Учебное пособие для молодых специалистов

Полупроводниковые детекторы одиночных фотонов в ближнем инфракрасном спектре

Автор:
Инициалы: Козий Андрей Анатольевич
email: a.koziy98@gmail.com
github: /akoziy98

Москва, 2021

Оглавление

ГЛАВА 1

Введение

Данное учебное пособие ставит своей целью обучить молодых ученых для работы в области детекторов одиночных фотонов (ДОФ – single photon detector – SPD) в ближнем инфракрасном спектре. Более конкретно, в основном будут рассматриваться устройства ДОФ на основе InGaAs/InP однофотонных лавинных фотодиодах (ОЛФД – single photon avalanche diode – SPAD). Данный тип ОЛФД применяется в основном для детектирования фотонов с длиной волны $\lambda \in [1310, 1550]$ нм. А детекторы на их основе применяются в основном в устройствах квантового распределения ключа (КРК – quantum key distribution – QKD).

Данный учебник является интерактивным. По каждой главе будут приведены задания, которые необходимо выполнять в специальной тестирующей системе. Можете не волноваться, система тестирования и научные статьи будут локально скачаны на ваше устройство. Однако, в тексте присутствует много ссылок на интернет-статьи, особенно в разделе про Python. Как недостаток, проходить задания вы сможете только на устройствах ПК, поскольку сделать поддержку приложений на мобильных устройствах слишком трудоемко. Наполнение каждой главы будет достаточно скромным. Основной акцент будет ставиться на внешние источники, на которые в тексте будут приведены гиперссылки. Поэтому внимательно следите за всеми подсвечивающимися объектами – это может быть важно. Если ссылка ведет на какую-либо научную статью или статью в интернете, то гарантируется, что данная статья будет находиться на вашем локальном устройстве.

Мы можем с уверенностью сказать, что полное выполнение задач из данного учебника сделает вас желанным специалистом не только для нашей группы в компании QRATE, но и в целом на арене научных организаций. Но не спешите радоваться. Полное прохождение заданий из данной книги является трудоемким процессом.

Данный учебник состоит из следующих глав, которые необходимо будет освоить:

- Python – приведены основные особенности языка, без вникания в особые тонкости. Библиотеки numpy, pandas, scipy, matplotlib.
- Статистика – про методы статистической обработки данных, уравнения, гипотезы, и т.д.
- Детекторы – во многом теоретический раздел, в котором необходимо будет читать статьи, отвечать на вопросы по ним, и выполнять работы над реальными данными.
- Выпускное задание – я бы назвал это уровнем advanced как с точки зрения языка python, так и научной части. Но и награда велика – результат можно будет опубликовать в журнал Q3 - Q1.

Связь данных разделов друг с другом является нелинейной. То есть нельзя просто взять и прочитать все от первой страницы до последней, начиная с первого, и заканчивая последним разделом.

Вас будет мотать с одного раздела на другой, пока учебник не будет пройден полностью, и вы не выполните все задания. Но можете не волноваться, мы будем сообщать, на какую главу вам нужно будет переходить каждый раз, а в данной главе будет приведен общий маршрут изучения с основными вехами.

Надо поместить правила работы с тестирующей системой

Надо разместить маршрут изучения учебника

А теперь просим проследовать в главу Python, и обратиться к первому разделу 2.1.

ГЛАВА 2

Python

2.1 Введение

Перед тем, как начинать изучать питон, его, очевидно, необходимо правильно установить на свой компьютер. Начнем мы с установки Анаконды: Anaconda URL.

Размер загрузчика около 500 мб. Запускайте его, нажимайте последовательность: Next; I Agree; All Users + Next; в выборе папки лучше оставить все как есть по дефолту + Next; а вот теперь очень важный шаг – нужно дать доступ работать анаконде и питону из любой консоли, которую вы будете вызывать на компьютере. Поэтому ставим обе галочки в предложенном меню и нажимаем кнопку Install.

Поздравляю, теперь Python установлен на ваш компьютер. Теперь разберемся в среде разработки. Мы предлагаем установить PyCharm. Можем просто зайти на сайт PyCharm и установить версию Community. Но это можно сделать и через анаконду. Давайте по ней прогуляемся. Открываете приложение Anaconda на вашем компьютере. Если вы его не видите, то сделайте поиск в меню пуск по ключевому слову "anaconda" – ее ярлык имеет вид зеленого круга с выколотой серединой. В разделе Home выбирайте PyCharm Community и устанавливайте его – это бесплатно. В анаконде удобно смотреть установленные пакеты. Об этом будет более подробно объяснено в разделе про Import.

