**Работа с библиотекой Matplotlib**

**Теоретическая часть**

**pyplot.errorbar**

Результаты измерений в физике чаще всего представлены в виде величин с ошибками. Функция [plt.errorbar](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.errorbar.html" \l "matplotlib.pyplot.errorbar" \t "_blank) позволяет отображать такие данные:

import numpy as np

import matplotlib.pyplot as plt

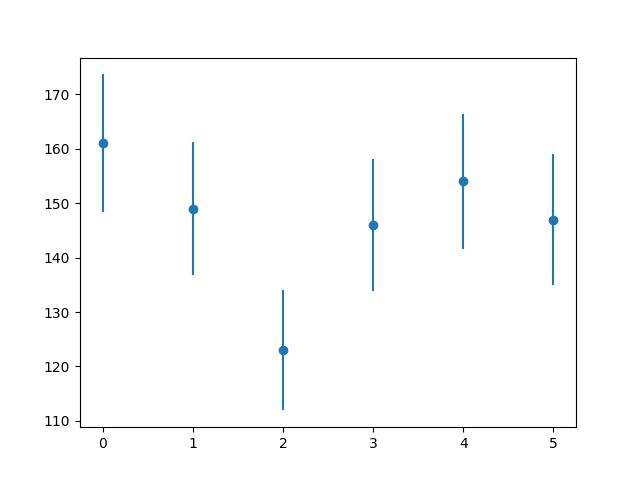
rg = np.random.Generator(np.random.PCG64(5))

x = np.arange(6)

y = rg.poisson(149, x.size)

plt.errorbar(x, y, yerr=np.sqrt(y), marker='o', linestyle='none')

plt.show()



Ошибки можно задавать и для значений по горизонтальной оси:

rg = np.random.Generator(np.random.PCG64(5))

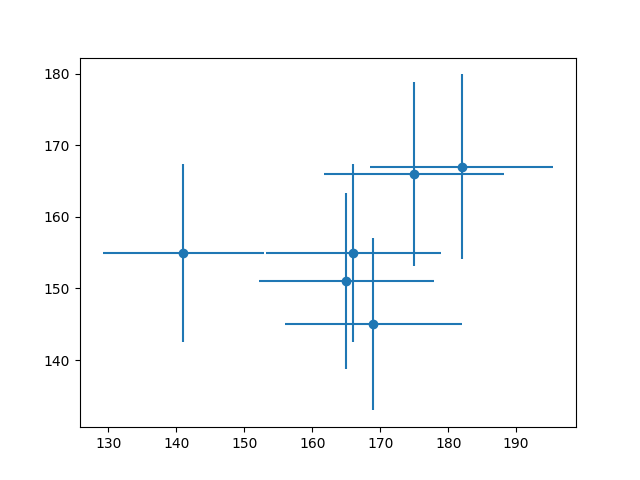
N = 6

x = rg.poisson(169, N)

y = rg.poisson(149, N)

plt.errorbar(x, y, xerr=np.sqrt(x), yerr=np.sqrt(y), marker='o', linestyle='none')

plt.show()



Ошибки измерений могут быть асимметричными. Для их отображения в качестве параметра yerr (или xerr) необходимо передать кортеж из двух списков:

rg = np.random.Generator(np.random.PCG64(11))

N = 6

x = np.arange(N)

y = rg.poisson(149, N)

yerr = [

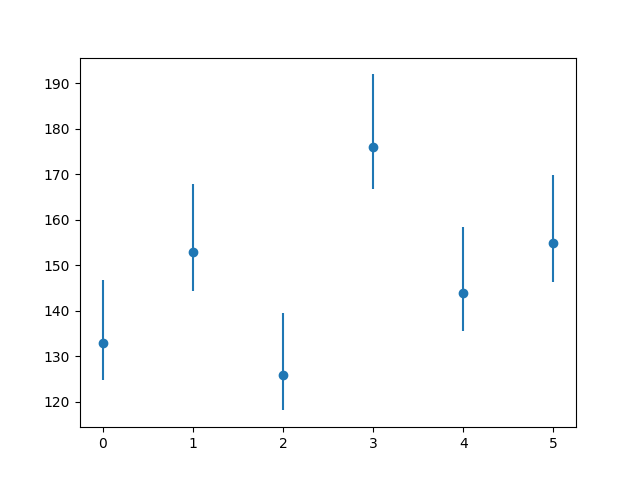
0.7\*np.sqrt(y),

1.2\*np.sqrt(y)

