10. Одни ли мы во Вселенной? Парадокс Ферми

Ученые подсчитали возможное количество цивилизаций в нашей Галактике с помощью Уравнения Дрейка, учитывающего электромагнитные излучения, а точнее, радиоволны. В 2017 году уравнение было дополнено, и теперь оно учитывает новые экзопланеты, открытые американским спутником «Кеплер». Результаты были опубликованы в научном журнале *Astrobiology* и повергли всех в шок. Вероятность того, что кроме человечества есть ещё хотя бы одна технологически развитая цивилизация, проживающая на другой планете, составляет 1 из 10 миллиардов триллионов! Однако, все мы помним знаменитую фразу нобелевского лауреата по физике Энрико Ферми: «Где все?»

Ферми считал существование внеземных форм жизни более вероятным, чем возможность межзвёздных перелётов. Его знаменитый вопрос, получивший название Парадокса Ферми, звучит следующим образом: «Если бы там кто-то был, то он бы уже с нами связался». Согласно Институту SETI даже с помощью современных и относительно скромных космических технологий можно исследовать всю Галактику и колонизировать её за 10 миллионов лет. Этот срок может показаться большим, но он составляет лишь одну тысячную от времени существования Млечного Пути! Для некоторых людей Парадокс Ферми – доказательство того, что мы одни во Вселенной, а для других выводы, основанные на нём, неубедительны.

В этой главе мы исследуем проблему отсутствия радиосигналов от других цивилизаций и вычислим вероятность того, что одна цивилизация обнаружит другую, на основе зоны распространения её радиоволн и результатов вычисления уравнения Дрейка. Мы также воспользуемся стандартным пакетом библиотеки Питона для создания графического интерфейса (GUI) – `tkinter`. С помощью него мы легко и быстро создадим графическую модель Млечного Пути.

Проект №17: Построение модели Млечного Пути

Наша Галактика, Млечный Путь, относится к довольно распространённому виду спиральных Галактик. См. Рис. 10-1.

Рис. 10-1. Спиральная Галактика NGC 6744, брат-близнец Млечного Пути

В поперечном срезе Млечный Путь представляет собой сплюснутый диск с выпуклостью посередине, в центре которой, скорее всего, находится сверхмассивная чёрная дыра. Из центра Галактики расходятся в разные стороны четыре спиральных рукава, состоящие из относительно плотного скопления газа, пыли и звёзд. Размеры Млечного Пути представлены на Рисунке 10-2.

Рисунок 10-2. Упрощённый вид Галактики Млечный Путь в разрезе. Расстояние указано в световых годах.

Центр Галактики считается относительно неблагоприятным местом для возникновения жизни из-за высокого уровня радиации, вызванной плотным скоплением звёзд.

Итак, в рамках данного проекта мы будем рассматривать Галактику как диск, без учёта выпуклости, а также допустим вероятность существования технологически развитых цивилизаций возле её центра (см. Модель Галактики на Рисунке 10-2).

ЦЕЛЬ

Нам известно количество технологически развитых цивилизаций и средний размер зоны распространения радиоволн. Вычислим вероятность того, что одна цивилизация обнаружит радиосигналы *другой* цивилизации. Изобразим зону распространения радиоволн Земли на двумерной модели Млечного Пути.

СТРАТЕГИЯ

Для выполнения проекта мы сделаем следующее:

1. Найдём количество технически высокоразвитых цивилизаций при помощи Уравнения Дрейка.
2. Выберем размер их зоны распространения радиоволн.
3. Напишем формулу для вычисления вероятности обнаружения одной цивилизации другой цивилизацией.
4. Построим графическую модель Галактики и изобразим зону распространения радиоволн Земли.

Чтобы этапы создания проекта совпадали с этапами написания программы, все этапы будут подробно разобраны в отдельных параграфах. Обратите внимание, что для первых двух задач не требуется Питон.

ОПРЕДЕЛЕНИЕ ЧИСЛА ЦИВИЛИЗАЦИЙ

Число высокоразвитых цивилизаций можно определить при помощи Уравнения Дрейка:

$N = R^* \cdot f_p \cdot n_e \cdot f_l \cdot f_i \cdot f_c \cdot L$

в котором:

N = количество цивилизаций в нашей Галактике с достаточно большим диапазоном радиоволн (который можно обнаружить)

R^* = средний показатель образования звёзд в Галактике (количество новых звёзд в год)

f_p = количество звёзд, имеющих планеты

n_e = среднее количество планет с пригодными для жизни условиями

f_l = количество планет, на которых есть жизнь

f_i = количество планет, на которых есть высокоразвитые цивилизации

f_c = количество планет, на которых есть высокоразвитые цивилизации, которые отправляют радиосигналы о своём существовании в космос

L = промежуток времени, измеряемый в годах, на протяжении которого цивилизации отправляют свои радиосигналы

Благодаря недавним успехам в области обнаружении экзопланет, значения первых трёх переменных определены довольно точно. Что касается переменной n_e, то согласно последним исследованиям на 10-40% планет могут обитать *простейшие формы* жизни.

Что касается других переменных, то в качестве единственного образца мы будем использовать Землю. Наша планета существует 4,5 миллиарда лет, вид *Homo sapiens* существует лишь 200 000 лет, первые цивилизации появились 6 000 лет назад, а радиосигналы мы посылаем в космос лишь на протяжении 112 лет. Что касается переменной L, то войны, эпидемии, эпохи оледенения, падения астероидов, извержения супервулканов, взрывы сверхновых звёзд и выбросы коронального вещества на Солнце могут лишить цивилизацию возможности передавать радиосигналы. И чем меньше время передачи сигнала, тем меньше вероятность того, что цивилизации будут существовать на одном временном промежутке.

В статье в Википедии об Уравнении Дрейка (https://en.wikipedia.org/wiki/Drake_eqiiation), сказано, что Дрейк вместе со своими коллегами установил, что количество цивилизаций, одновременно передающих сигналы в Галактике, колеблется от 1 тысячи до 100 миллионов. Согласно последним исследованиям, это количество находится в пределах от 1 (только мы) до 15 600 000 цивилизаций (Таблица 10-1).

Переменные	Данные на 1961 год	Данные на 2017 год	Ваши собственные данные
R^*	1	3	
f_p	0.35	1	
n_e	3	0.2	
f_l	1	0.13	
f_i	1	1	
f_c	0.15	0.2	
L	$50 \cdot 10^6$	$1 \cdot 10^9$	
N	$7.9 \cdot 10^6$	$15.6 \cdot 10^6$	
** показаны средние значения диапазонов			

В качестве входных данных для нашей программы мы можем использовать данные из таблицы, интернета или ваши собственные вычисления (в последнем столбце таблицы).

ОПРЕДЕЛЕНИЕ РАЗМЕРА ЗОНЫ РАСПРОТРАНЕНИЯ РАДИОВОЛН

Если радиоволны специально не направлены единым лучом, то такие радиоволны являются случайными. Они представляют собой «утечку данных с планеты». Так как мы предпочитаем не заявлять о своём присутствии инопланетянам, которые могут прилететь и съесть нас, то практически все наши сигналы случайны. На данный момент радиоволны составляют сферу вокруг Земли диаметром около 225 световых лет.

Расстояние в 225 световых лет безусловно впечатляет. Но вопрос заключается в другом. Какой диаметр зоны распространения радиосигнала является достаточным для того, чтобы данную цивилизацию смогла обнаружить другая цивилизация? Распространение радиоволн подчиняется Закону обратных квадратов, согласно которому плотность сигнала постоянно ослабевает с пройденным расстоянием. Также плотность снижается из-за поглощения и рассеивания сигнала. Затем наступает момент, когда сигнал ослабевает настолько, что он уже становится неотличим от фонового шума. Даже с помощью самых современных технологий, реализованных в проекте Breakthrough Listen, занимающимся выпуском радиотелескопов, мы можем засечь наш собственный радиосигнал на расстоянии всего лишь в 16 световых лет.

Так как мы хотим узнать, почему мы не обнаружили другие цивилизации, то в этом проекте мы будем исходить из того, что другие цивилизации обладают аналогичными технологиями. Также мы предположим, что инопланетяне, как и мы, страдают паранойей и не посылают в космос сигналы, которые могут выдать их присутствие. Поэтому мы будем исследовать зону распространения радиосигналов в диапазоне чуть меньше и чуть больше земного, т. е. диаметром от 30 до 250 световых лет. Хотя мы не можем определить зону радиосигнала на расстоянии 250 световых лет, будет интересно узнать, какова будет вероятность, если бы мы могли это сделать.

ФОРМУЛА ДЛЯ ВЫЧИСЛЕНИЯ ВЕРОЯТНОСТИ ОБНАРУЖЕНИЯ ДРУГИХ ЦИВИЛИЗАЦИЙ

Так как количество технологически развитых цивилизаций в Галактике увеличивается, то также возрастает и вероятность обнаружения одной цивилизации другой цивилизацией. Это интуитивно понятно, но как же вычислить вероятность математически?

Одно из преимуществ компьютера состоит в том, что мы можем найти решение проблемы с помощью метода грубой силы. И оно может оказаться вовсе не очевидным. Мы создадим трёхмерную модель диска Млечного Пути, случайным образом распределим цивилизации и измерим расстояние между ними при помощи одного из многих инструментов Питона для подсчёта Евклидова расстояния.

Но так как нужно будет проанализировать сотни миллионов цивилизаций, то этот способ окажется весьма неэффективным с точки зрения компьютерных вычислений.

Учитывая, что у нас большое количество неизвестных, то мы будем производить приблизительные вычисления. Объём зоны распространения радиосигнала представляют собой сферу. Для удобства мы представим данную сферу в виде куба с таким же объёмом. Чтобы найти количество зон распространения радиосигналов нужно разделить объём всей Галактики на объём одного куба. Таким образом, мы получим объём одной единицы галактического пространства.

