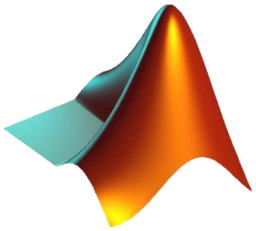
Оглавление

Имитационная модель канала связи OFDM в Matlab/Octave

Цель практики

Цель практики заключается в освоении базовых подходов имитационного моделирования в связной тематике, используя язык программирования Matlab. Кроме того, вы лучше изучите основою технологию, которая входит в основу физического уровня новейших систем связи – технологию OFDM.

Почему Matlab?



Вам предлагается изучить основы языка программирования Matlab, который используется в большинстве мировых R&D центрах для разработки, моделирования и прототипирования новых телекоммуникационных технологий. Мотивацией для изучения Matlab может служить тот факт, что при собеседовании в лучшие зарубежные R&D центры часто интересуются про навык пользования этим инструментом, поэтому владение Matlab'ом повышает вероятность вашего успеха.

[.....]

Процедура практики

Самостоятельное выполнение	*С помощью видеокурса*
Если у вас есть навык владения ПО Matlab и творческий интерес, то Вам предлагается уникальная возможность реализовать модель самостоятельно, опираясь только на текстовое описание модели и задание.	Можно выполнить практику с помощью предлагаемого видеокурса, в котором последовательно выполняются все блоки модели. Естественно, выполнение задания по вариантам никто не отменял.

Общее задание на практику

Реализовать полную модель OFDM канала связи, включающую: формирование и обработку сигнала, многолучевой канал распространения с аддитивным белым гауссовым шумом, блок оценки вероятности битовых ошибок.

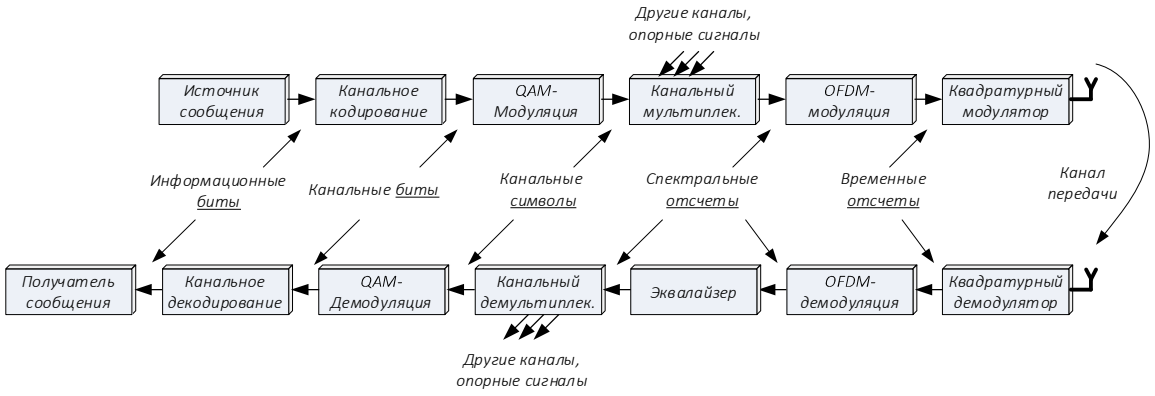
Осуществить передачу текстового сообщения по каналу. Рассчитать количество битовых ошибок при различном отношении сигнал/шум и конфигурации сигнала.

[.....]

Описание смысла модели

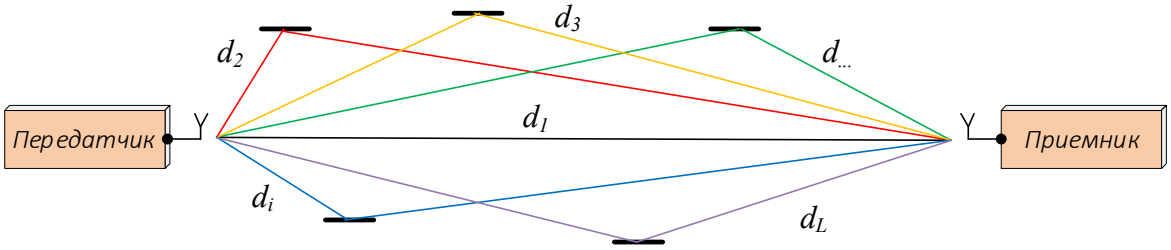
Структурно, модель разделяется на 3 части: передатчик, приемник, канал передачи между ними и блок расчета ошибок. Опишем более подробно каждую из этих частей:

1) В модели передатчика пользователь вводит текстовое сообщение, которое преобразуется в битовое. Далее, это сообщение кодируется с помощью сверточного кодера и перемешивается. Закодированное битовое сообщение модулируется с помощью QAM-модуляции и размещается на поднесущих OFDM-символа вместе с опорным сигналом. Далее, происходит формирование OFDM-символа на выходе модели передатчика.

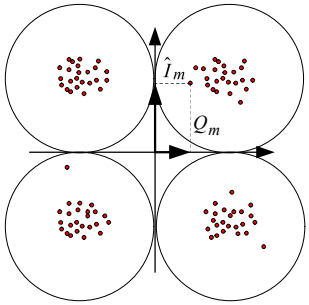


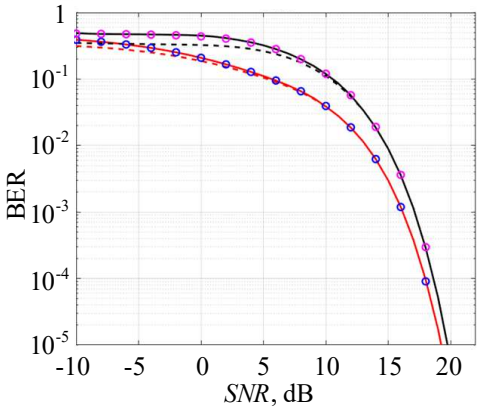
2) Далее, сигнал поступает в модель многолучевого канала передачи, в котором происходит затухание сигнала в зависимости от расстояния

между передатчиком и приемником, а также накладывается мультипликативная помеха и АБГШ. Соответственно, передаваемый сигнал приобретает искажения.



3) Искаженный сигнал поступает в модель приемника. В первую очередь происходит демодуляция OFDM-сигнала и выделение опорных поднесущих, с помощью которых оценивается АЧХ канала передачи. На основании этой оценки рассчитывается корректирующая АЧХ эквалайзера, которая используется для эквалайзирования. После эквалайзирования, из спектра принятого OFDM-символа выделятся QAM-символы, которые демодулируется. Демодулированное битовое сообщение перемешивается в обратном порядке, декодируется в декодере Витерби и преобразовывается в текстовое сообщение в знаковом декодере. В конечном итоге, если сообщение было передано без ошибок, то пользователь увидит на выходе приемника то сообщение, которое он ввел в передатчике.