]

plt.errorbar(x, y, yerr=yerr, marker='o', linestyle='none')

plt.show()



Функция pyplot.errorbar поддерживает настройку отображения графика с помощью параметра fmt и всех именованных параметров, которые доступны в функции pyplot. Кроме того, здесь появляются параметры для настройки отображения линий ошибок ("усов"):

* ecolor: str — цвет линий ошибок
* elinewidth: float — ширина линий ошибок
* capsize: float — длина "колпачков" на концах линий ошибок
* capthick: float — толщина "колпачков" на концах линий ошибок

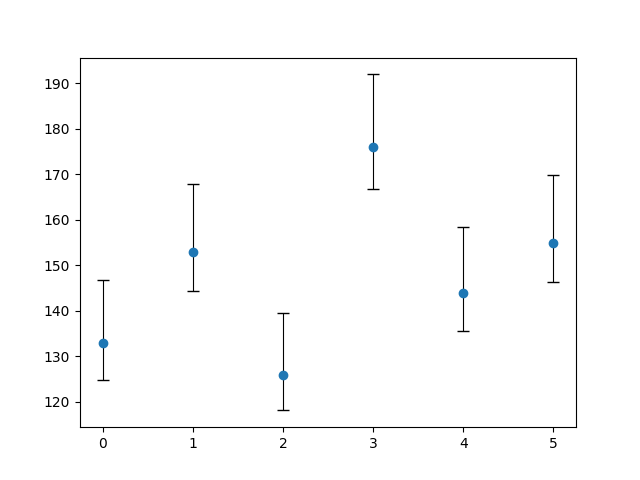
и некоторые другие. Изменим параметры отрисовки данных из предыдущего примера:

*# ...*

plt.errorbar(x, y, yerr=yerr, marker='o', linestyle='none',

ecolor='k', elinewidth=0.8, capsize=4, capthick=1)

plt.show()



**Настройки отображения**

**Диапазон значений осей**

Задавать диапазон значений осей в matplotlib можно несколькими способами. Например, так:

pyplot.xlim([0, 200]) *# диапазон горизонтальной оси от 0 до 200*

pyplot.xlim([0, 1]) *# диапазон вертикальной оси от 0 до 1*

**Размер шрифта**

Размер и другие свойства шрифта, который используется в matplotlib по умолчанию, можно изменить с помощью объекта matplotlib.rcParams:

matplotlib.rcParams.update({'font.size': 14})

Объект matplotlib.rcParams хранит множество настроек, изменяя которые, можно управлять поведением по умолчанию.

**Подписи осей**

Подписи к осям задаются следующим образом:

plt.xlabel('run number', fontsize=16)

plt.ylabel(r'average current ($\mu A$)', fontsize=16)

В подписях к осям (и вообще в любом тексте в matplotlib) можно использовать инструменты текстовой разметки TeX, позволяющие отрисовывать различные математические выражения. TeX-выражения должны быть внутри пары символов $, кроме того, их следует помещать в r-строки, чтобы избежать неправильной обработки.

**Заголовок**

Функция pyplot.title задает заголовок диаграммы. Применим наши новые знания:

import numpy as np

import matplotlib.pyplot as plt

import matplotlib

*# задаем размер шрифта*

matplotlib.rcParams.update({'font.size': 12})

rg = np.random.Generator(np.random.PCG64(11))

x = np.arange(6)

y = rg.poisson(149, x.size)

yerr = [

0.7\*np.sqrt(y),

1.2\*np.sqrt(y)

]

plt.errorbar(x, y, yerr=yerr, marker='o', linestyle='none',

ecolor='k', elinewidth=0.8, capsize=4, capthick=1)

*# добавляем подписи к осям и заголовок диаграммы*

plt.xlabel('run number', fontsize=16)

plt.ylabel(r'average current ($\mu A$)', fontsize=16)

plt.title(r'The $\alpha^\prime$ experiment. Season 2020-2021')