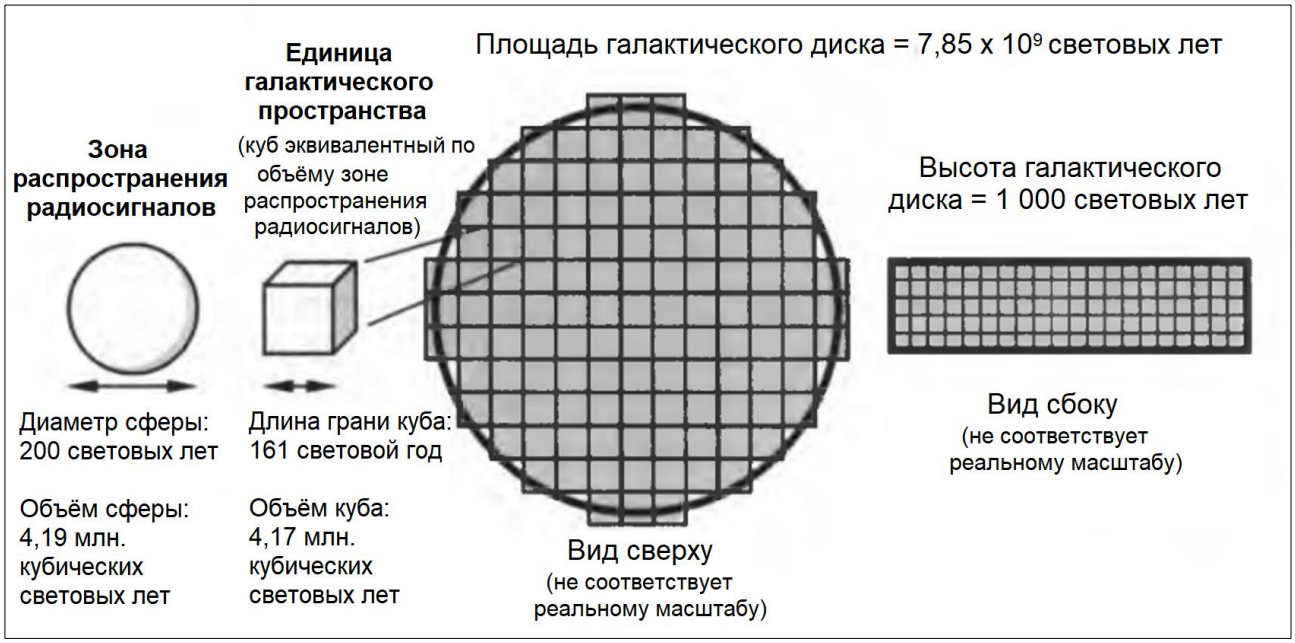

Рисунок 10-3: Модель Галактики. Единица галактического пространства представлена в виде куба. Объём куба равен объёму зоны распространения радиосигналов с диаметром в 200 световых лет

Объёмы можно вычислить по следующим уравнениям, в которых R – это радиус галактического диска, а r – это радиус зоны распространения радиосигналов.

объём диска = $\pi \cdot R^2 \cdot$ высота диска

объём сферы распространения радиосигналов = $4/3 \cdot \pi \cdot r^2$

количество единиц галактического пространства = объём диска / объём зоны распространения радиосигналов

Представьте куб того же объёма, что и сфера. Это число показывает, сколько таких кубов поместится в галактическом диске. И каждый куб имеет свой уникальный номер.

Чтобы разместить цивилизацию, нам нужно случайным образом выбрать номер куба. Если одна и та же единица галактического пространства выбрана дважды, то это означает, что в этом месте обитает две цивилизации. Предположим, что цивилизации, находящиеся в одной единице пространства, т.е. в одной зоне распространения радиосигналов, могут обнаружить друг друга, хотя вовсе не обязательно (см. Рисунок 10-4).

Но так как мы будем использовать большое количество цивилизаций, то погрешность в вычислениях будет стремиться к нулю, также как и в ситуации, когда мы складываем много округлённых чисел.

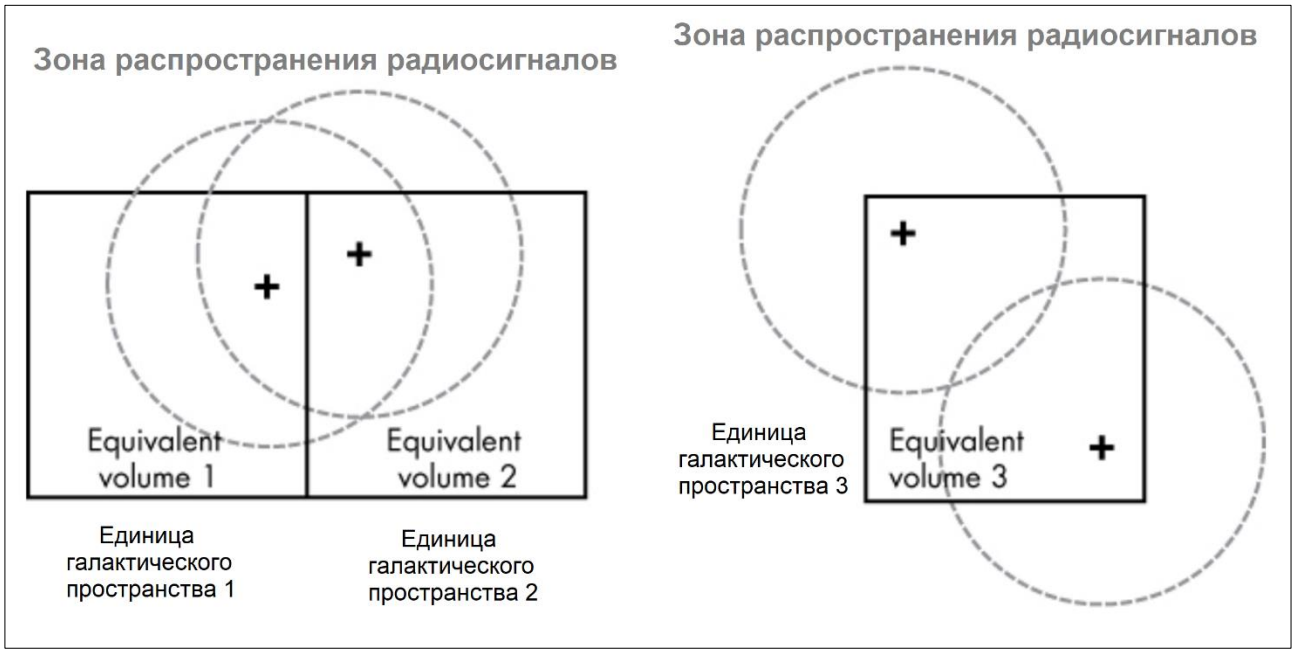

Рисунок 10-4: Сложности обнаружения на примере двух цивилизаций с пересекающимися зонами распространения радиосигналов

Вычисления числа цивилизаций и / или зон распространения радиосигналов можно представить в виде уравнения с многочленами, с помощью которого можно будет вычислить все возможные варианты. Многочлен – это сумма или разность одночленов. Квадратное уравнение, которое мы все изучали в школе, является примером уравнения с многочленами второй степени.

$$\mathbf{ax^2 + bx + c = 0}$$

Графики уравнений с многочленами строятся относительно несложно, поэтому они идеально подходят для нашей задачи. Но чтобы наша формула работала с разным количеством цивилизаций и с зонами распространения радиосигналов разного размера, нам нужно будет посчитать *соотношение* количества цивилизаций к общему объёму.

На Рисунке 10-5 каждая точка обозначает вероятность того, что одна цивилизация обнаружит другую, относительно соотношения, изображённого на оси Ox. Уравнение, изображённое на графике, является уравнением второй степени. Благодаря ему можно вычислить вероятность существования любого количества цивилизаций в одной зоне распространения радиосигналов (одной единице галактического пространства или в одном кубе): от 1 до 5 цивилизаций. Если в данном пространстве обитает более 5 цивилизаций, то мы считаем вероятность обнаружения другой разумной расы равной 1 или 100%.

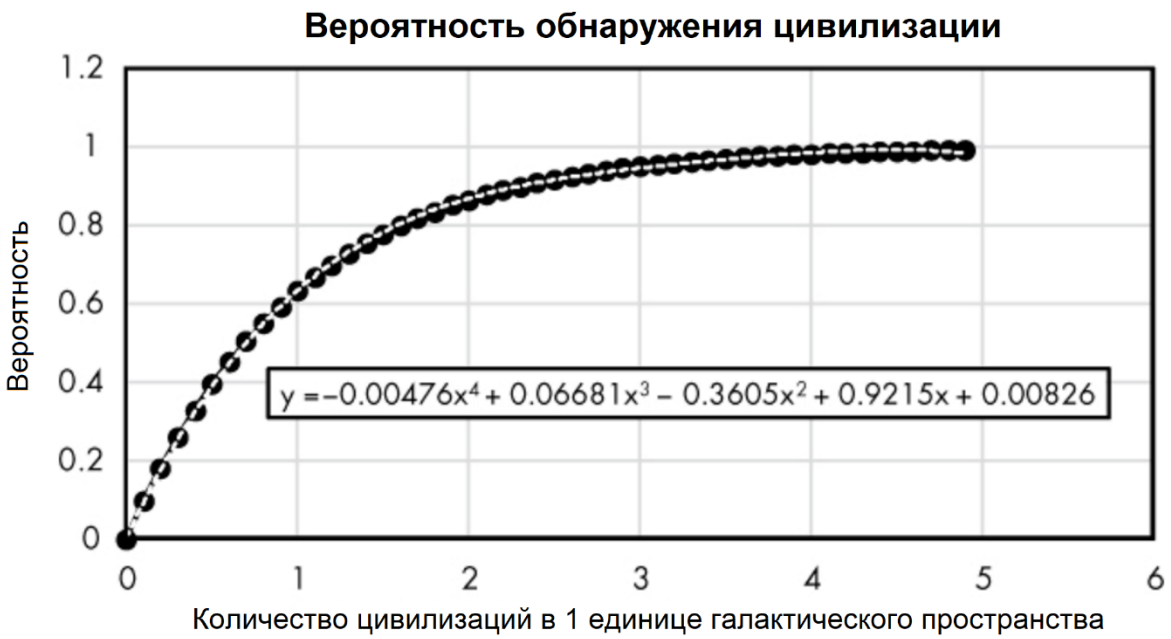

Рисунок 10-5: Вероятность того, что одна цивилизация обнаружит другую, в зависимости от числа цивилизаций, находящихся в одной единице галактического пространства

На Рисунке 10-5 на оси `0x` изображено количество цивилизаций, находящихся в одной единице галактического пространства. Например, число 0,5 означает, что в одной единице галактического пространства находится 0,5 цивилизации, 2 означает 2 цивилизации и так далее. На оси `0y` изображена вероятность того, что в 1 единице галактического пространства будет находиться более 1 цивилизации.