Теперь мы можем писать код со всеми удобствами в PyCharm: там есть поддержка правописания, подсказки, удобный дебагер. Однако недостаток подобных IDE в том, что нельзя на лету дополнять свой код, или выводить какие-то переменные после завершения работы программы. Необходимо дописывать print() или что-то в этом духе и заново запускать программу. А если выполнение вашего кода требует много времени, то считайте, что вы потратили время в пустую. Для решения данной проблему существует крайне удобный интерфейс, который называется jupyter-notebook. В нем мы пишем код в ячейках, после чего их запускаем. Переменные, объявленные в одной ячейке сохраняются, и мы можем их вызвать из другой ячейки. Это бывает крайне удобно, когда необходимо тестировать работу функций, которые вы видите впервые. Так получается гораздо быстрее, чем через полноценную IDE – PyCharm.

Сперва про запуск PyCharm. По дефолту данная программа открывает файлы с разрешением .py. Можно настроить, чтобы она открывала и юпитеровские ноутбуки, но лучше их не смешивать. Создайте в папке файл, который, например, будет иметь название `test.py` – внимание на разрешение. В Windows 7 с такой установкой разрешения могут быть проблемы – разрешение .py будет считаться как имя. В Windows 10 с этим точно нет проблем, файл корректно изменит свое разрешение. Скажите устройству, что будете открывать файлы данного типа через PyCharm. IDE должна выдать вам предложение по выбору интерпретатора – выбирайте Python 3.8. В целом все. Для теста, что все работает, вы можете набрать следующую команду в первой строчке: `print("Hello, PyCharm!")`. Запуск кон-

кретного кода также достаточно прост – найдите на верхней панели меню `Run`, выберите там иконку зеленого треугольника с названием `Run...`, и вам должно высветится предложение по поводу запускаемого файла. Выберете файл с названием `test`. Внизу располагается консоль вывода – там вы будете видеть все принты и различные выводимые ошибки. Теперь, ваш выбор файла с кодом сохранится, и вы можете его быстро запускать комбинацией клавиш `Shift + F10`. Если вам необходимо вызвать другой код, то повторите процедуру с выбором ячейки `Run` на верхней панели, а далее по уже пройденному алгоритму.

Замечательно, мы теперь умеем писать код и запускать его в PyCharm. Давайте теперь научимся работать в Jupyter-notebook. В той же папке откройте консоль. Это можно сделать, например, зажав клавишу `Shift` и кликнув правой кнопкой мыши на свободное пространство. Выберите графу `Открыть окно PowerShell здесь` – перед вами откроется консоль. В консоли вводим команду `jupyter-notebook.exe` и нажимаем enter. Можно не дописывать команду до конца, например можно ввести `jupyter-no` и нажать tab – название дополнится автоматически. Если вы установили анаконду правильно, то в браузере откроется окно jupyter. Чтобы создать новый файл, кликните на кнопку `new` в правом верхнем углу и выберите ячейку `Python 3`. Откроется новое окно, которое непосредственно и представляет юпитеровский ноутбук. Чтобы переименовать файл, нажмите на название `Untitled` и введите новое имя блокнота – например `test`. Данный файл сохранятся в папку с разрешением `.ipynb`. Теперь сделаем все то же самое, что и в пайчарме: в пустой ячейке напишем строчку `print("Hello, PyCharm!")` и нажмем комбинацию Shift + enter. Ячейка выведет нашу запись, а ниже добавится еще одна ячейка – пустая. Вы можете посмотреть хоткеи юпитерского ноутбука, если нажмете клавишу `H`, при этом отщелкнув текущую ячейку для записи.

С вводной частью можно закончить, теперь вы знаете примитивы интерфейса для разработки кода. Далее мы не будем к этому возвращаться, поэтому если что-то забыли – то обращайте внимание на данное введение.

Если вы чо-то не поняли про PyCharm, или хотите немного подробнее о нем узнать, перед тем как начнем, то обратите внимание на данный сайт: про Pycharm. Аналогично и с jupyter-notebook: про jupyter-notebook.

В данном разделе мы будем использовать задачи из следующей книги: Сборник задач по Python. Мы будем приводить условие задачи и ссылку на упражнение, а вы должны будете решить эту задачу. Ваше решение вы должны набирать в специально подготовленном питоновском файле, в котором будет задано название функции, формат входа и формат выхода. Файлы для заполнения решения вы можете найти в папке `HW/Python/task_2_*`, где `*` обозначает номер секции, в которой было получено данное задание. Например, если вы получили задание в секции `2.3`, то вам следует искать питоновский файл для заполнения решения в папке `HW/Python/task_2_3`. Решение вашей задачи будет проверяться тестирующей системой, исполнение которой можно вызвать через файл `checking.bat`.

