**Практическая работа 9**

**Тема. Столбиковые диаграммы, круговые диаграммы и диаграммы в полярных координатах**

**Столбиковые диаграммы и гистограммы**

Основной функцией модуля *pyplot* для создания столбиковой диаграммы является *ax.bar*, которая формирует график из прямоугольных полос, определяемых их левой границей и высотой. Например:

*ax.bar([0, 1, 2], [40, 80, 20])*

По умолчанию задано значение ширины прямоугольников 0.8, но это значение можно изменить с помощью (третьего) аргумента *width*. Если необходимо отцентрировать прямоугольники по вертикали, то для аргумента *align* устанавливается значение *'center'* или выполняется вычисление требуемого положения левых границ:

*w = 0.5*

*x, y = np.array([0, 1, 2]), np.array([40, 80, 20])*

*ax.bar(x, y, w, align='center') # Самый простой способ центрирования столбцов по вертикали.*

*Ax.bar(x – w/2, y, w) # Или вычисление положения их левых границ.*

Дополнительные аргументы, в том числе аргументы, определяющие диаграммы погрешностей, описаны в табл. 1.

**Таблица 1.** Аргументы методов *ax.bar* и *ax.barh*

|  |  |
| --- | --- |
| Аргумент | Описание |
| left | Последовательность координат x левых границ столбцов (но см. также align) |
| width | Ширина столбцов. Если задано скалярное значение, то ширина всех столбцов одинакова. Можно передать массив с разными значениями ширины столбцов |
| bottom | Координаты y нижних границ столбцов |
| height | Последовательность значений высоты столбцов |
| color | Цвет заливки столбцов (скалярное значение или массив) |
| edgecolor | Цвет границ столбцов (скалярное значение или массив) |
| linewidth | Значения толщины линий границ столбцов в пт (скалярное значение или массив) |
| xerr, yerr | Предельные значения для диаграмм погрешностей, как для метода errorbar (скалярное значение или массив) |
| error\_kw | Словарь именованных аргументов, соответствующих специализированным параметрам настройки внешнего вида диаграмм погрешностей |
| align | По умолчанию значение 'edge' определяет выравнивание столбцов по их левым границам (для вертикальных столбцов) или по нижним границам (для горизонтальных столбцов). Значение 'center' центрирует столбцы по их осям |
| log | Если установлено значение True, то используется ось с логарифмической шкалой |
| orientation | 'vertical' (по умолчанию) или 'horizontal' |
| hatch | Определяет тип штриховки столбцов: один из символов '/', '\', '|', '-', '+', 'x', 'o', 'O', '.', '\*'. Повторение символа дает более плотную штриховку |

По умолчанию метод *ax.bar* создает диаграмму с вертикальными столбцами. Диаграмма с горизонтальными столбцами формируется при установке значения *orientation='horizontal'* или при использовании аналогичного метода *ax.barh*.

**Пример 1.** Программа в листинге 1 создает столбиковую диаграмму частоты встречаемости букв английского алфавита по результатам вычисления оценки при анализе текста романа Германа Мелвилла «Моби Дик» (Moby-Dick). Вертикальные столбцы отцентрированы и помечены соответствующей буквой (см. рис. 1).