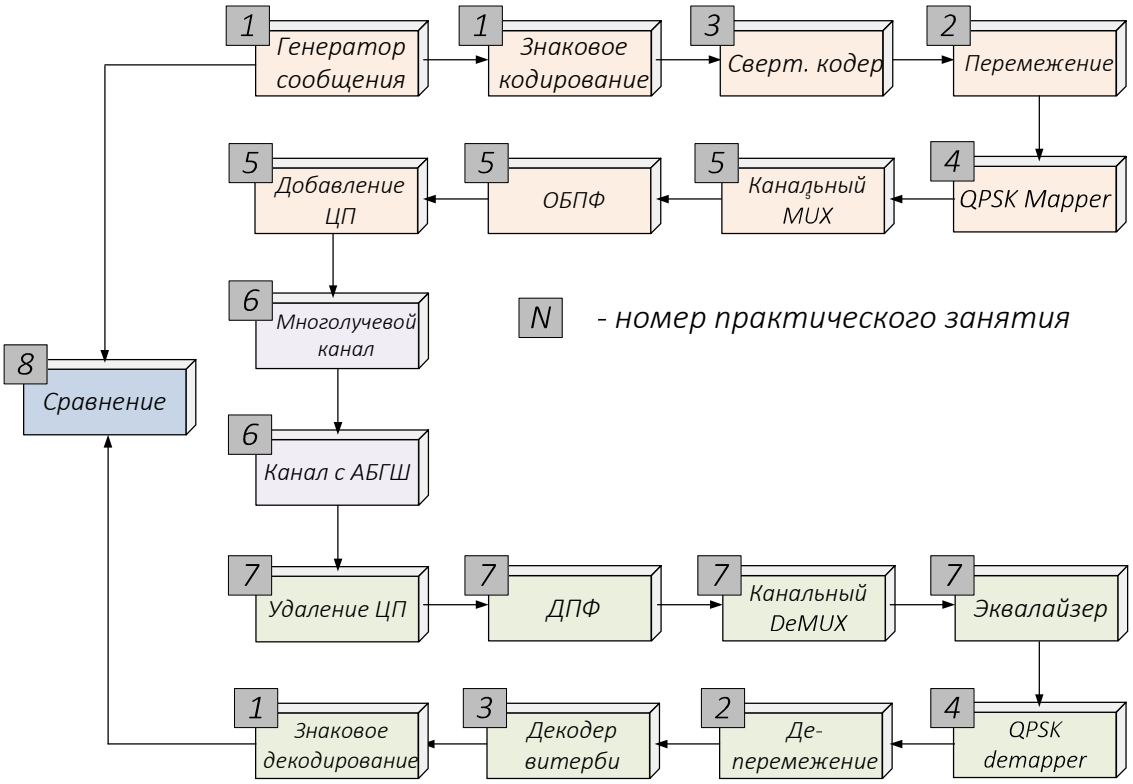




4) Последним этапом является расчет символьных (количество ошибочно принятых букв) и битовых (после Витерби-декодирования) ошибок между переданным и принятым сообщением. В результате, Вы построите график вероятности битовых ошибок от отношения сигнал-шум подобный тому, который представлен ниже.

План реализации по занятиям

Реализация базовой имитационно модели разделена на 8 частей, а для каждой части подготовлен соответствующий видеоурок. Общая структурная схема модели представлена на рисунке ниже. В сером квадрате указан номер видеоурока, на котором поясняется разработка этого конкретного блока модели.



Описание входов, выходов и параметров модельных блоков

К каждому блоку модели прилагается описание входов, выходов и параметров. Поясним, что это значит.

а) Входные переменные - сигнал, которые поступают на вход блока. В модели этот сигнал представлен в виде вектора, содержащего сигнальные отсчеты. Используются 3 различных сигнала, содержащие: текстовые символы, биты и квадратурные отсчеты (представленные в виде комплексных чисел). Дополнительно, в некоторых блоках используются структуры (вид представления данных в Matlab), которые позволяют

удобно объединить несколько различных сигналов в один. В каждом модельном блоке поясняется – какие сигналы должны быть на входе.

б) Выходные переменные – сигнал, который должен быть на выходе блока. Те же самые *правила*, что и для входных переменных.

в) Дополнительные параметры – параметры, которыми пользователь может управлять вручную для регулирования функционала модельного блока. Например, у пользователя должны быть возможность изменить мощность шума или количество лучей в модели многолучевого канала передачи. Все параметры, которые приведены в описании блока должны быть реализованы.

Практика 1. Знаковое кодирование

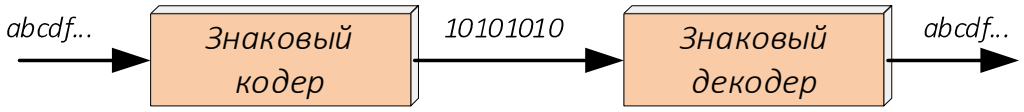
Задание

Реализовать знаковое кодирование и декодирование текстового сообщения.

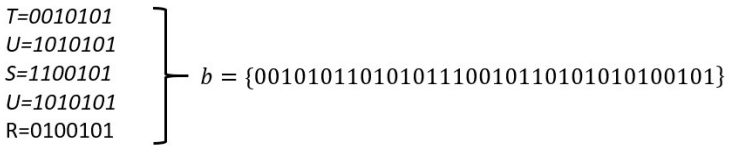
[.....]

Описание знакового кодера

Задача знакового кодера – преобразовать входное символьное сообщение в набор двоичных символов (бит). На вход функции поступает текстовое сообщение длительностью от 30 до 100 символов. Длина сообщения задается вариантом. Алфавит сообщения состоит из 64 символов: латинский (маленькие и заглавные буквы, 52 символа), цифры (0…9), пробел и точка. Требуется преобразовать текстовое сообщение в битовое. Для этого нужно выбрать разрядность кодового слова (количество бит на 1 символ сообщения), а каждый символ должен быть закодирован индивидуальным кодом. На выходе функции должно быть битовое сообщение. Запрещено использовать встроенную в Matlab функцию преобразования формата данных.



На рисунке ниже показан пример символьного кодирования слова TUSUR с помощью алфавита ASCII, согласно которому каждый символ кодируется семизначным кодовым словом.



Биты				5	0	1	0	1	0	1	0	1
				6	0	0	1	1	0	0	1	1
1	2	3	4	7	0	0	0	0	1	1	1	1
0	0	0	0		NUL	DLE	SP	0	@	P	`	p
1	0	0	0		SOH	DC1	!	1	A	Q	a	q
0	1	0	0		STX	DC2	"	2	B	R	b	r
1	1	0	0		ETX	DC3	#	3	C	S	c	s
0	0	1	0		EOT	DC4	$	4	D	T	d	t
1	0	1	0		ENQ	NAK	%	5	E	U	e	u
0	1	1	0		ACK	SYN	&	6	F	V	f	v
1	1	1	0		BEL	ETB	'	7	G	W	g	w
0	0	0	1		BS	CAN	(	8	H	X	h	x
1	0	0	1		HT	EM	)	9	I	Y	i	y
0	1	0	1		LF	SUB	*	:	J	Z	j	z
1	1	0	1		VT	ESC	+	;	K	[	k	{
0	0	1	1		FF	FS	,	<	L	\	l	|
1	0	1	1		CR	GS	-	=	M	]	m	}
0	1	1	1		SO	RS		>	N	^	n	~
1	1	1	1		SI	US	/	?	O	-	o	DEL

Описание входных/выходных переменных кодера

Входные переменные:	вектор с текстовым сообщением битами
Выходные переменные:	вектор с битами
Параметры:	нет

[.....]

Описание знакового декодера

Знаковый декодер — это обратная функция знакового кодера, выполняющая инверсную операцию. На его вход поступает битовое сообщение произвольной длины. Требуется преобразовать это битовое сообщение в текстовое, используя такую же знаковую кодировку, как в функции знакового кодирования. На выходе функции должно быть такое же текстовое сообщение, как и на входе знакового кодера.

Описание входных/выходных переменных декодера

Входные переменные:	вектор с битами
Выходные переменные:	вектор с текстовым сообщением
Параметры:	нет

Практика 2. Помехоустойчивое кодирование

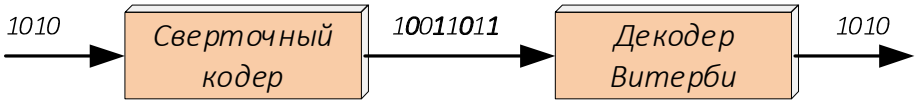
Задание

Реализовать операцию сверточного кодирования и витерби-декодирования битового сообщения.

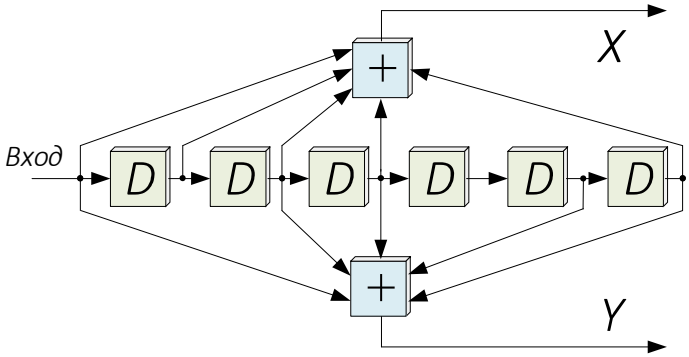
[.....]