Конечно же, в сборнике задач есть и решения на все поставленные задачи. И в питоновском файле, начинающегося со слов `test_` тоже. Но мы вас убедительно просим не пользоваться данными решениями, и выполнять задания самостоятельно. Вы ведь читаете данный учебник, чтобы получить знания, а не научиться копировать чужие решения. Поэтому еще раз убедительно вас просим выполнять задания добросовестно.

Также в качестве задач мы будем ссылаться на сайт codeforces, и просить вас решить задачи на данном сайте. Примеры задач вы можете посмотреть здесь: Задачи с codeforces.

С введением можно закончить, поэтому можете переходить к следующему разделу 2.2.

2.2 Простейшие операции

Пробегите по части 1 в учебнике Сборник задач по Python. Не стоит дословно все читать, просто посмотрите основные примеры кода. Думаю, этого будет достаточно. Оцените, понимаете ли вы следующие темы:

- сохранять в переменную результат операций с другими переменными
- округление чисел
- типы int и float
- простейшие функции из библиотеки `math`
- как оставлять комментарии
- конкатенация строк, вычисление длины строки

Если да, то переходим к заданиям, которые расположены в папке `HW/Python/task_2_2`. Если нет, то лучше просмотрите учебник еще раз. Но в целом можете начать решать задачи.

Перед тем, как отправляться в питоновский файл с решением, приведем здесь пример его структуры.

В тексте кода есть структуры типа `#TODO:`, ниже которых вам нужно выполнить поставленную задачу. В описании обычно явно указано, что от вас требуется. Пример такого кода представлен ниже:

```
1 """
2 TODO: which library we need to import to work with mathematical functions?
3 """
4 import
```

В данном примере вам нужно дописать в `import` слово `math`.

Предположим, вам дали задание: реализуйте функцию, которая должна принимать одно число, и возвращать его квадрат.

Здесь нам необходимо немного разорвать повествование, и буквально в двух словах ввести понятие функции, чтобы вы могли работать с тестирующей системой. Функция в Python имеет конструкцию `def function_name(argument1, argument2):`. Здесь `def` – это специальное слово, обозначающее, что вот сейчас будет задана функция. `function_name` – это имя функции. Называйте функции так, чтобы любой читающий ваш код человек смог примерно представить, что делает ваша функция. Не бойтесь длинных названий. Например, назвать функцию `get_square_number` хорошая идея, в то время как название `func` давать не следует. В заданиях вам уже будет задано имя функции, которую следует реализовать. Параметры `argument1` и `argument2`, которые нам следует передать в функцию. Это могут быть в целом любые объекты – строки, int, float, и более сложные объекты. Через двоеточие мы будем делать подсказку, какие типы данных ожидаются на вход. Также подсказки будут стоять и на то, какой тип данных ожидается на выходе, и обозначается через знак $->$. С учетом подсказок определение функции `function_name` можно записать как: `def function_name(argument1 : int, argument2 : str) -> int:`.

Если функция должна выдавать что-то в результате своего выполнения, то используйте ключевое слово `return` в теле функции. Если не должна ничего выдавать, то не используйте ключевое слово

Вызов функции можно реализовать следующим образом: `function_name(arg1, arg2)`. Если эта функция должна выдавать некоторое значение, то вам нужно еще присвоить некоторому элементу, например `a`, выход данной функции: `a = function_name(arg1, arg2)`.

Более подробно про функции вы можете прочитать вот здесь: про функции в Python.

Приведем пример задачи с квадратом числа.

```
1 def get_square_number(a : float) -> float:
2     # TODO: realise square of the a
```

Ожидаемая конструкция данной функции может быть следующей:

```
1 def get_square_number(a : float) -> float:
2     # TODO: realise square of the a
3     a_square = a * a
4     return a_square
```

Провести тестирование работы вашей функции вы можете даже не вызывая окно `main`, а просто просить вывести различные значения в свободных от функций месте. Главное – не использовать при этом отступы.

```
1 def get_square_number(a : float) -> float:
2     # TODO: realise square of the a
3     a_square = a * a
4     return a_square
5
6 b = get_square_number(10)
7 print(b)
```

И напоследок. Если вас просят, чтобы функция возвращала не одно значение, а несколько, то оборачивайте выводимые переменные в квадратные скобки – конструктор list (более подробно об этом позже). Например, вам надо вывести `a1, a2, a3`, вы должны написать `return [a1, a2, a3]`.

Ладно, еще одна фишка. Ключевое слово `pass`, которое вы видите в функциях в питоновском файле, это "заглушки", чтобы интерпретатор не ругался на пустые функции. Если вы начинаете оформлять очередную функцию, то вам нужно убрать данное ключевое слово.

Вот теперь вы точно готовы к выполнению первых ваших заданий.

P.S. Как только вы выполните данные задания, предлагаю отдохнуть от питона и заняться другими делами. Начнем вникать в однофотонный лавинный фотодиод (ОЛФД) 4.1.

