南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Communication Systems Design

Lab 2: Packet Transmission

Dr. Wu Guang
wug@sustech.edu.cn
Electrical & Electronic Engineering
Southern University of Science and Technology

MATLAB面向对象编程

User-Defined Classes

What Is a Class Definition

A MATLAB® class definition is a template whose purpose is to provide a description of all the elements that are common to all instances of the class. Class members are the properties, methods, and events that define the class.

Define MATLAB classes in code blocks, with subblocks delineating the definitions of various class members. For syntax information on these blocks, see Class Components.

Attributes for Class Members

Attributes modify the behavior of classes and the members defined in the class-definition block. For example, you can specify that methods are static or that properties are private. The following sections describe these attributes:

- Class Attributes
- Method Attributes
- Property Attributes
- Event Attributes

Class definitions can provide information, such as inheritance relationships or the names of class members without actually constructing the class. See Class Metadata.

See Specifying Attributes for more on attribute syntax.

Kinds of Classes

There are two kinds of MATLAB classes—handle classes and value classes.

- Value classes represent independent values. Value objects contain the object data and do not share this data with copies of the object. MATLAB numeric types are value classes. Values objects passed to and modified by functions must return a modified object to the caller.
- Handle classes create objects that reference the object data. Copies of the instance variable refer to the same object. Handle objects passed to and modified by functions affect the object in the caller's workspace without returning the object.

For more information, see Comparison of Handle and Value Classes.

Constructing Objects

For information on class constructors, see Class Constructor Methods.

For information on creating arrays of objects, see Construct Object Arrays.

帮助

classdef

Documentation

CONTENTS Close

< All Products

< MATLAB

< Advanced Software Development

< Object-Oriented Programming

< Sample Class Implementations

< MATLAB

< Advanced Software Development

< Object-Oriented Programming

< Class Definition

< Class Definition and File Organization

< MATLAB

< Functions

classdef

ON THIS PAGE

Syntax

Description

Examples

See Also

classdef

Class definition keywords

Syntax

```
classdef classname
   properties
      PropName
   end
   methods
      methodName
   end
   events
      EventName
   end
   enumeration
      EnumName
   end
end
```

MyClass.m

```
1  classdef MyClass                          % 类名
2
3     properties
4        Prop                                % 静态属性
5     end
6
7     methods
8        function obj = MyClass(val)         % 构造函数
9           if nargin > 0
10             obj.Prop = val;               % 属性赋值
11          end
12       end
13    end
14
15 end
```

Description

`classdef` *`classname`* begins the class definition and an `end` keyword terminates the `classdef` block. Only blank lines and comments can precede `classdef`. Enter a class definition in a file having the same name as the class, with a filename extension of `.m`.

Class definition files can be in folders on the MATLAB® path or in class folders whose parent folder is on the MATLAB path. Class folder names begin with the `'@'` character followed by the class name (for example, `@MyClass`). For more information on class folders, see Class Files and Folders .

For more information on classes, see Classdef Block and Class Definition.

`properties` begins a property definition block, an `end` keyword terminates the `properties` block. Class definitions can contain multiple property definition blocks, each specifying different attribute settings that apply to the properties in that particular block.

For more information on properties, see Property Syntax.

Note: Properties cannot have the same name as the class.

`methods` begins a methods definition block, an `end` keyword terminates the `methods` block. This block contains functions that implement class methods. Class definitions can contain multiple method blocks, each specifying different attribute settings that apply to the methods in that particular block. It is possible to define method functions in separate files.

帮助

classdef

Documentation

CONTENTS Close

< All Products

< MATLAB

< Advanced Software Development

< Object-Oriented Programming

< Sample Class Implementations

< MATLAB

< Advanced Software Development

< Object-Oriented Programming

< Class Definition

< Class Definition and File Organization

< MATLAB

< Functions

classdef

ON THIS PAGE

Syntax

Description

Examples

See Also

classdef

Class definition keywords

Syntax

```
classdef classname
   properties
      PropName
   end
   methods
      methodName
   end
   events
      EventName
   end
   enumeration
      EnumName
   end
end
```

Description

`classdef` *classname* begins the class definition and an e
as the class, with a filename extension of `.m`.

Class definition files can be in folders on the MATLAB® pa
example, `@MyClass`). For more information on class folder

For more information on classes, see Classdef Block and

`properties` begins a property definition block, an `end` key
apply to the properties in that particular block.

For more information on properties, see Property Syntax.

Note: Properties cannot have the same name as the cla

`methods` begins a methods definition block, an `end` keywo
specifying different attribute settings that apply to the methods in that particular block. It is possible to define method functions in separate files.

the same name

ame (for

settings that

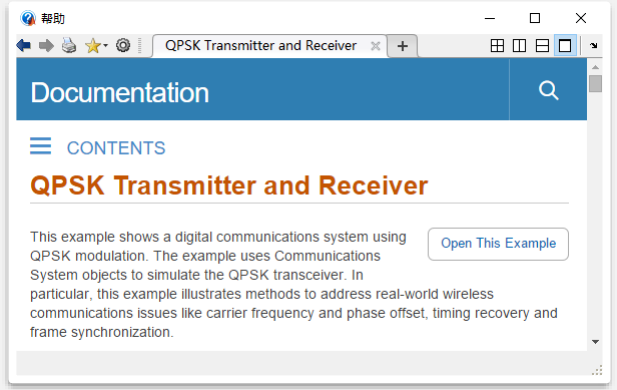
d blocks, each

BasicClass.m*

```
1  classdef BasicClass                                  % 类名称
2      properties
3          Value                                        % 类属性
4      end
5      methods
6          function obj = BasicClass(val)               % 构造函数
7              if nargin == 1
8                  if isnumeric(val)
9                      obj.Value = val;
10                 else
11                     error('Value must be numeric')
12                 end
13             end
14         end
15         function r = roundOff(obj)                   % 方法
16             r = round([obj.Value], 2);
17         end
18         function r = multiplyBy(obj,n)               % 方法
19             r = [obj.Value] * n;
20         end
21         function r = plus(o1, o2)                    % 方法
22             r = o1.Value + o2.Value;
23         end
24     end
25 end
```

Assignments (Lab1)

- Read the example '**QPSK Transmitter and Receiver**' in Communications System Toolbox.
- Explain the functions of the following six subcomponents respectively,
  - (1) Automatic Gain Control
  - (2) Coarse frequency compensation
  - (3) Fine frequency compensation
  - (4) Timing recovery
  - (5) Frame Synchronization
  - (6) Data decoder



- Implement '**16-QAM** Transmitter and Receiver' according to the example.
- Compare the BER between QPSK and 16-QAM under different EbN0 condition.

End-to-End Simulation

Modulation, error control coding, filtering, synchronization, link-level BER

Communications System Toolbox™ enables you to simulate link-level models of communications systems. Through bit error rate simulations, you can analyze system response to the noise and interference inherent in communication channels, explore what-if scenarios, and evaluate the tradeoffs between competing system architectures and parameters.

Error correction, interleaving, modulation, filtering, synchronization, equalization, and MIMO components provide the functionality to characterize communication systems.

Frequently Viewed Topics

- Digital Modulation
- QPSK Transmitter and Receiver
- End-to-End QAM Simulation with RF Impairments and Corrections
- Raised Cosine Filtering
- Introduction to MIMO Systems
- DVB-S.2 Link, Including LDPC Coding
- Parallel Concatenated Convolutional Coding: Turbo Codes

Sources
Input signals and sequences

Source Coding
Format signals for processing

Error Detection and Correction
Channel coding: block, Hamming, BCH, Reed-Solomon, LDPC, convolutional, CRC

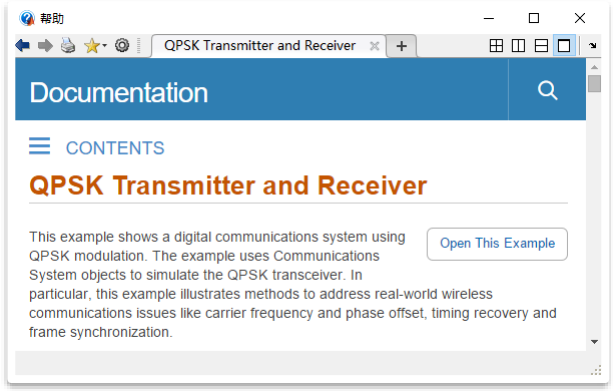
Signal Operations
Scrambling, puncturing, delay management, and bit operations

Interleaving
Block and convolutional interleaving

Modulation
Analog and digital modulation

Filtering
Filtering and pulse shaping

Synchronization and Receiver Design
Carrier frequency and phase recovery, timing frequency and phase recovery, AGC, I/Q imbalance compensation, phase-locked loops