Описание помехоустойчивого кодирования

Задача кодера – внести избыточные биты в сообщение, чтобы повысить помехоустойчивость этого сообщения. На его вход поступает битовое сообщение произвольной длины.



Скорость кодирования: 1/2. Полином кодера: [171_{oct}, 133_{oct}]. Схема кодирования представлена на рисунке ниже. Очередность выхода: поочередное поступление битов с выходов в порядке X, Y, X, Y,.....



На выходе сверточного кодера должно быть битовое сообщение, длина которого в 2 раза больше, чем длина сообщения на входе. Для проверки вашего программного кода может быть использована функция convenc**.** Для того, чтобы эта функция была доступна в Octave, необходимо загрузить библиотеку «телекоммуникации» (!! напишите и активируйте pkg load communications в командном окне !!).

Описание входных/выходных переменных кодера

Входные переменные:	вектор с изначальными битами
Выходные переменные:	вектор с закодированными битами
Параметры:	нет

[.....]

Описание декодера Витерби

На вход функции декодера Витерби поступает битовое сообщение, которое декодируется с помощью функции **vitdec**. Необходимо использовать те же самые параметры кодера, как в функции сверточного кодирования. Длина битового сообщения на выходе будет в 2 раза короче, чем на входе.

Описание входных/выходных переменных

Входные переменные:	вектор с закодированными битами
Выходные переменные:	вектор с изначальными битами
Параметры:	нет

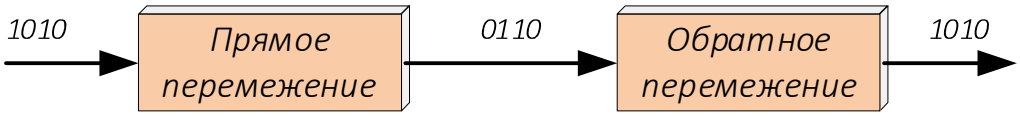
Практика 3. Перемежение

Задание

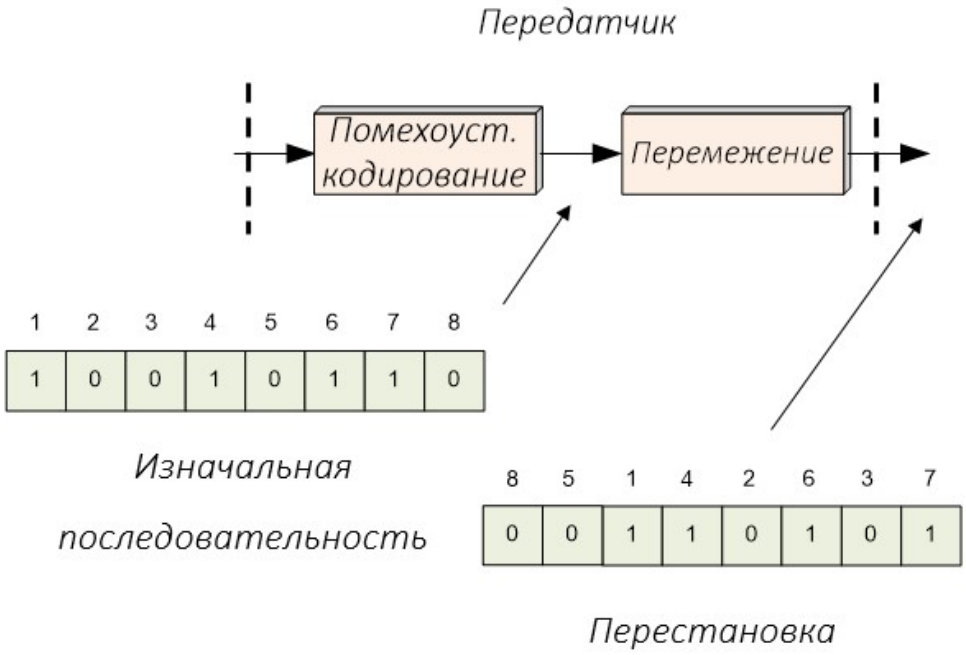
Реализовать операцию прямого и обратного перемежения закодированного битового сообщения.
[.....]

Описание прямого перемежения

Задача перемежителя – перемешать биты в кодовом слове (с выхода сверточного кодера), используя псевдослучайный закон перестановки. Внутри перемежителя должен быть описан закон перестановки (можно использовать случайный закон).



Для перестановки бит можно сформировать вектор, содержащий позицию бит после перестановки. Например, используя вектор бит: {10010110} и вектор позиций бит после перестановки {85142637}, можно осуществить перестановку и выходной вектор будет {00110101}.



Описание входны х/выходных переменных

Входные переменные:	вектор с изначальными битами
Выходные переменные:	вектор с перемешанными битами
Параметры:	нет

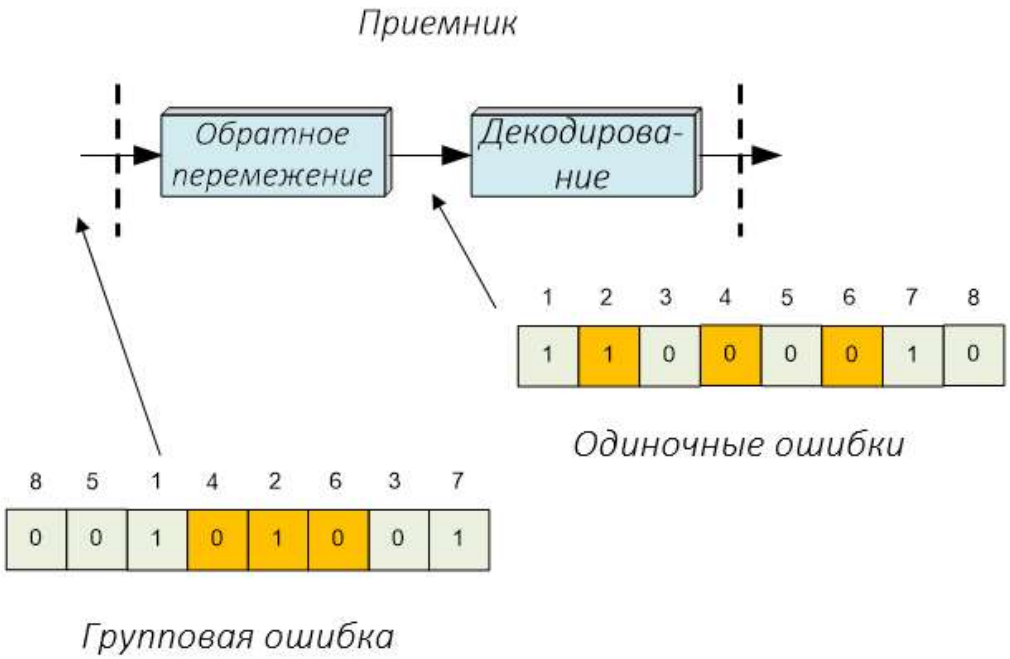
[.....]

Описание обратного перемежителя

Задача обратного перемежителя – перемешать биты в обратном порядке, используя такой же закон перестановки бит, как в перемежителе. Биты на входе прямого перемежителя и на выходе обратного перемежителя должны совпадать.