Следует также отметить, что согласно Рисунку 10-5, требуется большое число цивилизаций, чтобы хотя бы у одной из них был сосед. Представьте, что в 999999 единицах пространства из 1 000 000 находится, по крайней мере, по две цивилизации, и мы используем свои суперспособности, чтобы случайным образом добавить ещё одну цивилизацию. Вероятность один на миллион, что у этой цивилизации будет один единственный сосед. Это равняется вероятности того, что мы сможем найти иголку в стоге сена!

Примечание

Главное правило построения компьютерной модели – начинайте с простого и постепенно увеличивайте сложность. Самое простое предположение – это то, что технологически развитые цивилизации распределены в Галактике случайным образом. В «Трудных проектах» на стр. 214 у вас будет возможность усложнить вашу модель, используя понятие об обитаемых зонах Галактики.

КОД, ВЫЧИСЛЯЮЩИЙ ВЕРОЯТНОСТЬ ОБНАРУЖЕНИЯ ДРУГОЙ ЦИВИЛИЗАЦИИ

Код, вычисляющий вероятность обнаружения другой цивилизации, выбирает случайным образом места в Галактике из заданного числа мест и цивилизаций. Затем подсчитывает число единиц пространства с одной цивилизацией и повторяет вычисления до тех пор, пока не будет произведена оценка вероятности. Затем то же самое повторяется для другого количества цивилизаций. Результат представляет собой вероятность числа цивилизаций, приходящихся на 1 единицу пространства, а не фактическое число цивилизаций, и записывается в виде многочлена. Поэтому программу нужно запустить лишь один раз.

Для создания уравнения с многочленами и проверки данных мы будем использовать `NumPy` и `matplotlib`. Библиотека `NumPy` поддерживает работу с большими многомерными массивами и матрицами и содержит большое количество математических функций. Библиотека `matplotlib` поддерживает создание двухмерных и простых трёхмерных моделей, а библиотека `NumPy` является её математическим аппаратом.

Есть несколько способов установить дистрибутивы Питона для научной работы. Первый способ – использовать `SciPy`, библиотеку Питона с открытым кодом для научных и технических вычислений (*https://scipy.org/index.html*). Если вы будете заниматься анализом и визуализацией данных, то тогда вам следует загрузить бесплатный пакет, например, Anaconda или Enthought, который работает на Windows, Linux и MacOS. Эти пакеты избавят вас от необходимости загружать библиотеки для научной работы и проверять их на совместимость. Список пакетов программ и ссылки на сайты можно найти здесь: *https://scipy.org/install.html*.

В качестве альтернативного варианта вы можете загрузить библиотеки напрямую, используя pip. Инструкции на сайте: *https://scipy.org/install.html*. Так как в PowerShell имеется много зависимостей, то эти библиотеки нужно устанавливать одновременно. Если у вас стоит Windows, то можете запустить следующую команду для Python 3 через PowerShell из папки с Питоном (вы можете не печатать цифру 3, если у вас установлена только одна версия Python – Python 3).

```
$ python3 -m pip install --user numpy scipy matplotlib ipython jupyter
pandas sympy nose
```

Таким образом установятся все необходимые модули. Что касается кода 10-1 и 10-2, то вы можете набрать его вручную или скачать с сайта *https://www.nostarch.com/impracticalpython/*.

Вычисляем вероятность обнаружения цивилизаций в заданном диапазоне

Во Фрагменте кода 10-1 импортируются модули и выполняется вся работа, описанная выше. Подстановка значений в многочлен и проверка с помощью библиотеки `matplotlib` будет описана во второй части.

probability_of_detection.py, часть 1

```
❶ from random import randint
from collections import Counter
import numpy as np
import matplotlib.pyplot as plt

❷ NUM_EQUIV_VOLUMES = 1000  # Number of locations in which to place civiliza-
tions.
MAX_CIVS = 5000  # Maximum number of advanced civilizations.
TRIALS = 1000  # Number of times to model a given number of civilizations.
CIV_STEP_SIZE = 100  # Civilizations count step size.

❸ x = []  # x values for polynomial fit.
y = []  # y values for polynomial fit.

❹ for num_civs in range(2, MAX_CIVS + 2, CIV_STEP_SIZE):
    civs_per_vol = num_civs / NUM_EQUIV_VOLUMES
    num_single_civs = 0  # A counter to keep track of the number of locations
containing a single civilization.
    ❺ for trial in range(TRIALS):
        locations = []  # Equivalent volumes containing a civilization.
        ❻ while len(locations) < num_civs:
            location = randint(1, NUM_EQUIV_VOLUMES)
            locations.append(location)
        ❼ overlap_count = Counter(locations)
        overlap_rollup = Counter(overlap_count.values())
        num_single_civs += overlap_rollup[1]

    ❽ prob = 1 - (num_single_civs / (num_civs * TRIALS))

    ❾ # Print ration of civs-per-volume vs. probability of 2+ civs per loca-
tion.
    print("{:.4f} {:.4f}".format(civs_per_vol, prob))
    ❿ x.append(civs_per_vol)
    y.append(prob)
```

Фрагмент кода 10-1: Импортируются модули. Случайным образом выбирается единица пространства с заданной зоной распространения радиосигналов. Вычисляется вероятность одновременного нахождения нескольких цивилизаций в одной единице пространства.

Мы импортируем уже нам знакомый модуль `random` и модуль `Counter`, с помощью которых мы будем подсчитывать количество цивилизаций в одной единице пространства в зависимости от того, сколько раз модуль `random` выберет эту единицу ❶. Чуть позже мы познакомимся с работой модуля `Counter`. Мы импортируем библиотеку `NumPy` для построения многочлена и библиотеку `matplotlib` для его последующей проверки.

Создадим константы, в которых будет храниться информация, введённая пользователем: число единиц пространства (схематически представленных в виде кубов), максимальное число цивилизаций, число симуляций и шаг выборки цивилизаций для модуля `Count` ❷. Так как результаты предсказуемы, то можно использовать большой шаг со значением 100, и точность от этого не пострадает. Обратите внимание, что у нас получатся аналогичные результаты, независимо от выбранного количества единиц галактического пространства. Мы можем взять 100 единиц или 100 000, и результаты будут идентичны.

Создадим список для хранения значений переменных `x` и `y` ❸. На оси `Ox` будет представлено количество цивилизаций в 1 единице пространства, а на оси `Oy` будет представлена вероятность обнаружения одной цивилизации другой цивилизацией.

Далее начинается серия вложенных циклов. Первый цикл отвечает за количество цивилизаций в нашей модели ❹. Нужны, по крайней мере, две цивилизации, чтобы они смогли обнаружить друг друга. Поэтому для того чтобы в подсчётах были задействованы все возможные цивилизации, мы увеличим максимальное количество цивилизаций (переменная `max_civs`) на 2. Создадим константу `civ_step_size` для хранения значения шага.

Подсчитаем количество цивилизаций, обитающих в одной единице пространства, (`civs_per_vol`) и запустим счётчик (`num_single_civs`), который будет отслеживать число единиц пространства, содержащих только одну цивилизацию.

Теперь с помощью цикла `for` проведём заданное число симуляций ❺. В каждой симуляции будет использоваться одно и тоже число цивилизаций. Создадим переменную «единицы пространства» (`locations`) и присвоим ей пустой список. Затем случайным образом выберем номер единицы галактического пространства для каждой цивилизации ❻ и добавим его в конец списка. Повторяющиеся значения – `overlap_count` – в списке будут обозначать места в Галактике с несколькими цивилизациями.

Запустим счётчик `Counter` по этому ❼ списку и получим значения. В конце цикла получим количество единиц пространства, в которых находится только по одной цивилизации, и добавим их в счётчик `num_single_civs`. Ниже приведён пример работы этих функций:

```
>>> from collections import Counter
>>> alist = [124, 452, 838, 124, 301]
>>> count = Counter(alist)
>>> count
Counter({124: 2, 452: 1, 301: 1, 838: 1})
>>> value_count = Counter(count.values())
>>> value_count
Counter({1: 3, 2: 1})
>>> value_count[1]
3
```

В списке `alist` содержится пять чисел, и одно из них (124) встречается дважды. Функция `Counter` создаёт словарь, в котором ключи – это числа из обрабатываемого списка, а значения – это их количество (то есть то, сколько раз они встретились в исходном списке). Передаём функции `Counter` значения, хранящиеся в переменной `count`, с помощью метода `values()`. Функция `Counter` создаст другой словарь, в котором значения из предыдущего словаря станут ключами, а то сколько раз они встретились станут значениями. Если вы хотите узнать, сколько чисел встретилось только один раз, то используйте метод `value_count[1]`, который возвратит количество чисел без дубликатов. Мы узнаем, сколько раз встретилась единица, то есть количество единиц пространства, содержащих лишь одну цивилизацию.