MATLAB R2016a

主页 | 绘图 | 应用程序 | 编辑器 | 发布 | 视图

新建 打开 保存 查找文件 比较 打印 转至 查找 插入 注释 缩进 断点 运行 运行并前进 运行节 前进 运行并计时

C: ▸ Program Files ▸ MATLAB ▸ R2016a ▸ toolbox ▸ comm ▸ commdemos ▸

当前文件夹

名称

+slstart
codegen
html
private
animatescattereye.m
applyOFDMChannel.m
beacon_chan5_01_44MHz.mat
beacon_chan9_01_44MHz.mat
berResultsDVBS2Demo.mat
bersim.m
bertooltemplate.m
BERwithPCTExample.m
calculateOFDMBER.m
capturedc4fm.mat
cdma2000SimulinkData.mat
cdma2000SimulinkExample.slx
cdma2000WaveformGenerationExample.m
chandemo_802_16.m
chandemo_cost207.m
chandemo_hf.m
chandemo_multipathfading.m
channel_vis.m
checkCodegenLicenseOFDM.m
checkCodegenLicenseQAM.m
clockBERwithPCT.m
comm80211RxIcons.mat
comm_analyzeLogicFromSimulink.m
commacceleration.m
commaccelerationbaseline.m
commaccelerationreportresults.m
commaccelerationsystemobjects.m
commadaptivemimo.fig
commadaptivemimo.slx

详细信息

选择文件以查看详细信息

编辑器 - C:\Program Files\MATLAB\R2016a\toolbox\comm\commdemos\commQPSKTransmitterReceiver.m

commQPSKTransmitterReceiver.m

```
43    % other System objects in order to model larger components of the system
44    % under test
45
46    %% Initialization
47    % The <matlab:edit('commqpsktxrx_init.m') commqpsktxrx_init.m> script
48    % initializes simulation parameters and generates the structure prmQPSKTxRx.
49    prmQPSKTxRx = commqpsktxrx_init; % QPSK system parameters
50
51    useScopes = true; % true if scopes are to be used
52    printReceivedData = true; % true if the received data is to be printed
53    compileIt = false; % true if code is to be compiled
54    useCodegen = false; % true to run the generated mex file
55
56    %% Code Architecture for the System Under Test
57    % This example models a digital communication system using QPSK modulation.
58    % The function runQPSKSystemUnderTest models this communication
59    % environment. The QPSK transceiver model in this script is divided into
60    % the following four main compone
61    %
```

工作区

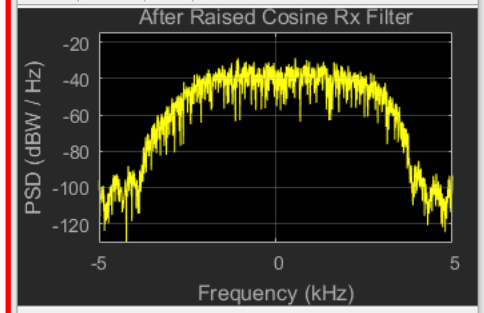
名称	值
BER	[0.0025;26;103...
compileIt	0
printReceiv...	1
prmQPSKTx...	*1x1 struct*
useCodegen	0
useScopes	1

命令行窗口

```
>> commQPSKTransmitterReceiver
Hd  p  Hi<  ld 000
Hello world 001
Hello world 002
Hello world 003
Hello world 004
Hello world 005
Hello world 006
Hello world 007
Hello world 008
```

Spectrum Analy...

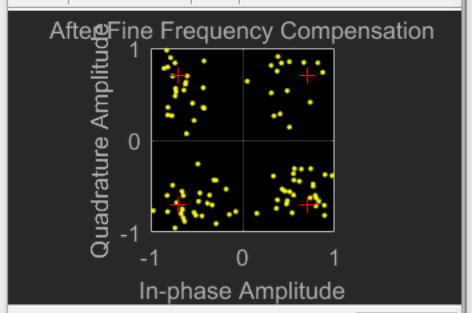
File Tools View Playback Help



Stopped | RBW=9.77 Hz | Sample Rate=10 kHz | T=1.98

Constellation Di...

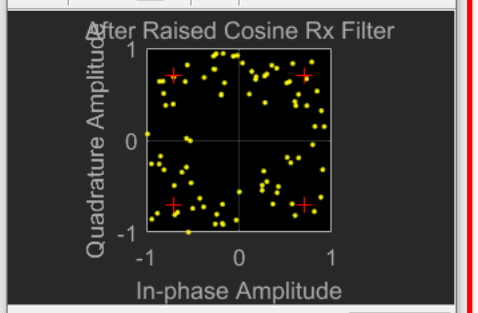
File Tools View Help



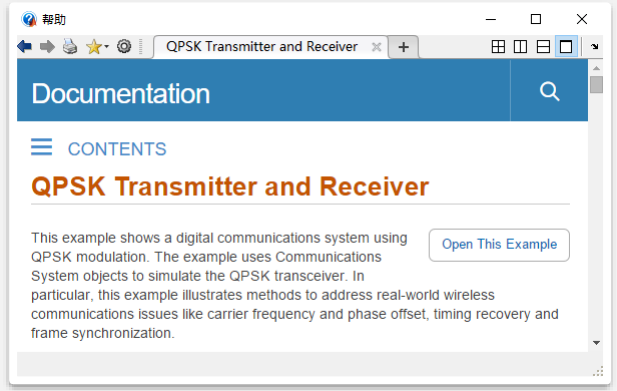
Stopped | Frame=100

Constellation D...

File Tools View Help



Stopped | Frame=100



commQPSKTransmitterReceiver.m

commqpsktxrx_init

runQPSKSystemUnderTest

General simulation parameters
Tx parameters
Channel parameters
Rx parameters

QPSKTransmitter

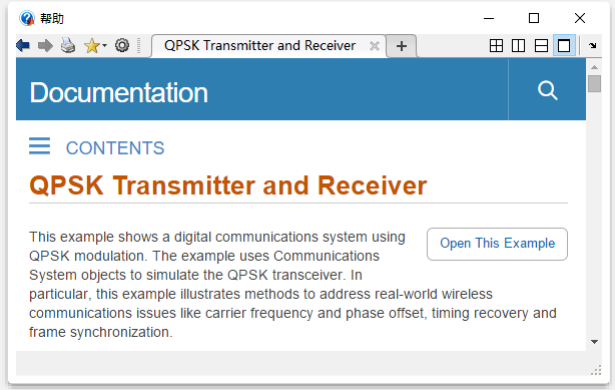
QPSKReceiver

QPSKChannel

createQPSKScopes

hTx
hRx
hChan
hScopes

for
step(hTx,…)
step(hRx,…)
step(hChan,…)
step(hScopes,…)
end



QPSKTransmitter

QPSKBitsGenerator

comm.QPSKModulator

dsp.FIRInterpolator

QPSKChannel

comm.PhaseFrequency Offset

dsp.VariableFractional Delay

comm.AWGNChannel

comm.CarrierSynchronizer

comm.SymbolSynchronizer

FrameFormation

QPSKReceiver

QPSKDataDecoder

comm.AGC

dsp.FIRDecimator

comm.PSKCoarseFrequ encyEstimator

comm.PhaseFrequencyOffset

QPSKTransmitter.m

```
classdef QPSKTransmitter < matlab.System
%#codegen
% Generates the QPSK signal to be transmitted

%   Copyright 2012 The MathWorks, Inc.

    properties (Nontunable)
        UpsamplingFactor = 4;
        MessageLength = 105;
        DataLength = 174;
        TransmitterFilterCoefficients = 1;
        ScramblerBase = 2;
        ScramblerPolynomial = [1 1 1 0 1];
        ScramblerInitialConditions = [0 0 0 0];
    end

     properties (Access=private)
        pBitGenerator
        pQPSKModulator
        pTransmitterFilter
    end
```

QPSKTransmitter.m

```
    methods
        function obj = QPSKTransmitter(varargin)
            setProperties(obj,nargin,varargin{:});
        end
    end

    methods (Access=protected)
        function setupImpl(obj)
            obj.pBitGenerator = QPSKBitsGenerator(...
                'MessageLength', obj.MessageLength, ...
                'BernoulliLength', obj.DataLength-obj.MessageLength, ...
                'ScramblerBase', obj.ScramblerBase, ...
                'ScramblerPolynomial', obj.ScramblerPolynomial, ...
                'ScramblerInitialConditions', obj.ScramblerInitialConditions);
            obj.pQPSKModulator  = comm.QPSKModulator('BitInput',true, ...
                'PhaseOffset', pi/4);
            obj.pTransmitterFilter = dsp.FIRInterpolator(obj.UpsamplingFactor, ...
                obj.TransmitterFilterCoefficients);
        end

        function transmittedSignal = stepImpl(obj)
            % Generates the data to be transmitted
            [transmittedData, ~] = step(obj.pBitGenerator);
```

.....

