Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Отчёт по лабораторной работе №12
Дисциплина: Телекоммуникационные технологии
Тема: GNU Radio

Выполнил студент гр. 3530901/80201 В.А. Пучкина

Преподаватель: Н.В. Богач

Санкт-Петербург
2021

Содержание

Список иллюстраций

1 Передача сигнала

Первый этап - передача сигнала QPSK. Генерируется поток битов и модулируется на сложное созвездие. Для этого используется блок Constellation Modulator, который использует Constellation Rect. Object и другие настройки для управления передаваемым сигналом.

Объект созвездия позволяет определить, как кодируются символы. Блок модулятора может использовать эту схему модуляции с дифференциальным кодированием или без него. Модулятор созвездия ожидает упакованные байты, поэтому используется генератор случайного источника, предоставляющий байты со значениями 0-255.

Чтобы обеспечить желаемую скорость передачи данных, количество выборок на символ должно быть минимальным. При моделировании это значение важно только для того, чтобы обеспечить соответствие этой скорости по всей потоковой диаграмме. Мы будем использовать количество выборок на символ равное 4. Это больше, чем нужно, но полезно для визуализации сигнала в различных областях.

Наконец, мы устанавливаем значение избыточной пропускной способности. Поточный график (`flowgraph`) mpsk_rrc_rolloff.grc (Рис.1) генерирует следующий рисунок (Рис.2), показывающий различные значения избыточной пропускной способности.

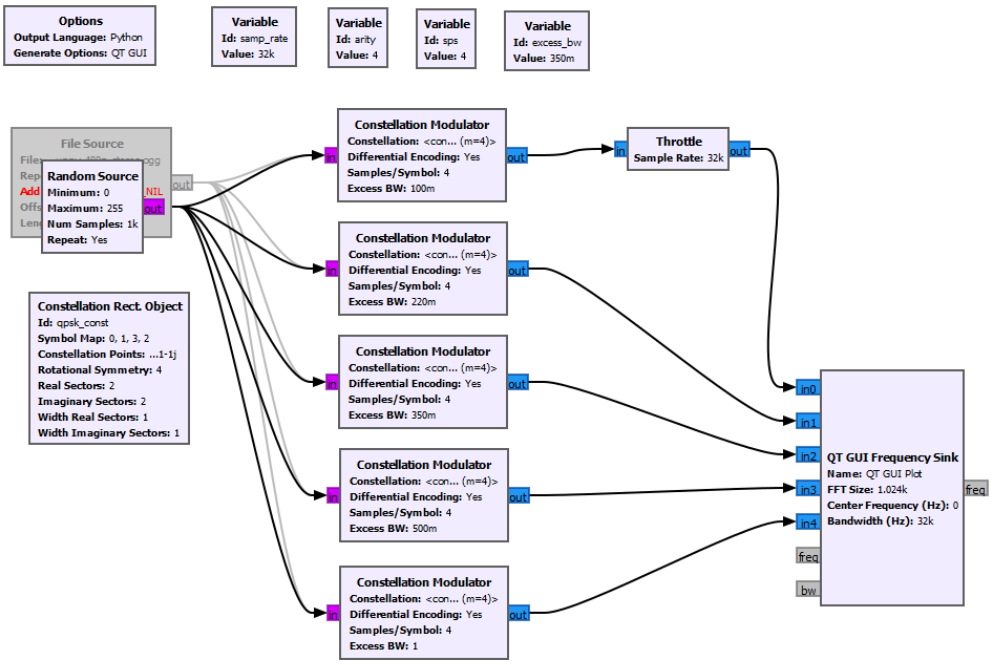

Рис. 1: Flowgraph mpsk_rrc_rolloff.grc.

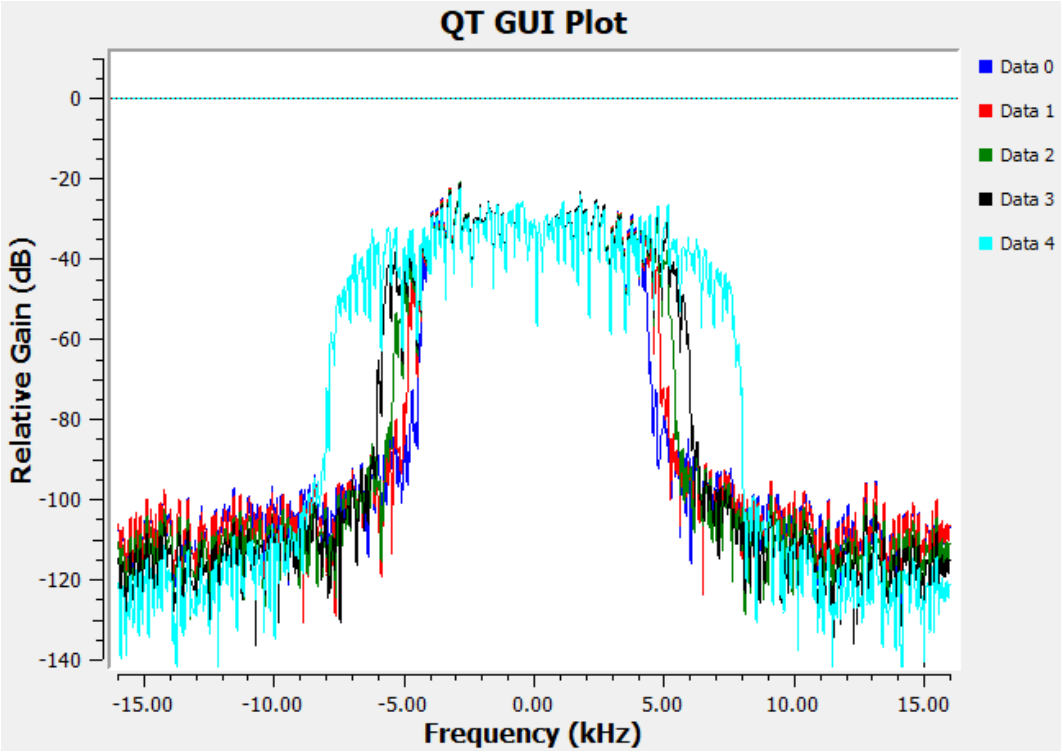

Рис. 2: Избыточная пропускная способность.

Типичные значения находятся в диапазоне от 0,2 (красная кривая) до 0,35 (зеленая кривая).

Теперь изучим файл `mpsk_stage1.grc`, передающий созвездие QPSK. Его `flowgraph` представлен на Рис.3.

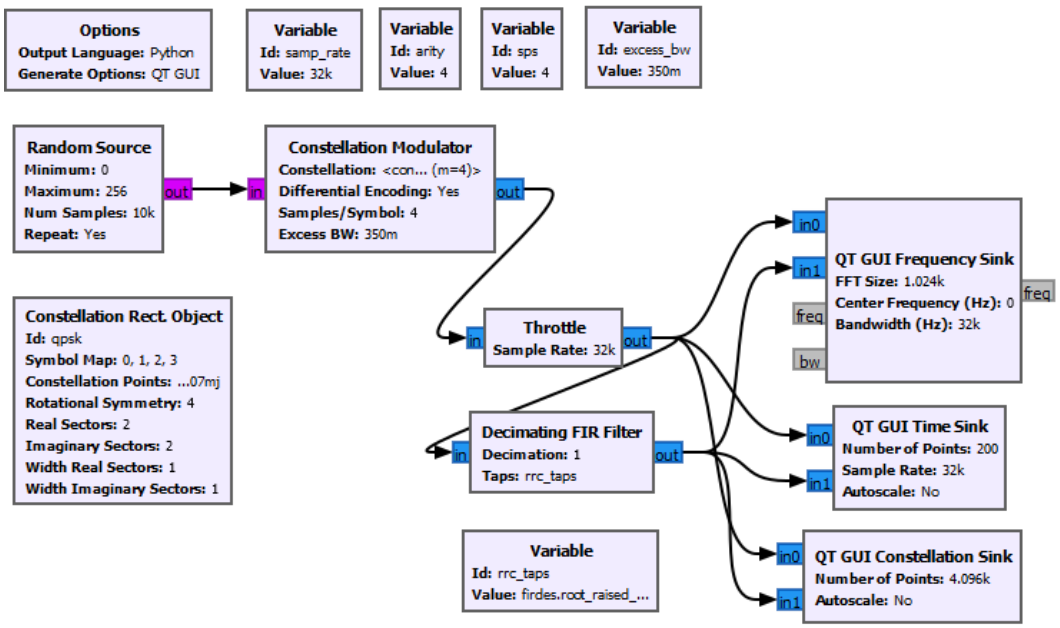

Рис. 3: Flowgraph mpsk_stage1.grc.