Теперь мы используем результаты из `Counter` для вычисления вероятности нахождения нескольких цивилизаций в одной единице пространства, учитывая распространение цивилизаций ❽. Единица минус число единиц пространства с 1 цивилизацией, делённая на количество цивилизаций, умноженных на количество симуляций.

Выводим на экран количество цивилизаций в одной единице пространства и вероятность того, что они обнаружат друг друга ❾. Ниже представлены первые несколько строк:

```
0.0020  0.0020
0.1020  0.0970
0.2020  0.1832
```

Этот первый шаг служит для проверки работы программы, и он необязателен. Сделайте эту строку комментарием, если вы хотите ускорить время выполнения программы. В конце добавьте значения в списки `x` и `y` ❿.

Создаём формулу для вычисления вероятности и проверяем результаты

Во Фрагменте кода 10-2 мы будем использовать библиотеку `NumPy` для создания полиномиальной регрессии, в которой будет рассматриваться вероятность обнаружения и коэффициент цивилизаций на подсчитанный объём во Фрагменте 10-1. Мы будем использовать многочлен в следующем файле для вычисления вероятности. Чтобы проверить совпадение фактических и прогнозируемых значений, мы построим их графики с помощью библиотеки `matplotlib`.

probability_of_detection.py, часть 2

```
❶ coefficients = np.polyfit(x, y, 4)  # 4th order polynomial fit.
❷ p = np.poly1d(coefficients)
print("\n{}".format(p))
# The predicted x-axis values.
❸ xp = np.linspace(0, 5)
The predicted y-axis probability.
❹ _ = plt.plot(x, y, '.', xp, p(xp), '-')  # p(xp)
❺ plt.ylim(-0.5, 1.5)
❻ plt.show()
```

Фрагмент кода 10-2: Вычисление полиномиальной регрессии и вывод на экран графика для проверки расчётов.

Создадим переменную «коэффициенты» – `coefficients` – и сохраним в ней результаты вычислений метода `polyfit()` из библиотеки `NumPy` ❶. Этот метод принимает в качестве аргументов списки `x` и `y` и целое число, представляющее собой степень соответствующего многочлена. Метод возвращает вектор коэффициентов `p`, который минимизирует квадратичную ошибку.

Если вы выведите значения, хранящиеся в переменной `coefficients`, на экран, то получите следующее:

```
[-0.00475677  0.066811  -0.3605069   0.92146096  0.0082604 ]
```

Чтобы получилось полное выражение, передайте значения переменной `coefficients` в `poly1d` и сохраните результаты в новой переменной ❷. Выведите значение на экран, и у вас получится многочлен аналогичный многочлену на Рисунке 10-5:

```
      4       3       2
-0.004757 x + 0.06681 x - 0.3605 x + 0.9215 x + 0.00826
```

Чтобы проверить правильно ли многочлен воспроизводит входные данные, нужно построить график, где на оси `0x` будет изображено количество цивилизаций в одной единице галактического пространства, а на оси `0y` – вероятность обнаружения одной цивилизации другой цивилизацией.

Мы воспользуемся методом `linspace()` из библиотеки `NumPy` для вычисления значений на оси `0x`. Этот метод возвращает равномерно распределённые числа в заданном интервале. Укажем значения `(0, 5)`, так как они покрывают весь диапазон вероятностей ❸.

Теперь отобразим полученные значения – `xy` – и прогнозируемые значения – `xp, p(xp)` – на одном графике. Вызовем метод `plot()` и передадим в него в качестве первого аргумента список со значениями `x` и в качестве второго аргумента список со значениями `y`. Третий аргумент – это способ отображения, мы укажем точку (см. Рис. 10-5) ❹. Затем передадим прогнозируемые значение на оси `0x` (`xp`), и чтобы получить значения вероятности на оси `0y`, передадим `p` ту же переменную: `p(xp)`. Отобразим результаты с помощью тире.

Ограничим ось `0y` диапазоном от `0.5` до `1.5` ❺ и с помощью метода `show()` отобразим график (см. Рис. 10-6) ❻. У нас получился простой график, единственная цель которого – показать правильность работы полиномиальной регрессии. В многочлене можно увеличить или уменьшить третий аргумент в шаге ❶.

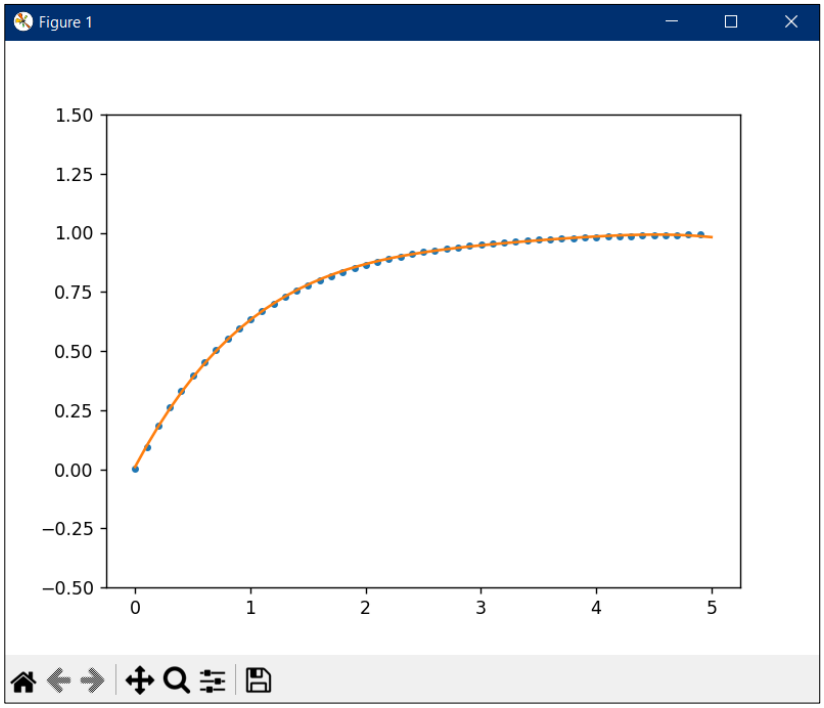

Рис. 10-6. Вычисленные значения представлены точками. Прогнозируемые значения представлены в виде тире и образуют линию.

Благодаря этим данным мы можем мгновенно определить вероятность обнаружения для любого числа цивилизаций. Нужно лишь решить многочлен с помощью Питона.

ПОСТРОЕНИЕ ГРАФИЧЕСКОЙ МОДЕЛИ

Графическая модель будет представлять собой двумерное изображение галактического диска сверху. На ней также будет изображена зона распространения радиосигналов Земли, и мы увидим, какое крошечное место в Галактике занимает наша планета.

Построение модели Млечного Пути сводится к построению модели спиральных рукавов. Спиральный рукав представляет собой логарифмическую спираль. Эта геометрическая форма так часто встречается в природе, что её назвали *spira mirabilis*, что переводится с латыни как «чудесная спираль». Если мы сравним Рисунок 10-7 с Рисунком 10-1, то мы заметим, как похожа структура урагана на структуру Галактики. Центр урагана напоминает сверхмассивную чёрную дыру, а облачный вихрь – горизонт событий.

Рисунок 10-7: Ураган Игорь

Спирали расходятся в стороны из центральной точки, называемой *полюсом*, поэтому их лучше всего изобразить с помощью *полярной системы координат* (Рисунок 10-8). В отличие от Декартовой системы координат вместо x и y используется r и θ (тета). r – это расстояние от центра, а θ (тета) – это угол между r и осью 0x. Координаты центра: (0, 0).

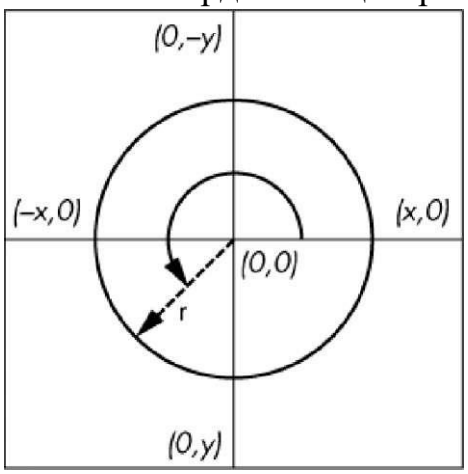

Рисунок 10-8: Полярная система координат

Уравнение логарифмической спирали в полярной системе координат выглядит следующим образом:

$r = ae^{b\theta}$

r – расстояние от центра, θ – угол между r и осью `0x`, e – основание натурального логарифма, a и b – коэффициенты.

С помощью этой формулы мы нарисуем первую спираль, затем сделаем поворот и нарисуем ещё одну, пока у нас не получится четыре рукава Млечного Пути. Мы построим спирали из окружностей разных размеров, которые будут символизировать звёзды. Рисунок 10-9 – это пример одного из вариантов реализации графической модели. Вы можете сами поэкспериментировать с переменными, пока не добьётесь наилучшего результата.

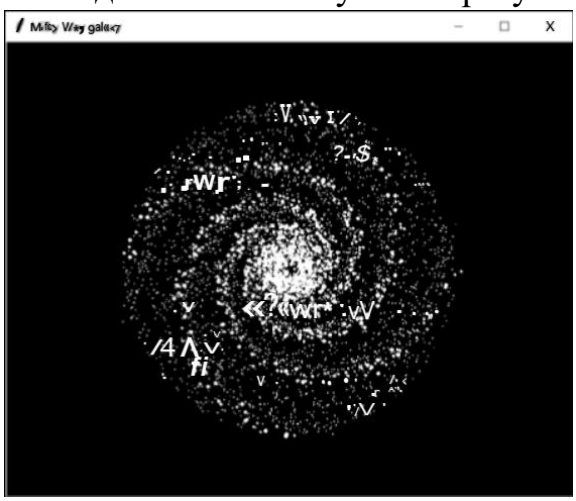

Рисунок 10-9: Модель Млечного Пути, созданная с помощью логарифмических спиралей

Я создал изображение на Рисунке 10-9 при помощи `tkinter` (произносится как "ти-кей-интер"). Это стандартная библиотека Питона для создания графического интерфейса (GUI). Хотя `tkinter` изначально был предназначен для создания окон, кнопок, полос прокрутки и тому подобного, в нём можно создавать графики, таблицы, заставки экрана, простые игры и многое другое. Главное его преимущество – это то, что он входит в стандартный дистрибутив Питона и работает во всех операционных системах, поэтому нам не нужно будет устанавливать дополнительные библиотеки. К нему имеется хорошо написанная документация, и он прост в использовании.