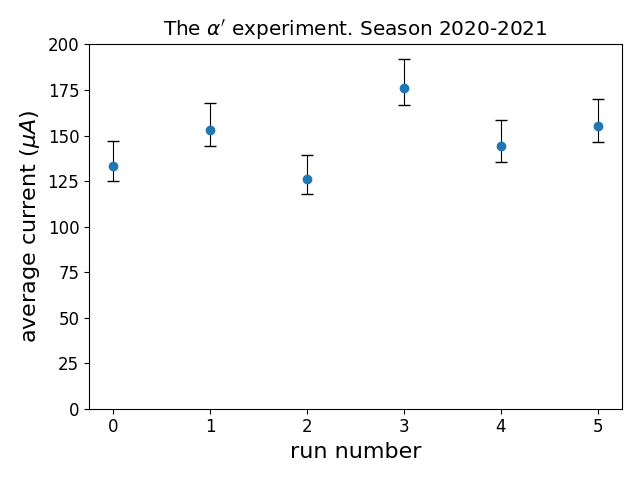
*# задаем диапазон значений оси y*

plt.ylim([0, 200])

*# оптимизируем поля и расположение объектов*

plt.tight\_layout()

plt.show()



В этом примере мы использовали функцию pyplot.tight\_layout, которая автоматически подбирает параметры отображения так, чтобы различные элементы не пересекались.

**Легенда**

При построении нескольких графиков в одних осях полезно добавлять легенду — пояснения к каждой линии. Следующий пример показывает, как это делается с помощью аргументов label и функции [pyplot.legend](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.legend.html" \t "_blank):

import numpy as np

import matplotlib.pyplot as plt

import matplotlib

matplotlib.rcParams.update({'font.size': 12})

x = np.linspace(0, 1, 100)

f1 = 0.25 - (x - 0.5)\*\*2

f2 = x\*\*3

*# указываем в аргументе label содержание легенды*

plt.plot(x, f1, ':b', label='1st component')

plt.plot(x, f2, '--r', label='2nd component')

plt.plot(x, f1+f2, 'k', label='total')

plt.xlabel(r'$x$', fontsize=16)

plt.ylabel(r'$f(x)$', fontsize=16)

plt.xlim([0, 1])

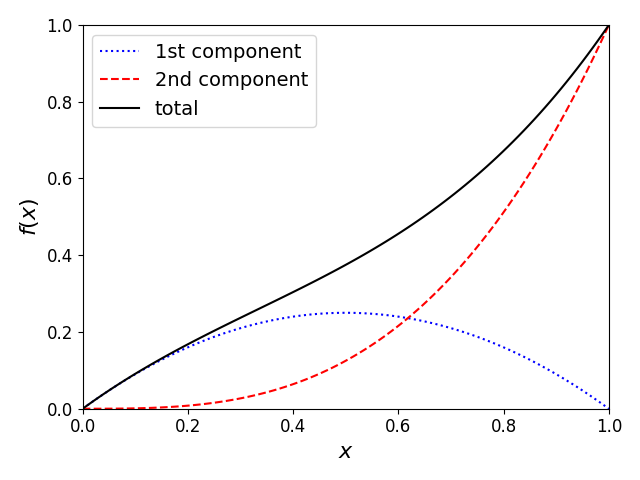
plt.ylim([0, 1])

*# выводим легенду*

plt.legend(fontsize=14)

plt.tight\_layout()

plt.show()



Функция pyplot.legend старается расположить легенду так, чтобы она не пересекала графики. Аргумент loc позволяет задать расположение легенды вручную. В большинстве случаев расположение по умолчанию получается удачным.

**Сетка**

Сетка во многих случаях облегчает анализ графиков. Включить отображение сетки можно с помощью функции pyplot.grid. Аргумент axis этой функции имеет три возможных значения: x, y и both и определяет оси, вдоль которых будут проведены линии сетки. Управлять свойствами линии сетки можно с помощью именованных аргументов, которые мы рассматривали выше при обсуждении функции pyplot.plot.