QPSKChannel.m

```
classdef QPSKChannel < matlab.System
%#codegen

%   Copyright 2012-2013 The MathWorks, Inc.


    properties (Nontunable)
        DelayType = 'Triangle';
        RaisedCosineFilterSpan = 10;
        PhaseOffset = 47;
        SignalPower = 0.25;
        FrameSize = 100;
        UpsamplingFactor = 4;
        EbNo = 7;
        BitsPerSymbol = 2;
        FrequencyOffset = 5000;
        SampleRate = 200000;
    end

    properties (Access=private)
        pPhaseFreqOffset
        pVariableTimeDelay
        pAWGNChannel
    end
```

QPSKChannel.m

```
    properties (Constant, Access=private)
        pDelayStepSize = 0.05;
        pDelayMaximum = 8;
        pDelayMinimum = 0.1;
    end

    methods
        function obj = QPSKChannel(varargin)
            setProperties(obj,nargin,varargin{:});
        end
    end

    methods (Access=protected)
        function setupImpl(obj, ~, ~)
            obj.pPhaseFreqOffset = comm.PhaseFrequencyOffset(...
                'PhaseOffset', obj.PhaseOffset, ...
                'FrequencyOffset', obj.FrequencyOffset, ...
                'SampleRate',obj.SampleRate);
            obj.pVariableTimeDelay = dsp.VariableFractionalDelay(...
                'MaximumDelay', obj.FrameSize*obj.UpsamplingFactor);
            obj.pAWGNChannel = comm.AWGNChannel('EbNo', obj.EbNo, ...
                'BitsPerSymbol', obj.BitsPerSymbol, ...
                'SignalPower', obj.SignalPower, ...
                'SamplesPerSymbol', obj.UpsamplingFactor);
         end
```

QPSKReceiver.m

```
classdef QPSKReceiver < matlab.System

% Copyright 2012-2015 The MathWorks, Inc.

    properties (Nontunable)
        DesiredAmplitude = 1/sqrt(2);
        ModulationOrder = 4;
        DownsamplingFactor = 2;
        CoarseCompFrequencyResolution = 50;
        PhaseRecoveryLoopBandwidth = 0.01;
        PhaseRecoveryDampingFactor = 1;
        TimingRecoveryDampingFactor = 1;
        TimingRecoveryLoopBandwidth = 0.01;
        TimingErrorDetectorGain = 5.4;
        PostFilterOversampling = 2;
        FrameSize = 100;
        BarkerLength = 13;
        MessageLength = 105;
        SampleRate = 200000;
        DataLength = 174;
        ReceiverFilterCoefficients = 1;
        DescramblerBase = 2;
        DescramblerPolynomial = [1 1 1 0 1];
        DescramblerInitialConditions = [0 0 0 0];
        PrintOption = false;
    end
```

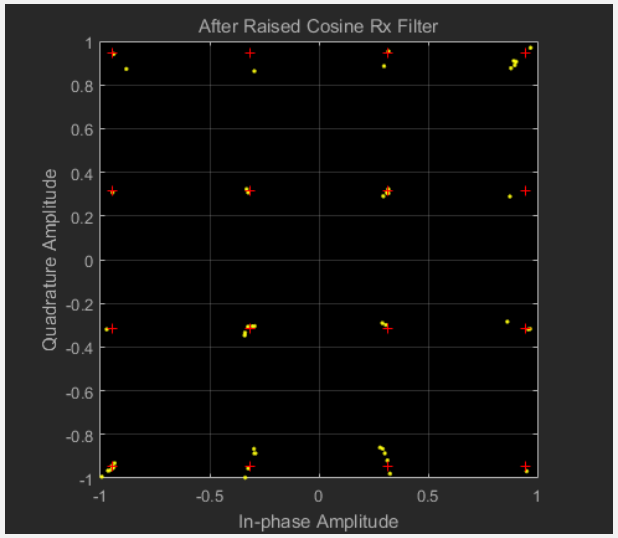
QPSKReceiver.m

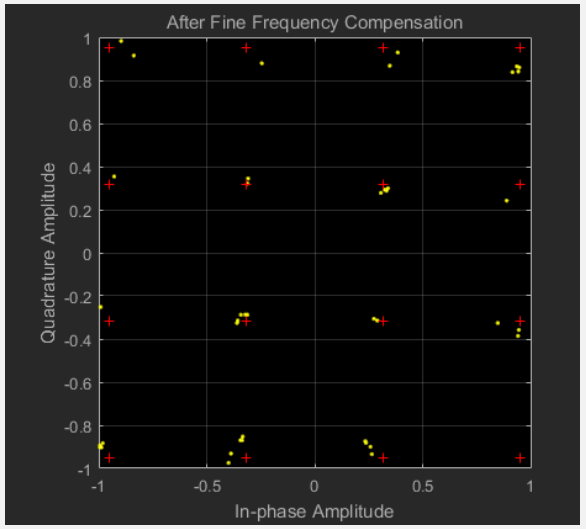
```
    properties (Access = private)
        pAGC
        pRxFilter
        pCoarseFreqEstimator
        pCoarseFreqCompensator
        pFineFreqCompensator
        pTimingRec
        pFrameSync
        pDataDecod
        pBER
     end

    properties (Access = private, Constant)
        pUpdatePeriod = 4 % Defines the size of vector that will be processe
        pBarkerCode = [+1; +1; +1; +1; +1; -1; -1; +1; +1; -1; +1; -1; +1];
        pModulatedHeader = sqrt(2)/2 * (-1-1i) * QPSKReceiver.pBarkerCode;
    end

    methods
        function obj = QPSKReceiver(varargin)
            setProperties(obj,nargin,varargin{:});
        end
    end
```

.....

QPSKTXRXSim.m
16QAM

PhaseOffset = 0,
EbNo = 40,
FrequencyOffset = 0;

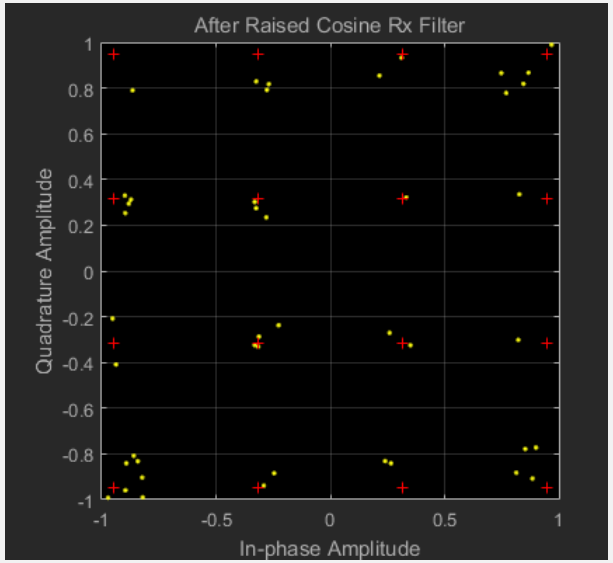


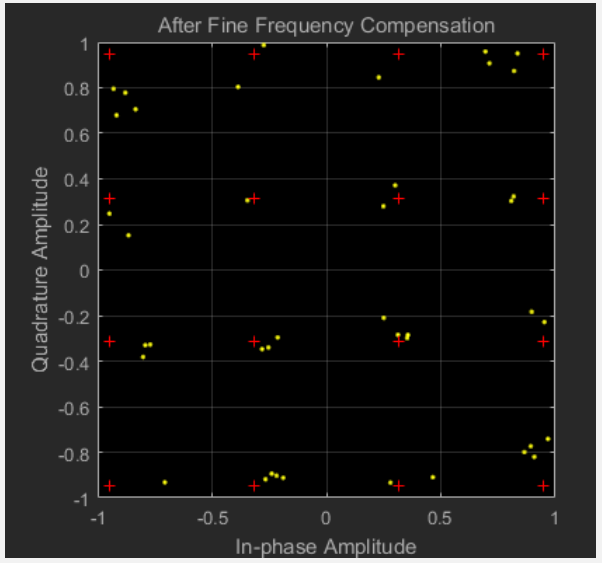


命令行窗口

```
  c  f`<Do^qu:  %1%
    r   R      W&J#m"q
Z% T`z:"i   L]KJ2
He  nJW'o}Dd=1003
Hello world 1004
Hello world 1005
Hello world 1006
Hello world 1007
Hello world 1008
Hello world 1009
Hello world 1010
Hello world 1011
```

PhaseOffset = 0,
EbNo = 20,
FrequencyOffset = 0;





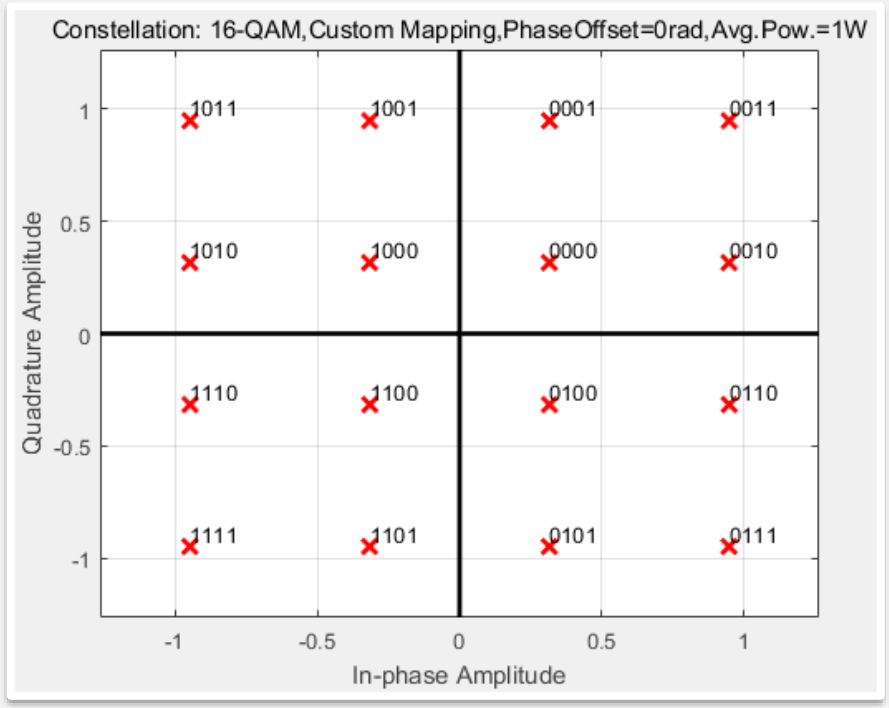
命令行窗口

```
T  /    G\L].0Wd&  5
 ello world 1077
F8U r"$<-?O      _J
mello world 1079
h    8b7pIm:C     %
t7llo world 1081
Hello world 1082
Hello world 1083
Hello world 1084
Hello world 1085
Hello world 1086
Hello world 1087
Hello world 1088
```