На большинстве устройств с Windows, macOS, и Linux уже установлен `tkinter`. Если у вас его нет, или вы хотите установить последнюю версию, то вы можете сделать это на сайте: *https://www.activestate.com/*. Напоминаю, что если модуль уже установлен, то вы сможете импортировать его в интерпретатор без ошибок:

```
>>> import tkinter
>>>
```

В книгах по Питону начального уровня иногда описывается `tkinter`. Официальная документация на сайте: https://docs.python.org/3/library/tk.html. Список дополнительных материалов указан в разделе «Дальнейшее чтение» на с.212.

Выбор масштаба графической модели

На нашей графической модели один пиксель будет равняться одному световому году, а ширина одного пикселя – диаметру зоны распространения радиоволн. Таким образом, когда мы будем менять размер диаметра зоны распространения радиоволн, будет меняться масштаб всей модели, и она будет строится заново. Мы будем использовать следующее уравнение:

```
радиус диска в заданном масштабе = радиус диска / диаметр зоны распространения радиоволн
```

Радиус диска равен 50 000 световых лет.

Когда мы выбираем маленький диаметр зоны распространения радиоволн, то графическая модель сжимается. Когда мы выбираем большой диаметр, то она расширяется. (Рисунок 10-10).

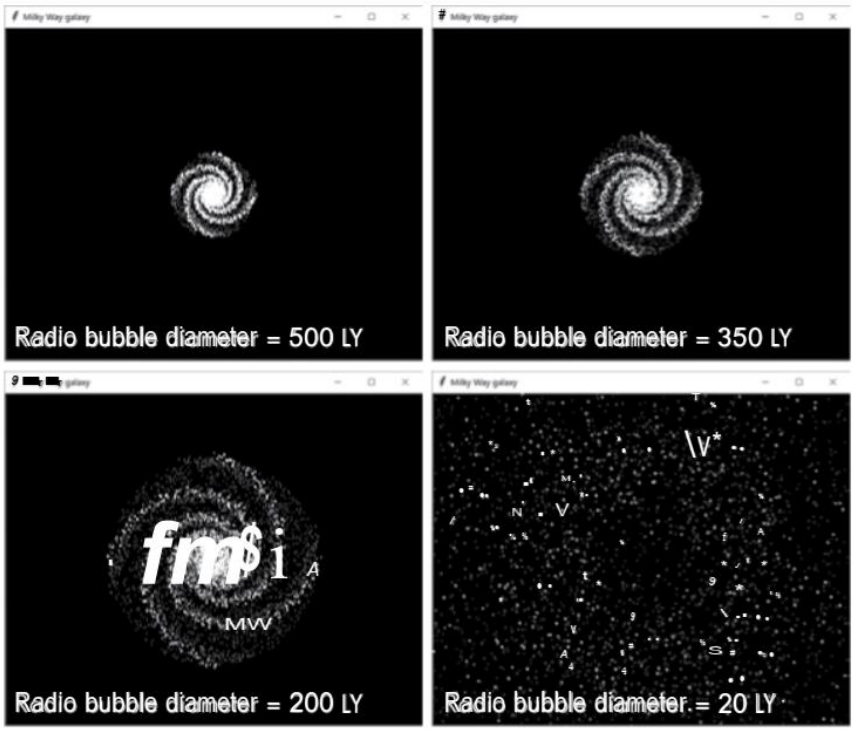

Рисунок 10-10: Вид модели Галактики в зависимости от диаметра зоны распространения радиосигналов

Пишем код для создания модели Галактики

Наш код будет подсчитывать вероятность обнаружения одной цивилизации другой цивилизацией. Мы сможем указать любое число цивилизаций, обитающих в одной единице галактического пространства, и любой диаметр зоны распространения радиосигналов. С помощью этого кода мы построим графическую модель Галактики. Если мы будем использовать такой же диаметр зоны распространения радиосигналов как у Земли, то на модели будет указано приблизительное местоположение нашей Солнечной системы, подписанное красным цветом. Код можно скачать с сайта: *https://www.nostarch.com/impracticalpython/*.

Входные данные и ключевы параметры

Во Фрагменте кода 10-3 представлено начало файла *galaxy_simulator.py*. Сначала мы импортируем модули и запишем константы.

```
❶ import tkinter as tk
from random import randint, uniform, random
import math

#
================================================================================
❷ # MAIN INPUT
# Scale (radio bubble diameter) in light-years:
❸
SCALE = 225  # enter 225 to see Earth's radio bubble

# Number of advanced civilizations from the Drake equation:
❹ NUM_CIVS = 15_600_000
#
================================================================================
```

Фрагмент кода 10-3: Импорт модулей и присвоение значений константам.

Импортируем модуль `tkinter` и запишем его как `tk`, чтобы не нужно было полностью писать название при вызове классов ❶. Если вы пользуетесь версией Питон 2, то напишите `Tkinter` с заглавной буквы. Также нам потребуется модули `random` и `math`.

Визуально отделим блок программы с входными данными: перед началом блока поставим знак #, а на следующей строке поставим несколько знаков “=”, и то же самое поставим в конце блока. В переменной «МАСШТАБ» – `SCALE` – будет храниться длина диаметра зоны распространения радиосигналов в световых годах ❸. В переменной «ЧИСЛО ЦИВИЛИЗАЦИЙ» – `NUM_CIVS` – будет хранится число цивилизаций. Мы можем использовать результаты вычисления уравнения Дрейка или подставить свои данные ❹.

Настраиваем окно, выводимое на экран, с помощью tkinter и присваиваем значения константам

Во Фрагменте кода 10-4 создаётся объект библиотеки `tkinter`, на котором можно рисовать объекты. Здесь будет изображена карта Галактики, наша графическая модель. Мы присвоим значения константам, описывающим размеры Млечного Пути.

```
# Set up display canvas.
❶ root = tk.Tk()
root.title("Milky Way Galaxy")
# 'c' stands for canvas.
❷ c = tk.Canvas(root, width=1000, height=800, bg='black')
❸ c.grid()
# This sets the origin coordinates (0, 0) to the center of the canvas.
❹ c.configure(scrollregion=(-500, -400, 500, 400))

# Actual Milky Way dimensions (light-years).
❺ DISC_RADIUS = 50_000
DISC_HEIGHT = 1_000
❻ DISC_VOL = math.pi * DISC_RADIUS ** 2 * DISC_HEIGHT
```

Фрагмент кода 10-4: Создаём окно и область для окна программы, присваиваем значения константам.

Создадим окно и назовём его стандартным именем `root` ❶. Это окно верхнего уровня, на котором будет изображена вся информация. На следующей строке присвоим окну имя «Галактика Млечный Путь», которое будет написано на рамке в верхней левой части окна (Рисунок 10-9).

Затем добавим к главному окну компонент под названием виджет. Виджет означает "гаджет Виндоус". В библиотеке `tkinter` 21 базовый виджет: надписи, рамки, кнопки, полосы прокрутки и другие. Создаём виджет `Canvas`, который будет содержать все графические объекты ❷. Это виджет общего назначения, предназначенный для графики и сложных макетов. Определите параметры главного окна: ширину, высоту и цвет заднего фона. Назовите окно `c`, что означает «холст» *(canvas)*.

Мы можем разделить виджет *canvas* на строки и столбцы как таблицу. Каждая ячейка в этой сетке может содержать свой виджет, и эти виджеты могут занимать несколько ячеек. В пределах ячейки можно выровнять виджет при помощи `sticky`. Для управления каждым виджетом в окне необходимо пользоваться `grid`. Так как у нас в проекте только один виджет, поэтому не нужно передавать в `grid` никакие значения ❸.

Теперь осталось настроить холст с помощью `scrollregion` ❹. В нём мы пропишем начало координат (0, 0) в центре холста, чтобы мы смогли нарисовать рукава Галактики в полярной системе координат. Если не прописать эти значения, то начало будет в левом верхнем углу холста.

Аргументы, переданные в `configure`, устанавливают границы холста. Они должны составлять половину его ширины и высоты. Например, если вы установите границы прокрутки `600`, `500`, тогда холст должен быть размером `1200`, `1000`.

Данные значения подходят для вывода изображения на маленьком ноутбуке, но вы можете менять их, если вам нужно окно побольше.

Далее идёт раздел с константами, описывающими размеры Галактики ❺. В дальнейшем вам нужно будет прописать часть этих переменных в функциях. Мы прописали их в глобальном пространстве имён, чтобы объяснение кода было более логичным. Первая константа – это радиус галактического диска. Вторая константа – это высота галактического диска (см. Рисунок 10-2). Третья константа – это объём диска ❻.