Задача 1. Возведение числа в квадрат

Напишите функцию, которая принимает число – тип float, и выводит его квадрат. Округлите до двух знаков после запятой.

Имя функции: `get_square_number`.

Задача 2. Приветствие

Напишите функцию, которая принимает имя – тип str, а возвращает строку, в которой было бы выражено приветствие человека с данным именем. Например, если имя человека "Alice", то ожидаемая строка будет: "Hello Alice, glad to see you!".

Имя функции: `hello_function`.

Задача 3. Площадь комнаты

Напишите функцию, запрашивающую у пользователя длину и ширину комнаты – тип float. Функция должна возвращать площадь комнаты, округленную до одного знака после запятой.

Имя функции: `area_rectangular_room`.

Задача 4. Сложные проценты

Представьте, что вы открыли в банке сберегательный счет под a % годовых. Проценты банк рассчитывает в конце года и добавляет к сумме счета. Напишите программу, которая запрашивает у пользователя сумму первоначального депозита `value`, процент по вкладу `a`, после чего рассчитывает и возвращает сумму на счету в конце первого, второго и третьего годов. Все суммы должны быть округлены до двух знаков после запятой.

Имя функции: `compound_interest`.

Задача 5. Расстояние между точками на земле

Как известно, поверхность планеты Земля искривлена, и расстояние между точками, характеризующимися одинаковыми градусами по долготе, может быть разным в зависимости от широты. Таким образом, для вычисления расстояния между двумя точками на Земле одной лишь теоремой Пифагора не обойтись.

Допустим, $(t1, g1)$ и $(t2, g2)$ – координаты широты и долготы двух точек на поверхности Земли. Тогда расстояние в километрах между ними с учетом искривленности планеты можно найти по следующей формуле:

$$distance = 6371.01 \times arccos(sin(t1) \cdot sin(t2) + cos(t1) \cdot cos(t2) \cdot cos(g1 - g2))$$

Примечание. Число 6371.01 в этой формуле, конечно, было выбрано не случайно и представляет собой среднее значение радиуса Земли в километрах.

Напишите программу, в которой пользователь будет вводить координаты двух точек на Земле (широту и долготу) в градусах (не радианах!). На выходе мы должны получить расстояние между этими точками при следовании по кратчайшему пути по поверхности планеты. Результат округлите до двух цифр после запятой.

Подсказка. Тригонометрические функции в Python оперируют радианами. Таким образом, вам придется введенные пользователем величины из градусов перевести в радианы, прежде чем вычислять расстояние между точками. В модуле math есть для этого удобная функция.

Имя функции: `earth_distance_between_points`.

Задача 6. Свободное падение

Напишите программу для расчета скорости объекта во время его соприкосновения с землей. Пользователь должен задать высоту в метрах, с которой объект будет отпущен h. Начальная скорость объекта v. Если скорость направлена к земле – то $v > 0$, если от земли $v < 0$. Предположим, что ускорение свободного падения равно 9.8 м/с2. При данных параметрах можно вычислить время до падения объекта на землю. Формулу получите самостоятельно. Значение времени округлите до двух знаков после запятой.

Имя функции: `free_fall_time`.

Задача 7. Сумма цифр в числе

Разработайте программу, запрашивающую у пользователя целое четырехзначное число и подсчитывающую сумму составляющих его цифр. Например, если пользователь введет число 3141, программа должна вывести значение 9, потому что: $3 + 1 + 4 + 1 = 9$.

Имя функции: `sum_of_digits`.

Задача 8. Вчерашний хлеб

Пекарня продает хлеб по цене `cost` за буханку. Скидка на вчерашний хлеб составляет $a\%$. Количество приобретенных вчерашних буханок хлеба n. Функция должна возвращать три величины: обычная цена за буханку, цена со скидкой, общая стоимость приобретенного хлеба, если бы скидки не было и общая стоимость приобретенного хлеба с учетом скидки. Все величины округлите то целого.

Имя функции: `yesterday_bread`.

2.3 Конструкции if-else-for-while

ГЛАВА 3

Статистика

3.1 Введение

В ходе данного раздела вам придется обращаться к следующей книге: Практическая статистика для специалистов Data Science. В ней собрано не так уж и много теории, нет доказательств и сложных выводов. Только важные формулы и пример кода на языке R (что нам не нужно) и Python (а вот это то что надо!). Не обращайте внимание, что данная книга предназначена для аналитиков данных. Эти знания можно применить для обработки вообще любых данных.

Для полноценной работы с данной книгой по статистике вам необходимо клонировать ее репозиторий: ссылка на репозиторий. Там приведены данные для тестовых примеров.

Мы будем здесь приводить только те темы, которые точно вам понадобятся для обработки данных с физических устройств, для моделирования процессов и так далее. Поэтому в какой-то момент отойдем от данной книге по статистике и перейдем к ссылкам на различные интернет-источники и статьи.