На графике созвездия можно увидеть эффекты повышающей дискретизации

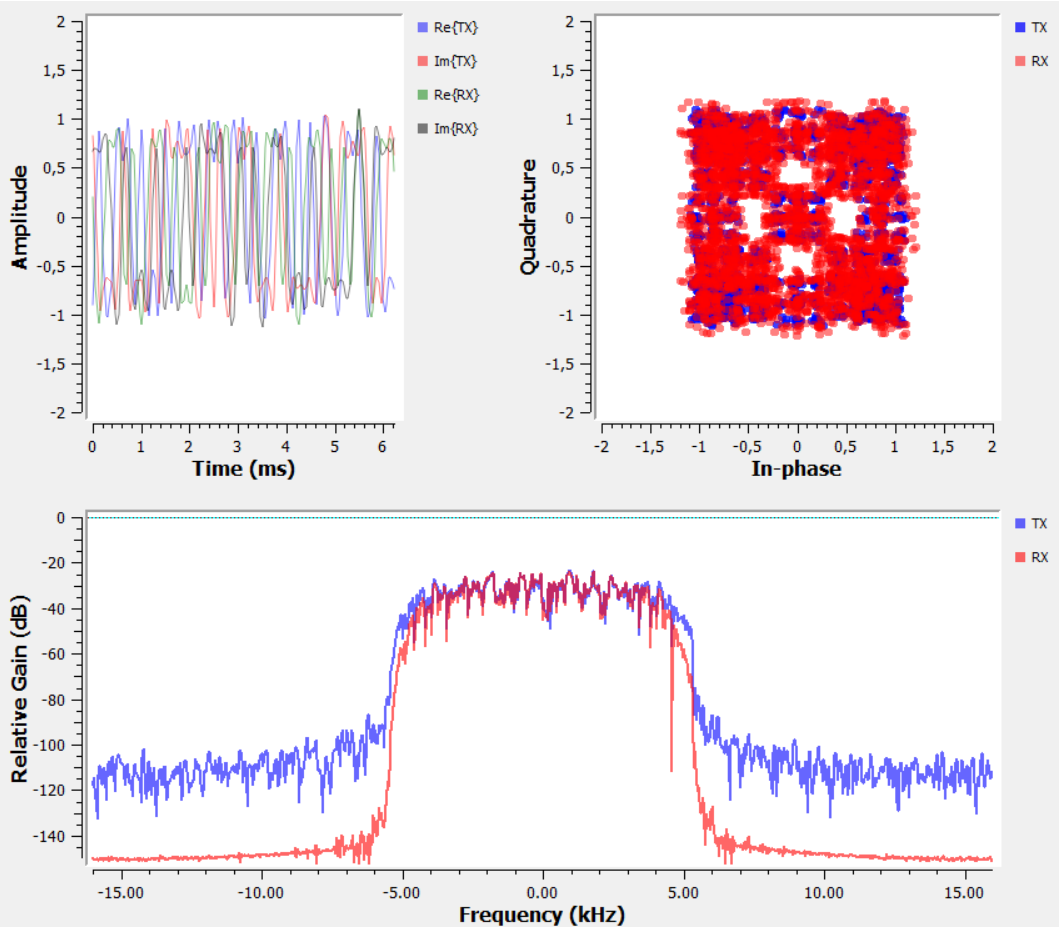

Рис. 4: Созвездие QPSK.

и процесс фильтрации. В этом случае фильтр RRC добавляет преднамеренные самоинтерференции (межсимвольные помехи, ISI). Это плохо влияет на принимаемый сигнал, потому что символы смешиваются.

Видно, что частотный график показывает сигнал красивой формы, переходящий в шум. Если бы мы не применили формирующий фильтр к сигналу, то передавали бы прямоугольные волны, которые производили бы много энергии в соседних каналах. Благодаря уменьшению внеполосных излучений теперь сигнал остается в пределах полосы пропускания канала.

На стороне приема мы избавляемся от ISI с помощью другого фильтра. Таким образом, происходит сворачивание двух фильтров RRC вместе. В результате получается фильтр с приподнятым косинусом, который является формой фильтра Найквиста.

Фильтрация является просто сверткой, поэтому выходной сигнал RRC-фильтра на приемной стороне представляет собой сигнал в форме приподнятого косинусоидального импульса с минимизированным ISI. Другое преимущество состоит в том, что при отсутствии эффектов канала мы используем согласованный фильтр на приемнике.

2 Добавление нарушений канала

Теперь рассмотрим влияние канала и то, как сигнал искажается между моментом передачи и моментом приёма. Изучим файл `mpsk_stage2.grc`. Будем использовать самый простой блок модели канала GNU Radio, позволяющий смоделировать несколько основных проблем,:

1. Шум. Тепловой шум в приемнике вызывает аддитивный белый гауссовский шум (AWGN). Мы устанавливаем мощность шума, регулируя значение напряжения шума модели канала. Можно рассчитать напряжение шума, исходя из желаемого уровня мощности, зная другие параметры моделирования.

2. Разные `clock`, которые определяют частоту радиомодулей. Грубо говоря, одно радио номинально работает на частоте `fc`. С учётом его недостатков, это означает, что фактически оно работает на частоте `fc+f_delta_1`. Другое радио аналогично будет иметь своё смещение - `f_delta_2`. Таким образом, полученный сигнал будет смещён на `f_delta_1 + f_delta_2`

3. Идеальная точка выборки. Эта проблема связана с предыдущей. Два радиомодуля работают с разной скоростью, а потому идеальная точка выборки неизвестна.

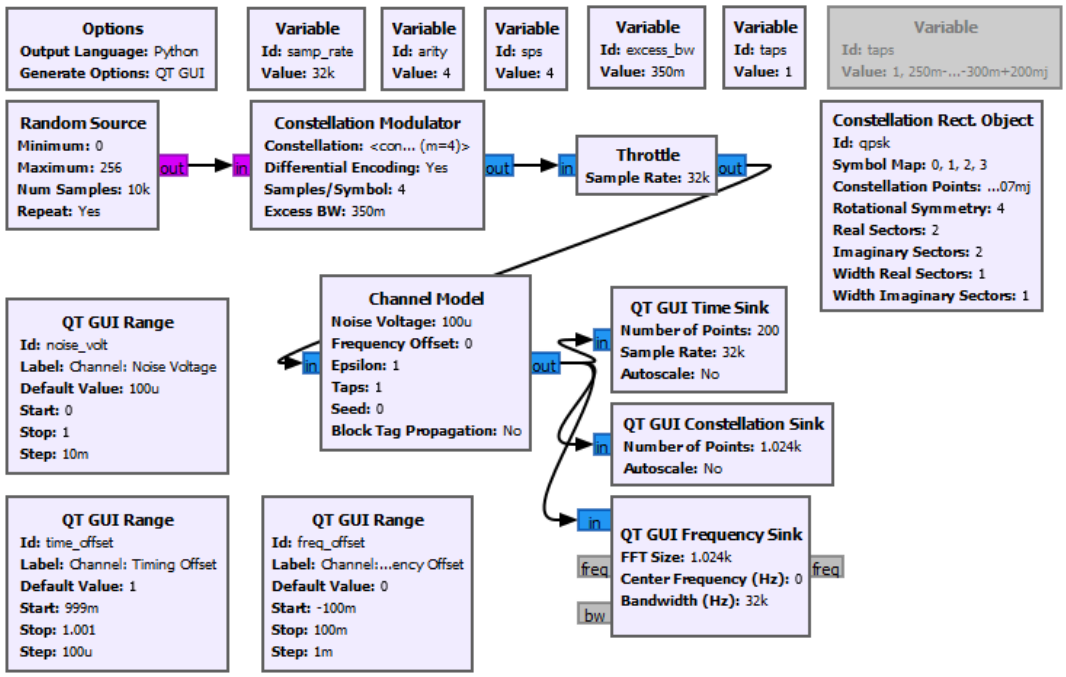

Рис. 5: Flowgraph mpsk_stage2.grc.

Запустим наш `flowgraph`. Мы можем поэкспериментировать с эффектами аддитивного шума, сдвига частоты и временного сдвига и посмотреть на результирующий сигнал.

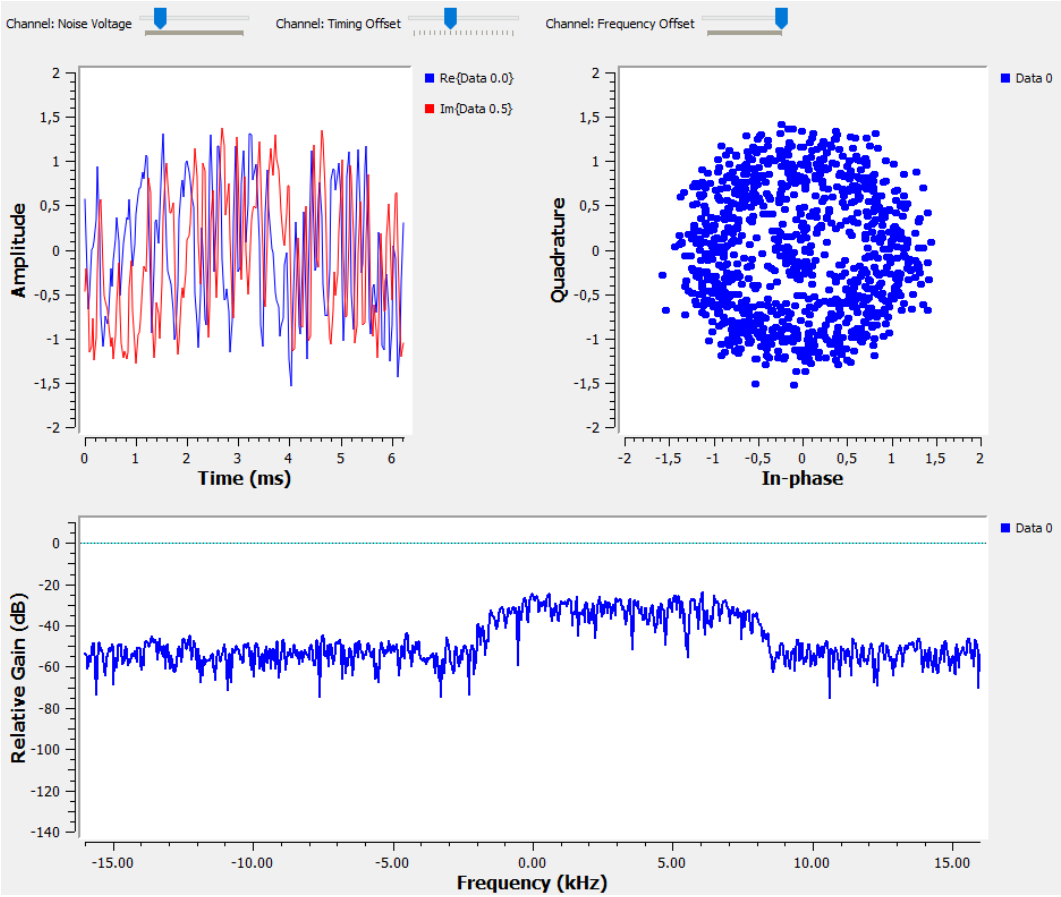

Рис. 6: Добавление нарушений канала.

График созвездия показывает нам облако образцов, намного хуже того, с чего мы начали на последнем этапе. Теперь, исходя из полученного сигнала, мы должны отменить все эти эффекты.

3 Восстановление timing

Теперь изучим процесс восстановления. Существует множество алгоритмов восстановления, однако мы будем использовать алгоритм восстановления многофазных `clock`. Попытаемся найти наилучшее время для дискретизации входящих сигналов, что позволит максимизировать отношение сигнал/шум (SNR) каждой выборки, а также уменьшить влияние межсимвольных помех (ISI).