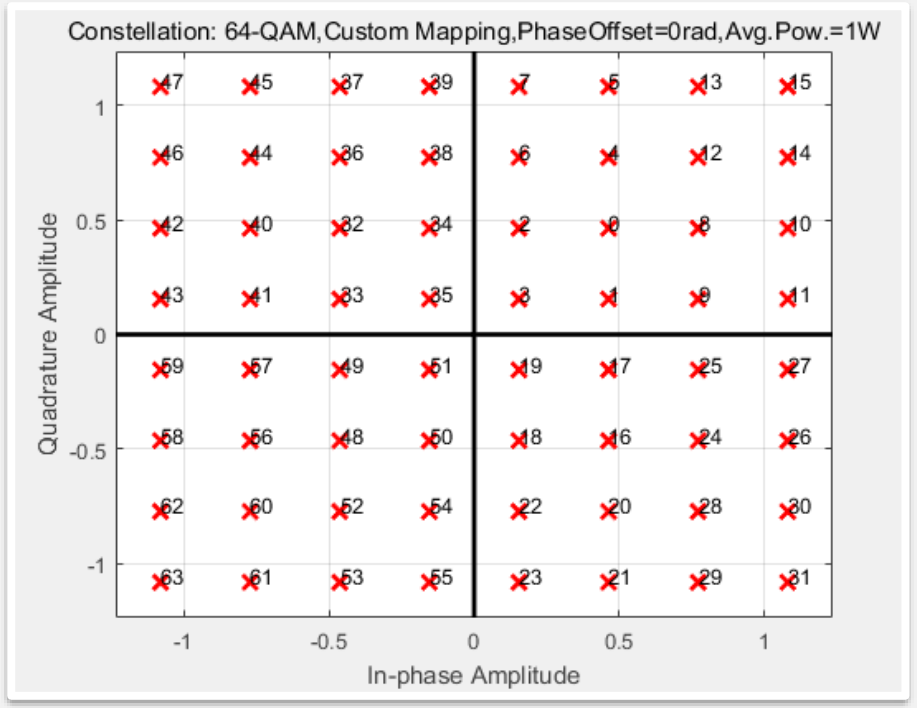
Compensation for the 16/64-QAM

Benefit from 16QAM and 64QAM

4 bits per symbol

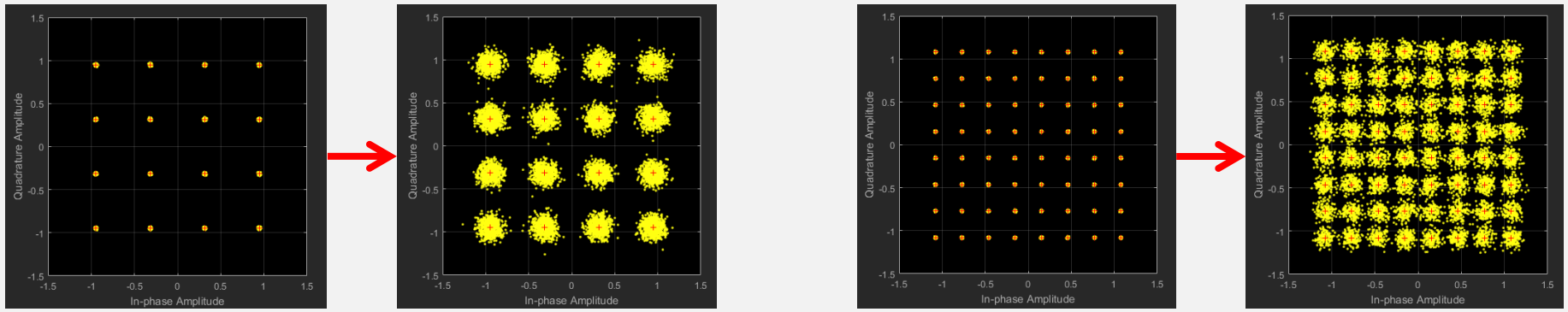


6 bits per symbol



Additive White Gaussian Noise

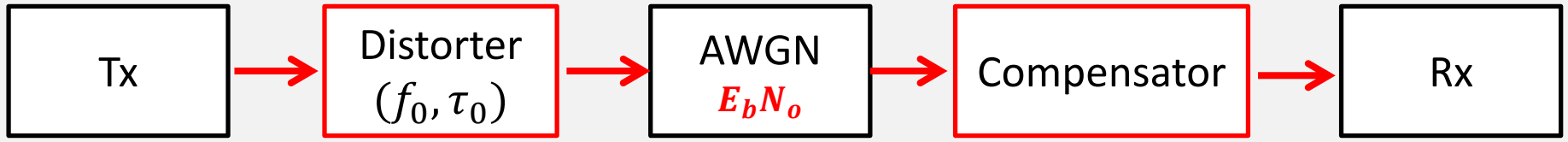
$$p_e \leq \frac{M-1}{2} e^{-\frac{d^2_{min}}{4}}$$