Описание входных/выходных переменных

Входные переменные:	вектор с перемешанными битами
Выходные переменные:	вектор с изначальными битами
Параметры:	нет



Практика 4. QPSK-модуляция

Задание

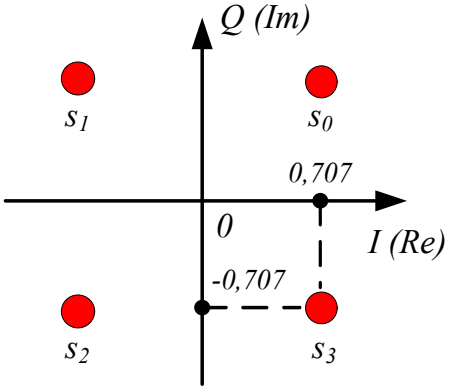
Реализовать операцию QPSK-модуляции битового сообщения и QPSK-демодуляции символов модуляции.

[.....]

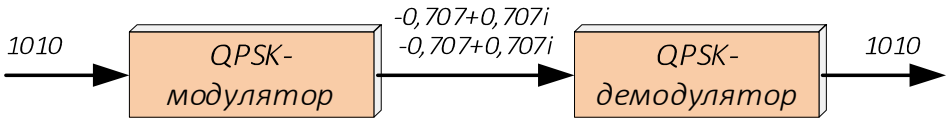
Описание работы QPSK-модулятора

Задачей QPSK-модулятора является преобразование входного битового потока в пары квадратурных амплитуд вида *I+jQ* в соответствии с заданной таблицей модуляции, которая представлена справа. Если на вход модулятора поступила пара бит 11, то но выходе должен получиться комплексный символ -0,707-0,707i и так далее. Количество бит на входе модулятора должно быть четным, в противном случае добавьте нулевой бит в конец входного битового сообщения. Поскольку в 1 комплексный символ записывается 2 бита сообщения, то длина сигнала на выходе модулятора будет меньше в 2 раза, чем длина сигнала на входе.

b_0, b_1	I, Q
00	0.707, 0.707
01	0.707, -0.707
10	-0.707, 0.707
11	-0.707, -0.707



(!) Запрещено использовать встроенные функции из библиотеки.



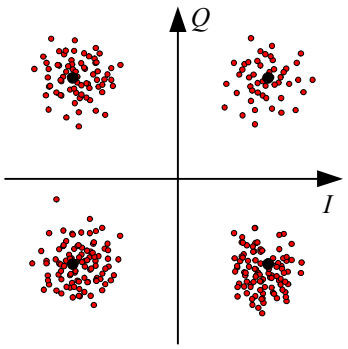
Описание входных/выходных переменных

Входные переменные:	вектор с битами
Выходные переменные:	вектор с комплексными символами
Параметры:	нет

[.....]

Описание работы QPSK-демодулятора.

Задачей QPSK-демодулятора является преобразование входного комплексного сигнала в битовый поток в соответствии с заданной таблицей модуляции, которая использовалась при модуляции. Однако, из-за действия шума, на входе демодулятора мы никогда не увидим идеального комплексного сигнала, который был на выходе модулятора. Поэтому, требуется указать область декодирования каждого символа.



Для QPSK область декодирования описывается очень просто. Например, для верхнего правого символа созвездия (0,707+0,707i) область декодирования будет соответствовать: I (реальная часть) > 0 и Q (мнимая часть) > 0, и тогда любой отсчет, который под действием шума попал в эту область, будет интерпретирован как 0,707+0,707i, что соответствует паре бит 00 (в соответствии с таблицей модуляции). Аналогично, можно определить области декодирования для остальных точек на созвездии.

Соответственно, каждый входной комплексный отсчет должен быть преобразован в пару бит. Длина сигнала на выходе демодулятора будет в 2 раза больше, чем длина сигнала на входе.

Описание входных/выходных переменных

Входные переменные:	вектор с комплексными символами
Выходные переменные:	вектор с битами
Параметры:	нет

Практика 5. OFDM-модуляция

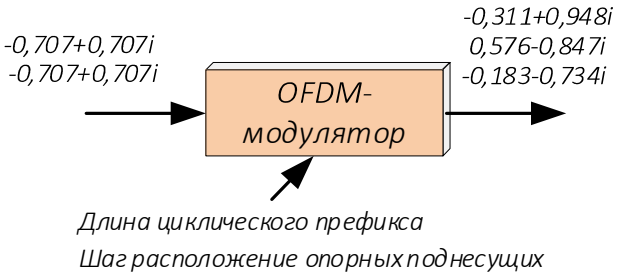
Задание

Реализовать формирование OFDM-символа, включающего в себя QPSK-символы и опорные сигналы.

[.....]

Описание работы OFDM-модулятора

Цель применения технологии OFDM состоит в борьбе с многолучевостью. В реализуемой имитационной модели задача блока OFDM-модулятора заключается в формировании OFDM-символа, включающего информационные символы и опорный сигнал. Опорный потребуется для оценки АЧХ канала передачи при эквалайзировании сигнала в приемнике. В информационных символах содержится передаваемое текстовое сообщение. Соответственно, на вход OFDM-модулятора должны поступать информационные QPSK-символы (вектор комплексных отсчетов), а на выходе – IQ отсчеты OFDM-символа (вектор комплексных отсчетов).

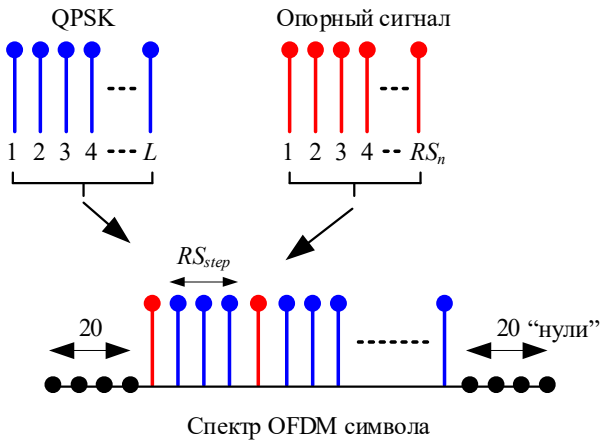


Рассмотрим подробнее блок модели. OFDM-модулятор включает в себя несколько процедур, которые последовательно описано далее:

1) Добавление опорного сигнала. Эта процедура алгоритм включает в себя последовательное выполнение следующих операций:

а) Расчет количества и индексов опорных поднесущих в зависимости от шага размещения, который обозначен здесь переменной ΔRS. В модели должен быть реализован алгоритм расположения опорных поднесущих между поднесущими с информационными данными. При этом,

период (или шаг) размещения опорных поднесущих ΔRS должен быть контролируемым пользователем, т.е. пользователь должен иметь возможность его изменять в программе. Этот алгоритм проиллюстрирован на рисунке справа. Общее количество опорных поднесущих в OFDM-символе обозначим переменной N_{RS}. Переменная N_{RS} напрямую зависит от ΔRS, потому что чем меньше шаг размещения, тем боле опорных поднесущих будет в OFDM-символе и, наоборот. Для расчета N_{RS} может быть использована следующая формула:



$$N_{RS} = \left\lfloor \frac{N_{QPSK}}{\Delta RS} \right\rfloor,$$

где N_{QPSK} – количество QPSK символов на выходе модулятора, а скобки $\lfloor \cdot \rfloor$ означают округление до ближайшего целого числа в меньшую сторону.

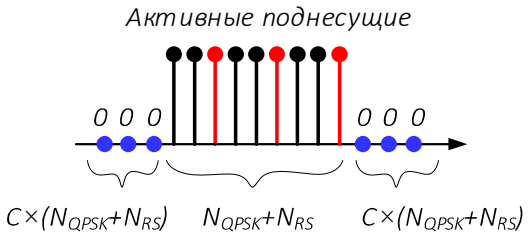
б) Формирование опорного сигнала. В качестве опорного сигнала могут использоваться любые комплексные сигнальные отсчеты количеством N_{RS} вида $I + jQ$, мощность которых нормирована на 1 Вт, т.е. $\sqrt{I^2 + jQ^2} = 1$. Например, для опорного сигнала можно сформировать вектор длиной N_{RS}, состоящий из одинаковых отсчетов $0.707 + j0.707$.