3.1 Проблема ISI

Для иллюстрации проблемы ISI изучим файл `symbol_sampling.grc`. В нём создаётся четыре отдельных символа из единиц в строке, после чего они фильтруются.

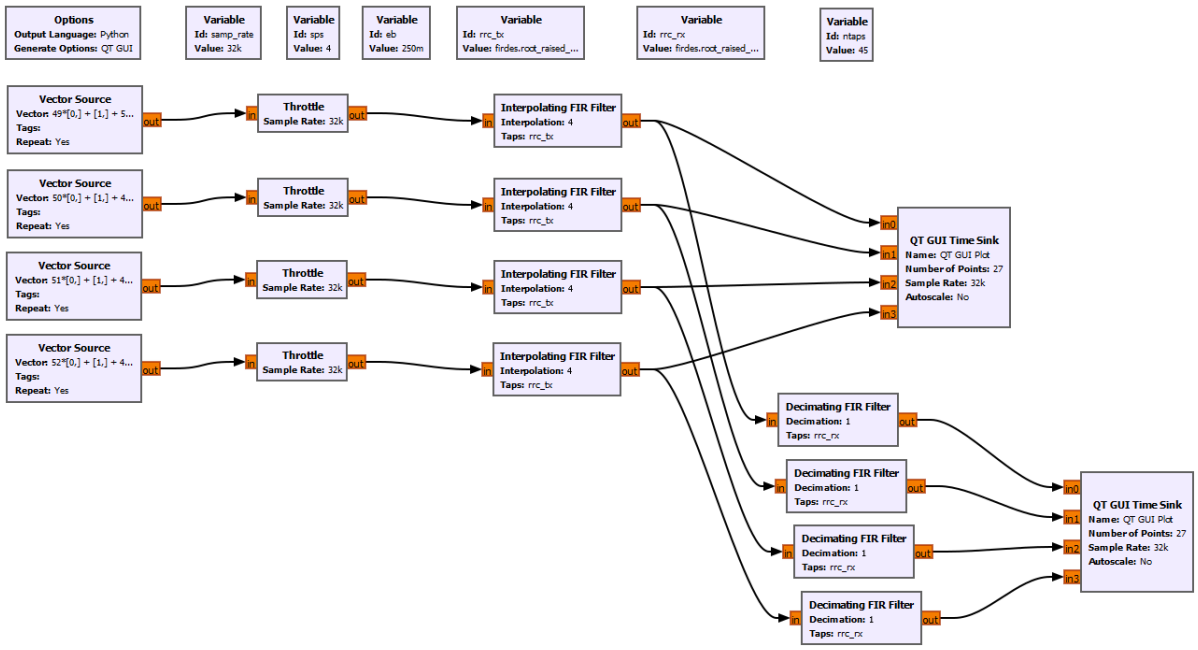

Рис. 7: Flowgraph symbol_sampling.grc

На Рис.8 можно увидеть различия между символами, отфильтрованными RRC и RC. Без фильтрации Найквиста в идеальной точке выборки каждого символа другие символы имеют некоторую энергию. Если мы сложим эти символы вместе, как в непрерывном потоке выборок, энергия этих других образцов складывается и искажает символ в этой точке. И наоборот, на выходе с RC-фильтром энергия от других отсчетов равна 0 в идеальной точке дискретизации для данного символа во времени. Это означает, что если мы делаем выборку в правильной точке выборки, мы получаем энергию только от текущего символа без помех от других символов в потоке.

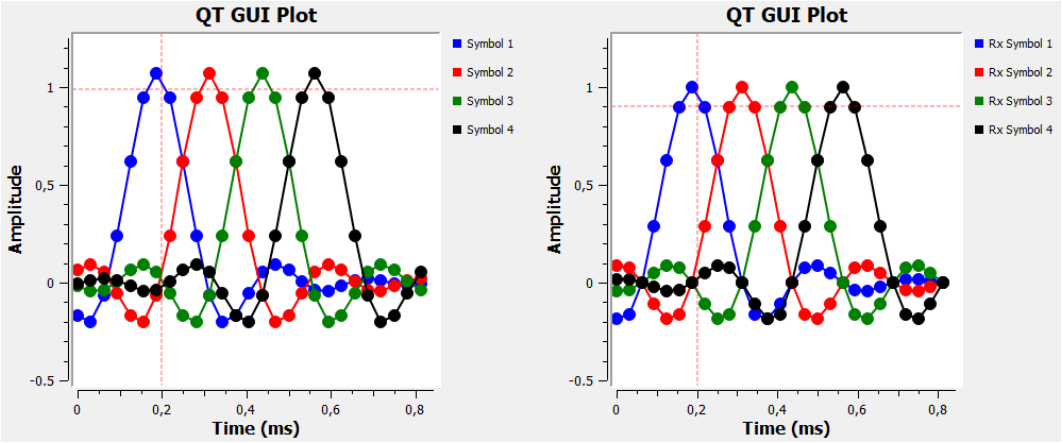

Рис. 8: Различия между символами, отфильтрованными RRC и RC.

Это моделирование позволяет легко настраивать количество выборок на символ, избыточную полосу пропускания фильтров RRC и количество ответвлений. Мы можем поэкспериментировать с этими различными значениями, чтобы увидеть, как они влияют на поведение точки выборки.

3.2 Проблема разных clock

Теперь рассмотрим влияние разных `clock` на точки выборки между передатчиком и приёмником. Для этого изучим файл `symbol_sampling_diff.grc`.

Все `clock` несовершенны, поэтому начинаются в разный момент времени и дрейфуют относительно других `clock`. Смоделируем это, добавляя `resampler`, который немного регулирует время выборки символов между передаваемым сигналом и приёмником.

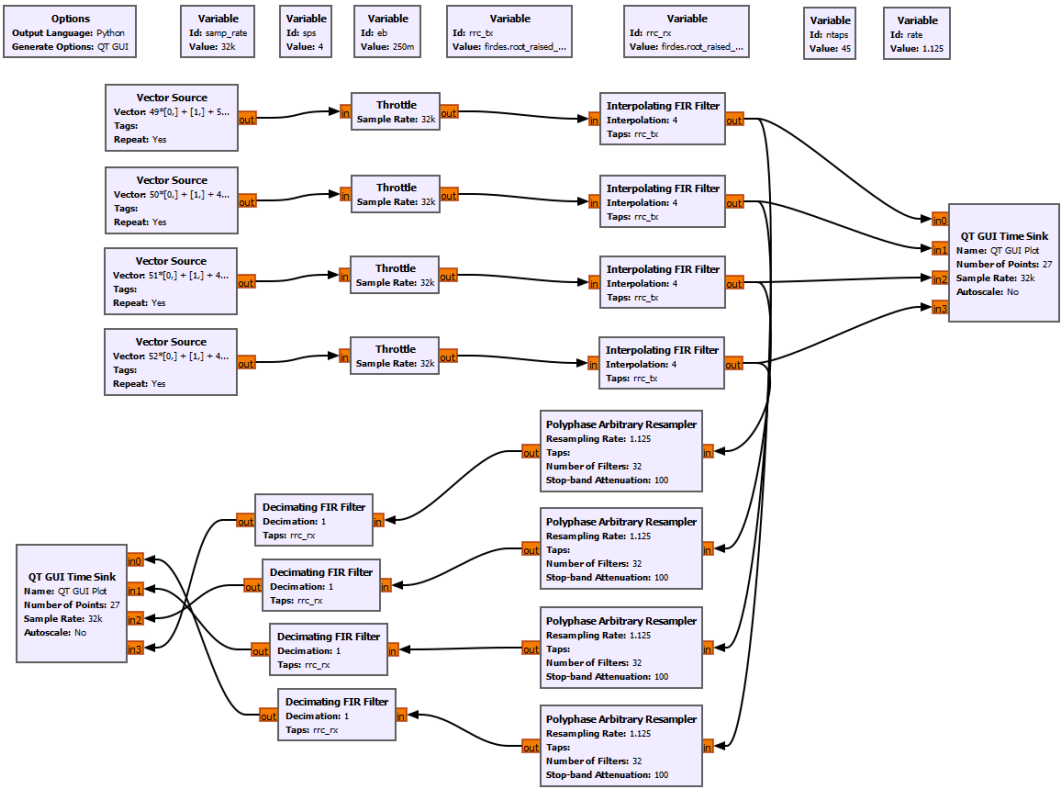

Рис. 9: Flowgraph symbol_sampling_diff.grc.

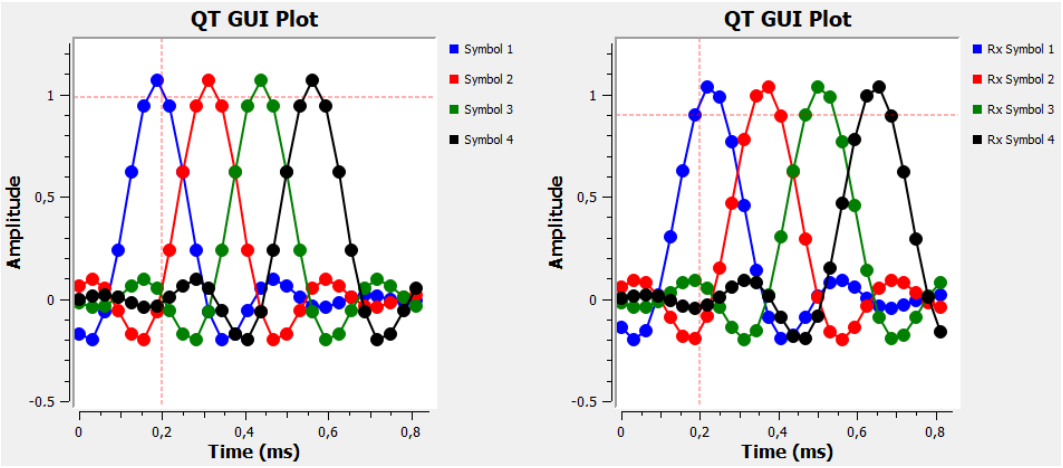

Рис. 10: Передача четырёх символов с рассинхронизацией.