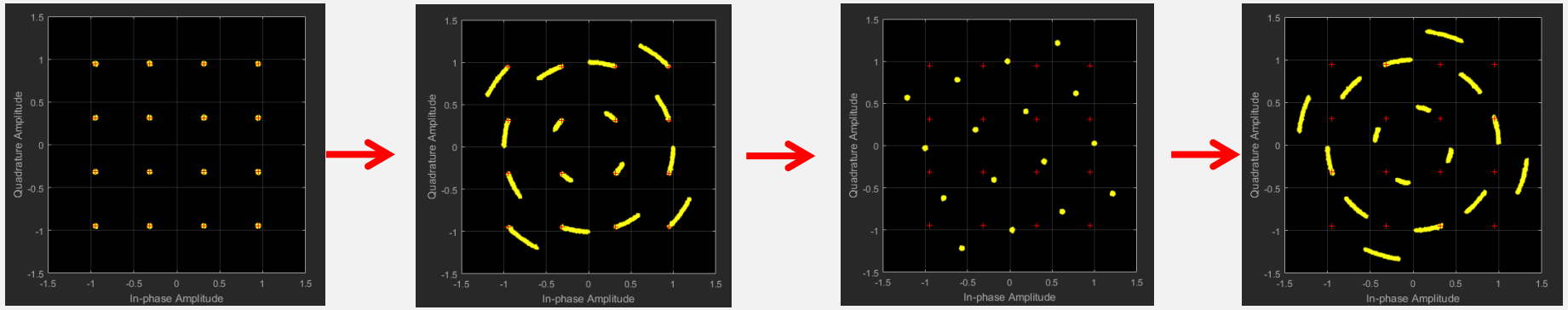


16QAM

64QAM

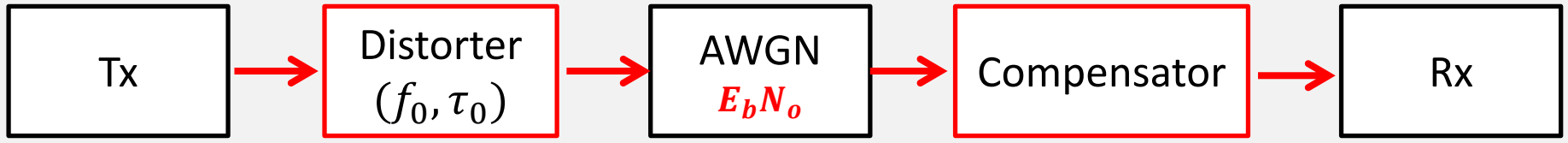
Frequency and phase offset

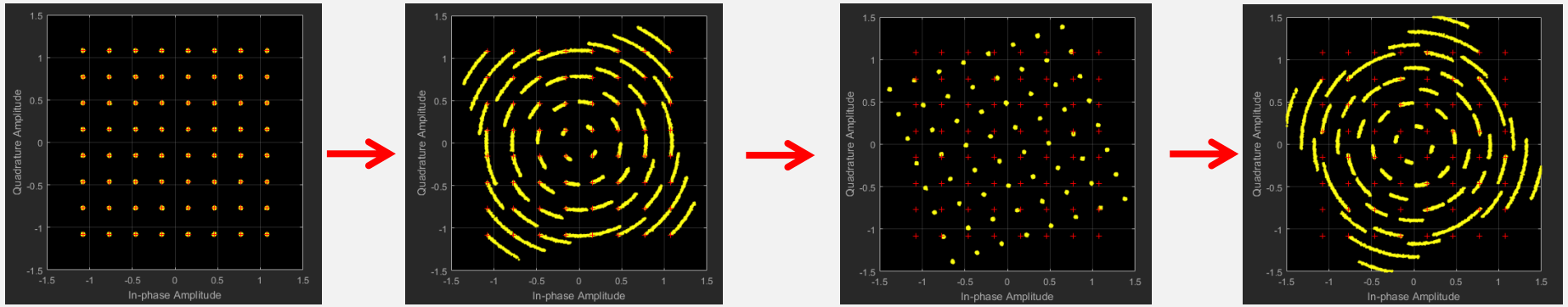




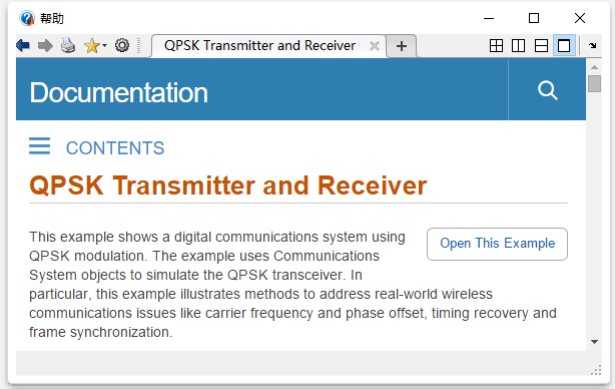
16QAM

Frequency and phase offset





64QAM



QPSKTransmitter

QPSKBitsGenerator

comm.QPSKModulator

dsp.FIRInterpolator

QPSKChannel

comm.PhaseFrequency Offset

dsp.VariableFractional Delay

comm.AWGNChannel

comm.CarrierSynchronizer

comm.SymbolSynchronizer

FrameFormation

QPSKReceiver

QPSKDataDecoder

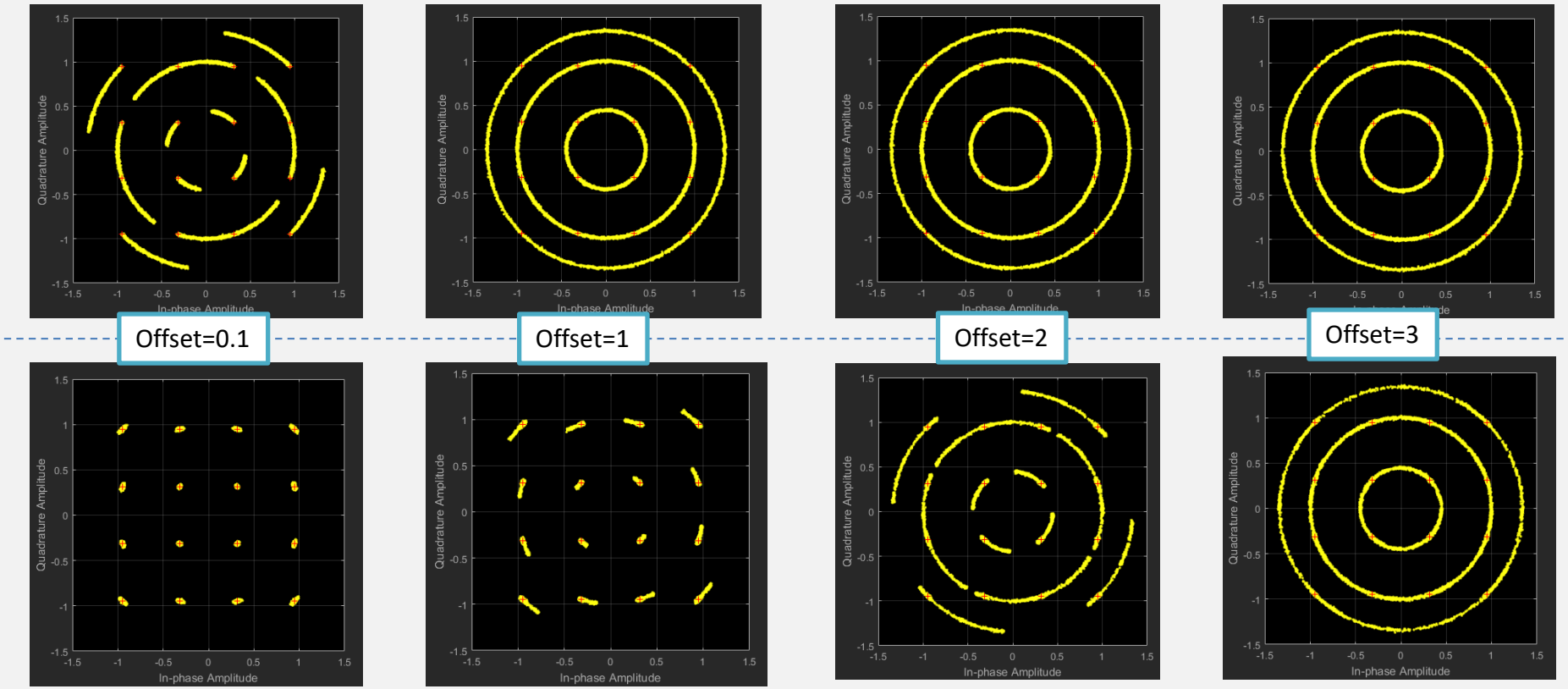
comm.AGC

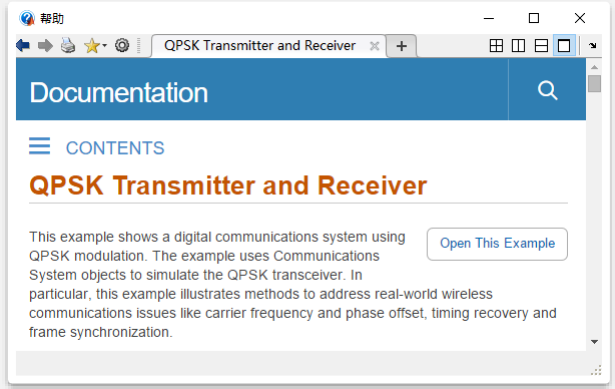