Изложение будет выстроено таким образом, что предполагается, что вы уже что-то знаете в статистике. И не будет много времени посвящено разбору терминов и так далее. Мы сделаем сухую выжимку и дадим ссылки на источники.

3.2 Среднее, среднеквадратическое отклонение и корреляция

Данный раздел более подробно вы можете изучить по главе 1 учебника про статистику.

Оценки среднего

На практике, всегда используются две оценки – среднее арифметическое и медиана. Разберемся в чем разница. Среднее арифметическое $\overline{x}$ выборки X определяется как:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{3.1}$$

где x_i – это элемент выборки, n – число элементов выборки.

Медиана – это такое значение элемента $\overline{x}$, что левее $\overline{x}$ находится половина элементов выборки, и правее $\overline{x}$ находится половина элементов выборки. То есть это просто середина в отсортированном массиве элементов выборки. На математическом языке это можно записать через мощности множеств следующим образом:

$$\overline{x} : |X < \overline{x}| = |X \geq \overline{x}| \tag{3.2}$$

Преимущество среднего арифметического в том, что оно быстро считается (за $O(n)$), и используется как один из параметров большого числа распределений. Недостаток – среднее арифметическое слишком чувствительно к выбросам.

Преимущество медианы в том, что она не чувствительна к выбросам. Однако, она долго считается (за $O(n \log n)$), и используется для малого числа распределений.

Оценки вариабельности

Теперь перейдем к оценкам вариабельности – это параметр, который позволяет оценить, какой разброс значений имеет выборка.

Самое часто применимое – это стандартное отклонение, или как еще называют – среднеквадратическое отклонение. Пусть у нас есть все та же выборка X, для которой мы вычислили среднее арифметическое $\overline{x}$. Отклонением называется следующая величина: $\Delta_i = x_i - \overline{x}$. Как мы видим, отклонение Δ_i может быть как положительным, так и отрицательным числом. Уже исходя из названия величины (среднеквадратическое отклонение), можно понять, как его определить:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}} \tag{3.3}$$

Сразу в глаза бросается знаменатель $n-1$. Он позволяет сделать оценку стандартного отклонения несмещенной. За этим стоит следующая интуиция. Когда мы вычисляли среднее, то у нас степень свободы была равна n – то есть размер выборки X. Когда мы вычисляем стандартное отклонение, то мы пользуемся не только выборкой, но и средним, которое получили на основании выборки. Таким образом, мы как бы понизили степень свободы.

Заметьте, что стандартное отклонение – это не то же самое, что и среднеквадратическая ошибка MSE или RMSE. Данные параметры – это всего лишь метрики ошибки, но никак не параметры распределений. В их определении в знаменателе стоит n:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x - \overline{x})^2. \tag{3.4}$$

RMSE – это просто корень из MSE.

Другая частая оценка вариабельности – это межквартильный размах. Квантиль уровня m – это такое значение x, что левее этого значения лежит m-я часть выборки, а правее $1-m$-я часть. Медиана – это квантиль на уровне 0.5 – то есть значение x, которое делит выборку пополам. А квартиль – это

квантиль на уровне 0.25 и 0.75 (от слова кварта – четверть). Можно привести следующую формулу для определения квантиля уровня m:

$$x^m : |X < x^m| = m \cdot |X| \tag{3.5}$$

Преимущества и недостатки стандартного отклонения и межквартильного размаха аналогичны среднему арифметическому и медиане. Из основного – стандартное отклонение чувствительно к выбросам, а межквартильный размах нет.

Корреляция

Давайте представим, что теперь у нас есть не только выборка X, но и Y. Например, на устройство установлены два датчика, и мы снимаем информацию с них одновременно. Мы понимаем, что в каждый момент времени мы измеряем некоторое состояние нашего устройства. А при изменении состояния у нас будут как-то изменяться показатели датчиков. Если X будет увеличиваться, и вместе с ним Y будет увеличиваться тоже, то говорят, что случайные величины X и Y скоррелированы. А если Y бы уменьшался, то случайные величины антискоррелироавны. Если же поведение X и Y не зависит друг от друга, то есть случайны, то такие величины нескоррелированы. Давайте посмотрим на формулу коэффициента корреляции Пирсона r:

$$r = \frac{1}{(n-1)\sigma_x\sigma_y}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}). \tag{3.6}$$

Коэффициент корреляции Пирсона хорошо работает в предположении, что выборки X и Y могут быть линейно скоррелированы. Например, если бы они были зависимы нелинейно, то данный коэффициент корреляции работал бы некорректно. Добавим интуиции: посмотрите на член, который стоит в сумме. По сути, мы берем измерения с наших датчиков попарно – за один и тот же момент времени. Если у нас присутствует положительная корреляция, то для элементов выборки приблизительно $i < n/2$, произведение $(x_i - \overline{x})(y_i - \overline{y})$ сначала большое, а затем уменьшается приблизительно до нуля. Причем это произведение будет положительным. При дальнейшем повышении индекса элемента выборки $i > n/2$, произведение тоже будет положительным и будет возрастать от нуля. А поскольку мы суммируем произведение отклонений по X и Y, то следует разделить на $n-1$ (вспоминаем про степени свободы). За счет деления на стандартные отклонения мы как бы "нормируем"данный параметр, и приводим его в диапазон: $r \in [-1, 1]$.