В matplotlib поддерживается два типа сеток: основная и дополнительная. Выбор типа сетки выполняется с помощью аргумента which, который может принимать три значения: major, minor и both. По умолчанию используется основная сетка.

Линии сетки привязаны к отметкам на осях. Чтобы работать с дополнительной сеткой необходимо сначала включить вспомогательные отметки на осях (которые по умолчанию отключены и к которым привязаны линии дополнительной сетки) с помощью функции pyplot.minorticks\_on. Приведем пример:

import numpy as np

import matplotlib.pyplot as plt

import matplotlib

matplotlib.rcParams.update({'font.size': 12})

x = np.linspace(-1, 1, 250)

plt.plot(x, x, label=r'$x$')

plt.plot(x, x\*\*2, label=r'$x^2$')

plt.plot(x, x\*\*3, label=r'$x^3$')

plt.plot(x, np.cbrt(x), label=r'$x^{1/3}$')

plt.legend(fontsize=16)

*# включаем дополнительные отметки на осях*

plt.minorticks\_on()

plt.xlabel(r'$x$', fontsize=16)

plt.xlim([-1., 1.])

plt.ylim([-1., 1.])

*# включаем основную сетку*

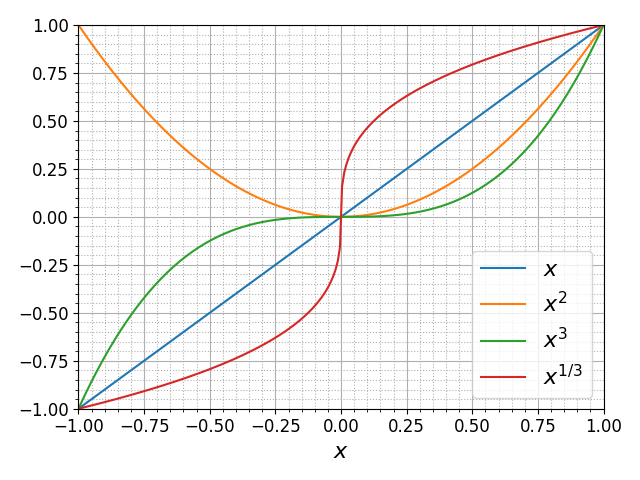
plt.grid(which='major')

*# включаем дополнительную сетку*

plt.grid(which='minor', linestyle=':')

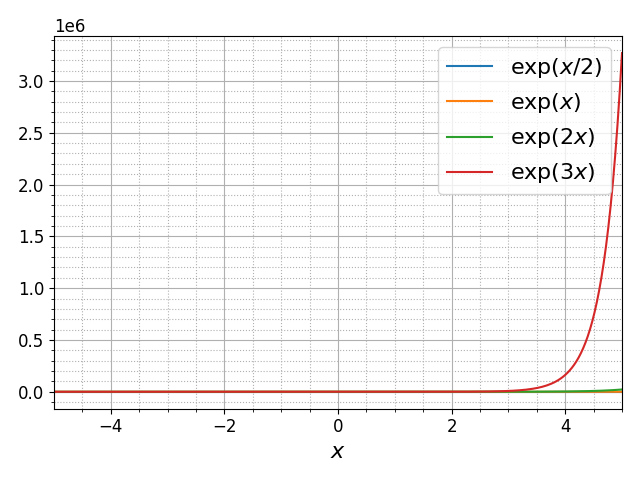
plt.tight\_layout()

plt.show()