в) Расчет индексов опорных поднесущих. Индексы поднесущих для RS должны быть рассчитаны с учетом шага ΔRS, указанного пользователем. Например, если $\Delta RS = 6$, то опорные поднесущие будут располагаться через 6 поднесущих, а в этом случае индексы опорных поднесущих будут: 1, 7, 13, …. Напоминаю, что общее количество индексов (как и опорных поднесущих) должно быть равно N_{RS}.

г) Размещение опорного сигнала, сформированного на этапе (б) по опорным поднесущим с индексами, рассчитанными на этапе (в).

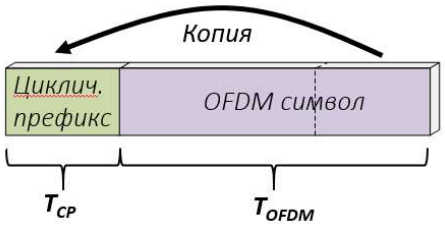
2) Добавление информационного сигнала. Чтобы реализовать эту процедуру, нужно рассчитать индексы поднесущих для информационного сигнала.

3) Добавление нулевого защитного интервала слева и справа от информационной части. Задача защитного интервала – борьба с внеполосным излучением. Требуется добавить в спектр OFDM-символа нулевые поднесущие слева и справа от активных поднесущих (которые используются для передачи информационного и опорного сигнала). Количество нулевых поднесущих N_Z с каждой стороны определяется параметром C по формуле $N_Z = \mathrm{C}(N_{RS} + N_{QPSK})$. Параметр C отличается для разных вариантов задания. Например, если $\mathrm{C} = 1/4,\ N_{RS} = 10, N_{QPSK} = 90$ то количество нулей слева и справа будет $N_Z = 25$.



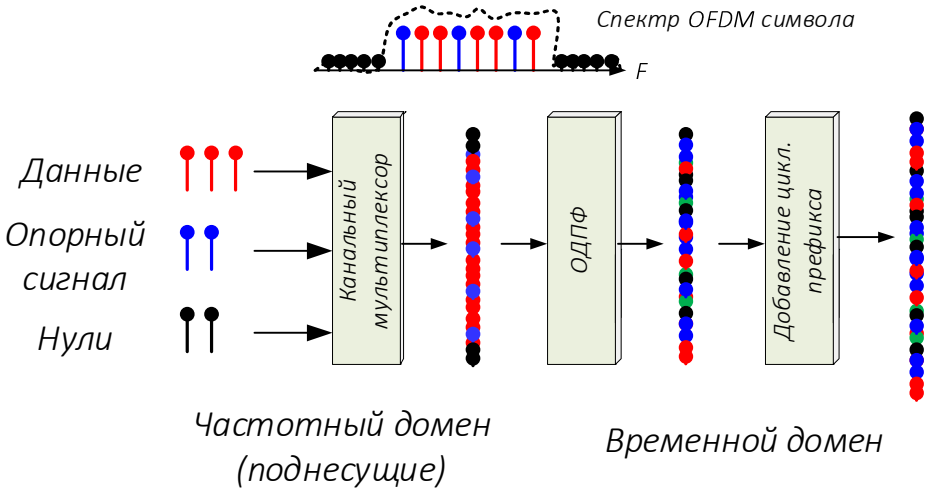
4) Обратное дискретное преобразование Фурье над всеми поднесущими для формирования OFDM-символа во временном домене. Размерность ОДПФ должна быть равна общему количеству всех поднесущих (включая нулевые, опорные и информационные).

5) Добавление циклического префикса. Задача циклического префикса – борьба с межсимвольной интерференцией. Циклический префикс является копией отрезка конца OFDM-символа длительностью T_{CP}, помещенной в начало OFDM символа. При этом не нарушается ортогональность за счет свойств ДПФ.



Сигнал на выходе блока OFDM-модулятора должен являться вектором с комплексными отсчетами длиной $T_{CP} + N_{RS} + N_{QPSK} + 2N_Z$.

Процедура формирования OFDM-символа визуализирована на рисунке ниже. Здесь канальный мультиплексор отвечает за расстановку сигналов (нулей, опорных сигналов и сигналов с данными) по поднесущим в частотном домене.



Описание входных/выходных переменных

Входные переменные:	вектор с комплексными символами
Выходные переменные:	вектор с комплексными символами
	параметры OFDM-символа
Параметры:	ΔRS -шаг опорных поднесущих
	T – размер циклич. Префикса

Практика 6. Модель канала передачи

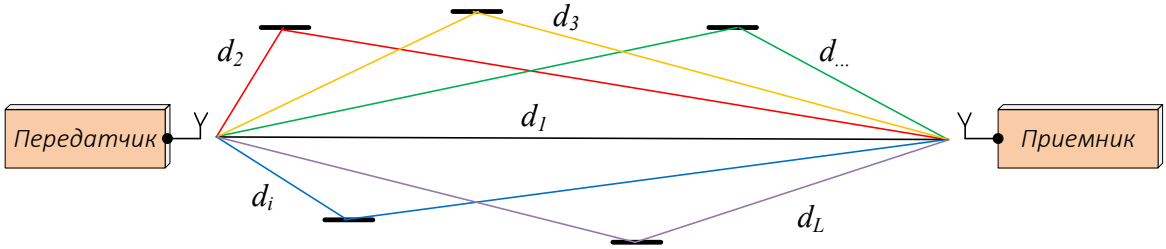
Задание

Реализовать модель многолучевого канала передачи с добавлением аддитивного белого гауссова шума заданной мощности.

[……]

Описание модели многолучевого канала передачи

Модель многолучевого канала передачи включает в себя несколько путей (лучей) распространения сигнала, которые складываются на входе приемника. Задача модели канала передачи состоит в имитации искажений, которые происходят при передаче сигнала в условиях сложной местности (например, городской застройки), в которой появляется эффект многолучевости.



Количество лучей определим параметром N_B. Длину сигнала с выхода передатчика обозначим символом L. В свою очередь, каждый луч имеет протяженность D (в метрах), исходя из которой рассчитываются величины затухания G и задержки τ для каждого луча. Примем условие, что параметр D выбирается случайным образом в диапазоне от 10 до 500 метров при каждом запуске симуляции (с помощью функции рандомайзера). Луч с минимальным D будет считается прямым, а его индекс в модели примем $i = 1$. Тогда, задержка τ_i (в дискретных отчетах) прихода i-го луча относительно прямого луча может быть найдена по формуле

$$\tau_i = \frac{(D_i - D_1)}{cT_s}, i = 1,2, \dots, N_B$$

где с – скорость света, а T_s – длительность дискретного отсчета в модели, которая может быть найдена $T_s = 1/B$, где B – полоса информационного сигнала, которая задается вариантом Вашего задания.

После расчета, величина τ_i должна быть округлена до ближайшего целого значения. В математической модели одним из вариантов реализации задержки в i -м луче может быть добавление целого количества τ_i нулей перед сигнальным отчетами:

$$S_i(k) = 0, \text{если } k \leq \tau_i$$

$$S_i(k) = s_{tx}(k - \tau_i), \text{если } k > \tau_i$$

где $i = 1,2, \dots, N_B$ – номер луча, $k = 1,2, \dots, L + \tau_i$ – номер отсчета в сигнальном векторе, S_{tx} – сигнал с выхода передатчика.

Далее, рассчитаем коэффициент ослабления для всех лучей. Для расчета коэффициента ослабления сигнала в i-м луче используется простейшая формула ослабления сигнала в вакууме

$$G_i = \frac{c}{4\pi D_i f_0}$$

где f_0 – это несущая частота сигнала, которая задается Вашим вариантом.

Для внесения ослабления в сигнал требуется умножить весь сигнальный вектор S_i на коэффициент ослабления G_i.

После реализации задержки и ослабления сигнала во всех лучах можно получить общий сигнал на выходе канала передачи (на входе приемника). Для этого требуется просуммировать сигнальные вектора друг с другом. Обозначим сигнал на выходе многолучевого канала переменной S_{mpy} и рассчитаем его

$$S_{mpy} = \sum_{i=1}^{N_B} G_i S_i$$

На этом можно считать, что часть модели добавления мультипликативной помехи можно считать завершенной. Далее, реализуем добавление шума.