dsp.FIRDecimator

comm.PSKCoarseFrequ encyEstimator

comm.PhaseFrequencyOffset

Freq. compensation with comm.CarrierSynchronizer





QPSKTransmitter

QPSKBitsGenerator

comm.QPSKModulator

dsp.FIRInterpolator

QPSKChannel

comm.PhaseFrequency Offset

dsp.VariableFractional Delay

comm.AWGNChannel

comm.CarrierSynchronizer

comm.SymbolSynchronizer

FrameFormation

QPSKReceiver

QPSKDataDecoder

comm.AGC

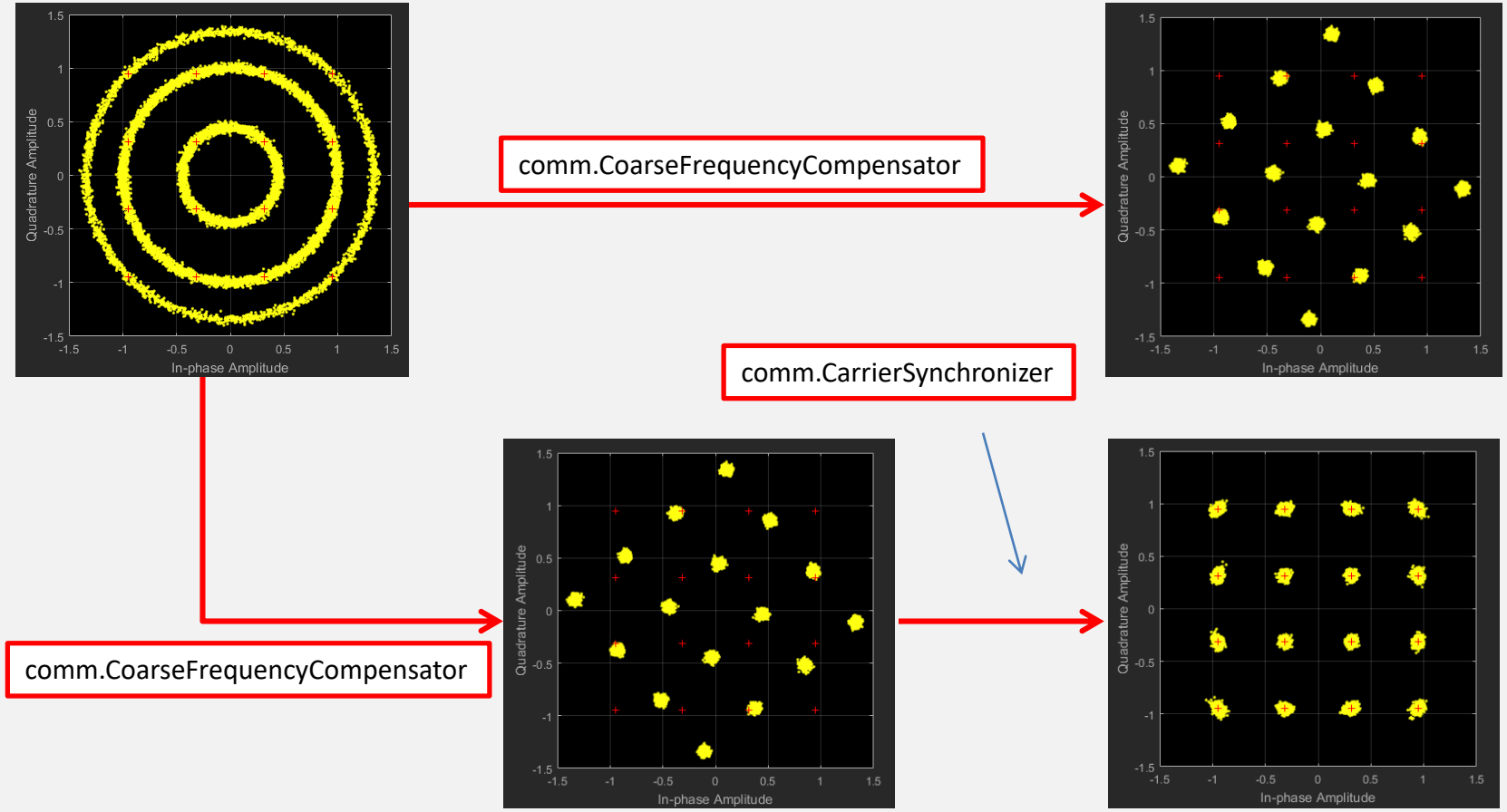
dsp.FIRDecimator

comm.PSKCoarseFrequ encyEstimator

comm.PhaseFrequencyOffset

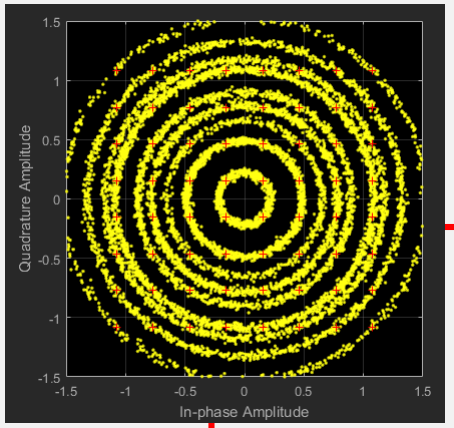
Freq. compensation with comm.CoarseFrequencyCompensator

