*федеральное государственное бюджетное образовательное учреждение высшего образования*

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | ***«Московский государственный технический университет  имени Н. Э. Баумана»***  ***(МГТУ им. Н. Э. Баумана)*** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | Информатики и систем управления |
|  |  |
| КАФЕДРА | Проектирования и технологии производства ЭА |

**ДОМАШНЕЕ ЗАДАНИЕ**

по курсу          Цифровая обработка сигналов

на тему                    Разработка MSK-модулятора/демодулятора на языке Python

|  |  |  |
| --- | --- | --- |
| Студент |  | Н. Р. Ахметов |
|  | (Подпись, дата) | (И.О.Фамилия) |
| Преподаватель |  | В. В. Леонидов |
|  | (Подпись, дата) | (И.О.Фамилия) |
| Отметки о сдачи домашнего задания: |  |  |
|  |  |  |

Москва, 2020

Оглавление

[Введение 3](#__RefHeading___Toc1253_3519493166)

[Теория 3](#__RefHeading___Toc1255_3519493166)

[Алгоритм работы программы 5](#__RefHeading___Toc1257_3519493166)

[Графическое отображение результатов работы программы 6](#__RefHeading___Toc1259_3519493166)

[Исходный код 7](#__RefHeading___Toc1261_3519493166)

# Введение

В рамках данного домашнего задания был разработан MSK-модулятор/демодулятор. Среда реализации вычислений ­ язык программирования Python. Среда отображения информации ­ пакет программ Matlab.

# Теория

MSK-тип модуляции, при котором изменяется частота несущего сигнала в зависимости от передаваемого сообщения. Сообщения подлежащее модуляции и сам модулированный сигнал показаны на рисунках 1 и 2 соответственно.

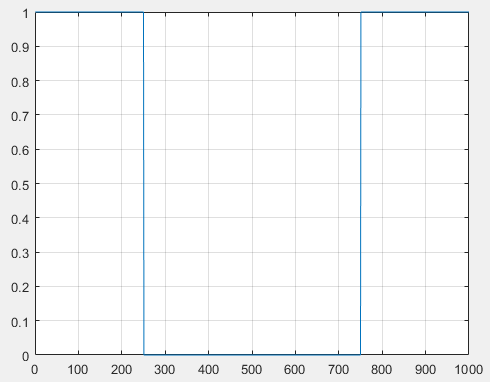


Рисунок Х ­цифровое сообщение, подлежащее модуляции (**1 0 0 1**, заданный период ­ 250 мС)

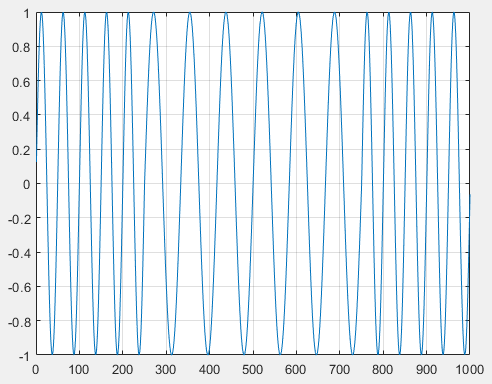


Рисунок Х ­ Модулированный сигнал

На рисунке Х закодирована цифровая последовательность «1 0 0 1». Поскольку это двоичный код, данным методом его можно закодировать с помощью двух синусоид с различными частотами: мы задаем сообщение, которое необходимо смодулировать, задаем временной отрезок, на котором нужно, чтобы было распределено сообщение, в зависимости от этого вычисляются частоты двух несущих синусоид и после этого происходит модуляция.