Аддитивный белый гауссов шум $\mathbf{n}$ накладывается на сигнал $\mathbf{S_{mpy}}$ с помощью сложения шумового вектора с этим сигналом. Далее, ознакомимся с примером процедуры формирования шумового вектора $\mathbf{n}$ в модели. Для этого может быть использована встроенная функция wgn. Пример формирования вектора с шумом представлен ниже

$$\mathbf{n} = \boldsymbol{wgn}(\boldsymbol{M}, \mathbf{1}, \boldsymbol{N_0}),$$

где M – длина вектора с шумом, а N_0 – спектральная плотность мощности шума в дБ. Учитывайте, что длина $\mathbf{n}$ должна совпадать с длиной $\mathbf{S_{mpy}}$.

Для управления мощностью шума в модели мы используем переменную спектральной плотности мощности шума N_0 (дБ), которая должна задаваться пользователем в качестве входного параметра. Добавление шума на сигнал реализуется с помощью формулы

$$\mathbf{S_{rx}} = \mathbf{S_{mpy}} + \mathbf{n}$$

Сигнальный вектор на выходе блока многолучевого канала должен иметь такую же длительность, как и на входе (потому что приемник обрабатывает сигнал идентичной длительности). Чтобы это осуществить, можно “вырезать” из сигнального массива $\mathbf{S_{rx}}$ первые L отчетов и подать их на выход модели канала передачи.

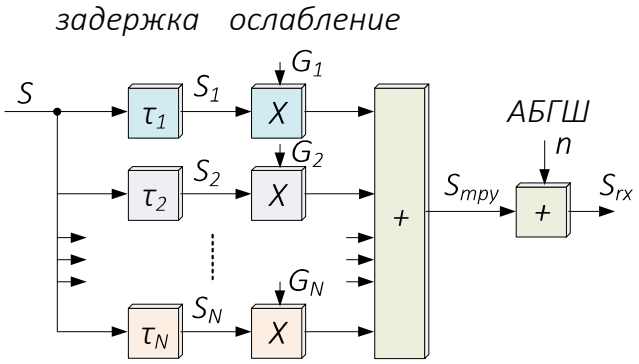
В этой модели пользователем будут задаваться параметры, которые будут влиять на тип многолучевого канала:

- Количество лучей в многолучевой модели, параметр N_B
- Мощность АБГШ в дБ, параметр N_0

Вместе с этим, для математического расчета в модели будут использованы следующие константы:

- Скорость света $\mathrm{c} = 3 \cdot 10^8$ м/с
- Несущая частота, Гц: задается вариантом из таблицы
- Полоса сигнала, Гц задается вариантом из таблицы

Структурная схема модели многолучевого канала передачи с АБГШ приведена на рисунке ниже.



Описание входных/выходных переменных

Входные переменные:	вектор с комплексными символами
Выходные переменные:	вектор с комплексными символами
Параметры:	N_B – количество лучей
	N_0 – мощность АБГШ

Практика 7. Эквалайзирование, OFDM демодуляция

Задание

Реализовать модель демодуляции OFDM-символа с последующей оценкой АЧХ канала и эквалайзированием.

[.....]

Описание работы математической модели

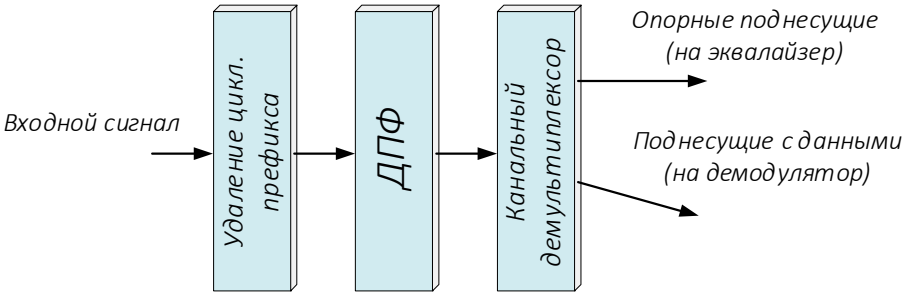
В задачу этого блока модели входит OFDM-демодуляция принятого искаженного сигнала, компенсация искажений и демодуляция принятого сообщения. Опишем последовательно каждую операцию.

1) OFDM-демодуляция принятого сигнала. Сигнал с выхода модели канала передачи поступает на вход модели приемника. Для OFDM-модуляции требуется совершить три последовательных операции, обратные операциям OFDM-модуляции:

а) Удаление циклического префикса длительностью T отсчетов из входного сигнала.

б) Прямое дискретное преобразование Фурье сигнала после удаления циклического префикса.

в) Удаление защитных нулевых интервалов слева и справа в спектре OFDM-символа.

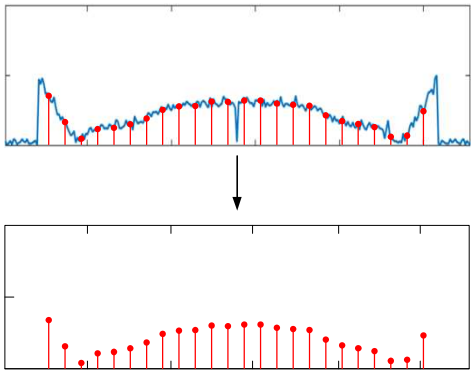


2) Обозначим переменной **C** сигнальный вектор, содержащий опорные поднесущие и поднесущие с данными. Далее, требуется выделить опорные сигналы из C и оценить с помощью них АЧХ канала передачи. Для этого нужно сделать последовательные операции:

а) В приемнике известны номера поднесущих, на которых передаются опорные сигналы. Требуется выделить значение сигнала на

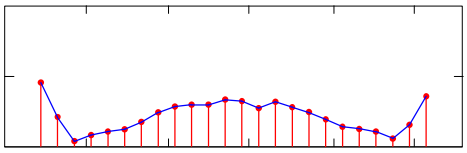
этих поднесущих. Обозначим вектор с принятыми опорными сигналами переменной $\mathbf{R}_{rx}$.

б) Для того, чтобы произвести оценку АЧХ канала передачи требуется разделить сигнальный вектор идеального опорного сигнала (который был в передатчика, обозначим его за $\mathbf{R}_{tx}$) на $\mathbf{R}_{rx}$. Обозначим $\mathbf{H}$ – вектор с АЧХ канала, тогда

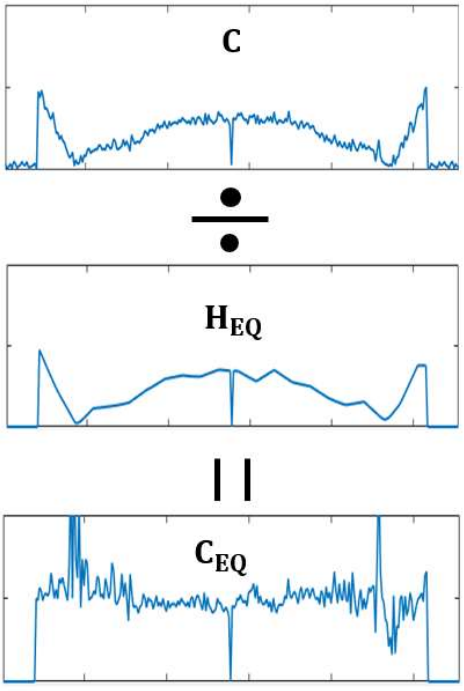


$$\mathbf{H} = \frac{\mathbf{R}_{rx}}{\mathbf{R}_{tx}}$$

в) Далее, требуется произвести интерполяцию полученной оценки $\mathbf{H}$ на весь спектр. Для этого может быть использована встроенная функция. Метод интерполяции – линейный.