Если же мы рассматриваем не только выборки X и Y, а еще какие-то, то мы будем говорить о матрице корреляций. Данная матрица получается путем рассмотрения попарных корреляций рассматриваемых выборок.

Задание

В папке `HW/Statistics/task_3_2` представлен файл юпитеровского ноутбука `covid_data_prepare.ipynb`. В данном ноутбуке представлен датасет по заболеваниям Covid-19 в разны странах. Определите для данного датасета:

1. среднее
2. медиану
3. стандартное отклонение
4. межквартильное расстояние
5. матрицу корреляции

Сделайте вывод о том, какие переменные скоррелированы и почему.

3.3 Распределение данных и выборок

Данный раздел более подробно вы можете изучить по главе 2 учебника про статистику.

Когда вы собираете любые данные, необходимо держать в голове, что они не полностью отражают реальность и реальные распределение, а всего лишь являются их отражением. Экспериментально реальный процесс мы можем полностью описать, только если проведем бесконечное число экспериментов. Тогда мы получим то, что называется генеральной совокупностью. Имея на руках генеральную совокупность, мы можем с уверенностью говорить, что мы понимаем, как работает тот или иной процесс.

Очевидно, что мы не можем позволить себе проводить бесконечное число измерений, и должны ограничиться только каким-то их конечным числом. Сделав конечное число измерений, мы получим то, что называется выборкой. Выборка – это часть генеральной совокупности. Собственно, большая часть статистики и работы с данными о том, как получить из выборки сведения по поводу генеральной совокупности.

Когда мы делаем выборку из генеральной совокупности, крайне важно держать в голове то, как конкретно мы это делаем. Наиболее репрезентативной выборка получается только тогда, когда мы составляем ее случайно. Так, измерения физической величины позволяет получить случайную выборку, потому что любое измерение – это по определению случайный процесс. В данном примере у нас нет иного выбора. Ситуация обстоит сложнее, когда мы сами можем выбрать, как составить выборку. Прекрасный пример – это социологические опросы. Можно делать соц опрос, отлавливая людей у метро и предлагая им пройти быстрый тест. Или ходя по домам и квартирам. Или рассылать в интернете. Вполне очевидно, если мы попробуем выполнить опрос с единым текстом, но по разным технологиям, то мы получим совершенно разные данные. С большой вероятностью, полученные выборки не будут являться репрезентативными, то есть не будут полностью отражать генеральную совокупность. Такие выборки будут сильно смещенными. Когда вы делаете выборку по определенным правилам, то всегда должны держать в голове, получится ли она смещенной, и какую коррекцию необходимо провести с данными, чтобы нивелировать это смещение.

Распределение, распределение статистики по выборкам, бутстрапирование

Анализ выборок позволяет получить различные статистики – медиану, среднее, квартили, стандартное отклонение и другие. Вопрос в том, будут ли данные статистики так же отражать статистики генеральной совокупности.

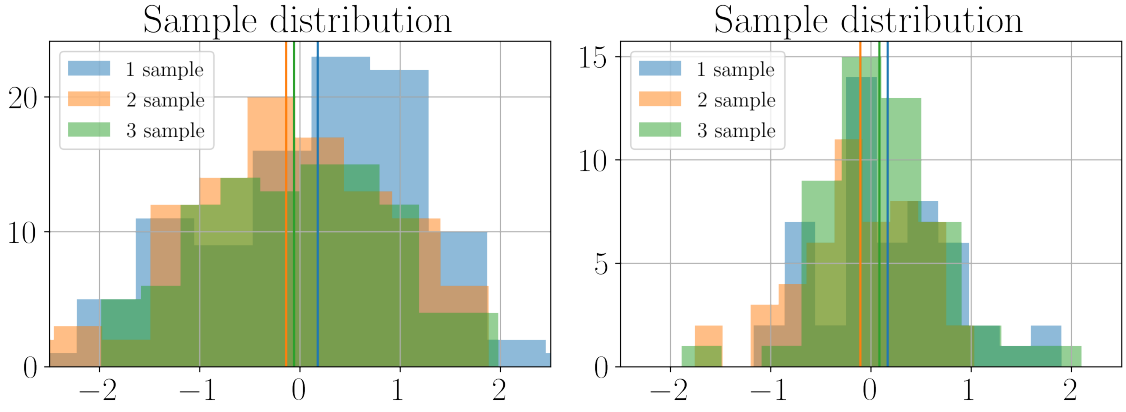

Рис. 3.1: Распределение оценки статистики по выборкам и распределение оценки средних по выборкам

