

Телекоммуникационные технологии

# Отчет по лабораторной работе № 12

Тема:

**Модуляция и демодуляция сигнала**

Самсонов Сергей

## Упражнение 12.1

**Задание:** Для передачи сигнала используется модулятор ( Constellation Modulator Block ), кодировщик символов Constellaton Rect. Object и генератор байтовых значений. Посмотреть, как избыточная пропускная способность влияет на форму фильтра.

**Решение:** Данный Flow Graph осуществляет передачу QPSK, показывая переданный сигнал, полученный на приемнике сигнал во времени и частотах и «созвездие».

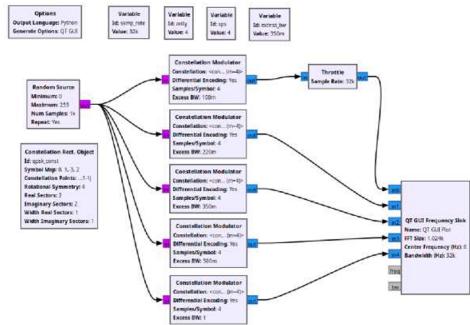


Рис. 1: Flow Graph mpsk-rrc-rolloff

## Заключение:

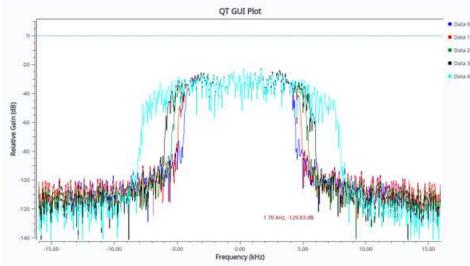


Рис. 2: График mpsk-rrc-rolloff

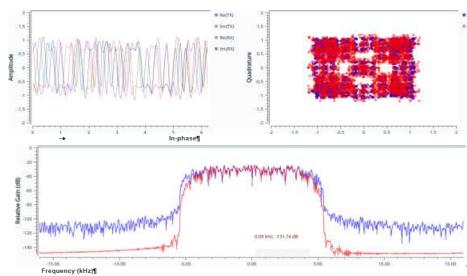


Рис. 3: График mpsk-stage1

Изменения excess bandwidth влияют на то, насколько крутыми будут границы (насколько плавно они будут понижаться).

На символ у нас идет 4 сэмпла. Это можно судить по увеличению дискретизации на созвездии. Фильтрация осуществляется путем вставки RRC-фильтром межсимвольных помех. На самом деле данный прием искажает принятый сигнал (размываются символы). Чтобы это поправить, добавим RRC-фильтр для приемника. Таким образом при свертке получаются импульсы приподнятого косинуса, при этом минимизируются помехи.

### Упражнение 12.2

**Задание:** При помощи Channel Model добавим искажения в канале передач, чтобы сымитировать шум (регулируется напряжением). Нужно учесть, что разные домены тактовых сигналов в приемнике и передатчике могут вызвать сдвиг. Есть еще одна проблема: приемник не знает идеальную точку сэмплирования. **Решение:**

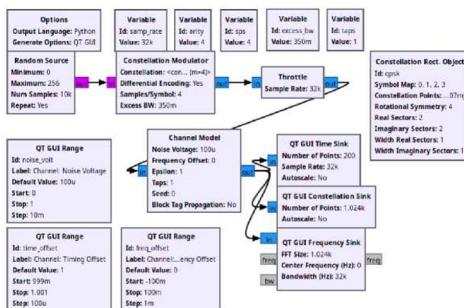
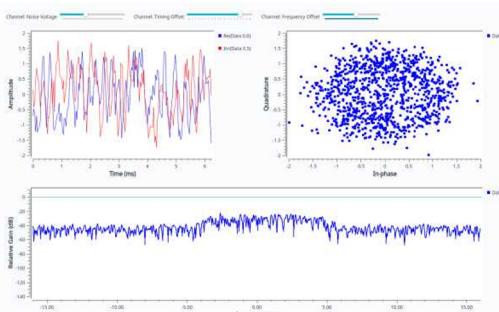


Рис. 4: Flow Graph mpsk-stage2

Все выше указанные проблемы можно сымитировать и настроить при помощи нашего Flow Graph, а также при помощи него можно наблюдать их влияние на сигнал.