**Логарифмический масштаб**

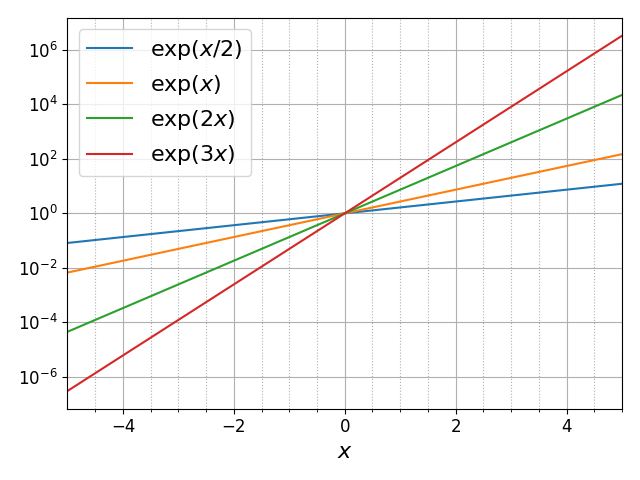
Функции pyplot.semilogy и pyplot.semilogx выполняют переключение между линейным и логарифмическим масштабами осей. В некоторых случаях логарифмический масштаб позволяет отобразить особенности зависимостей, которые не видны в линейном масштабе. Вот так выглядят графики экспоненциальных функций в линейном масштабе:



Добавление строки

plt.semilogy()

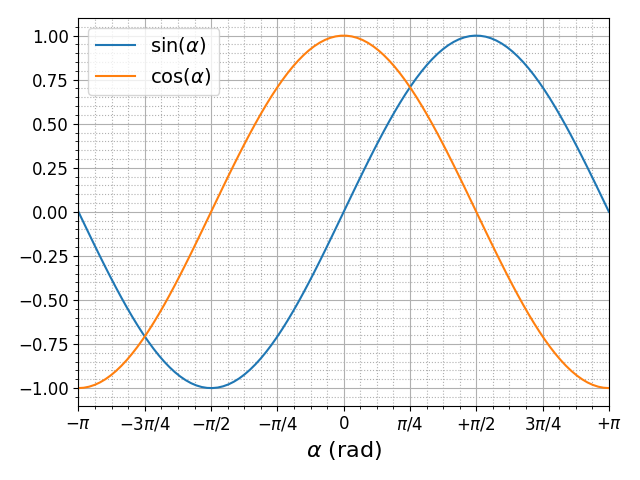
делает график гораздо более информативным:



Теперь мы видим поведение функций во всем динамическом диапазоне, занимающем 12 порядков.

**Произвольные отметки на осях**

Вернемся к первому примеру, в котором мы строили графики синуса и косинуса. Сделаем так, чтобы на горизонтальной оси отметки соответствовали различным долям числа pi и имели соответствующие подписи:



Метки на горизонтальной оси были заданы с помощью функции pyplot.xticks:

plt.xticks(

np.linspace(-np.pi, np.pi, 9),

[r'$-\pi$', r'$-3\pi/4$', r'$-\pi/2$', r'$-\pi/4$', r'$0$',

r'$\pi/4$', r'$+\pi/2$', r'$3\pi/4$', r'$+\pi$'])

Модуль pyplot.ticker содержит более продвинутые инструменты для управления отметками на осях.

**Задания на лабораторную работу**

1. Импортировать файл Experiment.csv, показывающий результаты, полученные с различных устройств в определенные даты.
2. Построить графики результатов эксперимента одновременно по трем датам (выбрать строки случайным образом). На графиках отобразить погрешности измерений:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Устройство 1 | Устройство 2 | Устройство 3 | Устройство 4 | Устройство 5 | Устройство 6 | Устройство 7 | Устройство 8 |
| -0,2; +0,2 | -0,12; +0,05 | -0,93; +0,4 | -0,03; +0,25 | -0,2; +0,2 | -0,1; +0,2 | -0,4; +0,2 | -0,6; +0,71 |

Графики построить с двумя различными стилями оформления (разной ширины, с «колпачками» и без, разного цвета и т.п.)

Задать подписи к осям на русском языке. Задать заголовок к графику. Отобразить легенду к построенным графикам.

1. То же, что и в задании 2, но с другим масштабом осей (выбрать произвольно). Шрифт – 14.
2. Импортировать данные из файла mlb\_players.csv. Построить для игроков из команды «OAK» гистограмму для имен и соответствующих возрастов. Включить для графика основную и дополнительную сетки.
3. Импортировать данные из файла function.csv. Построить график в линейном масштабе, в логарифмическом по оси *y*, в логарифмическом по оси *x*. Подписать оси.
4. На графике из задания 5 подписать ось *x* следующим образом: наименьшее значение - -α, наибольшее - + α. В середине – 0.