Давайте проведем следующую процедуру. Пусть у нас есть генеральная совокупность G. И мы делаем из данной генеральной совокупности m штук выборок, в каждой из которых содержится n измерений. Будем работать на примере средних арифметических. Обозначим среднее арифметическое

выборки i как $\overline{x_i}$, стандартное отклонение σ_i, а истинное среднее арифметическое генеральной совокупности как μ.

Обратимся к левому рисунку 3.1, на котором представлены выборки для нормального распределения $N(0,1)$, с количеством элементов, равным 100. Линиями обозначены положения среднего арифметического данных выборок. Как мы видим, положения средних находятся не в точке $x = 0$ (т.к. $\mu = 0$), а немного расходятся.

Теперь давайте попробуем взять все наши m выборок, посчитать для каждой из них статистику, а после построить ее распределение. Чтобы было интереснее, давайте скажем, что $n = 2$ и $m = 50$. Тогда количество точек во всех выборках будет равняться 100, как в предыдущем тесте.

Как мы видим, гистограммы на правом рисунке 3.1 уже, но это не то, о чем вы подумали. На них представлено распределение не самого измерения, а среднего из двух измерений. Более показательно отклонение линий средних – они имеют примерно такие же отклонения от истинного среднего $\mu = 0$, как и в случае, когда мы брали просто 100 точек.

Мы можем сделать вывод, что важно лишь то, каким количеством разных измерений мы обладаем, а не подход к вычислению статистик. Чем может быть полезен подход с вычислением среднего по большому количеству выборок. Основное преимущество – сильная экономия памяти. Ведь для работы со статистикой в случае 100 точек, мы должны хранить все их значения. А если мы работаем с 50 выборками, то мы должны хранить всего 50 точек на каждую желаемую для определения статистику. Звучит так себе, но если бы мы выбрали размер выборки равным 20, и количество выборок 5, то ситуация была бы гораздо благоприятней.

Но на этом магия обработки оценок выборок не заканчивается. Крайне важная вещь в работе с выборками – это центральная предельная теорема. В ней говорится, что если мы возьмем любое распределение случайной величины (не обязательно нормальное), и сделаем из нее много выборок, то распределение статистик будет близко к нормальному распределению. Понятие "близко"означает, что это может быть распределение Стьюдента, или несимметричное нормальное распределение.

Центральная предельная теорема крайне полезна для обработки данных из распределений, которые "не сильно" отличаются от нормального. Тогда статистики из выборок будут практически совпадать с нормальным распределением. Чего нельзя сказать про "сильно" отличающиеся от нормального распределений. На рисунке 3.2 представлено экспоненциальное распределение из 10000 измерений и распределение средних по 1000 выборок, в каждой из которых по 10 измерений, а также распределение среднего, полученного по бутстрапированным выборкам.

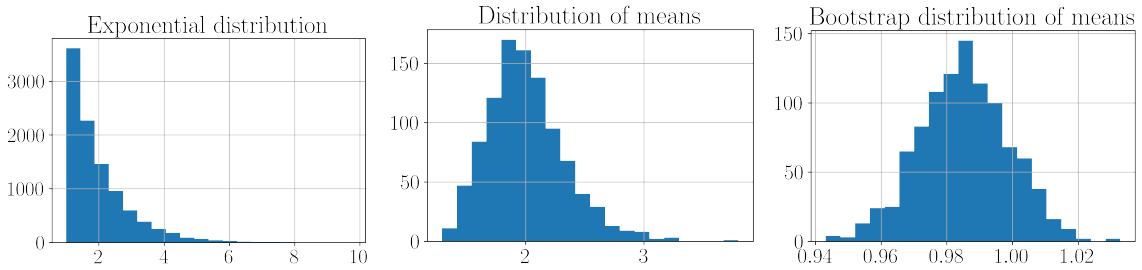

Рис. 3.2: Экспоненциальное распределение, распределение средних по большому числу выборок, и распределение средних по бутстрапированным выборкам.

Как мы видим на данных графиках, распределение средних по большому числу выборок перекошено в одну сторону. Если бы мы захотели оценить какую-нибудь метрику для среднего, например доверительные интервалы, это было бы не так тривиально.