### Заключение:



Получилось что-то бесформенное и явно не то, что нам нужно. Значит имитации проблем (по крайней мере некоторых из них) - работают.

### Упражнение 12.3

**Задание:** Для восстановления времени, нам нужно отыскать наилучшее время для сэмплинга входящего сигнала, чтобы максимизировать SNR и минимизировать ISI.

**Решение:** Следующий Flow Graph иллюстрирует проблему ISI, мы тут

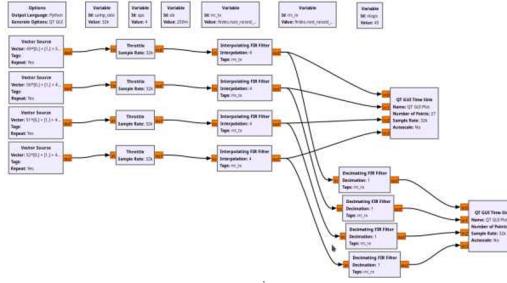


Рис. 5: Flow Graph symbol-sampling

создаем 4 символа-единицы подряд, а затем фильтруем их:

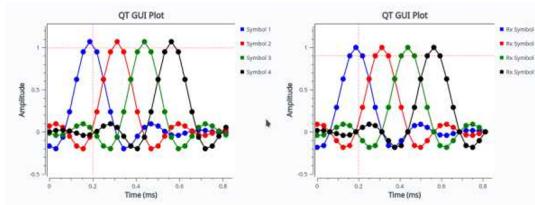


Рис. 6: График symbol-sampling

**Заключение:** Теперь посмотрим на эффект разных доменов тактовых

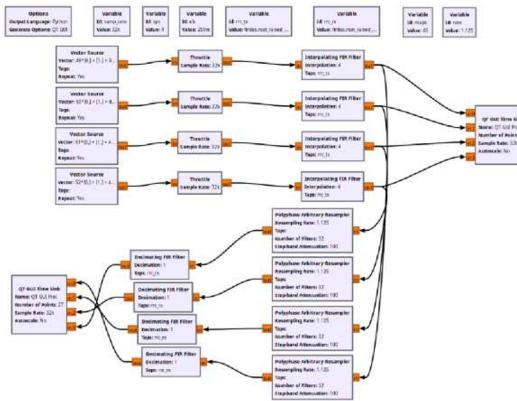


Рис. 7: Flow Graph symbol-sampling-diff

сигналов на передатчике и приемнике. Разница во времени специально завышена для наглядности. Для синхронизации используем техникой восстановления polyphase filterbank. Во-первых, она решит проблему с

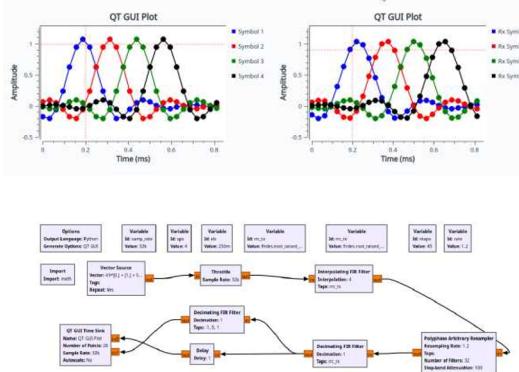


Рис. 8: График symbol-sampling-diff

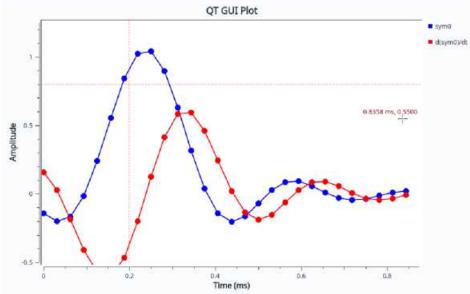


Рис. 9: График symbol-differential-filter

тактовыми доменами. Во-вторых, уменьшит ISI. В-третьих, понизит частоту дискретизации сигнала и будет производить по 1 сэмплу на символ.