**Листинг 1.** Частота встречаемости букв в тексте романа «Моби Дик»

*import numpy as np*

*import matplotlib.pyplot as plt*

*text\_file = 'moby-dick.txt'*

*letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'*

*# Initialize the dictionary of letter counts: {'A': 0, 'B': 0, ...}*

*lcount = dict([(l, 0) for l in letters])*

*# Read in the text and count the letter occurences*

*for l in open(text\_file,"r", encoding='utf-8').read():*

*try:*

*lcount[l.upper()] += 1*

*except KeyError:*

*# Ignore characters that are not letters*

*pass*

*# The total number of letters*

*norm = sum(lcount.values())*

*fig = plt.figure()*

*ax = fig.add\_subplot(111)*

*# The bar chart, with letters along the horizontal axis and the calculated*

*# letter frequencies as percentages as the bar height*

*x = range(26)*

*ax.bar(x, [lcount[l]/norm \* 100 for l in letters], width=0.8,*

*color='g', alpha=0.5, align='center')*

*ax.set\_xticks(x)*

*ax.set\_xticklabels(letters)*

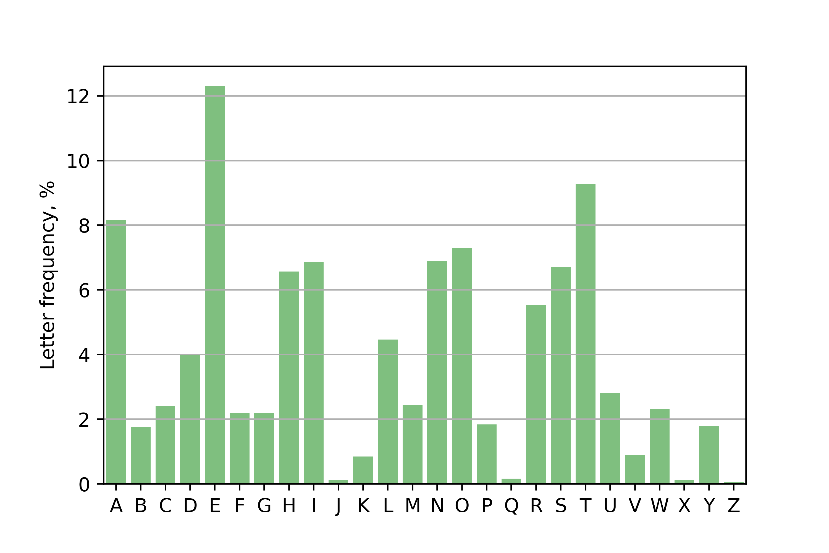
*ax.tick\_params(axis='x', direction='out')*

*ax.set\_xlim(-0.5, 25.5)*

*ax.yaxis.grid(True)*

*ax.set\_ylabel('Letter frequency, %')*

*plt.show()*



**Рис. 1.** Частота встречаемости букв в тесте романа «Моби Дик»

**Пример 2.** Файл *germany-energy-sources.txt* содержит данные о возобновляемых источниках электрической энергии, используемых в Германии с 1990 по 2018 г.:

Программа в листинге 2 отображает эти данные в виде составной столбиковой диаграммы, используя шаблоны штриховки Matplotlib для обозначения различных источников энергии (см. рис. 2).