Наша задача - синхронизировать clock передатчика и приёмника, используя только информацию в приёмнике из входящих выборок. Это задача восстановления clock или timing.

3.3 Блок синхронизации многофазных clock

Существуют различные алгоритмы для восстановления тактовой частоты на приёмнике. Мы будем использовать алгоритм восстановления `clock` многофазного блока фильтров.

Этот блок делает 3 важные вещи:

1. Восстанавливает тактовую частоту.
2. Согласует фильтр приёмника для устранения проблемы ISI.
3. Выполняет понижающую дискретизацию сигнала и производит выборки со скоростью 1 sps.

Итак, изучим файл `symbol_differential_filter.grc`. Нам нужен образец 0,22 мс. Фильтр разности ([-1, 0, 1]) генерирует дифференциал символа, и выходной сигнал этого фильтра в правильной точке выборки равен 0. Затем мы можем инвертировать этот оператор и вместо этого сказать, когда выход дифференциального фильтра равен 0, мы нашли оптимальную точку выборки.

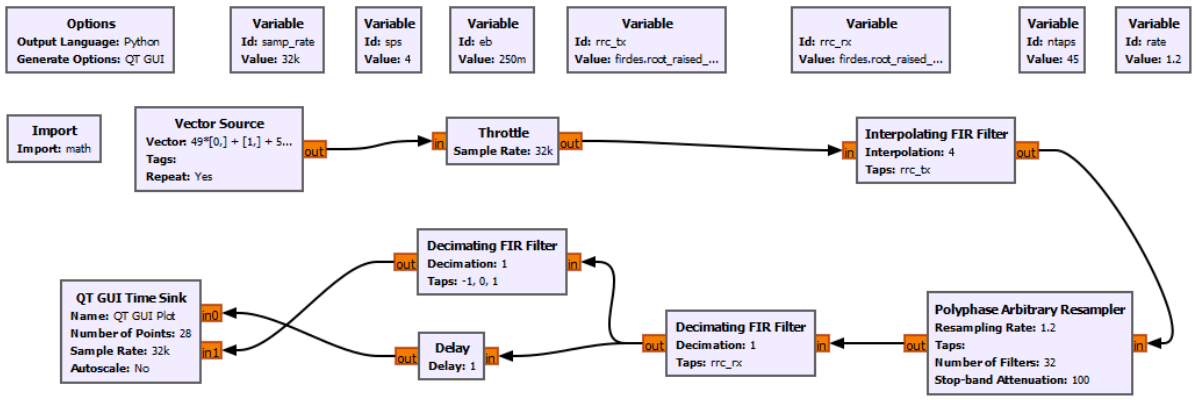

Рис. 11: Flowgraph symbol_differential_filter.grc.

Когда параметр скорости (`rate`) равен 1, смещения `clock` нет, а потому всё выглядит хорошо (Рис.12)

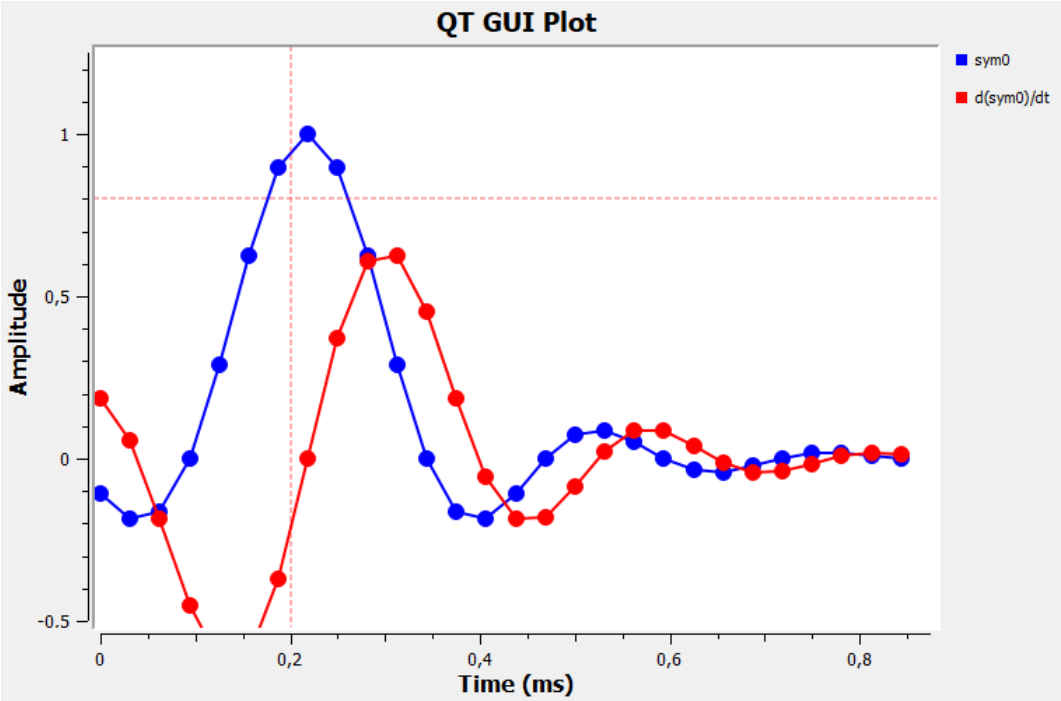

Рис. 12: Без смещения clock (rate = 1).

Теперь посмотрим, что происходит при смещении `timing`. Установим `rate = 1,2`. Результат можно увидеть на Рис.13.

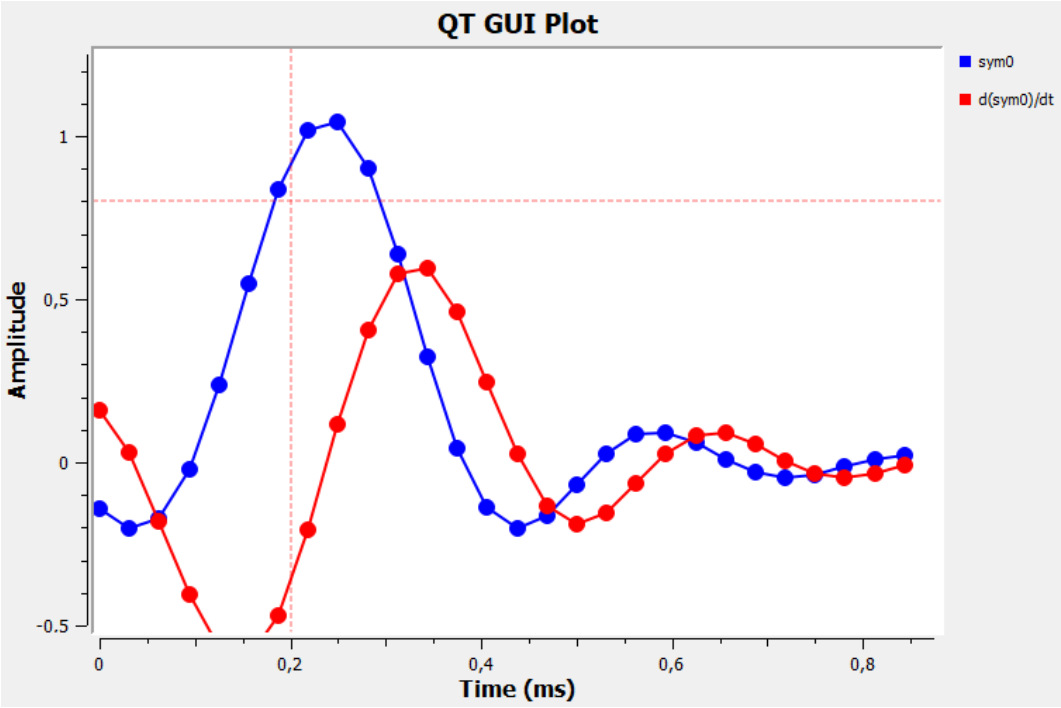

Рис. 13: Со смещением clock (rate = 1,2).

Можно заметить, что смещение `timing` происходит тогда, когда пик символа выключен, а производный фильтр не показывает нулевую точку.

Вместо использования одного фильтра можно создать серию фильтров, каждый с разной фазой. При достаточном количестве фильтров на разных фазах, один из них имеет правильную фазу фильтра, которая даст нам желаемое значение синхронизации.

Изучим файл `symbol_differential_filter_phases.grc`. У нас есть 5 фильтров, что означает 5 различных фаз.

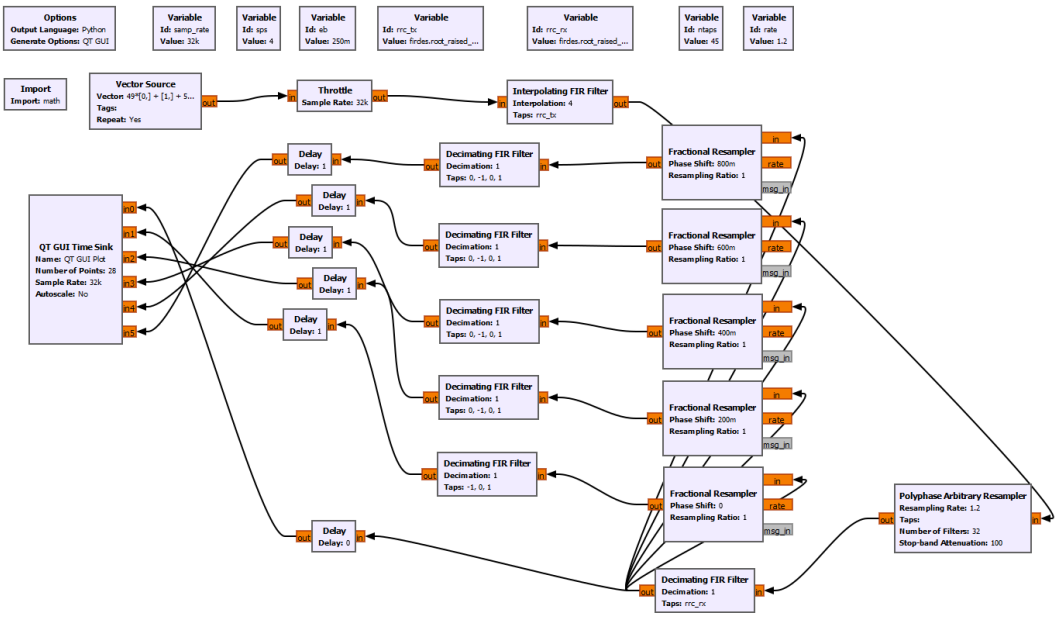

Рис. 14: Flowgraph symbol_differential_filter_phases.grc.