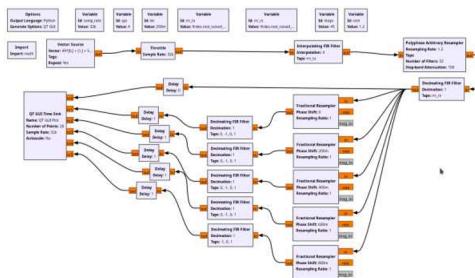


Рис. 10: Flow Graph symbol-differential-filter-phases

Можно попробовать переделать Flow Graph, чтобы перебрать разные фильтры с разными сдвигами по фазе, чтобы определить наиболее подходящий сдвиг.

Теперь можно настроить блок синхронизации полифазного тактирующего сигнала в приемнике. Нужно настроить 32 фильтра: Слева полученный сигнал до восстановления, а справа соответственно сигнал, уже прошедший фильтрацию. Можно отрегулировать Frequency Offset, чтобы минимизировать ошибки в интерпретации абсолютных значений.

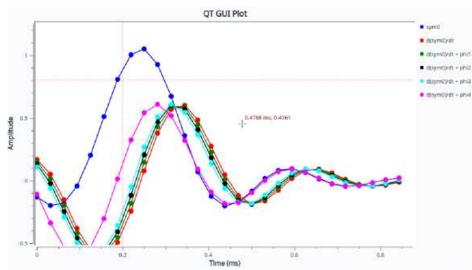


Рис. 11: График symbol-differential-filter-phases

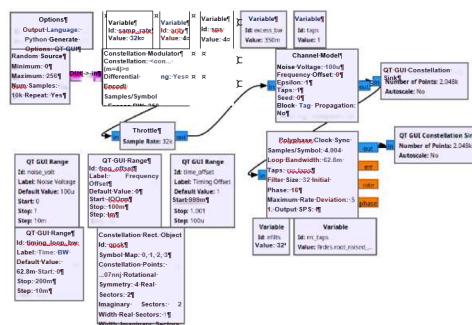


Рис. 12: График mpsk-stage3

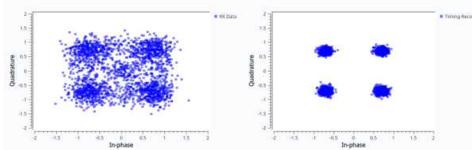
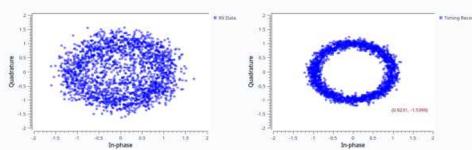


Рис. 13: График mpsk-stage3 x2



### Упражнение 12.4

**Задание:** Сигнал поступает на приемник с множества направлений сразу.

**Решение:** Если сигнал поступает с разных направлений, он может исказиться.

Чтобы это исправить, сделал следующий график. Здесь будет создан канал с 4 управляемыми параметрами, при установке некоторых из в 1 соответствующие частоты смогут пройти без помех, а при 0 они будут производить «глубокий нуль» в спектре (влияет и на соседние частоты).

**Заключение:** Получается эквалайзер, цель которого преобразить искаженный сигнал в горизонтальную прямую.

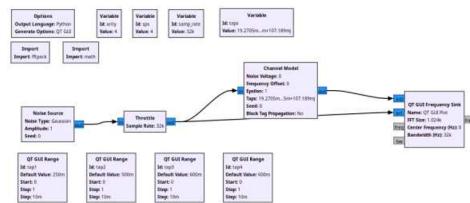


Рис. 14: Flow Graph multipath-sim

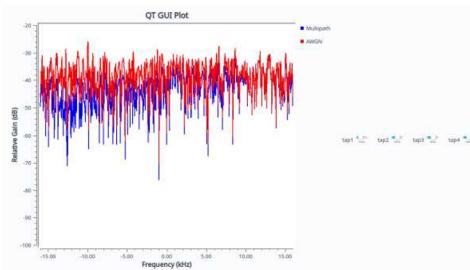


Рис. 15: График multipath-sim

### Упражнение 12.5

**Задание:** В GNU Radio есть два удобных встроенных эквалайзера: СМА и LMS DD. Constant Modulus Algorithm - это «слепой» эквалайзер, но он работает лишь с сигналами с постоянной амплитудой, что хорошо работает с цифровыми сигналы вроде MPSK.