**Листинг 2.** Визуализация производства электроэнергии из возобновляемых источников в Германии

*import numpy as np*

*import matplotlib.pyplot as plt*

*data = np.loadtxt('germany-energy-sources.txt', skiprows=2, dtype='f8')*

*years = data[:,0]*

*n = len(years)*

*# GWh to TWh*

*data[:,1:] /= 1000*

*fig = plt.figure()*

*ax = fig.add\_subplot(111)*

*sources = ('Hydroelectric', 'Wind', 'Biomass', 'Photovoltaics')*

*hatch = ['oo', '', 'xxxx', '//']*

*bottom = np.zeros(n)*

*bars = [None]\*n*

*for i, source in enumerate(sources):*

*bars[i] = ax.bar(years, bottom=bottom, height=data[:,i+1], color='w',*

*hatch=hatch[i], align='center', edgecolor='k')*

*bottom += data[:,i+1]*

*ax.set\_xticks(years)*

*plt.xticks(rotation=90)*

*ax.set\_xlim(1989, 2019)*

*ax.set\_ylabel('Renewable Electricity (TWh)')*

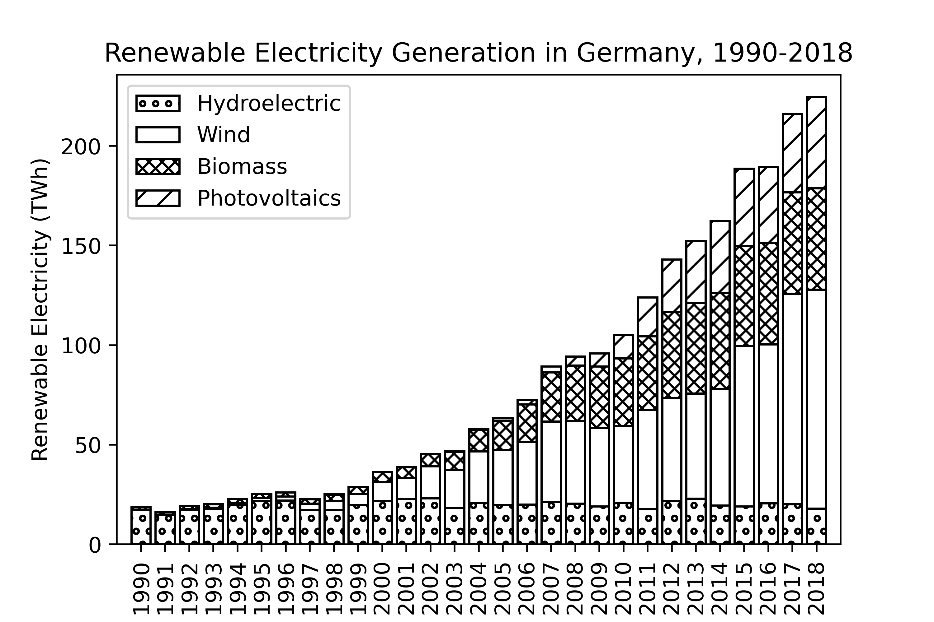
*ax.set\_title('Renewable Electricity Generation in Germany, 1990-2018')*

*plt.legend(bars, sources, loc='best')*

*plt.show()*

*plt.draw()*

*fig.savefig('Renewable energy in Germany.png', dpi=500)*



**Рис. 2.** Составная столбиковая диаграмма производства электроэнергии из возобновляемых источников в Германии, 1990 - 2018

**Круговые диаграммы**

Matplotlib круговую (секторную) диаграмму можно построить очень просто – нужно передать массив значений в метод *ax.pie*. Значения нормализуются по их сумме, если эта сумма больше 1, иначе значения напрямую интерпретируются как доли. Надписи, проценты, вынесенные сегменты и прочие эффекты определяются аргументами, описанными в табл. 2, и демонстрируются в примере 3.

**Таблица 2.** Аргументы для метода *ax.pie*

|  |  |
| --- | --- |
| Аргумент | Описание |
| colors | Последовательность спецификаторов цвета Matplotlib для заливки сегментов |
| labels | Последовательность строк надписей для сегментов |
| explode | Последовательность значений, определяющих дробную часть радиуса круговой диаграммы, на которую смещается каждый клинообразный сегмент (0 для отсутствия эффекта выноса сегмента) |
| shadow | True или False: определяет изображение или отсутствие декоративной тени под круговой диаграммой |
| startangle | Определяет поворот «начальной позиции» круговой диаграммы на заданное число градусов против часовой стрелки относительно горизонтальной оси |
| autopct | Строка формата для подписей к сегментам: соответствующие значения в процентах или функция, генерирующая требуемую строку из данных |
| pctdistance | Радиальное положение текста подписи autopct относительно радиуса круговой диаграммы. По умолчанию 0.6 (т. е. внутри диаграммы, но это может оказаться неприемлемым для слишком узких сегментов) |
| labeldistance | Радиальное положение текста надписи label относительно радиуса круговой диаграммы. По умолчанию 1.1 (снаружи, рядом с границей круговой диаграммы) |
| radius | Радиус круговой диаграммы (по умолчанию 1). Этот параметр полезен при создании перекрывающихся круговых диаграмм с различными радиусами |

**Пример 3.** Программа в листинге 3 графически изображает данные о выбросе в атмосферу парниковых газов с учетом массы в «углеродном эквиваленте» (данные взяты из отчета IPCC 2007 г.).