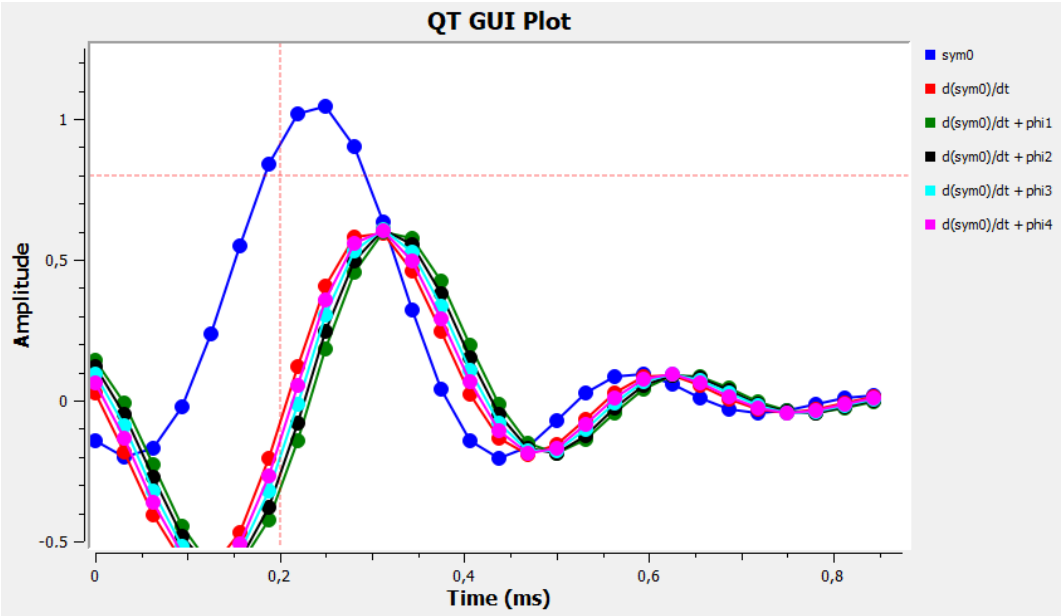

Рис. 15: Несколько фаз

На Рис.15 видно, что сигнал `d(sym0)/dt + phi3` имеет точку отсчета в 0. Это говорит о том, что наша идеальная точка дискретизации находится при этом фазовом сдвиге. Следовательно, если мы возьмем RRC-фильтр нашего приемника

и настроим его фазу на phi3 (который составляет $3 * 2pi/5$), то мы сможем исправить рассогласование по времени и выбрать идеальную точку дискретизации в это время дискретизации.

Итак, мы использовали один из пяти фильтров в качестве идеальной точки дискретизации. Однако 5 фильтров недостаточно. Любое смещение выборки между этими фазами по-прежнему приведет к несвоевременной выборке с добавленным ISI.

Давайте используем более 5 фильтров в нашем алгоритме восстановления тактовой частоты. Будем использовать 32 фильтра, чтобы получить максимальный коэффициент шума ISI, который меньше шума квантования 16-битного значения.

3.4 Использование блока многофазной синхронизации

Теперь используем блок многофазной синхронизации. Теперь давайте изучим файл `mpsk_stage3.grc`. `Flowgraph` принимает выходные данные модели канала и передает их через наш блок Polyphase Clock Sync. Этот блок настроен с 32 фильтрами и полосой пропускания петли $2pi/100$.

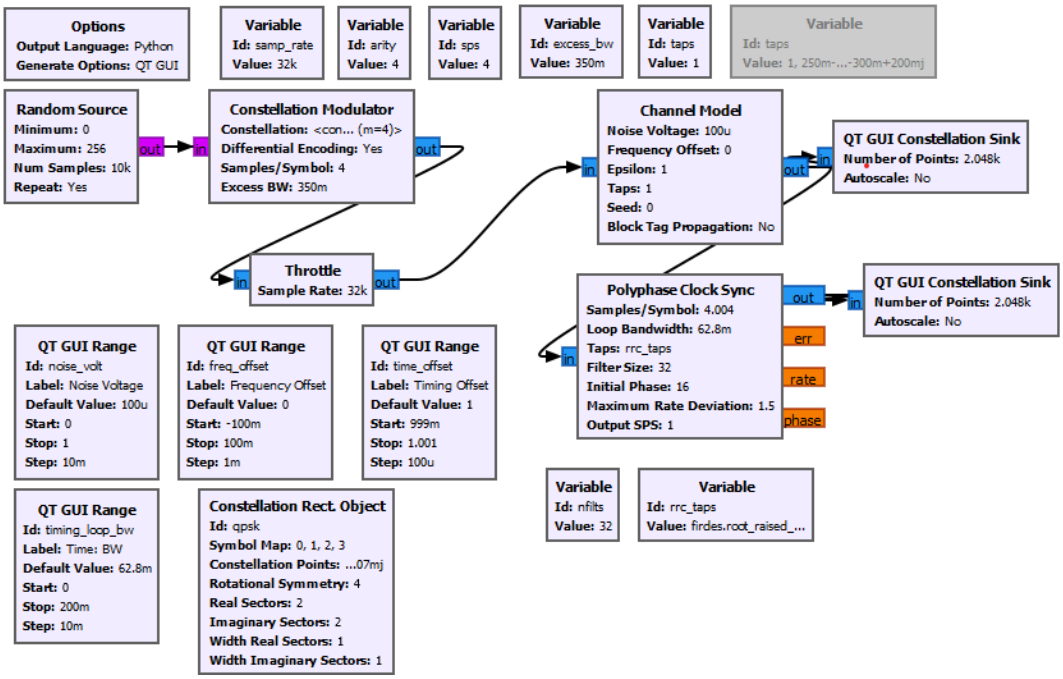

Рис. 16: Flowgraph mpsk_stage3.grc.

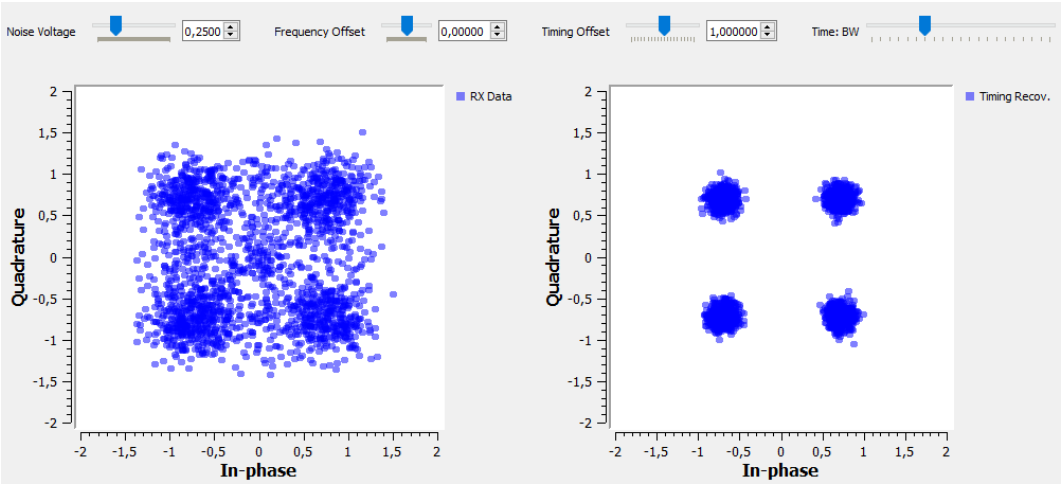

Рис. 17: Использование блока многофазной синхронизации.

На Рис.17 мы видим дав созвездия: слева - полученный сигнал до восстановления синхронизации; справа - после восстановления синхронизации. Это все еще немного шумно из-за ISI.

Мы также можем поэкспериментировать с изменением временного и частотного смещения.

4 Многолучевость

Многолучевость - следствие того, что в большинстве коммуникационных сред у нас нет единственного пути для прохождения сигнала от передатчика к приемнику. Каждый раз, когда встречается объект, отражающий сигнал, между двумя узлами может быть установлен новый путь. Каждый из этих отражающих путей будет отображаться на приемнике в разное время в зависимости от длины пути. Суммирование их вместе в приемнике вызывает искажения, как конструктивные, так и деструктивные.

Воздействие комбинации этих сигналов на приемник - искажение сигнала. Если разница во времени между отражениями мала по сравнению с шириной символа, искажение может быть внутри символа - внутрисимвольная интерференция. Если время отражения превышает время символа, отражение от одного символа будет влиять на следующие сигналы - еще одна причина межсимвольной интерференции.

Изучим файл `multipath_sim.grc`.