# Алгоритм работы программы

Чтобы понять, как работает алгоритм MSK-модулятора/демодулятора, рассмотрим его структурную схему в Simulink:

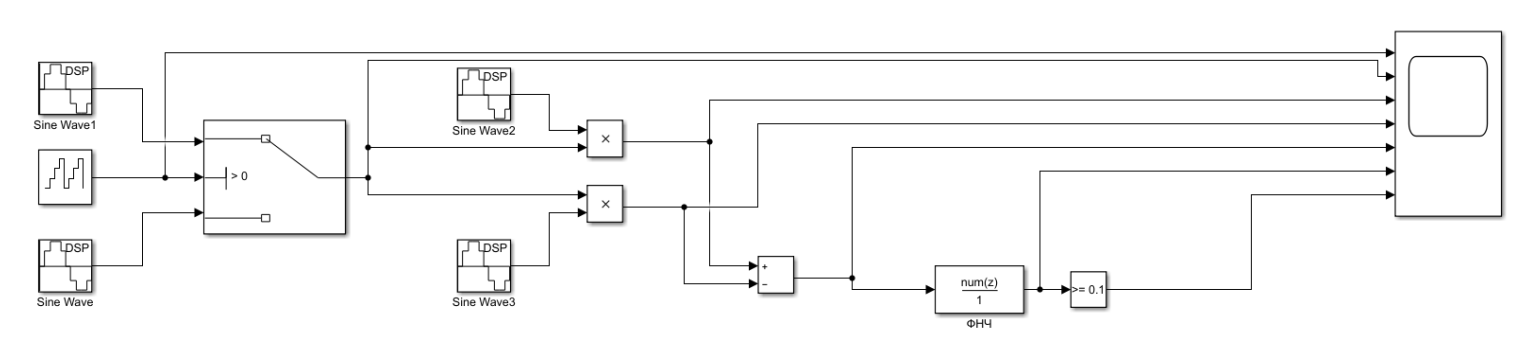


Рисунок Х ­ структурная схема MSK-модулятора/демодулятора в Simulink

Запускаем программу и вводим конфигурационные данные: файл БД для записи результатов, длину сообщения, период сигнала, и сами элементы сообщения.

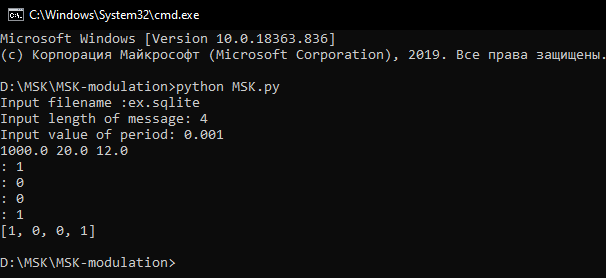


Рисунок Х ­ запуск программы на исполнение

После получения цифрового сообщения «1 0 0 1», мы должны его растянуть по всему временному отрезку (**48-53**): создаем список и заполняем его значения сообщения, растянутыми по всей длительности передачи одного бита. Далее формируется список, содержащй значения двух синусоид с разными частотами (**55-62**). Далее происходит модуляция (**65-70**): если в msg[i] находится «1», то в модулированном сообщении mod\_msg[i] будет частота f0, иначе f1.

Далее необходимо провести демодуляцию модулированного сигнала. Создаем два списка, в которые помещаем результат умножения модулированного сигнала на две несущие синусоиды и поэлементное вычитание второго списка из первого (**73-77**). После этого фильтруем сигнал с помощью скользящего среднего на 90 элементов (**86-123**) и пропускаем через триггер по уровню 0.2 (**126-133**).

Далее результаты необходимо отобразить в Matlab. В качестве элемента временного хранения передаваемых данных была выбрана БД **SQLITE** ‑ легковесная быстрая база данных, обычно применяемая для хранения данных мобильными приложениями.

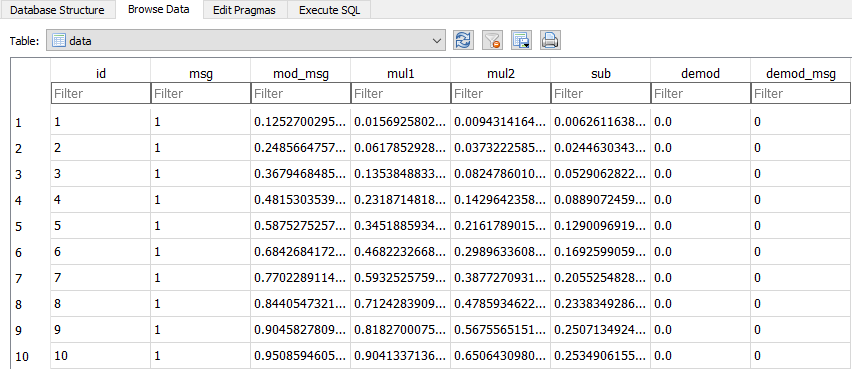


Рисунок Х ­ просмотр хранимых данных в таблице data БД ex.sqlite

Чистим содержимое БД, создаем таблицу data, в которой будем хранить модулированную и демодулированную последовательности, а также промежуточные результаты (**136-149**). Далее заполняем БД и закрываем соединение (**136-149**).