**Листинг 3.** Круговая диаграмма данных о выбросе в атмосферу парниковых газов

*import numpy as np*

*import matplotlib.pyplot as plt*

*# Annual greenhouse gas emissions, billion tons carbon equivalent (GtCe)*

*gas\_emissions = np.array([(r'$\mathrm{CO\_2}$-d', 2.2),*

*(r'$\mathrm{CO\_2}$-f', 8.0),*

*('Nitrous\nOxide', 1.0),*

*('Methane', 2.3),*

*('Halocarbons', 0.1)],*

*dtype=[('source', 'U17'), ('emission', 'f4')])*

*# 5 colours beige*

*colours = ['#C7B299', '#A67C52', '#C69C6E', '#754C24', '#534741']*

*explode = [0, 0, 0.1, 0, 0]*

*fig, ax = plt.subplots()*

*ax.axis('equal') # So our pie looks round!*

*ax.pie(gas\_emissions['emission'], colors=colours, shadow=True, startangle=90,*

*explode=explode, labels=gas\_emissions['source'], autopct='%.1f%%',*

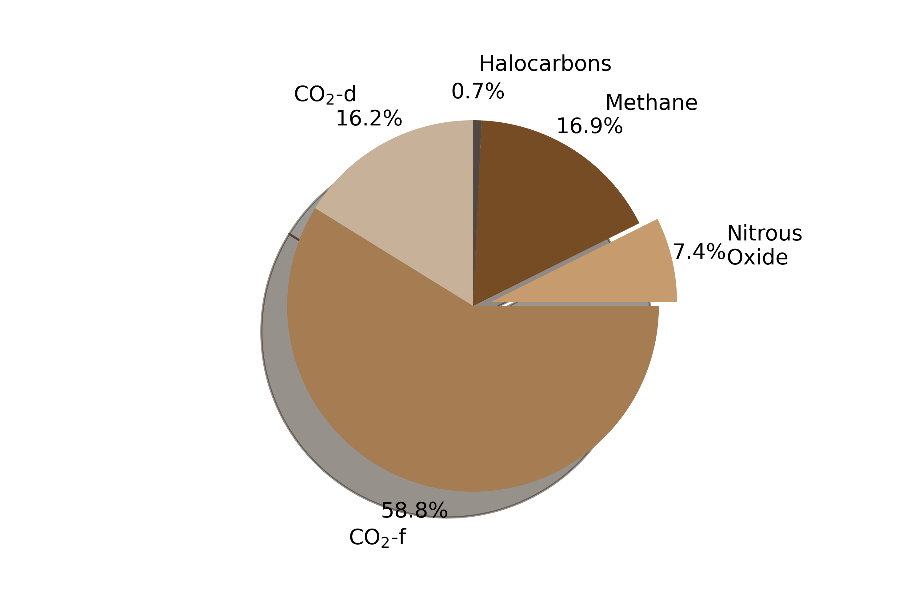
*pctdistance=1.15, labeldistance=1.3)*

*plt.show()*

*plt.draw()*

*fig.savefig('A pie chart of greenhouse gas emissions.png', dpi=500)*

Полученная в результате круговая диаграмма показана на рис. 3.



**Рис. 3.** Выброс в атмосферу парниковых газов в процентах по пяти различным источникам: CO2-d обозначает выброс CO2 из-за уничтожения лесов; CO2-f обозначает выброс CO2 при сжигании природного углеводородного топлива

Последние годы круговые диаграммы стали менее распространенными (их презрительно называют «Comic Sans визуализации данных»), несомненно, разумнее избегать их использования при сравнении большого количества категорий или весьма похожих значений. К счастью, Matplotlib не поддерживает трехмерные круговые диаграммы.

**Диаграммы в полярных координатах**

Диаграмма θ в полярных координатах с радиусом r, определяемая как функция от угла, создается с помощью метода *pyplot.polar*, или посредством определения представления по умолчанию при добавлении внутреннего графика в рисунок:

*fig = plt.figure()*

*ax = fig.add\_subplot(projection='polar')*

В примере 4 демонстрируются оба способа.

**Пример 4.** Система направленных антенн (антенная решетка) может использоваться для ориентации радиоволн в определенном направлении с помощью регулирования их количества, геометрической конфигурации, относительных амплитуд относительных фаз. Рассмотрим систему из *n* изотропных антенн (с равномерным излучением во всех направлениях) в позициях , равномерно распределенных с интервалом *d* по оси *x* от начала координат:



Если одна антенна генерирует вектор излучения , где , то суммарный вектор излучения всех  антенн равен



где  - относительная комплексная амплитуда возбуждения -й антенны, представляющая ее амплитуду и фазу, а множество  известен как множитель (коэффициент) решетки антенн. Можно выбрать , чтобы определить относительные комплексные амплитуды возбуждения для каждой антенны в начальный момент времени. Далее выбирается вариант рассмотрения только по азимутальному () распространению сигнала для изучения в плоскости  c установкой угла в полярных координатах . В этом случае получаем



Типовая относительная мощность излучения (коэффициент усиления) равна квадрату этой величины. Для двух одинаковых антенн:



В коде из листинга 4 относительная величина – коэффициент направления действия (КНД)  - изображена графически на рисунке 4 как функция от  для случая двух антенн на графике в полярных координатах при  и ,

**Листинг 4.** График коэффициента направленного действия (КНД) для системы из двух антенн

*import numpy as np*

*import matplotlib.pyplot as plt*

*def gain(d, w):*

*"""Return the power as a function of azimuthal angle, phi."""*

*phi = np.linspace(0, 2\*np.pi, 1000)*

*psi = 2\*np.pi \* d / lam \* np.cos(phi)*

*A = w[0] + w[1]\*np.exp(1j\*psi)*

*g = np.abs(A)\*\*2*

*return phi, g*

*def get\_directive\_gain(g, minDdBi=-20):*

*"""Return the "directive gain" of the antenna array producing gain g."""*

*DdBi = 10 \* np.log10(g / np.max(g))*

*return np.clip(DdBi, minDdBi, None)*

*# Wavelength, antenna spacing, feed coefficients.*

*lam = 1*

*d = lam*

*w = np.array([1, -1j])*

*# Calculate gain and directive gain; plot on a polar chart.*

*phi, g = gain(d, w)*

*DdBi = get\_directive\_gain(g)*

*plt.polar(phi, DdBi)*

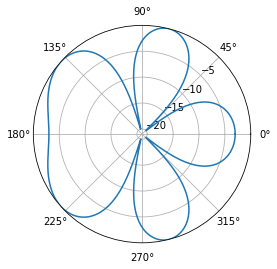
*ax = plt.gca()*

*ax.set\_rticks([-20, -15, -10, -5])*

*ax.set\_rlabel\_position(45)*

*plt.show()*

*plt.draw()*



**Рис. 4.** Коэффициент направленного действия (КНД) системы для двух антенн при  и ,

Методы бродкастинга библиотеки NumPy предоставляют более естественный способ расширения функциональности кода из листинга 4 для произвольного количества антенн. В следующем примере (листинг 5) метод объекта *Figure* *add\_subplot* вызывается с аргументом *projection='polar'* и возвращает соответствующий объект *Axes ax*.

**Листинг 5.** График коэффициента направленного действия (КНД) для системы из трех антенн

*import numpy as np*

*import matplotlib.pyplot as plt*

*def gain(d, w):*

*"""Return the power as a function of azimuthal angle, phi."""*

*phi = np.linspace(0, 2\*np.pi, 1000)*

*psi = 2\*np.pi \* d / lam \* np.cos(phi)*

*j = np.arange(len(w))*

*A = np.sum(w[j] \* np.exp(j \* 1j \* psi[:, None]), axis=1)*

*g = np.abs(A)\*\*2*

*return phi, g*

*def get\_directive\_gain(g, minDdBi=-20):*

*"""Return the "directive gain" of the antenna array producing gain g."""*

*DdBi = 10 \* np.log10(g / np.max(g))*

*return np.clip(DdBi, minDdBi, None)*

*# Wavelength, antenna spacing, feed coefficients.*

*lam = 1*

*d = lam / 2*

*w = np.array([1, -1, 1])*

*# Calculate gain and directive gain; plot on a polar chart.*

*phi, g = gain(d, w)*

*DdBi = get\_directive\_gain(g)*

*fig = plt.figure()*

*ax = fig.add\_subplot(projection='polar')*

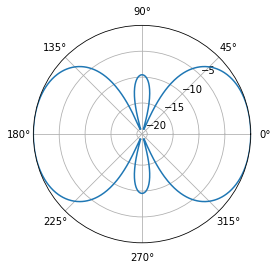
*ax.plot(phi, DdBi)*

*ax.set\_rticks([-20, -15, -10, -5])*

*ax.set\_rlabel\_position(45)*

*plt.show()*

Полученный в результате график показан на рис. 5.



**Рис. 5.** Коэффициент направленного действия (КНД) системы из нескольких (трех) антенн при  и ,, 

**Задание:**

- реализовать примеры 1 - 3, изменив цвет на любой

- реализовать примеры 4, 5 с применением длины волны 

Все графики разместить в репозитории с практической работой 9 с оформленным readme.md. Также в репозитории должен быть файл проекта python jupyter c названием, соответствующим номеру практической работы. В настройках репозитория открыть доступ преподавателю.