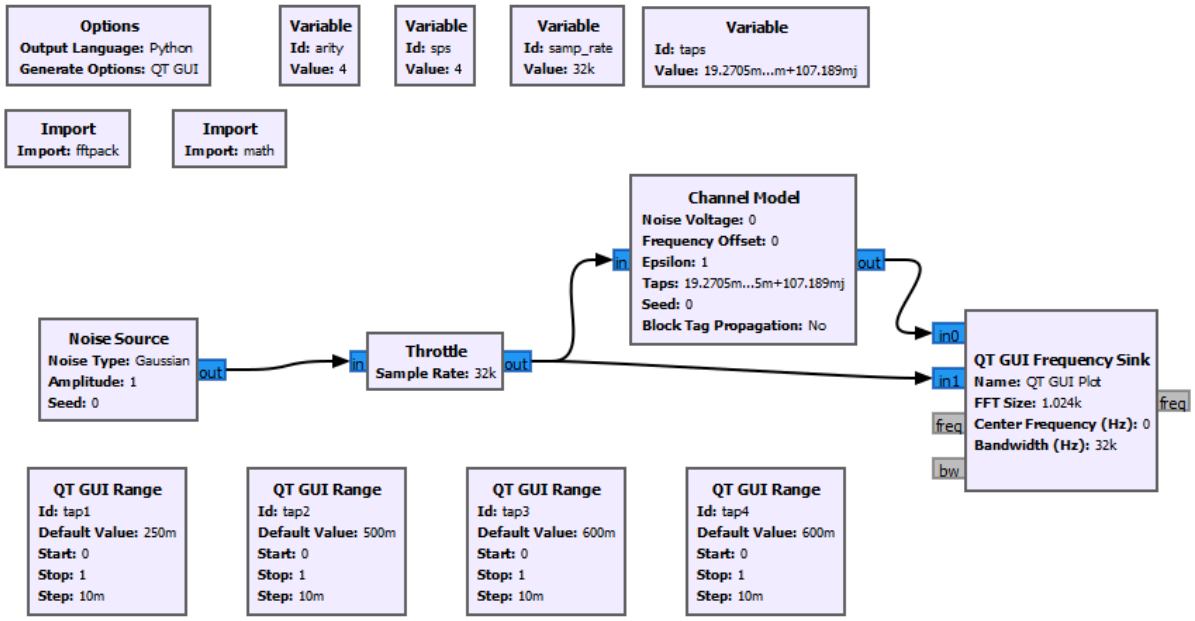

Рис. 18: Flowgraph multipath_sim.grc.

В этом `flowgraph` происходит настройка модели канала таким образом, чтобы предоставить каналу пять элементов управления эквалайзером, четыре из которых мы можем изменить. Эти элементы управления настроены одинаково по частоте, и мы можем настроить их от 0 до 1. При значении 1 элемент управления позволит этим частотам проходить без помех. При значении 0 они создадут глубокий ноль в спектре, который повлияет на все частоты вокруг него.

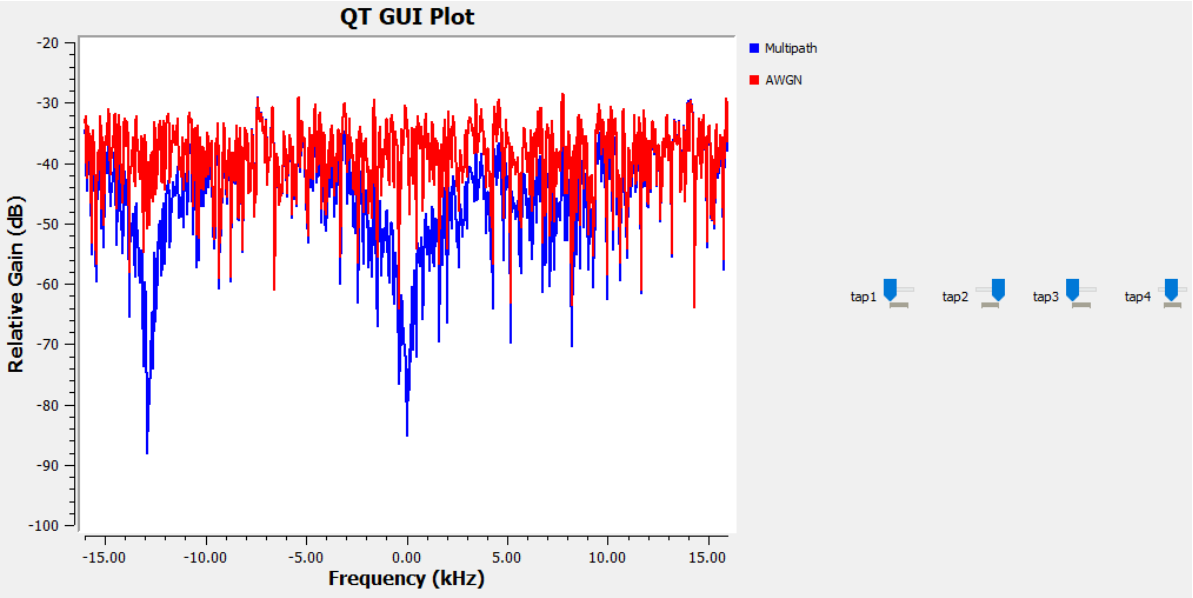

Рис. 19: Многолучевой канал.

Как видно на Рис.19 многолучевой канал создает некоторые искажения в сигнале. Задача эквалайзера - инвертировать этот канал. По сути, мы хотим отменить искажение, вызванное каналом, чтобы выходной сигнал эквалайзера был ровным. Но вместо того, чтобы настраивать каждый tap вручную, можно использовать алгоритмы, которые обновляют эти tap за нас. Наша задача - использовать правильный алгоритм эквалайзера и настроить параметры.

Одним из важных параметров здесь является количество tap в эквалайзере. Как мы видим в нашем моделировании, пять tap дают довольно грубый контроль над частотной характеристикой. Чем больше tap, тем больше времени требуется как на вычисление ответвлений, так и на запуск эквалайзера.

5 Эквалайзеры

GNU Radio имеет два легко используемых эквалайзера: эквалайзер CMA и эквалайзер LMS-DD.

5.1 Эквалайзер CMA

CMA или алгоритм постоянного модуля - это слепой эквалайзер , но он работает только с сигналами с постоянной амплитудой или модулем. Это означает, что цифровые сигналы, такие как MPSK, являются хорошими кандидатами, поскольку они имеют точки только на единичном круге.

Изучим файл `mpsk_stage4.grc`. В нём используется алгоритм CMA с 11 tap.

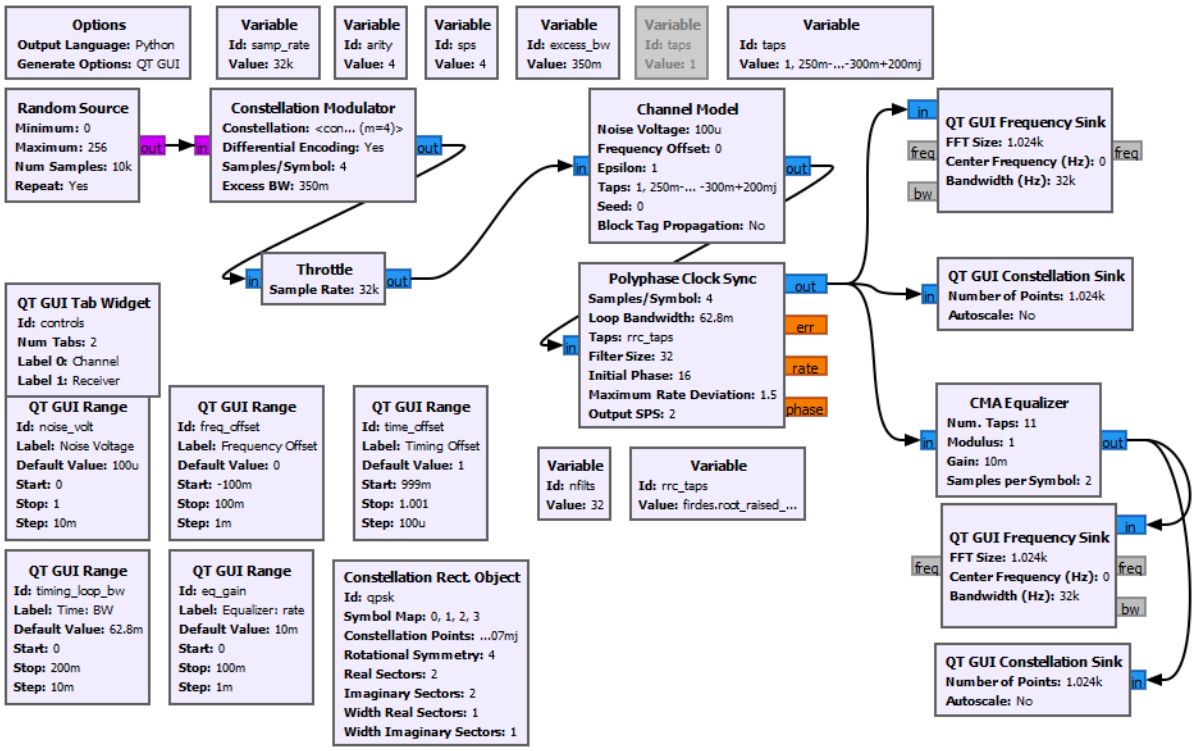

Рис. 20: Flowgraph mpsk_stage4.grc.

На Рис.21 можно видеть сходимость алгоритма CMA. Поскольку у нас есть и тактовая синхронизация, и блок эквалайзера, они сходятся независимо. В итоге мы можем увидеть эффект синхронизированного по времени многолучевого сигнала до и после эквалайзера. Перед эквалайзером у нас очень некрасивый сигнал даже без шумов. Эквалайзер прекрасно понимает, как инвертировать и сократить этот канал, чтобы у нас снова был хороший, чистый сигнал.