Определяем масштабы модели Галактики и вычисляем вероятность обнаружения другой цивилизации

Во Фрагменте кода 10-5 представлены две функции. Первая функция определяет масштаб Галактики в зависимости от выбранного размера зоны распространения радиосигналов. Вторая функция вычисляет вероятность обнаружения цивилизациями друг друга.

```
# Set up display canvas.
❶ def scale_galaxy():
    """Scale galaxy dimensions based on radio bubble size (scale)."""
    disc_radius_scaled = round(DISC_RADIUS / SCALE)
    ❷ bubble_vol = 4 / 3 * math.pi * (SCALE / 2) ** 3
    ❸ disc_vol_scaled = DISC_VOL / bubble_vol
    ❹ return disc_radius_scaled, disc_vol_scaled


❺ def detect_prob(disc_vol_scaled):
    """Calculate probability of galactic civilizations detecting each other."""
# Ratio of civilizations to scaled galaxy volume.
    ❻ ratio = NUM_CIVS / disc_vol_scaled
    ❼ if ratio < 0.002:  # Set very low ratios to probability of 0 (zero).
        detection_prob = 0
    elif ratio >= 5:  # Set high ratios to probability of 1.
        detection_prob = 1
    ❽ else:
        detection_prob = -0.004757 * ratio ** 4 + 0.06681 * ratio ** 3 - 0.3605 * \
                         ratio ** 2 + 0.9251 * ratio + 0.00826
    ❾ return round(detection_prob, 3)
```

Фрагмент кода 10-5: Определение масштаба модели Галактики и вероятность обнаружения другой цивилизации.

Создадим функцию «вычислить размер Галактики» – `scale_galaxy()`, которая будет вычислять размеры Галактики на основании радиуса зоны распространения радиосигналов. Она будет использовать константы из глобального пространства имён, поэтому ей не нужно передавать аргументы. Подсчитаем размер галактического диска и объём зоны распространения радиоволн при помощи уравнения нахождения объёма сферы и сохраним результаты в переменной «объём зоны распространения радиоволн» – `bubble_vol` ❷.

Затем разделим объём галактического диска на «объём зоны распространения радиоволн» – `bubble_vol`, чтобы получить число единиц пространства (the

`scaled disc volume`) ❸. Таким образом мы узнаем, сколько единиц пространства в Галактике, т.е. мест, в которых могут обитать другие цивилизации. Функция возвращает значения `disc_radius_scaled` и `disc_vol_scaled` ❹.

Теперь напишем функцию «вычислить вероятность» – `detect_prob()`, которая будет вычислять вероятность того, что одна цивилизация обнаружит другую цивилизацию. В качестве аргумента она будет принимать количество единиц пространства (`the scaled disc volume`) ❺. Количество цивилизаций делим на количество единиц пространства, и данный коэффициент подставляем в многочлен на место `x` ❻. Так как у полиноминальной регрессии могут быть проблемы в конечных точках, то мы будем использовать условные операторы, чтобы задать маленькие коэффициенты для вероятности наступления события равной нулю и большие для вероятности равной единице ❼. В других случаях будет использоваться многочлен, полученный в результате работы файла *probability_of_detection.py* ❽. Функцию будет возвращать вероятность, округлённую до трёх знаков после запятой ❾.

Полярная система координат

Во Фрагменте кода 10-6 представлена функция, которая выбирает случайным образом координаты единицы пространства `(x, y)` в полярной системе координат. Наша функция будет выбирать местоположения звёзд, которые мы поместим в нашу графическую модель. Мы не будем использовать ось `z`, так как у нас двумерная модель.

```
# Set up display canvas.
❶ def random_polar_coordinates(disc_radius_scaled):
    """Generate uniform random (x, y) point within a disc for 2D display."""
    ❷ r = random()
    ❸ theta = uniform(0, 2 * math.pi)
    ❹ x = round(math.sqrt(r) * math.cos(theta) * disc_radius_scaled)
    y = round(math.sqrt(r) * math.sin(theta) * disc_radius_scaled)
    ❺ return x, y
```

Фрагмент кода 10-6: Функция, которая случайным образом выбирает значения x и y в полярной системе координат

В качестве аргумента функция принимает радиус галактического диска ❶. С помощью функции `random()` мы случайным образом выберем число с плавающей точкой в диапазоне от 0,0 до 1,0 и сохраним его в переменной `r` ❷. Затем мы случайным образом выберем значение теты (`theta`) в диапазоне от 0 до 360 градусов (2 π – это эквивалент 360 градусов в радианах).

С помощью данных уравнений мы равномерно распределим точки в каждой единице пространства на галактическом диске:

$x = \sqrt{r} * \cos r$

$y = \sqrt{r} * \sin \theta$

Уравнения будут вычислять координаты в диапазоне от 0 до 1. Умножаем результаты на радиус галактического диска ❹. Возвращаем значение `x` и `y` ❺.

Строим рукава Галактики

Во Фрагменте кода 10-7 представлена функция, которая строит рукава Галактики при помощи уравнения логарифмической спирали. Возможно, спираль и чудесная, но все чудеса заключаются в том, чтобы добавить деталей к изначально построенному рукаву. Мы будем менять размеры звёзд, слегка менять их положение случайным образом, дублировать каждый рукав, немного сдвигать его назад и делать его звёзды чуть более тусклыми.

galaxy_simulator.py, часть 5

```
# Set up display canvas.
❶ def spirals(b, r, rot_fac, fuz_fac, arm):
    """
    Build spiral arms for tkiner display using logarithmic spiral formula.
    :param b: arbitrary constant in logarithmic spiral equation
    :param r: scaled galactic disc radius
    :param rot_fac: rotation factor
    :param fuz_fac: random shift in star position in arm, applied to 'fuzz' variable
    :param arm: spiral arm (0 = main arm, 1 = trailing stars)
    :return: Picture with arms of the Milky Way galaxy.
    """
    ❷ spiral_stars = []
    ❸ fuzz = int(0.030 * abs(r))  # randomly shift star location
    theta_max_degrees = 520
    ❹ for i in range(theta_max_degrees):  # range(0, 600, 2) for no black hole
        theta = math.radians(i)
        x = r * math.exp(b * theta) * math.cos(theta + math.pi * rot_fac) \
            + randint(-fuzz, fuzz) * fuz_fac
        y = r * math.exp(b * theta) * math.sin(theta + math.pi * rot_fac) \
            + randint(-fuzz, fuzz) * fuz_fac
        spiral_stars.append((x, y))
    ❺ for x, y in spiral_stars:
        ❻ if arm == 0 and int(x % 2) == 0:
            c.create_oval(x - 2, y - 2, x + 2, y + 2, fill='white', outline='')
        elif arm == 0 and int(x % 2) != 0:
            c.create_oval(x - 1, y - 1, x + 1, y + 1, fill='white', outline='')
        ❼ elif arm == 1:
            c.create_oval(x, y, x, y, fill='white', outline='')
```

Фрагмент кода 10-7: Функция, строящая рукава Галактики.

Создадим функцию под названием «спирали» – `spirals()` ❶. Её параметры описаны в строках документации. Первые два параметра: b и r, – из логарифмического уравнения спирали. Параметр `rot_fac` – это коэффициент вращения. Он позволяет нам вращать рукав относительно центра и создавать новые рукава. «Коэффициент размытия» – `fuz_fac` – показывает, насколько нужно отодвинуть звёзды от центра рукава. Параметр «рукав» – `arm` – позволяет выбрать действия с главным рукавом или второстепенным рукавом, на котором будут изображены более тусклые звёзды. Второстепенный рукав будет сдвинут – нарисован позади главного рукава – и его звёзды будут меньше.

Создадим пустой список, в котором будут храниться координаты звёзд, из которых будет состоять рукав ❷. Создадим переменную, в которой мы будем умножать случайное значение на модуль радиуса галактического диска ❸. Уравнение спирали само по себе создаёт звёзды, выстроенные в линию (посмотрите на первые два изображения на Рисунке 10-11). Размытие немного сдвинет часть звёзд из спирали влево и часть вправо. Результат можно посмотреть на третьем изображении на Рисунке 10-11. Я нашёл эти значения методом проб и ошибок. Вы также можете поэкспериментировать с числами.

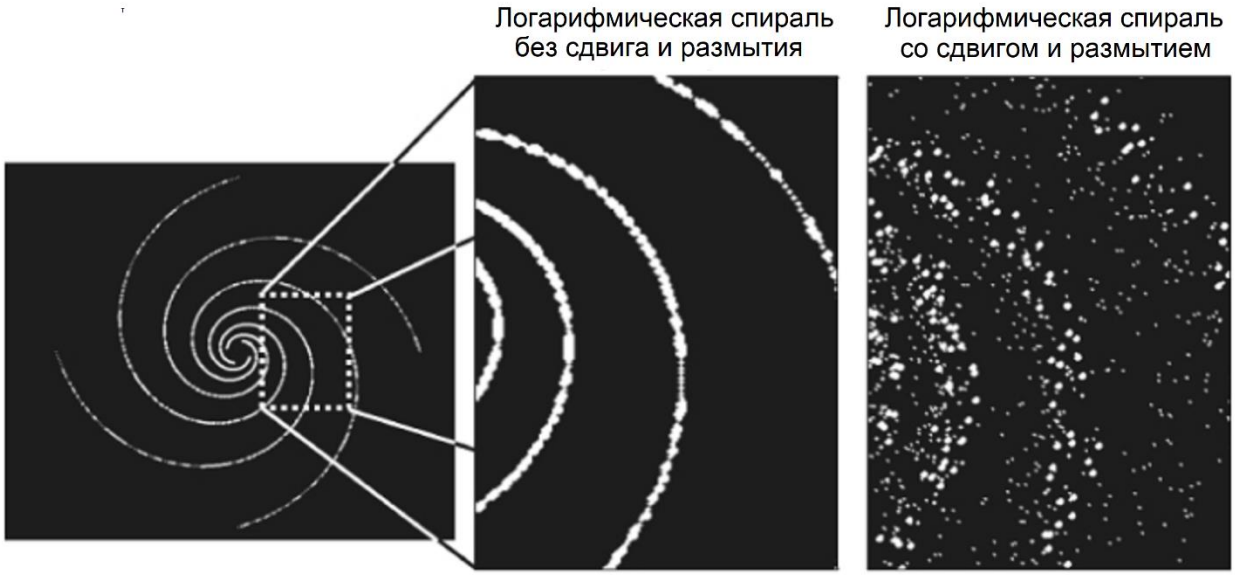

Рисунок 10-11: Наполняем пространство между рукавами: сдвигаем спирали и случайным образом меняем расположение звёзд

Теперь мы будем строить рукава. Сначала мы будем использовать диапазон значений, который будет представлять собой θ (тета) в логарифмическом уравнении спирали ❹. В диапазоне от 0 до 520 получается Галактика, как на Рисунке 10-9, в центре которой находится чёрная дыра. Можно использовать диапазон (0, 600, 2), и тогда в центре вашей Галактики будет находится плотное скопление звёзд (Рисунок 10-12). Поэкспериментируйте с этими значениями, пока не получите понравившийся вам результат. С помощью цикла `for` пройдёмся по значениям в θ (тета) и применим логарифмическое уравнение спирали. На оси `0x` будет отображено значение косинуса, а на оси `0y` – значение синуса. Добавим в конец списка `spiral_stars` полученные значения координат.