16QAM

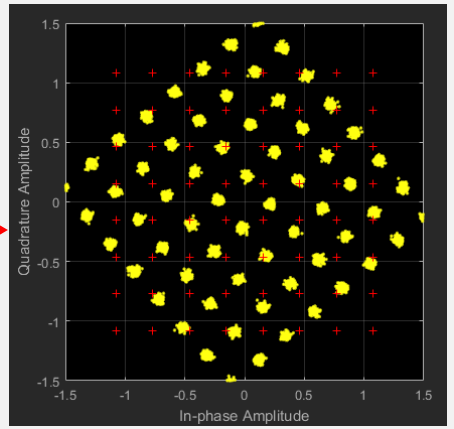


Freq. compensation with comm.CoarseFrequencyCompensator

64QAM

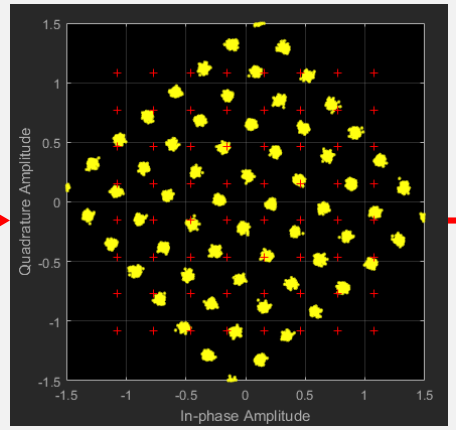


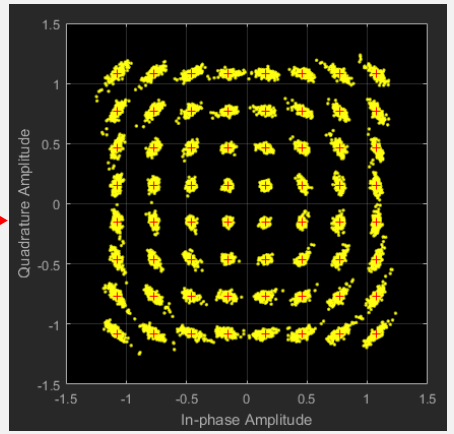
comm.CoarseFrequencyCompensator

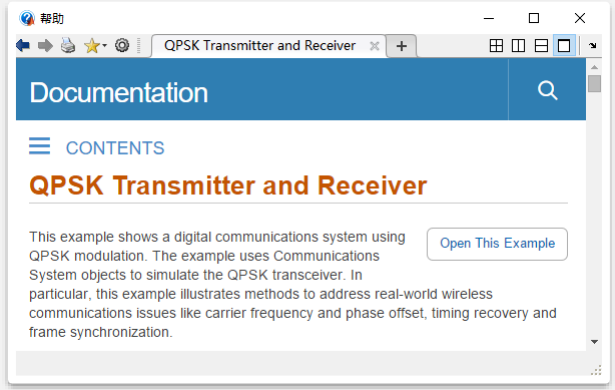


comm.CarrierSynchronizer

comm.CoarseFrequencyCompensator







QPSKTransmitter

QPSKBitsGenerator

comm.QPSKModulator

dsp.FIRInterpolator

QPSKChannel

comm.PhaseFrequency Offset

dsp.VariableFractional Delay

comm.AWGNChannel

comm.CarrierSynchronizer

comm.SymbolSynchronizer

FrameFormation

QPSKReceiver

QPSKDataDecoder

comm.AGC

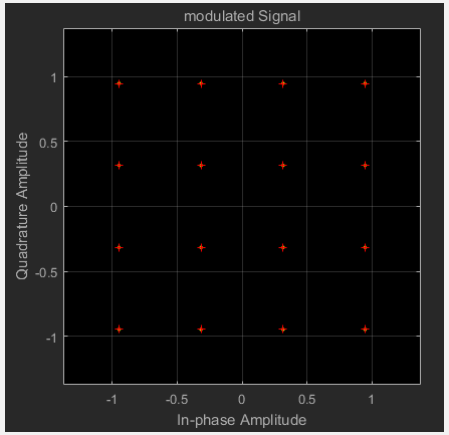
dsp.FIRDecimator

comm.PSKCoarseFrequ encyEstimator

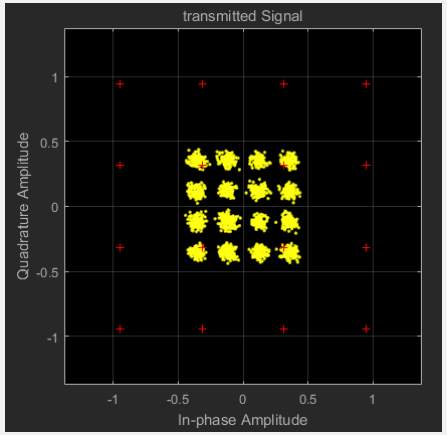
comm.PhaseFrequencyOffset

Symbol timing with comm.SymbolSynchronizer

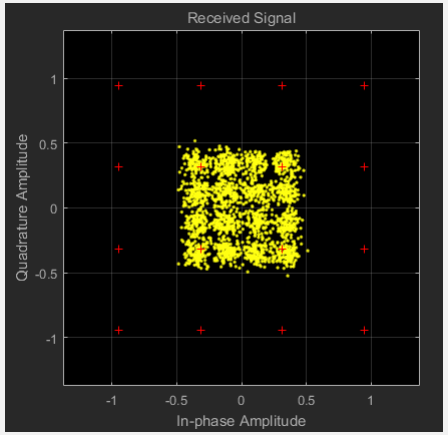
Modulated Signal



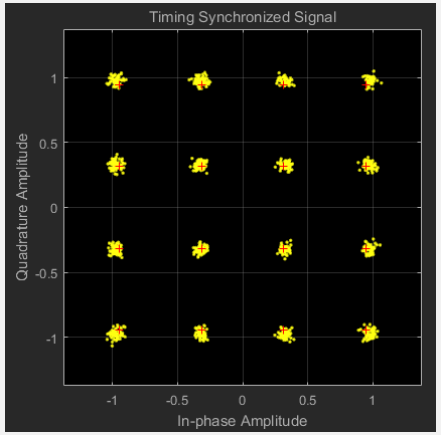
Transmitted Signal



Received Signal

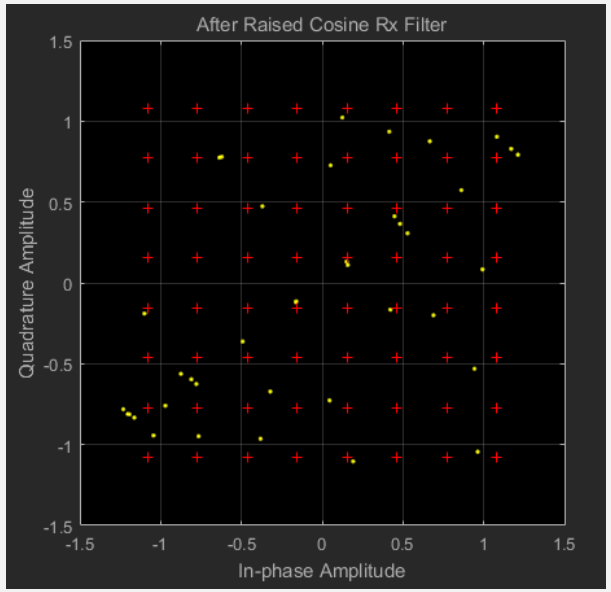


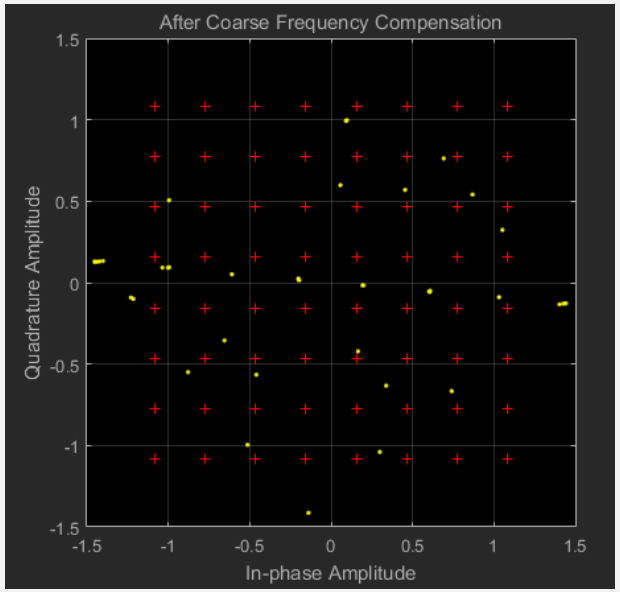
Timing Synchronized Signal

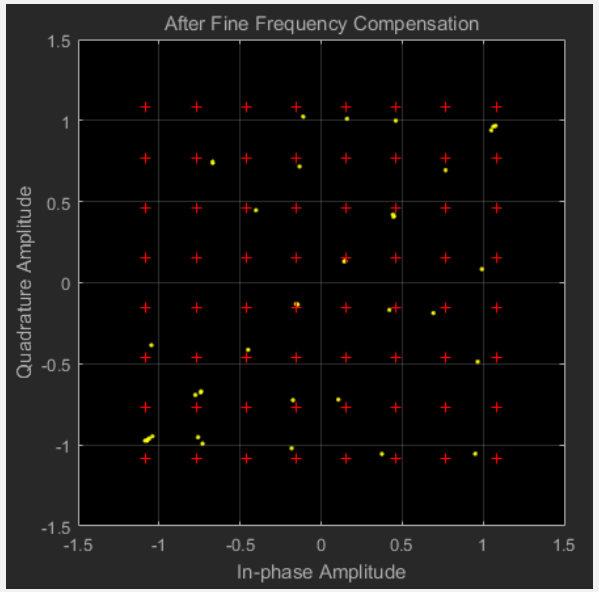


Compensation steps

comm.PSKCoarseFrequency Estimator → comm.CarrierSynchronizer → comm.SymbolSynchronizer → FrameFormation →

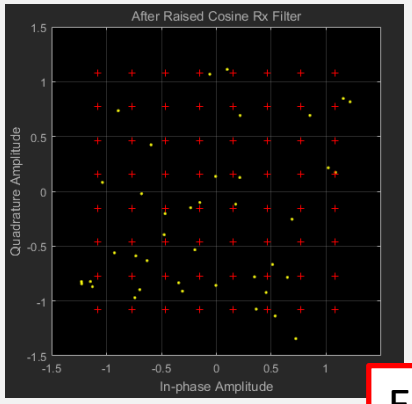


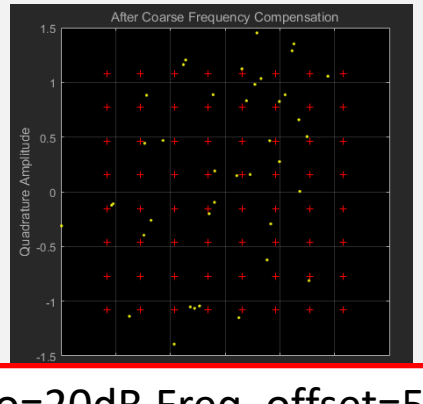


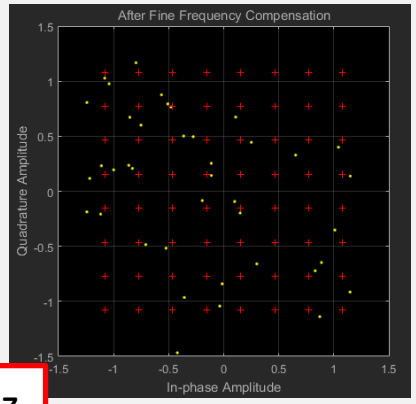


EbNo=50dB， Freq. offset=50Hz

Comparison of 64-QAM and 16-QAM



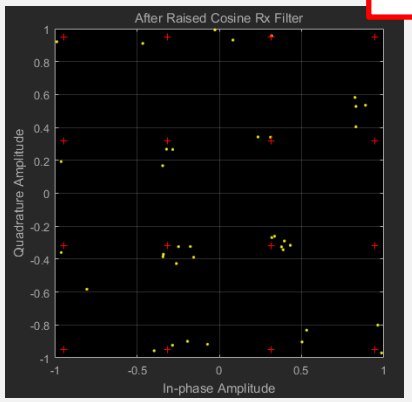


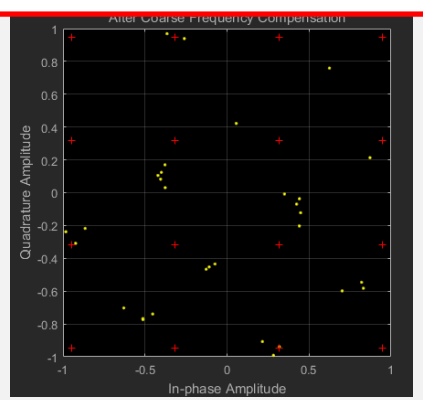


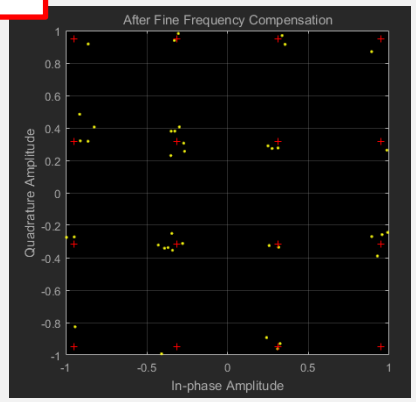
命令行窗口

```
  5cY&w#;%M79/s    -
) 6-HD    fU@    "6PO<
JelqH%  [8aC*  J    )O
  %   uw\R  R  7iaB|Dc
(b8X(~5'   7 t9  ?Xli
/xLsJ  .=tx
p`b v=
v<  8L>i9
c&
z=h
Hk"
}l    {-o  $H  #B*
```