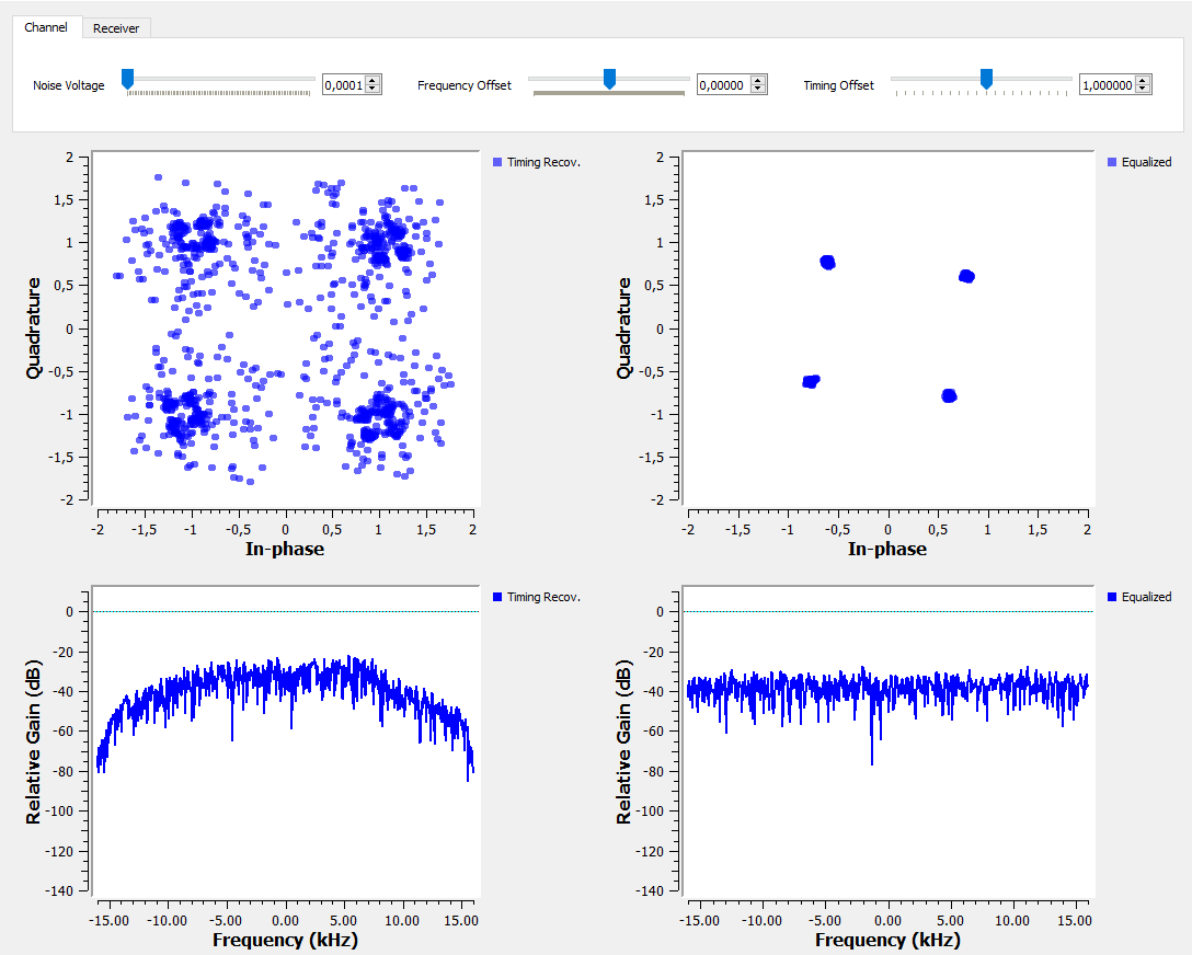

Рис. 21: Использование эквалайзера CMA.

5.2 Эквалайзер LMS-DD

Эквалайзер LMS-DD - это эквалайзер, ориентированный на решение по алгоритму наименьшего среднего квадрата. В отличие от слепого эквалайзера (CMA), этот эквалайзер должен знать принимаемый сигнал. Эквалайзеру необходимо знать точки созвездия для корректировки, и он использует решения о выборках, чтобы сообщить, как обновлять ответвления для эквалайзера.

Заменим эквалайзер CMA на эквалайзер LMS-DD в файле `mpsk_stage4.grc`.

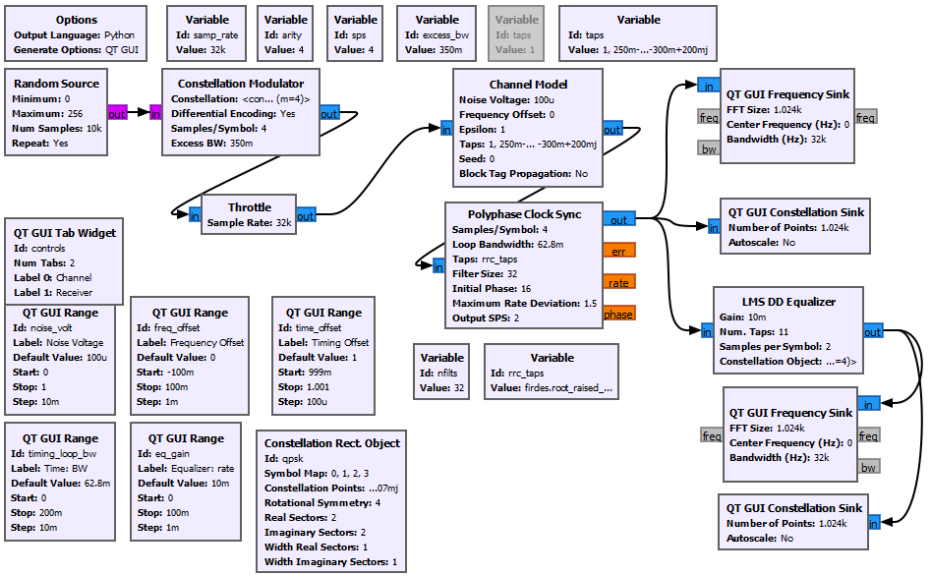

Рис. 22: Flowgraph mpsk_stage4.grc с эквалайзером LMS-DD.

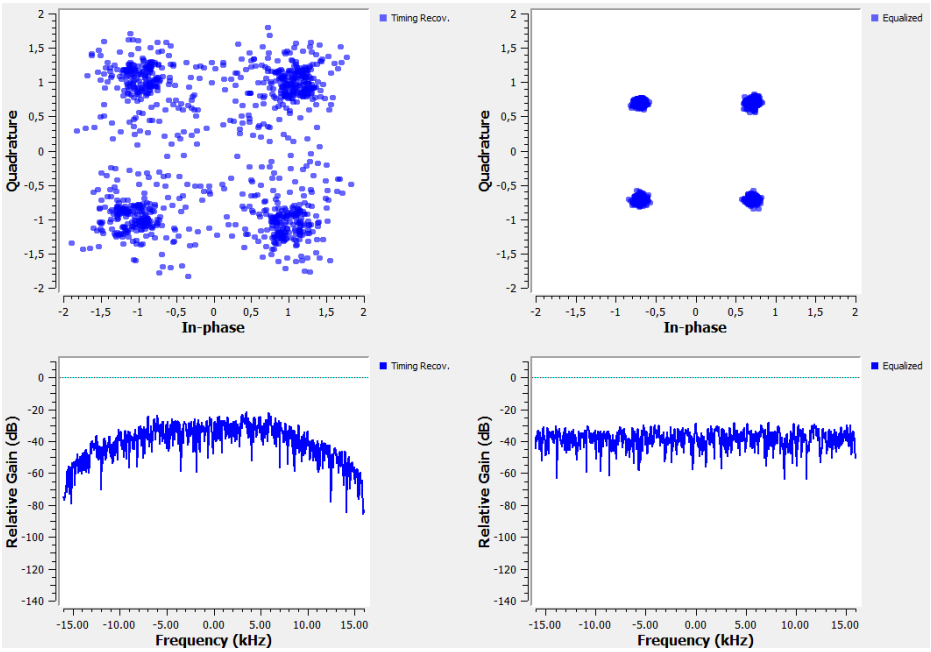

Рис. 23: Использование эквалайзера LMS-DD.

Этот эквалайзер отлично подходит для сигналов, которые не соответствуют требованиям постоянного модуля алгоритма CMA, поэтому он может работать с такими вещами, как модуляция типа QAM. Когда сигнал хорошего качества, этот эквалайзер может производить сигналы более высокого качества, потому что он напрямую знает сигнал.

6 Фазовая и точная частотная коррекция

Мы выровняли канал, однако у нас всё еще есть проблема смещения фазы и частоты. Эквалайзеры, как правило, адаптируются не быстро, поэтому смещение частоты может быть легко за пределами возможностей эквалайзера. Теперь нам нужно исправить любой сдвиг фазы, а также любой сдвиг частоты.

Во-первых, мы будем использовать цикл второго порядка, чтобы мы могли отслеживать как фазу, так и частоту во времени. Во-вторых, тип восстановления, с которым мы будем иметь дело, предполагает, что мы выполняем точную коррекцию частоты. Поэтому мы должны находиться в приличном диапазоне идеальной частоты.

Изучим файл `mpsk_stage5.grc`. В нём используется цикл Костаса.

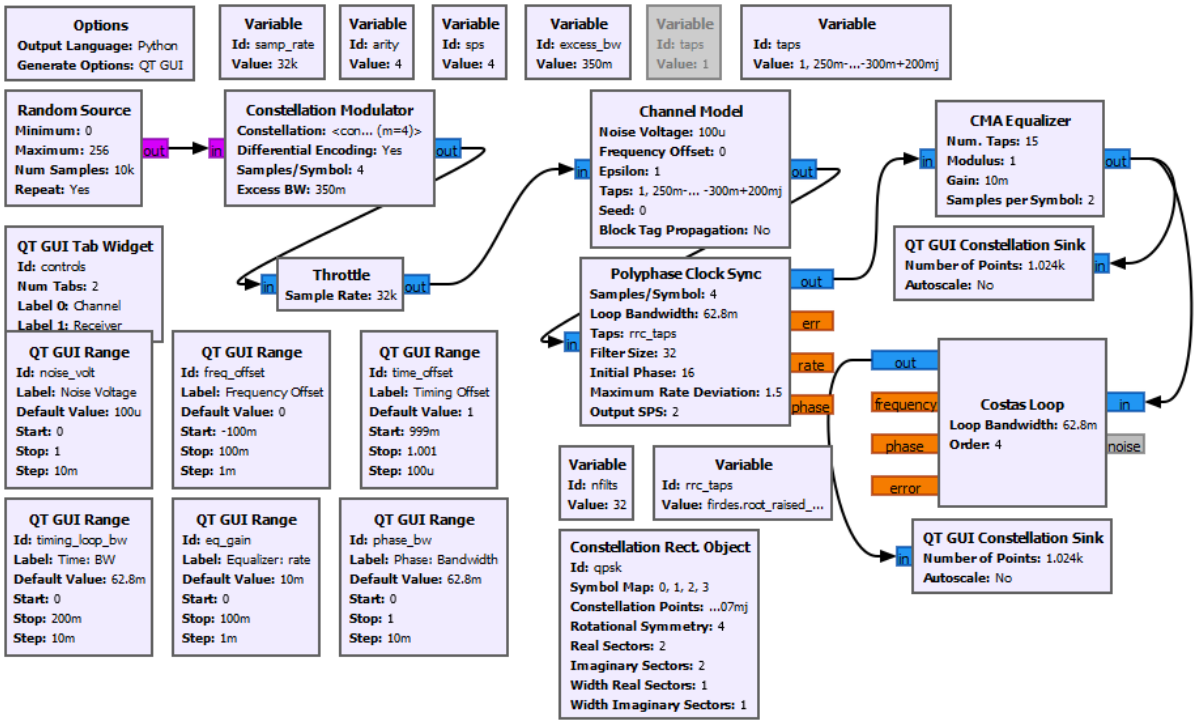

Рис. 24: Flowgraph mpsk_stage5.grc.

