digit_number)} \u00B1
{round(int(error), digit_number)}"
            return f"{round(value, digit_number)} \u00B1 {round(error,
digit_number)}"

```

```

def lower_upper_limits(domain, fun, popt, pcov, nsigma=1):
    '''Вычисляет минимальные и максимальные значения функции
при заданных параметрах и стандартных отклонениях
в пределах nsigma*стандартное отклонение
Возвращает:
minA :(np.array) - массив минимальных значений
maxA :(np.array) - массив максимальных значений'''
    err = np.sqrt(np.diag(pcov))
    corr = pcov /np.expand_dims(err, 1) / np.expand_dims(err, 0)
    deviations = np.eye(popt.size)*np.expand_dims(err, 1)
    df = np.expand_dims(fun(domain, *popt), 1) -
fun(np.expand_dims(domain, 1), *(np.expand_dims(popt,
1)+np.eye(popt.size)*np.expand_dims(err, 1)))
    dfmatrixes = np.expand_dims(df, 1)*np.expand_dims(df, 2) *
np.expand_dims(corr, 0)
    sigm_f = np.sqrt(np.sum(dfmatrixes, axis=(1, 2)))
    return (fun(domain, *popt)-nsigma*sigm_f, fun(domain, *popt)
+nsigma*sigm_f)

```

```

def plot_approx(x, y, f, par_names_str, fun_name, yerr=None,
visApproxError=True, p0=None):
    if yerr is not None:

```

```

        popt, pcov = curve_fit(f, x, y, sigma=yerr,
absolute_sigma=True, p0=p0)
    else:
        popt, pcov = curve_fit(f, x, y, p0=p0)
        err = np.sqrt(np.diag(pcov))
        for v, er in zip(popt, err):
            print(v, '+-', er)
        approx_label = f"аппроксимация функцией {fun_name}"
        if (err == np.inf).any():
            print('Аппроксимация не сошлась. Попробуй задать начальные
параметры p0')
        else:
            par_names = [x.split(',') for x in
par_names_str.strip().split("\n") if (len(x) != 0)]
            for i in range(len(par_names)):
                t = par_names[i]
                if len(t) == 2:
                    name, units = t
                else:
                    name, units = *t, ''
                approx_label += f"\n{name} = {err_pres_formatter(popt[i],
err[i])} {units}"
            teorPoints = 50
            indent = 0.05 # отступ по бокам от области экспериментальных
данных
            a, b = x.min(), x.max()
            X = np.linspace(a-(b-a)*indent, b+(b-a)*indent, num=teorPoints)
            plt.plot(X, f(X, *popt), 'r', label=approx_label)
            if visApproxError and (pcov != np.inf).all():
                nsigma = 2
                minA, maxA = lower_upper_limits(X, f, popt, pcov,
nsigma=nsigma)
                plt.fill_between(X, minA, maxA, alpha=0.4, label=f"область
{nsigma} $\sigma$ ошибки")

def make_beautiful_plot(xname, xunit, yname, yunit):
    plt.minorticks_on()
    plt.grid(True, which='minor', linewidth=0.2)
    plt.grid(True, which='major')
    plt.xlabel(fr"{xname}, {xunit}")
    plt.ylabel(fr"{yname}, {yunit}")
    plt.title(fr"{yname}({xname})")
    plt.legend()

```

Загрузка данных

#Ввод данных

```

df = pd.read_excel("measurements.ods", engine="odf", sheet_name=0) #
для таблиц ods (требуется установка pip install odfpy)
# df = pd.read_excel("measurements.xls") # excel таблицы (требуется
установка pip install xlrd)

```

```

# df = pd.read_excel("measurements.xlsx")      # excel таблицы
(требуется установка pip install openpyxl)
# вторым индексом нужно задать номер колонки с данными
x = df.iloc[:, 0].dropna()
y = df.iloc[:, 1].dropna()
# если погрешности по одной из осей не измерены, можно поставить
значение None
xerr = np.ones_like(x)*4; None;
yerr = np.ones_like(x)*0.05
df.head()

```

	Q, мВт	R, Ω	калориметр	R, ом	t, с	первое тело	R,
0	0.000249	16.735029	NaN	10.0	1.0	NaN	
1	4.446031	16.812200	NaN	11.0	3.0	NaN	
2	9.670230	16.853135	NaN	12.0	4.0	NaN	
3	19.561403	16.926607	NaN	13.0	5.0	NaN	
4	32.830711	17.030702	NaN	14.0	6.0	NaN	

	t, с.1	второе тело	R, ом.2	t, с.2	третье тело
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN

```

# параметры построения графика и аппроксимации прямой
# пользовательские параметры. Переменные x, y, xerr, yerr должны
содержать данные
''' также необходимо изменить
название графика,
название функции ,
набор параметров с размерностями через запятую '''
visApproxError = True      # рисовать ли доверительную область
аппроксимации
xname, xunit = r"$x$", ""
yname, yunit = r"$y$", ""
# по умолчанию автоматически посчитаются размерности коэффициентов
линейной аппроксимации
par_names_str = fr'''
k,  $\frac{{{yunit.replace('$', '')}}}{{{{xunit.replace('$', '')}}}}$ 
b, {yunit}
'''
# function to approximate
fun_name = r"$f(x) = kx + b$"

```