EbNo=20dB,Freq. offset=50Hz





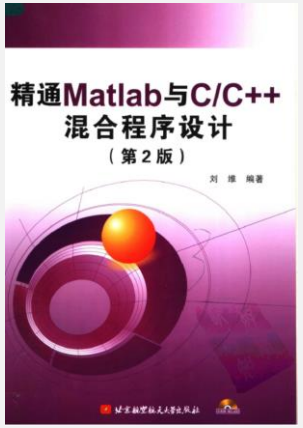


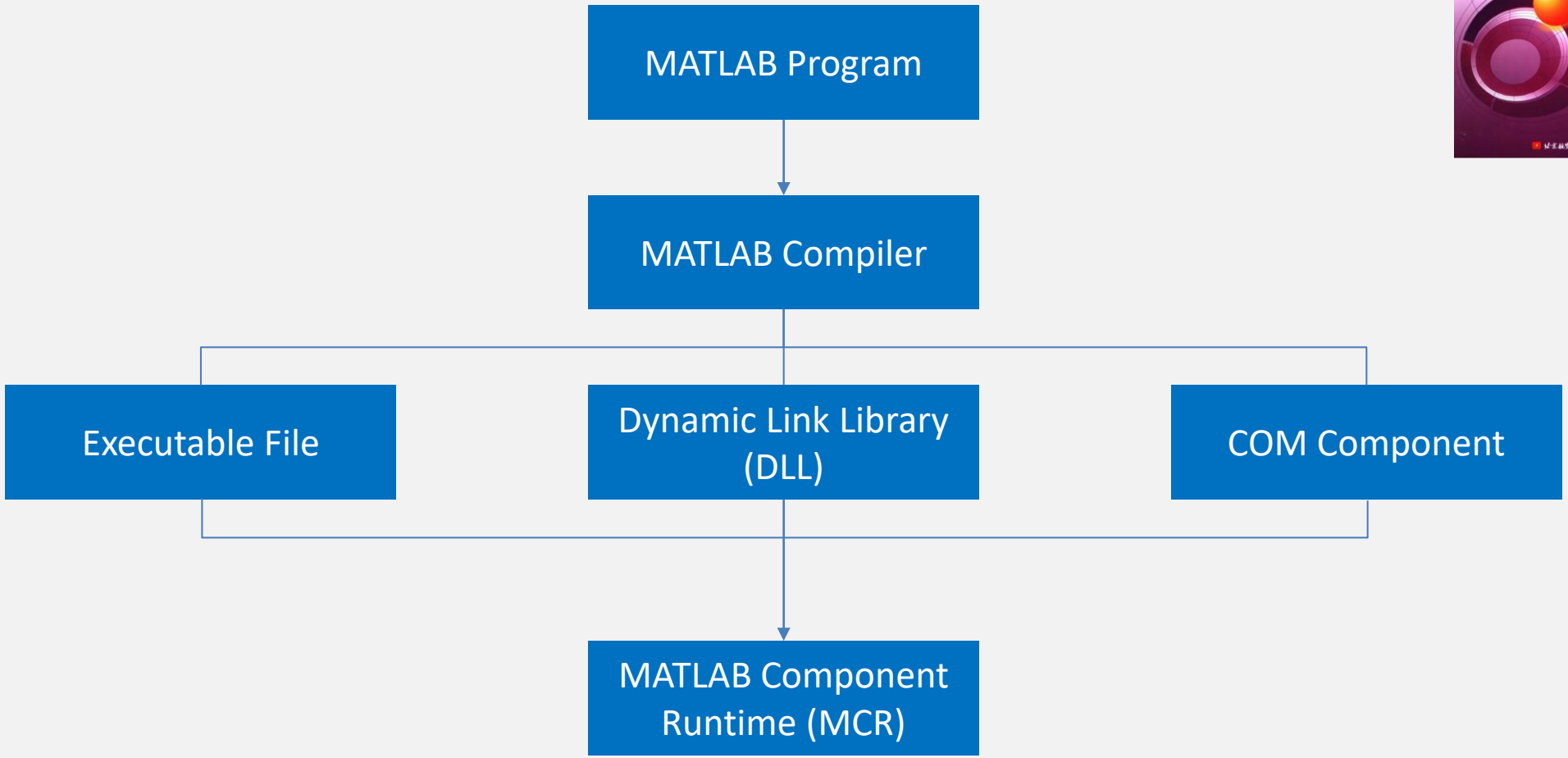
命令行窗口

```
  2|\n|@  ^qr  R%1%
[  r<!'i  O  &J--"u
}bE+kP9n    d!^c  |
Hk/     world EDD.
Hello world 1004
Hello world 1005
Hello world 1006
Hello L  rld 1007
Helo? world 1008
Hello world 1009
```

Compilation and acceleration

MATLAB Compiler Architecture





MEX File Generation

编辑器 - C:\Program Files\MATLAB\R2016a\toolbox\comm\commdemos\commQPSKTransmitterReceiver.m

commQPSKTransmitterReceiver.m

```
46      %% Initialization
47      % The <matlab:edit('commqpsktxrx_init.m') commqpsktxrx_init.m> script
48      % initializes simulation parameters and generates the structure prmQPSKTxRx.
49 —    prmQPSKTxRx = commqpsktxrx_init; % QPSK system parameters
50
51 —    useScopes = true; % true if scopes are to be used
52 —    printReceivedData = false; %true if the received data is to be printed
53 —    compileIt = true; % true if code is to be compiled
54 —    useCodegen = true; % true to run the generated mex file
```

Code Generation Report

MATLAB code | Call stack | C code

Target Source Files

- AGC.c
- AGC.h
- AWGNChannel.c
- AWGNChannel.h
- CarrierSynchronizer.c
- CarrierSynchronizer.h
- CoarseFrequencyEstimatorBase.c
- CoarseFrequencyEstimatorBase.h
- FrameFormation.c
- FrameFormation.h
- PSKCoarseFrequencyEstimator.c
- PSKCoarseFrequencyEstimator.h
- QPSKBitsGenerator.c
- QPSKBitsGenerator.h
- QPSKChannel.c
- QPSKChannel.h
- QPSKDataDecoder.c
- QPSKDataDecoder.h
- QPSKReceiver.c
- QPSKReceiver.h
- **QPSKTransmitter.c**
- QPSKTransmitter.h
- SymbolSynchronizer.c
- SymbolSynchronizer.h
- SystemCore.c
- SystemCore.h
- VariableFractionalDelay.c
- VariableFractionalDelay.h
- abs1.c
- abs1.h
- de2bi.c
- de2bi.h
- eml_int_forloop_overflow_check.c
- eml_int_forloop_overflow_check.h
- error.c

File: **QPSKTransmitter.c** (QPSKTransmitter.c)

```
1   /*
2    * QPSKTransmitter.c
3    *
4    * Code generation for function 'QPSKTransmitter'
5    *
6    */
7
8   /* Include files */
9   #include "rt_nonfinite.h"
10  #include "runQPSKSystemUnderTest.h"
11  #include "QPSKTransmitter.h"
12  #include "SystemCore.h"
13
14  /* Variable Definitions */
15  static emlrtRSInfo x_emlrtRSI = { 45, "QPSKTransmitter",
16    "C:\\Program Files\\MATLAB\\R2016a\\toolbox\\comm\\commdemos\\QPSKTransmitter.m"
17  };
18
19  static emlrtRSInfo y_emlrtRSI = { 48, "QPSKTransmitter",
20    "C:\\Program Files\\MATLAB\\R2016a\\toolbox\\comm\\commdemos\\QPSKTransmitter.m"
21  };
22
23  static emlrtRSInfo ab_emlrtRSI = { 51, "QPSKTransmitter",
24    "C:\\Program Files\\MATLAB\\R2016a\\toolbox\\comm\\commdemos\\QPSKTransmitter.m"
25  };
26
```

Summary | All Messages (0) | Target Build Log

C source code generated on: 11-Mar-2020 13:21:42

Coding target: C MEX Function

Number of errors: 0

Number of warnings: 0

Number of notices: 0

Tell Us What You Think

We value your feedback. Please take a few minutes to answer this short questionnaire regarding the Code Generation Report.

>>Provide Feedback

Check your C-compiler

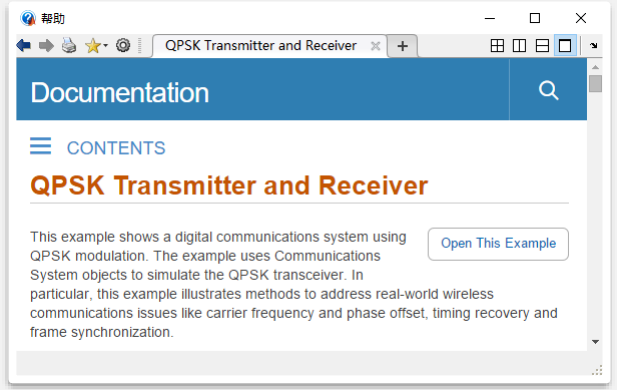
命令行窗口

```
>> mex -setup
MEX 配置为使用 'Microsoft Windows SDK 7.1 (C)' 以进行 C 语言编译。
警告: MATLAB C 和 Fortran API 已更改，现可支持
    包含 2^32-1 个以上元素的 MATLAB 变量。不久以后，
     您需要更新代码以利用
     新的 API。您可以在以下网址找到相关详细信息:
     http://www.mathworks.com/help/matlab/matlab_external/upgrading-mex-files-to-use-64-bit-api.html。

要选择不同的 C 编译器，请从以下选项中选择一种命令:
MinGW64 Compiler (C)  mex -setup:'C:\Program Files\MATLAB\R2016a\bin\win64\mexopts\mingw64.xml' C
Microsoft Windows SDK 7.1 (C)  mex -setup:C:\Users\0TA\AppData\Roaming\MathWorks\MATLAB\R2016a\mex_C_win64.xml C

要选择不同的语言，请从以下选项中选择一种命令:
 mex -setup C++
 mex -setup FORTRAN
```

MATLAB Product Family – Release 2016a									
Compiler	MATLAB	MATLAB Compiler	MATLAB Compiler SDK				MATLAB Coder	SimBiology	Fixed Point Designer
	For MEX-file compilation, `loadlibrary`, and external usage of MATLAB Engine and MAT-file APIs	*Excel add-in for desktop*	*C/C++ & COM*	*.NET*	*Java*	*Excel add-in for MPS*	*For all features*	*For accelerated computation*	*For accelerated computation*
MinGW 4.9.2 C/C++ (Distributor: TDM-GCC) Available at no charge	✓						✓ 6	✓	✓
Microsoft Visual C++ 2015 Professional	✓	✓	✓	✓ 4			✓	✓	✓
Microsoft Visual C++ 2013 Professional	✓	✓	✓	✓ 4			✓	✓	✓
Microsoft Visual C++ 2012 Professional	✓	✓	✓	✓ 4			✓	✓	✓
Microsoft Visual C++ 2010 Professional SP1	✓	✓	✓	✓ 4			✓	✓	✓
Microsoft Windows SDK 7.1 Available at no charge; requires .NET Framework 4.0	✓	✓	✓				✓ 6	✓	✓

Assignments

- Read the example '**QPSK Transmitter and Receiver**' in Communications System Toolbox.
- Explain the functions of the following six subcomponents respectively,
  - (1) Automatic Gain Control
  - (2) Coarse frequency compensation
  - (3) Fine frequency compensation
  - (4) Timing recovery
  - (5) Frame Synchronization
  - (6) Data decoder



- Implement '**16-QAM** Transmitter and Receiver' according to the example.
- Compare the BER between QPSK and 16-QAM under different EbN0 condition.

- Question ?