Блок Costas Loop может синхронизировать BPSK, QPSK и 8PSK. Этот блок использует цикл второго порядка и поэтому определяется параметром полосы пропускания цикла.

Мы установили шум, временной сдвиг, простой многолучевой канал и частотный сдвиг. После эквалайзера все символы находятся на единичном круге, но вращаются из-за смещения частоты, которое еще не исправлено. На выходе блока цикла Костаса можно увидеть заблокированное созвездие и дополнительный шум, с которым мы ничего не можем поделать.

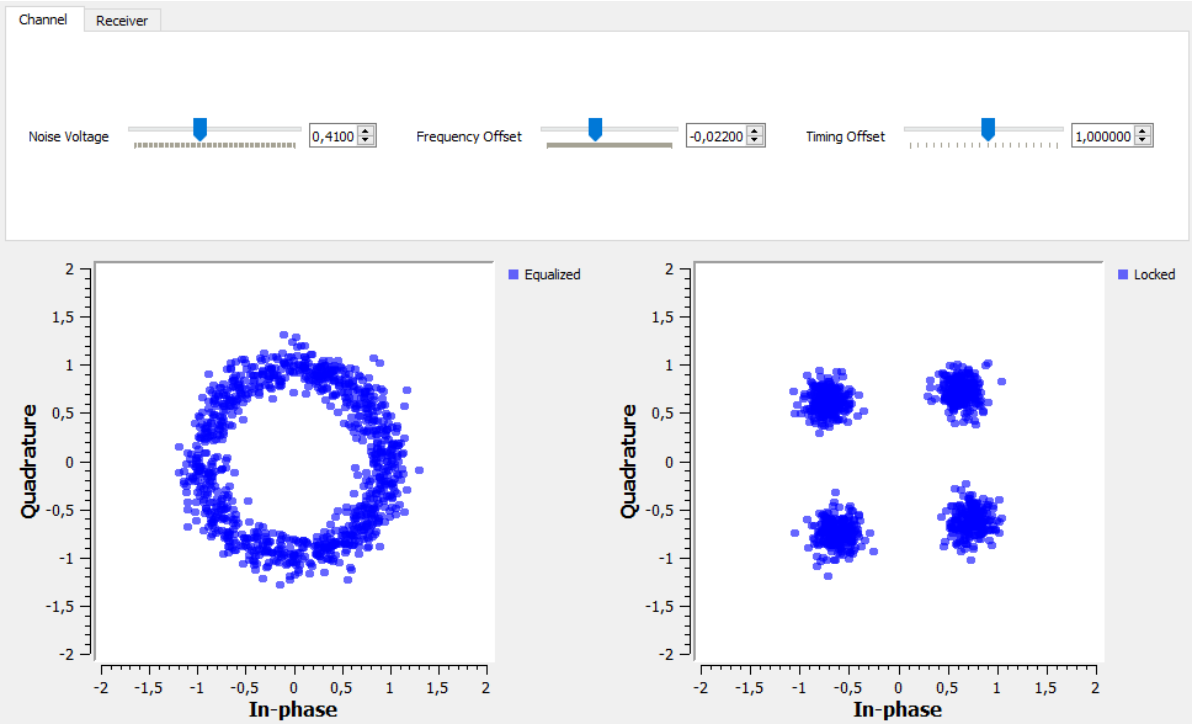

Рис. 25: Коррекция фазы и частоты.

7 Расшифровка

Теперь мы можем перейти к декодированию сигнала. Теперь давайте изучим файл `mpsk_stage6.grc`. В этом `flowgraph` добавлен Constellation Decoder после цикла Костаса. Мы передавали дифференциальные символы, установив для параметра Differential в блоке Constellation Modulator значение True. Т.е., на самом деле мы передавали не само созвездие, а разницу между символами созвездия.

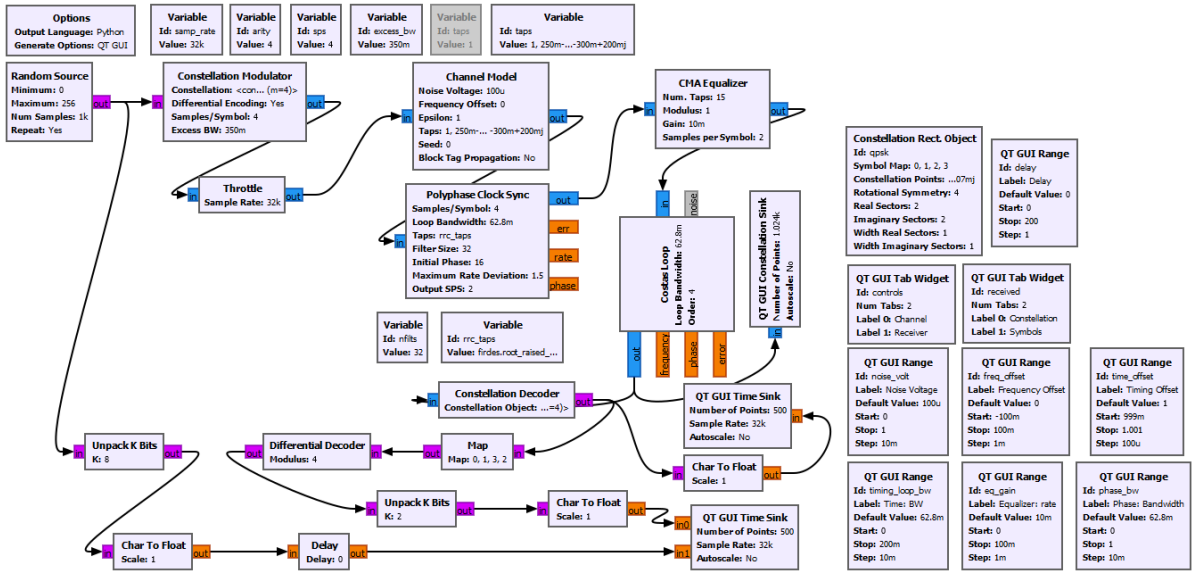

Рис. 26: Flowgraph mpsk_stage6.grc.

В этом `flowgraph` используется блок Differential Decoder для преобразования кодированных дифференциальным кодом символов обратно в их исходные символы. Блок Map используется для преобразования символов из дифференциального декодера в исходные символы. Блок Unpack bits используется для распаковки исходных символов от 0 до 3 (2 бита на символ) в биты. Теперь у нас есть исходный битовый поток данных.

Как мы узнаем, что это исходный битовый поток? Для этого сравним его с входным битовым потоком, но не напрямую. Необходимо установить задержку на переданные биты, используя Delay. Это связано с тем, что в цепочке приёмника есть множество блоков и фильтров, которые задерживают сигнал, а потому принятый сигнал отстаёт на некоторое количество бит.

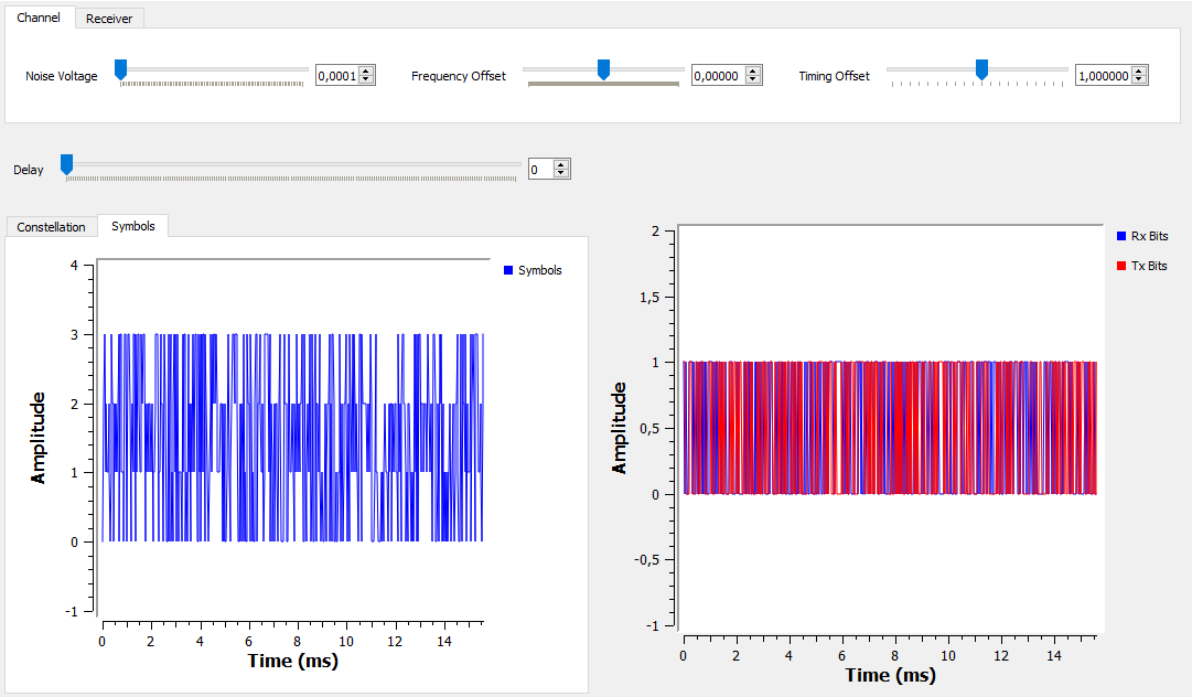

Рис. 27: Сравнение данных.

8 Выводы

В ходе выполнения данной лабораторной работы была изучена демодуляция PSK. Были изучены проблемы искажения сигнала и эффектов канала. Также были изучены основные моменты, необходимые для восстановления сигнала: сроки восстановления, многолучевые каналы, фазовая и частотная коррекции, декодирование символов и порядок битов.