**Решение:** Пример ниже демонстрирует MPSK-эквалайзер (параметры подстроены согласно прошлому опыту).

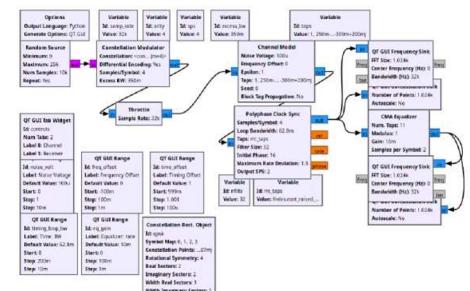


Рис. 16: Flow Graph mpsk-stage4

Можно видеть, как СМА сходится. Тут и тактирующий сигнал синхронизирован, и блок эквалайзера, и сходятся они независимо, но одна стадия влияет на другую, поэтому некоторое взаимодействие между ними все же есть.

График mpsk-stage4

**Заключение:** Исходный сигнал (слева) хоть и не зашумлен, но все равно выгля- дит криво. Тем не менее эквалайзер смог инвертировать и

избавиться от канала.

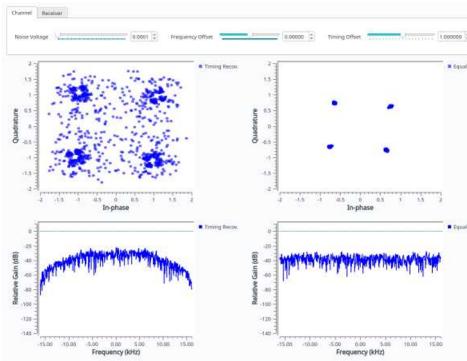


Рис. 17: Flow Graph mpsk-stage4-lms-dd

Рис. 18: График mpsk-stage4-lms-dd

### Упражнение 12.6

**Задание:** Поскольку после применения эквалайзера у нас все еще остается проблема со смещением в фазе и частоте. Попробуем использовать цикл второго порядка, чтобы следить за фазой и частотой с течением времени. Кроме того, нужно сделать достаточно хорошую коррекцию частот для удачного восстановления, поэтому нужно будет убедиться, что мы достаточно близки к идеальной частоте, иначе цикл не будет сходиться.

**Решение:**

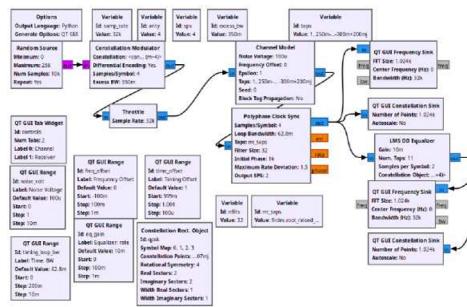


Рис. 19: Flow Graph mpsk-stage5

Воспользуемся циклом Костаса. Соответствующий блок может синхронизировать BPSK, QPSK и 8PSK.