После этого открываем матлаб и запускаем скрипт **readFromFile.m**. Открываем БД (**4**), считываем данные (**6**), преобразуем их из типа данных «ячейка» в численный (double или int64) и отображаем (**9-30**).

# Графическое отображение результатов работы программы

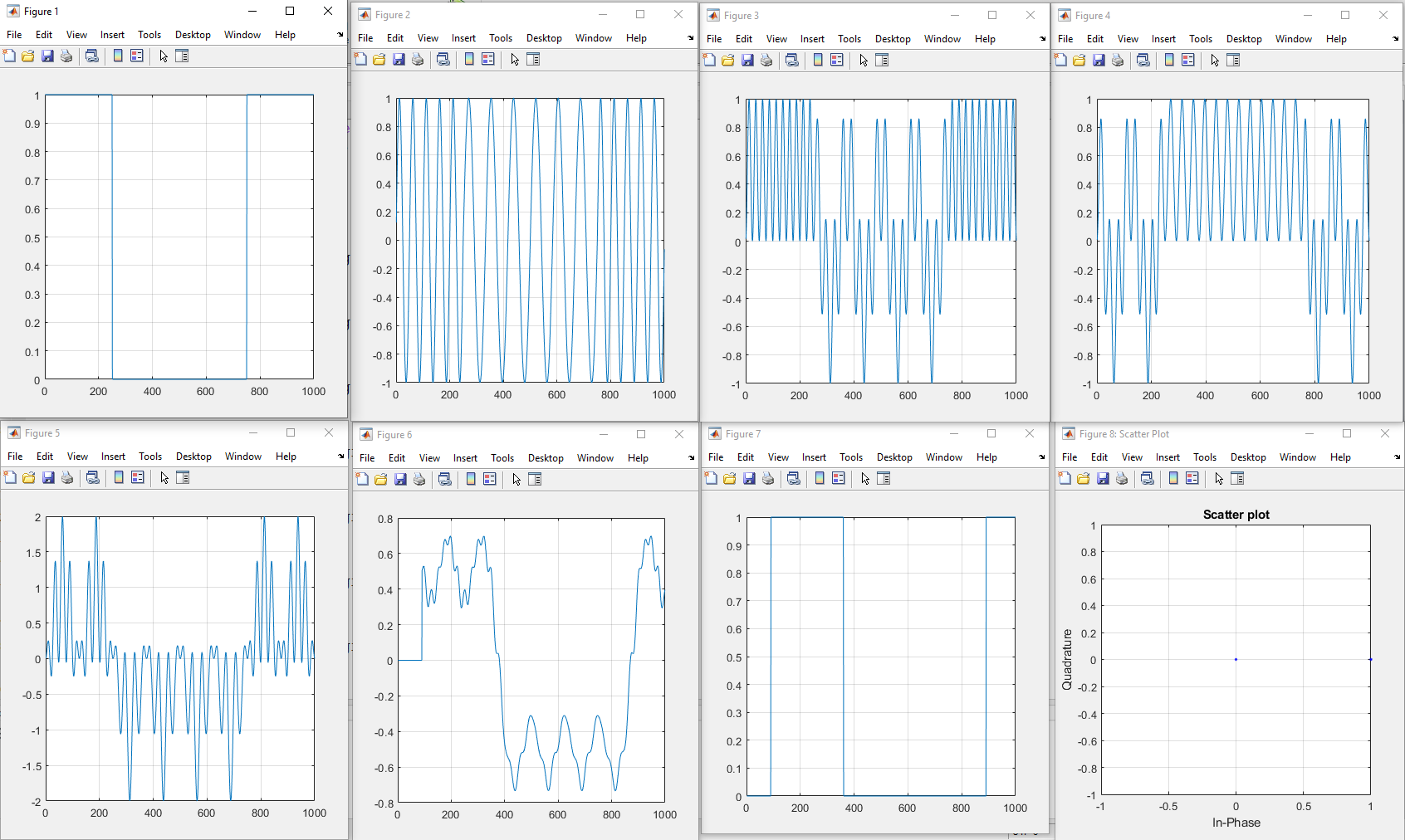


Рисунок Х ­результаты работы программы

# Исходный код

**Листинг 1 ­MSK.py**

|  |
| --- |
| **1 import math**  **2 import sqlite3**  **3**  **4 inp\_msg = list()**  **5 msg = list()**  **6 ts = list()**  **7 x0 = list()**  **8 x1 = list()**  **9 mod\_msg = list()**  **10 mul1 = list()**  **11 mul2 = list()**  **12 sub = list()**  **13 demod = list()**  **14 i = 0**  **15 pi = 3.14**  **16**  **17 fname = input("Input filename :")**  **18**  **19 try :**  **20 conn = sqlite3.connect('ex.sqlite')**  **21 except :**  **22 print("Something wrong.")**  **23 quit()**  **24**  **25 count = int(input("Input length of message: "))**  **26**  **27 if count <= 0 : quit()**  **28**  **29 t = float(input("Input value of period: "))**  **30 fs = 1 / t**  **31 f0 = fs \* 0.02**  **32 f1 = 0.6 \* f0**  **33**  **34 print(fs, f0, f1)**  **35**  **36 i = 0**  **37 while i < count :**  **38 bit = int(input(": "))**  **39 if bit == 0 or bit == 1 :**  **40 inp\_msg.append(int(bit))**  **41 i = i + 1**  **42 else :**  **43 print("Incorrect bit.")**  **44**  **45 print(inp\_msg)**  **46**  **47 i = 0**  **48 for bit in inp\_msg:**  **49 while i < fs / len(inp\_msg) :**  **50 msg.append(bit)**  **51 i = i + 1**  **52 i = 0**  **53**  **54 i = 0**  **55 while i < fs :**  **56 if len(ts) < 1 :**  **57 ts.append(1 / fs)**  **58 else :**  **59 ts.append(ts[i - 1] + (1 / fs))**  **60 x0.append(math.sin(2 \* pi \* f0 \* ts[i]))**  **61 x1.append(math.sin(2 \* pi \* f1 \* ts[i]))**  **62 i = i + 1**  **63**  **64 i = 0**  **65 for bit in msg :**  **66 if bit == 1 :**  **67 mod\_msg.append(x0[i])**  **68 else :**  **69 mod\_msg.append(x1[i])**  **70 i = i + 1**  **71**  **72 i = 0**  **73 while i < len(x0) :**  **74 mul1.append(mod\_msg[i] \* x0[i])**  **75 mul2.append(mod\_msg[i] \* x1[i])**  **76 sub.append(mul1[i] - mul2[i])**  **77 i = i + 1**  **78**  **79 i = 0**  **80**  **81 while i < 90 :**  **82 demod.append(0)**  **83 i = i + 1**  **84**  **85 i = 10**  **86 while i < len(sub) :**  **87 demod.append((sub[i - 1] + sub[i - 2] + sub[i - 3] 88 + sub[i - 4] + sub[i - 5] + \**  **89 sub[i - 6] + sub[i - 7] + sub[i - 8] + sub[i - 9] 90 + + sub[i - 10] + \**  **91 sub[i - 11] + sub[i - 12] + sub[i - 13] + sub[i – 92 14] + sub[i - 15] + \**  **93 sub[i - 16] + sub[i - 17] + sub[i - 18] + sub[i – 94 19] + sub[i - 20] + \**  **95 sub[i - 21] + sub[i - 22] + sub[i - 23] + sub[i – 96 24] + sub[i - 25] + \**  **97 sub[i - 26] + sub[i - 27] + sub[i - 28] + sub[i – 98 29] + sub[i - 30] + \**  **99 sub[i - 31] + sub[i - 32] + sub[i - 33] + sub[i – 100 34] + sub[i - 35] + \**  **101 sub[i - 36] + sub[i - 37] + sub[i - 38] + sub[i 102 - 39] + sub[i - 40] + \**  **103 sub[i - 41] + sub[i - 42] + sub[i - 43] + sub[i – 104 44] + sub[i - 45] + \**  **105 sub[i - 46] + sub[i - 47] + sub[i - 48] + sub[i 106 - 49] + sub[i - 50] + \**  **107 sub[i - 51] + sub[i - 52] + sub[i - 53] + sub[i 108 - 54] + sub[i - 55] + \**  **109 sub[i - 56] + sub[i - 57] + sub[i - 58] + sub[i 110 - 59] + sub[i - 60] + \**  **111 sub[i - 61] + sub[i - 62] + sub[i - 63] + sub[i – 112 64] + sub[i - 65] + \**  **113 sub[i - 66] + sub[i - 27] + sub[i - 68] + sub[i 114 - 69] + sub[i - 70] + \**  **115 sub[i - 71] + sub[i - 72] + sub[i - 73] + sub[i 116 - 74] + sub[i - 75] + \**  **117 sub[i - 76] + sub[i - 77] + sub[i - 78] + sub[i 118 - 79] + sub[i - 80] + \**  **119 sub[i - 81] + sub[i - 82] + sub[i - 83] + sub[i 120 - 84] + sub[i - 85] + \**  **121 sub[i - 86] + sub[i - 87] + sub[i - 88] + sub[i 122 - 89] + sub[i - 90]) / 89)**  **123 i = i + 1**  **124**  **125 i = 0**  **126 demod\_msg = list()**  **127 while i < len(demod)**  **129 if demod[i] > 0.2 :**  **130 demod\_msg.append(1)**  **131 else :**  **132 demod\_msg.append(0)**  **133 i = i + 1**  **134**  **135**  **136 c = conn.cursor()**  **137 c.executescript('''**  **138 DROP TABLE IF EXISTS data;**  **139 CREATE TABLE data(**  **140 id INTEGER PRIMARY KEY AUTOINCREMENT NOT 141 NULL,**  **142 msg INTEGER NOT NULL,**  **143 mod\_msg FLOAT NOT NULL,**  **144 mul1 FLOAT NOT NULL,**  **145 mul2 FLOAT NOT NULL,**  **146 sub FLOAT NOT NULL,**  **147 demod FLOAT NOT NULL,**  **148 demod\_msg INTEGER NOT NULL**  **149 )''')**  **150**  **151 i = 0**  **152 while i < len(msg) :**  **153 c.execute("INSERT INTO data (msg, mod\_msg, mul1, 154 mul2, sub, demod, demod\_msg) VALUES 155 (?, ?, ?, ?, 156 ?, ?, ?)", (msg[i], mod\_msg[i], mul1[i], mul2[i], 157 sub[i], demod[i], demod\_msg[i]))**  **158 i = i + 1**  **159 conn.commit()**  **160conn.close()** |

**Листинг 2 ­readFromFile.m**

|  |
| --- |
| **1 clc;**  **2 clear;**  **3**  **4 conn = sqlite('../ex.sqlite','readonly');**  **5 sqlquery = 'SELECT \* FROM data';**  **6 results = fetch(conn,sqlquery);**  **7 close(conn);**  **8**  **9 figure;**  **10 plot(cell2mat(results(:, 2))), grid on;**  **11**  **12 figure;**  **13 plot(cell2mat(results(:, 3))), grid on;**  **14**  **15 figure;**  **16 plot(cell2mat(results(:, 4))), grid on;**  **17**  **18 figure;**  **19 plot(cell2mat(results(:, 5))), grid on;**  **20**  **21 figure;**  **22 plot(cell2mat(results(:, 6))), grid on;**  **23**  **24 figure;**  **25 plot(cell2mat(results(:, 7))), grid on;**  **26**  **27 figure;**  **28 plot(cell2mat(results(:, 8))), grid on;**  **29**  **30 scatterplot (cell2mat(results(:, 8))), grid on;** |