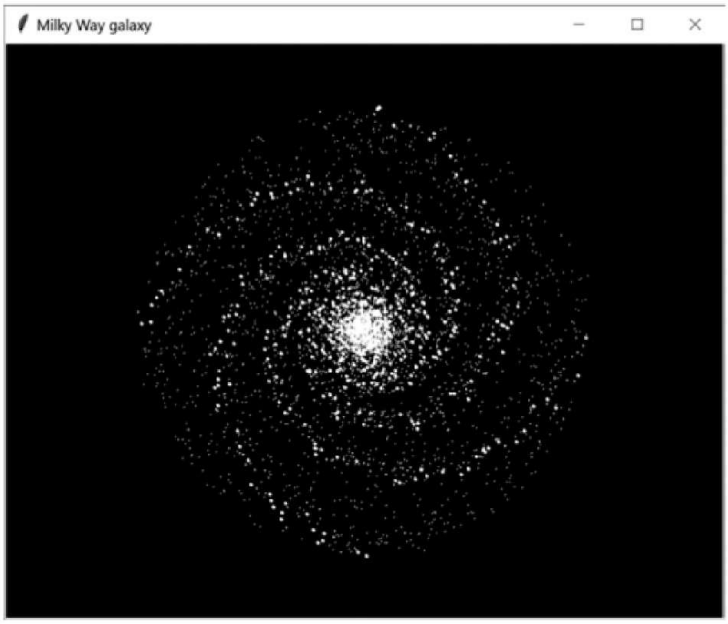

Рисунок 10-12: Модель Галактики без черной дыры в центре (сравните с Рисунком 10-9).

В функции `main()` мы определим переменную «коэффициент поворота» – `rot_fac`, которая будет поворачивать спираль относительно центра. После того как программа построит четыре главных рукава, она при помощи `rot_fac` построит четыре новых рукава, которые будут слегка смещены относительно первых четырёх. Второстепенные звёзды создадут «дымку», расположившись слева от каждой дуги ярких звёзд. (См. Рисунок 10-11).

Теперь с помощью цикла `for` пройдёмся по получившемуся списку с координатами звёзд ❺. С помощью условного оператора выберем главный рукав и координаты, в которых `x` – чётное число, т.е. делится на 2 без остатка: `int(x % 2) == 0` ❻. Для создания звёзд воспользуемся виджетом для холста под названием `create_oval`. Первые четыре аргумента отвечают за прямоугольник, в который будет вписан овал. Чем больше число после `x` и `y`, тем больше овал. Выберите белый цвет для заливки и уберите контур, для этого контур нужно приравнять пустоте: `outline=''`. По умолчанию контур представляет собой тонкую чёрную линию.

Если `x` – нечётное число, т.е. не делится на два без остатка: `int(x % 2) != 0`, то сделаем шаг для звезды поменьше. Если рукав имеет значение 1, то звезда находится во второстепенном рукаве. Поэтому мы её сделаем максимально маленькой ❼.

Примечание

Звёзды используются исключительно для визуального эффекта. Ни их размер, ни их количество не соответствует реальному масштабу. В действительности они будут гораздо меньшего размера и в существенно большем количестве: более 100 миллиардов.

Создаём звёздную дымку

В пространстве между рукавами Галактики также находятся звёзды, поэтому следующая функция (Фрагмент кода 10-8) будет случайным образом разбрасывать точки по нашей модели независимо от расположения рукавов. Она создаст сияние подобно тому, которое мы наблюдаем на снимках далёких галактик.

galaxy_simulator.py, часть 6

```
❶ def star_haze(disc_radius_scaled, density):
    """
    Randomly distribute faint tkinter stars in galactic disc.
    :param disc_radius_scaled:
    :param density:
    :return: a light haze in the picture )) _^_^_ ((
    """
    ❷ for i in range(0, disc_radius_scaled * density):
        ❸ x, y = random_polar_coordinates(disc_radius_scaled)
        ❹ c.create_text(x, y, fill='white', font=('Helvetica', '7'), text='.')
```

Фрагмент кода 10-8: Создаём функцию «звёздная дымка»

Создадим функцию «звёздная дымка» – `star_haze()`, которая будет принимать два аргумента: радиус диска и множитель (целое число), который функция будет использовать для увеличения количества случайных звёзд ❶. Поэтому если вам нравится густой туман, а не лёгкая дымка, то увеличьте значение плотности при вызове функции в файле `main()`.

Запускаем цикл `for`, в котором максимальное значение диапазона равно радиусу галактического диска, умноженному на плотность – `density` ❷. Значение радиуса нужно для того, чтобы количество звёзд было пропорционального галактическому диску на графике. Затем вызываем функцию `random_polar_coordinates()`, чтобы получить значения координат ❸.

В завершении создадим объект canvas, используя случайные координаты ❹. Так как мы использовали самый маленький размер овала для звёзд, размещённых на рукаве и вдоль него, то вместо метода `create_oval()`, мы используем метод `create_text()`. Благодаря этому методу мы можем представить звезду в виде точки.

Вы можете менять размеры шрифта и сделать вашу звёздную дымку максимально красивой.

На Рисунке 10-13 представлена модель Галактики без звёздной дымки (слева) и модель Галактики со звёздной дымкой (справа).

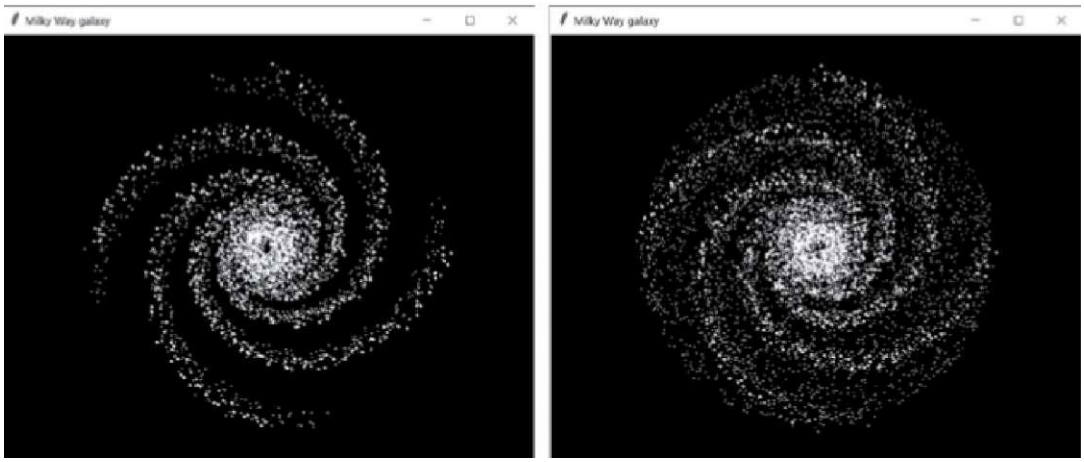

Рисунок 10-13: Модель Галактики без звёздной дымки (слева) и модель Галактики со звёздной дымкой (справа)

Вы можете поэкспериментировать с дымкой. Например, можно увеличить количество звёзд и раскрасить их в серый цвет, или при помощи цикла менять как размер, так и цвет. Только не красьте звёзды в зелёный цвет, во Вселенной таких звёзд нет!

Создаём функцию main()

Во Фрагменте кода 10-9 представлена функция `main()` из файла `galaxy_simulator.py`. Она будет создавать Галактику в заданном масштабе, определять вероятность обнаружения других цивилизаций, строить графическую модель и выводить на экран статистику. В ней также будет написан главный цикл для отображения окна программы библиотеки `tkinter`.

galaxy_simulator.py, часть 8

```
# Set up display canvas.
def main():
    """Calculate detection probability & post galaxy display & statistics."""
    ❶ disc_radius_scaled, disc_vol_scaled = scale_galaxy()
    detection_prob = detect_prob(disc_vol_scaled)

    # Build 4 main spiral arms & 4 trailing arms.
    ❷ spirals(b=-0.3, r=disc_radius_scaled, rot_fac=2, fuz_fac=1.5, arm=0)
    spirals(b=-0.3, r=disc_radius_scaled, rot_fac=1.91, fuz_fac=1.5, arm=1)
    spirals(b=-0.3, r=-disc_radius_scaled, rot_fac=2, fuz_fac=1.5, arm=0)
    spirals(b=-0.3, r=-disc_radius_scaled, rot_fac=-2.09, fuz_fac=1.5, arm=1)
    spirals(b=-0.3, r=-disc_radius_scaled, rot_fac=0.5, fuz_fac=1.5, arm=0)
    spirals(b=-0.3, r=-disc_radius_scaled, rot_fac=0.4, fuz_fac=1.5, arm=1)
    spirals(b=-0.3, r=-disc_radius_scaled, rot_fac=-0.5, fuz_fac=1.5, arm=0)
    spirals(b=-0.3, r=-disc_radius_scaled, rot_fac=-0.6, fuz_fac=1.5, arm=1)
    star_haze(disc_radius_scaled, density=8)

    # Display legend.
    ❸ c.create_text(-455, -360, fill='white', anchor='w',
                    text='One Pixel = {} LY'.format(SCALE))
    c.create_text(-455, -330, fill='white', anchor='w',
                    text='Radio Bubble Diameter = {} LY'.format(SCALE))
    c.create_text(-455, -300, fill='white', anchor='w',
                    text='Probability of detection for {:,} civilizations = {}'
                    .format(NUM_CIVS, detection_prob))

    # Post Earth's 225 LY diameter bubble and annotate.
    ❹ if SCALE == 225:
        ❺ c.create_rectangle(115, 75, 116, 76, fill='red', outline='')
        c.create_text(118, 72, fill='red', anchor='w', text="<------- Earth's Ra-
dio Bubble")

    # Run tkinter loop.
    ❻ root.mainloop()


❼ if __name__ == '__main__':
    main()
```

Фрагмент кода 10-9: Функция main().