**Заключение:** Этот блок, как и все наши другие, использует цикл второго порядка, а потому и имеет соответствующий параметр, связанный с пропускной способностью. Ему также нужно знать степень PSK-модуляции (4 для QPSK). Можно видеть, как после работы эквалайзера все символы расположены на единичной окружности, но из-за сдвига

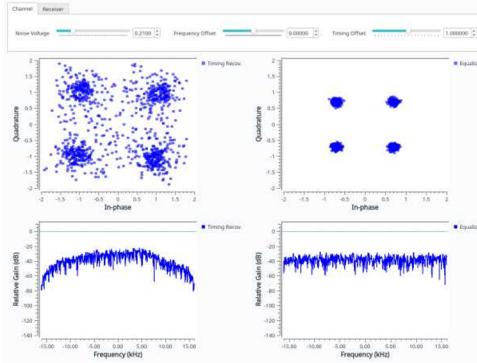


Рис. 20: График mpsk-stage5

частот, они как бы «размазаны» по ней, а не соответствуют нашим 4 точкам. После цикла Костаса мы уже видим ровно 4 исходные точки, но некоторый шум тоже присутствует.

### Упражнение 12.7

**Задание:** Сделать декодер.

**Решение:** После цикла Костаса мы вставляем еще Constellation Decoder ,

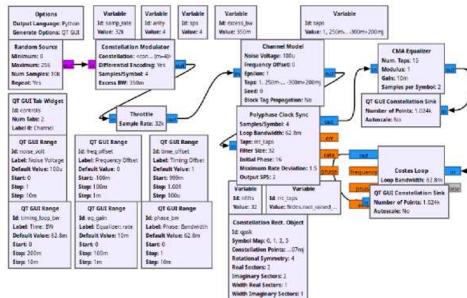


Рис. 21: Flow Graph mpsk-stage6

но это еще не все. На этом этапе мы видим символы 0..3, потому что на это рассчитан наш алфавит, но у нас нет никакой уверенности, что эти 0..3 правильно соответствуют тем 0..3, что мы передавали изначально. Нам удавалось обходить этот вопрос, потому что мы использовали дифференциальное кодирование в Constellation Modulator 'е. Выключим теперь это. Flow Graph использует блок Differential Decoder , чтобы перевести дифференциально-закодированные символы обратно в исходные при помощи сдвигов фаз, а не абсолютной фазы. Но даже так наши символы еще не совсем верные. Во время синхронизации математика и физика была на нашей стороне, а теперь нам надо интерпретировать символы, основываясь на том, что кто-т оказал, какими они были. То есть, нам просто надо знать, как они соотносятся. Для этого мы воспользуемся блоками Map и Unpack Bit .

**Заключение:** Чтобы убедиться, что теперь мы действительно получаем тот самый поток данных, мы просто сравним, что было в начале, с тем, что получили (ведь это симуляция, мы имеем доступ к любой информации). Передатчик создает упакованные биты, поэтому при помощи Unpack Bit мы должны их распаковать из 8 бит на байт в 1 бит на байт, затем превратить их в float 0.0 и 1.0, потому что Time Sink умеет только float и complex принимать. Но напрямую сравнивать значения пока нельзя, потому что в приемнике есть еще много узлов, которые задерживают данные, поэтому нам нужен еще и блок Delay .

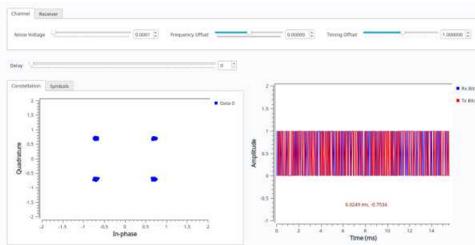
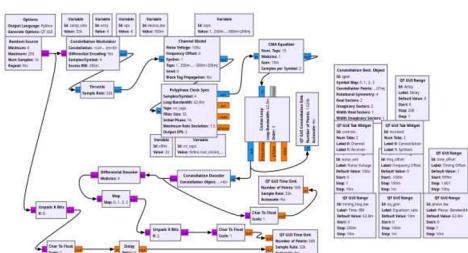
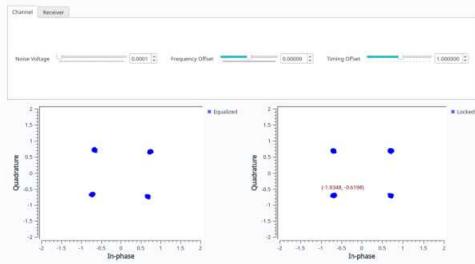


Рис. 22: График mpsk-stage