```

def f(x, k, b):
    return b + k * x
# конец пользовательских параметров

# построение графика

# аппроксимация
plot_approx(x, y, f, par_names_str, fun_name, yerr=yerr,
visApproxError=visApproxError)

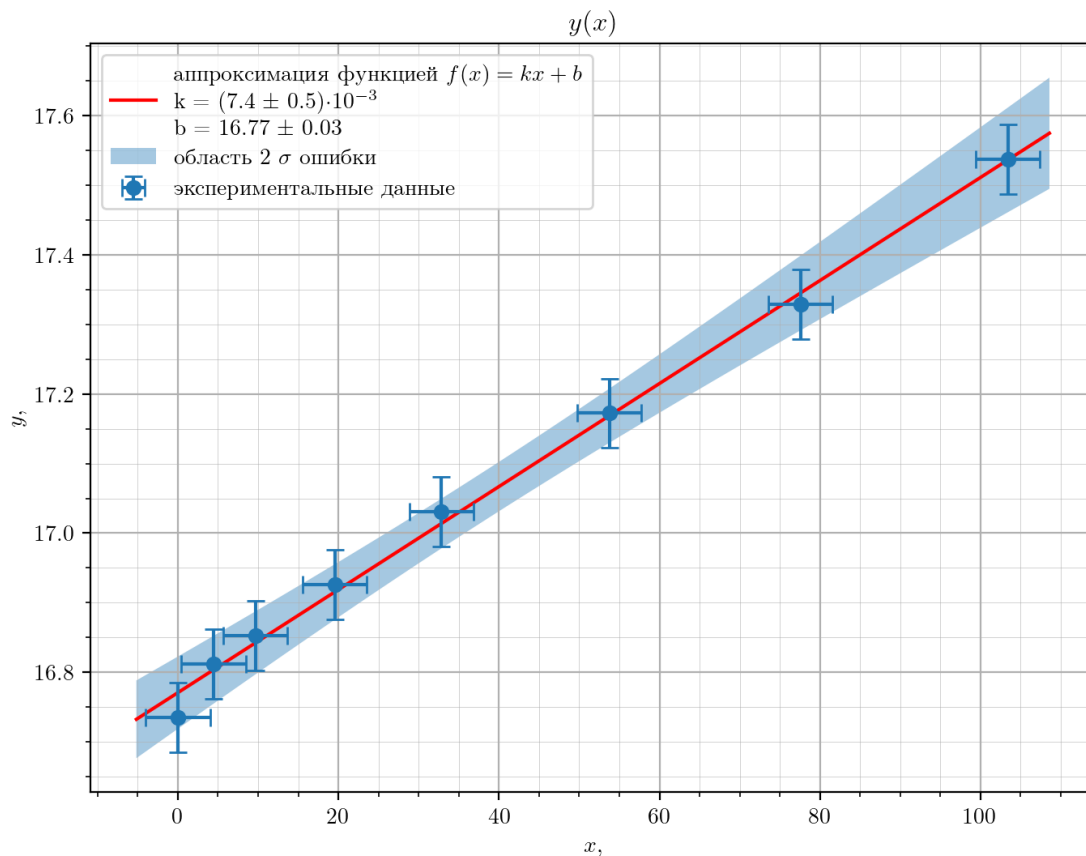
# экспериментальные точки
plt.errorbar(x, y, fmt="o", xerr=xerr, yerr=yerr,
label="экспериментальные данные", capsize=4, ms=6)

make_beautiful_plot(xname, xunit, yname, yunit)

if not os.path.exists(impath):
    os.makedirs(impath)
plt.savefig(os.path.join(impath, f"{time_string()}.jpg"), dpi=300)

0.007399683890071949 +- 0.0005043000684851718
16.77088871247657 +- 0.02594407282353369

```



```

# шаблон аппроксимации экспонентой
# пользовательские параметры. Переменные x, y, xerr, yerr должны
содержать данные
''' также необходимо изменить
название графика,
название функции,
набор параметров с размерностями через запятую '''
visApproxError = True # рисовать ли область ошибки аппроксимации
xname, xunit = r"$t$", "с"
yname, yunit = r"$h$", "см"
par_names_str = fr'''
$f_0$, {yunit}
$\Delta f$, {yunit}
$T_0$, {xunit}
'''

# function to approximate
fun_name = r"$f(x) = f_0 + \Delta f e^{-x/T_0}$"
def f(x, f_0, df, T0):
    return f_0 + df*np.exp(-x/T0)
# конец пользовательских параметров
p0 = [y[0], y.max()-y.min(), (x.max()-x.min())/8] # адекватное
начальное приближение параметров

# параметры построения графика и аппроксимации относительно
произвольной функцией
# пользовательские параметры. Переменные x, y, xerr, yerr должны
содержать данные
''' также необходимо изменить
название графика,
название функции ,
набор параметров с размерностями через запятую '''
visApproxError = True # рисовать ли область ошибки аппроксимации
par_names_str = r'''
k,
b,
'''
xname, xunit = r"$x$", ""
yname, yunit = r"$y$", ""
# function to approximate
fun_name = r"$f(x) = kx + b$"
def f(x, k, b):
    return b + k * x
# конец пользовательских параметров

```

Блоки для добавления в ячейку построения графика

```

# выбор поднабора для построения на графике и аппроксимации
filter = x < 15000
x = x[filter]
y = y[filter]

```

```
xerr = xerr[filter]
yerr = yerr[filter]

# опциональная загрузка данных разных измерений
N = 0 # номер измерения
stride = 3 # расстояние между колонками с данными
x = df.iloc[:, stride*N+1].dropna()
y = df.iloc[:, stride*N+0].dropna()
xerr = np.ones_like(x)*0.1
yerr = np.ones_like(x)*1

# загрузка имени серии из таблицы
label = df.iloc[:, stride*N+stride-2].name # если label есть, его
# надо добавить в label errorbar'a
# загрузка имени и размерности величин из названия колонки
xname, xunit = x.name.split(' ', 1)
yname, yunit = y.name.split(' ', 1)

plt.xscale('log') # устанавливает логарифмический масштаб по оси "x"
```