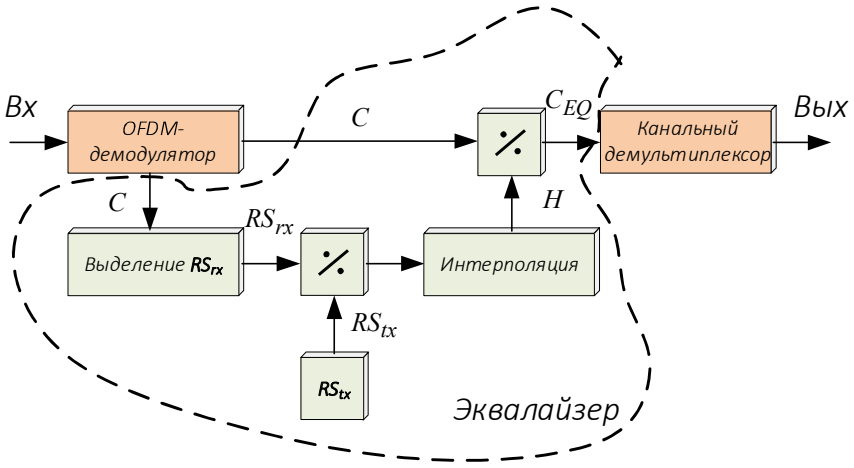
3) После интерполяции нужно произвести эквалайзирование, т.е. выравнивание спектра и компенсацию искажений. Математически, операция эквалайзирования очень простая – нужно спектральные отсчеты принятого сигнала $\mathbf{C}$ разделить на корректирующую АЧХ эквалайзера $\mathbf{H}_{\mathbf{EQ}}$. Обозначим переменной $\mathbf{C}_{\mathbf{EQ}}$ – спектральные отсчеты после эквалайзирования, тогда



$$\mathbf{C}_{\mathbf{EQ}} = \frac{\mathbf{C}}{\mathbf{H}_{\mathbf{EQ}}}$$

4) После эквалайзированияа, нужно извлечь принятые QPSK-символы с поднесущих, которые предназначены для передачи данных. Это и будет сигнал на выходе блоке OFDM-демодулятора.

Общая структурная схема модели OFDM-демодулятора представлена на рисунке ниже.



Описание входных/выходных переменных

Входные переменные:	вектор с комплексными символами с выхода канала передачи параметры OFDM-символа (из блока модулятора)
Выходные переменные:	вектор с комплексными символами (QPSK)
Параметры:	нет

Связь блоков модели

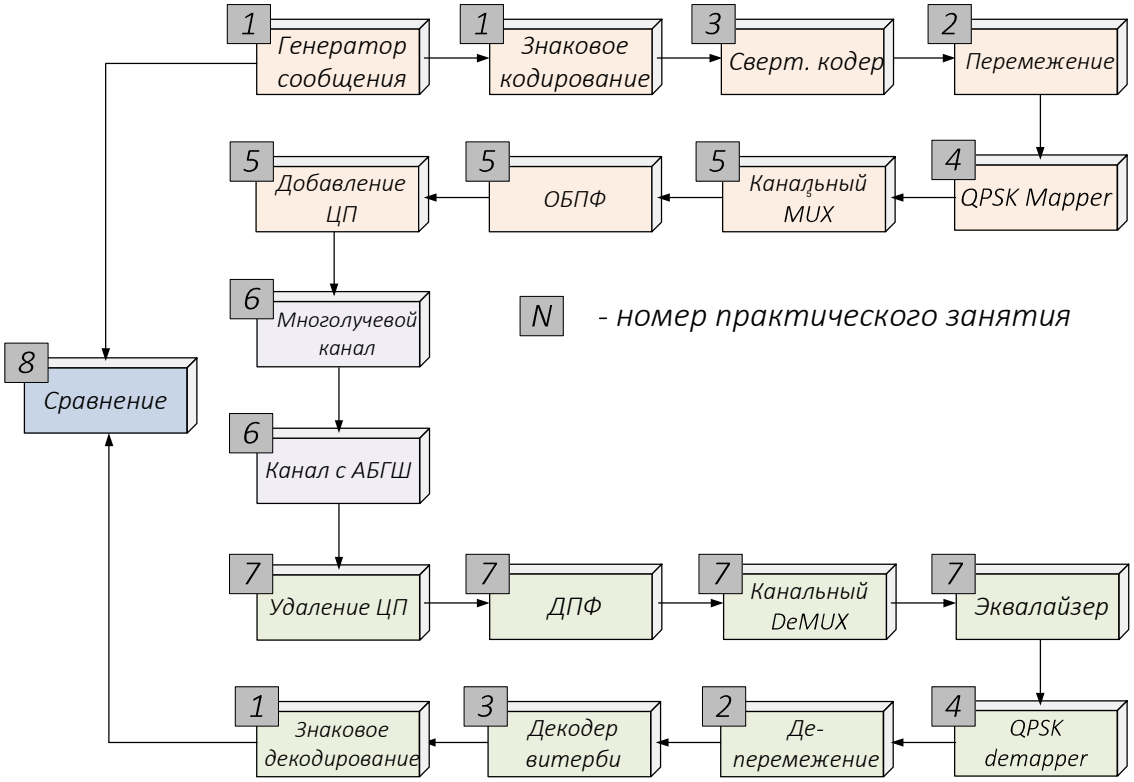
По завершению 7-й практики можно соединить все модельные блоки в последовательном, правильном порядке. Структурная схема модели продублирована на рисунке ниже. Логика выполнения программного кода модели должна соответствовать этой структурной схеме.

1) Сигнал с выхода генератора сообщений должен поступать на вход блока знакового кодера.

2) Сигнал с выхода знакового кодера должен поступать на вход блока сверточного кодирования.

3) Сигнал выхода сверточного кодера должен поступать на вход блока перемежителя.

4) и т.д.



Практика 8. Расчет BER. Построение графиков

Задание

Реализовать модель расчета вероятности битовых ошибок, построение графиков и осуществить имитационное моделирование.

[.....]

Описание работы математической модели

Расчет вероятности битовых ошибок требуется произвести после декодирования сигнала в приемнике, т.е. сравниваются битовые сообщения после знакового кодера (в модели передатчика) и перед знаковым декодером (в модели приемника). Формула для расчета вероятности битовых ошибок (BER):

$$BER = \frac{количество\ ошибочных\ бит}{общее\ количество\ бит}$$

Пример графического окна приведен на рисунке ниже. На графиках должны быть представлены следующие зависимости:

1) Спектр переданного OFDM-символа

2) Спектр принятого OFDM-символа до эквалайзирования $\mathbf{C}$

3) Спектр принятого OFDM-символа после эквалайзирования $\mathbf{C_{EQ}}$

4) Сигнальное созвездие QPSK-сигнала в передатчике

5) Сигнальное созвездие QPSK-сигнала в приемнике

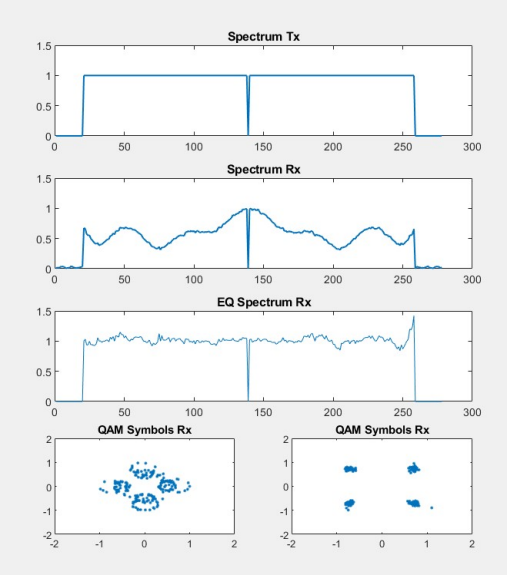


Таблица с вариантами и комбинациями

N – номер вашего ФИО в списке группы. Зная N, нужно выбрать комбинацию заданий из таблицы с комбинациями. Далее, выбрать параметры модели из таблицы с вариантами в соответствии со своей комбинацией.