Сначала вызываем функцию `scale_galaxy()`, которая будет определять объём и радиус галактического диска ❶. Затем вызываем функцию «определить вероятность» – `detect_prob()` – и передаём ей переменную `disc_vol_scaled`. Создаём переменную «вероятность обнаружения» – `detection_prob`. – и сохраняем в ней полученный результат.

Теперь построим графическую модель Галактики ❷. Несколько раз вызываем функцию «спирали» – `spirals()`, и каждый раз слегка её видоизменяем. Параметр `arm` отвечает за главные рукава и за более тусклые второстепенные рукава. Переменная «коэффициент поворота» – `rot_fac` – определяет, куда и насколько поворачивается рукав. Небольшое изменение в коэффициенте поворота между рукавами 0 и 1 (например, с 2 до 1.91) позволяет более тусклому рукаву немножко отойти от яркого рукава. В конце вызывается функция «звёздная дымка» – `star_haze()`, и получается полная модель. Опять же, вы можете поэкспериментировать с параметрами.

Теперь отобразим текст и статистику ❸: масштаб, размер зоны распространения радиосигналов, вероятность обнаружения другой цивилизации и количество цивилизаций. В качестве аргументов функция принимает координаты, цвет текста, выравнивание текста слева (`w` означает `"west"`) и сам текст. Используйте {:,}, чтобы вставить точку для разделения разрядов. Это часть нового метода форматирования, о котором вы можете прочитать, перейдя по ссылке: *https://docs.python.org/3/library/string.html#string-formatting*.

Если пользователь выбрал диаметр зоны распространения радиосигналов равный 225 световым годам ❹, то есть равный зоне распространения радиосигналов Земли, то мы разместим красный пиксель, показывающий приблизительное местоположение Солнечной системы, и подпишем его ❺. В `tkinter` есть несколько способов для того, чтобы отобразить один единственный пиксель. Мы воспользуемся методом `create_rectangle()`. Однако вы можете создать линию длиною в один пиксель, как показано ниже:

```
c.create_line(115, 75, 116, 76, fill='red')
```

Если использовать метод `create_rectangle()`, то первые два аргумента будут точки `(x0, y0)`, соответствующие верхнему левому углу, а значения `(x1, y1)` будут соответствовать местоположению одного пикселя прямо за нижним правым углом. Если использовать метод `create_line()`, то в качестве аргументов нужно передать начало и конец линии. По умолчанию длина линии составляет один пиксель.

В конце функции `main()` выполните функцию `tkinter mainloop()`, так называемый событийный цикл ❻. Благодаря ей окно программы будет отображено на экране, пока вы его не закроете.

В глобальном пространстве имён завершите программу и разрешите ей работать самостоятельно или в качестве импортированного модуля в другой программе ❼.

Конечный результат представлен на Рисунке 10-14, на котором видна зона распространения радиосигналов Земли и чёрная дыра в центре.

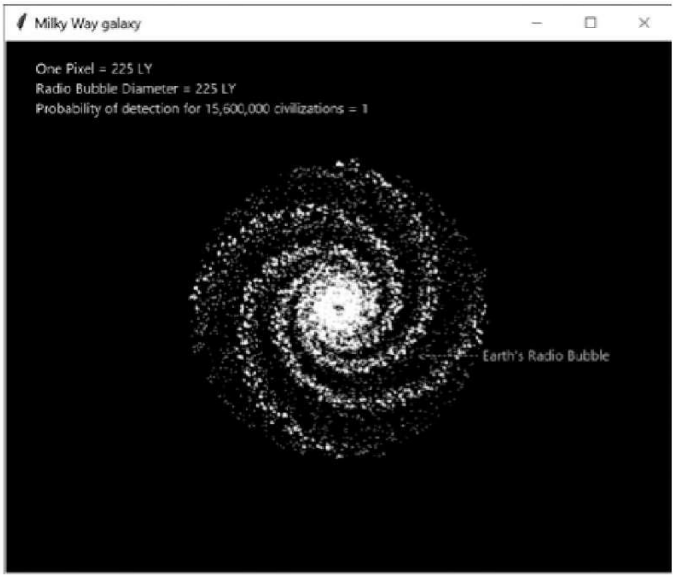

Рисунок 10-14: На модели изображена зона распространения радиосигналов Земли диаметром в 225 световых лет

На данном рисунке зона распространения радиоволн Земли имеет размер с кончик иглы. Однако если бы диапазон обнаружения других цивилизаций составлял 112,5 световых лет и если бы количество цивилизаций действительно совпадало с результатами вычисления Уравнений Дрейка, то вероятность обнаружения другой цивилизации равнялась бы 1 или 100%!

РЕЗУЛЬТАТЫ

Учитывая большую неопределённость данных на входе и ряд упрощений, мы стремимся в данном случае не к высокой точности. Нам важно найти правильное направление. Сможем ли мы или другая цивилизация нашего уровня развития обнаружить другую цивилизацию, которая преднамеренно не посылает сигналы в космос? Согласно Рисунку 10-15 скорее всего нет.

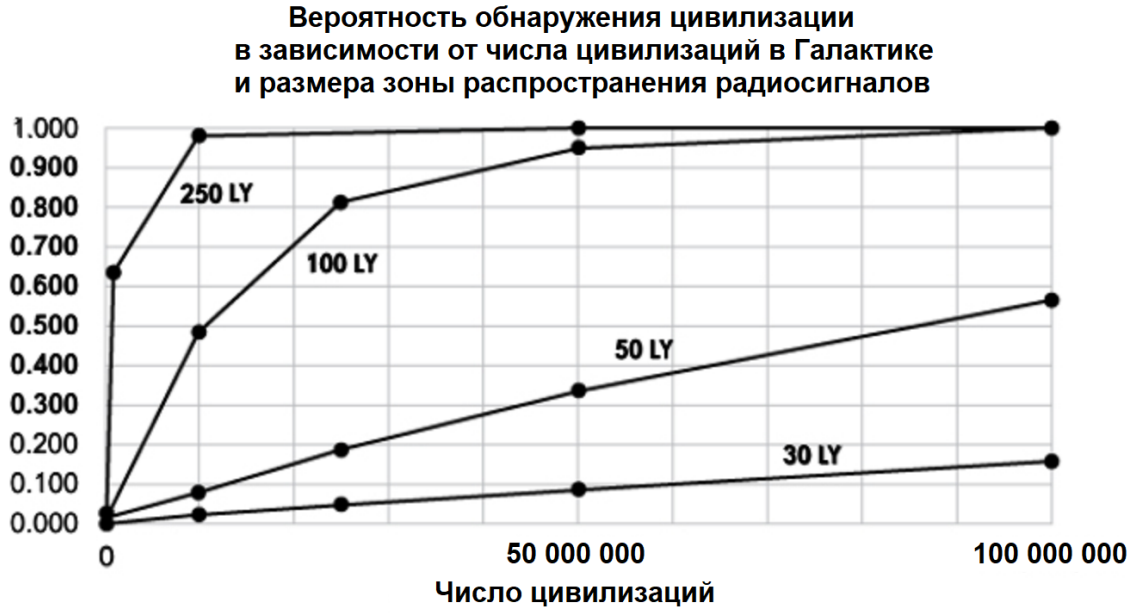

Рисунок 10-15: Вероятность обнаружения другой цивилизации в зависимости от разного диаметра зоны распространения радиоволн и разного числа цивилизаций в Галактике

При помощи современных технологий мы можем обнаружить радиосигналы на расстоянии до 16 световых лет, что эквивалентно зоне распространения радиосигналов с диаметром 32 световых года. Даже если в Галактике действительно проживает 15,6 млн. технологически развитых цивилизаций, как было подсчитано с помощью Уравнения Дрейка, вероятность обнаружения зоны распространения радиосигналов диаметром в 32 световых года составляет менее 4%!

Посмотрим ещё раз на Рисунок 10-14 и постараемся осознать необъятность и пустоту нашей Галактики. У астрономов есть даже специальное слово – Ланиакея, что переводится с гавайского языка как «необъятные небеса».

Наша Земля, как описал её Карл Саган, – это «пылинка, попавшая в солнечный луч». Недавние исследования показали, что окно возможностей для обнаружения цивилизаций, испускающих радиоволны, существенно меньше, чем мы считали. Если другие цивилизации последуют нашему примеру и перейдут на цифровые спутниковые сигналы, то поток радиосигналов уменьшится на четыре порядка. Мы все станем, сами того не желая, невидимыми, проживём около сотни лет в периоде рассвета, а затем исчезнем.

Учитывая все эти данные, не удивительно, что правительство больше не финансирует исследования по поиску внеземных форм жизни с помощью радиотелескопов. Сейчас усилия сдвинулись в сторону использования оптических методов, которые ищут определённую концентрацию газов в атмосферах экзопланет, свидетельствующих о процессах разложения биологических веществ и о промышленном производстве.

ПОДВЕДЕНИЕ ИТОГОВ

В данной главе мы впервые использовали библиотеки: `matplotlib` и `NumPy`. Мы создали многочлен для подсчёта вероятности обнаружения радиосигналов от других цивилизаций. Также мы использовали модуль `tkinter` для визуализации данных.

Перевод Зубарева Андрея
г. Краснодар, 21 сентября 2022