К всеобщему благу, есть очень мощный метод для определений метрик различных статистик, в котором не нужно задумываться про распределение непосредственно статистик. Это метод бутстрапирования. Суть этого метода в том, что мы выбираем из единственной нашей выборки размера n, n случайных элементов с возвратом (то есть мы можем взять один элемент несколько раз) и считаем по нему интересующую нас статистику. Например среднее. После чего проделываем данную операцию k раз. В результате мы получаем k различных оценок на среднее, которые можем выразить в виде гистограммы. Преимущество данного подхода в том, что мы получаем распределение, которое очень

похоже на нормальное, что позволяет нам пользоваться аналитическими формулами для нормального распределения.

Бутстрапирование позволяет не задумываться над формой статистики, и разрешает применять универсальный метод обработки данных для выборок с любыми распределениями. Однако, метод бутстрапирования обладает существенным недостатком – оно требует больших вычислительных мощностей для применения данного алгоритма. Потому что, если мы имели выборку размером n и решили сделать из нее k бутстрапированных выборок, то нам потребуется в k раз больше операций для работы с ней.

Самостоятельно "потрогать" различные распределения и оценку статистик по выборкам вы можете на данном сайте.

ГЛАВА 4

Детекторы

4.1 Введение

Для начала следует разобраться в том, какие детекторы одиночных фотонов сейчас самые лучшие на рынке, какой у них механизм работы, в чем преимущество одного перед другими. Все это вы можете узнать из обзора: Обзор про SPAD и SNSPD (id: koziy_quant).

Поскольку данный учебник направлен на подготовку специалистов к работе с ДОФ на основе ОЛФД, то необходимо разобраться, какая электроника управляет этими диодами, в чем ее особенности, какие электрические схемы бывают. Все это вы можете узнать из обзора Обзор про управляющую электронику SPAD (id:koziy_microel).

После того, как вы разберетесь в этих статьях, вам необходимо пройти тесты по изученному материалу. Тестирующая система находится в папке `HW/SPD/task_4_SectionNumber_id`. Здесь `SectionNumber` – это номер раздела, в котором было выдано задание. Если задание выдано в разделе 4.1, то `SectionNumber = 1`. `id` – это идентификатор статья, которая была выдана на изучение. Параметр `id` указывается в тексте, когда выдается задание на прочтение статьи. В секции может быть выдано несколько статей, и по каждой из них предлагается пройти тестирование.

Далее, чтобы устроить полную смену деятельности, и узнать про каждое из направлений для изучения, давайте перейдем к главе статистики 3.1.

4.2 Моделирование InGaAs/InP ОЛФД

В данном разделе мы начнем изучать, какие процессы происходят внутри ОЛФД. Поскольку мы не можем так просто заглядывать в устройства микроэлектроники, то единственным механизмом познания остается моделирования. Суть следующая – вы составляете физическую модель устройства, проводите моделирование устройства при определенных параметрах, проводите эксперимент на реальных устройствах. Если результаты модели и реальных экспериментов совпадают, то это значит, что вы познали суть устройства, и полностью понимаете, как оно работает. Это открывает новые горизонты для улучшения текущей структуры устройства, или для создания абсолютно нового концепта устройства.

По данной причине моделирование ОЛФД является крайне важным инструментом, в котором надо разобраться. В данном разделе мы будем разбираться в двух достаточно знаковых работах, которые позволяют глубоко погрузиться в суть происходящих процессов. Заметьте, что в предложенных статьях речь идет именно про InGaAs/InP ОЛФД. ОЛФД с иной структурой будет рассмотрен в следующем разделе.

Первая статья: InGaAs/InP Single-Photon Avalanche Diode With Reduced Afterpulsing and Sharp Timing Response With 30 ps Tail (id: tosi_30ps).

Вторая статья: Design Criteria for InGaAs/InP Single-Photon Avalanche Diode (id: acerbi_design).

Не забудьте пройти тесты по данным статьям.

4.3 Моделирование InGaAs/InAlAs ОЛФД

Мы с вами уже изучили внутреннее строение устройства, которое называется front-illuminated InGaAs/InP ОЛФД. В настоящее время на рынке наиболее популярны устройства back-illuminated InGaAs/InP. Плюс ко всему идут разработки ОЛФД на основе материалов InGaAs/InAlAs. В данном разделе вам будет предложено три статьи про моделирование back-illuminated InGaAs/InAlAs ОЛФД.

Первая статья: Optimization of InGaAs/InAlAs Avalanche Photodiodes (id: chen_optimization).

Вторая статья: Theoretical Studies on InGaAs/InAlAs SAGCM Avalanche Photodiodes (id: cao_theor_studies).

Третья статья: Theoretical Analysis of InGaAs/InAlAs Single-Photon Avalanche Photodiodes (id: cao_theor_analysis).

Не забудьте пройти тесты по данным статьям.

ГЛАВА 5

Выпускное задание

5.1 Введение