Например, Ваша комбинация 1,2,3,4,5,6. Тогда, ваш вариант задания будет следующий:

длина сообщения = 35 символов, $N_B = 7$, $T_{CP} = 1/16$, $\Delta RS = 6$ В = 13 МГц, $f_0 = 1{,}9$ГГц.

Таблица с комбинациями

N	1	2	3	4	5	6	7	8	9
Комб.	1,2,3, 4,5,6	2,4,6, 1,7,2	6,2,3, 2,2,2	7,7,2 1,2,3	5,3,6 1,3,4	2,2,1, 1,3,3	5,2,4 1,5,3	3,3,3 2,2,2	1,1,1 1,1,1
N	10	11	12	13	14	15	16	17	18
Комб.	6,5,4, 2,1,3	7,4,2, 1,2,3	7,7,7 1,2,3	4,3,4, 1,2,4	7,6,2, 5,5,5	2,5,1 5,5,5	2,3,1 6,7,7	3,2,2 2,2,1	5,4,1 2,2,2

Таблица с вариантами

N	1	2	3	*4*	*5*	*6*
	Длина сообщения, симв	Кол-во лучей N_B	Длина префикса, T_{CP}	Шаг опорных поднсущих, ΔRS	Полоса, B, МГц	Несущая, f_0 ГГц
1	35	5	½	3	10	2
2	40	7	¼	4	11	2.1
3	45	6	1/16	5	8	1.7
4	50	9	1/8	6	9	2.4
5	55	10	¼	7	13	2.2
6	60	4	½	8	12	1.9
7	65	8	1/8	9	7	1.8

Скрипт декодера Витерби для Octave

```
function [vb] = ConvDecoder(input_bits)

k = 7;
G1 = 171;
G2 = 133;
length_dibits = length(input_bits )/2;
[yzli, ves,add_to_metr,pyt,in_delay_pipe] = genyzl(k,G1,G2);
nves=length(ves);

vb=[];
pyti=[];
kb=1;

for i=1:length_dibits
    dibits(i)={[input_bits(2*i-1),input_bits(2*i)]};
end

metk=[];

for n=1:length_dibits

    met1=[];
    met2=[];

    for i=1:nves
        met1=[ met1 sum(xor(dibits{n}, ves{1,i}))];
        met2=[ met2 sum(xor(dibits{n}, ves{2,i}))];
    end
    met=[met1;met2];
    metk=[metk; met];

    % считаем метрики путей
    for i=1:2
        for j=1:nves
            sw(i,j)=met(i,j)+add_to_metr(pyt(i,j));
        end
    end

    add_to_metr = min(sw);
    pyti=[pyti; pyt]; % добавляются все пути, потом чистятся

    for k=1:nves
        if (sw(1,k) <= sw (2,k) )
            pyti(2*n,k)=0; %обнуляем один из путей
        else
            pyti(2*n-1,k)=0;
        end
    end

    kp=n;  %
    while kp>kb
        for k=1:nves
            if (sum([pyti(2*kp-1,:)==k pyti(2*kp,:)==k])==0)
                pyti(2*kp-2,k)=0;
                pyti(2*kp-3,k)=0;
            end
        end
        kp=kp-1;
    end

    %проверяем единственность пути
    p1=length(find(pyti(2*kb-1,:)));
    p2=length(find(pyti(2*kb,:)));
    if ((p1==1) && (p2==0)) || ((p1==0) && (p2==1));
        nb1=find(pyti(2*kb-1,:)); nb2=find(pyti(2*kb,:));
        if(p1==1) && (p2==0); vb=[vb in_delay_pipe(1,nb1)]; kb=kb+1; end;
        if(p1==0) && (p2==1); vb=[vb in_delay_pipe(2,nb2)]; kb=kb+1; end;
```

```
    end;

    if (n == length_dibits)
        min_end_metr=min(min(sw));

        for i=1:2
            for j=1:nves
                if(sw(i,j))==min_end_metr
                    end_way=pyti(2*length_dibits,j)+pyti(2*length_dibits-1,j);
                    vb(length_dibits)=in_delay_pipe(i,j); %декидируем
                end
            end
        end
          delt=length_dibits - kb+1;        %сколько бит не декодировали
            for d=delt:-1:2;                %идем с конца и декодируем неизвестные биты
                 vb(kb-2+d) = in_delay_pipe(1,end_way);
                 end_way = pyti(2*(kb-2)+2*d ,end_way ) + pyti(2*(kb-2)+2*d-1 ,end_way );
            end
    end
end
```

Скрипт доп. Функции к декодеру Витерби для Octave

```
function [ yzli, ves,add_to_metr,pyt,in_delay_pipe ] = genyz1;

k = 7;
G1 = 171;
G2 = 133;


G1=dec2bin(oct2dec(G1));
G1=str2num(reshape(G1,[],1))';
G2=dec2bin(oct2dec(G2));
G2=str2num(reshape(G2,[],1))';

%%формируем параметры декодирования
yzli=[];

for i=0:2^(k-1)-1

    yzli=[yzli {dec2bin(i,k-1)}];
end
% yzli    =  [{'00'} {'10'} {'01'} {'11'} ];


% формируем pyt
for i = 1:2^(k-1)
    yzel=cell2mat(yzli(i));
    yzel1=yzel(2:k-1) ;
%     oneorzero=yzel(1);
d=0;
    for j=1:2^(k-1)
    yzel=cell2mat(yzli(j));
    yzel2=yzel(1:k-2) ;
        if yzel1 == yzel2
            d=d+1;
            if d==1
                pyt(1,i)=j;
            else
                pyt(2,i)=j;
            end
        end
    end
end

%формируем веса
for i=1:2^(k-1)
    nomer_pred=pyt(1,i);
    yzel_pred=cell2mat(yzli(nomer_pred));

    yzel=cell2mat(yzli(i));
    yzel=yzel(1);
    if yzel=='1'   yzel_mat(1)=1;
     else          yzel_mat(1)=0;    end

    for j=1:k-1 %в числовую форму
        if yzel_pred(j)=='1'   yzel_mat(j+1)=1;
        else              yzel_mat(j+1)=0;    end
    end

    out_s1=0;
    out_s2=0;
    for j=1:k %само кодирование
        out_s1  = out_s1 + G1(j)*yzel_mat(j);
        out_s2  = out_s2 + G2(j)*yzel_mat(j);
    end
    out1      =   mod(out_s1,2);
    out2        =   mod(out_s2,2);
    ves (1,i) = {[ out1 , out2 ]}; %то что на выходе если 0
```

```
        nomer_pred=pyt(2,i);
    yzel_pred=cell2mat(yzli(nomer_pred));

    yzel=cell2mat(yzli(i));
    yzel=yzel(1);
    if yzel=='1'   yzel_mat(1)=1;
     else          yzel_mat(1)=0;    end

    for j=1:k-1 %в числовую форму
        if yzel_pred(j)=='1'   yzel_mat(j+1)=1;
        else              yzel_mat(j+1)=0;    end
    end

    out_s1=0;
    out_s2=0;
    for j=1:k %само кодирование
        out_s1  = out_s1 + G1(j)*yzel_mat(j);
        out_s2  = out_s2 + G2(j)*yzel_mat(j);
    end
    out1      =   mod(out_s1,2);
    out2        =   mod(out_s2,2);
    ves (2,i) = {[ out1 , out2 ]}; %то что на выходе если 0

end

% добавляем метрики к неправдоподобным путям
add_to_metr = [0];
for i = 1:2^(k-1)-1
    add_to_metr=[add_to_metr 20];
end

% формируем инделайпайп
for i = 1:2^(k-1)
    yzel=cell2mat(yzli(i));
        if yzel(1)=='1'   in_delay_pipe(1,i)=1; in_delay_pipe(2,i)=1;
        else              in_delay_pipe(1,i)=0; in_delay_pipe(2,i)=0;   end
end
```